Workflows

1 April 2026

DISGUISE

Contents

Workflows

3D Modelling

3D Fundamentals Overview 1
Understanding .obj files 2
Exporting 3D models 3
Optimising 3D Models 10

Augmented Reality

AR 13
AR Spatial Calibration 14

Calibration Projection

Projector Calibration 19
Manual Projector Calibration 20
MultiPose Projector Calibration 27
Calibration Errors 29
Quick Calibration 32
QuickCal Projectors 37
Creating & deleting reference points in QuickCal 47
OmniCal Overview 50
OmniCal Camera Setup 62
OmniCal Capture 89
OmniCal Calibration 102
OmniCal Alignment 109
OmniCal Multi-Pose Alignment 118
OmniCal Mesh Deform 126
OmniCal RigCheck 131

EVO

EVO 138
MA3D 143
Capture 147
Depence 149

OSC

OSC Workflows 152
OSC Sockpuppet 153

Sockpuppet

Sockpuppet 159
Patch Assignments 161
Bank Editor 170
Sockpuppet Personality Editor 176
Sequencing with Sockpuppet 185

HTTP

HTTP Sockpuppet 188

Contents

Motion Capture

Motion Capture Overview 190

Vicon Emulator 191

Skinned Mesh Skeleton

Skeleton 196

Skeletal Point Tracking Source 202

ISAAC Bridge

ISAAC Bridge 206

3DGS

Volinga

Volinga 3DGS 209

Virtual Production

Virtual Production Overview 212

2.5D workflow 215

Cuebric Disguise Integration 222

Mesh Mapping Workflow 225

Shot Recorder Setup 228

Depth Workflows 232

Anamorphic Lenses 236

ARRI Skypanels Setup 238

Green Screen Workflow 244

Using Sony’s Moiré Alert with Designer 248

UV Mapping

UV mapping documentation 255

What is UV mapping? 256

How does Disguise Designer sample UV maps? 258

Editing UV maps 261

The UV map as the Hardware Output 264

The UV map as the content template 267

LED Screen Examples

LED Screen Examples Overview 272

Example 1 - MiSTRIP star 273

Example 2 - OLite column 278

Example 3 - DNA Spiral 282

Example 4-Millenia Tower 285

Projection Examples

Projection Examples Overview 288

Projection Examples- Cylindrical mapping 289

Projection Examples - Pelt mapping 295

Projection Examples - Perspective mapping 298

Projection Examples - Pixel-perfect mapping 303

Contents

Projection Examples - Planar mapping 308
Projection Examples - Shrink wrapping 311
Projection Examples - Spline mapping 315

Virtual Reality

Virtual Reality 319

Extended Reality

xR Workflow Overview 323
xR Project Setup 324
xR Stage Setup 325
MR Set 327
Video Receive Delay 332
Observations 335
Spatial Calibration 340
Spatial Tracker Delay 351

Colour Calibration

Colour calibration 354
Inverse LUT Method 360
API xR Calibration 363
Set Extension 366
Virtual Cameras 368
Camera Switching 374
Inner-Outer Frustum Workflow 377
Multi Frame LED Processor Mapping 380
Depth of Field Workflow 385
xR for VR Workflow 390

3D Fundamentals Overview

This sub-chapter will teach you the **fundamental** knowledge necessary to **prepare** 3D models for Designer.



The London Ralph Lauren Flagship Store in Designer.

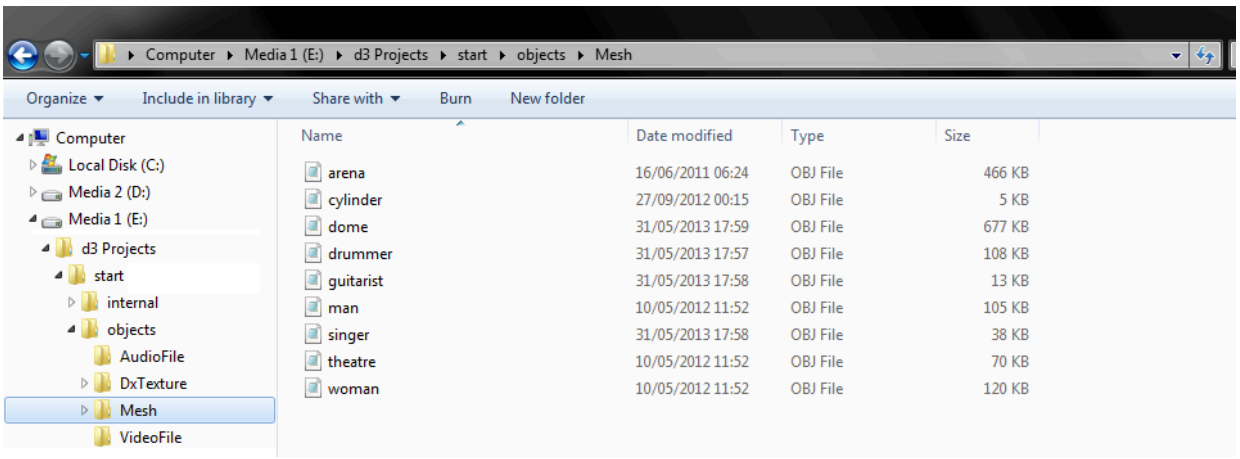
This section includes:

- Understanding the .obj files
- Exporting 3D models
- Optimising 3D models

Understanding .obj files

An **.obj** file is a format containing data for an object's vertices, polygons and texture coordinates.

The image below shows a Disguise project containing common **.obj** files which can be separated into two categories: **Screens** and **Props**. A Screen is an object representing a **video display**, for example a projection dome, and requires a **UV map** to display video content. A Prop is an object representing a **scenographic element**, for example a theatre venue, and does not always require a UV map. To read more about where to place media, see the **Placing media files for a project** page. To read more about UV mapping, see the **What is UV mapping** page.



Common .obj files contained in the Disguise Start project.

Helpful Resources

Wavefront OBJ File Format Summary

Exporting 3D models

Designer supports the .obj format for 3D meshes. This topic covers the .obj export settings unique to Blender, 3ds Max, Maya, and Cinema 4D, which should be followed to ensure the object is exported to Designer correctly.

Custom 3D models can be loaded into your project and utilized for previsualisation. These models be used as either display surfaces or as props. The supported format for a 3D model is .obj.

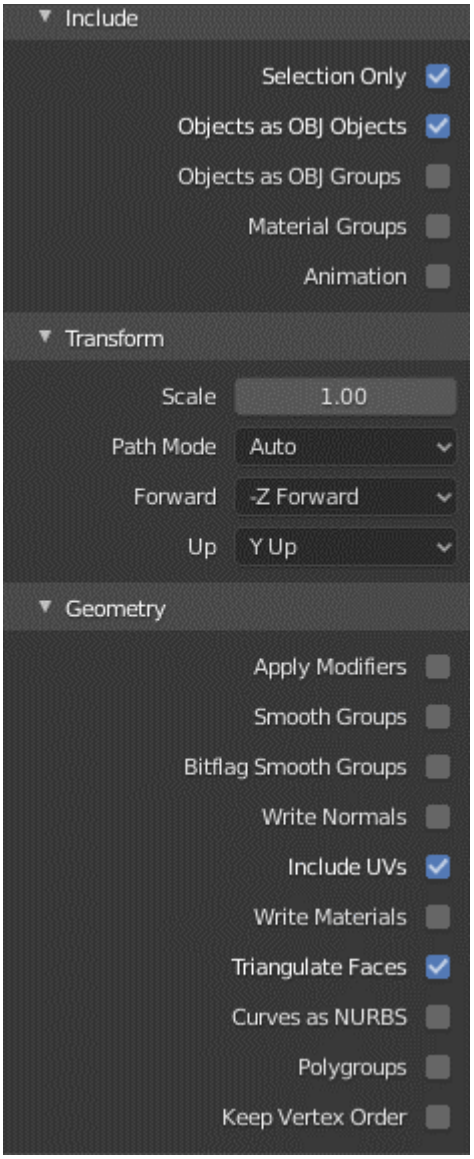
Custom 3D objects can either have a baked in UV unwrap, or not. Objects with a UV unwrap can be used as screen objects to display content, while non-unwrapped ones can only be used as Prop objects used for the visualiser.

Outlined in these tabs are the needed export settings for a custom OBJ to be used in Disguise.

> **Note**
>
> Other types of 3D objects known as Alembic files can also be imported into Designer; these types of objects included animation that can be triggered from within Designer. For more information on these types of files, please visit this link: **Alembic files**.

Blender Export Settings

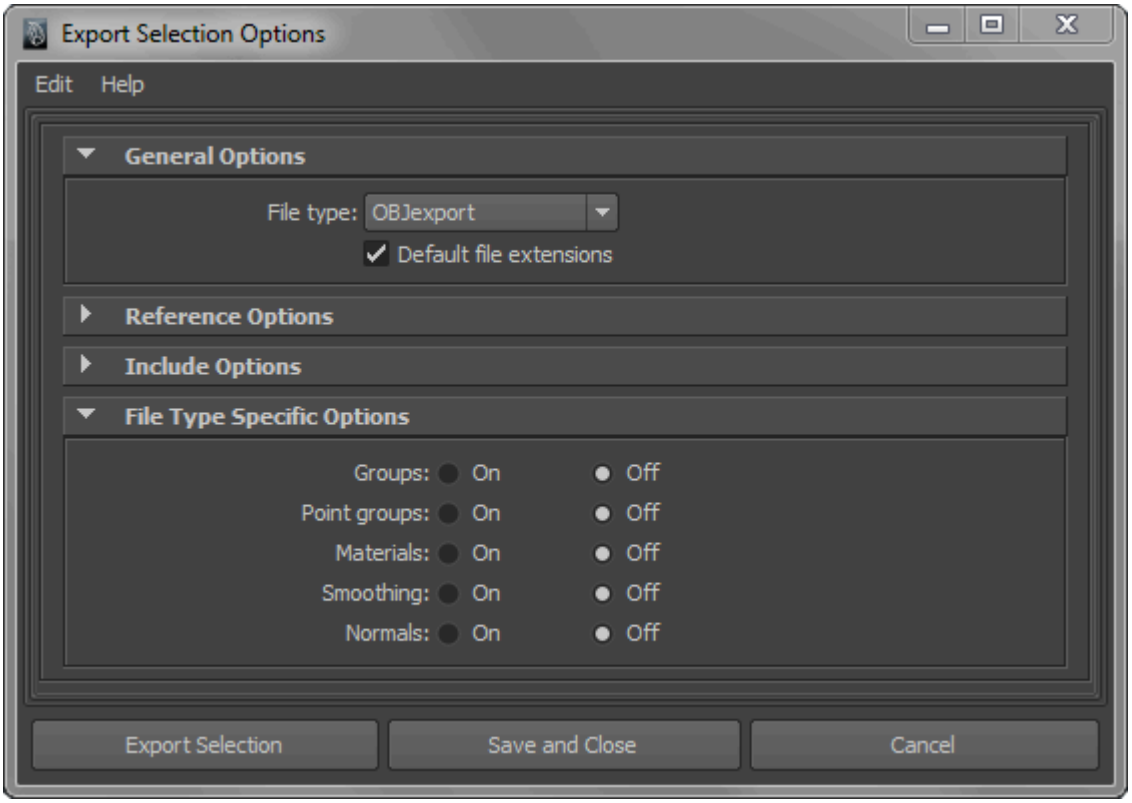
1. Select the object to be exported from the 3D application. Note that the selection can not contain any scene elements such as lights or cameras.
2. Select the 3D application's export utility.
3. Select the .obj format from the list of available formats.



Export settings for Blender

Maya Export Settings

1. Select the object to be exported from the 3D application. Note that the selection can not contain any scene elements such as lights or cameras.
2. Select the 3D application's export utility.
3. Select the .obj format from the list of available formats.

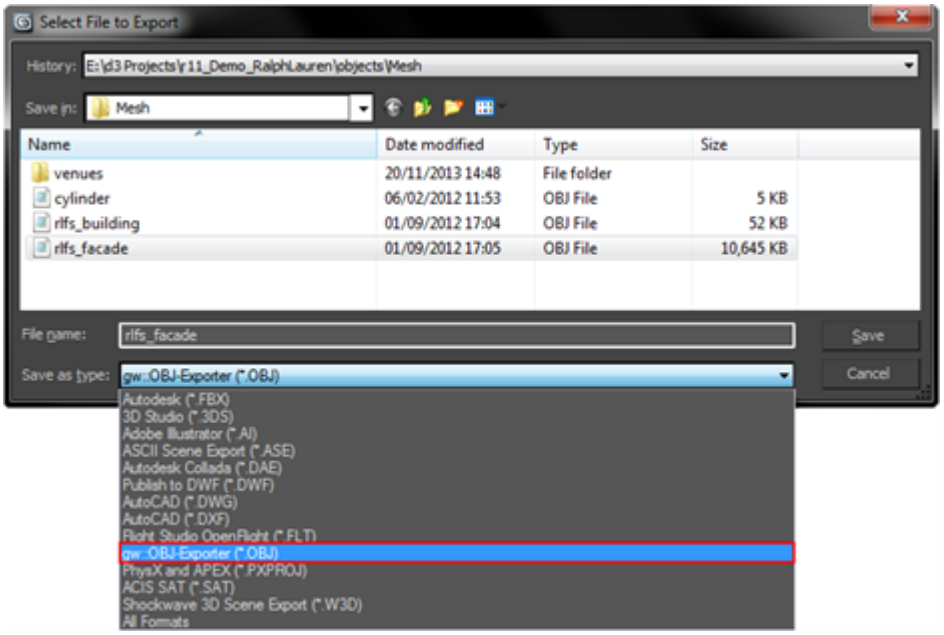


Export settings for Maya

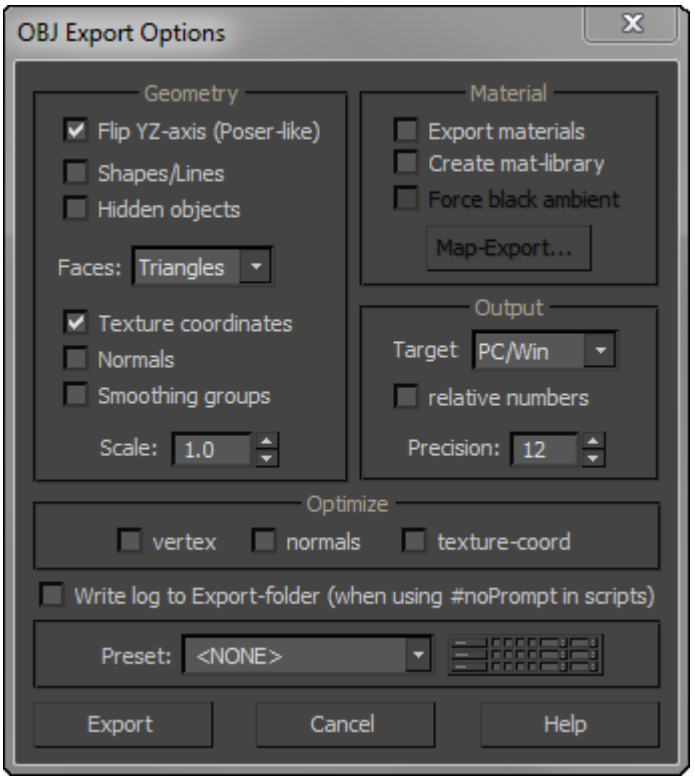
3ds Export Settings

1. Select the object to be exported from the 3D application. Note that the selection can not contain any scene elements such as lights or cameras.
2. Select the 3D application's export utility.

3. Select the .obj format from the list of available formats.



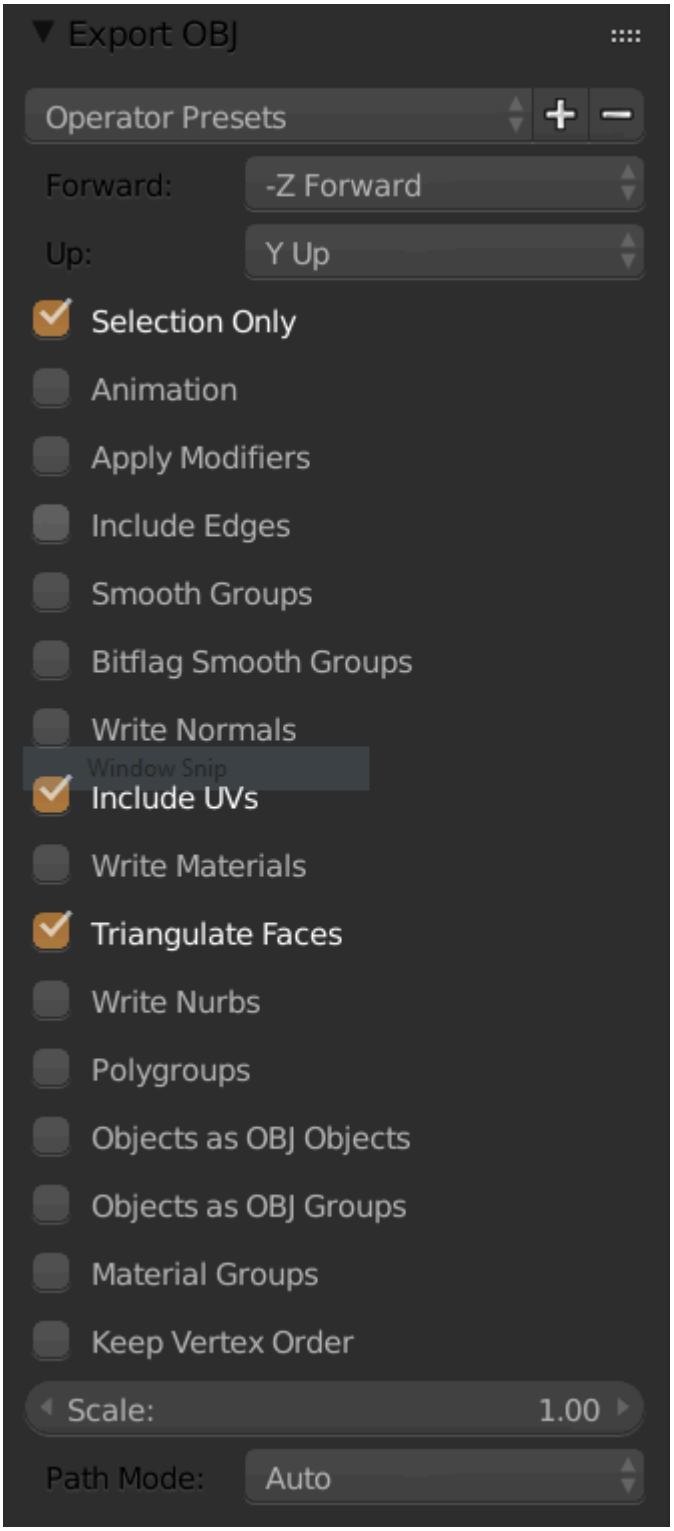
- Deselect shapes, lines, hidden objects, normals, smoothing groups, material, and material libraries. Disguise does not support these.
- If the object is to become a screen or a textured prop, select texture coordinates, which will export the object's UV map.
- Select flip yz-axis to ensure the object is orientated correctly in Designer.



Export settings for 3ds Max

Save the .obj file in the Disguise Project folder in the following directory: Projects > Project Name > objects > Mesh.

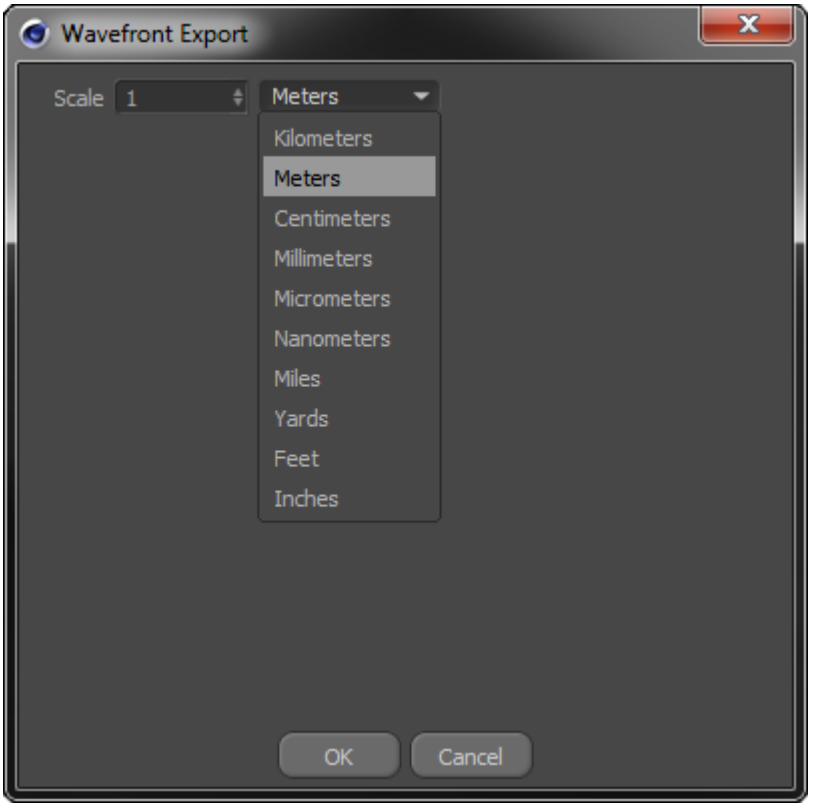
To read more about where to place media, see the **Placing media files for a project** page.



A list of .obj files exported from 3ds Max to a Disguise Project folder. **Please note:** We recommend triangulating faces before exporting models for use in Designer.

C4D Export Settings

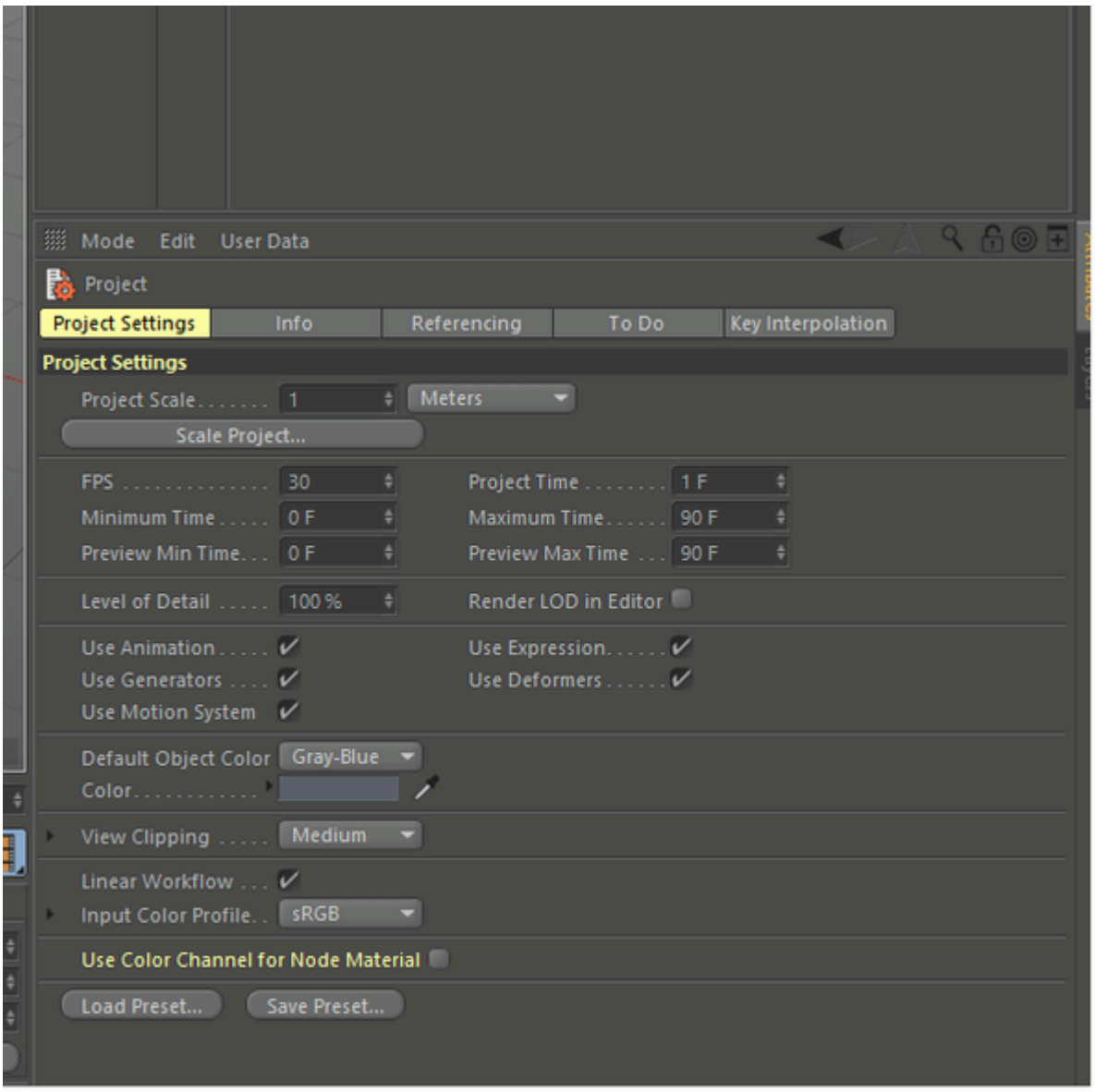
1. Check the project settings are set to scale in **meters** (C4D defaults to cm).
2. Select the object to be exported from the 3D application. Note that the selection can not contain any scene elements such as lights or cameras.
3. Select the 3D application's export utility.
4. Select the .obj format from the list of available formats.



Export settings for C4D

1. Check the project settings are set to scale in **meters** (C4D defaults to cm).
2. Select the object to be exported from the 3D application. Note that the selection can not contain any scene elements such as lights or cameras.
3. Select the 3D application's export utility.

4. Select the .obj format from the list of available formats.



Optimising 3D Models

When preparing objects for Designer there are a number of processes to consider.

- Objects should be divided into Screens and Props.
- Screens should only include the surfaces necessary to display video content and may require re-modelling by detaching or deleting unnecessary polygons.
- Detached objects can be used in Designer as props, which are helpful for simulating a show in the **Stage visualiser** to present to the client.
- Objects should not exceed 150,000-200,000 vertices and may require remodelling by welding unnecessary vertices.

Dividing 3D models into screens and props

In a 3D application, an object can be detached into multiple sub-objects. This is necessary when dividing a 3D model into screens and props for Designer. Screens are video displays such as LED panels, projection surfaces and DMX fixtures, and props are scenographic elements such as audience seating, stage floors and truss pieces.

Screens

A new Designer project contains the following .obj files to be used as screens:

- rectangle.obj
- cylinder.obj
- concave.obj
- dome.obj

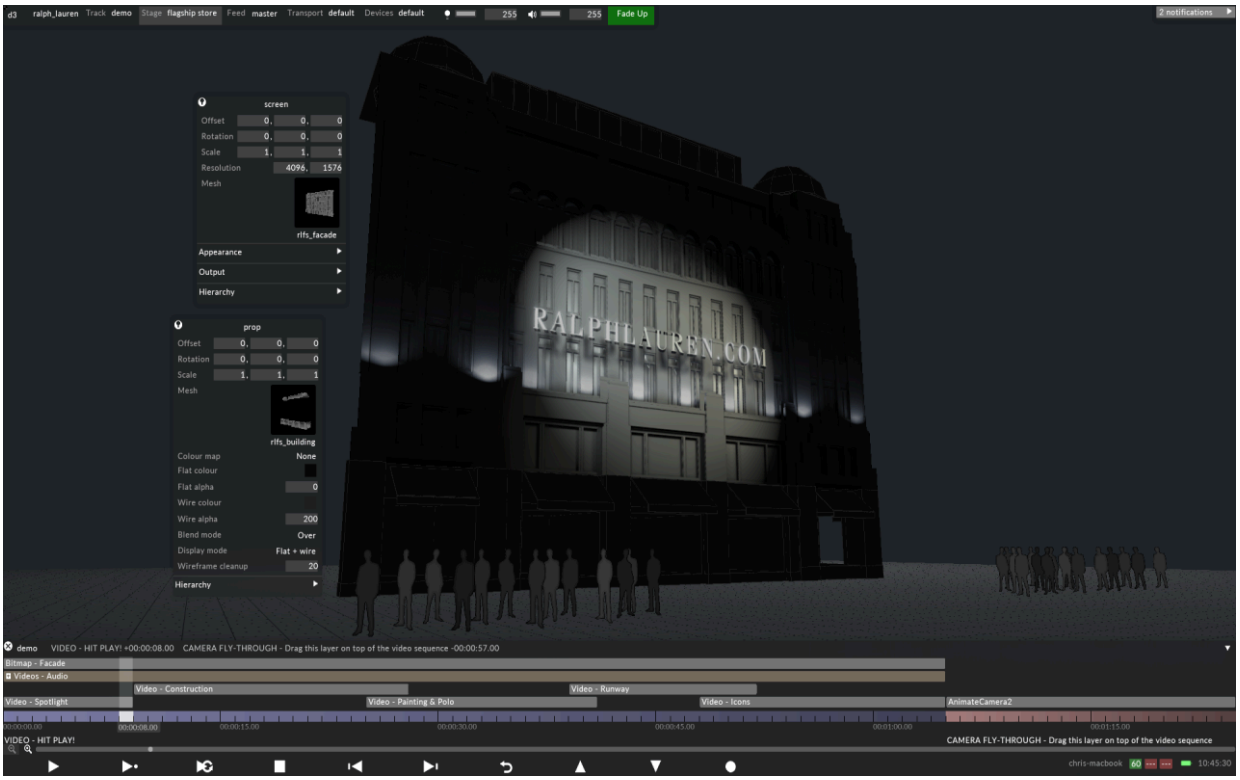
> **Note**
>
> The rectangle.obj, concave.obj and puffersphere.obj file is generated in the software and is not visible in the mesh library.

Complex screens, for example a building or a car, require 3D modelling and UV mapping tools available in 3D applications. For further information see the LED screen examples and the Projection screen examples.

Props

Parts of an object which are not required to display video content can be used in Designer as props. The image below shows the Ralph Lauren London Flagship Store divided into two sub-

objects: a screen to display video content and a prop to simulate the building's top and bottom. In Designer a prop can be customised to simulate the object more realistically including the ability to map paint textures to it. For further information see the **Editing Props** page.



The Ralph Lauren Flagship Store divided into a screen and a prop in Disguise.

Deleting overlapping faces

Delete all overlapping faces from the object. It is important to understand when video content is mapped to the screen in Designer, any surfaces with overlapping faces will display overlapping content. To test this, map a **Radar** layer to the screen.

Reducing the number of vertices

In the Designer, the Stage should not exceed 150,000-200,000 vertices. It is recommended to reduce the total number of vertices by welding them together, which should be done in a 3D application. Be careful not to delete too many vertices because this may lower playback quality, but on the other hand too many vertices may lower playback performance. A balance between quality and performance is necessary.

Please note: during a live show it is recommended to remove props from the Stage in order to optimise playback performance. For further information see the **Creating and removing props** page.

Helpful resources

3ds Max

- Support and learning

Maya

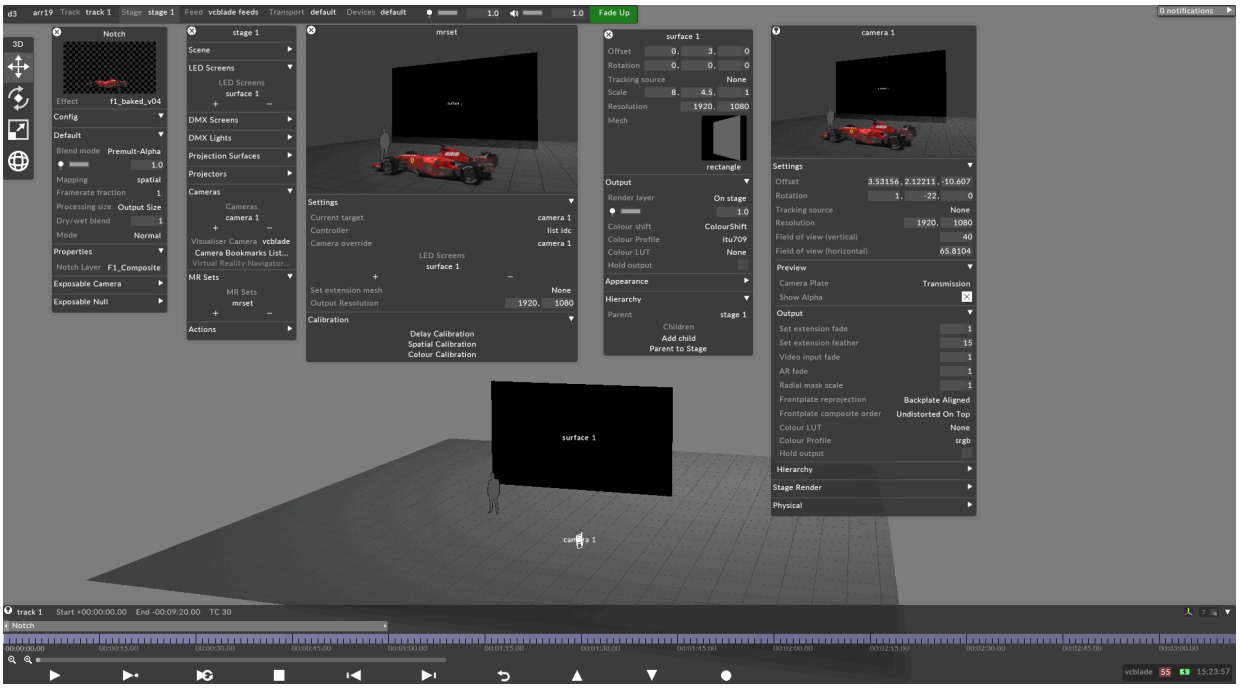
- Support and learning

AR

AR content from a generative content engine, such as Notch, can be placed in the 3D scene overlaying existing video content or a live video input using this workflow.

1. Create an MR set.
2. Create a virtual camera(s).
3. Create an LED screen on the stage, check that the Render Layer (located in the Output tab of the layer) is set to ‘On Stage’.
4. In the MR Set properties editor, add the LED surface to the MR set, create an indirection controller, and select the virtual camera(s) as the resource.
5. Create a Notch Layer and select the desired Notch block that you want to use as an AR object.
6. Create a Spatial mapping for the Notch layer and add the virtual camera(s) as a screen.
7. Set the Camera plate of the Spatial mapping to Backplate.

At this point, you will see the AR object in the Camera transmission preview and the MR Set preview windows.



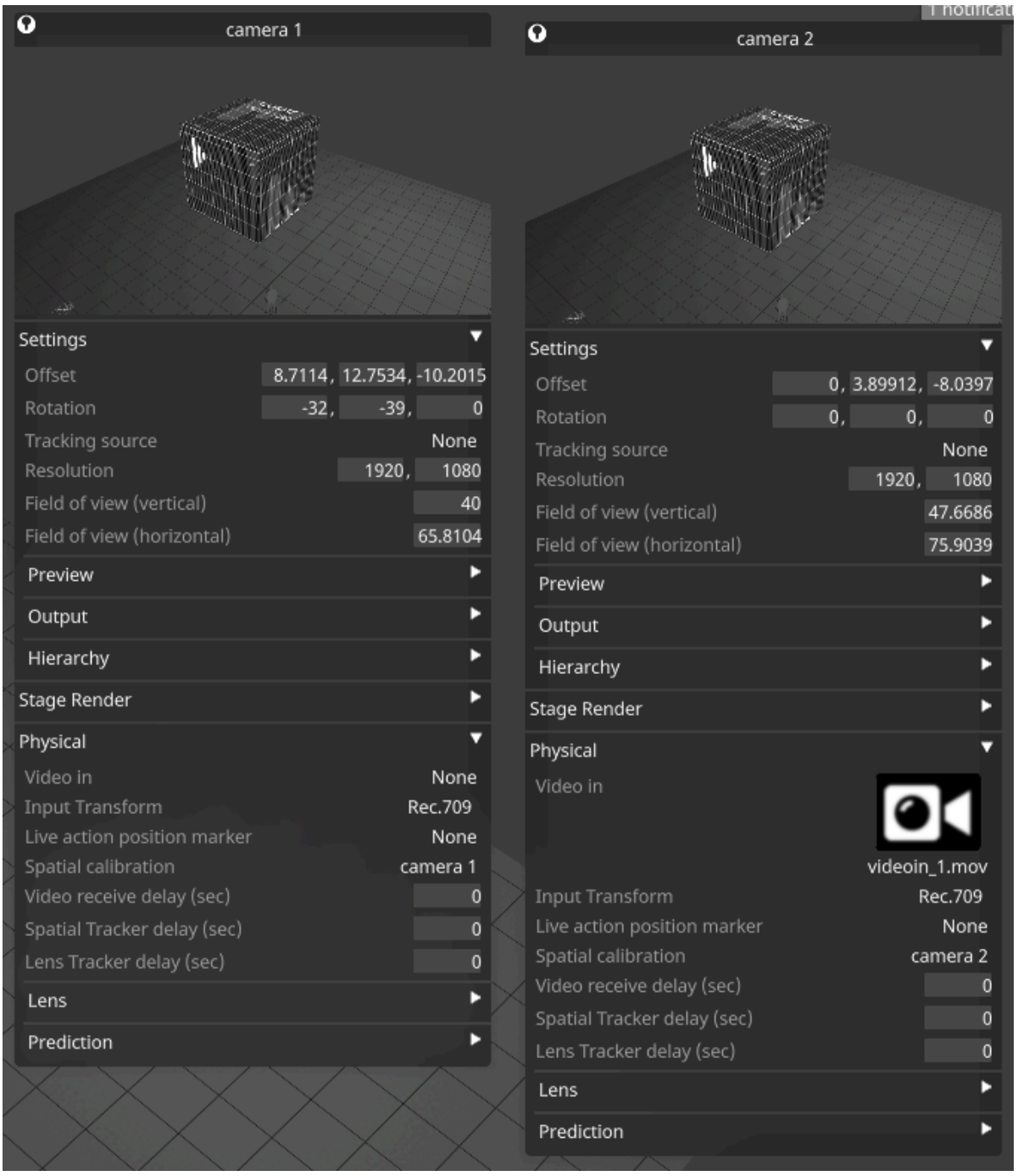
AR Spatial Calibration

r20 introduces a new way to manually align AR objects without the need to perform an xR calibration from an MR Set.

This workflow involves an AR object which requires a virtual camera. The following steps are intended as a practice scenario:

1. Add 2 cameras to the stage; Camera 1 represents the video input (real world) camera while camera 2 represents the virtual (composite) camera.
2. Patch video input 1 to Camera 1.

3. On Camera 2, expand the Physical tab, and then set the Video in to Video Input 1.

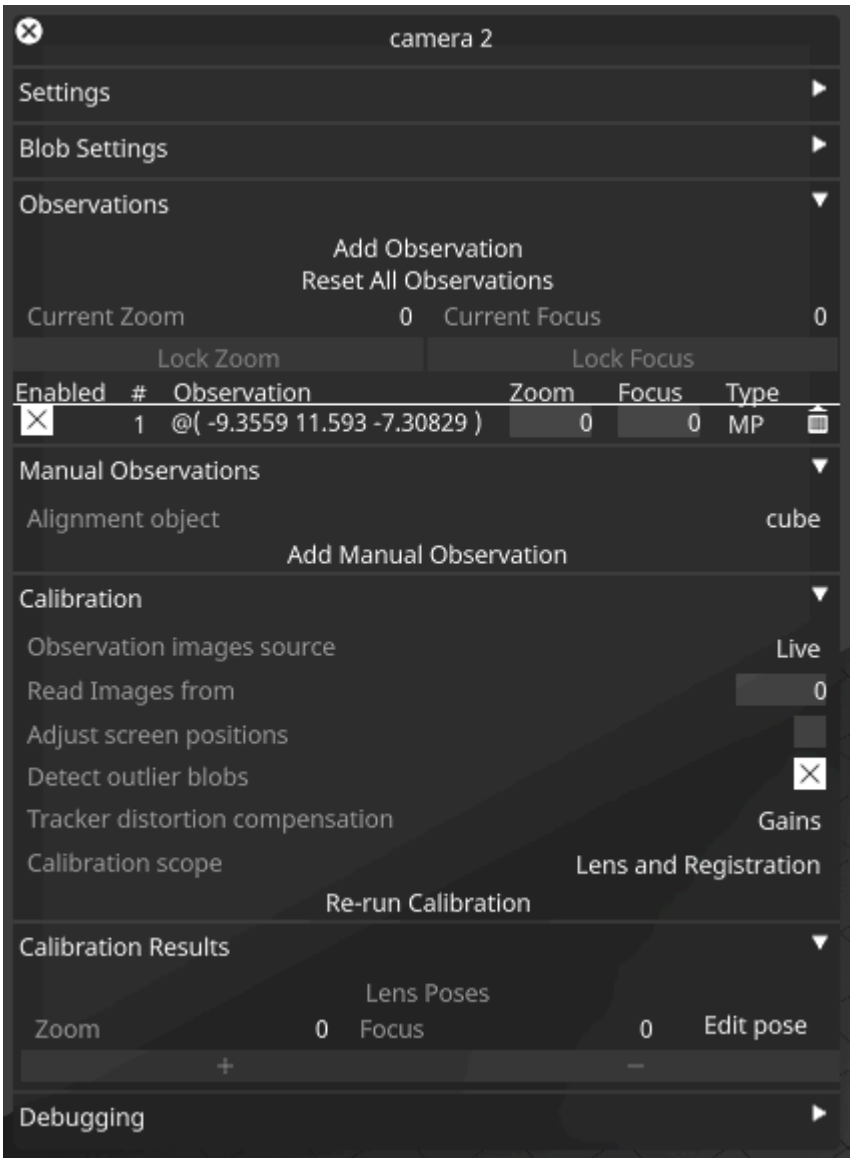


4. Add the AR object to the stage.
5. Add a virtual lineup layer to the timeline.

Next, open the properties editor for camera 2 (virtual camera), then follow these steps:

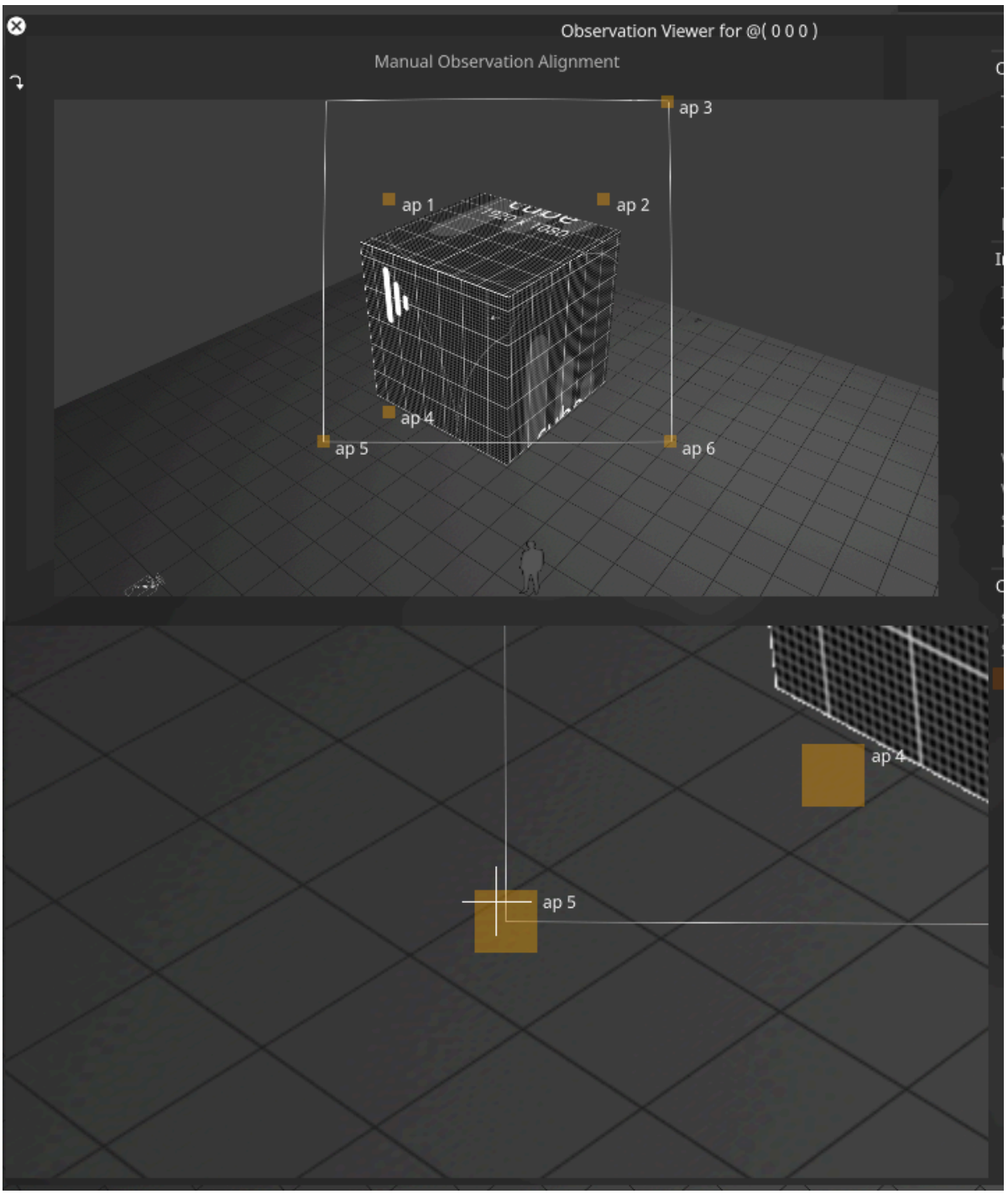
1. Go to the **Spatial Calibration** field in Physical tab.

2. Create a new spatial calibration, then right-click on the calibration to open its properties editor.



3. In the Manual Observation tab, add an alignment object- this is the object you're aligning to
4. Click **Manual Observation**.

5. Right-click on the manual observation that you created to open the Observation Viewer.



6. Click on each of the vertices of the wireframe object in the Observation Viewer (check the number of points needed message) and drag to the corresponding vertices of the alignment object.
7. Click on Update Calibration once all points have been aligned
8. Add additional observations as needed using the same process.

> **Note**
>
> Each time you move your camera, you will need to take another manual observation. A minimum of 3 observations are required to get it aligned where it will stay aligned when you move the camera.

Projector Calibration

Projection Calibration is the process of calibrating your projectors to the 3D space.

Designer offers three different calibration methods. Each has their own benefits and draw backs. We recommend planning which form of calibration will be used during pre-production.

- Manual calibration
- QuickCal
- OmniCal

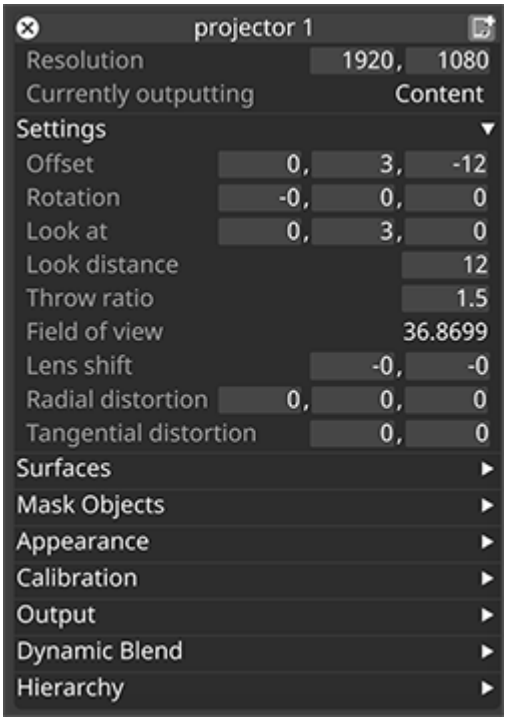
Manual Projector Calibration

> **Warning**
>
> Reset all digital and optical warps inside the projector, including lens-shift, otherwise the lineup features will be interrupted.

- Open the projector editor by either right-clicking the projector directly in the Stage, or by right-clicking the projector from the screens list in the **Stage Editor**.
- Familiarize yourself with the **properties of a projector**. After this follow the instructions explained below.

Set the correct resolution

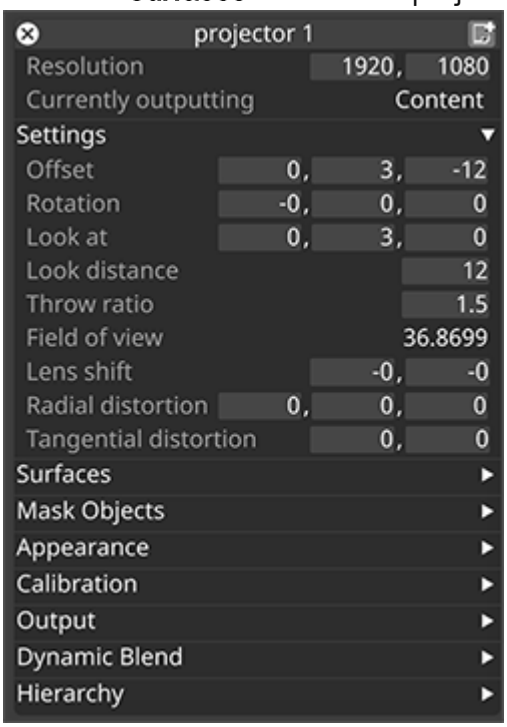
Set the resolution of the projector to match the corresponding output head's resolution.



> **Warning**
>
> If the resolution of the projector doesn't match the real world output, you will end up with an incorrect calibration.

Add surfaces

Under the **Surfaces** tab add the projection screens that the particular projector is covering.



> **Warning**
>
> If you don't add a screen to the projector the output will stay black.

Add the projector outputs to the Output Feeds

Ensure all of the Feed rectangles from the projector outputs have been added to the output heads.

Please see adding feed rectangles for information how to do this.

Place the projector correctly

Change the **pos** (projector position) and **the throw ratio** (lens value) properties so that the projector covers the required part of the screen surface. A laser measure may be required to calculate the correct position by measuring the distance from the physical projector to the video surface.

Adjust the Look at position

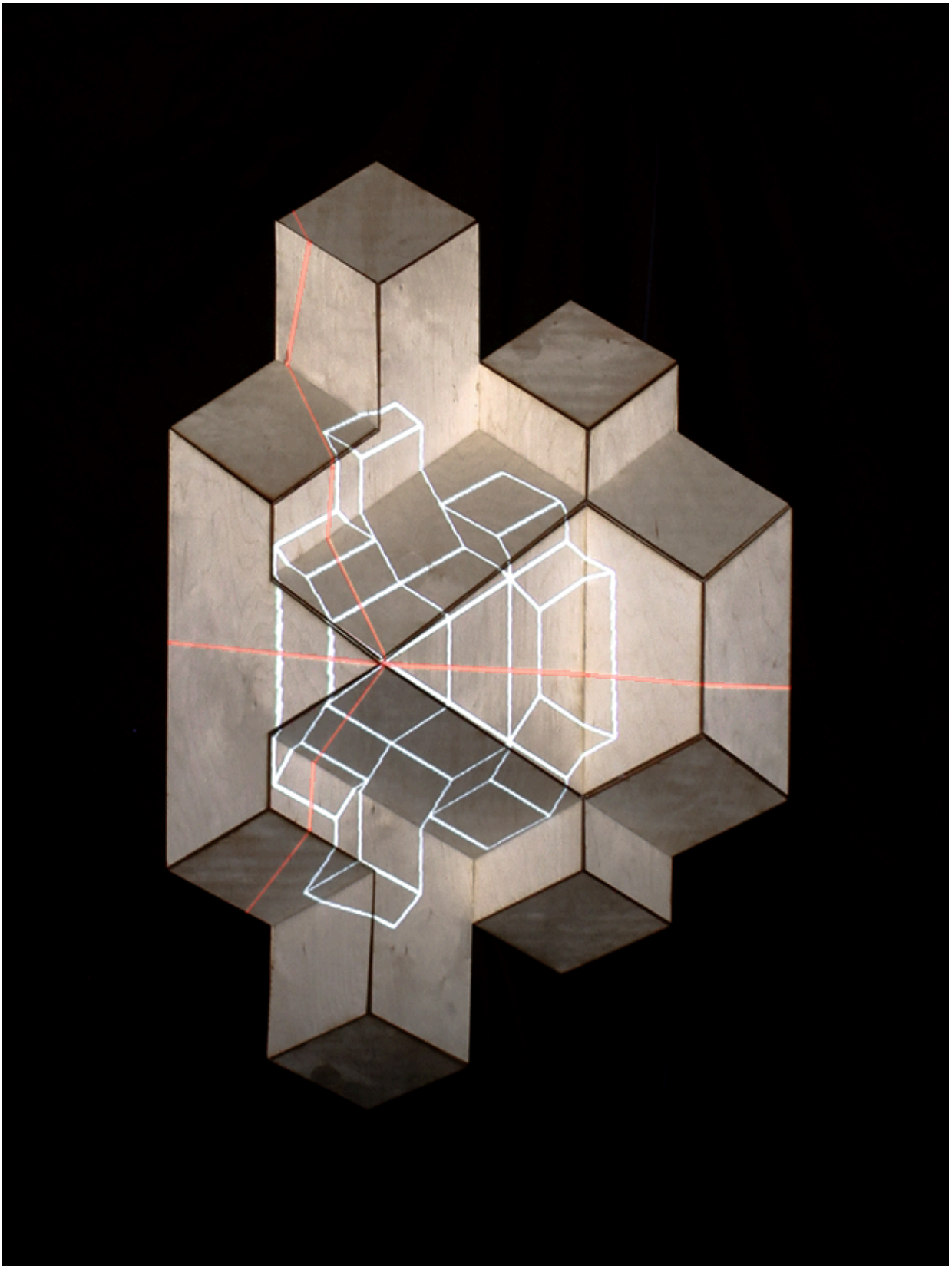
The **Look at** position of a projector defines the centre point of its corresponding output feed. When lining up a virtual projector to the physical projector it is therefore crucial to match the Look at position to its corresponding point in the real world.

Disguise Designer has a built-in wireframe feature allowing the system to generate a line-drawing based on the 3D mesh of the projection surface. In the same output frame Designer will also output a red cross positioned in the centre of the output feed. When the wireframe test pattern is applied, Designer will output this red cross regardless of the orientation of the projector.

Consequently, if the look at position of the projector is aligned to the red cross on the physical projection surface, the virtual projector and the physical projector are orientated around the same point which is a great starting point for an accurate manual lineup.

Align the look at position of the projector

- Change the output mode to **wireframe** by clicking the **output** tab at the bottom of the projector editor.
- Return to the Stage level. Set the **step** value of the look at position to 0.01 to enable smoother scrolling of the look at values. To change the step value, right-click the property **name** and change the value in the Step value property.
- Begin aligning the look at position of the projector to the red cross by comparing the look at positions crosshair in the Stage level and the red cross being outputted from the physical projector.



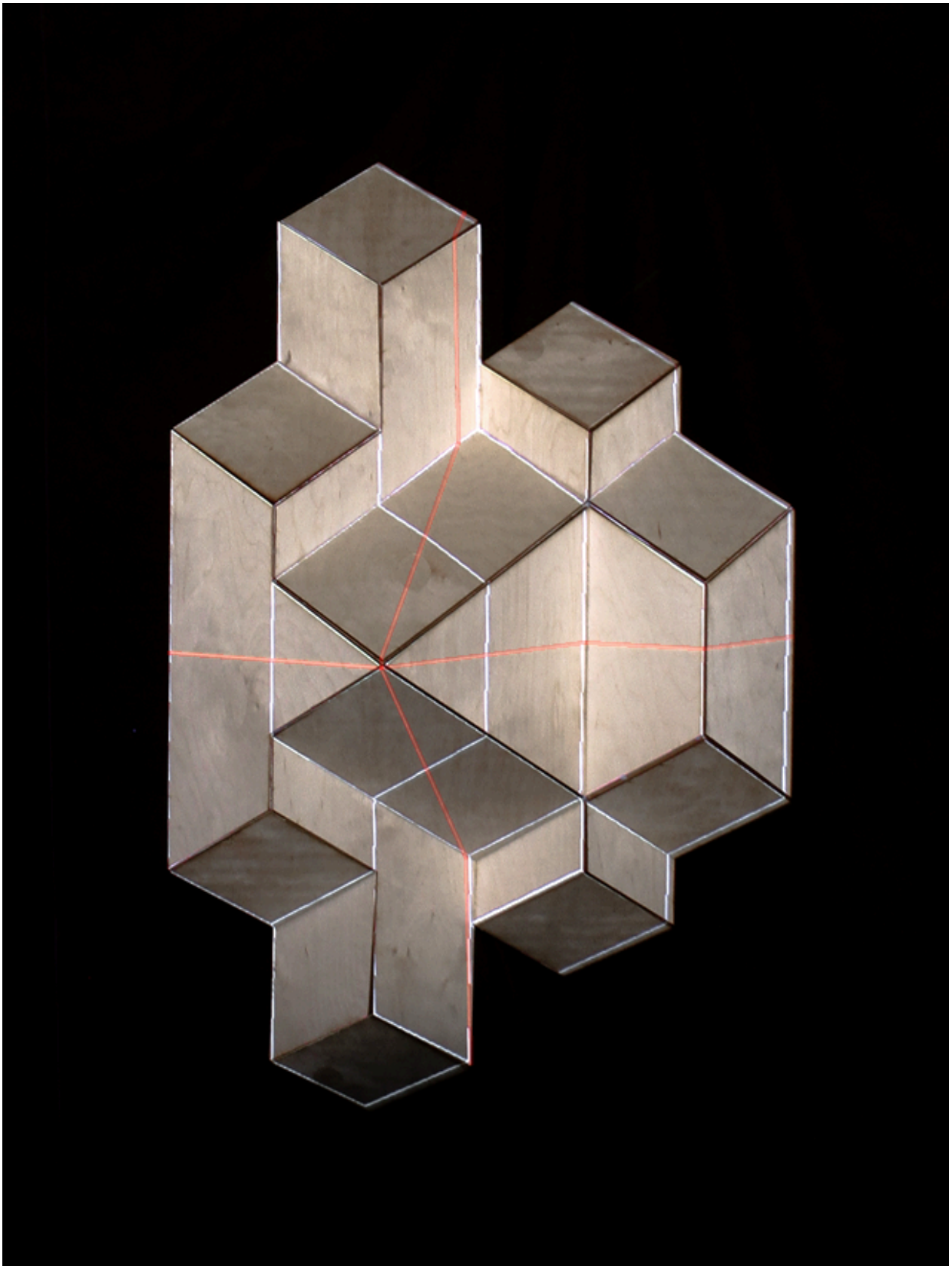
Look at position of the virtual projector now matches its corresponding point on the physical video screen, in this example a wooden sculpture.

Adjust the throw: Ratio

- To zoom in on the physical video screens content, change the **throwRatio** value. This value corresponds to the lens size of the physical projector.
- If the exact lens size is known (for example a fixed lens size is being used), type it in.
- If a zoom lens is being used, set the start value to the lowest (or highest) value in the zoom range. Slowly change the value by scrolling the mouse wheel in the property field.

Fine tune property values

- After setting the initial lens value, try not to edit the position properties. Instead, start adjusting the rotation parameters if needed, in particular the x and y rotations. Aim to establish parallel lines mapping onto the video screen globally rather than focusing only on one part of the screen. Adjust the step value if needed.
- Go back and fine tune the values of the lookAt position to center the output to the physical video screen. **Remember to establish parallel lines**.
- Adjust the throw ratio to zoom in/out of the content on the video screen. Assuming that the 3D mesh is accurate to the physical video screen, the mapping should gradually fall into place.



Projector has been manually calibrated; the 3D mesh test pattern now lines up with the wooden sculpture.

If the projection surface does not match the 3D model after carrying out manual calibration the lineup may need to be fine-tuned. Please see the sub-chapter **Warping outputs** for more

information.

MultiPose Projector Calibration

The QuickCal configuration includes a Multi-Pose workflow to improve the calibration of spaces where projection surfaces move or rotate. This is helpful when a surface used for calibration is not fully covering the space in its current pose, or if there are no good visible reference points that cover the whole space. A typical example is a long narrow(ish) object like a car, that is being projected on while being rotated on a turntable.

This workflow is for manual QuickCal calibration, and not to be confused with Multi-Pose Alignment in OmniCal's QuickAlign, which is for 3D alignment based on automation input.

Multi-Pose Projection Workflow

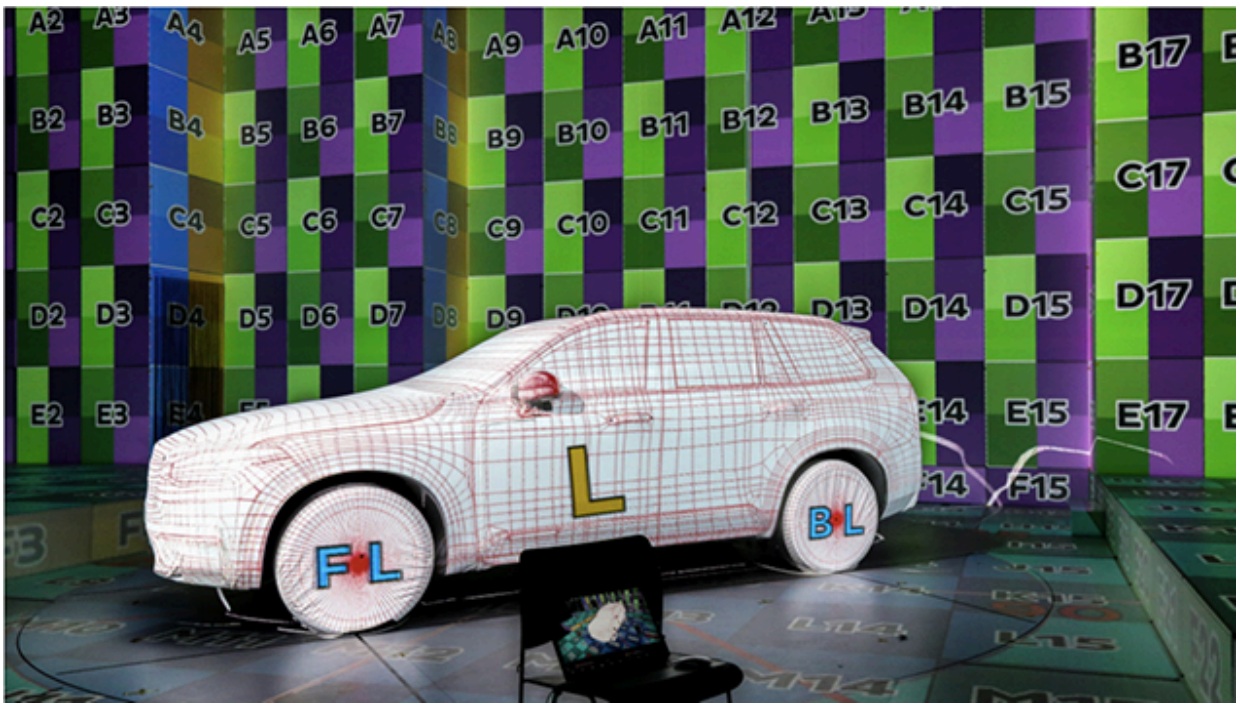
1. Ensure that the object's position and rotation are correctly controlled via an automation system.
2. Place the moveable object in a particular position (the "pose") and then perform a QuickCal calibration, as usual.
3. Then use automation to rotate or move the object to a second position. This will change the current 3D position of any existing reference points.
4. Line up the existing reference points to the respective correct 2D coordinates for this pose.
5. Add and line up new reference points, if necessary, until the projected image is correct and sharp in this pose.
6. This process is then repeated for as many poses as are required.

Unlike in the single-pose workflow, we see the reference points rotate along with the model. Once a reference point has been aligned, its 3D position at that time is stored along with the 2D lineup coordinate. When automation input changes the current 3D position of a reference point, it will not affect the previously saved lineups.

Modify an existing pose

An existing pose can be modified by moving the object back near a previous position using the automation system. QuickCal has the option to hide markers for poses other than the current one. This may help find an earlier pose. The match does not have to be exact. When a reference point is lined with a 3D position close to an earlier pose, then the earlier pose gets overwritten.

Example



Example application : a car on a rotating turntable, with encoder

In this image is an example of a car on a motorised turntable. During configuration, as the car rotates, the calibrated points gradually drift from their locations on the real model, showing that the calibration isn't perfect across the whole space. Once the car is in a new pose, we simply drag the existing markers to the correct positions, and add new ones if necessary, until the projected image is correct and sharp across all possible car positions in the space.

Typically it would be enough to use two poses (e.g. front and side view), so that the extremes of the space are covered by lined up QuickCal points.

Calibration Errors

The process of calibrating a camera or projector requires a set of input data and a model that expresses the geometry and physics. This model is a set of known equations that are determined by unknown parameters. For projectors or cameras the model usually involves a *3D position and orientation*, as well as lens parameters such as *focal length*, *lens shift* or *distortion coefficients*.

During the calibration process suitable model parameters are estimated, based on the input data observations.

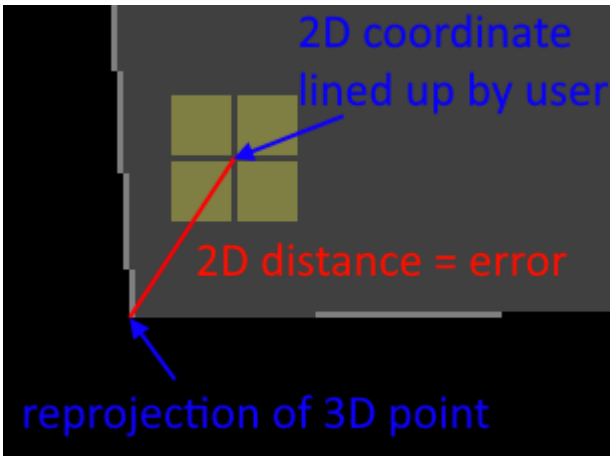
Calibration errors are a measure of how well the estimated model parameters conform to the input data.

Calibration in Disguise typically involves several pairs of 2D and 3D points. The source of these point pairs can be for example:

- manual input (2D lineup position in QuickCal)
- automatic detection (2D blob positions in OmniCal or xR)
- existing file data (3D mesh vertices used in QuickCal)
- reconstructed 3D points (based on automatically detected 2D blobs in OmniCal or xR)

Error measurements are often calculated as **Reprojection Errors**: the input 3D points are projected back onto the projector (or camera) image using the equations and parameters of the calibrated model.

The resulting 2D points are compared against the original input 2D points. The difference between them (usually the absolute 2D distance) is used for the error calculation. The total error is aggregated over all point pairs in different ways (see below).



The unit for these reprojection errors is usually in pixels.

Lower error values are better.

Typically, a reprojection error below 1 pixel is desirable, although in some circumstances higher values can be acceptable. This is a judgement call that depends on the shape of the surface, what content is used, or the effective resolution that the content will be visible at on the projection surface. If the audience is far away, or the resolution is very high, then higher errors may not be noticeable.

In QuickCal it is more difficult to achieve true subpixel accuracy, since the user can only line up the 2D cursor at full pixel boundaries. This introduces systematic errors of up to a half pixel in either direction.

In OmniCal and xR, the 2D blobs are detected with sub-pixel accuracy due to the relative size of the blobs in the camera images.

There are several ways of calculating errors from the above-mentioned point differences:

Average or Mean absolute error (Avg)

The average (or mean) error is simply the sum of absolute errors divided by the number of errors. The use of absolute values means that it cannot be negative. A value of 0 would be a perfect result. Individual differences have a proportional influence on the total average error. This makes it easy to understand and a good measure of success.

Root Mean Square error (RMSE)

The RMSE is the square root of the average of squared individual errors. It also cannot be negative, and a value of 0 would be a perfect result. Because the individual differences are squared, larger errors have a disproportionately large effect on the total RMSE. Consequently, RMSE is sensitive to outliers.

This is why OmniCal will remove any outliers from the data set before and between calibration steps. The RMS error is often used during the calibration itself to drive the process faster in the best “direction”. But it is also used at the end of a calibration as a measure of success.

In QuickCal, the RMS reprojection error is always calculated again after the calibration completes. For some algorithms, QuickCal also reports back the internal RMS error from the algorithm itself. In the UI this is shown as “rest”.

Notes

- It is possible for two different calibration input data sets to have the same average error, but different RMS errors.
- A calibration error is not always a guarantee for a successful calibration that is visually convincing. If there are only few input point pairs, then it is possible that redundant information or mistakes lead to unrealistic model parameters, despite low error values.

Calibration Error Visualisation

In Designer different colours are used to give operators a clearer idea of what calibration errors mean.

Below is a list of colours used in QuickCal and OmniCal and the meaning they represent. The meaning depends if the error value is a total error like RMSE, or whether it is only for a single 2D/3D point pair from the input data set. Any error value is alway represented by a gradient between any of these colours, so a green yellow is better than pure yellow, which again is better than orange.

Colour	Error	Meaning for RMSE	Meaning for Individual Point Pair
Green	Low error (good quality)	Calibration was successful.	Good contribution to calibration.
Yellow	Medium-high error	Calibration quality is borderline acceptable. Need to make a visual judgement whether it looks good enough or not.	Potentially problematic contribution to calibration. If there too many point pairs like this, then overall calibration may not be good enough
Red	High error	Unacceptable quality	Bad contribution to calibration (a single bad point pair can make a calibration fail). **QuickCal**: operator should re-align or remove it. **OmniCal**: operator should aim to filter it out using outlier handling
Magenta	Not calibrated	Calibration failed.There were not enough 2D/3D input point pairs fo the selected algorithm. Input data was bad / inconsistent.	No contribution to calibration. **OmniCal**: rejected due to outlier handling.

Further Reading

- Mean absolute error [Wikipedia]
- Root-mean-square deviation [Wikipedia]

Quick Calibration

QuickCal is a user-driven process of calibrating a virtual projector's position, orientation and lens properties to match the real-world projector. QuickCal is based on reference points on given 3D meshes for a projector's target surfaces, and user-defined 2D image coordinates that are manually assigned to these 3D reference points. As long as the 3D meshes are good matches to the real-world projection surfaces, QuickCal can accurately calculate the projector parameters.

The basic process of QuickCal is relatively simple : drop reference points onto the 3D model of the projection surface, and then 'line up' by dragging each point in the projector's output raster until it hits the corresponding point on the real surface. Once you've done enough points (about 10-15 per projector) Designer can work out exactly where the projector is, and what lens qualities it has.

3D mesh accuracy

When calibrating projectors using QuickCal it is *crucial* that the 3D mesh object is modelled accurately to the physical model. The reason for this is that Disguise's calibration algorithm assumes that the virtual reference points link exactly to their corresponding real-world points. The best way to generate a highly accurate 3D model is to laser scan the physical model, or to laser cut the physical model based on the same 3D file you later use in Designer. For building projections we recommend a mesh with a +-5mm error margin.

Using QuickCal

To start using QuickCal, first move the virtual projector into roughly the right position and orientation, and set up its lens qualities to roughly match the real projector. You don't have to be particularly accurate in this, but when you're trying to select projectors in the visualiser for editing, it's easier when they're in roughly the right place. Then move on to creating reference points, and then line them up.

Tips & Tricks

General QuickCal background information

The calibration goal is to calculate a projector's projection matrix so that it produces physically valid projections on its assigned projection surfaces in the real world (the scene). This means that the matrix does not need to be absolutely physically correct, but should be correct for at least this projection area (or rather volume). Indeed, it helps to think of the calibration as a process that works on a volume in space, rather than on 2-dimensional surfaces.

The calibration process uses as main inputs:

3D Reference Points (in world space)

These are usually created from vertices of the mesh/object. Note that the mesh needs to match the real object, at least in the vertices used for creating Reference Points. In d3 Reference Points are 3D objects themselves, and a certain point can be used by several Projectors.

2D coordinates (in the projector image plane)

These are created during line up of a Reference Point in a given Projector. In Designer the Projector Configuration stores a pair that links the 2D coordinate to the 3D Reference Point.

Projector matrix

The projection matrix is a linear transformation that represents the projector's position, orientation and internal sensor and lens properties. Note that Designer currently does not support correction of non-linear lens distortion in QuickCal. Lens distortion is only calculated and compensated for in OmniCal.

In Designer the projection matrix is applied to all 3D meshes in the Surfaces list, so they get transformed and projected on to the virtual Projector's image plane. This creates the projector's output image that is visible in the Feed Scene.

Choice of Reference Points and Order of line-ups

A Reference Point and the 2D line-up coordinate together form a correspondence point pair. Several such pairs are required for a calibration. QuickCal supports several algorithms that have different requirements on the minimum number of points (e.g. 4, 6, or more).

It doesn't matter for the calibration itself which object/mesh the Reference Points come from, as long as that 3D point exists in the real world as a recognisable feature. Object corners are usually good choices.

It is possible to use Reference Points from several objects for a single Projector.

Important requirements are:

- The real-world equivalents of the Reference Points are actually visible by the physical projector (lie inside its light cone).
- All Reference Points that are used for a projector:
  - span a large enough 3D volume. Often this is referred to as "adding enough depth". The main issue is to ensure that the Reference Points do not all lie on the same (virtual) plane in 3D space.
  - should evenly cover a large enough area in the Projector's image. It's a good idea to use Reference Points that get lined-up in all four corners of the projector output image.

Additionally, if you want to verify the wireframe of an object during QuickCal line-up, then the objects needs to be added to the Projector's Surfaces list.

It is worth to keep in mind that calibrating creates a working volume for a projector, using the 3D Reference Points as a helper. So the volume in which the projector "is valid in" is always limited by the extent of Reference Points you have lined up. As an example, lining up 6 points that are all in the same image corner of the projector will likely produce a calibrated matrix that doesn't work well for the rest of the image. The same goes for points that are in close proximity in the real world, i.e. the calibrated matrix will only be valid in the small volume within those points.

The order in which Reference Points are created and lined up does not affect the calculation of the projection matrix. The calibration will produce the same result no matter what points are lined up first.

However, to help the calibration work out things faster, it makes sense to line up Reference Points from extreme parts of the scene first, before adding points that are closer to each other. Otherwise there may not be a good visual output until enough points are lined up so that they cover the whole scene with enough depth.

Short QuickCal Rules

1. leave the Calibration Method setting at Auto, unless you really have a good reason to force a specific algorithm. Auto will choose the best algorithm based on how many Reference Points you have, and what their 3D relationship is.
2. Don't "cheat" with your line-up coordinates unless you absolutely have to.
3. Never "cheat" until all Reference Points have been added and lined-up.
4. Line up the extreme parts of the mesh(es) and projector image first (e.g. image corners Left/Right/Top/Bottom and Close/Far in the scene). Then improve on that by adding more points. E.g. concentrate on Reference Points in areas where the projected images are still "off" when looking at the real object.

Limited Scene Depth and number of Reference Points

Meshes/Objects that are flat or have little depth are more difficult to use for calibration. This is even more problematic if the real world object has few or no visible features, like a wall where only 4 corners are easily usable as Reference Points.

However, with 4 correspondence point pairs it's already possible to do a simple calibration (usually IterLM algorithm). This calibration is not able to calculate internal lens parameters such as lens shift or throw ratio. It will take values for these parameters from the manual projector settings. Note that these settings are still editable when using QuickCal with under 6 points, or if certain algorithms are set. It can be difficult to get these values right by hand, but if the throw ratio and lens shift of the physical projector is roughly known, then adjusting this can help improve the calibration significantly.

With 6 or more correspondence point pairs a different algorithm is used (usually Zhang algorithm), which also calculates lens shift and throw ratio parameters automatically. If the Reference Points provide little or no depth, then this algorithm has problems finding a good projector matrix. This is because mathematically it is very difficult to differentiate between some internal lens parameters and 3D parameters. For example throw ratio (or focal length, if you prefer) and distance to an object are interchangeable to a certain degree. Think of the famous Dolly Zoom effect used by Hitchcock in the film “Vertigo”, where the face always stays the same size, only fore- and background objects get smaller or disappear. This ambiguity can only be resolved by more depth. It is necessary to line up more points from the scene background/foreground to constrain the calibration better, essentially thereby increasing the calibration volume.

In practice, it may be preferable in certain rare situations to stick with 4 or 5 points, rather than adding a 6th, because the switch to the different algorithm may make the visual result worse rather than better.

Correspondence Point influence on calibration

A correspondence point pair is not like guide points for splines, or a fixed warp or rule that associates the two to each other.

Instead, they are inputs into an iterative algorithm that tries to solve a matrix so that all of these point pair conditions are (more or less) satisfied. But due to mesh/reality discrepancies, line-up inaccuracies, constraints of the optical model, even rounding errors, etc. there will never be a perfect matrix that fits all these pairs. There are many matrices though that will be good enough, so that the error for each of these point pairs is small (e.g. ideally below 1 pixel). If the inaccuracies in the input data are high, then some correspondence point pairs will basically contradict each other strongly. In that case the overall error will be high, too (e.g. 5-10 pixels, or even more). If the error is due to mesh/object discrepancies, then “cheating” might help (see below).

Cheating

If the 3D model and the physical object don’t match exactly, then QuickCal allows to “cheat” a bit. “Cheating” isn’t recommended, but often it’s not possible to get a better 3D model while being on-site.

There are several ways to “cheat”:

- Moving the line-up cursor slightly off from where it should be, so that the overall calibration looks better and the reprojection error gets smaller.

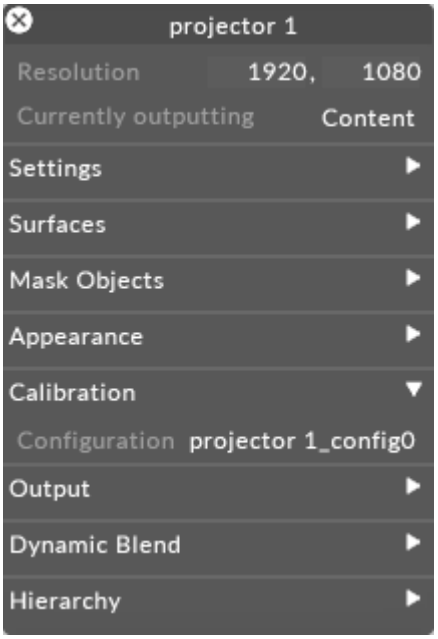
> **Note**
>
> The advanced option 'Disable reference point bounds' allows moving the 2D line-up coordinate outside of the projector's image, e.g. to a negative value.

- Moving a 3D Reference Point away from its original vertex position. In QuickCal this can be done in "Manage" mode by holding the Shift key and moving the 3D Reference Point left, right, up or down. This will move the point in 3D space according to the current visualiser view, and allows placing the point in a non-vertex position. By default, Reference Points snap to the nearest mesh vertex, indicated by a green cross.

The problem with any of these tricks is to know which Reference Points to cheat with, and how it eventually affects the calibration.

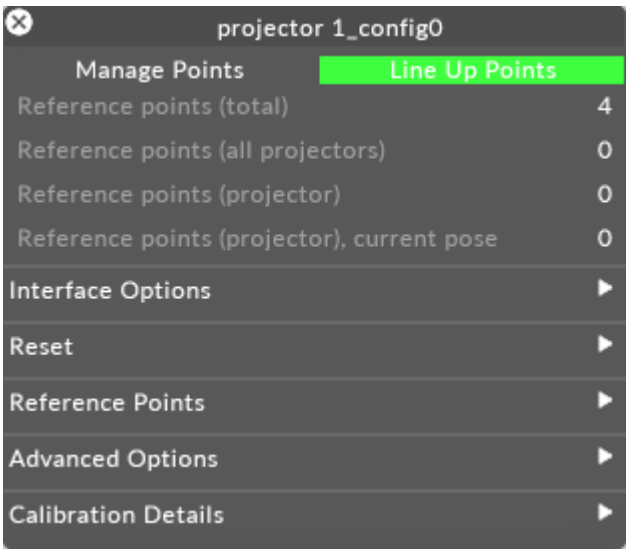
QuickCal Projectors

1. Right-click the projector you wish to line up.
2. Right-click the configuration file to open the QuickCal editor.



Projector configuration menu.

3. At the top of the QuickCal editor select **line up points**.



Line up point selection.
You must have the projector you're configuring assigned to an output otherwise the button is greyed-out

4. Left-click a point (representing a reference point on the video screen), hold the left-click and drag the cursor in the output and match it to its physical corresponding point.

> **Note**
>
> The output of the projector you working with will tinted the colour that you have set in the **appearance** tab in the projectors configuration.

5. Once the point is in the right position release the point and now its set.
6. To adjust cursor location using finer increments use the arrow keys to adjust in 1px movements.

When you have adjusted a point it will be displayed in the colour that matches the projectors colour.



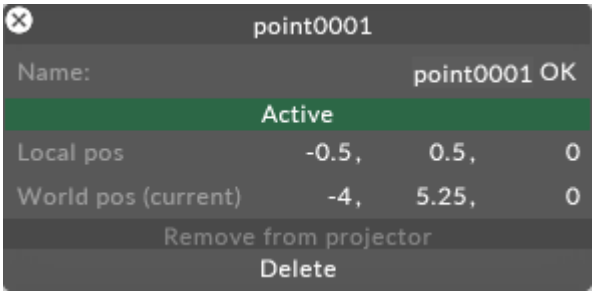
Buckingham Palace with points calibrated.

Remove a reference point from the projector

This will remove the 2D lineup between the selected reference point and the current projector. The reference point itself will not be deleted.

To remove a reference point

- Right-click on the reference point that you wish to remove from the projector lineup. The options menu will open.



To remove a reference point from a projector

- Left-click on the remove from projector button.

Rotate Controls

This option allows you to rotate your controls whilst lining up, This is especially useful when your projectors are rotated for example if they are portrait in orientation rather than landscape.

There are four options

- None: Mouse and keyboard operate normally.
- 90 degrees clockwise: mouse and keyboard are rotated 90 degrees clockwise
- 180 degrees: mouse and keyboard are rotated 180 degrees.
- 180 degrees anti-clockwise: mouse and keyboard are rotated 180 degrees anti-clockwise.

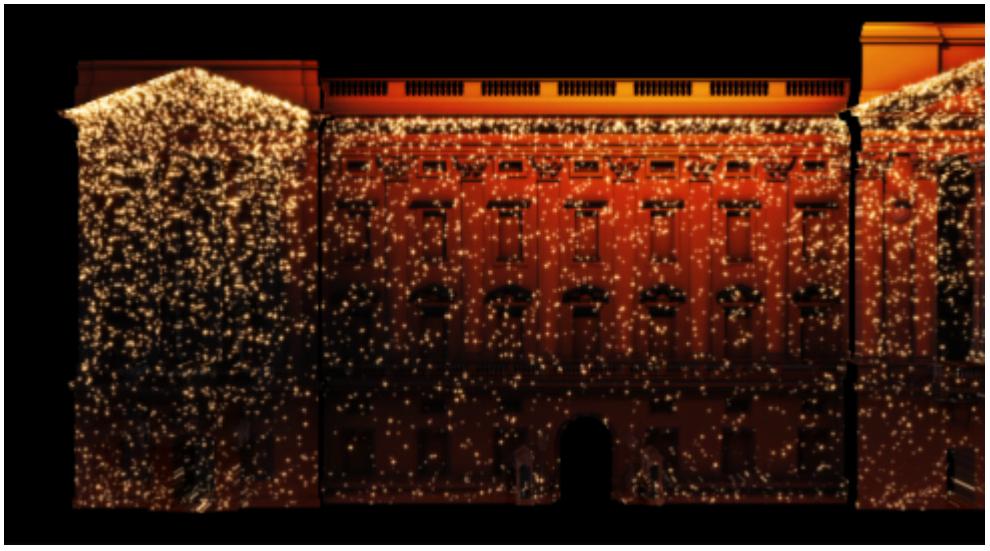
Lineup output mode

Lineup output mode allows you to change what is applied to the output whilst lining up.

During QuickCal lineup all lined up reference points will be drawn on top of the selected output.

You can choose between the following output modes:

- **Content**: Outputs the content that's on the timeline.



- **Wireframe**: Outputs the wire frame of the model.

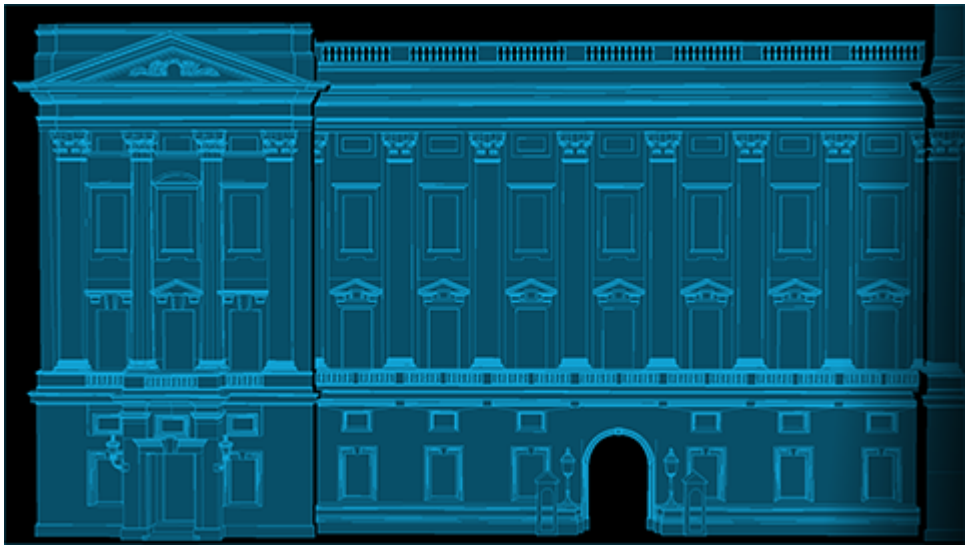


Image of wireframe applied to Buckingham Palace.

- **Identify**: Outputs a full colour grid that has the projectors name on it.



- **Grid**: Outputs a grid that is applied to the model

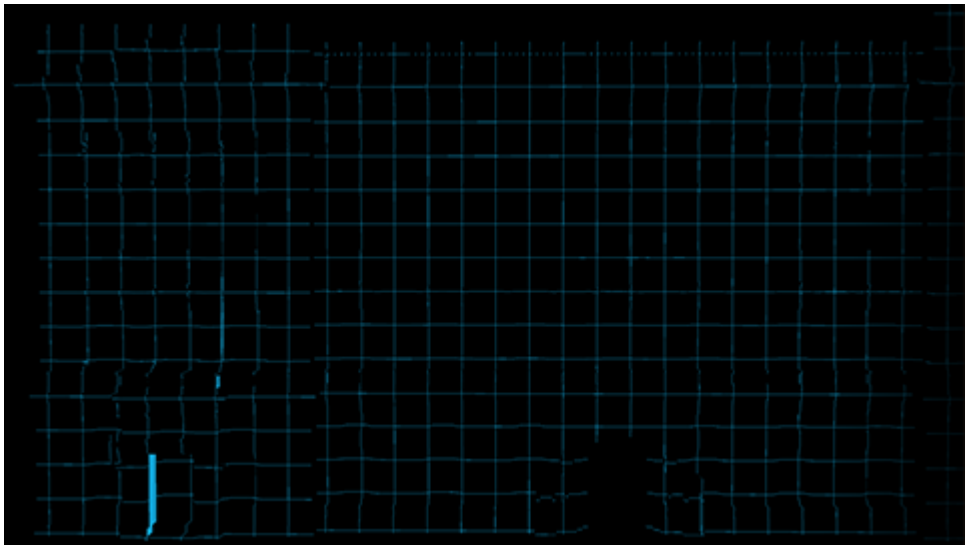


Image of grid pattern applied to Buckingham Palace.

- **None**: Outputs nothing but the line up points.

Cursor Types

There are four cursor types to choose from:

- Horizontal

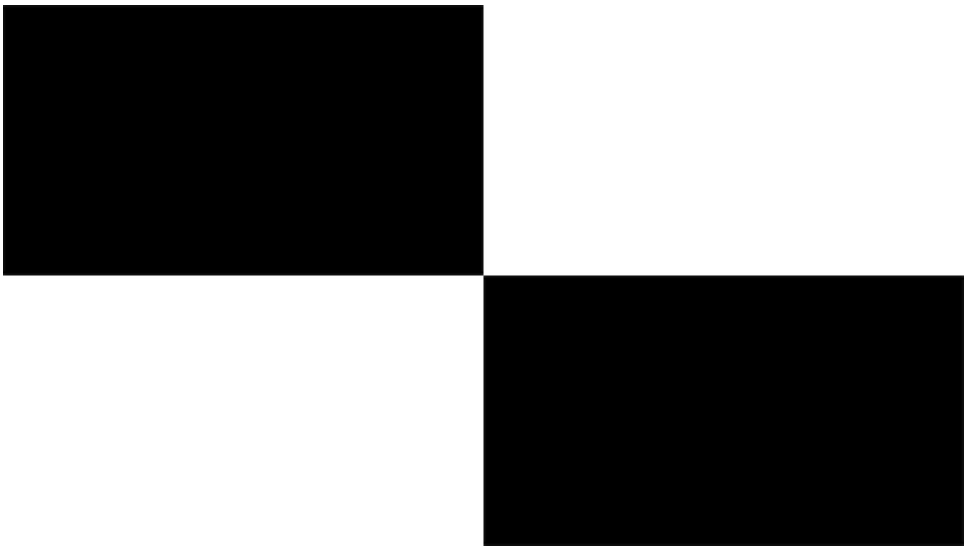


Image of horizontal lineup cursor.

- Diagonal

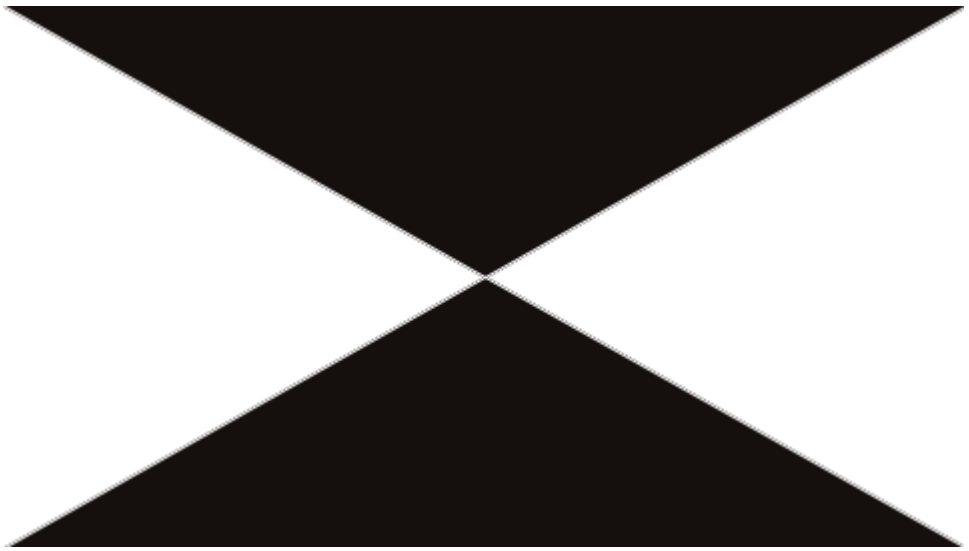


Image of horizontal diagonal cursor.

- Horizontal Lines

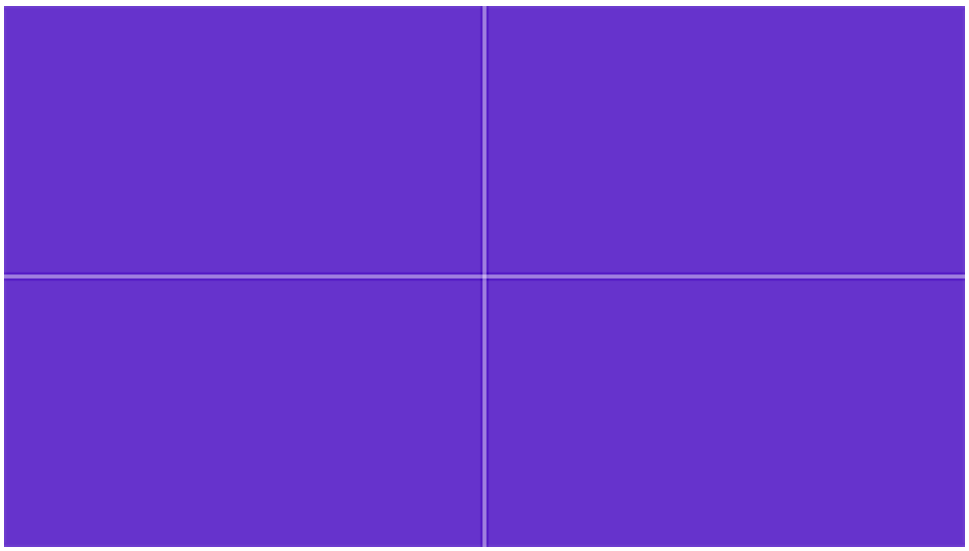


Image of horizontal lines cursor.

- Diagonal Lines

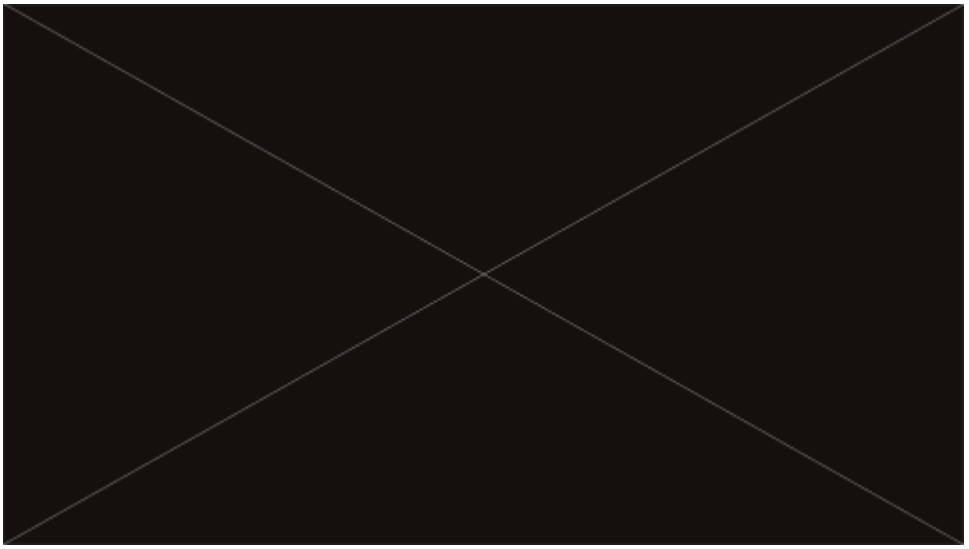


Image of diagonal lines cursor.

Marker Size

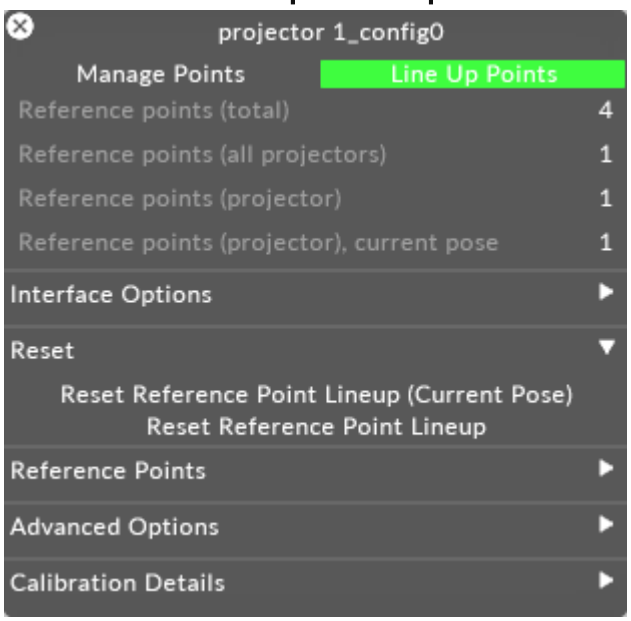
This setting allows you to adjust the size of the marker that's left behind once you have calibrated a point. The default marker size is 16.

Reset options

To reset the projector so no points have been selected or calibrated either:

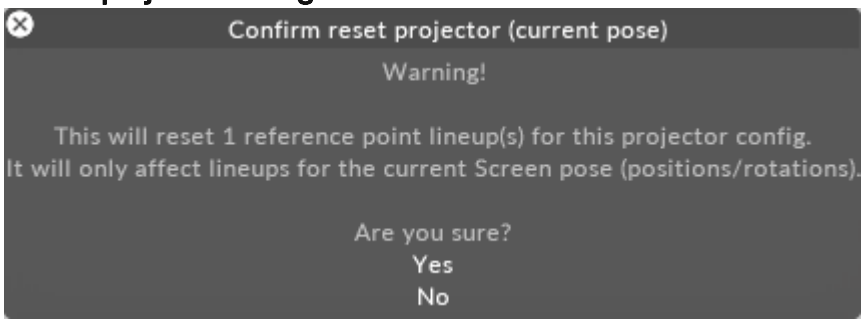
- select **reset reference point lineup (current pose)** to reset the points for that pose in a multi-pose line up (see the Multi-pose calibration sub-chapter for more info on this topic)

- select **reset reference point lineup**



Reset reference point lineup.

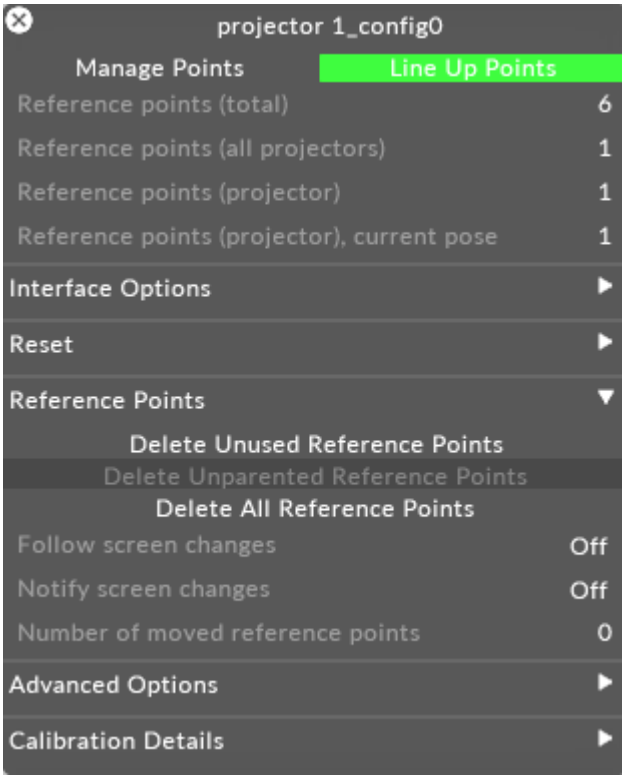
- A warning message will pop up when you select **reset projector config,**Click **yes** to **reset projector config.**



Confirm reset projection config.

Removing reference pointsS

There are 3 options for removing reference points under the **Reference points** tab:



Deleting reference points.

Delete unused reference points

This option deletes reference points which have not been associated with a projector lineup.

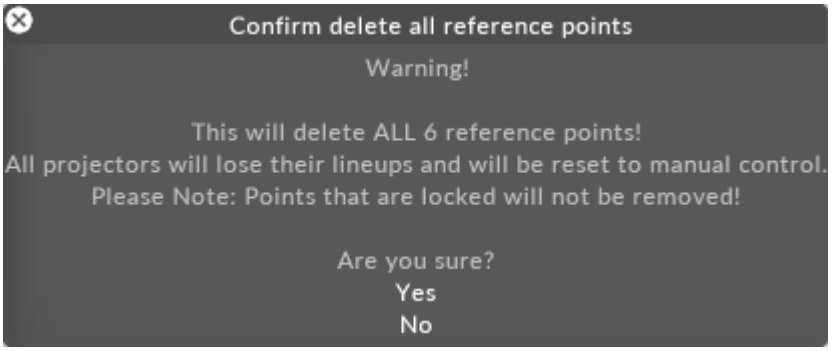
Delete unparented reference points

This option deletes reference points which are not parented to a screen

Delete all reference points

This option deletes all reference points from the virtual stage.

A warning message will pop up when you select **delete all reference points , Click** yes **to delete all reference points.**



Confirm delete of all reference points.

Delete all reference points is a global action, It will affect all the reference points on the model not just the ones associated with the current projector.

Advanced options

Follow screen changes

This option allows you to choose if the reference points follow screen changes. For example, if you change your screen position, the reference points will move with the screen.

This will automatically update any projector's QuickCal calibration using the new 3D positions of the reference points

Notify screen changes

This options allows you to choose if you want to be notified of screen changes. For example if the screen position changes and the points move with the screen you will be notified. The notification will ask you whether you want to update the QuickCal calibration with the new 3D positions. It will also show by how much each reference point has moved.

No. of moved reference points

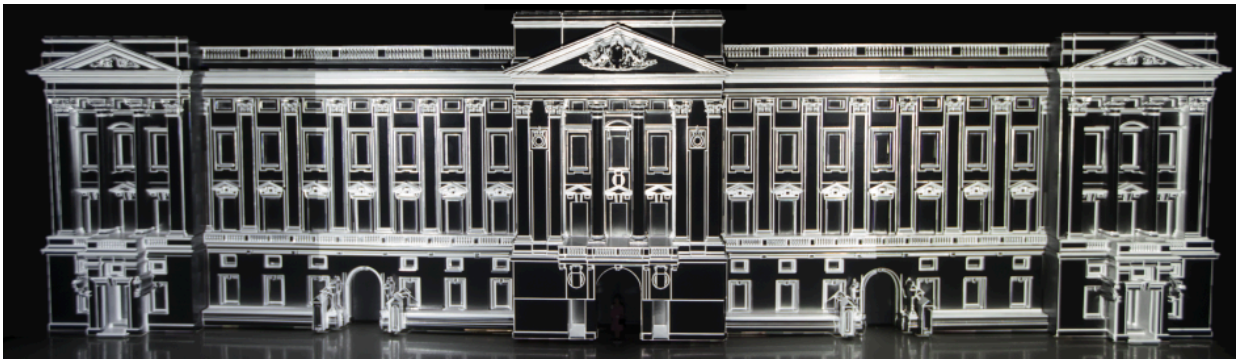
This informs you how many points have moved with a screen position change for example.

Auto z-clipping

If auto z-clipping doesn't work, this means some parts of a Screen, which are either very close or very distant, may only be partially visible. then manually adjusting these near/far values can make sure that the projector renders the whole screen.

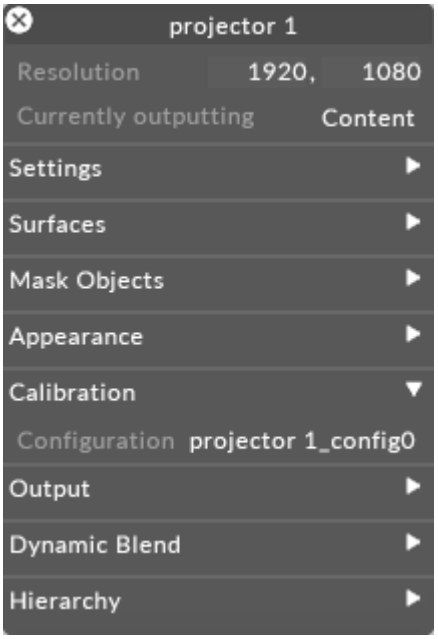
Lineup Result

Here is a real world photo of a lineup of Buckingham Palace without the **edge blending**.



Creating & deleting reference points in QuickCal

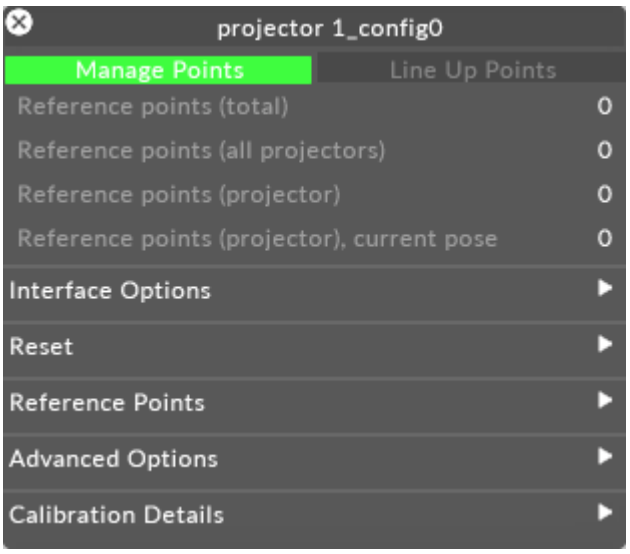
1. Open the projector editor.
2. Locate the **Calibration** tab.



Projector config editor selecting the configuration tab.

3. click the configuration file to open the QuickCal editor. If **left-clicking** you will instead open the Projector Configuration manager which stores all the configuration files for the existing projectors.

4. Select **manage points**.



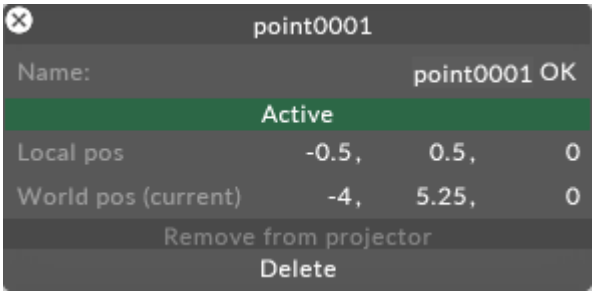
Projector config editor, Selecting manage points.

5. The d3 cursor will now turn in a circle. This indicates that you are in **manage points** mode and when you click on the model a reference point will be created at that location.
6. Create a minimum of 6 reference points and place them on **clearly** identifiable real-world reference points. This is to make it easier to link the virtual reference points to the corresponding real-world points at a later stage in the lineup process. Usually, there is no need to create more than 6-12 reference points per virtual projector but for more complex geometry it may be necessary to create more. Some complex building mapping projects have required up to 30 reference points per projector.
7. Notice that multiple projectors can use the same reference points.



To delete a reference point

1. Right-click on the point that you wish to delete.
   The options menu will open.



Delete a reference point from the virtual model. 2. Left-click on the **delete** button.

> ⚠ **Warning**
>
> Reference points that are lined up in another projector cannot be deleted (since this would affect the other projector's calibration).

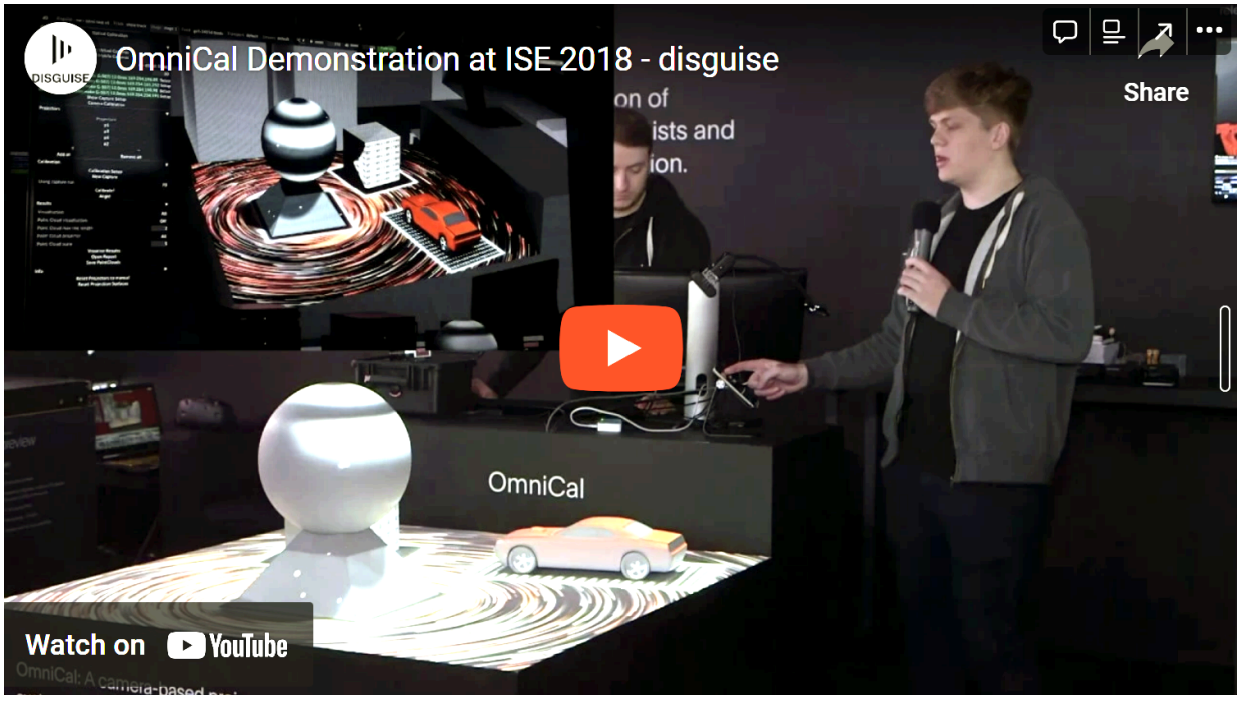
OmniCal Overview

OmniCal is an advanced camera-based system for projection mapping projects. It includes tools to plan projection setups, calibrate and align projectors to the Designer coordinate system, align projection surfaces, adjust mesh inconsistencies and deform meshes based on captured point clouds.

One of the big frustrations of our powerful 3D workflow has always been achieving accurate 3D models and calibrating those - solutions so far have been based around laser scanning or having a skilled CAD person on site who can modify the CAD model when inconsistencies are found between the 3D mesh and reality.

OmniCal removes the requirement to have accurate 3D models using its powerful calibration and Mesh Deform tools.

View the full Product information in this **Technical Specifications** document.



Features

- Robust and reliable **cameras** using Ethernet-based connections.
- OmniCal captures images of structured light patterns to construct a 3D representation of projection surfaces as a point cloud.
- It uses the point cloud to calibrate the relationship between projection surfaces, projectors and cameras.

- The QuickAlign tool helps users to manually align projection surfaces in the Designer project to match their real world positions and proportions. This is an “offline” process which doesn’t require access to the physical stage. Multiple scene objects can be aligned separately, if their relative pose is not known in advance.
- The single click Mesh Deform tool deforms a 3D model to match the real world (using the point cloud data).
- RigCheck provides easy re-calibration tools. If only projectors have moved since the previous calibration this is a single click. If cameras or projection surfaces have moved then an operator will need to adjust the previous alignment using newly captured camera images.
- The OmniCal simulation workflow allows planning with simulated cameras and lenses, view their coverage and perform test calibrations to ensure that the system and camera setup will perform as required on site.
- Supports 360° projection environments.
- Designed for calibration of surfaces and scenes with 3D depth.

> ⚠ **Note**
>
> The simulator should be used as part of the production workflow to assess project suitability.

Hardware

The OmniCal machine vision (MV) system consists of multiple Disguise **OmniCal cameras** and **suitable lenses**, with a choice of four field-of-view options.

The use of standard Ethernet technology and **Power over Ethernet (PoE)** contribute to a reliable, low-maintenance system that is ideal for fixed installations.

The OmniCal MV system is sold in kits of up to 4 or 8 cameras, or in custom arrangements, depending on project requirements. For more information, please see **OmniCal Wired Kits**.

Current limitations

- The quality of the calibration will depend on having suitable lighting conditions.
- Requires constant light levels during capture process.
- Works best with low ambient light levels.
- As mentioned above a Simulation must be run first to check project suitability. For large projects we recommend doing separate calibrations with groups of projectors and manually blending overlaps between the groups.
- Requires non-reflective, opaque projection surfaces (no gauzes or mirrors).

- Needs a few clearly defined feature points on the 3D mesh and real object, which can be visually identified on the camera images. Projection surfaces with sharp corners work well for example, but NOT smooth surfaces with no features.
- OmniCal requires depth from the projectors point of view. Scene depth is particularly important when using moving elements such as Automation.

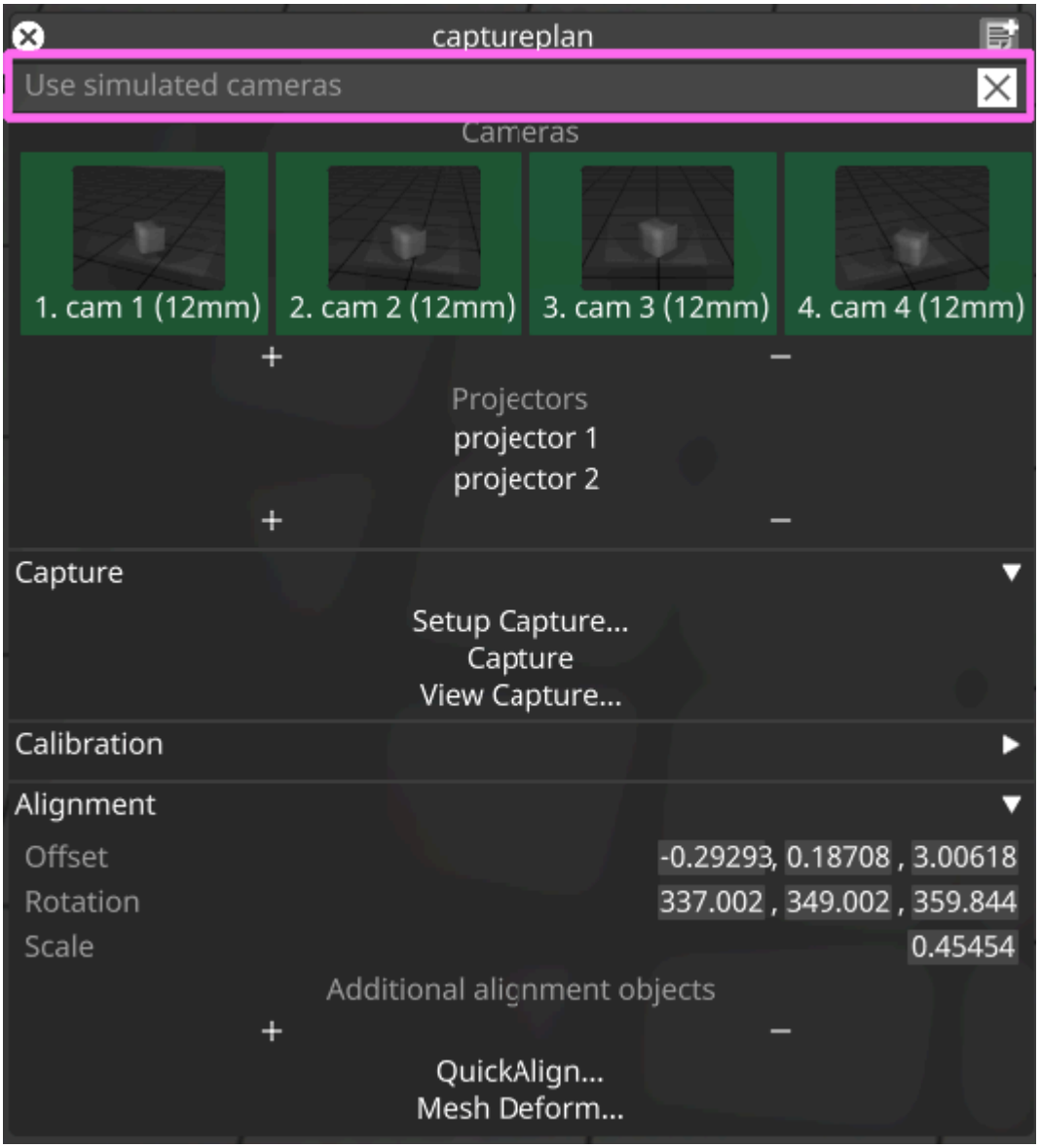
Workflow

For a trained OmniCal operator

Create a simulated Camera Plan

1. Use the Disguise simulator to check project suitability.
2. You can place virtual cameras and simulate a capture and calibration. You will need to have a project file with projectors and projection surfaces in the same configurations and positions as they will be on-site.
3. The basic rule for camera placement is that at least 2 cameras need to see every point on your projection surface(s). Also cameras should also have a large angular separation. i.e the directions they face should not be parallel.

4. The simulation will help you determine how many cameras are needed, their positions and lenses and the calibration parameters. It will also show you the ideal calibration results you should expect on-site. Note that these ideal results are without real-world influences like unsuitable lighting conditions, reflections, occlusions, movement in the scene during capture etc.

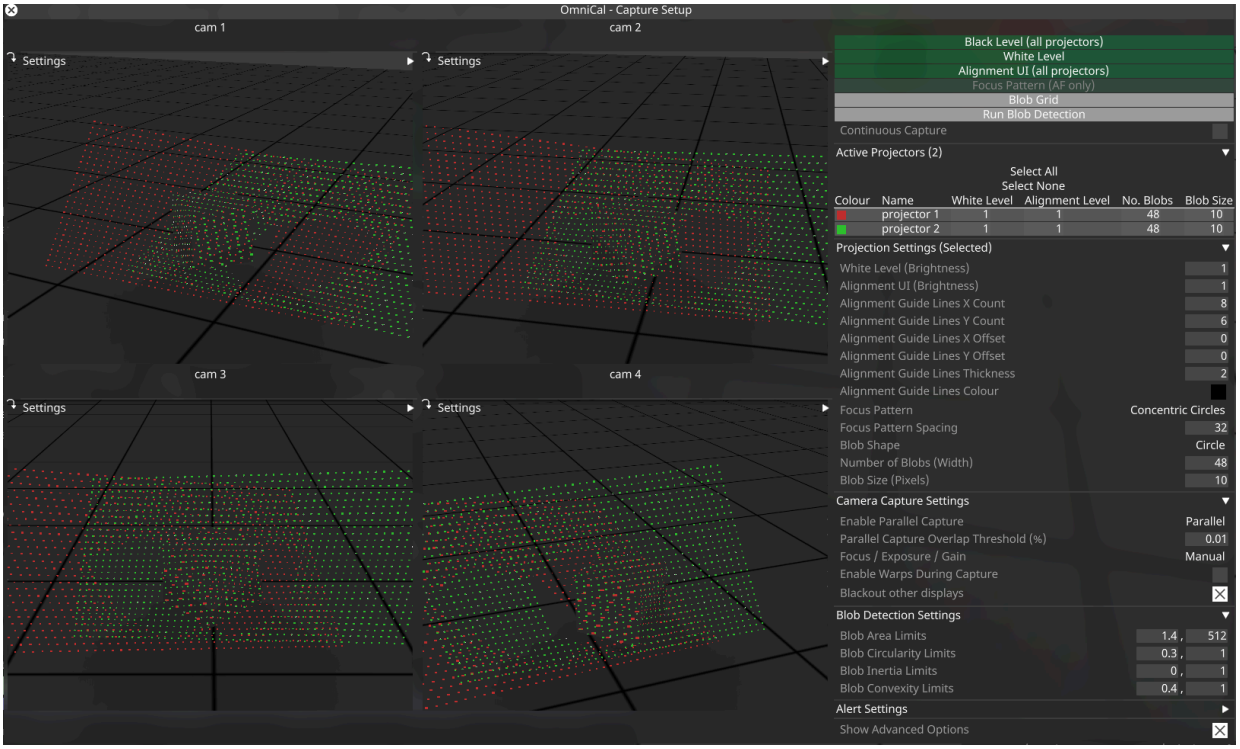


Checkbox to use simulated cameras

Setup Cameras On-site

1. You will need to make sure the position, orientation and field-of-view of your real cameras matches your simulated Camera Plan. To help you with this, there is a camera setup editor that shows what the cameras are looking at.
2. When mounting the OmniCal MV system, you need to manually adjust the physical focus and aperture (iris) on the lens, so that the images of the scene are sharp and well exposed. Exposure time can be controlled from within Disguise.

3. From the **Capture Setup** tool, check how well Blob Detection is working. Blobs are the dots that are projected in the structured light patterns.
4. You may need to adjust camera parameters (like exposure time) according to the light level to get the best results.

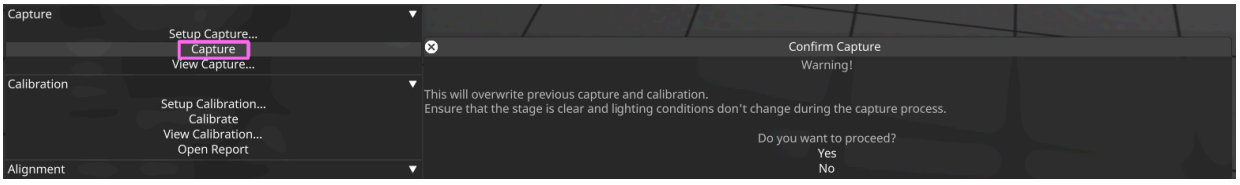


Example of Capture Setup with Blob Detection

Capture

> **Warning**
>
> A clear stage without movement or major lighting changes is required for this.

1. Capture is an automatic ‘one-button’ process that typically takes less than a minute. Exact duration depends on number of projectors, cameras and the resolution of the structured light pattern (number of blobs).
2. Once this is complete, the physical stage is free. The next steps can be done “offline”.

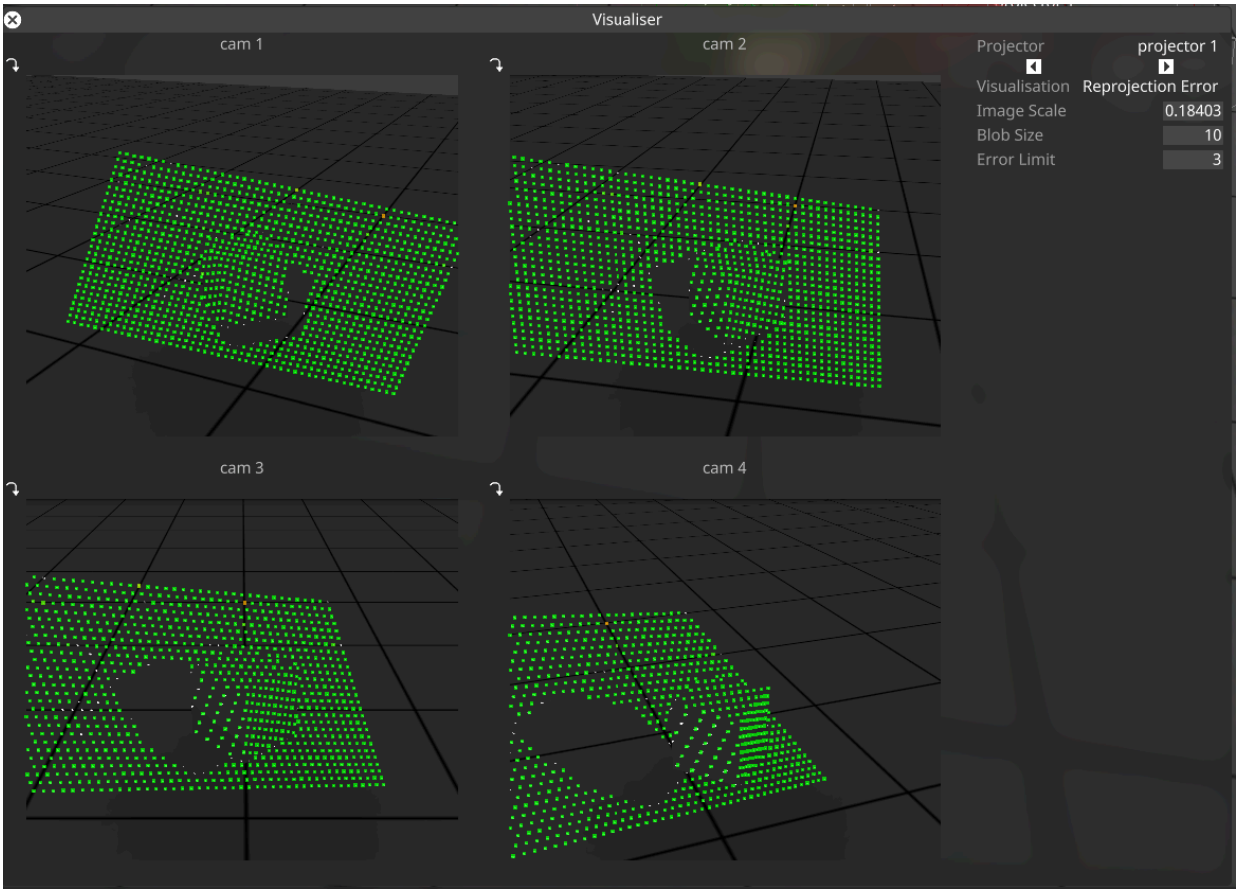


Confirm Capture Window

Calibration

1. You can view the point cloud after this stage and see check the calibration errors in pixels for each projector.

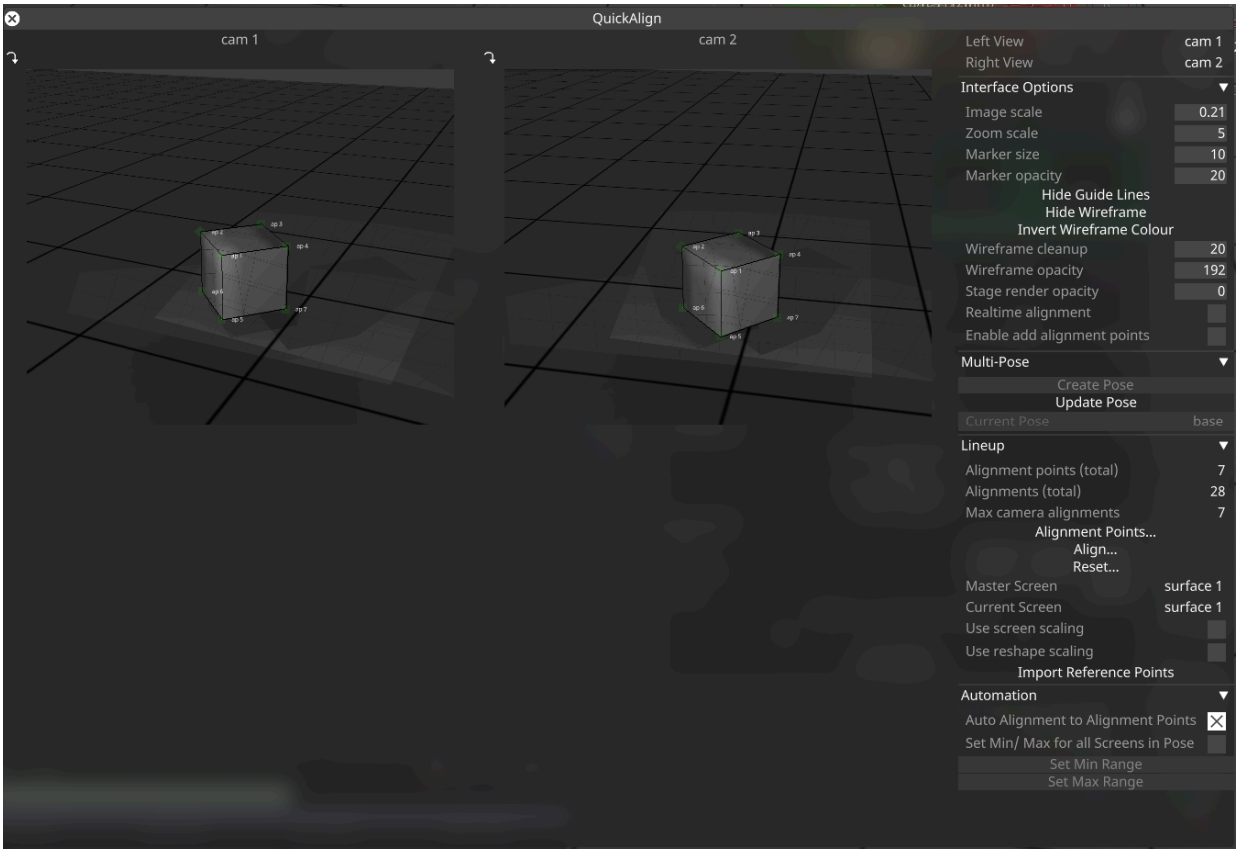
2. You may need to adjust calibration parameters to get the best results, but usually these will be chosen automatically.



Example of successful calibration in Visualiser

Alignment

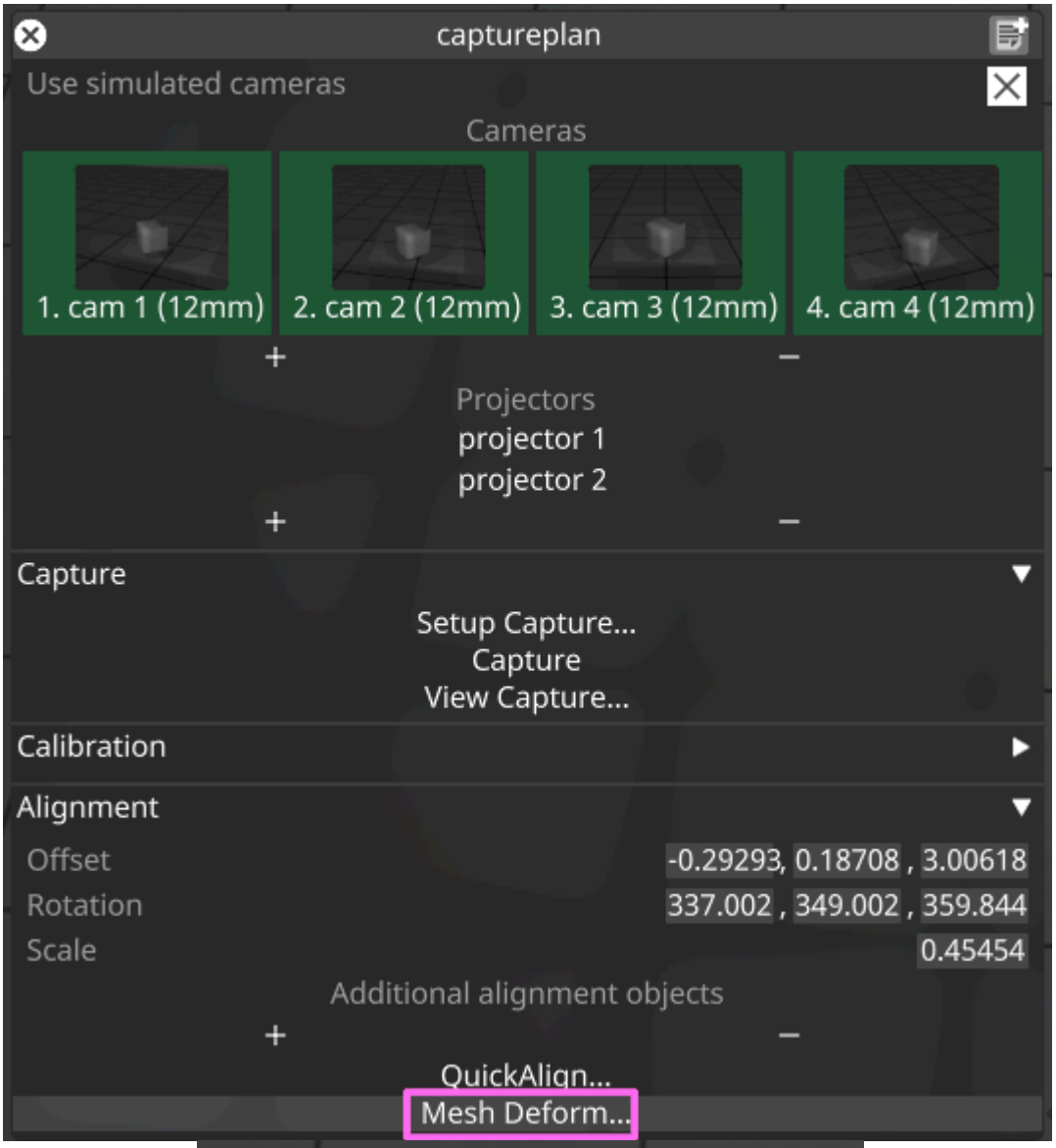
1. This is a manual step which aligns the point cloud with the projection surfaces in Disguise.
2. Users add alignment points to camera images to line up wireframe views of the projection surfaces with reality.
3. This only needs to be done once as long as cameras or projection surfaces do not move.
4. Re-shape points can also be added to correct the shape of the mesh. This can be thought of as a 3D warp from the camera's point of view.



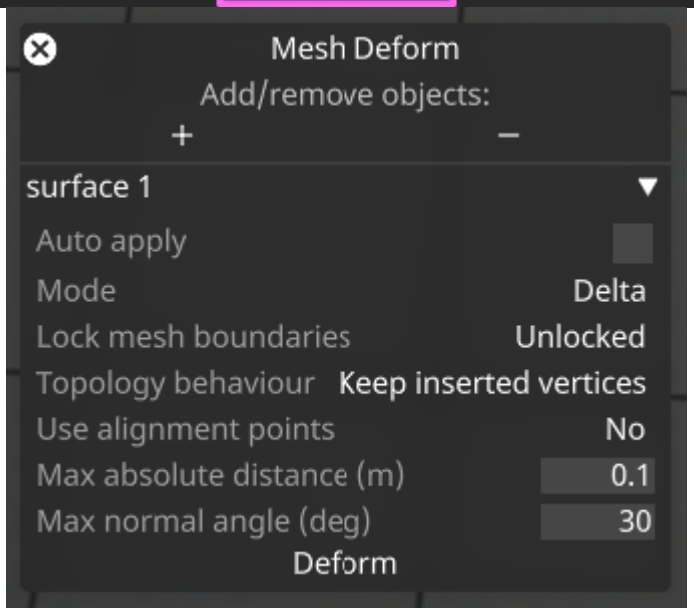
Example of successful alignment in QuickAlign Widget

Mesh Deform

This is a final key step which deforms a mesh in Designer to match the real world by using the depth information from the point cloud.



Mesh Deform Button



Mesh Deform Widget

For an untrained operator (recalibration)

Select Camera Plan

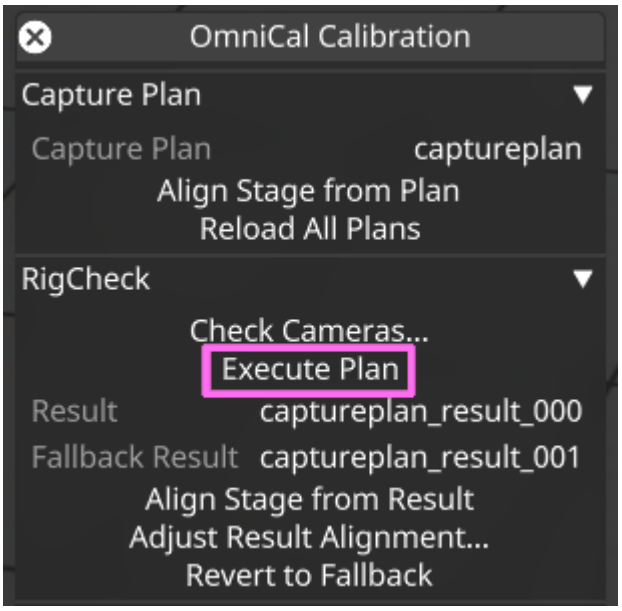
A user would select a Camera Plan previously created by a trained operator which contains known good settings for Capture, Calibration and Alignment.

Rig Check

This tool allows a user to compare live camera images to those from a previous Capture to check whether cameras or projection surfaces have moved. If so, the user can adjust the alignment reference points by dragging them into the correct positions.

Execute Plan

1. A button which triggers a new Capture and Calibration using the settings from the Camera Plan.
2. No user interaction is required after this point. Projectors will automatically be calibrated at the end of this process.



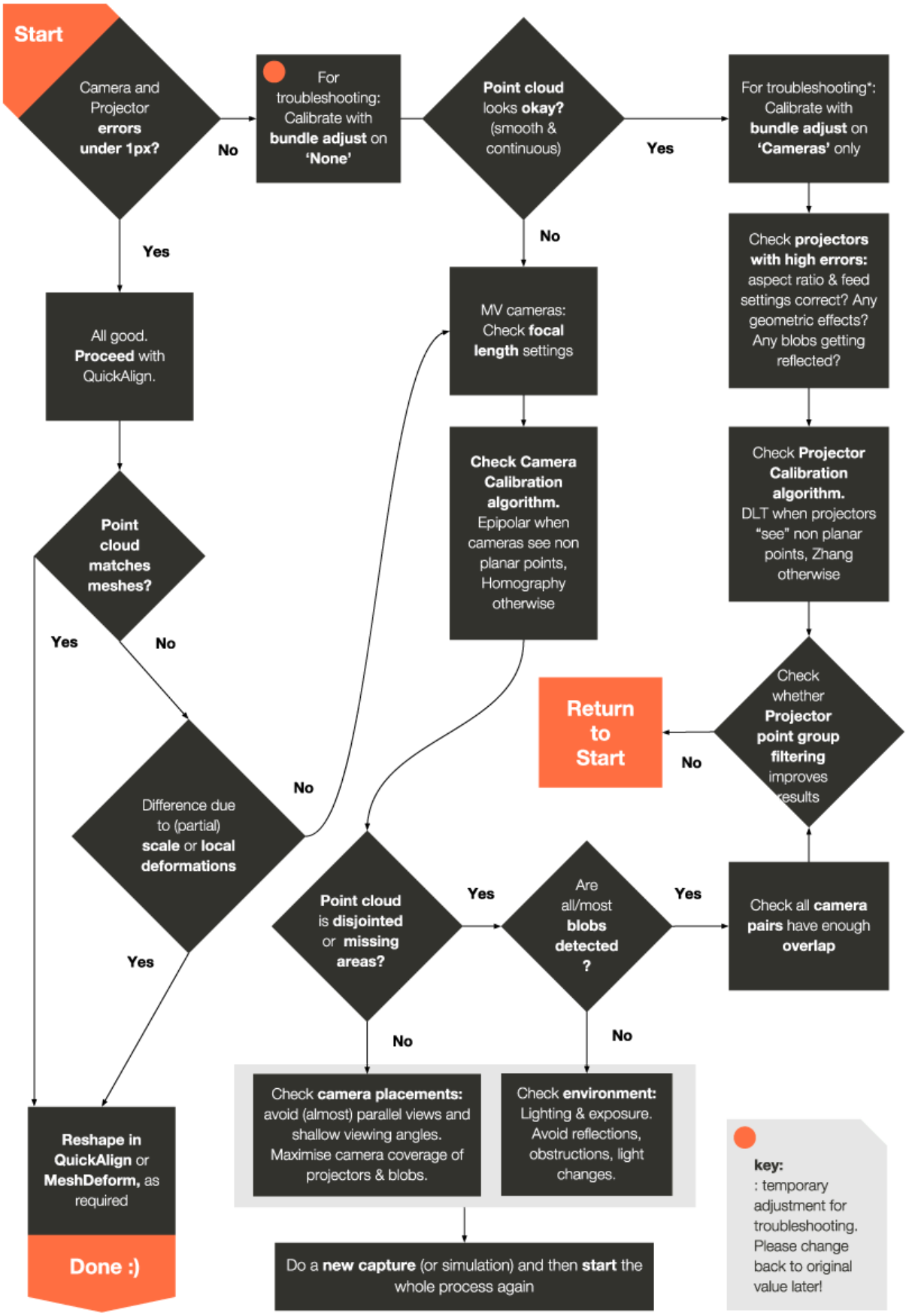
Execute Plan Button

Tips and Tricks

- The fast way to tell if you have a good calibration is to look at the reprojection scores for each projector and camera in the calibration report (this is found at the very bottom) - any score below 1 pixel is considered good, similar to the error margin that would be accepted when using QuickCal.
- Anything above 1 pixel usually indicates that something went wrong. In simulations you will normally see errors of around 0.5 pixels or below.
- Avoid reflective surfaces as they can cause issues with calibration

- Use surfaces with a lot of depth features as they make the calibration more accurate. It is especially important to have depth from each projector's "point of view". For example, if all the visible blobs from a projector land on a flat surface it will not be calibrated correctly. One way to fix this is to place an object in your environment temporarily during a capture to provide depth information.
- Each blob needs to be seen by at least 2 cameras to be used in a calibration
- Ensure blobs from across a projectors output can be seen. For example, if only blobs from the top left of a projector are detected it won't be calibrated correctly.
- Ensure a large difference in angles of attack between cameras.
- Capture Setup is important for good blob detection - you will most likely have to change the blob size, grid density and camera exposure to suit your environment.
- Blobs should be as small as possible while still being detected by cameras. This improves the calibration accuracy. Also if they are too large they won't be detected at all.
- Elongated blobs can cause higher calibration errors. Try reducing the blob size to handle this. Avoid large angles between the projector and projection surface normal e.g 45 degrees
- More blobs doesn't mean a better calibration. Usually the default of grid size of 32 is sufficient. Use more blobs if you require a detailed point cloud for Mesh Deform.
- Avoid lighting gradients, if the light level changes across an image blob detection may not work as well.
- If you a calibrating a perfectly flat surface and getting strange results, toggle the epipolar/homography camera calibration algorithm under the Plan's Calibration Setup window and see if you get a better result.
- If most blobs are landing on a flat surface this can skew the calibration results in favour of those areas. Enabling planar point removal in the Plan's Calibration Setup window may improve your results.
- It can be difficult to line up geometrically symmetrical shapes (cubes / pyramids / bowl shapes). You could embed features or identification letters and numbers into the OBJ. You can also name reference points in the QuickAlign window.
- Surfaces without corners or visible reference points such as domes and cylinders are difficult to line up.
- If you need to use mesh deform, use the point cloud visualisation mode to preview the results
- Point cloud visualization affects performance, once you have verified the validity of your calibration, turn it off to ensure good performance.

Troubleshooting



OmniCal Troubleshooting guide

OmniCal Camera Setup

> **Warning**
>
> OmniCal camera networks must use a separate network adaptor than any other network traffic (d3net, media, ...)! This is because the GigEVision protocol streams video at very high bandwidths and does not tolerate network interference.

Summary

1. GigEVision network setup.
2. Install Vimba or Vimba X for Windows.
3. Set up camera positions and configure exposure and lens settings.
4. Configure cameras in Designer.
5. Troubleshoot any network or camera issues.

OmniCal Machine Vision (MV) system

The camera setup described here must be done before you run Designer.

Network setup

> **Warning**
>
> OmniCal MV cameras will use all the available network bandwidth. Therefore, they must always remain on a dedicated physical network, separated from any other traffic. Sharing the camera switch with other devices or using network adaptor virtualisation (e.g. VLANs) cannot be supported, as any other traffic leads to dropped packets and potentially to disconnected cameras.

> **Note**
>
> - When Camera Discovery is enabled, machine vision cameras continuously capture images and stream them to the Director. This only affects the camera network and not Disguise directly; however, it consumes CPU time processing these packets. During show playback camera discovery should therefore be disabled.
> - Camera Discovery can be disabled explicitly in the **OmniCal Calibration** window and is also turned off automatically when the Calibration and Plan windows are closed.

For information about used network ports, please see **Network ports and activity**.

Network infrastructure

The OmniCal MV system is based on the **GigE Vision (R) industry standard**, which requires a bandwidth of 1 Gbit/s or higher. Camera capture speed scales with the available usable bandwidth. For example, on 10 Gbit/s setups the discovery of MV cameras and the transmission of captured images will be much faster.

The MV cameras are powered via **Power over Ethernet** (PoE), which simplifies installation by combining power and data through a single Ethernet cable. PoE must be provided either by the network switch or a PoE injector. The power requirement over PoE is quite low at 2.8W.

Ensure that all network infrastructure parts (switches, cables...) match the desired bandwidth and power specifications.

In the case of wired cables (as opposed to fibre), appropriate cables are required for reliable performance over long distances and in the presence of electromagnetic interference (EMI).

Minimum cable type for runs up to 100 meters, depending on network bandwidth:

- 1 Gb/s: Cat 6
- 10 Gb/s: Cat 6A

Multicast considerations

OmniCal does not actively make use of multicast technology. Therefore we recommend disabling it in network switches, if possible.

A GigE switch that has multicast enabled will be able to replicate traffic to multiple subscribed GigE Vision receivers. Only the first receiver will be able to control camera parameters; all others can only view image data.

In practice, multicast is irrelevant for OmniCal, because only one Director server can run the image capture process and receive the images. The Disguise recommendation is to disable multicast, unless operators require it for purposes other than OmniCal.

Disguise OmniCal cameras support multicast network configurations, when configured correctly. Refer to the **Switch Setup** section for configuration guidance.

Network adaptor settings

The following settings need to be set on the network adaptor on the Director server that connects to the switch with the cameras. You may need to update to the latest drivers to see some of these advanced options.

- Jumbo Frames: Enabled
  - Packet / frame size (MTU) should be set to 8228 bytes or the largest supported value.
    - e.g. 9000 or 9014, depending on whether the adaptor's driver includes the packet header in the count or not.
- DMA Coalescing: Disabled
  - DMA Coalescing can increase latency, so should be disabled for GigEVision use.
- Flow Control:
  - Can usually be left in the default setting.
  - In most network situations, enabling Flow Control for Rx/Tx is beneficial; however, it does not improve GigEVision performance for OmniCal cameras.
    - If the network adaptor is not exclusively used for OmniCal (e.g. when testing or demoing from a laptop) then Flow Control is best enabled.
    - If the network adaptor is exclusively used for OmniCal (i.e. on Disguise servers) then it's preferred to disable Flow Control.
    - In some edge cases it may be necessary to enable Flow Control to allow stable camera discovery, see **Troubleshooting section**.
- Interrupt Moderation Rate: Extreme
  - This reduces the number of system interrupts, which reduces CPU usage, and thus improves performance for high throughput use cases.
  - In some situations (e.g. smaller packet sizes) it may be beneficial to use a less aggressive setting such as *High* or *Adaptive*, see **Troubleshooting section**. This is because a lower interrupt moderation rate makes the system more responsive to incoming packets, which helps when large amounts of individual packets are received (as is the case with packet sizes smaller than 9 kbyte Jumbo frames).
- Transmit buffers: 256 bytes
  - The adaptor won't be sending a lot of data, so set it to 256 bytes or the closest low value that is supported by the driver.
- Receive buffers: max setting available, e.g. 4096 bytes
  - This is because the adaptor will mainly need to receive many large packets.

- Receive-Side Scaling (RSS): Enabled
  - RSS helps to better balance CPU usage for network events, so should usually be enabled.

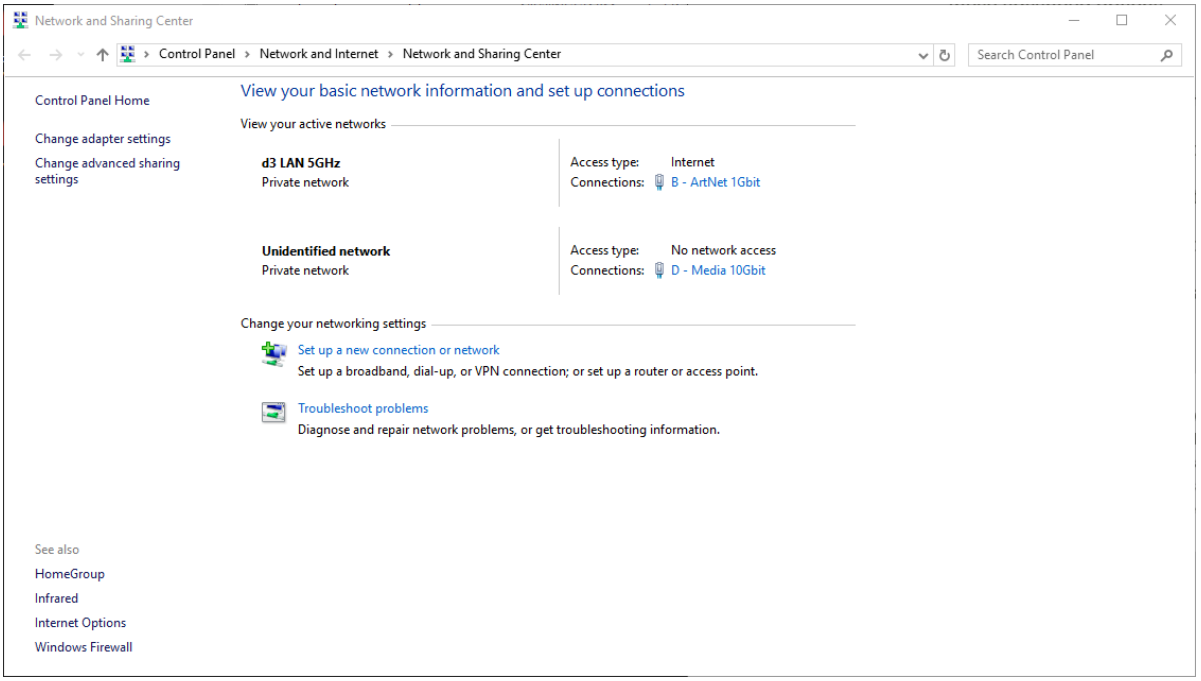
See GigE installation manual for further explanation.

Setting up the network adaptor

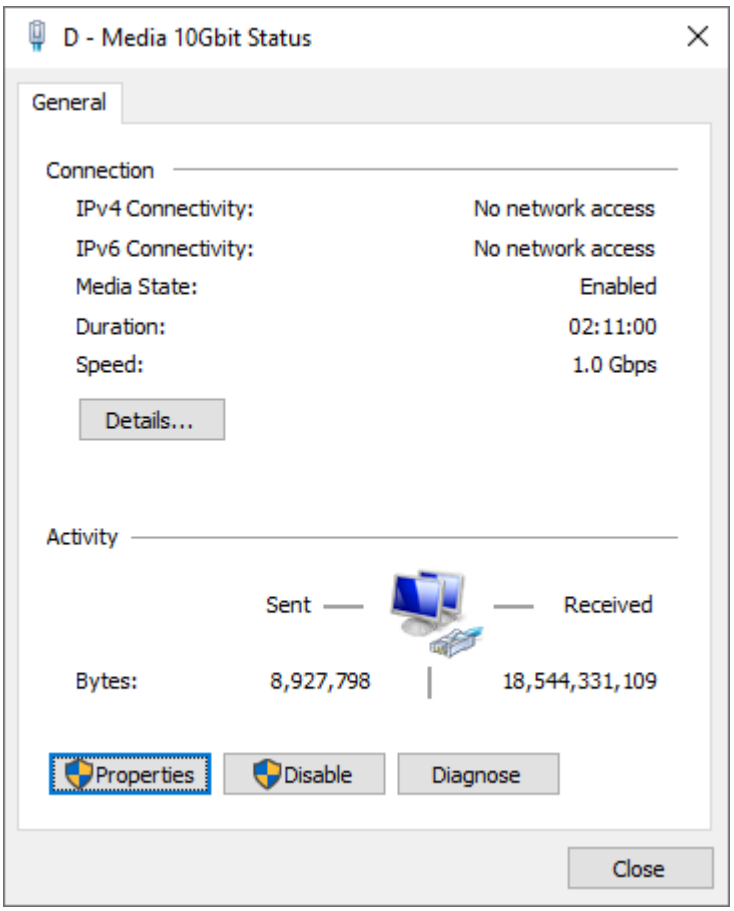
Using a 1 Gb port on the Director server for the camera communication is sufficient for small numbers of cameras. But a higher bandwidth port (e.g. 10 Gb) is preferred (even on a 1 Gb network), because their max receive buffer size is larger.

> **Note**
>
> Some network adaptors may show some of the settings under an *Advanced* button. Other adaptors may not provide some settings at all: external network adaptors such as Promise SANLink3 may only offer the Jumbo Frames setting.

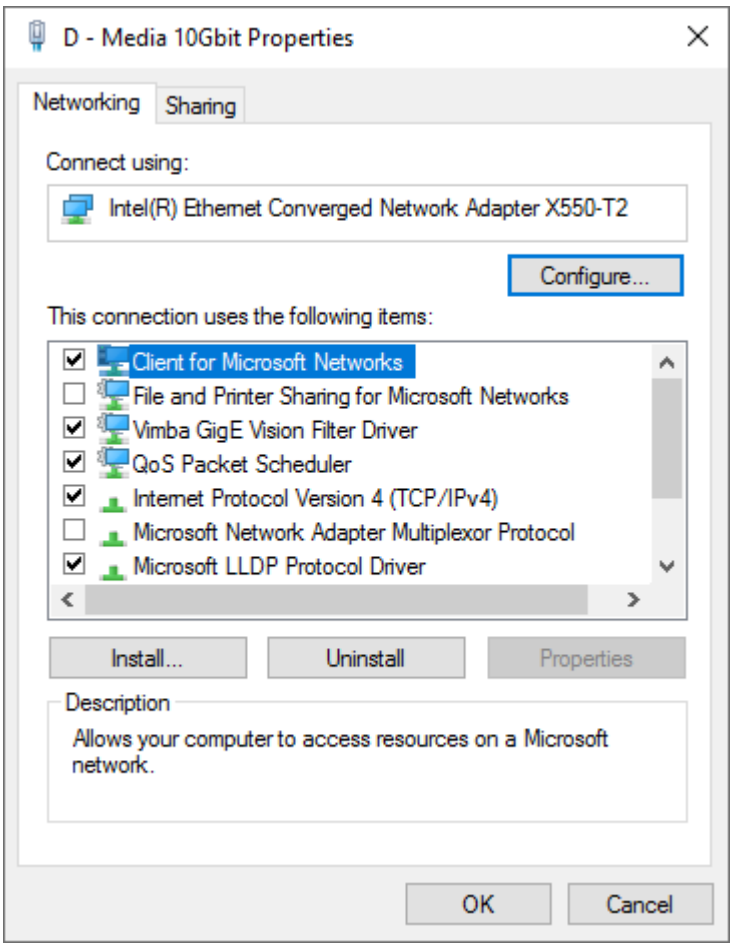
1. Navigate to the **Network and Sharing Center** and click to open the **D - Media 10Gbit adaptor**.



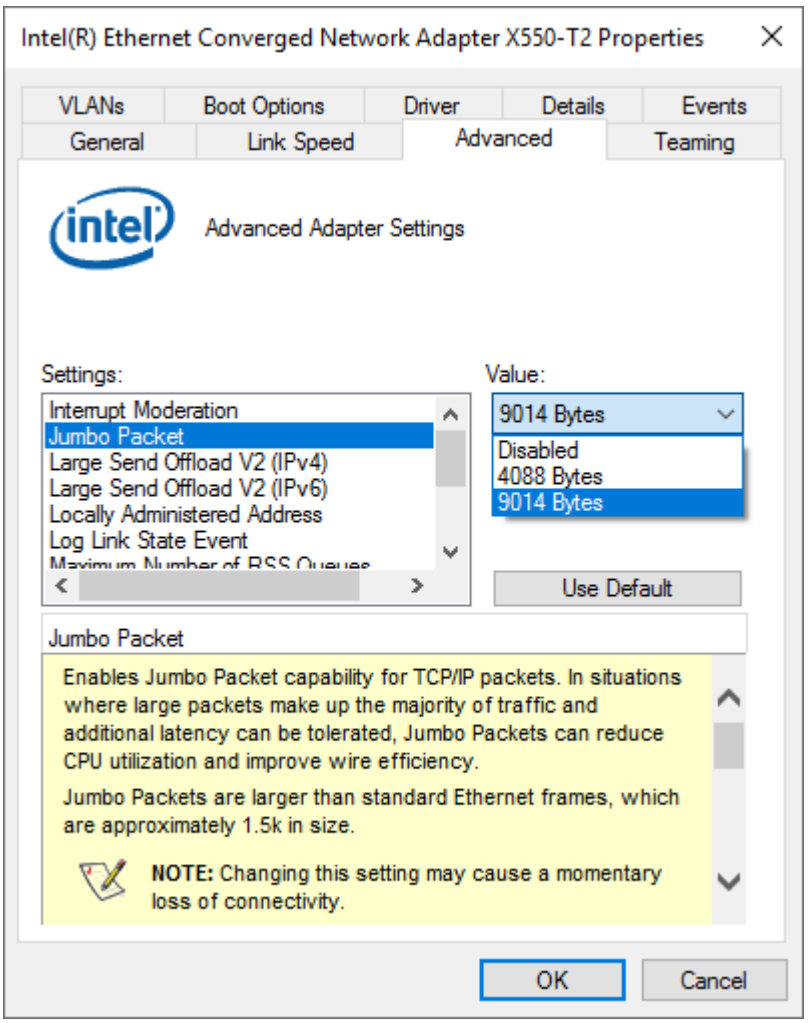
2. On the General tab, click to open **Properties**.



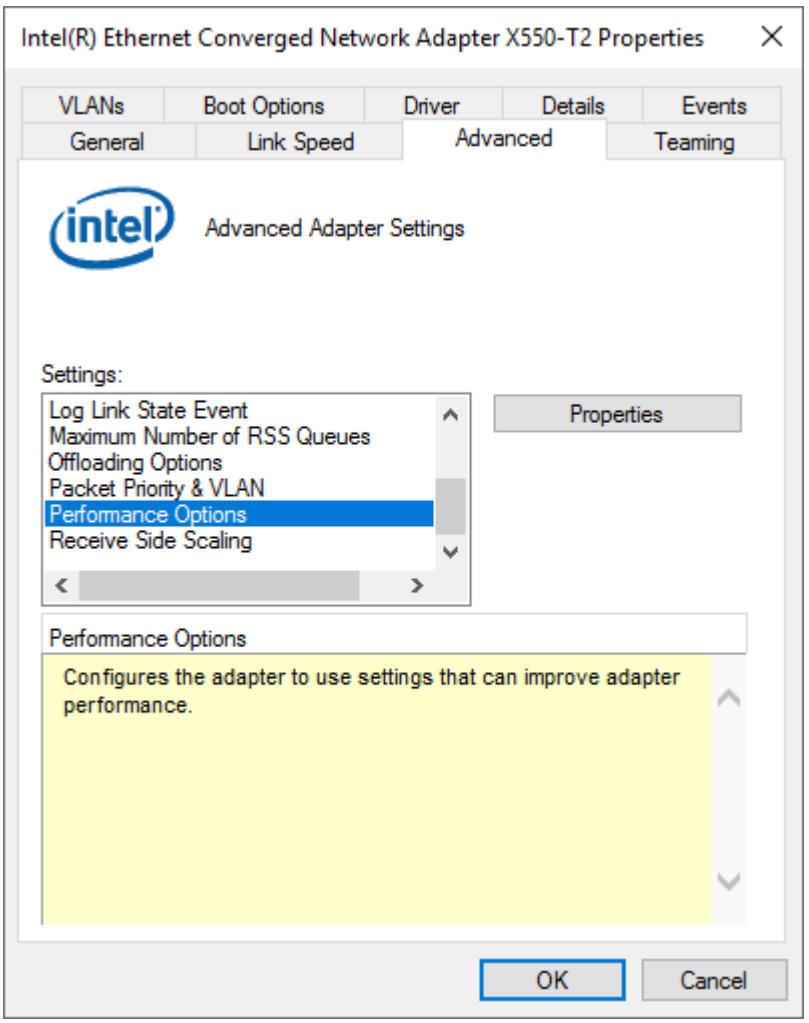
3. Enable the following settings and click **Configure.**



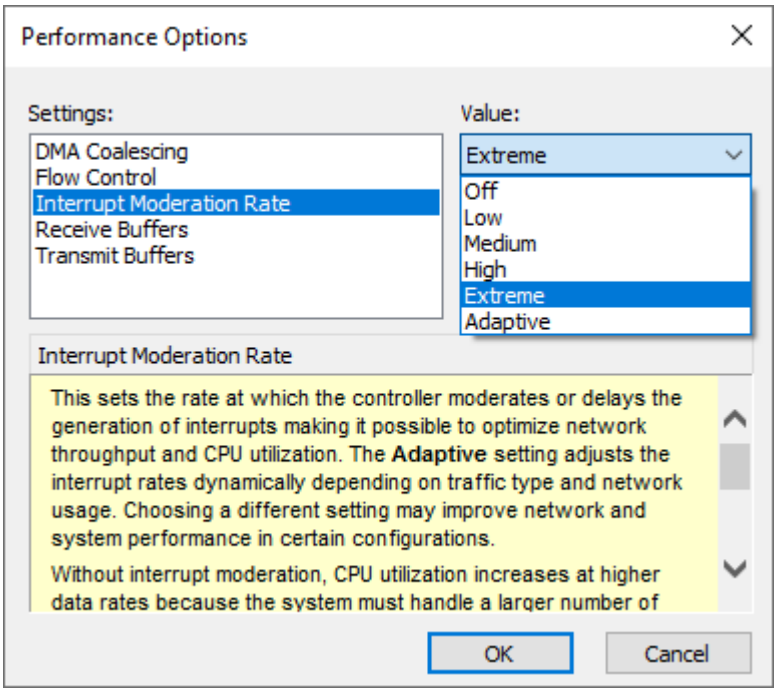
4. Select **Jumbo Packet** and assign it a value of **9014 bytes.**



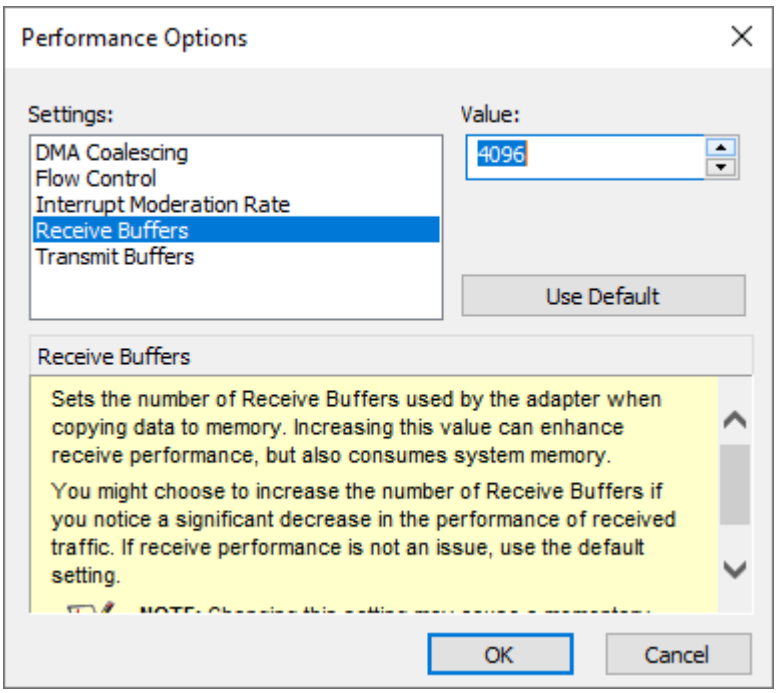
5. On the Advanced tab, select **Performance Options.**



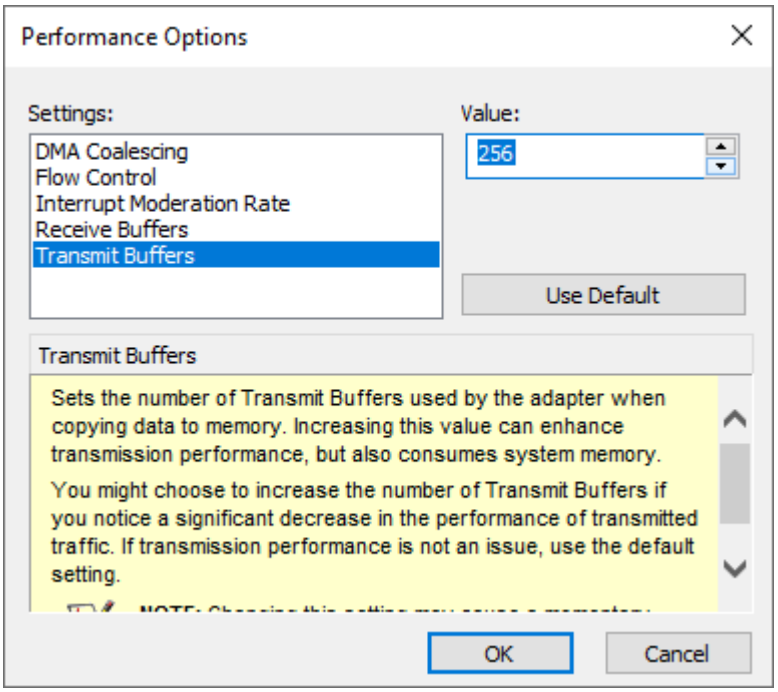
6. Set the **Interrupt Moderation Rate** to **Extreme**.



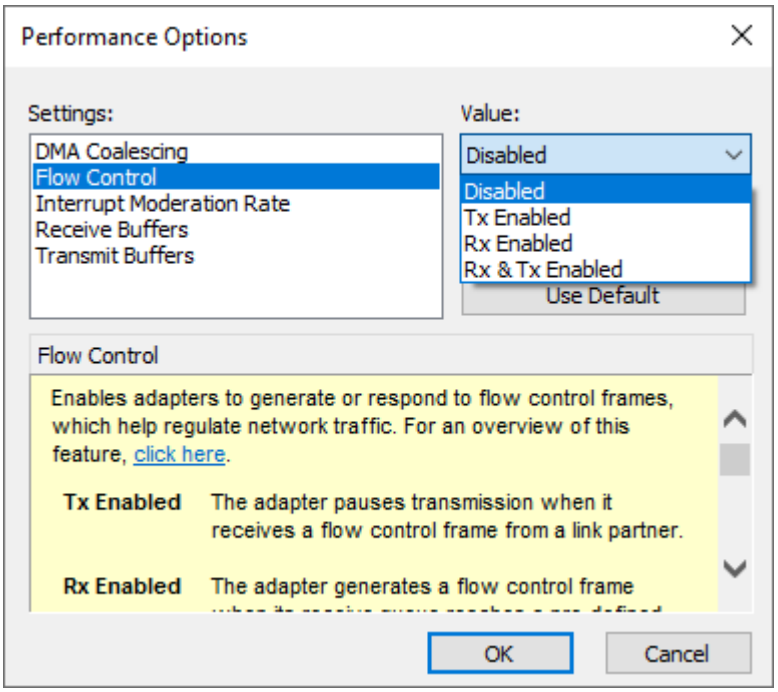
7. Select **Receive Buffers** and set it to **4096**.



8. Select **Transmit Buffers**, set it to **256** and click **OK**.



9. Select **Flow Control** and click **Disabled** or **Enabled**, as required (see Network adaptor settings).



Jumbo Packets

Jumbo Packets allow sending more data in a single packet, for example, 9 kbyte instead of the default of 1500 bytes. This reduces the number of packet headers being sent and increases the amount of actual data sent per second. Therefore, Jumbo Packets increase the overall *usable* bandwidth, and also reduce the CPU usage on the sender and receiver's side. Jumbo Packets can be configured for different packet / MTU sizes, typically around 8-9 kbyte. The value quoted by network adaptor drivers or switches can also vary, depending on whether they include any packet headers in the value or not.

Using Jumbo Packets can arguably make a network more fragile to interference, since dropped packets are larger, so resending them requires more work and data. So when using Jumbo Packets, the system should always be set up to minimise dropped packets. In Designer, you can use Plan Camera Stats to check how your system performs regarding dropped packets and frames.

GigEVision protocol requirements

GigEVision was chosen as the basis for OmniCal machine vision cameras. It is an interface standard for high-performance industrial cameras on standard low-cost IP networking infrastructure. GigEVision comprises of several UDP-based protocols that optimise for fast image transfer, but do not guarantee packet delivery to the receiver. For video data, it uses a custom packet-resend algorithm, in which packets get dropped altogether if it's not possible to get a full frame within a reasonable time. If the network is too saturated (e.g. camera bandwidth configured too high) or it experiences interference (other network devices on the

camera switch), then some packets or even whole image frames may get dropped, cameras may time out, and in some cases, cameras may even lock up and need to be rebooted.

The **Network adaptor settings** explained above are required for best performance of GigEVision cameras. In some circumstances, it may be possible to use OmniCal successfully with different settings, but this will likely affect performance and reliability, and is therefore not recommended.

Switch Setup

1. Connect a PoE network switch with a **bandwidth of 1 Gb/s or higher** to the Director.
2. **Enable jumbo frames/packets** by setting the max packet size to the highest it will go (usually around 9k, see also **above section**).
3. If possible, **disable multicast** on the switch with the MV cameras. If multicast is required for other reasons, then ensure correct configuration of related settings (see below).

> **Note**
>
> When using OmniCal with more than 6 cameras, a network switch with at least 10 Gb/s is strongly recommended. The 10 Gb switch should be connected to a 10 Gb network port on the Director.

Multicast settings for switches

If multicast is required to be enabled on the switch that connects the MV cameras to the Director, then the settings below need to be configured correctly. See **Multicast considerations** for additional information.

> **Warning**
>
> Some GigE switches only support a maximum PacketSize (MTU) of 1500 bytes in multicast mode. This breaks the Jumbo Packet requirement (see **above**). **Switches with this limitation must have multicast disabled**. See the Allied Vision **GigE installation manual** for further explanation.

IGMP Snooping

Enable *IGMP Snooping* to ensure multicast camera connections are stable and protected from excess traffic.

IGMP Snooping is a mechanism to reduce unnecessary Internet Group Management Protocol (IGMP) network traffic. It prevents hosts on a local network from receiving traffic for a multicast group they have not explicitly joined. Network switches implement *IGMP Snooping* by keeping

a map of links to multicast transmissions, so they can prune unnecessary multicast traffic from ports that are not an intended receiver.

IGMP Querier

Enable *IGMP Querier*, because isolated networks without (internet) router need a querier configured.

IGMP Version

IGMP Version **v2** is recommended for best compatibility in AV networks.

Immediate Leave

Immediate Leave should be set to **Recommended**, as it improves multicast efficiency.

Multicast Flooding

Multicast Flooding should be **Disabled**, because multicast should follow IGMP membership only.

OmniCal MV system setup (in Windows)

Install Vimba SDK and drivers for MV cameras

The *Vimba* / *Vimba X* software installs camera drivers, SDK, and the *Vimba Viewer* application used for testing and troubleshooting.

Download the correct Vimba SDK

There are two separate SDK variants from the camera manufacturer: *Vimba* (which is no longer receiving updates) and the new *Vimba X*. Designer releases since r27.7 only work with *Vimba X*. Designer notifies users if an unknown or incompatible SDK version is installed.

More recent Designer versions are optimised for newer versions of the *Vimba X* SDK, so please download and install the appropriate version:

Designer Version	Required SDK
>= r32.4	Vimba X SDK 2025-3
>= r27.7 and < r32.4	Vimba X SDK 2023-4
< r27.7	Vimba SDK

Vimba and *Vimba X* can be safely installed side-by-side on a system. This may be helpful if users need to switch between older and newer Designer releases (e.g. for rental servers). It is, however, not possible to install different versions of *Vimba X* side-by-side. Disguise recommends installing the appropriate version (see above).

> **Note**
>
> When switching between r27.6 (or earlier) and r27.7 (or later) versions of Designer, it is necessary to enable the respective *Vimba* or *Vimba X* drivers. This can be achieved via the **Vimba X Driver Installer** included in the Vimba X installation, which is much quicker than re-installing the respective SDK. This driver tool can change or disable Vimba (X) drivers for each network adaptor separately.
>
> Make sure that no cameras are in active use (streaming) before making changes to the drivers! Disabling a driver while a camera is active can lead to blue screens of death (BSOD).

Install Vimba SDK and drivers

For simplicity, the rest of this guide will just refer to “Vimba”.

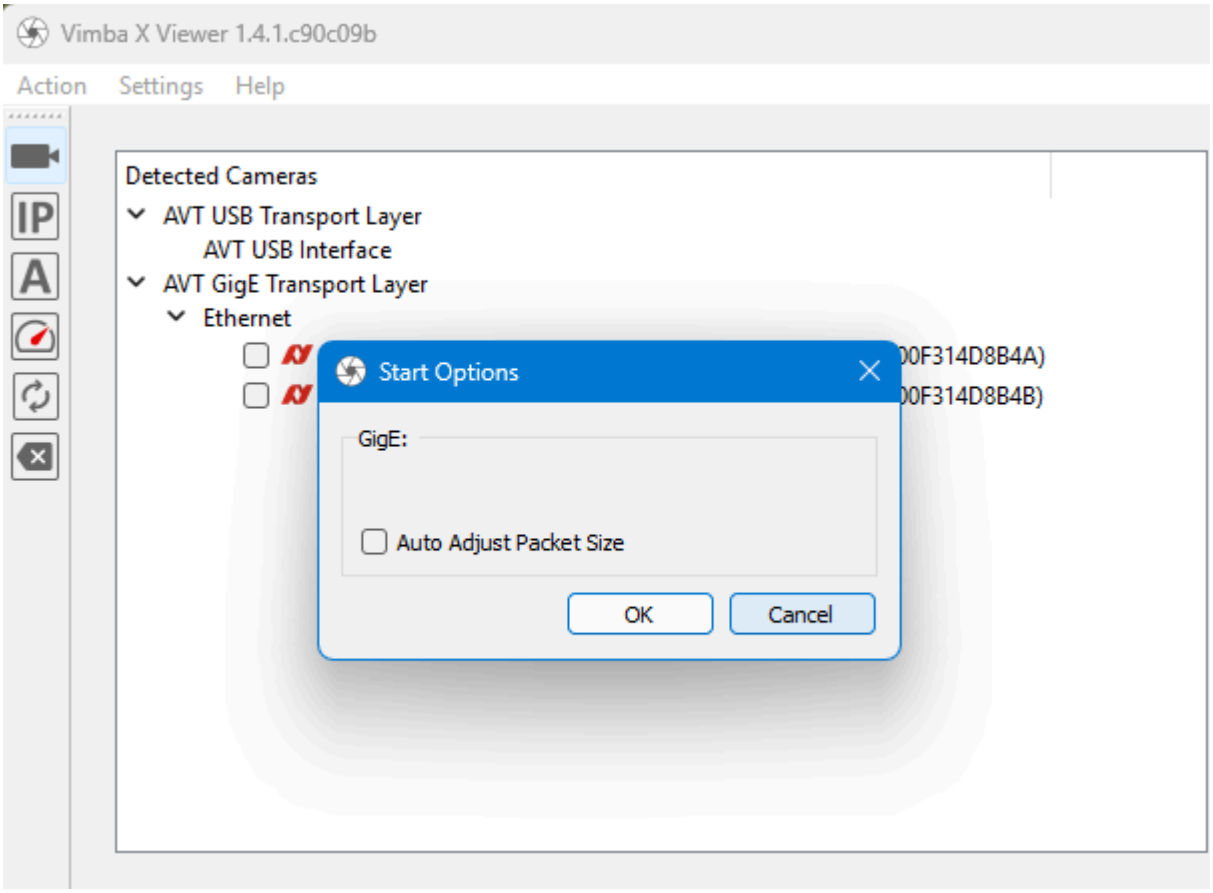
1. Disable any VPN connections before installing the Vimba drivers. They can be re-enabled afterwards.
2. Quit any existing Vimba applications, and stop any ongoing OmniCal capture processes.
3. Install Vimba or Vimba X for Windows SDK from the Disguise Download Resources.
4. In the Vimba installer, select **Application Development**.
5. Keep **install drivers** checked and complete the installation as normal.
6. Click **Start** and open **Vimba Viewer**.
7. Plug your cameras in, if they are not already.
    1. They will appear in the **Detected Cameras** list in Vimba, in white.
    2. Entries in the camera list may display a red lock icon if the camera is already opened elsewhere (see below).

Disable automatic packet size adjustment in Vimba Viewer

Vimba Viewer has some default settings that should be **disabled** after installation. If they are left enabled, then opening Vimba Viewer will interfere with how Designer configures the cameras.

Please follow these steps to disable any interfering Vimba settings:

- In Vimba Viewer, go to **Settings**, then **Start Options**.
- Disable **“Auto Adjust Packet Size”** in the GigE section, see screenshot:



Verify camera communication

1. Open Vimba and select a camera.
2. Press the play button and verify images are streaming.

In case camera connections are lost and regained (e.g. unplug/plug event), the Vimba Viewer software should detect them again automatically. In case they don't, you can press the **Refresh** button in the top-left corner.

Configure cameras in Vimba

You can right-click or double-click on the cameras to see and adjust the metadata of the camera.

This window also shows the Play button in the top left hand corner, on pressing this, the camera image should appear in this window, which can be zoomed using the mouse scroll button.

Focus, aperture, and focal length

It's easiest to adjust position, orientation, focus, and aperture (iris) while looking at the live preview in Vimba Viewer. Position and orient the cameras so they look at the object that is to

be projected onto. Ideally, the whole camera image should be taken up by the projected areas. It does not matter if the camera is mounted upside-down or right way up.

It is recommended to open the physical lens aperture (iris) as far as it can go (i.e. smallest F-Stop value). This will allow the most amount of light to enter the camera. You can then use the exposure time to control the total amount of light that hits the sensor for each frame. If the aperture is not fully open, then a longer exposure time is needed to achieve a similar overall brightness in the image. Any increased exposure time will also lead to slightly longer overall OmniCal capture times.

Adjust focus as needed by checking the image sharpness in Vimba Viewer. If the projected surface does not have many sharp visual features, then set up the projectors to output a suitable test pattern, so you can better assess the image sharpness. You may need to experiment with changing exposure time and focus settings, alternating between them, until you can clearly see a sharp image that is not too blown out (i.e. bright areas causing bloom/bleed in neighbouring camera pixels).

Note down the focal lengths used by each camera (they are written on the lens body); you will need these values later in Designer.

Brightness, Exposure Time, and other settings

Exposure time

Exposure time will heavily depend on the light levels in the calibration environment and the aperture setting of the lens. On the right-hand side, you will see a value in milliseconds that allows you to calculate roughly the FPS the camera is producing. As mentioned above, longer exposure times will lead to lower FPS and thus slower OmniCal captures.

Other parameters do not need to be adjusted. Gain may need adjusting in exceptional circumstances, but in 99% of all cases, adjusting aperture and exposure time is enough.

All

The “All” tab allows you to type in a filter pattern and search through all settings. This is useful for changing the DeviceUserID (see also below). Just type in the settings name and click Search. The DeviceUserID will be visible inside Disguise and is used to map OmniCal Plan cameras to physical cameras.

Renaming cameras in Vimba Viewer

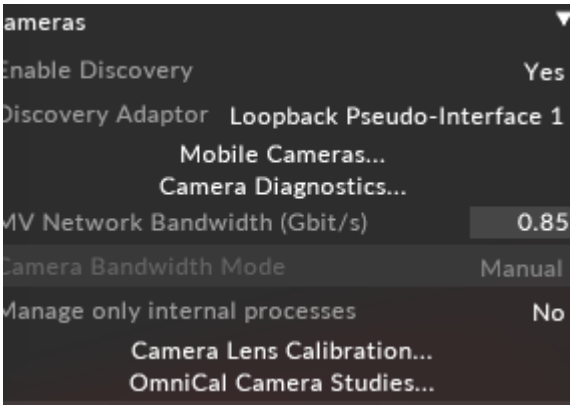
We recommended adding unique names for each camera for easier identification within Designer. Follow these steps to set this up:

1. Close Designer.
2. Open Vimba Viewer.

3. Locate the setting “DeviceUserID” (not to be confused with a similar setting labelled “DeviceID”, which cannot be changed).
4. Rename the camera as desired.
5. Press **Enter** and close this window.
6. Close Vimba Viewer when all cameras have been renamed and re-open Designer.
   - You may need to update the OmniCal Plan settings with the new camera device names.
   - Previous OmniCal captures may no longer be fully usable due to the camera name change.

Connect to cameras in Designer

There is a separate program in the Disguise software called `VimbaCamServer.exe` which is used to discover and connect MV cameras on the network. *VimbaCamServer* will find cameras on any local network adaptor, just like Vimba Viewer. *VimbaCamServer* is a separate process that communicates with Designer via a TCP-based network protocol.



*The **Cameras** collapsible widget contains network settings and camera lists for troubleshooting.*

Camera discovery

The **OmniCal Calibration** editor configures and enables camera discovery on the network. Usually, *VimbaCamServer* is run locally on the Director server, so Designer can directly communicate with it. Set the *Discovery Adaptor* to the localhost “Loopback” adaptor, which is also the default. *VimbaCamServer* is launched automatically from within Disguise if this is set, and discovery is enabled. In this typical setup, the network switch with the cameras needs to be connected directly to a dedicated network adaptor on the Director server.

> **Note**
>
> If you have many MV cameras on a network (especially if it is only 1Gbit/s), then not all cameras will be discovered immediately. They may appear one by one over the course of several seconds.

VimbaCamServer and Designer

The default "Loopback" adaptor essentially describes a local "network" connection between Designer and *VimbaCamServer*. The latter handles the actual camera GigEVision communication and captures images as PNG files for use in Designer. It communicates via the discovery adaptor with Designer, which in turn processes information and images within OmniCal. So the discovery adaptor is different to the adaptor that actually connects a machine to the camera switch.

Therefore, regarding network bandwidth, it's the network adaptor to the camera switch that matters, not the Loopback adaptor (since that is basically only limited by the PCI bus).

The *VimbaCamServer* can also be run separately, e.g. on a stand-alone computer. In that case, the *Discovery Adaptor* inside Disguise needs to be selected as the network port with which the Disguise server machine is connected to this other computer. The Director server does not need a direct connection to the cameras or the network switch they are on. In other words, the discovery adaptor needs to be set to the network adaptor that the camera server app is on. The *VimbaCamServer* can be anywhere as long as it can see and connect to the cameras, and it can communicate with the Director.

Having *VimbaCamServer* on a different machine can reduce CPU load on the Director during OmniCal captures (it won't have an impact when not capturing). However, this requires manual process management on the other machine, so it is not a recommended use-case.

Network ports used by OmniCal

VimbaCamServer and Designer communicate with each other using different network protocols and ports:

1. An OSC device `omnicaldiscovery` that broadcasts from the Director, to discover the VimbaCamServer process. These OSC broadcasts are sent every 30 frames, unless OmniCal capture is in progress.
   By default, this OSC device uses port 9001, but this can be changed via the Device Manager.
   Note that the `omnicaldiscovery` OSC device is only visible in the current Device Manager while OmniCal **camera discovery** is enabled.
2. A proprietary TCP-based protocol to send image requests, etc., and reply with ACKs and actual image data.
   This protocol uses ports 10101 and 10102 for sending from and receiving to the Director. These ports cannot currently be changed.

List of discovered cameras

The **Discovered Cameras** button opens a list of OmniCal cameras that have been discovered by Designer. The background colour for each camera row will indicate the state of the camera:

- **Green**: This means cameras are ok and receiving data from *VimbaCamServer* (with respect to the stats measuring incomplete frames between each receipt of stats).
- **Red**: This means there have been incomplete frames between the last read and the current read. The count of incomplete frames will continue to increase. The red colour will reset if there have been no incomplete frames between each read of the stats.
- **Grey**: This indicates the camera is disabled, e.g. due to incompatibility or licensing restrictions.
- **Brown**: This indicates the camera is disconnected / offline.

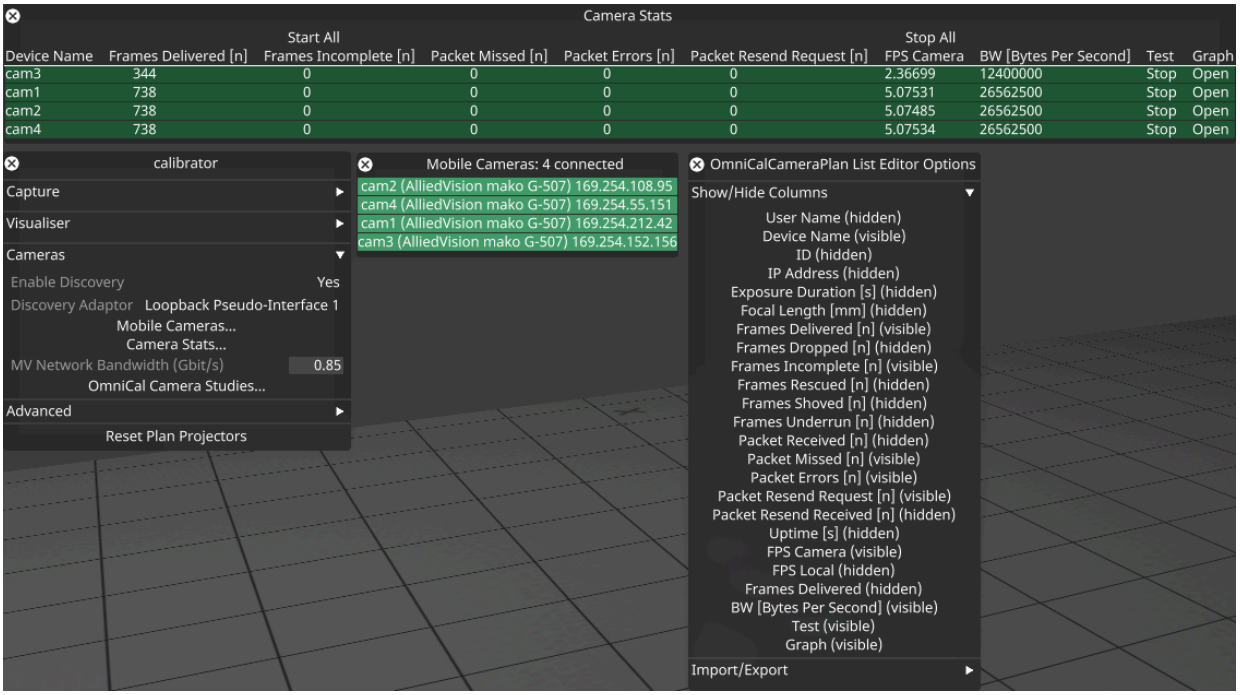
Camera statistics

Plan Camera Stats overview

The **Plan Camera Stats** button opens a window with a list of OmniCal cameras that are in the current Plan. It offers tools to test and troubleshoot cameras and the network by starting camera streaming without capturing images to disk. The window contains a table with rows for every camera, which collect various statistics like captured and dropped packets / frames or the camera temperature. Right-click the title bar to configure which kind of statistics should be shown in the table.

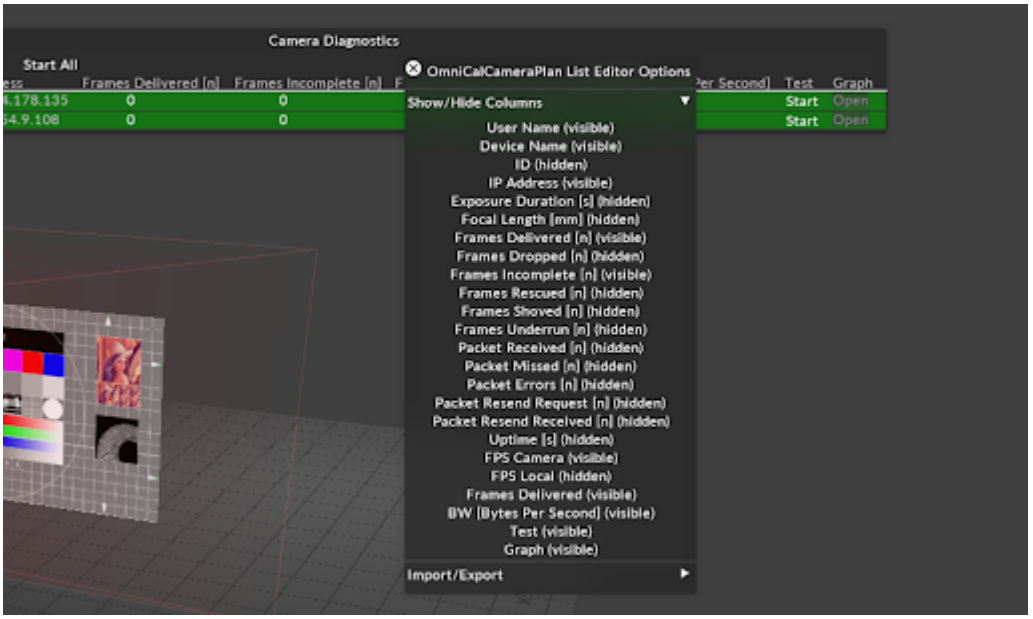
It is possible to test streaming for all cameras in the current Plan or only for a few selected cameras. This can help to identify whether an issue is specific to a certain camera or cable, etc.

A certain amount of dropped packets can be tolerated in a properly set up camera network, but dropping entire frames should be avoided. These stats can be used to evaluate the impact of changes to e.g. network settings or the usable bandwidth (see **next section**). Checking the impact on dropped packets/frames here is much quicker than trying out a full OmniCal capture, which might time out half-way through.



How to use Plan Camera Stats

1. Make sure to assign the physical OmniCal cameras to the cameras in the OmniCal Plan. If you see **Plan Camera Stats** greyed out, then you have not finished this step, or no physical cameras have been discovered.
2. The columns display various camera stats and feedback of camera properties (such as IP, name).
3. Right-click the widget header to show available columns and enable or disable them:



4. Click **Start** to enable streaming and stats feedback for a single camera. Click **Stop** to turn the stream off again.
5. Click **Start All** to enable streaming and stats feedback for all cameras.

6. The **Open** button in the **Graph** column opens a graph window that shows how the stats change over time.

> **Note**
>
> The **Plan Camera Stats** window only displays cameras that are used in the currently active Plan.

Plan Camera Stats columns

The values in each **Plan Camera Stats** table row will update while a camera is streaming. Starting to stream will reset the values for each camera that is streaming.

Some of the collected statistics are specific to GigEVision network protocols. This will be indicated with a "[GigE]" tag at the end of the description. For additional information about those stats, please visit the Allied Vision GigE features reference (chapter "Statistics", page 163).

Meaning of camera statistics columns

User Name

The user-specified name given to the associated camera in the OmniCal Plan.

Device Name

The user-specified name of the OmniCal camera, as set in Vimba Viewer.

ID

The unique hardware ID of the OmniCal camera.

IP Address

The current IP address of the OmniCal camera.

Exposure Duration [s]

The exposure time in seconds. Smaller values can lead to faster capture, but also darker images.

Focal Length [mm]

The camera lens focal length in millimetres, as set in the OmniCal Plan.

Frames Delivered

The number of complete, error-free frames that were successfully delivered to the host. [GigE]

Frames Dropped

The number of incomplete frames received by the host due to missing packets (excluding shoved frames). [GigE]

Frames Incomplete

Frames incomplete provides some feedback about the stability of the camera connections in *VimbaCamServer*. The value may be larger than **Frames Dropped**.

It measures how many video frames were not completed for any reason, e.g.:

- not all packets were received, despite resend attempts (packet loss)
- frame is now obsolete (see **Frames Shoved**)
- failed to validate the frame data
- exceptions when handling frame data from Vimba API

See Troubleshooting below for information on how to resolve incomplete frames and other network issues.

Frames Rescued

The number of frames that initially had missing packets but were successfully completed after packet resend. [GigE]

Frames Shoved

The number of frames dropped because the transfer of a following frame was completed earlier, making the previous frame obsolete. [GigE]

Frames Underrun

The number of frames missed due to the non-availability of a user-supplied buffer. [GigE]

This means the Director was not ready to copy a completed frame from the camera. Values above zero may indicate issues with server system load, e.g. the CPU is busy with other work, or threading / priority issues. On machines with few CPU cores (e.g. 4) this may be resolved by adjusting advanced machine settings such as `omniCalMaxThreadsPng`.

Packet Received

The number of error-free packets received by the driver. This number should grow steadily during continuous acquisition. [GigE]

Packet Missed

The number of packets missed due to network issues. [GigE]

If everything is configured correctly, this number should remain zero, or at least very low compared to the **Packet Received** value.

Packet Errors

The number of improperly formed packets. If this number is not zero, it suggests a possible cable or camera hardware failure. [GigE]

Packet Resend Request

The number of missing packets that were requested to be resent from the camera. [GigE]

If everything is configured correctly, this number should remain zero, or at least very low compared to the **Packet Received** value.

Packet Resend Received

The number of packets that were actually resent by the camera. [GigE]

Uptime [s]

The elapsed time in seconds since the camera stream was started. [GigE]

FPS Camera

The rate at which the camera is acquiring frames, derived from the timestamps of completed frames. [GigE]

FPS Local

Inverse of time interval between the last two frames (faulty or not) received by the host. No averaging is performed. [GigE]

In case of error-free reception, **FPS Local** is similar to **FPS Camera**, except that the host (system) clock is used instead of frame timestamps for measuring the time interval between frames. In case of errors, the two values may differ significantly, since **FPS Local** includes incomplete frames.

Frames Delivered Delta

Difference in **Frames Delivered** between the most recently received camera stats and the previous value.

BW [Bytes Per Second]

The network bandwidth in bytes/second that is allocated to this camera. [GigE]

This is an even share of the total MV Network Bandwidth configured in Designer.

GevSCP Packet Size [1]

Packet size (MTU) in bytes for camera device (GigE Vision SCPS protocol). [GigE]

This value should be the same as the Jumbo Packet setting configured for the network adaptor. If it is lower (e.g. the default 1500 byte), then this indicates that Jumbo Packets are not

enabled across the whole signal chain.

GVSP Packet Size [1]

Packet size (MTU) in bytes for camera stream (GigE Vision Stream Protocol GVSP). [GigE]

This value should be identical to **GevSCP Packet Size**.

Mainboard Temperature [C] [1]

Temperature of the camera main board in degrees Celsius. Accuracy: +/- 1 °C [GigE]

Camera Bandwidth

The field **MV Network Bandwidth (Gbit/s)** configures the total amount of usable bandwidth for OmniCal MV cameras. The network needs to reliably provide this bandwidth. Typically, this should be set to a value about 80-90% of the theoretical bandwidth of the link between the camera switch and the server running *VimbaCamServer* (e.g. the Director). The default value is 0.85 Gbit/s, which should work for most 1G network setups. On a true 10G link to the Director a value of up to 8.5 Gbit/s should work.

If the network cannot reliably provide this usable bandwidth (e.g. due to traffic or interference from other devices), it will lead to dropped packets and potentially disconnected cameras.

In this case it may help to reduce the **MV Network Bandwidth** value. Reducing the total bandwidth value will further slow down the capture, but allow for more room for packet re-sends, to avoid dropped frames. The appropriate bandwidth value depends on the quality of the network setup and the nature of any interference.

OmniCal camera specs and bandwidth

OmniCal MV cameras “Disguise G-507B” have a 5.1 MegaPixel image sensor with 2464x2056 pixels and capture 8-bit monochrome images. The size of a single image is 5065984 bytes, and the respective bandwidth would be 40.5 MBit/s. At their maximum framerate of 23 FPS, a single camera would require 932.1 MBit/s bandwidth, which is roughly above the effective capacity of a 1G Ethernet connection.

In OmniCal, the typical frame rate of each camera is much lower than 23 FPS, partly because they are mostly used in dark environments, where longer exposure times are needed. Exposure times have a direct effect on the maximum frame rate that can be used. Assuming a frame rate of e.g. 10 FPS, a single camera would need a bandwidth of about 405 MBit/s, which is almost half of the effective bandwidth of a 1G Ethernet connection.

Since OmniCal is typically used with more than 2 cameras, Designer needs to carefully manage the usable bandwidth of every camera, so the total camera bandwidth does not exceed the available bandwidth of the network connection. Note that this also has a further impact on the maximum frame rate that can be achieved: more cameras means a lower maximum framerate (assuming network bandwidth stays the same).

Bandwidth management in Designer

Every time an OmniCal capture is started, Designer (or rather its sub-process *VimbaCamServer*) sets the bandwidth for each camera explicitly. Every camera gets an equal share of the total bandwidth set in Designer. For example, when using 4 cameras and 0.8 Gbit/s, every camera will be configured to use 0.2 Gbit/s.

This will overwrite any bandwidth setting configured externally, e.g. in Vimba Viewer.

The bandwidth also has an impact on the practical FPS and capture speed for each camera. The more cameras are connected, the longer each camera will need to stream a whole frame. Therefore, with equal bandwidth, more cameras result in reduced maximum FPS and longer capture time.

Bandwidth management in Vimba Viewer

Any automatic bandwidth or packet size adjustment in Vimba Viewer should be **disabled**, because it will interfere with how Designer configures the cameras.

> **Note**
>
> Some problematic settings are enabled by a default installation of Vimba X. Please see here on how to **disable automatic packet size adjustment in Vimba Viewer**.

Troubleshooting

Always verify first that all network settings are as required for **switches** and **network adaptors**.

Camera has red lock icon in Vimba Viewer

This indicates that the camera is already opened elsewhere, for example, another laptop running Vimba Viewer or Disguise Designer.

Camera access is exclusive; if you have a camera opened for capture in Designer, you will not be able to view it in Vimba and vice versa.

No image is displayed in Vimba Viewer

On older servers, it may be worth trying to disable Jumbo Frames on the network adaptor that connects to the camera switch, or at least configuring them to a lower size. This has been observed in the past with legacy 4x4 pro servers, where Jumbo Frames with an MTU size over 2034 bytes was unable to get complete images from the cameras, due to packet loss. An alternative workaround would be to limit the packet size on the switch.

Use Vimba Viewer to verify the GVSP packet size setting is 2034 or below. This value is not set directly, but negotiated automatically between network adaptor, switch and cameras.

Network adaptor gets disabled in Windows after applying suggested settings

Try reverting the Interrupt Moderation Rate to the default value, which is *Adaptive* on most Intel adaptors. It may be necessary to re-enable the adaptor manually in Windows Advanced Network Settings.

Cameras drop out shortly after capture starts

If cameras work individually in Vimba Viewer but not when capturing multiple cameras with OmniCal, this could indicate an incorrect multicast configuration. Check whether multicast is enabled on the switch and either disable it or ensure *IGMP Snooping* is enabled. For more information, see the sections on **multicast** and **switch setup**.

If this does not help, please follow the steps below.

Very slow image capture, dropped packets, incomplete frames, or cameras become unresponsive

1. Ensure that Vimba Viewer does not overwrite the packet size set by Designer, see **instructions here**.
2. Verify that the cameras are communicating using **Jumbo Frames**.
    - The reported GVSP packet size in Vimba Viewer should match the values you set for the network adaptor and switch (e.g. 9000 bytes).
    - You can also check the GVSP value in the `console_vimbacamserver.txt` log file.
    - A value of 1500 indicates that Jumbo Frames were not enabled somewhere in the signal chain.
    - Any value between 1500 and 8000 indicates that some networking equipment in the signal chain is configured for smaller Jumbo Frames than desired. If this cannot be addressed, it may be necessary to reduce the total usable bandwidth in Disguise Designer (see **Camera Bandwidth**).
    - Any network switch that does not support Jumbo Frames must be removed from the signal chain.

3. Use the **Plan Camera Stats window** to get more information about the network behaviour.
   - Test whether any network issues are specific to a certain camera or cable by streaming them individually.
     - If issues like dropped packets only occur with a certain camera, then try swapping cables and check other network equipment in that camera's signal path.
   - Add more cameras one-by-one to test whether problems only start after using a certain number of cameras.
     - If this is the case, it may be necessary to reduce the total usable bandwidth in Disguise Designer (see **Camera Bandwidth**).
     - Reducing the bandwidth will lead to slower image capture. But in a congested network this may still improve the capture performance, as it will lead to fewer dropped packets that then don't need to be re-sent.
4. Ensure that the camera network is clean from any other interference.
   - Verify that OSC devices in Designer (including the `omnicaldiscovery` OSC device) do not clash on their port numbers. See **Network ports used by OmniCal** for more information.
   - Verify that the camera network is physically separate, with no other devices on it.
   - Use Wireshark to monitor network traffic on the OmniCal network adaptor to spot any devices sending packets on this network. Devices such as lighting desks might flood the network with messages, or other machines might incorrectly use this network for copying data.
5. As mentioned in **Network setup**, VLANs and adaptor virtualisation are not officially supported, as they make it very difficult to guarantee the required bandwidth for the OmniCal cameras.
   - If a dedicated physical network is not available, then packets will be dropped whenever there is other traffic, which can lead to camera timeouts during capture.
   - The likelihood of this happening can be reduced by reducing the total usable bandwidth in Disguise Designer (see **Camera Bandwidth**).
6. Try reverting the *Interrupt Moderation Rate* to the default value, which is *Adaptive* on most Intel adaptors.
   - This usually helps in situations where the MTU / packet size is smaller than the recommended value of 8-9 kbytes (see **Interrupt Moderation Rate** and **Jumbo Frames / Packets**).
7. Try changing the *Flow Control* setting from *Enabled* to *Disabled*, or vice versa (see **Flow Control**).

Restarting unresponsive cameras

The easiest way to restart a camera is to disconnect and reconnect it from the PoE switch. For managed switches, this can usually be done without physically touching the cables, via a

configuration GUI (e.g. browser interface) or even API commands.

Further Reading

GigE Vision (R) industry standard

Allied Vision GigE installation manual

Allied Vision GigE features reference

GeniCam Network Card Performance Settings

Advanced Driver Settings for Intel® Ethernet 10 Gigabit Server adaptors

Microsoft introduction to Receive Side Scaling

OmniCal Capture

Capture is the process of projecting structured light patterns, taking images of these and detecting blobs within these images.

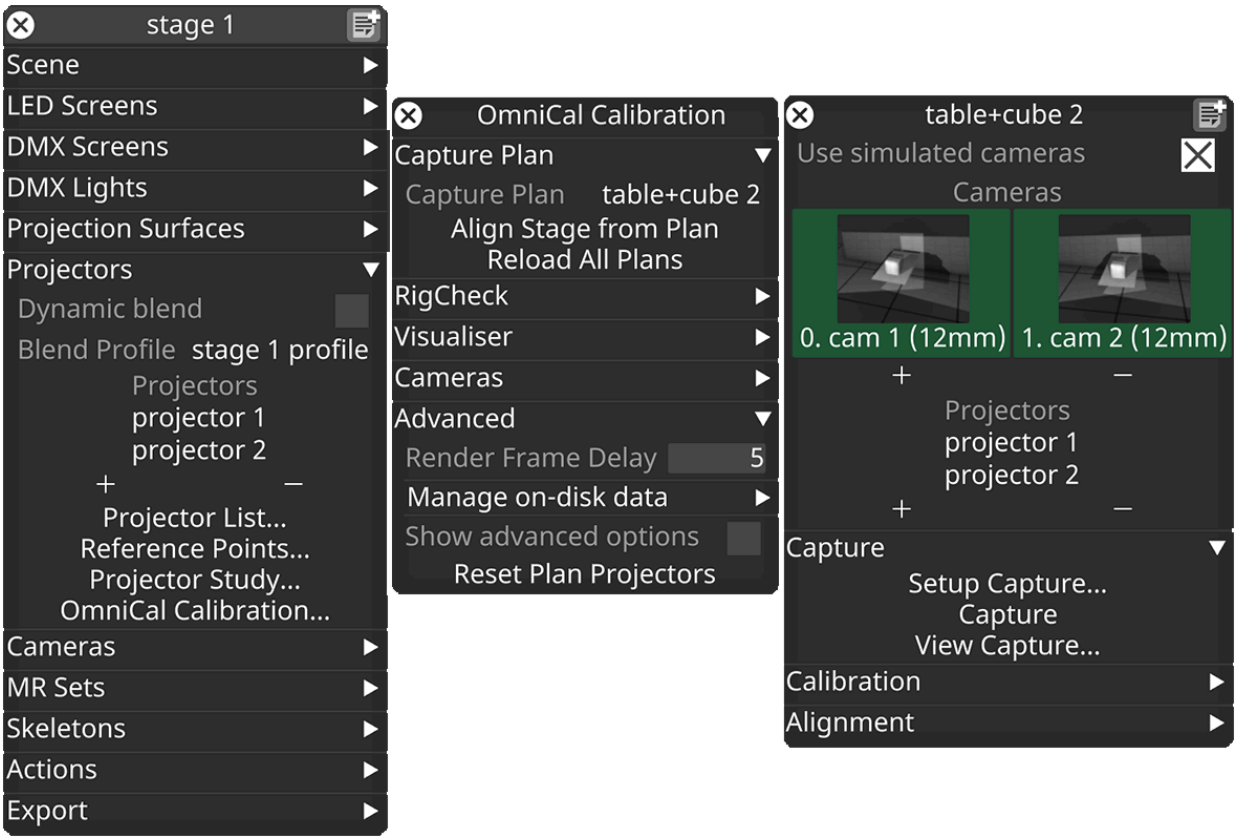
After completing camera and network setup, OmniCal is ready to capture using the Plan workflow:

1. Configure the positions and properties of **cameras and projectors**.
2. **Setup** the Capture.
   This may require adjusting exposure, projection and alert settings.
3. **Perform** the Capture and wait for completion.
   The capture may take a while to complete. OmniCal will show the progress in the UI and may alert users of issues that require their attention.

OmniCal Stage Plan

In the **Stage** editor, under **Projectors**, click **OmniCal Calibration** to open the **Calibration** editor. This widget controls the main OmniCal behaviour. Right-click on any **Capture Plan** to open its **OmniCalStagePlan** editor.

In the **OmniCalStagePlan** we recommend working through the sections from top to bottom: **Capture**, **Calibration**, **Alignment**, and **Mesh Deform** (if required).



Plan Workflow

This page details the OmniCal Plan workflow intended for trained users. For re-calibration of existing Plans, please see the RigCheck workflow.

Configure cameras and projectors

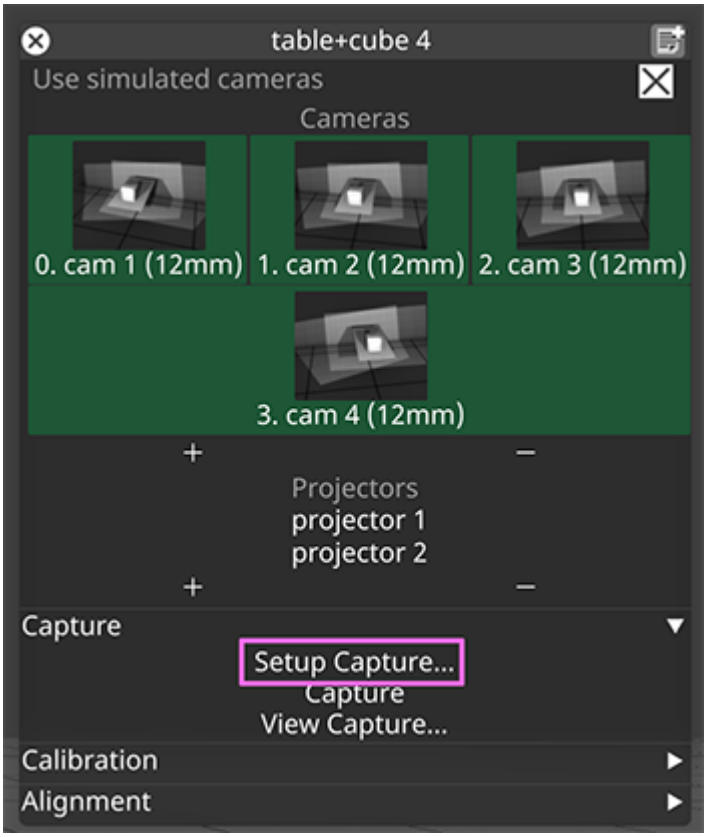
1. Left-click the calibration editor from the stage editor to open it.
2. To create a new Capture Plan, left-click on **CapturePlan**.
   This opens the **capture plan manager**.
3. Enter a name in the **New OmniCalStagePlan** field and click **OK**.
4. Right-click the newly created capture plan.
5. To work with simulated capture (without physical cameras), enable **Use simulated cameras**.
6. Click the **+** icon to add a new plan camera to the plan.
7. Right-click on each of the plan cameras to open its camera plan editor, to make further changes:
   - Adjust camera settings like offset, rotation or focal length as needed.
     The pose will be used for visualisation, and also for simulated captures.
   - If physical cameras are connected, then click **Physical OmniCal Camera** and select the appropriate camera from the list of discovered cameras.
   - Ensure that the configured focal length matches the physical lens.

8. Click the **+** icon to add projectors to the plan.
   - Ensure that projector outputs are correctly assigned in the **Feed View**.
   - Ensure that projection surfaces are correctly assigned to the projectors.

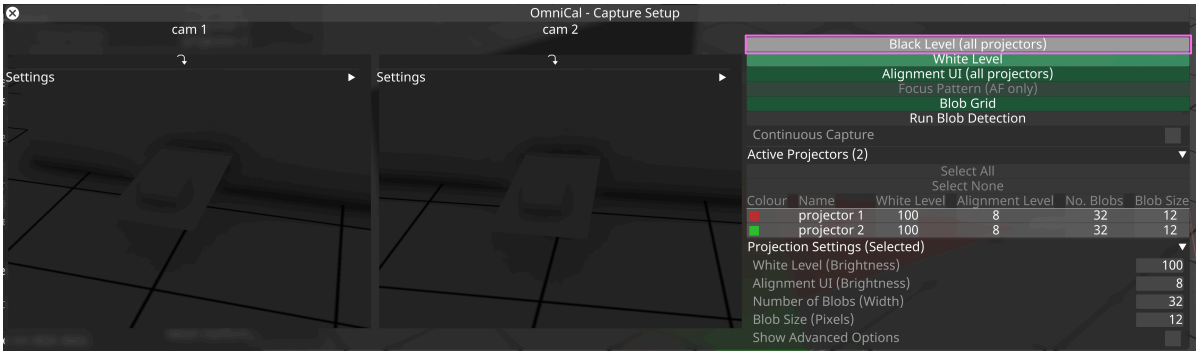
Capture Setup

When using physical cameras (no simulation), **Capture Setup** requires the camera network to be set up correctly (see previous chapter OmniCal Setup.

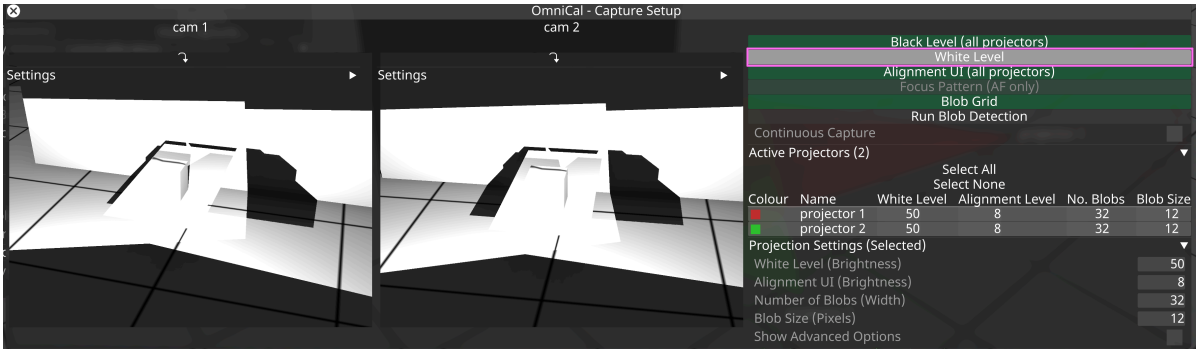
1. Left-click **Setup Capture** to open the **Capture Setup** window.



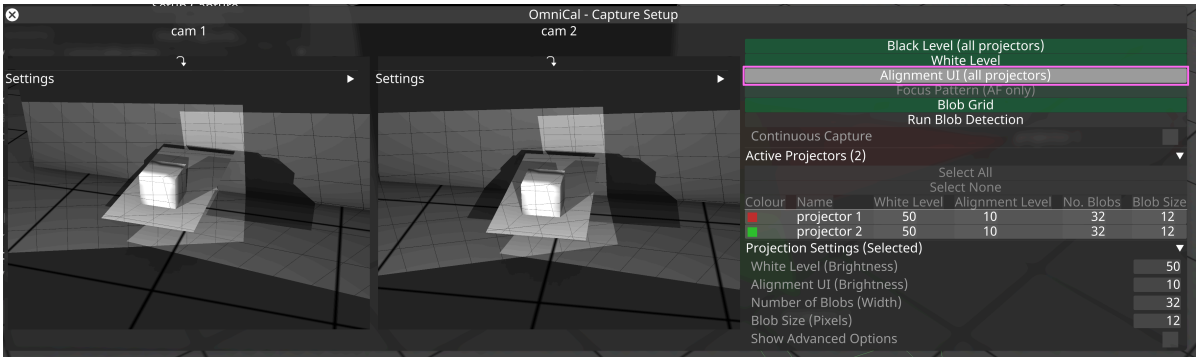
2. Click **Black Level (all projectors)** to capture the OmniCal camera view when projectors are outputting black. Black Level images should be fairly dark, but still allow recognition of important scene elements. Adjust using **lens aperture (iris)** on the physical lens and **exposure time** in Designer. Only use **gain** if the desired look cannot be achieved with other means.



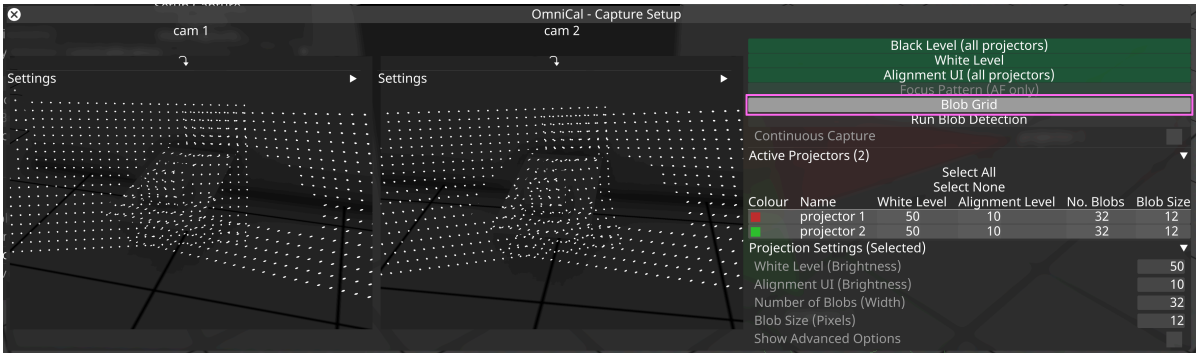
3. Click **White Level** and adjust the White Level (Brightness) value to make sure the camera images are not overexposed. Click on any projector to view and adjust it individually. The goal is to achieve a similar brightness for all projectors in the camera images.



4. Click **Alignment UI (all projectors)** and adjust the Alignment (Brightness) value to make sure you can clearly see the grid on the projection surface. The goal is to have alignment images where you are able to visually identify parts of the real-world projection surface, so they can later be matched up in QuickAlign (after Capture and Calibration).



5. Click **Blob Grid** to see how many blobs are projected, and how well they cover the surfaces you are calibrating.
   Then adjust the blob size and the grid size (Number of Blobs) according to your projector and camera setup.

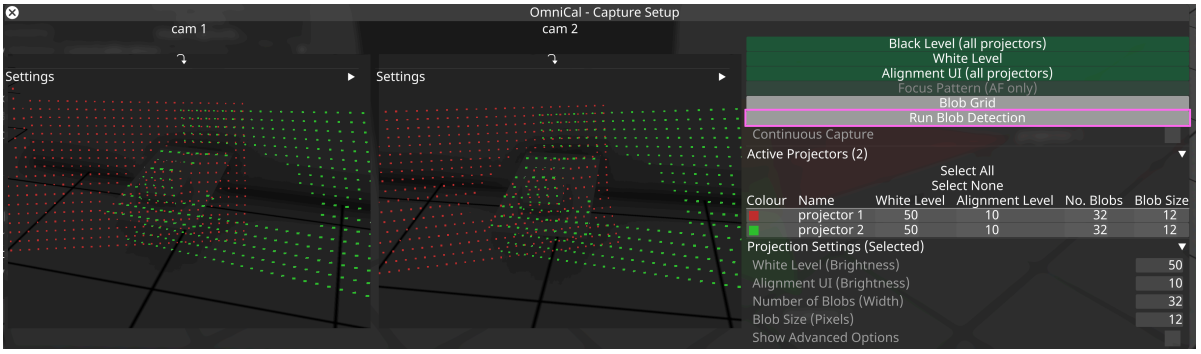


- The **blob size** is the size of projected blobs in pixels (in each projector's output image). This should be chosen as small as possible, but so that blobs are still clearly visible in all camera images. Note that the same blob may appear smaller in cameras that are further away or use wider-angled lenses (smaller focal length).

- The **grid size** is the number of blobs projected horizontally. Note that high values for grid size and larger blob sizes may lead to blobs touching each other. This must be avoided. Blob detection will fail if there is no clear gap between individual blobs.
- Blobs should appear with enough contrast against the background (i.e. Black Level) to make them easy to recognise.

> **Note**
>
> Generally speaking, blobs should be as small as possible whilst still remaining detected by all cameras. More blobs do not necessarily mean better calibration, but will increase calibration time. 32 blobs across the image width should be sufficient for most use cases. Projecting more blobs is useful when **Mesh Deform** is used after calibration and alignment, as it increases the number of inserted vertices.

6. Click **Run Blob Detection** to test the blob detection on all OmniCal cameras.



- A test blob detection will take a few moments until all cameras have taken images for all projectors. Results will be highlighted in each camera view.
- Detected blobs will be shown in the respective projector colour.
- At this stage the camera aperture or exposure may need adjusting, to improve blob detection and make it consistent for all cameras.
  - To adjust exposure, click on the camera name and adjust the exposure time in the Camera Plan Editor.
  - Adjusting the camera lens aperture requires physical access to the focus ring on the physical camera lens.

> **Note**
>
> A good way of finding a suitable exposure time is to turn **Continuous Capture** on, select **Blob Grid** mode and then adjust the camera exposure time until the blob levels are clearly visible. Then turn **Continuous Capture** off again.

Configure OmniCal Alerts

Alert settings allow users to adjust how OmniCal behaves when it encounters potential issues, such as blurry camera images. If a configured threshold is crossed, OmniCal will alert the user via a notification.

At the moment, alert settings only affect **OmniCal post-capture processes** that run towards the end of a capture. Some settings apply to both the Plan and RigCheck workflow, and some only to RigCheck.

These alert settings are user-configurable, to allow adjusting to specific project or environmental demands.

OmniCal alert settings typically consist of absolute or relative thresholds (percentages) that the current capture is compared against:

- **Absolute thresholds** are compared directly against actual values calculated during the capture. Depending on the process, a notification is either triggered if the current value is greater or smaller than the threshold.
- **Relative thresholds** are used to compare an absolute value from the current (RigCheck) capture against the equivalent value from the earlier Plan capture. A deviation greater than the configured percentage will trigger a notification.

> **Note**
>
> OmniCal Alerts cannot currently be queried via the OmniCal API. This will be added in a future release.

Settings for camera focus detection

Camera focus detection checks captured images whether they are out-of-focus (i.e. blurry) by calculating a **blur score** for each camera.

Threshold Plan workflow

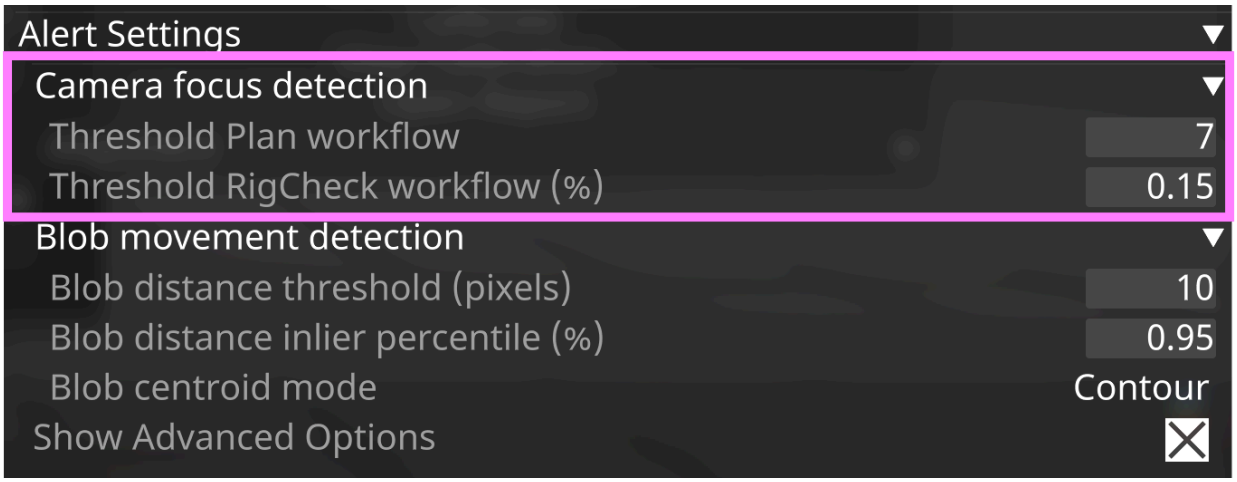
Absolute threshold for blur scores. Detects if the camera lens is potentially out of focus. Smaller values mean “less in focus” or more blurry.

- Unit: unitless
- Range: 0 - 1000. Default: 7
- Threshold type: absolute
- Workflow: Plan and RigCheck

Threshold RigCheck workflow (%)

Relative threshold for comparing blur scores between RigCheck and the original Plan. Detects if the camera lens focus has deteriorated since the initial installation.

- Unit: percentage
- Range: 0 - 1. Default: 0.15
- Threshold type: relative
- Workflow: RigCheck



Settings for blob movement detection

Blob movement detection checks whether captured blob coordinates have moved between the original Plan and a subsequent RigCheck. If the measured movement exceeds the defined limits, then OmniCal generates a notification.

Blob movement detection only applies to the RigCheck workflow.

Blob distance threshold (pixels)

Absolute threshold for average blob movement when comparing blob coordinates of the current RigCheck to the original Plan. OmniCal will show a notification if the calculated value is above this threshold.

- Unit: pixels
- Range: 0 - 200. Default: 10
- Threshold type: absolute

Blob distance inlier percentile (%)

Which percentile of blob centroid distances to use for calculations (assuming a normal distribution of the distances). Values outside the specified percentile are ignored for the blob movement calculation. Setting this below 100% helps to prevent the effect of outliers distorting the average distance result.

Default: 0.95, i.e. 95%.

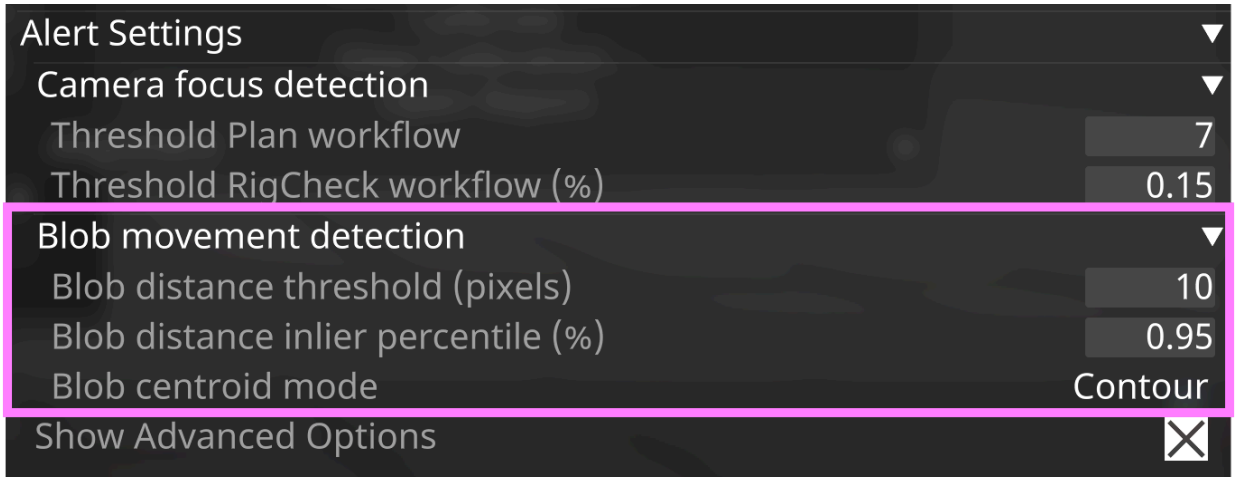
Blob centroid mode

Determines which blob centre coordinate to use for calculations: *Contour*, *Ellipse*, *BoundingBox*, or *BrightestPoint*. This is an advanced setting that normally does not need to be

adjusted.

For each blob, OmniCal stores 4 different centre coordinates. These centre values may vary depending on the shape of the blob (as it appears in the camera). For example, a projected circle may only be partially visible in a camera, leading to a deviation between the centres of, e.g. the detected contour and an ellipse modelled around it.

Default: *Contour*.



Perform a capture

Capture preparation to avoid interference

During an ongoing capture, we recommend the following:

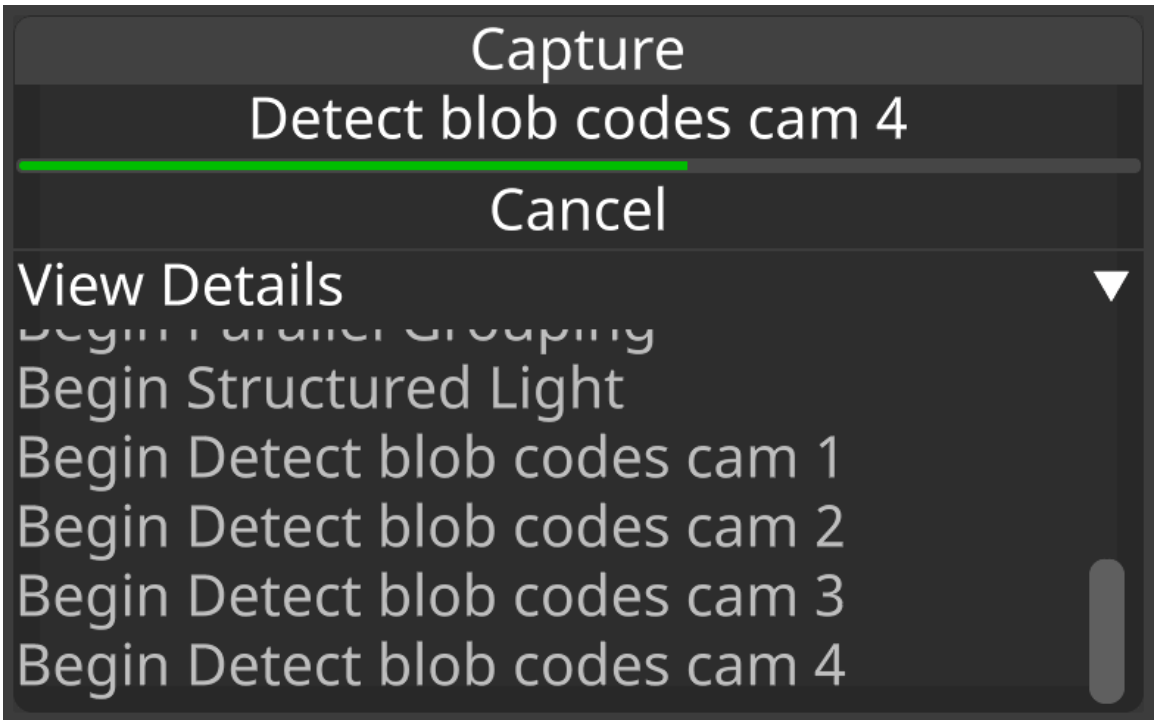
- The stage should be clear of interfering objects that might obscure the projection surface (e.g. ladders or other construction equipment).
- No changing light levels.
- No reflective surfaces in the projection path.
- No moving projection surfaces.
- No people walking across the projections or camera views.

Any such interference may lead to incorrect capture data that causes bad calibration results. If you notice any such interference or other capture problems, then cancel the current capture, remove the source of the problem and start a new **Capture**.

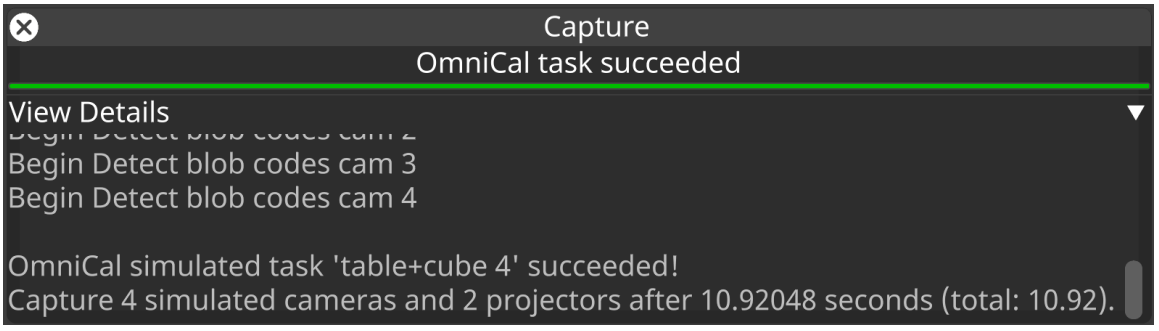
To perform a capture

1. Click **Capture** from the plan editor.

2. The system will perform a capture.
    - The time taken depends on the number of projectors and cameras, whether projectors are converging or not and any overlap between projectors. For example, 4 cameras and 4 non-converging projectors takes roughly a minute.

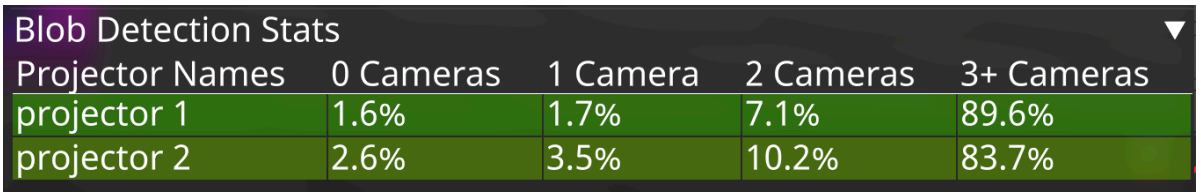


    - OmniCal will run some **post-capture processes** towards the end of the capture. These processes compare images against **configurable thresholds**, and may produce notifications if any thresholds are triggered.



3. Verify the results of the capture by clicking **View Capture**.
4. Click **Blob Detection Results** to see which blobs were detected in the cameras.
5. Verify that the blob detection results are as expected. These results should be consistent with what you saw earlier in the capture setup.

6. Blob statistics show how many cameras observe / detect a projector's blobs. An observer ratio of 50% or higher would be ideal.



Blob Detection Stats

Projector Names	0 Cameras	1 Camera	2 Cameras	3+ Cameras
projector 1	1.6%	1.7%	7.1%	89.6%
projector 2	2.6%	3.5%	10.2%	83.7%

Post-capture processes

During the OmniCal capture, certain post-capture processes will automatically run on some of the captured images. The purpose of these processes is to detect whether any conditions outside of Designer may affect the quality and validity of the capture.

Images are compared against **configurable thresholds**, and if any thresholds are triggered, OmniCal will notify the operator with relevant information and suggestions for next steps.

Camera focus detection

Camera focus detection helps to ensure that all OmniCal camera lenses are in focus. If a camera is found to potentially be out of focus, a notification will be issued. Operators may then need to manually adjust the lens focus on the physical camera.

Camera focus detection uses Fourier Transform algorithms to calculate a *blur score* for each camera. This score is compared to thresholds to detect whether the image is likely blurry.

Users can configure the behaviour of this process using **threshold settings**. The thresholds may also get automatically adjusted by OmniCal, based on the nature of the scene (see below). If the blur score for a camera is below the adjusted threshold, then this camera's lens is likely out of focus.

Meaning of calculated blur score

The blur score is a unitless measure that determines how relatively in or out of focus a camera lens is. A higher blur score means a more focused, and a lower score describes a more blurry image.

Note that the blur score also depends on the mage content, and can vary for a given camera, even if the lens focus is not modified. This is because the blur score calculation is sensitive to scene composition and environmental influences, including:

- Light distribution in the scene
  - Very dark (underexposed) or very bright (overexposed) areas are equivalent to low frequencies, so will lead to worse blur scores.

- Natural structure of the scene
  - Uniform colour / structure areas are equivalent to low frequencies, so will lead to worse blur scores.

> **Note**
>
> The blur score can only be meaningfully compared between roughly equivalent images, e.g. same camera pose, similar lighting conditions, same projector output, etc.

If a captured image was found to have any of the above influences, then the **configured threshold** will be automatically adjusted accordingly, to prevent false positive warnings.

Blob movement detection

Blob movement detection helps to confirm that the scene and rig setup, camera alignment, and overall positioning remain consistent during the RigCheck workflow.

This process checks whether any of the projected blobs have shifted or moved in any camera images between the original Plan and a subsequent RigCheck. It identifies blob centres in both datasets, measures the distance between matching centres, and calculates the overall average movement in pixels.

Blob movement detection also detects if a camera can no longer “see” blobs from a certain projector. This may indicate that a projector is turned off, or its footprint has changed considerably.

Users can configure the behaviour of this process using **threshold settings**. If the measured movement exceeds any of the defined limits, then OmniCal generates a notification.

Meaning of detected Blob movement

If blob movement detection triggers a notification, then this can mean one (or more) of the following:

- One or more cameras have moved.
  - The RigCheck will need alignment adjustment for each affected camera.
- One or more projection surfaces have moved.
  - The RigCheck will need alignment adjustment for each affected surface.
- One or more projectors have moved.
  - OmniCal will automatically update all projector calibrations.
  - As long as the projection surface is still covered sufficiently, no further action is required.
  - In case 2D warps or manual blend masks are used, these will likely need adjusting.

OmniCal cannot currently determine which of these situations has occurred. Operators are advised to visually check the scene for alignment issues and adjust them using the Result Aligner, if needed.

Managing OmniCal data on disk

OmniCal Plan data on disk

OmniCal's Plan workflow stores a mix of data on the hard disk, rather than in the .d3 project itself.

This includes:

- OmniCal settings
- Captured images
- Calculated blob detection results
- Calibration results
- Alignment data
- Report data

This data is stored in subfolders of the Designer project's `internal/opticalstageplan/` folder. Every Plan has its own set of data, and its subfolder has the same name as the associated Plan object in Designer.

The data in the **internal** subfolders is managed together with the corresponding Plan object in Designer: When the Plan object is deleted, and the trash in Designer is cleared, any associated folder (including images, config and result files) is also deleted from the hard disk.

Storing many Plans in a project folder can fill up the hard disk over time. It is usually a good idea to delete any Plans that are no longer in use. This not only frees up disk space; it also reduces confusion if there are many different Plans with similar names.

> **Note**
>
> Whenever a new Capture is started in the Plan workflow, all previous capture and calibration data for this Plan is first deleted from the hard disk. OmniCal settings (including alignment and Mesh Deform) are not deleted, though.

OmniCal RigCheck data on disk

RigCheck result data is treated the same as Plan data, except that it is stored in the Designer project subfolder `internal/opticalresult/`. At the beginning of a RigCheck, all settings (including alignment data) are copied from the Plan to the new RigCheck result. See also Managing RigCheck data on disk.

Comparing Plan variations

Often it is necessary to compare the impact of different Plan setups or settings, e.g. the number of cameras or their placement or lens options. Sometimes it's only the impact of calibration settings that needs to be compared.

This is best achieved by copying the relevant Plan and editing the copy:

- Right-click on the Plan title bar, choose a new name and click **Duplicate**.
- Make any required changes to the copied Plan's settings.
- Start a new Capture and/or Calibration for the new Plan.

That way it's possible to switch between the two plans without losing any previous settings or results. This is more flexible and will improve understanding of the impact of any changes to a Plan setup or its settings.

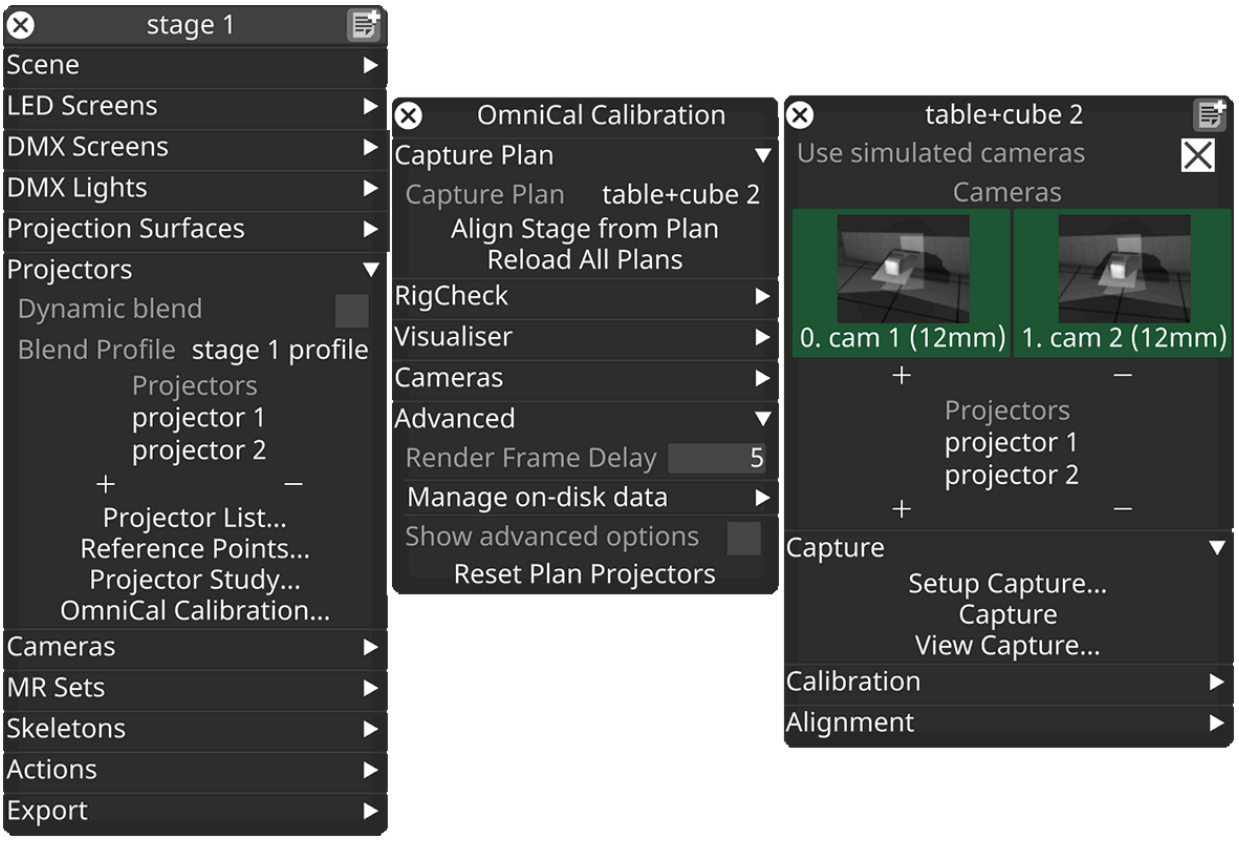
OmniCal Calibration

Calibration is the automatic process of calculating camera and projector poses and settings using the blobs detected during OmniCal Capture. Calibration also produces a consistent 3D point cloud formed from all blobs.

OmniCal Stage Plan

In the **Stage** editor, under **Projectors**, click **OmniCal Calibration** to open the **Calibration** editor. This widget controls the main OmniCal behaviour. Right-click on any **Capture Plan** to open its **OmniCalStagePlan** editor.

In the **OmniCalStagePlan** we recommend working through the sections from top to bottom: Capture, Calibration, Alignment, and Mesh Deform (if required).



Calibration

The calibration process triangulates the position of the blobs detected in a capture as well as the relative positions of cameras and projectors and their lens intrinsics. From this point, you no longer need access to the physical stage to continue the calibration process.

Calibration Stages

OmniCal calibration has several stages:

1. Camera Calibration
   - Camera poses are calibrated based on the detected blob information and pre-calibrated lens data.
2. Camera Graph Calculation
   - Cameras are connected to each other on a graph, depending on their mutual overlap.
   - This graph is saved in the project, and re-used in later RigChecks, to ensure that alignments remain compatible.
3. Point Cloud Reconstruction
   - A combined point cloud is generated from all blobs seen by all camera pairs.
4. Projector Calibration
   - Projector poses and intrinsics are calibrated using the point cloud and projected blob coordinates.
5. Bundle Adjustment
   - An iterative process that optimises the pose and intrinsics for all cameras and projectors by minimising the total reprojection error.
6. Final Point Cloud Reconstruction

Calibration Settings

Click **Setup Calibration...** to open the **OmniCal Calibration Setup** widget.

In most cases the default settings suffice. If calibration results are unsatisfactory despite good capture quality, adjust one setting at a time to clearly assess its impact.

> **Note**
>
> Most calibration settings include tooltips within Designer to provide contextual information on when and how to adjust them.

Camera Calibration settings

In the **Camera Calibration** separator we can set the **Calibration Model** to one of three modes:

- **Auto**: Attempt to detect appropriate model. Only manually select another mode if 'Auto' fails.
- **Epipolar**: Used with scenes "with depth", i.e. where blobs are not just projected on a flat surface, but are spread across 3 dimensions.

- **Homography**: Used with scenes where blobs are mostly projected on a flat surface, or the depth provided is very shallow (e.g. < 1cm).

Bundle Adjustment settings

In the **Bundle Adjustment** separator we can set **Bundle Adjustment target** to one of five modes:

- Auto
- Cameras
- Cameras and Projectors
- Cameras then Projectors
- Cameras then each Projector

The number of **iterations** controls how much time is spent on iteratively improving the overall calibration result. There are two separate iterations settings, that are relevant, depending on the selected BA target:

- **Iterations (combined)**
  - Used for **Cameras** and combined **Cameras and/then Projectors** modes.
  - High values will lead to longer calibration duration, especially for large numbers of cameras and projectors.
- **Iterations (individual projectors)**
  - Used for **Cameras then each Projector** mode.
  - This mode is faster, so the iterations can be set much higher than for the combined modes.

Point Cloud Reconstruction settings

A setting under **Point Cloud Reconstruction** that is often useful is **Camera graph optimisation**. This should be enabled for camera setups where all cameras overlap with each other, so that they form a closed loop, i.e. the first and last camera overlap with each other. Enabling the camera graph optimisation prevents gaps or discontinuities where the point cloud boundaries meet. This will give more consistent reconstruction and calibration results for typical setups for 360-degree projections inside of or on domes, cylinders, spheres or the walls of a room.

Camera graph BA iterations controls a separate Bundle Adjustment pass for the camera graph optimisation. The default is 500, but it is rare to need more than 50 iterations. This step is much faster than later Bundle Adjustment steps that involve cameras and projectors.

Projector Calibration settings

Projector Calibration Algorithm of **Projector Calibration** has three modes:

- DLT: Direct Linear Transformation

- Zhang
- Auto: will try all available algorithms and select the best result.

Perform Calibration

Once you have finished adjusting the Calibration Settings, click **Calibrate** in the Plan editor and confirm with clicking “Yes” to start the calibration process. Depending on the number of cameras and projectors, this can take from a few seconds to tens of minutes.

After starting a calibration, a new window will open that shows its progress. This window also contains detailed messages about the current **calibration stage**. Once the calibration has finished, you can close this progress window. The messages from the progress window are saved to the OmniCal report, for later viewing.

OmniCal Report

The OmniCal report is useful for assessing the success and quality of a calibration, and for potential trouble-shooting.

Click **Open Report** to see a summary of the current Plan or RigCheck Result calibration. A new window will open which details the calibration success using RMS reprojection errors and other metrics. The report also shows the detailed messages from the earlier calibration progress window.

Error values below 1 pixel are considered good, and above 1px usually point to something going wrong in the process (physical setup, scene, configuration, ...). The report values are displayed using a colour between **Green** and **Red** to better highlight problematic camera or projector errors. If everything is green, then the calibration was a success.

> **Note**
>
> Projector calibration depends on the quality of the earlier camera calibration step. Therefore any larger camera errors will likely result in either high projector errors or unreliable calibration results. This can be useful to determine at what stage in the OmniCal process an issue arises.

OmniCal Score

The OmniCal Score is a simple way to assess and compare calibration quality between Plans or RigChecks. It was introduced in r32.4 as a “single value” measure that shows whether a calibration was successful, and how good it is.

The basis of the OmniCal Score is the rounded up **aggregate projector RMS error**, enhanced with penalty terms for any issues that were identified during the calibration. These added penalties make the OmniCal score more useful for identifying success than standard RMS

errors, which sometimes can be low despite unrealistic calibrated parameters. While the OmniCal Score is not an RMS value itself, it uses the same open-ended range, where 0 is the theoretical best value, and a value of 1 or less is considered ideal.

The OmniCal Score contains penalty terms for issues identified during the calibration, including, but not limited to:

- Projector with high RMS error.
- Projector with invalid z-clipping values.
- Projector result is mirrored or pointing backwards.
- Projector result with non-square pixels or skewed image axes.
- Projector result with unrealistic focal length, lens shift or lens distortion coefficients.

The OmniCal Score is also contained in RigCheck API responses, to make it easy for API callers to compare a result against an earlier one.

Camera Table

This table lists each OmniCal camera of the current calibration against its calculated RMS error. It also shows whether the calibration resulted in square pixels, i.e. the aspect ratio for a single pixel in the camera is 1:1. Non-square camera pixels are usually the result of incorrect input data, e.g. wrong camera focal length, incorrectly scaled Feed Rects or incorrect settings in the physical projector.

Below the table is the aggregate camera RMS error, which combines all camera errors into a single value for easier comparison between different calibrations.

Projector Table

This table lists each projector of the current calibration against its calculated RMS error. It also shows whether the calibration resulted in square pixels, i.e. the aspect ratio for a single pixel in the camera is 1:1. Non-square projector pixels are usually the result of incorrect input data, e.g. incorrectly scaled Feed Rects or incorrect settings in the physical projector.

Below the table is the aggregate projector RMS error, which combines all projector errors into a single value for easier comparison between different calibrations. The OmniCal Score is based on this value.

Calibration and Point Cloud visualisation

Once calibration is complete, you can view the point cloud and pixel errors for each projector. To do so, go to the **OmniCal Calibration** widget and open the **Visualiser** section. Set the **Visualisation Source** to **Plan** and set the **Visualisation Mode** to **All** or **Only Point Clouds**.

The generated point cloud will now be shown in the stage visualiser.

> **Note**
>
> The point cloud is drawn using the Plan's current alignment settings, so it may not be aligned well to the stage until QuickAlign has been performed. The cloud visualisation is helpful in achieving an initial manual alignment, which will make QuickAlign easier to understand.

Point Cloud Visualisation

The **Point Cloud visualisation** option provides several visualisation modes that help with analysing issues such as interference from other light sources, or movement during a capture. The default and most useful mode is **ProjectorReprojError**, which visualises the individual reprojection errors for each projected blob after projector calibration.

Point Cloud Visualisation	Description
Off	Disable point cloud visualisation.
Observations	Draw number of camera pairs that see a cloud point. The observation count affects the 3D reconstruction: **Red**=0 (failed), **Blue**=1 (OK), **Green**=2+ (better quality).
InlierScore	Draw cloud point inlier category using colours. **Red**: invalid or outlier, **Orange**: outlier, **Yellow**: projector calibration outlier, **Blue**: camera calibration outlier, **Green**: inlier.
InliersOnly	Only draw cloud points that are inliers (**Blue**) after applying camera calibration classifiers such as RanSaC or LMedS.
OutliersOnly	Only draw cloud points that are outliers (**Red**) after applying camera calibration classifiers such as RanSaC or LMedS. Reveals unstable points, possible scene movement, mismatched blob codes etc.
Deform	Draw distance of cloud points to aligned mesh (along point normal), using Mesh Deform threshold settings.
ProjectorInliersOnly	Only draw cloud points that are inliers (**Blue**) after projector calibration.
ProjectorOutliersOnly	Only draw cloud points that are outliers (**Red**) after projector calibration.
ProjectorReprojError	Draw cloud point reprojection errors after projector calibration. The point colour indicates the error quality between **Green**=good, **Yellow**=acceptable and **Red**=bad/outlier. Helps identify problematic points or projection areas.

Point labels

With the **Point labels** option a text label can be rendered for each cloud point. The text content depends on the option value. The default value is **Off**.

Note that rendering cloud point labels has a significant impact on performance.

Point labels	Description
Off	Disable point labels.
GridCode	Draw x, y index that corresponds to the blob in the projector grid. This can be helpful for identifying points as part of OmniCal troubleshooting.
ProjectorReprojError	Draw cloud point reprojection errors after projector calibration. This is a single RMS error value per point and given in pixels. Lower values are better. Values larger than 1px may indicate problems with the capture, such as light interference, movement etc., especially if they appear in clusters.

Re-running calibrations

Re-run a calibration by modifying a setting and clicking **Calibrate** again. Note that issues introduced during image capture (lighting changes, obstructions, etc.) will require re-running **Capture**.

Re-running the calibration with different settings is useful:

- If the calibration quality was decent, but not yet good enough. Tweaking camera or projector calibration settings can help in some cases to get better results.
- If there were incorrect calibration settings (e.g. camera calibration was set to “Homography”, but the blobs land on non-planar surfaces).
- Simply to experiment with settings, e.g. when trouble-shooting issues.

Limitations of OmniCal

OmniCal can accurately calibrate projectors, including their particular lens distortion coefficients. Any rectilinear lens is supported, including ultra-wide-angle / ultra-short-throw lenses with throw ratios around 0.35, even if they use built-in mirrors.

However, OmniCal currently does not support fish-eye lenses.

Another minor limitation is that OmniCal cannot correct chromatic aberration in projector lenses. This is a less common phenomenon in projection (compared to photography), but it is noticeable at the corners of some ultra-short-throw lenses.

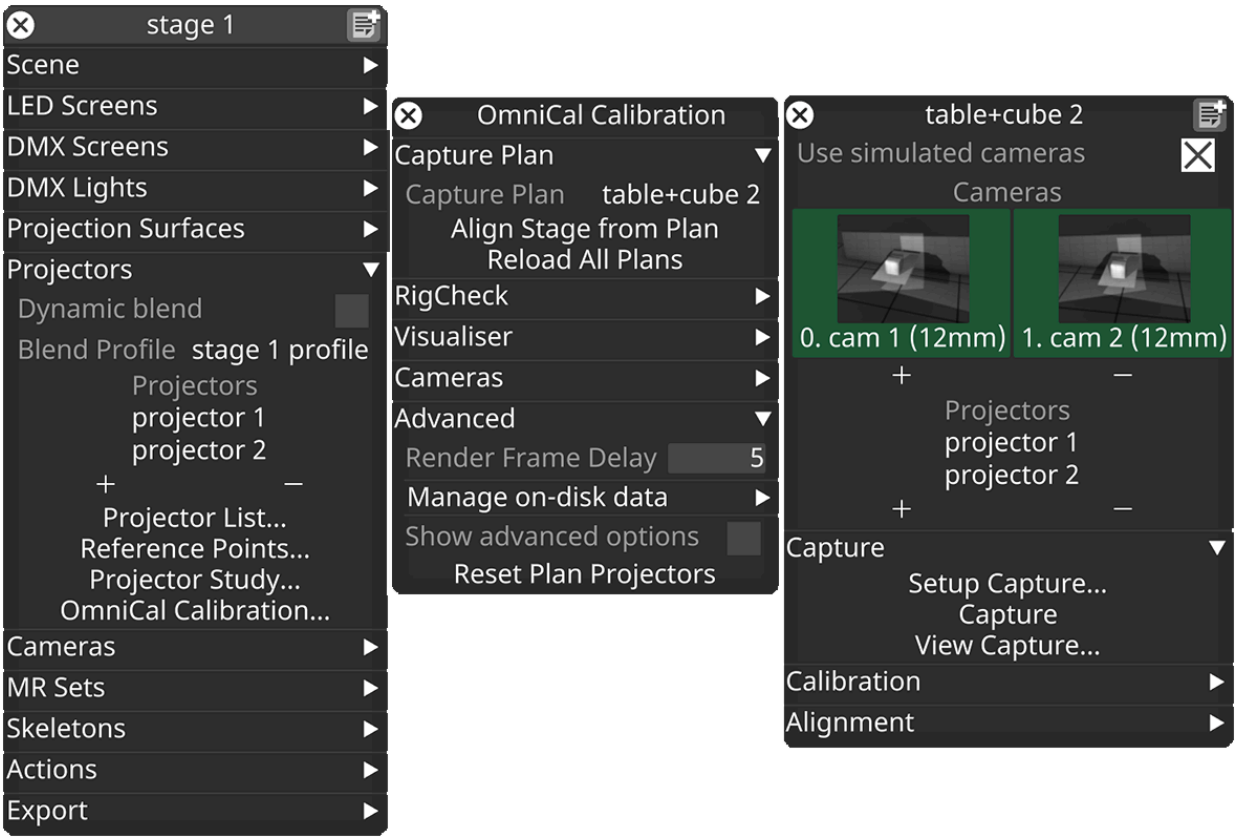
OmniCal Alignment

Alignment is a user-guided process that is required to align the calibrated data with the coordinate system in Designer. OmniCal also includes the ReShape feature to adjust 3D model vertices based on the user alignment.

OmniCal Stage Plan

In the **Stage** editor, under **Projectors**, click **OmniCal Calibration** to open the **Calibration** editor. This widget controls the main OmniCal behaviour. Right-click on any **Capture Plan** to open its **OmniCalStagePlan** editor.

In the **OmniCalStagePlan** we recommend working through the sections from top to bottom: Capture, Calibration, Alignment, and Mesh Deform (if required).

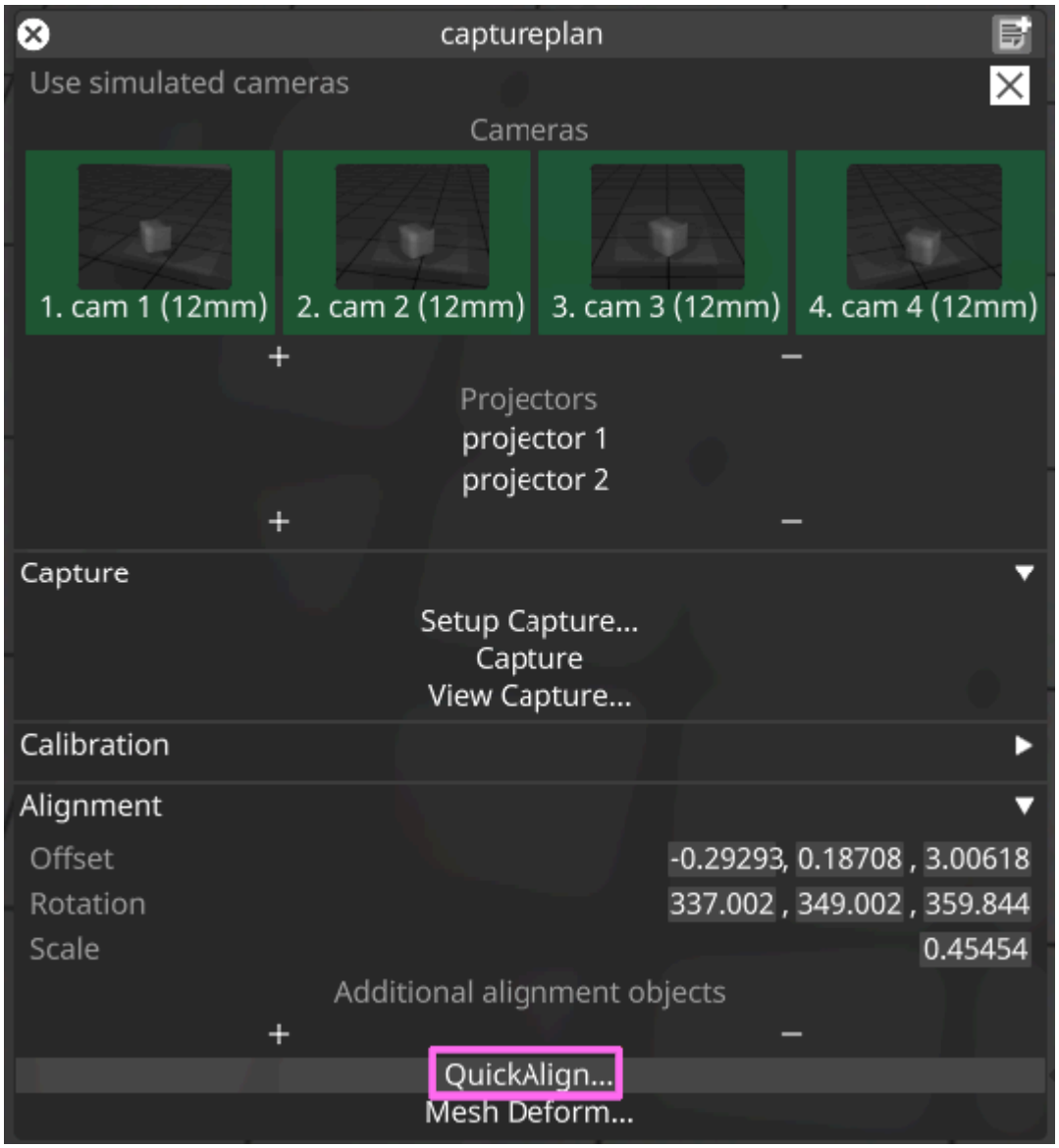


Alignment

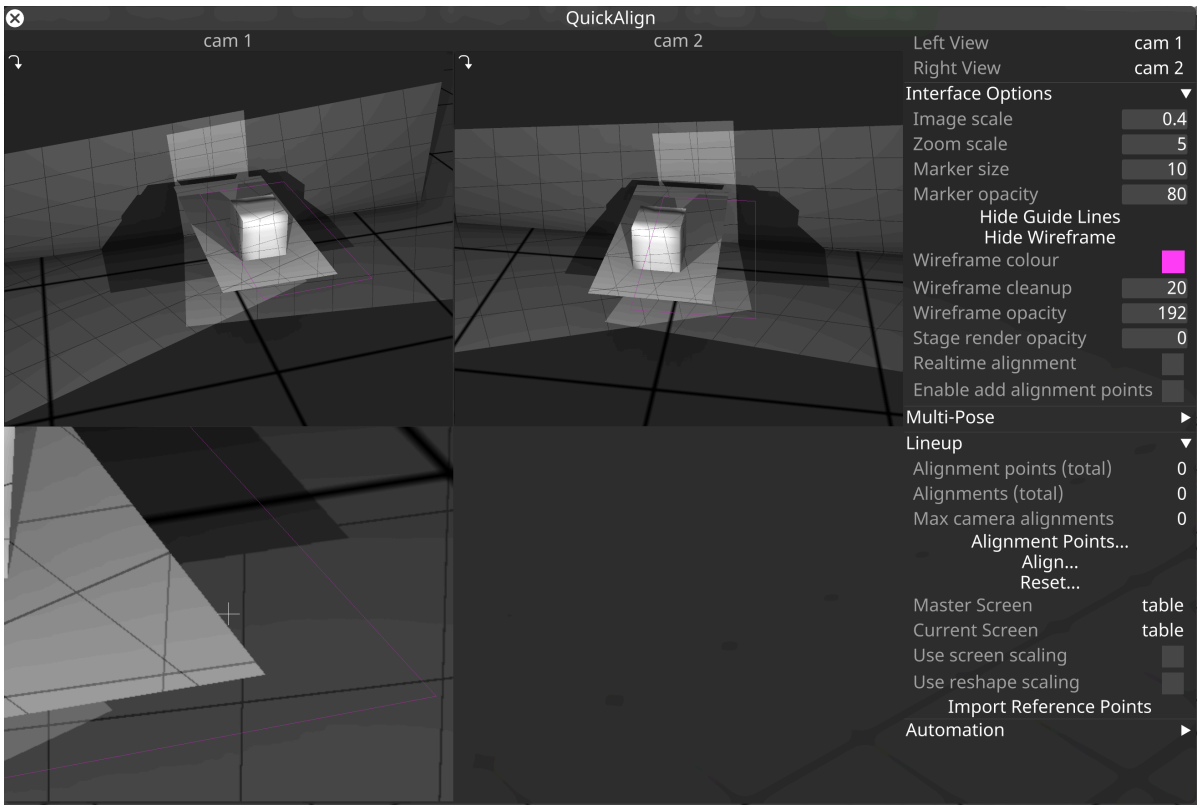
Alignment is the process of registering or aligning the coordinate system of the captured point cloud with that of Designer. It is a required step to make the projector output match the physical scene after OmniCal calibration has finished. A good alignment is a requisite for further steps like Mesh Deform.

Perform Alignment

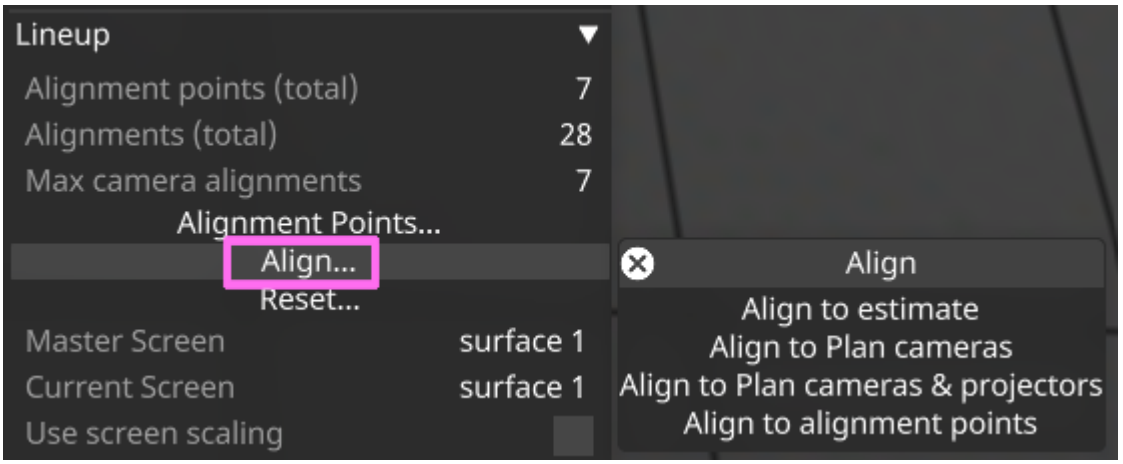
1. In the plan editor, roughly edit the point cloud **Alignment** via **Position**, **Rotation** and **Scale**. This does not need to be exact at all, just close enough in terms of scale and rotation that further steps are made easier.



2. Once the point cloud is roughly aligned (i.e. somewhat close to the projection surfaces in the visualiser), click **QuickAlign...** to open the tool for detailed alignment. In QuickAlign you should see a reprojected wireframe of the first projection surface, drawn on top of the alignment image taken for each camera.
    - The top two views relate to the cameras currently selected as **Left View** and **Right View** in the top right corner.
    - If the cursor is located in one of the camera views, then a zoomed-in view will be shown below it. This is essentially a magnifying glass for precise alignment of vertices to pixels in the camera image.
    - The **Interface Options** allow adjustments of QuickAlign UI elements to improve visibility during the alignment process. Most importantly, the alignment point marker size, opacity and the wireframe opacity and colour can be adjusted.



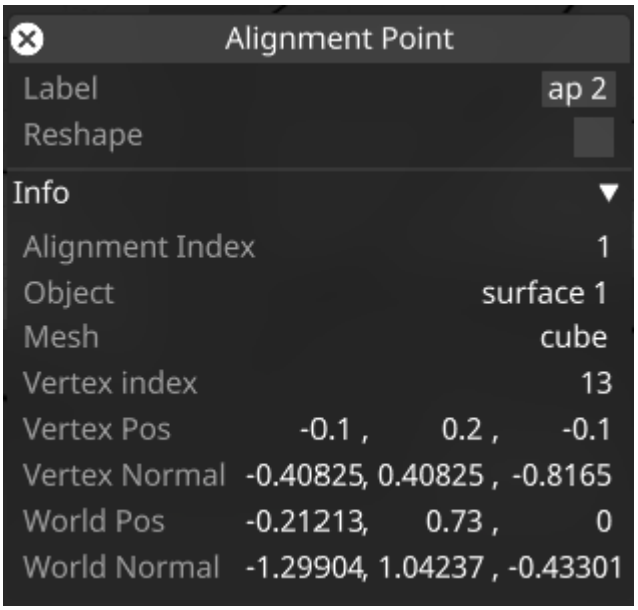
3. There are tools to automatically align the point cloud to the stage. In **QuickAlign** click **Align...** to open a menu with alignment methods:
   - **Align to estimate** creates an initial alignment estimate based on point cloud coordinates and vertices of all projection surfaces. This is not accurate, and requires that there is a good overlap between point cloud and all meshes, with little to no missing or extra areas.
   - **Align to Plan cameras** uses the positions of plan cameras. Requires that the planned positions of plan cameras are somewhat realistic.
   - **Align to Plan cameras & projectors** uses the positions of plan cameras and projectors. Requires that the planned positions of plan cameras and projectors are somewhat realistic.
   - **Align to alignment points** uses the manually created alignment points in each camera. Use this option when you want to (re-)apply the current alignment point data (see below) to the stage.



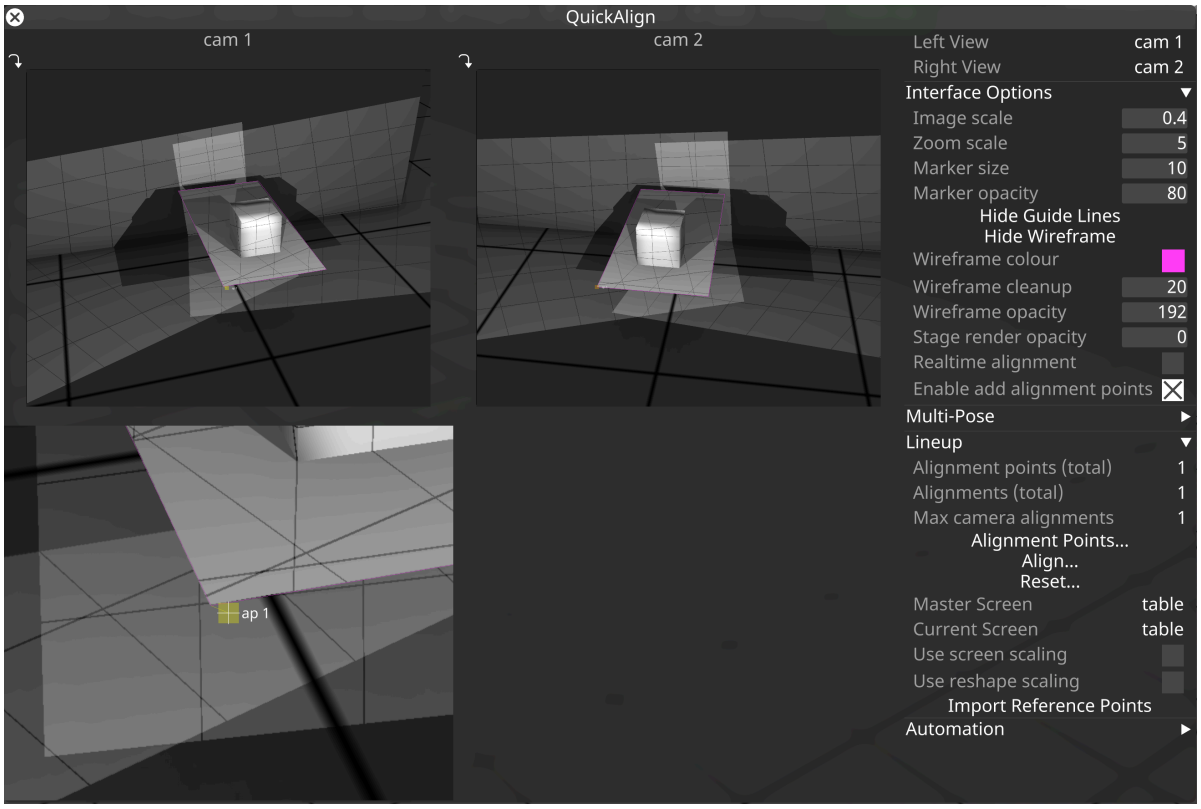
4. In most real-world environments the automatic Alignment tools don't provide a good enough alignment. To manually improve the alignment, create **AlignmentPoints** and line them up in multiple cameras. At least 3 alignment points need to be lined up in at least 2 cameras each. It's best to edit the alignment points one by one in each camera view. Alignment points are best placed on mesh vertices that correspond to distinct features in the physical model (e.g. a corner of a building). Not every camera will see every alignment point, so additional points may be needed.

5. Create **AlignmentPoints**:

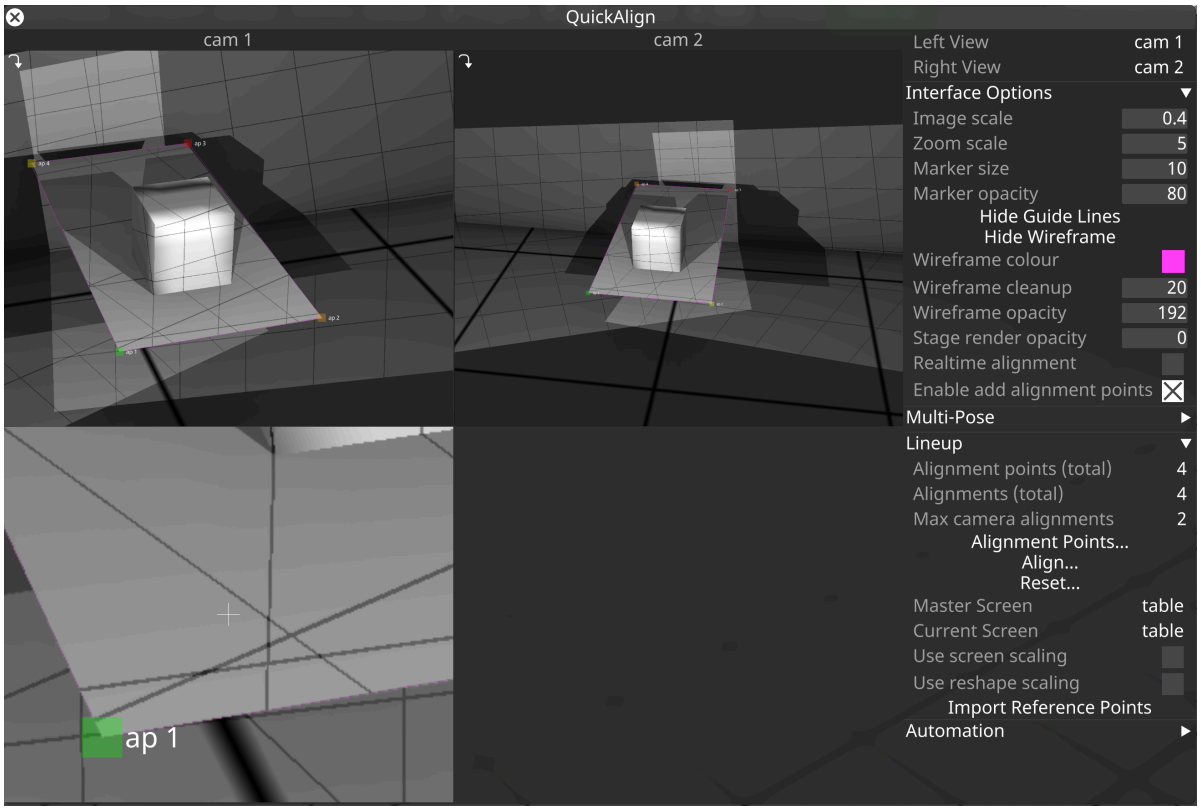
- Enable the checkbox **Enable add alignment points**.
- Click near a vertex on the wireframe to create an AlignmentPoint for this mesh vertex.
- Alternatively, AlignmentPoints can be imported from existing QuickCal projector ReferencePoints.
- Right-click on a point to access and edit its properties.



6. Drag the AlignmentPoint to the corresponding area in the camera image. Do the same for the corresponding point in the second camera view.



7. Repeat this process for a minimum of three points.
    - The currently selected AlignmentPoint will flash its marker. While it is selected, it can be moved using the arrow keys on the keyboard.
    - Hold **SHIFT** + arrow keys to move the AlignmentPoint marker faster.
    - Hold **CTRL** + mouse movement for slower fine control movement.



AlignmentPoint markers are drawn in different colours, based on their alignment state:

1. **Red**: This point is not being used as part of the alignment, but has been added to the view.
2. **Yellow**: It has been aligned only in the current camera. It is not used in the alignment calculation until it gets aligned in another camera.
3. **Orange**: It has been aligned only in a different camera.
4. **Green**: It has been aligned in at least two cameras, and therefore is used in the alignment calculation.

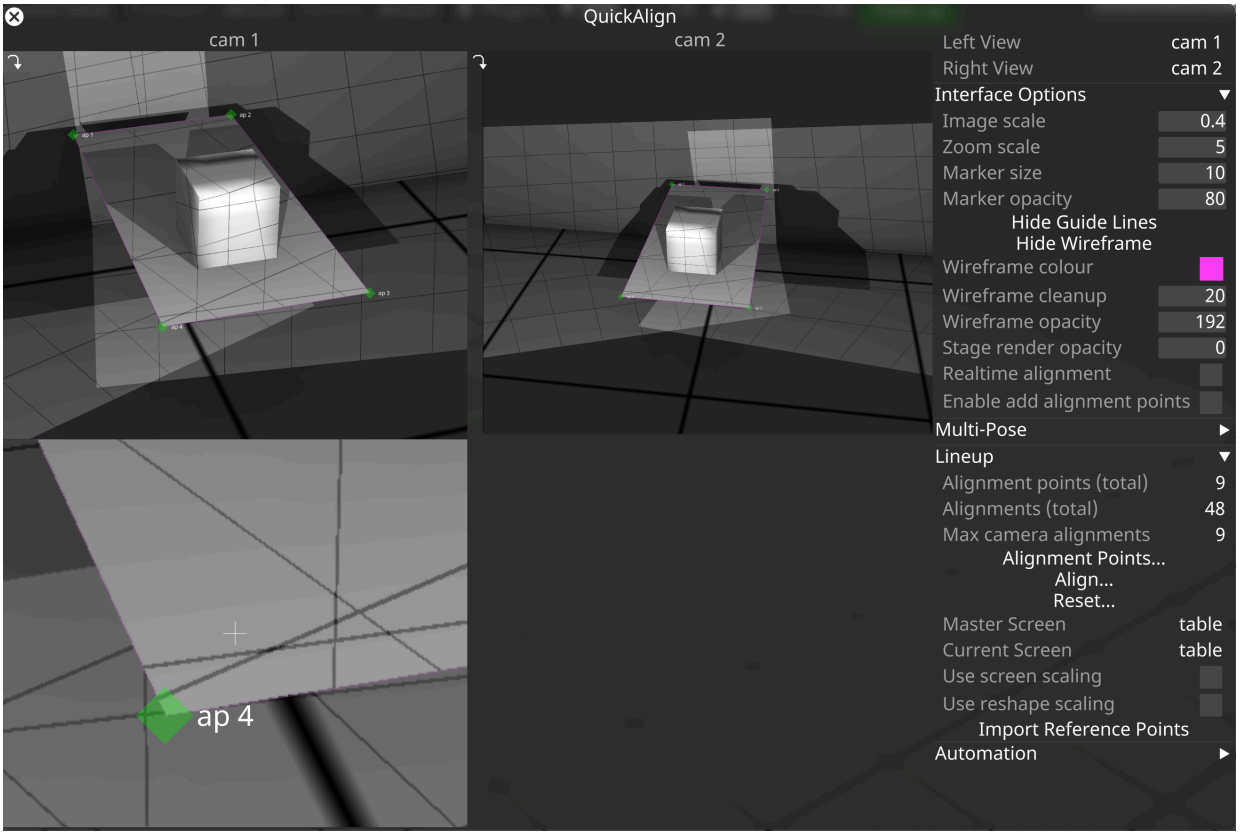
Multi-screen Alignment

If there is more than one screen in the plan, it may be helpful to use **Multi-screen alignment**. This allows aligning surfaces in the stage based on the captured images and a common master screen. If the relative offset and rotation between all surfaces in the stage matches the relationships of the actual physical surfaces, then this may not be needed.

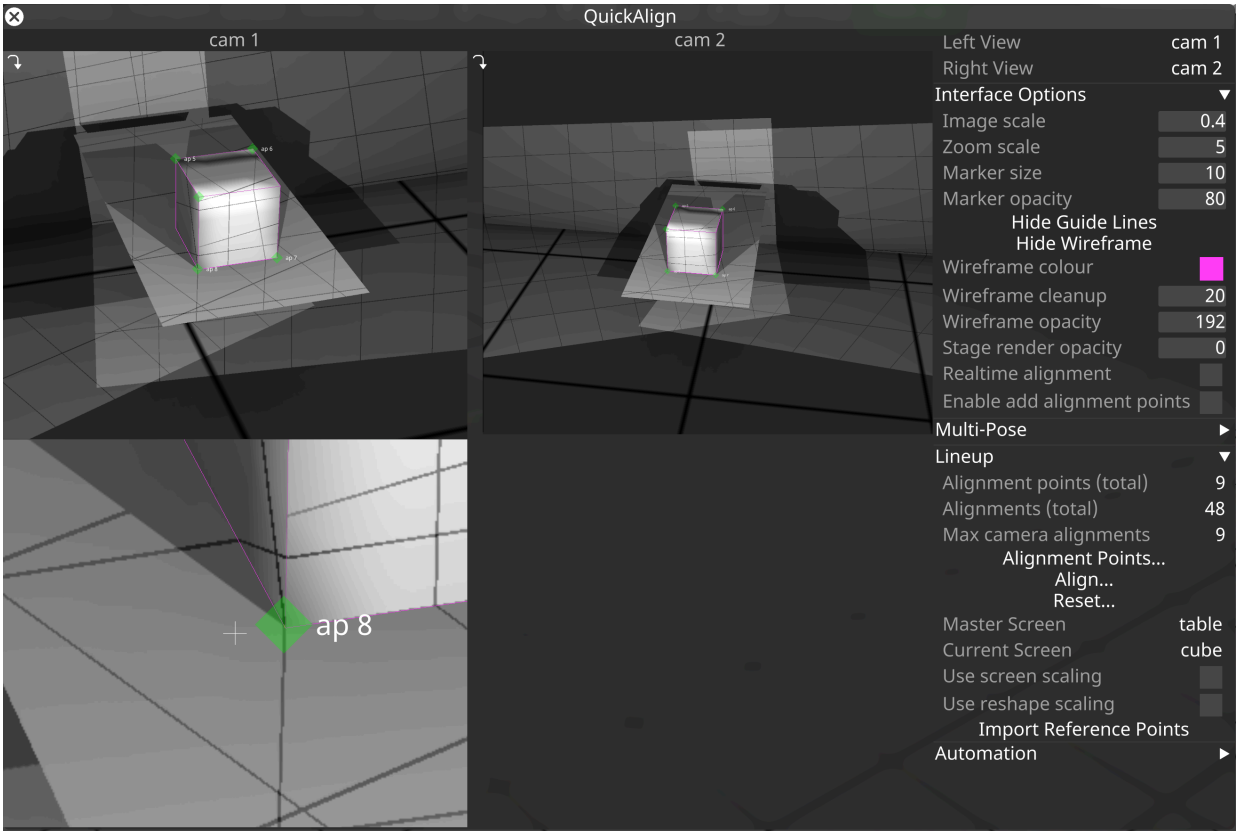
- The master screen should be aligned first.

- The master screen acts as a registration base for all other screens. Reshaping or scaling of the master screen will therefore affect all other screens.
- When aligning secondary screens, they will be moved to the correct position relative the Master screen.
- When stage automation is used to control physical screen positions, it is best to choose a non-moving surface as the master screen.

An example for how to align multiple surfaces can be seen in the demo video from the Overview page, at around 2:40 minutes.



Aligned Master screen



Aligned secondary screen

Alignment Reshape

If the proportions of the model match the physical surface exactly, then the alignment should fit perfectly. But if the proportions are not correct, you may need to perform **Alignment reshape**. Reshape allows adjustments to mesh vertices based on lined-up alignment points. This can be useful for adjustments such as stretching or squeezing the projection surface, or even moving some internal mesh elements (such as a window) by a certain amount. Typical applications are in theatre where scene elements are built according to a CAD drawing, but the physical construct ends up with minor deviations due to the construction process or limitations imposed by the stage.

The reshape process does not use the captured point cloud, but simply adjusts existing mesh vertices and mesh proportions. This is done according to the calibrated cameras, captured alignment images and alignment point data provided by the user. UV coordinates for a vertex are not modified during Reshape.

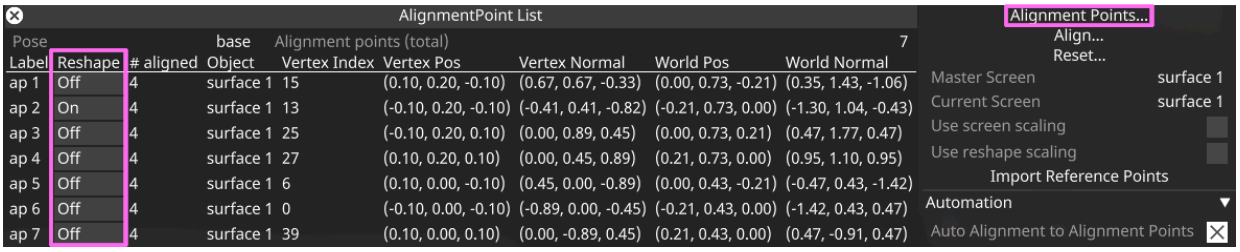
> **Note**
>
> When reshape causes stretching between vertices, this will in effect reduce the spatial resolution of the content, since any texture is now covering a slightly larger physical area. If a mesh requires significant changes (such as moving a building's window by a meter) then it may be a good idea to export the reshaped mesh, clean it up and adjust the affected UVs in a 3D modelling software, then export it again as an OBJ file into Designer. Note that such UV changes will also affect content for this screen, so this may then also need re-rendering.

Enable Reshape

1. To enable **Reshape** for an alignment point hold **SHIFT** and left-click it. This will turn it into a reshape point which is indicated by a diamond marker (45 degree rotated).
2. Left-click the point again (without holding **SHIFT**) to turn it back to a non-reshaped alignment point.

Alternatively, right-click an AlignmentPoint to open its editor, and toggle the **Reshape** checkbox.

The **AlignmentPoint List** widget offers a quick way to toggle **Reshape** for all alignment points.



AlignmentPoint List Widget

An example for how to use reshape points can be seen in the demo video from the Overview page, at around 4:24 minutes.

OmniCal Multi-Pose Alignment

Multi-pose alignment is an QuickAlign tool to allow projection on moving objects that are controlled by simple automation.

OmniCal QuickAlign positions secondary (non-master) objects relative to the master object. The normal workflow already allows users to align multiple objects separately from the master object. This is useful when the relative distance and angle between objects in the real world are unknown.

A multi-object scene supports projection on moving scenery if accurate 3D tracking data is available for all moving objects. However, often objects are moved on stages using 1D automation systems that provide accurate 1D motor information instead of 3D tracking data. Multi-pose alignment exists for this use case: it allows users to create accurate 3D poses for a moving object, where each pose is linked to a certain automation axis value.

Multi-pose alignment expands the alignment of secondary objects (but not the master screen) to cover motion. It adds the ability to align an object in multiple poses (position and orientation), where each aligned pose is associated with the automation value recorded at that time. Designer can then interpolate between the aligned poses based on the current incoming automation value. The interpolation produces new 3D poses between the recorded and aligned poses. This enables correct projections onto moving objects, even though the automation system itself only provides a single-axis value.

Currently, only movement along a linear path is supported, which requires 2 poses per object. These poses are typically chosen as the two extreme poses where the object is visible.

This workflow is for Multi-Pose Alignment in OmniCal, and not to be confused with QuickCal's Multi-Pose calibration of e.g. rotating objects.

Prerequisites

- A stationary **master screen** is required, in addition to any projection objects that are moving.
- **Accurate Meshes**: Multi-pose is designed to interpolate between object poses. These poses are determined by alignment points, therefore an accurate 3D model is required.
- For optimal calibration, the master screen should match real-world scale, as it sets the reference for the entire setup. If the master screen scale is incorrect, adjustments may be needed for all secondary screens.

Limitations

- Only supports linear paths, with a single input value from the automation system.

- Does not support calculating rotation pivot points.
  - If an object rotates between poses, the system will simply linearly interpolate between these rotations, which may not be what is desired.
- Multi-pose alignment only uses a single calibration (it is not multi-pose calibration like QuickCal). Therefore, there are some important requirements for projector calibration:
  - Projectors need to be calibrated with depth (e.g. using the DLT algorithm). Blobs cannot be on a single plane (coplanar).
  - Projection will only be accurate within the calibrated “space”, i.e. around the volume enclosing the blobs. Surface movement should be limited to this area. As always, Blobs that land on the floor or areas behind the object will be helpful in extending the calibrated volume.
- No mesh interpolation is done.
- Multi-pose alignment is not supported by the Rig Check workflow.
  - It shouldn’t normally be necessary to use Multi-pose on a regular basis.
  - It should only be necessary to redo it when there are changes to the automation system, causing screens to move along a different path.

Workflow Overview

1. Extract high-quality meshes.
2. Set up automation axes for all moving screens.
3. Perform a capture and calibration.
4. Align master screen and all static screens in the base pose.
5. Create two poses for each moving screen.

Detailed Steps

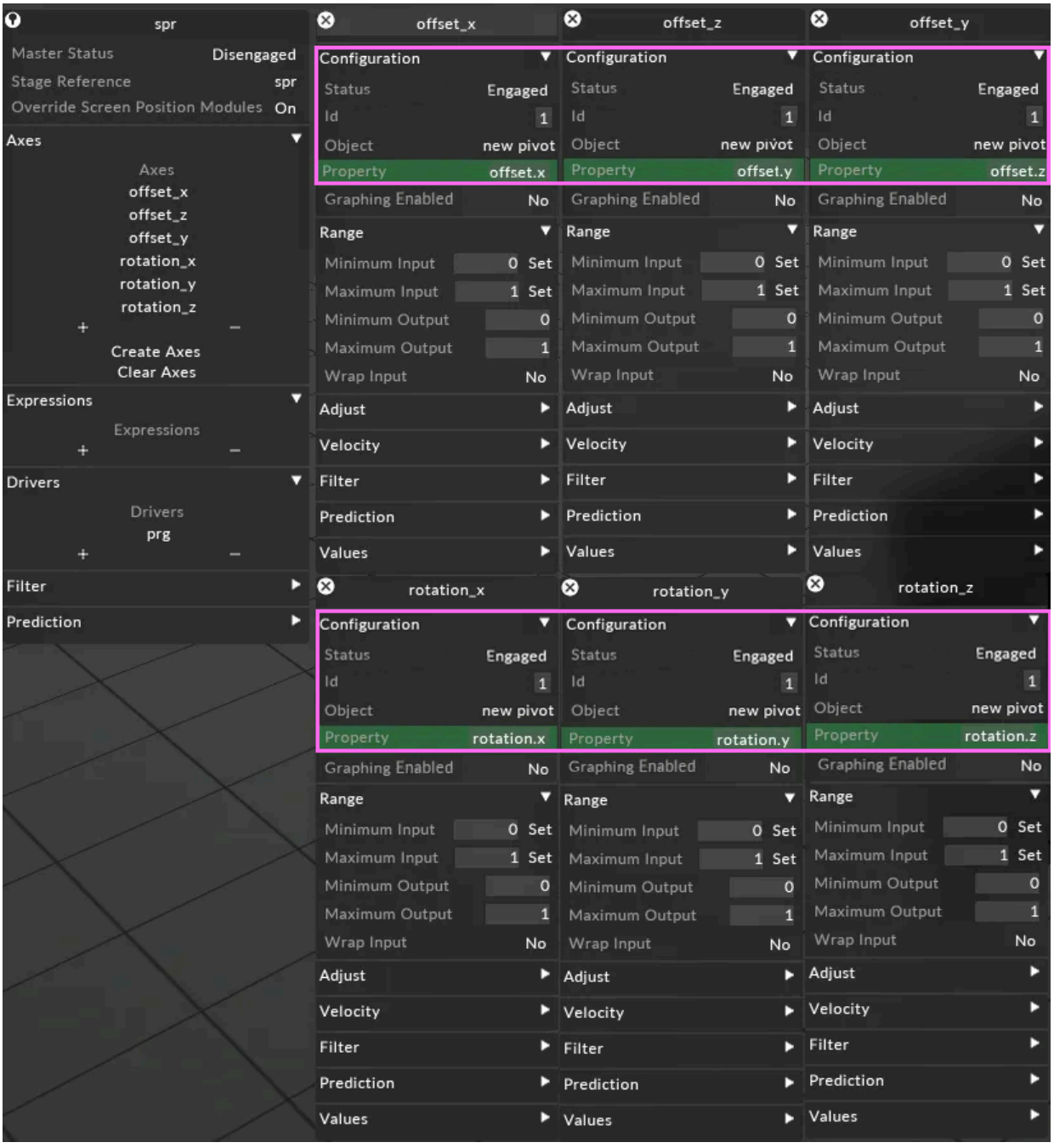
1. Extract high-quality meshes

The multi-pose workflow requires that you have accurate meshes. You can skip this step if you already have these or if you are in a simulation.

Please see Extracting a mesh using OmniCal for further information.

2. Set up automation axes for all moving screens

1. Create an automation device and driver using the regular workflow.
2. Create 6 axes, all with the same input ID from automation, for XYZ position and XYZ rotation of the object as follows.



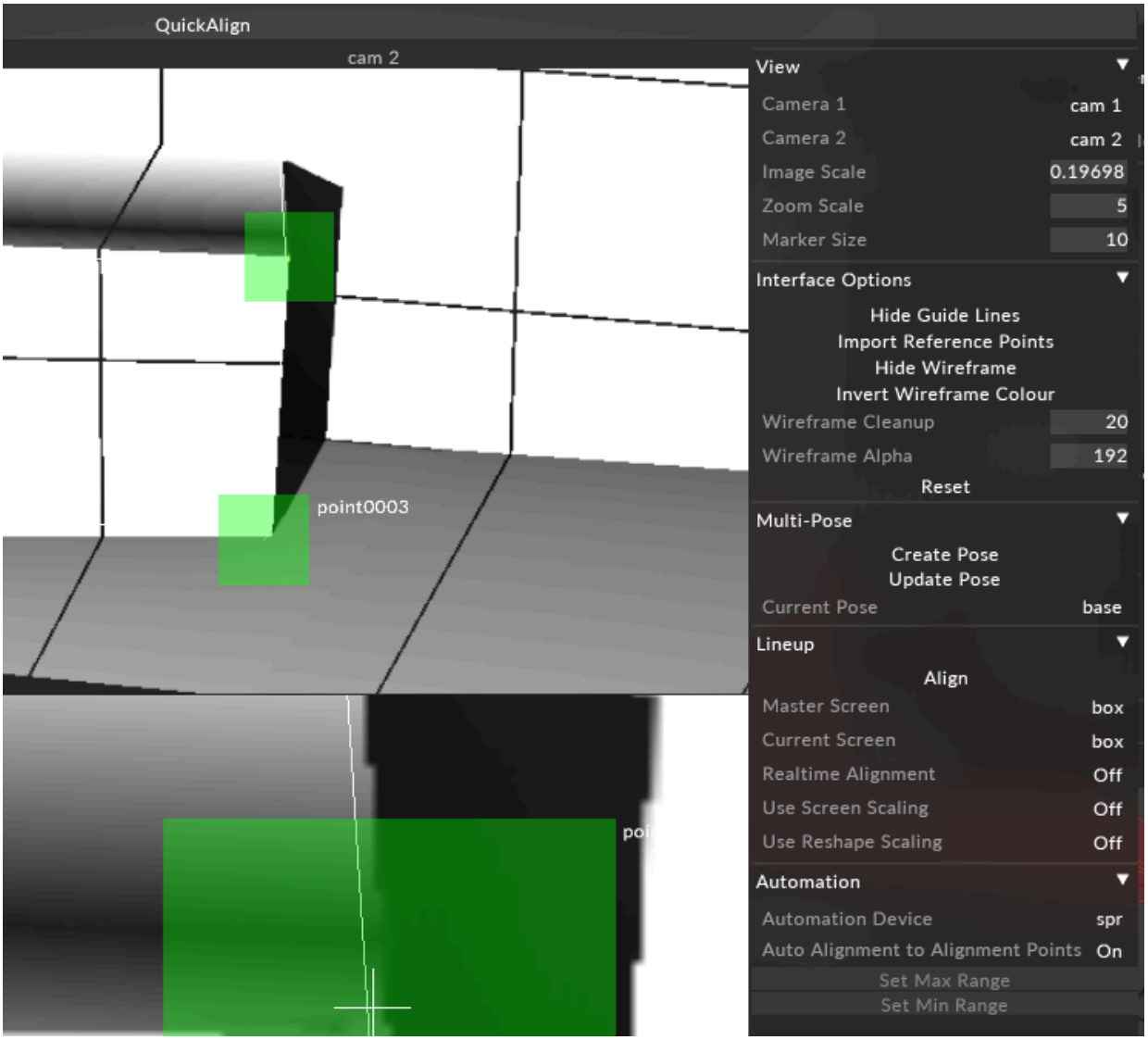
Set up automation axes for all moving screens

> **Note**
>
> All 6 axes are required even if the object is moving in a single dimension and no rotation is expected. This is because QuickAlign will calculate a composite rotation and translation of the surface. Ignoring rotation will mean the translation will not be correct. This will become especially apparent when the local origin of the mesh is far away from its centroid.

3. Perform a capture and calibration

1. Follow the regular OmniCal workflow to capture and calibrate.
2. Open the **Quick Align** editor.

4. Align the master screen and all static screens in base pose



Align the Master screen

1. Make sure that the **Current Pose** in QuickAlign is set to **base** and that the **Master Screen** and **Current Screen** are set to the actual master screen object in the visualiser.
2. Perform an alignment, as usual.
3. Repeat for all static screens.

4. You can only use reshape tools in the base pose. In other poses, the current mesh is being moved, but no further reshaping is supported.
   - Note that for other poses, the UI does not yet prevent you from turning alignment points into reshape points.
   - If you need to reshape a mesh for which you need to use the Multi-Pose workflow, then you can either do this by aligning and reshaping in the base pose, or doing it in a separate step, and export+re-import the mesh (probably preferred).

5. Create two poses for each moving screen

In most cases, it will be enough to create two poses: a minimum and maximum pose, each for the most extreme position in which the object will be visible / projected on. For example, if a projection surface is moved from left to right, then these are the most left and most right positions in which enough alignment points are still visible to produce a reliable alignment.

When a new pose is created, a new short capture is taken by OmniCal. This capture only contains alignment images that are used within QuickAlign to help with object alignment. These captures do not contribute to the calibration result, only to multi-pose alignment.

You can right-click on a pose to view the screens and automation data contained within it.

Create min Pose

1. Enable OmniCal camera discovery.
2. With the Quick Align editor open, ask your automation operator to move the moving object to its minimum visible position.
3. Click **Create Pose**.
   This will take a capture and present the camera images for the user to align to. It will also take a snapshot of the current automation input values.
4. Select the object which has just been moved by automation as the **Current Screen** to be aligned.
5. Perform an alignment of the screen at this minimum position.
   Each pose can contain multiple screens, so you can repeat this for each screen that is visible. Alternatively, you could create a separate pose per screen.
6. Click **Set Min Range** in the automation section of the Quick Align editor.
7. The minimum pose has now been set.

Create max Pose

1. Keep the Quick Align editor open, and ask your automation operator to move the object to its maximum visible position.
2. Repeat the same pose creation steps, but in step 6. select **Set Max Range** in the automation section of the Quick Align editor.

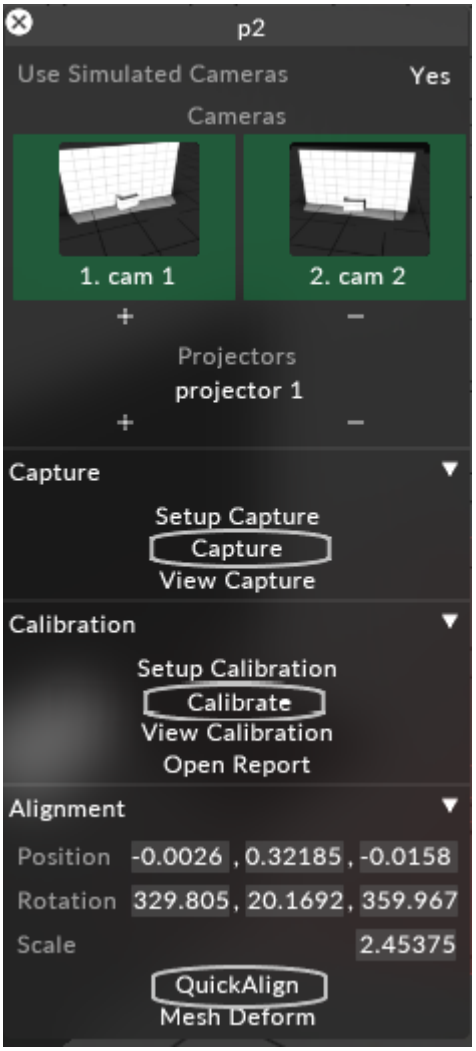
3. Click **Normalise Rotation** (always required unless a rotation of >= 180 degrees is required between the poses).
   For example, after a multi-pose alignment, if you get -0.2deg as the min output and 359.7deg as the max output, then the min output should be set to 359.7 or the max output should be set to -0.3deg. This is because the direction of rotation is ambiguous.
4. Close the Quick Align editor and ensure automation is engaged.

Extracting a mesh using OmniCal

Although not specific to the multi-pose workflow, generating accurate meshes simplifies calibration workflows.

Follow the regular OmniCal workflow:

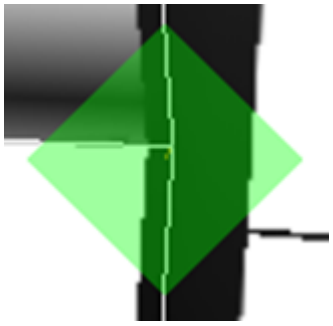
1. Create a plan
2. Capture
3. Calibrate
4. Align.



Capture, calibrate, QuickAlign in the OmniCal plan

Reshape the alignment points

Use the reshape tool (shift + drag alignment points to display a diamond) to match the shape of the actual object in real-world space. Once happy, export the reshaped mesh. These will be written to the output/ folder of the project, so you'll need to copy these in to the Mesh folder of your project.



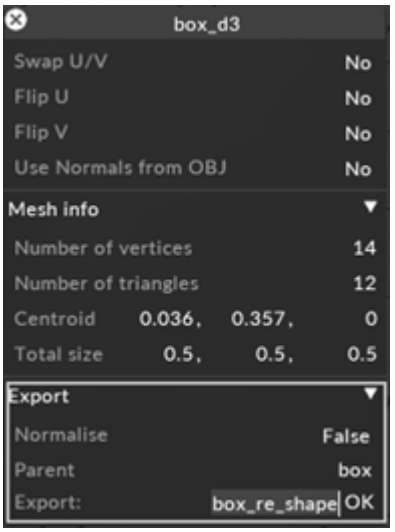
Reshape Alignment Point



Alignment Point

Export the mesh

To export a mesh, right-click the mesh in the screen editor to display the mesh editor. Select False for normalise (this will ensure the offset and rotation of your screen is correct when using the new meshes), type a name for the new mesh and click OK. The mesh will be exported to the following folder: `output/mesh/filename.obj`.



Export mesh editor

Reset the alignment points

1. Once the newly reshaped mesh is copied into the Mesh folder, Designer will ingest it into the system, and it should appear in the list of meshes for a screen. Assign these meshes to the projection surfaces.
   Once this is done, the alignment points will no longer be valid (because the mesh has changed).
2. Open the **Quick Align** editor.
3. Select **Reset Alignment** in Quick Align to remove the alignment points. If necessary, realign the screen using alignment points, but without using the reshape tools.



Reset alignment

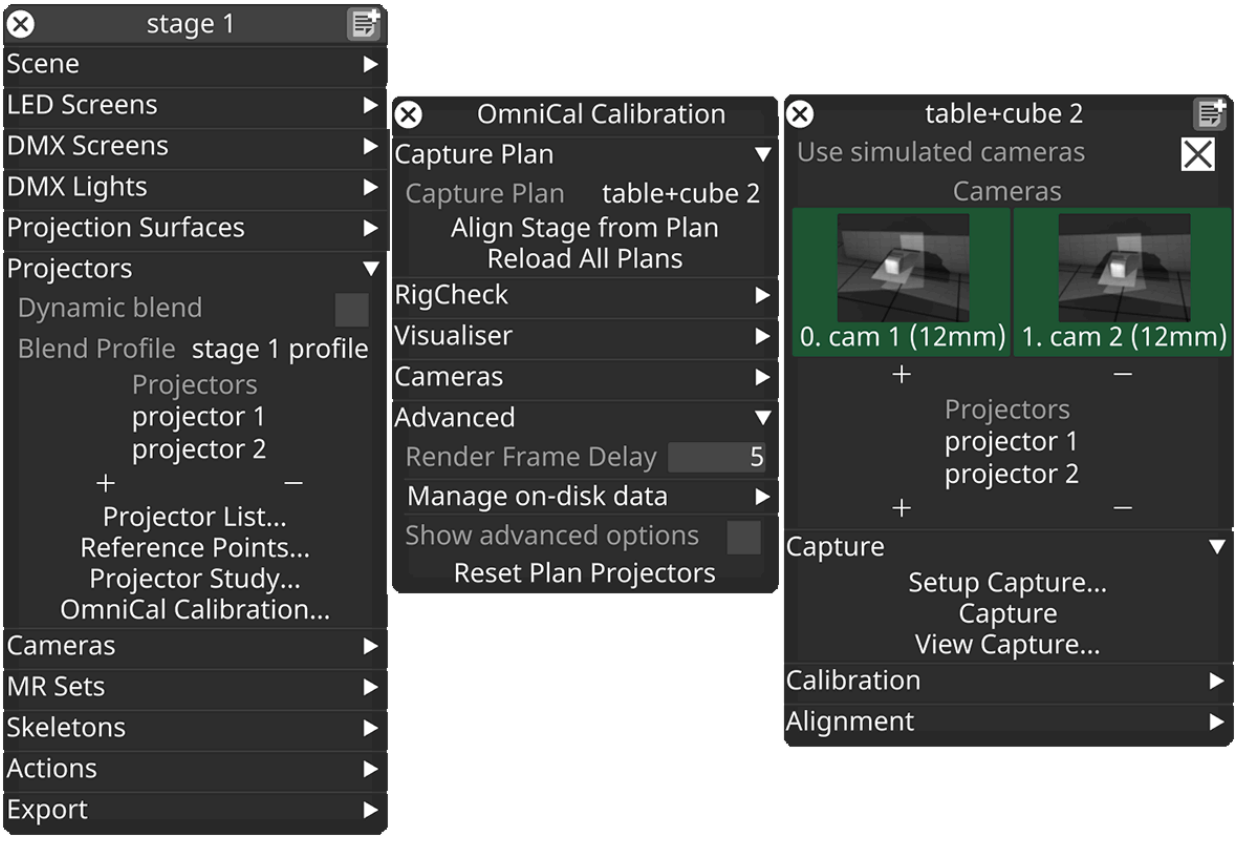
OmniCal Mesh Deform

Mesh Deform is an automatic process that enhances the existing 3D model by inserting new vertices based on the aligned point cloud, and adjusting existing vertices accordingly.

OmniCal Stage Plan

In the **Stage** editor, under **Projectors**, click **OmniCal Calibration** to open the **Calibration** editor. This widget controls the main OmniCal behaviour. Right-click on any **Capture Plan** to open its **OmniCalStagePlan** editor.

In the **OmniCalStagePlan** we recommend working through the sections from top to bottom: Capture, Calibration, Alignment, and Mesh Deform (if required).



Mesh Deform

If aligning and reshaping is not enough to make the projection match the physical object, then this is likely caused by shape differences that are not captured in the current 3D mesh. The mesh will need deforming to match the real world object. A typical situation is that *Reshape* has handled certain vertex deviations, but there are remaining differences like uneven surface

areas that are modelled as flat in the 3D mesh, or differences in curvature between the real world object and the 3D mesh.

Mesh Deform is an OmniCal feature that enhances the 3D mesh so it conforms to the captured point cloud. This is achieved by inserting new vertices into the mesh, and splitting triangles accordingly. The new vertices are based on points in the captured and aligned point cloud. Finally, a sophisticated smoothing algorithm adjusts all existing vertex coordinates according to their neighbours (including any newly inserted vertices).

Mesh Deform is able to deal with deformations in physical projection objects that are not or not adequately represented in the original mesh. For example it can accurately represent the actual surface of a curved screen, including any imperfections that arise from the construction process.

UV coordinates of original mesh vertices are not affected during Mesh Deform. Any inserted vertices get assigned UVs that are interpolated from their neighbours.

> **Note**
>
> Typically Mesh Deform is run once at the beginning of tech setup. As long as the physical projection surface does not change over time, the deformed mesh will be valid for any future capture and calibrations. See also **Point cloud size and detail** below.

Mesh Deform requirements

To use Mesh Deform correctly, a few requirements must be satisfied by the original mesh and the current state of the project:

- The original mesh must be “watertight” in areas that belong together.
  - If e.g. the North and East wall of a building are not connected in the mesh at their common edge (i.e. vertices are welded together), then they will likely get torn away from each other during Mesh Deform. Vertices with the same 3D position must be welded in the mesh, so that e.g. the top right vertex of the East wall is the same vertex as the top left vertex of the North wall, etc.
  - Holes (e.g. for windows of a building) are allowed, but it may be necessary to prevent unwanted deformations that could introduce twists to the window. This can be done by adding AlignmentPoints to the four corner vertices of each window and anchoring them (see **Anchor points** below).
- The OmniCal and Designer coordinate system must have been aligned, so the captured point cloud and original mesh overlap properly.
- For best Mesh Deform results it is recommended to first use Reshape to get the best possible match between mesh and point cloud.

Mesh Deform settings

Point cloud size and detail

Ideally, Mesh Deform is performed using an OmniCal capture that uses a high number of projected blobs (see grid size). The denser a point clouds is, the more detail it will provide for Mesh Deform. Therefore a larger grid size will provide a more accurate and higher quality deformed mesh. If the physical projection surface does not have many local deformations, then the default grid size is often sufficient.

Dense captures take a longer time to complete and calibrate, so only use a high grid size when a higher quality Mesh Deform is actually needed. When using high grid sizes, it is recommended to use a separate OmniCal plan with lower grid size for faster (e.g. daily) re-calibration. This is especially helpful if the calibration concerns multiple surfaces, but not all require deforming.

> **Note**
>
> It's easiest to start off with an OmniCal plan using a normal number of blobs (e.g. grid size of 48 or less), to first figure out correct camera placement and settings. If it's necessary to perform a higher quality Mesh Deform, then make a copy of this OmniCal plan and simply increase the grid size to a higher value before going through all steps from capture to Mesh Deform.

Distance threshold

This threshold denotes the maximum absolute distance between a point in the point cloud and the nearest mesh surface, along the point normal's direction. It is used to decide whether or not to add a new vertex for a certain cloud point (and deform its neighbours accordingly), or not. If the cloud point is further away from the current aligned (and potentially reshaped) mesh than this threshold, then Mesh Deform will not add a new vertex at the cloud point's 3D position.

This threshold is useful to exclude noisy or irrelevant data from the point cloud, e.g. trees in the foreground.

Default value: 0.1 m

Depending on the size and scale of a physical projection surface, this value may need to be increased or decreased to get better looking results.

Angle threshold

This threshold denotes the maximum angle between the normal vector of a cloud point and the normal vector of the nearest mesh surface. It is used to decide whether or not to add a new vertex for a certain cloud point (and deform its neighbours accordingly), or not. If the

angle is larger than this threshold, then Mesh Deform will not add a new vertex at the cloud point's 3D position.

This threshold is useful to prevent smoothing over edges and corners (e.g. right angles between floor and wall).

Default value: 30 degrees.

Anchor points

Anchor points in Mesh Deform are vertices that do not get modified during the smoothing operation. They are used as fixed guides. Any other vertices get interpolated based on the original vertex position, its non-anchor neighbours and the anchor points near it.

Mesh Deform treats all newly inserted vertices as anchor points, since they are directly derived from the captured point cloud. Other vertices (like mesh borders or alignment points) can also be treated as anchor points. This will influence the output mesh, since those vertices will keep their original aligned (or reshaped) position. This can make sense if there is for example a high confidence on the calculated location of reshaped points. But in most cases where Mesh Deform is required this is not desirable, so these options are not enabled by default.

Anchor points can be modified using these settings:

- **Lock mesh boundaries**: Unlocked / Locked
  - Whether mesh boundary vertices are treated as anchor points or not.
  - Has no effect on closed meshes without boundary edges, e.g. spheres.
  - Default: Unlocked.
- **Use alignment points**: No, Yes, If aligned in 2+ cameras
  - Whether Alignment points from QuickAlign will be treated as anchor points or not.
  - The third option only anchors alignment points that are aligned in more than 2 cameras. Having several camera intersections can provide a higher reliability on the calculated 3D vertex position.
  - Default: No.

Visualising Mesh Deform

You can preview the Mesh Deform results by changing the Point Cloud visualisation mode to **Deform** in the calibration editor. The results are updated in real-time so you can try out different deform settings using this. The lines indicate where each existing mesh vertex would be moved to. Newly inserted vertices will not be shown, though.

Using Mesh Deform

It is a good idea to try the default Mesh Deform settings first, and carefully inspect the result. If the deformed mesh has included or excluded cloud points incorrectly, then decrease or

increase the distance and angle parameters as required and try again.

1. Click **Mesh Deform...** from the Alignment tab of the Plan editor.
2. Select the screen you wish to deform by left-clicking the **screen property** in the deform editor.
3. Adjust parameters for this screen, as required.
4. Click **Deform**.

Mesh Deform Performance

Running a Mesh Deform operation may take a while, depending on the number of vertices and triangles in the original mesh, as well as the size of the captured point cloud.

After clicking **Deform**, the process will run in the background until it is complete. This may take several seconds (for simple meshes) up to tens of minutes (for complex meshes and dense point clouds). Mesh Deform is very CPU intensive, so although it's possible to continue using Designer while this is running, the impact on performance is very noticeable.

OmniCal RigCheck

RigCheck is the tool for quickly re-calibrating without advanced knowledge of the rest of the OmniCal system. It is primarily used by operators at fixed installations to re-calibrate shows after minor external changes, by comparing live camera images to those from a previous Capture to check whether cameras or projection surfaces have moved. If so, the user can adjust the alignment reference points by dragging them into the correct positions.

Accessing RigCheck

1. Open the stage editor by right-clicking Stage in the dashboard, or by right-clicking the floor in the visualiser.
2. Left-click **OmniCal Calibration** to open the OmniCal Calibration editor.
3. Ensure that you have a plan already calibrated by a trained operator.
4. Left-click **Check cameras**.

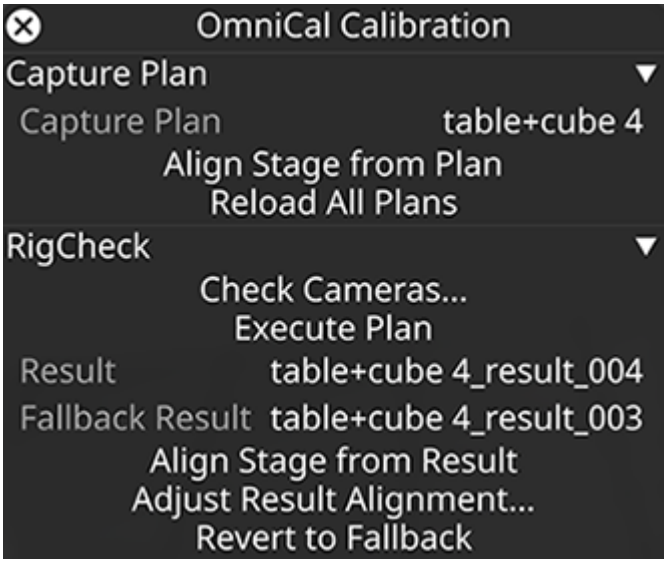
> ⚠ **Warning**
>
> - The stage should be clear.
> - No changing light levels.
> - No people walking across stage.

Workflow

The RigCheck workflow is used after a full OmniCal capture, calibration and alignment has been completed using the Plan workflow.

In RigCheck, the current plan is used as a template to repeat all required OmniCal steps with the same settings, but at a later time. The original plan itself is not modified in this workflow. Each RigCheck result stores its own set of settings and data. RigCheck allows minor alignment adjustments that differ from the original Plan to account for minor changes in projector or camera movement. It is therefore suited for recalibration of proven setups, e.g. when being run by day-to-day operators on fixed installations.

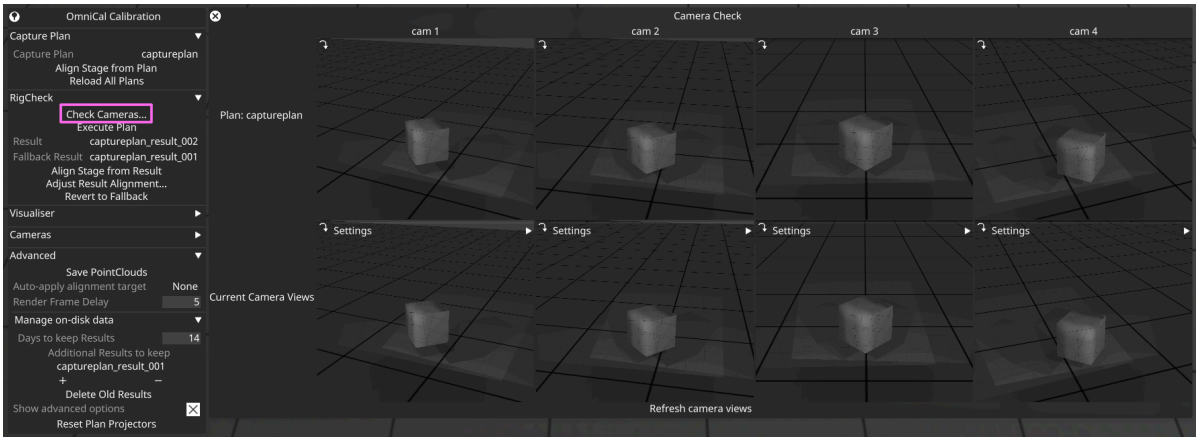
The **RigCheck** separator in the main **OmniCal Calibration** window groups all relevant functionality to prepare, run or align a RigCheck.



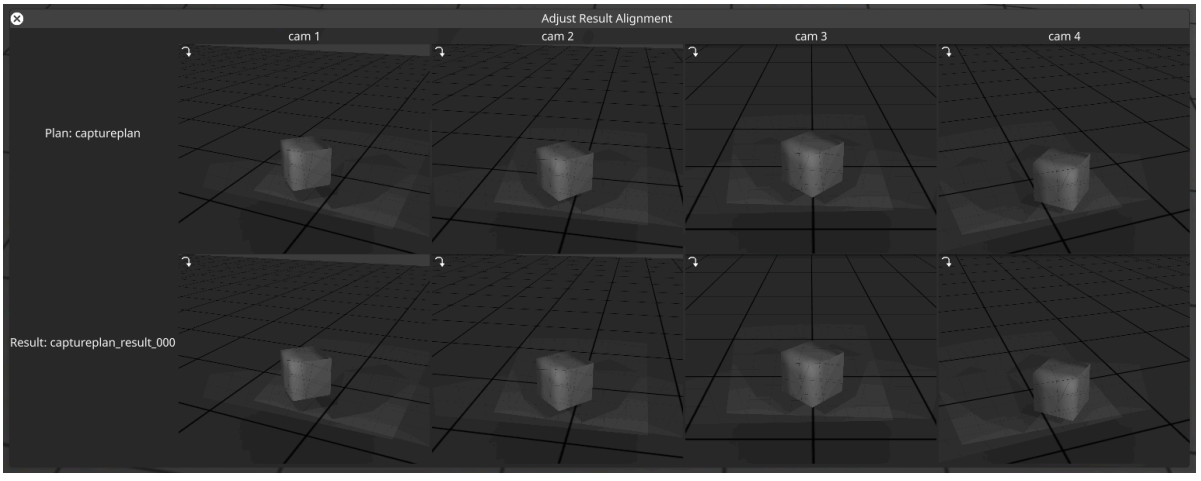
OmniCal Plan with several RigCheck Results.

Using RigCheck

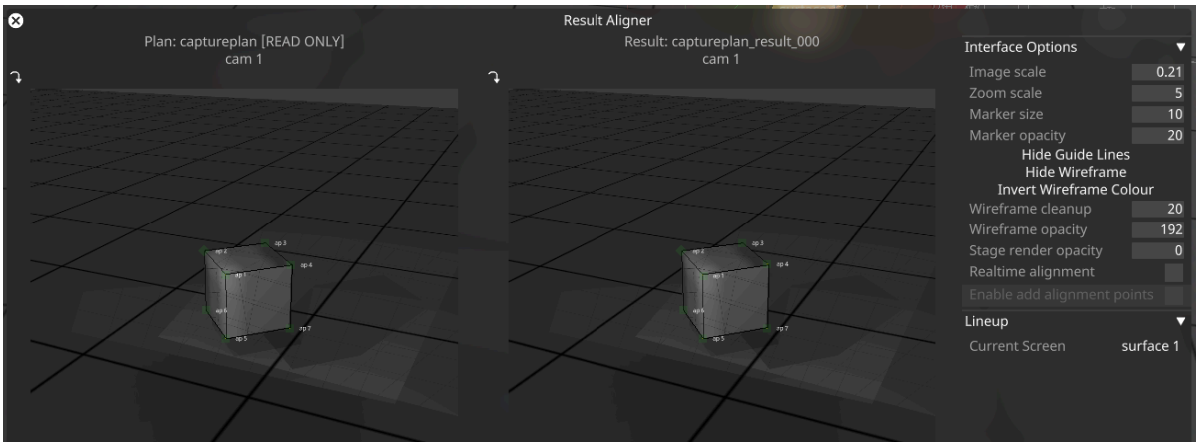
1. Ensure that you have a plan already calibrated by a trained OmniCal operator. It should be selected in the *OmniCal Calibration* editor.
2. Left-click **Check Cameras** in the OmniCal Calibration editor.
3. Verify there are no errors when the **Check Cameras** window opens. Errors can include the following:
    - **a.** Cameras are unavailable.
    - **b.** Camera names have been modified in the plan. The plan capture is used as a reference for RigCheck. Any changes to cameras names after a completed plan capture will invalidate it.
4. The **Check Cameras** window will open which shows camera pairs consisting of the original plan capture and a snapshot capture of the current stage. You can take a new snapshot using the Refresh Camera Views button.



5. Ensure that the camera views match:
    - **a.** They don't need to be identical, but the coverage of projection surfaces should match and they should be in approximately the same positions.
    - **b.** The brightness of the images should also match. The exact camera settings specified in the plan (like exposure time) will be used here, so any differences would be either due to the physical aperture or focus of the camera lens, or the lighting environment. It is therefore recommended that the focus and aperture of machine vision cameras be locked in place for repeatability.
6. Once you are happy with the camera views, close the **Check Cameras** window.
7. Left-click **Execute Plan**. This will perform a capture, calibration, alignment and optionally a mesh deform (if it was configured to run automatically in the plan's deform settings). A new Result object is created which contains the capture, calibration etc data.
8. Ensure that the correct Plan and Result are selected in the OmniCal Calibration editor.
9. Left-click **Align Stage from Result** in the OmniCal Calibration editor. Note that this will apply the result's alignment to the stage (if it hasn't been applied already). This will update both projectors and projection surfaces if they have alignment points. If Align Stage from Result is greyed out a tool tip should tell you the reason, which can include the following:
    - **a.** The Plan has no capture or calibration.
    - **b.** The Result has no capture or calibration.
10. Verify there are no errors when the **Adjust Result Alignment** window opens. Errors can include the following:
    - **a.** The camera names in the Plan and Result don't match. The Plan may have been modified since the Result was created.
11. The **Adjust Result Alignment** window will open which shows camera pairs consisting of the original Plan capture and the Result capture that was taken.



12. Left-click on a camera image to open the **Result Aligner** window. If any **Mesh Deforms** have been applied these will be undone in order for alignment points to be editable.



13. On the left hand side is the camera view from the original Plan (read only) and on the right the Result. You can then update the alignment points on the right to match the left. This will update the alignment on the stage as you do it. The controls are identical to **QuickAlign**, except that it's not possible to add/remove any alignment points.
14. Select the next screen by pressing the button at the bottom of the camera views or selecting it from the surfaces drop down and repeat the process of checking and adjusting alignment points.
15. Repeat this for all cameras. You could in theory only check cameras or projection surfaces which you know to have moved.
16. Close the **Adjust Result Alignment** window. You will be asked if you'd like to re-apply the **Mesh Deform**.

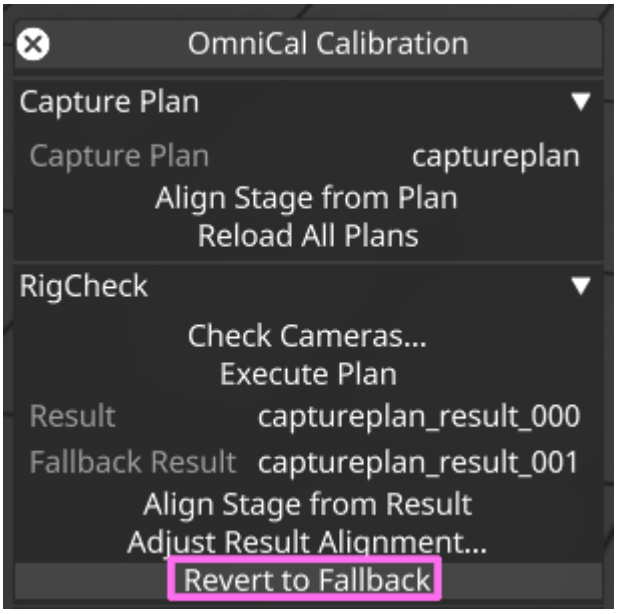
Reverting to a Fallback Result

Designer r32.4 added a convenient way to revert the OmniCal calibration state to a previous RigCheck Result, called the **Fallback Result**. This is useful when a RigCheck failed or produced suboptimal quality.

The fallback should typically be set to the most recent RigCheck Result with the best visual quality, as determined by an OmniCal operator after a visual inspection of the scene. The **Fallback Result** must be fully calibrated and aligned for all projectors, with no errors.

Click the **Revert to Fallback** button to switch from the current **Result** to the **Fallback Result**. This will revert the OmniCal state and update the calibration and alignment of all affected

projectors to the newly selected current **Result**.



Revert to Fallback button

RigCheck API

Essential parts of the RigCheck workflow are now also available via **API endpoints** for the Designer session.

> **Note**
>
> Alignment adjustments are currently not supported via the API. If such adjustments are needed, then these can usually be done by logging into a Designer session remotely.

Managing RigCheck data on disk

OmniCal RigCheck Result data on disk

Just like in the Plan workflow, RigCheck also stores a mix of data on the hard disk, rather than in the .d3 project itself.

The RigCheck data includes:

- OmniCal settings
- Captured images
- Calculated blob detection results
- Calibration results
- Alignment data
- Report data

This RigCheck data is stored in subfolders of the Designer project's `internal/opticalresult/` folder. Every RigCheck has its own set of data, and its subfolder has the same name as the associated "Result" object in Designer.

The data in the **internal** subfolders is managed together with the corresponding Result object in Designer: When the RigCheck Result object is deleted, and the trash in Designer is cleared, any associated folder (including images, config and result files) is also deleted from the hard disk.

Delete old and unused RigCheck Results

If a project has many RigCheck Results in Designer, then this may cause the hard disk to fill up over time. To free up disk space, users can move RigCheck results that are no longer needed to the trash, and then empty the trash.

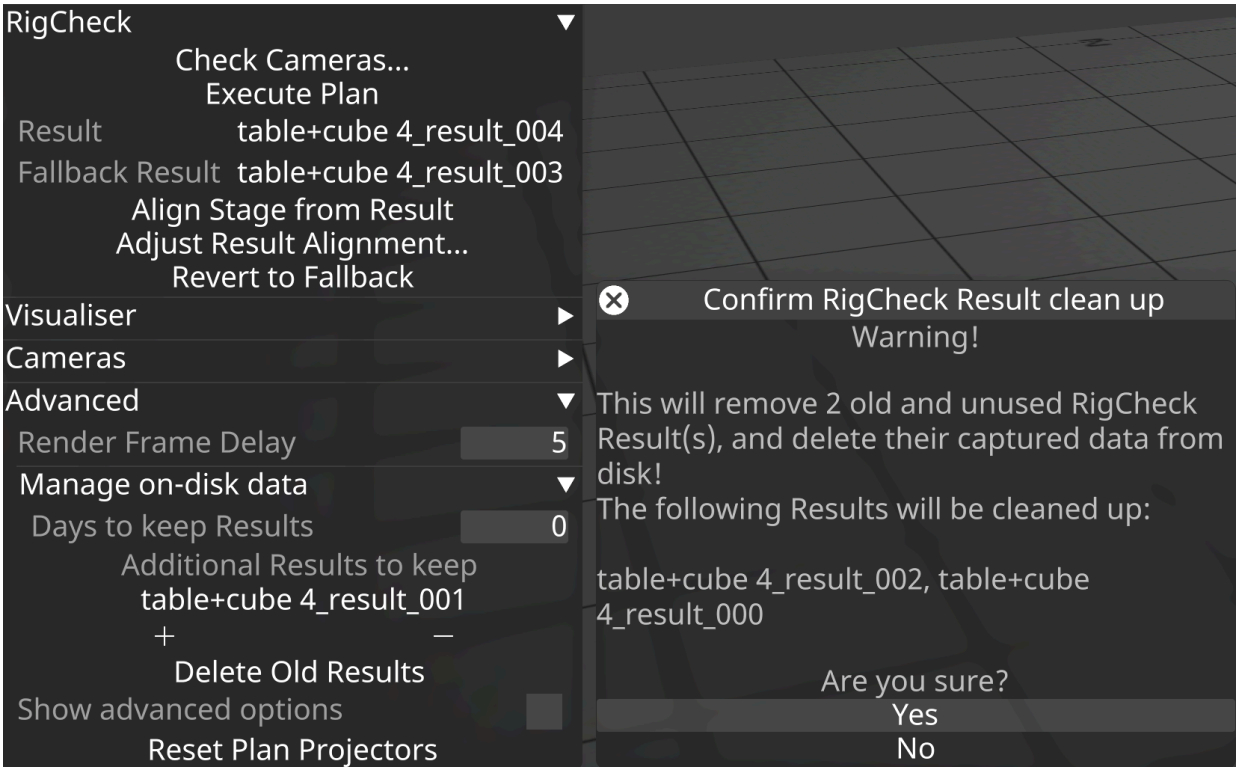
For projects that recalibrate on a regular basis (e.g. via API calls) this manual way of deleting RigCheck data can be too slow and error prone. Designer version r32.4 introduces an easier way to delete any RigCheck Results that are old and unused, via the **Delete Old Results** button or **API endpoint** `omnical/rigcheck/deleteoldresults`.

The API endpoint or button will delete any RigCheck Results that are both:

- Older than the **Days to keep Results** value configured in Designer.
- Not in use.

The following Results are considered "in use":

- Currently selected **Result**.
- Currently selected **Fallback Result**.
- Any result in the backup list, see **Additional Results to keep**.



*Example usage of **Delete Old Results**.*

Manage on-disk data for API usage

Users can configure **Days to keep Results** and **Additional Results to keep** in Designer to suit their needs, but they cannot currently be modified by the API.

The **API endpoint** `omnical/rigcheck/acceptresult` automatically adds the current Result to the **Additional Results to keep** backup list. This allows for cleanup automation by API callers, without the need to interact with Designer directly.

EVO

Disguise is a workflow platform, designed to be flexible and enable you to interoperate multiple functions within the platform to create your own workflows. One example of this is a collection of workflows known as EVO.

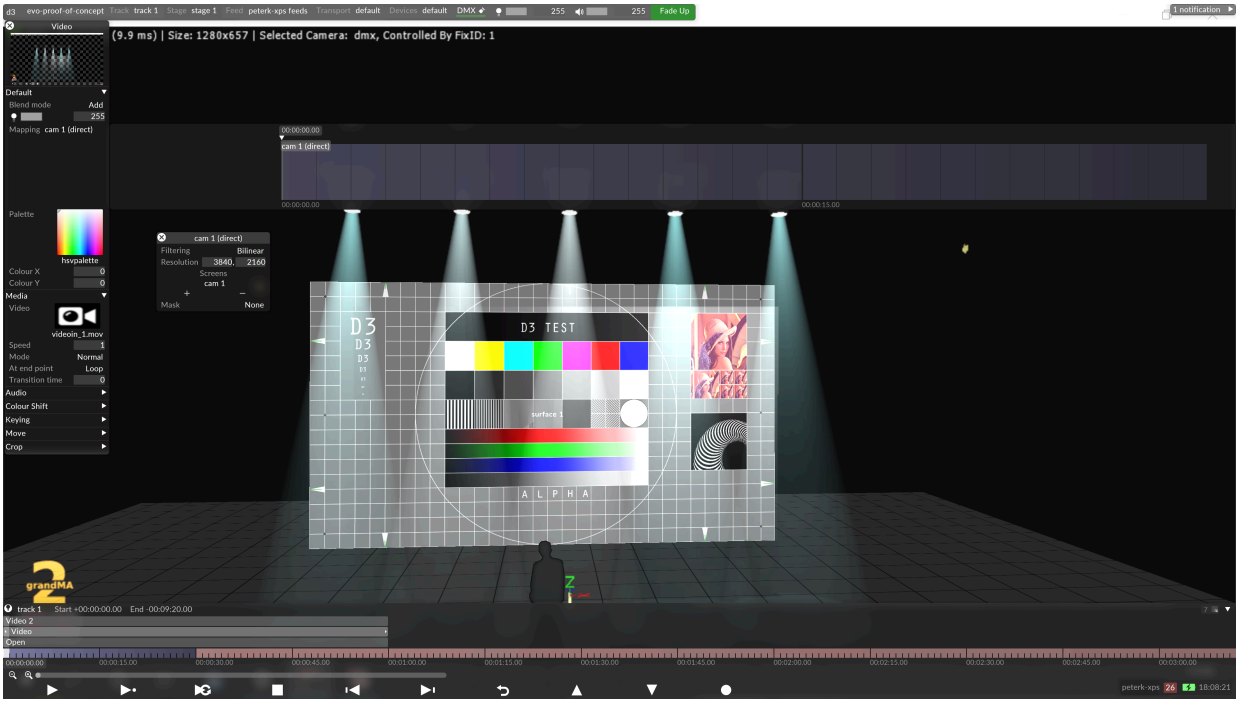
EVO (External Visualiser Overlay) is in essence two parts. An incoming video stream (NDI®) from a third party visualiser, and sharing of camera coordinates between the Disguise camera and the external visualiser's camera.

By combining these two workflows, you are able to create a seamless link between the two systems - essentially visualising the lighting and video systems together in one viewport.





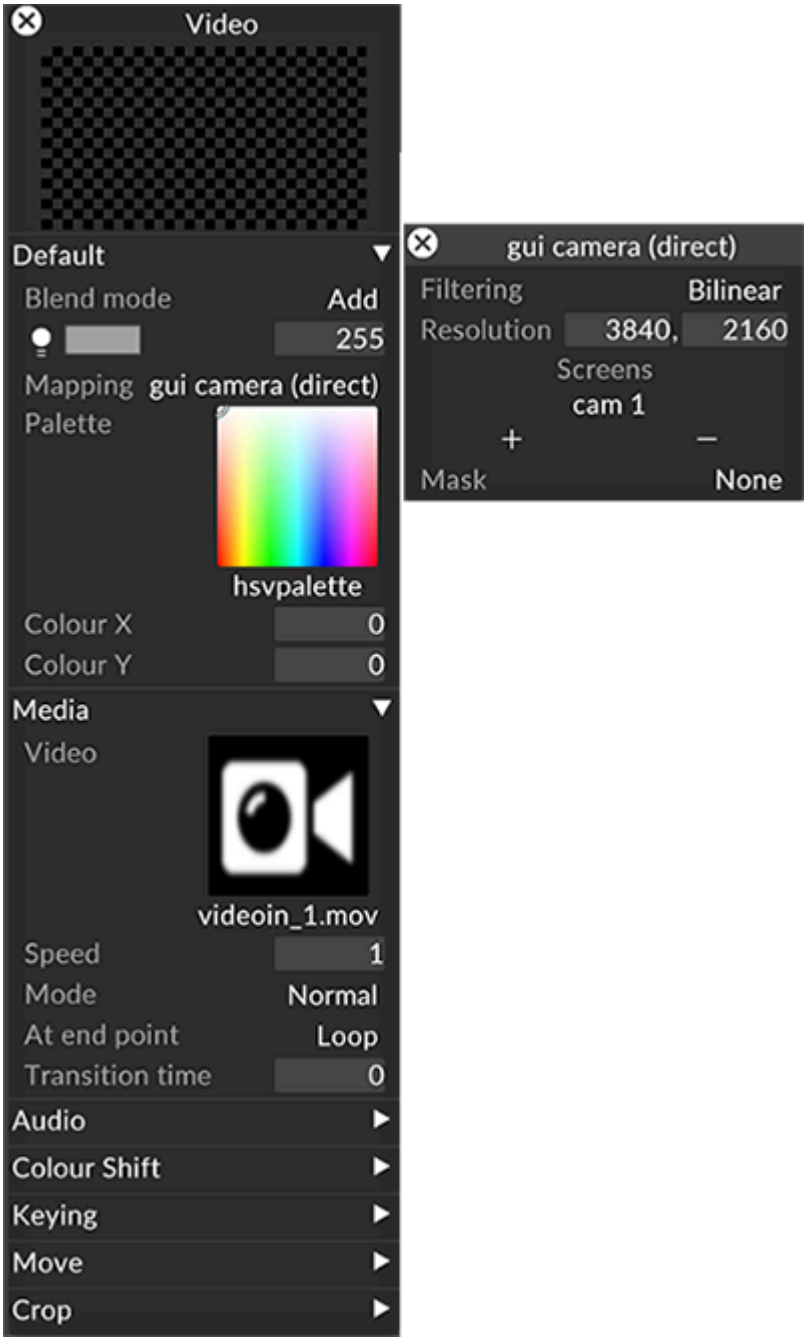
Watch online: https://www.youtube.com/watch?v=hGaobtrId40



NDI® Video Stream Setup

To set up the NDI® stream:

1. Start the **NDI® Screen Capture** on the lighting visualiser PC.
2. In the third party visualiser, set all video surfaces & props to render as black. It might be helpful to use the Export Stage to FBX option in Designer to export the stage directly into the visualiser, to ensure all objects are scaled identically.
3. Start the Designer project.
4. In Designer, create a new camera and position it as required. This will become your main visualiser camera, so you may also need to increase the resolution if you are running a 4K GUI (the new camera will default to 1920 x 1080.
5. Select the visualiser camera to be that new camera in the Stage menu.
6. In the Video Input Patch, map one of the Video Inputs to the incoming NDI® stream from the visualiser PC. Check with the preview function that this is routing correctly.
7. Add a new Video layer to the timeline.
8. In the Video layer, select the Video Input clip as the media asset.
9. Set the Video layer to the **Add** blend mode.
10. Create a new **Direct mapping** and add the new camera you created to that mapping.
11. Select the new mapping as the mapping for the video layer.



At this point the incoming NDI® stream from the external visualiser will be overlaid from the Disguise visualiser. If you manually line up the two cameras, this is all that is needed - but using the tracking and control modules in Designer it's possible to link the two cameras together, via open protocols (DMX, OSC etc).

Camera Position Setup

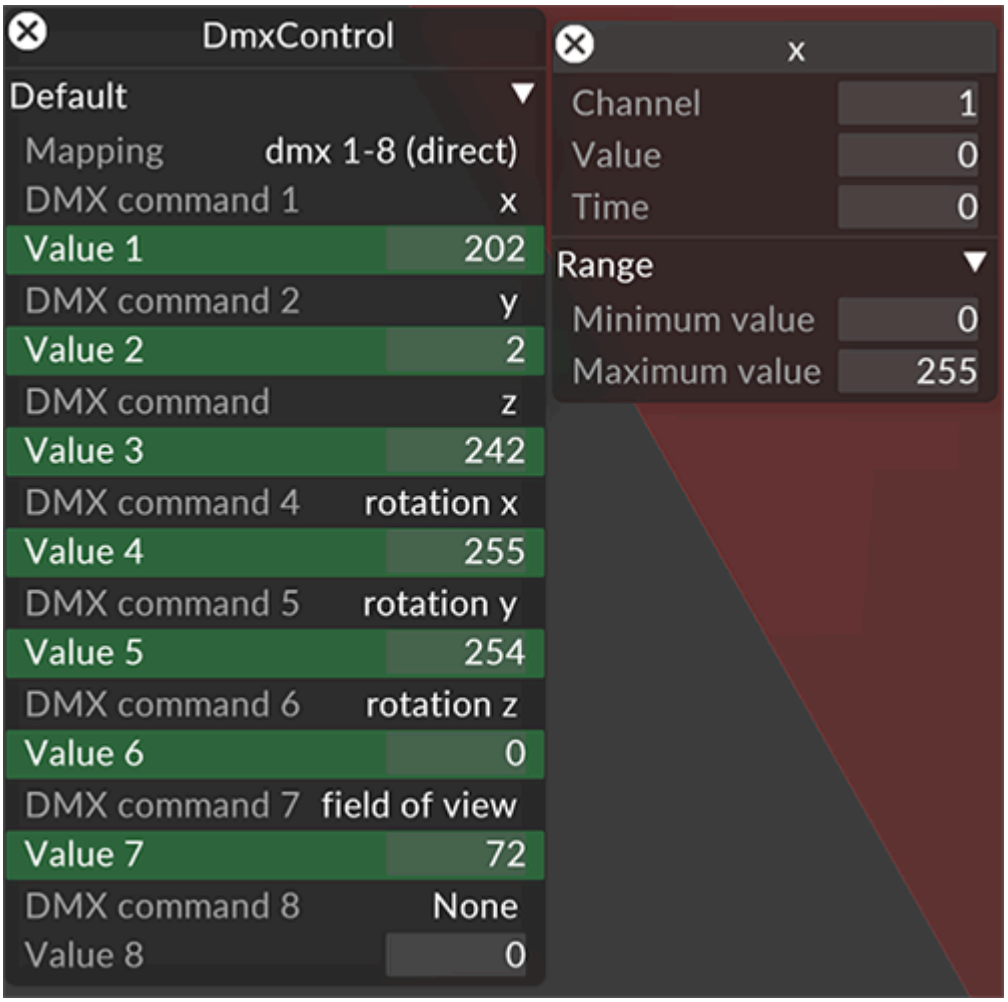
To set up the camera position - there are two options - either Disguise can receive the camera position from the visualiser, or the visualiser can receive the position from Disguise.

Below are generic instructions for both setups. Specific instructions for each visualiser are available on the relevant page for the specific visualiser you are working with.

Visualiser receiving position from Disguise - via DMX

In this setup, moving the Disguise camera will move the visualiser camera, leaving the Disguise operator in control of what is being looked at.

1. Add a **DMX device** to the project, and patch it to Output DMX.
2. Create a **DMX lights screen** and assign it to create an appropriate number of DMX addresses.
3. Add an **DMXLightsControl** layer to the timeline.
4. Select the DMX lights screen as the mapping for the DMXLightsControl layer.
5. Use an arrow to connect the control layer to the stage camera (the expression syntax created is `camera:{camera name}.offset.x)`.
6. Set up the appropriate commands to send from Designer to the visualiser - note that the DMXLightsControl layer sends only 8 bit values so you will need to convert these to 16 bit or 24 bit depending on the external visualiser requirements. See below.
7. Apply any scale factors needed in the expression to center the two worlds. You will need multiple DMXcontrol modules to send all the properties of the camera.
8. Patch the external visualiser according to the data stream you created.



> **Note**
>
> World Offsets - It might be helpful to set up Designer camera as a **Child** of a null object (prop). This means that you can set the world offset using the position and rotation properties of the null object, rather than modifying the expressions (above). Create the prop, then use **Add Child** to select the camera as a child of the prop. When the position of the prop is adjusted, the camera will move by the same relative value. No mesh needs to be selected on the prop.

Disguise receiving position from the visualiser - Open Layer workflow

1. Add a **DMX device** to the project file, and patch it to Input DMX.
2. Add an **Open layer** to the timeline.
3. Hold down Alt and drag an arrow between the Open Layer and the Cameras position and rotation properties. This will connect the Open layer to the camera Position & Rotation, enabling control of them from the timeline.
4. Right-click on each property and use an expression to connect each of these properties to the appropriate incoming DMX value - you may need to scale these values to match the world-scaling in the third party visualiser.

Disguise receiving position from the visualiser (or lighting console) - PositionReceiver workflow

For specific visualiser setup instructions, refer to the relevant page for the visualiser you are working with.

Specific visualiser setup instructions

- MA3D
- Capture
- Depence

MA3D

The MA3D and MA3 Visualiser from **MA Lighting** are compatible with the EVO workflow.

NDI® setup

1. Once the machine with the MA onPC and MA3D is ready, run **NDI® Scan Converter**. This app will convert the outputs of the GPU in video streams over IP.
2. Open the NDI® studio monitor and check that screen with the MA3D is working as an NDI® stream.

Disguise setup

1. Open a new project file, remove the projector out of the stage and create a new **virtual camera**.
2. Right-click on **Devices > Video input patch**. Navigate to **video.in1 > input configuration** and select the NDI® stream of the MA3D. Click on Start preview to check the stream, then stop the preview.
3. Create a **DMX device**, and use the IP address of the MA onPC or MA desk, and check the data with the DMX monitor.
4. Create a **PositionReceiver** Device and build new expressions to control the position x,y,z and rotation x,y,z of the virtual camera within Designer using the data coming from the MA lighting desk.

Expressions

Expressions are used within Designer to calculate the mathematical values needed to align the virtual worlds of both visualizers. The information that follows will be used in the expressions that need to be created within Disguise:

- The MA camera controller has a range from -1000m to 1000m for the x,y,z positions and a range of -720 to 720 degrees for the pan and tilt.
- The position x,y,z of the MA camera control has a resolution of 24bits for the x,y,z, pan and tilt. Expressions within Designer only support 16bits, meaning that the camera movement in Disguise will be smooth using 16bit data.
- The ‘position y’ of the MA camera controller is the ‘offset z’ value of the virtual camera within Designer; The ‘position z’ of the MA camera controller is the ‘offset y’ value of the virtual camera within Disguise.

These are the expressions you need to build:

Camera offset x



Camera offset y



Camera offset z

camera z	
Expression	dmx16:6.4
Object	camera evo
Property	offset.z
Minimum input	0
Maximum input	65536
Minimum output	-1000
Maximum output	1000
Output	-15.014

Camera pan (rotation y)

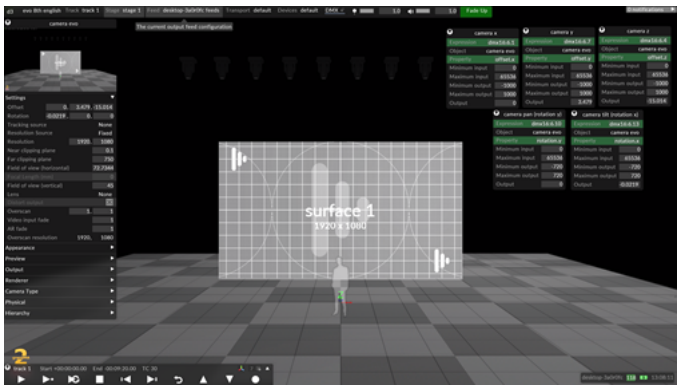
camera pan (rotation y)	
Expression	dmx16:6.10
Object	camera evo
Property	rotation.y
Minimum input	0
Maximum input	65536
Minimum output	-720
Maximum output	720
Output	0.70312

Camera tilt (rotation x)

camera tilt (rotation x)	
Expression	dmx16:6.13
Object	camera evo
Property	rotation.x
Minimum input	0
Maximum input	65536
Minimum output	-720
Maximum output	720
Output	-1.4502

If the expressions are correctly built the new virtual camera within Designer will move to the same x,y,z position location of the MA3D camera.

1. Create a new Video layer, map it to surface 1, and add some media.
2. Create another Video layer, change the blend mode to **Add**, and select **video-in** 1 as media; the thumbnail will display the MA3D NDI® stream with a checkerboard. Make a new **direct mapping** for your new virtual camera.
3. Go to **Stage > Visualizer** Camera and assign to the new virtual camera.
4. Confirm that the Disguise camera and the MA3D camera is aligned.
5. Open the virtual camera properties editor and change the background colour to black.



Once you have finished all the steps you can make the stage plane in MA3D invisible; it will not be needed once the alignment of the cameras is completed. You can start patching lights on the grandMA2 and add screens in Disguise to create your show.

Controlling the Disguise camera from MA3D

General Information

The MA 3D world is scaled between -1000m and +1000m, and rotation between -720 and +720°. DMX values for control are output as 24 bit control signals, using 3 DMX channels (bytes). Disguise only supports the input of 8 or 16-bit values, so we take the most significant bits from these values.

Using these real world values, we are then able to plug them into the following formulas to create the expressions needed to connect an external visualiser camera to the internal camera within Disguise:

```
{world centre offset in meters/degrees}+(dmx16:universe.address/65536)*{world size in meters/degrees}-{world size in met
```

GrandMA 3D Camera Control Expressions

Property	Expression
Position X	`1000+(dmx16:1.1/65536)*2000-2000`
Position Y	`1000+(dmx16:1.7/65536)*2000-2000`
Position Z	`1000+(dmx16:1.4/65536)*2000-2000`
Rotation X	`720-(dmx16:1.13/65535)*1440-1440`
Rotation Y	`720+(dmx16:1.10/65536)*1440-1440`
Rotation Z	No expression (GrandMA 3D does not support Rotation Z via DMX)

Note: Y and Z are flipped in GrandMA 3D, so we pick up opposing DMX values for Position Y.

Field of View

Set this manually.

MA 3D measures field of view as half horizontal value compared to Designer (eg if MA 3D is 22.5°, Designer field of view is 45°.

In the above expressions, the constants 1440 and 2000 are derived from the world scale of the GrandMA 3D scene.

- 2000 is the scale factor between the GrandMA 3D world positions into meter scaling.
- 1440 is the scale factor between the GrandMA 3D world rotations and degrees.
- The above constants can be modified for integration with other visualisation systems.
- Note that these expressions are only reading the top two bytes (16-bits) of the 24-bit values.
- At this point when you move the external visualiser camera, Designer visualiser camera will move too.

Using Disguise to control the MA3D camera

Disguise only supports the sending of 8 bit values from the DMXLightsControl layer, so we need to use expressions to split the 24 bits into separate bytes:

DMX Channel Assignments for Camera Control

Property	DMX Channel	Expression
High Byte Position X	1	`(((camera.offset.x+1000)*8388.608)/65536)`
Mid Byte Position X	2	`(((camera.offset.x+1000)*8388.608)%65536)/256)`
Low Byte Position X	3	`(((camera.offset.x+1000)*8388.608)%256)`
High Byte Position Y	4	`(((camera.offset.y+1000)*8388.608)/65536)`
Mid Byte Position Y	5	`(((camera.offset.y+1000)*8388.608)%65536)/256)`
Low Byte Position Y	6	`(((camera.offset.y+1000)*8388.608)%256)`
High Byte Position Z	7	`(((camera.offset.z+1000)*8388.608)/65536)`
Mid Byte Position Z	8	`(((camera.offset.z+1000)*8388.608)%65536)/256)`
Low Byte Position Z	9	`(((camera.offset.z+1000)*8388.608)%256)`
High Byte Rotation X	10	`((((camera.rotation.x*-1)+720)*((256*256*256)/1440)/65536)`
Mid Byte Rotation X	11	`((((camera.rotation.x*-1)+720)*((256*256*256)/1440)%65536/256)`
Low Byte Rotation X	12	`((((camera.rotation.x*-1)+720)*((256*256*256)/1440)%256)`

Property	DMX Channel	Expression
High Byte Rotation Y	13	`((((camera.rotation.y*-1)+720)*((256*256*256)/1440)/65536)`
Mid Byte Rotation Y	14	`((((camera.rotation.y*-1)+720)*((256*256*256)/1440)%65536/256)`
Low Byte Rotation Y	15	`((((camera.rotation.y*-1)+720)*((256*256*256)/1440)%256)`
High Byte Zoom	16	-
Mid Byte Zoom	17	-
Low Byte Zoom	18	-

Note: Y and Z are flipped in MA 3D world. Y is inverted in MA3D so we invert the expression with the negative conversion on the bytes.

Constants

- In the above expressions, the constants are derived from the world scale of the GrandMA 3D scene.
  - Position constant: **8388.608** is the scale factor between the GrandMA 3D world positions (-1000m to +1000m = 2000m) and the 24-bit DMX value (256 x 256 x 256) / 2000 = 8388.608.
  - Rotation constant: ((256*256*256)/1440): **11650.84444444** is the scale factor between the GrandMA 3D world rotations (-720 to +720 degrees = 1440 degrees) and the 24-bit DMX value (256 x 256 x 256) / 1440 = 11650.84444444

The above constants can be modified for integration with other visualisation systems.

You will need to patch these same values in the external visualiser.

At this point when you move the Disguise camera, the external visualiser camera will move.

The only final adjustments that need to be set manually are field of view.

Capture

Capture can be integrated into the EVO workflow. These are suggested expressions for controlling the visualizer camera within Capture from Disguise based on the following information available from Capture documentation.

```
bit depth = 16max value = 2 pow(bit depth) = 65536

output min = -32768 output max = 32768 output range = output max - output min = 65536

output to bit ratio = max value / output range = 1
```

Capture Camera control X, Y, Z:

Value	Expression
1	`(camera:camera.offset.x*100+32768%65536)/256`
2	`camera:camera.offset.x*100+32768%256`
3	(unassigned)
4	`(camera:camera.offset.y*100+32768%65536)/256`
5	`camera:camera.offset.y*100+32768%256`
6	(unassigned)
7	`((camera:camera.offset.z*-1)*100+32768%65536)/256`
8	`(camera:camera.offset.z*-1)*100+32768%256`

Camera Control Rotation:

Value	Expression
1	`(camera:camera.rotation.y*182+180*182.044%65536)/256`
2	`camera:camera.rotation.y*182+180*182.044%256`
3	(unassigned)
4	`(camera:camera.rotation.x*182+180*182.044%65536)/256`
5	`camera:camera.rotation.x*182+180*182.044%256`
6	(unassigned)

Value	Expression
7	`(camera.camera.rotation.z*182+180*182.044%65536)/256`
8	`camera:camera.rotation.z*182+180*182.044%256`

Note

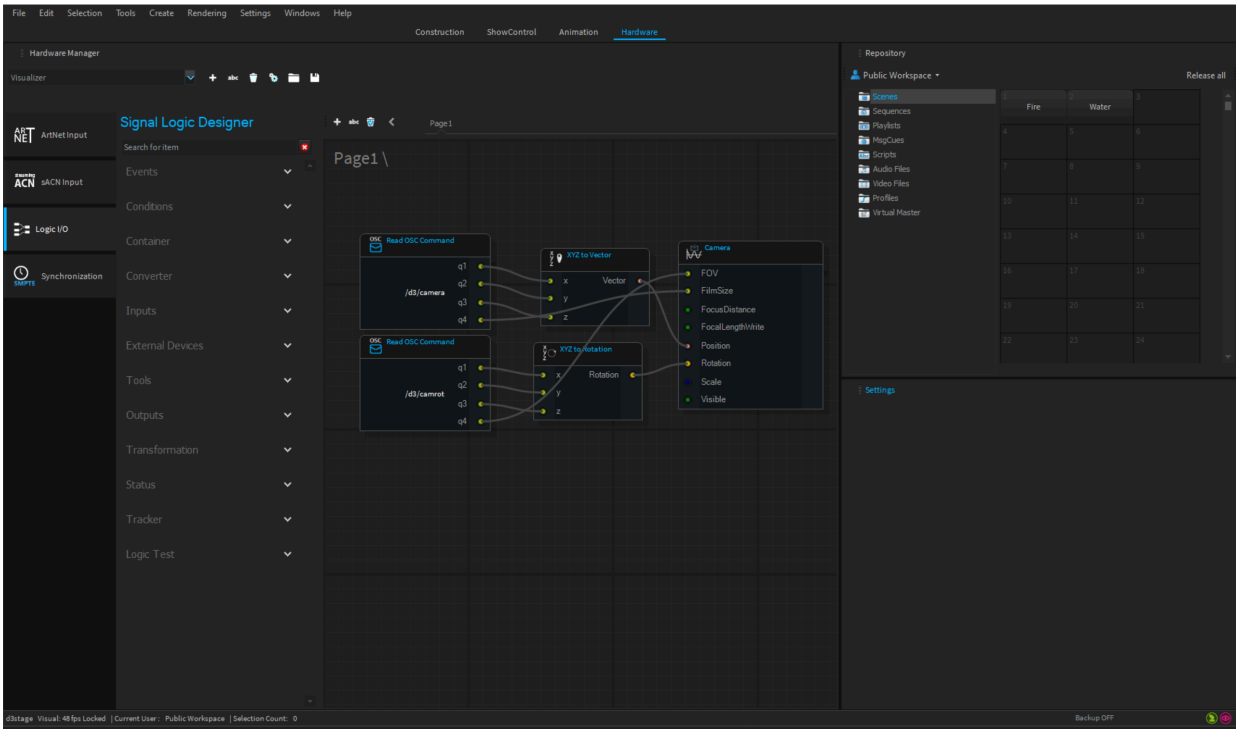
- The Camera position range is -32768 to 32768 for X, Y, and Z; this value is in cm*.
- The Camera rotation range is -180 to 180.
- Capture requires 16bit resolution.
- Z axis is inverted in Capture so you need to use camera z * -1.
- Expressions must be converted to Metres, so we use *100 in the expressions to convert cm to m.

Depence

Depence can be integrated into the EVO workflow and overlaid on top of the Disguise visualiser. These are suggested expressions for controlling the visualizer camera within Depence from Disguise. To do so requires the Depence R3 Animate module as well as the Stage and other relevant modules your project may require.

Depence R3 Logic I/O setup

1. In Depence, create a new Logic I/O page within the Hardware tab, and add a **Read OSC Command** node for each of the parameters you wish to control.
2. In the 3D hierarchy, select your camera and drag it into the Logic I/O page.
3. Add an **XYZ** to Vector node between the Read OSC command and the Position property of your camera.
4. Add an **XYZ** to Rotation node between the Read OSC command and the Rotation property of your camera.
5. Link through any of the other parameters you wish to control, such as FOV and Sensor Size.

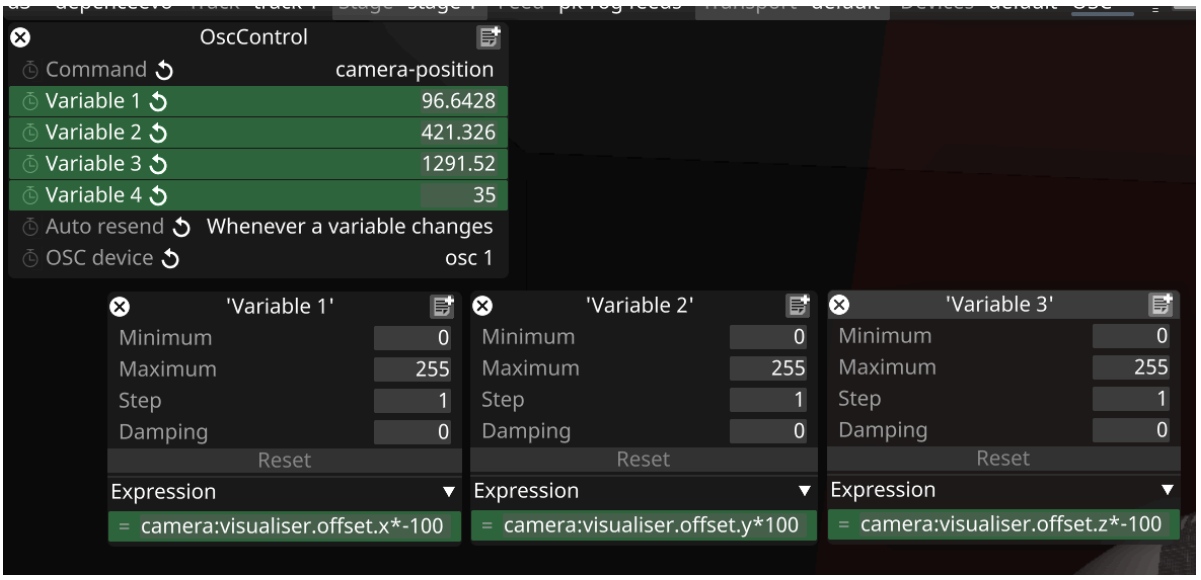


Depence Logic I/O Setup

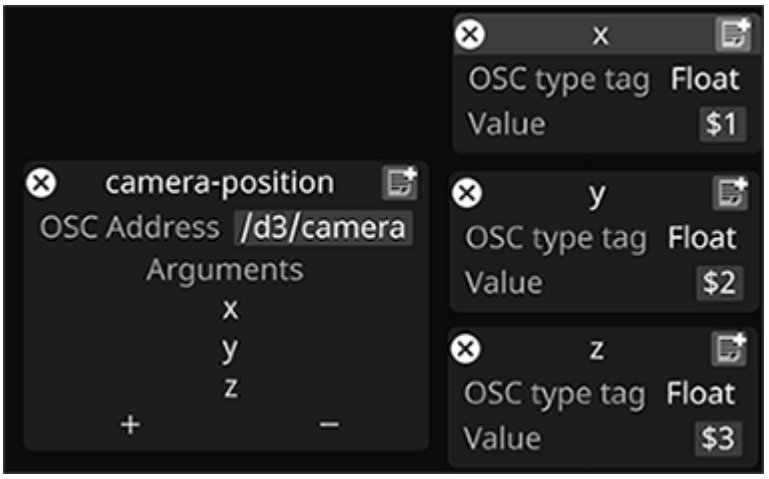
Designer Project Setup

1. In the Designer project, add an **OscControl** layer.

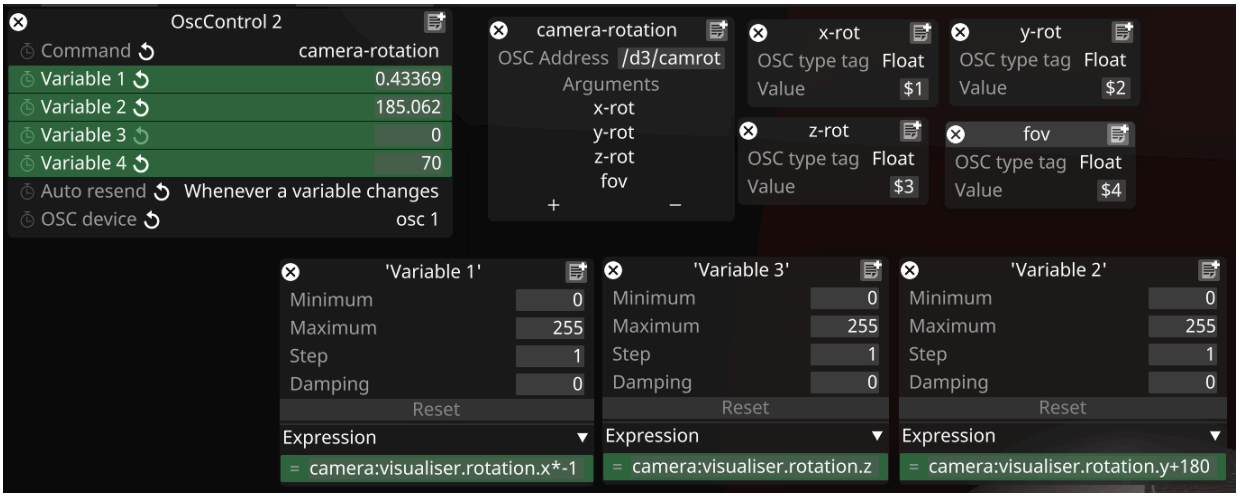
2. Set Variable 1, 2 and 3 to the X, Y and Z values of the camera you wish to control.



3. Add these expressions as arguments to the OscControl layer, using 2 and $3 for the X, Y and Z values.

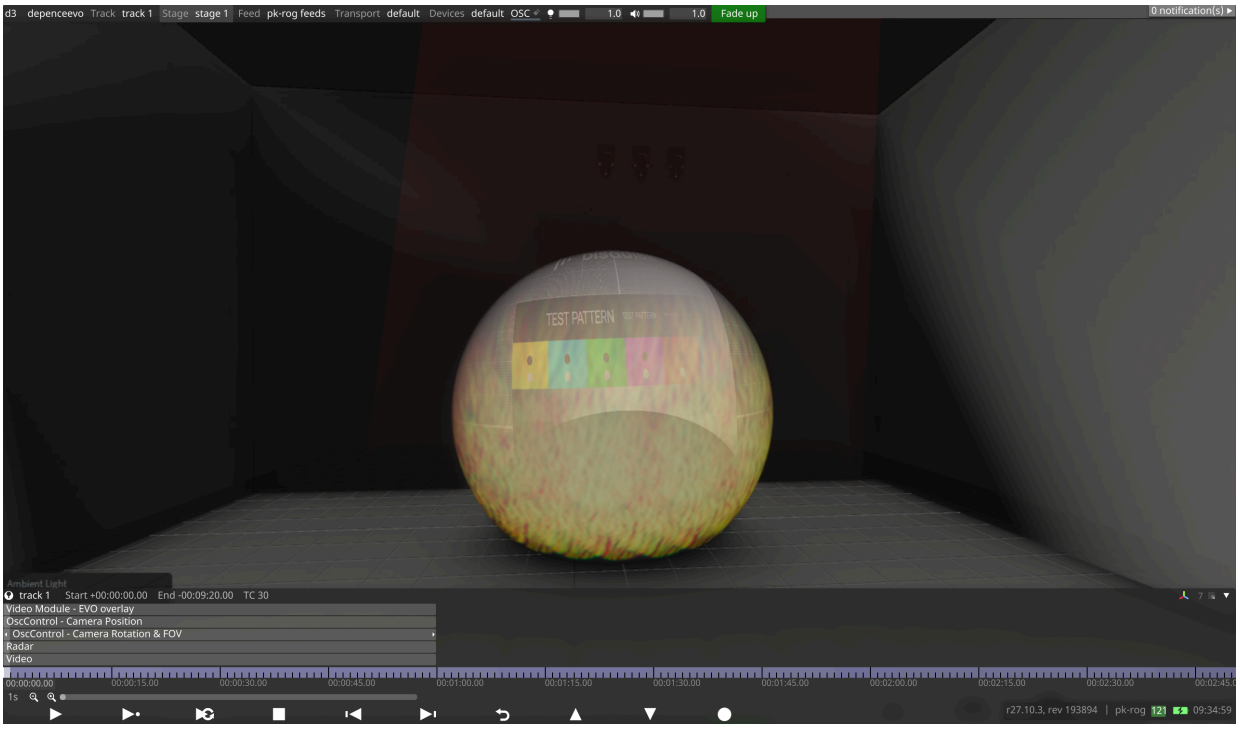


4. Repeat the above steps for the Rotation properties, using the appropriate arguments.



Control

After setting the above, moving the camera in d3 should move the camera in the Depence visualiser. If you combine this with the NDI® stream and compositing mentioned above, you will have a fully working integration.



> **Note**
>
> - Depence defaults to port 9000 for OSC communication, so you will need to change the port number in the OSC device in d3.
> - Note the above inversions and scale factors on the various parameters - this is to fit the d3 Camera coordinates to the Depence camera coordinates.

Further Information

- Depence website
- Depence documentation

OSC Workflows

Open Sound Control (OSC) is a versatile tool to control and monitor productions. In Designer, the OSC Device sends and receives OSC messages for flexible, network-based control.

It is important to note that we do not support OSC bundles, only messages.

The following pages provide information about controlling Designer:

- OSC Device need to be set up to access OSC messages on the network.
- OSC Transport Overview can be used to control and report on a Designer timeline transport.
- OSC Machine Control can be used to control and report on a Designer timeline transport.
- OSC Tester can be used to test sending and receiving OSC messages.
- OSC Sockpuppet can be used to control layers on the timeline.
- OSC Expressions can be used to build interactive capabilities within Designer-sequenced content.

OSC Sockpuppet

OSC Sockpuppet is a workflow that allows the user to control all the parameters of any layer via OSC.

Patching a layer to OSC sockpuppet

To patch a layer to OSC Sockpuppet, you can perform the following steps:

1. Right-click on the layer and select **Patch to OSC...**.
2. Select the OSC Device you would like to control your layer (you can change this later)

The layer will turn green, and each of its fields will be highlighted in green. This indicates the layer has successfully been patched to sockpuppet.

Selecting parameters to control

By default, all fields in the layer are now controlled by OSC.

> **Note**
>
> Immediately after patching, you can still control each field on the layer manually, however, if an OSC command is sent to set one of the values on one of the fields, that field will now no longer be manually controllable until the project is restarted or the field is disconnected from sockpuppet. This works the same as DMX sockpuppet, except with DMX Sockpuppet, you always have values for each parameter because there are always at least default values in the DMX buffer.

You can right-click on each individual parameter on the layer and select **Toggle Sockpuppet**, which will toggle sockpuppet control of that field, or **Isolate Sockpuppet**, which temporarily sets sockpuppet control for the field you selected and disables it for all the others. Once a field is isolated, toggling off the isolation will restore the fields to their pre-isolation state.

Controlling parameters

To control any field, you'll need to send an OSC message with the correct **address** and a valid **value**.

You can find the right address and example values to control any field by right-clicking on the layer on the timeline and selecting **Edit OSC Patch...** and then selecting **View Personality...**e OscPatch editor widget.

You can export this to a .txt file in the “output” folder next to the project so you can copy-paste them.

For resource fields, the “example” values will be the ones currently set in the layer. This means that you can you can set the field manually and then inspect the personality to find the exact value you should send.

We will give an overview of the correct formats here, however, do note that if there is a discrepancy between this page and the formats described in the personality, always use the personality formats as they will be most up-to-date.

Format of the address

The correct address format is:

```
/d3/layer/[name of layer]/[name of field]
```

The name of the layer and the field are both case sensitive. You may include spaces.

Format of the value

There are 4 kinds of parameters that you may control with OSC Sockpuppet:

1. Resources
2. Numbers
3. Options
4. Text

Resources

Resources refer to any “object” in Designer that has an editor. Examples are Mappings, VideoClips, Bitmap Textures, etc.

There are 3 ways to select a resource via OSC Sockpuppet

1. By resource path
2. By Bank + Slot Index
3. By Bank and Slot Indexes

By resource path

You can select a resource by sending its path as a string value. You must format the value you want to set as a “String” in your OSC sender. The syntax for that value is:

```
objects/[type of object]/[path]/[to]/[object]/[name of object].apx
```

The `[Type of object]` is an internal Designer type that is sometimes not user-visible. To see the correct type for your field, you should check the OSC personality for the field, which

will have the correct value.

The `[path]/[to]/[object]` is blank for objects you create within Designer, like Mappings, Textures, etc. But for objects that refer to files on disc, like VideoClips, they are the path to the object on disc starting at the first folder within the “objects” project folder.

For example, the file: `objects/VideoFile/sample/ada.jpg` Will be at: `objects/videoclip/sample/ada.jpg.apx`

The `[name of object]` is the name of the object within Designer. This is the name of an object, like a mapping, or the name of the file including its extension, like `.png` or `.jpg`.

All values for resources must be appended with `.apx`. This is an annoying implementation detail of Designer.

None of the values are case sensitive.

> **Note**
>
> Case sensitivity for the value is different from that for the address, which IS case sensitive.

You can also send an empty string to select `None` for the resource if the field supports being set to `None`

By bank + slot index

You can select a resource by sending a number as the value, which refers to a **Bank** and **Slot** that the resource is assigned to in the bank editor. You can read more about the bank editor here: bank editor. You must format the value as either an `Int` or a `Float` in your OSC Sender. The formula for working out this value is:

```
bank x 256 + slot
```

For example, if the resource is a mapping and is in bank 5 slot 26 and you would like to set it on a layer called **Video Layer 3**, you would do:

`/d3/layer/Video Layer 3/mapping`, Value: `1306`

Because `5 x 256 + 26` = `1306`

By bank and slot indexes

You can select a resource by sending two numbers at the same time, where the first number refers to the **Bank** and the second to the **Slot** that the resource is assigned to in the bank editor. You can read more about the bank editor here: bank editor. You must format the values to be `Int` or a `Float` in your OSC sender.

The correct syntax for this value is:

`bank slot`

> **Note**
>
> The above is the most common implementation, however, there are other implementations such as `[bank, slot]` or `bank, slot`. Please check the documentation of your OSC sender to see the correct syntax for sending multiple Values. (Some Senders will refer to "values" as "arguments", just to keep you on your toes!)

For example, if the resource is a video and is in bank 2 slot 12, and you would like to set it on a layer called **myVideoLayer**, you would do:

Address: `/d3/layer/myVideoLayer/video`, Value: `2 12`

Numbers

Numbers refer to any field in Designer that has a box where numbers can be entered. Examples are Brightness, Speed, Size, etc.

You can set the value of a Number by sending a number as the value, which refers to the value you would like to set the field to. You must format the values as either Ints or Floats in your OSC Sender.

The correct syntax for this value is:

`5` or `1.7`

For example, if you want to set the Brightness of a layer called *Video 5* to `0.3`, you would use:

Address: `/d3/layer/Video 5/brightness`, Value: `0.3`

Or if you would like to set the Frame Index to `243`, you would use:

Address: `/d3/layer/Video 5/frame_index`, Value: `243`

> **Note**
>
> For the above example, you could still send a Float value, for example `243.7`, but its value will be floored, so Designer will interpret the value as `243`. This behaviour will be the same for all fields that only accept integer values.

Options

Options refer to any field in Designer that presents a list when clicked. Examples are Blend Mode, At End Point, On Clock Lost, etc.

You can set the value of an option by sending a number as the value, which refers to the index of the position of the option in the list presented when the option is clicked. The list is 0-indexed. You must format the value as an Int or a Float in your OSC sender. An example of this indexing would be Blend Mode where `0 = Over`, `1 = Alpha`, `2 = Add`, `3 = Multiply`, etc.

The personality view you get when selecting "View Personality..." will print out a detailed list of all the options and their indexes for you, so you don't need to count their position in the editor. For more information, see the **Controlling Parameters** section.

The correct syntax for this value is:

`1`

For example, if you wanted to set the Mode of a video layer called *video2* to Reverse you would do:

Address: `/d3/layer/video2/mode`, Value: `5`

> **Note**
>
> You can also send a Float value, for example `5.9`, but its value will be floored, so Designer will interpret the value as `5`.

Text

Text refers to any field in Designer that has a box where text can be entered. Examples are Text on Text layer, Custom Parameters on a Web layer, a dynamic string field on a Notch layer, etc.

You can set the value of a text field by sending a string as the value, which is simply the string you want to be in the field. You must format the value as a String in your OSC sender.

The correct syntax for this value is:

`this is a string` or `"this is a string with quotes"` depending on the OSC Sender you are using.

For example, if you wanted to set the Text field of a text layer called *iHaveAQuestion* to "Why the heck is it called SockPuppet?!" you would do:

Address: `/d3/layer/iHaveAQuestion/text`, Value:
`Why the heck is it called SockPuppet?!`

Sockpuppet

Sockpuppet fully implements DMX, OSC and HTTP controls into Designer. This feature enables the user to fully control all layers and their parameters from a lighting desk or OSC/HTTP application. As a result, the operator is no longer restricted to timeline based sequencing, but can now program their project using the lighting desk's cue stack, or a combination of both; by having certain layers controlled from the lighting desk and others controlled from the Timeline. In short, Sockpuppet works by treating all of the layer types as fixture personalities.

Compatibility

Sockpuppet is fully compatible with any lighting desk capable of outputting Art-Net or sACN, as well as any OSC/HTTP platform capable of sending OSC/HTTP control strings.

Additionally, for added usability XML personalities can be exported from Designer and imported into any XML compatible console.

Synchronisation Across Machines

Prior to Designer r30.8, frame accurate sync across multiple machines cannot be guaranteed. From r30.8 onwards, the director now receives control data and distributes it synchronously to machines across the session, rather than have each machine process DMX data locally.

Network setup - console

First ensure the lighting desk and server are networked together and able to ping each other. You will need to assign the IP address and Subnet mask on both the lighting desk's Art-Net port and the server's Art-Net port. These will normally be addressed as shown below:

- server - DMX port - 2.0.0.x - 255.0.0.0
- lighting desk - Art-Net port - 2.0.0.x - 255.0.0.0

> **Note**
>
> **Please note**: Each device needs a unique IP address within the same IP range and subnet

Network setup - OSC

First ensure the OSC app and disguise server are networked together and able to ping each other. You will need to assign the IP address and Subnet mask (ensuring they are both set to the same IP and Subnet range) on the system hosting the OSC application and the server's

network port that is receiving the data. For OSC sockpuppet, ensure the following are configured correctly:

- OSC application send IP
- OSC application receive IP
- OSC application outgoing port
- OSC application incoming port
- Designer OSC device send IP
- Designer OSC device receive IP
- OSC Designer outgoing port
- OSC Designer incoming port

Network setup - HTTP

> **Note**
>
> **Please note**: Designer can only receive HTTP sockpuppet commands, and cannot send live sockpuppet values.

HTTP sockpuppet network configuration is setup in Manager > Machine Settings > Advanced Network Configuration > API http port.

Confirm the connection

To check the two devices can communicate with each other, a command line called ping should be used.

Setup

Once network connectivity has been confirmed between the server and the lighting desk, the next step is to create a DMX or OSC device in Designer. For HTTP sockpuppet please see this page for details on how to set it up.

Further Reading

- DMX setup
- OSC devices
- HTTP sockpuppet

Patch Assignments

Sockpuppet treats all of Disguise's layer types as fixture personalities. The Patch Assignments window is where we patch or assign our layers on our Timeline to incoming DMX values or channels. For OSC and HTTP the process is slightly different.

DMX

> ⚠ **Warning**
>
> In r31.1, we broke existing DMX personalities. This has been fixed in r31.2. If you have experienced this issue, please update to r31.2.

Patching a layer to DMX control

1. Access the patch assignments window by right-click on the d3 icon in the top left corner of the interface.
2. Navigate down to the Sockpuppet tab.
3. Beneath this you will find two fields, the first will be **Patch Assignments** and the second **Bank Editor**.
4. Left-click Patch Assignments to open the patch assignments window.

View Types

Click the right triangle next to Patch Assignments, and click on the view type field to open the list of available view options.

Patch grid

Displays all 512 channels in a grid format. You are able to patch to any free channels here or view which channels you have already populated with a patch. Left-click on Patch Grid to view different options to change the layout of the patch window.

Please note: this is the view type you will need to be in to patch your layers.

Patch list

This shows us a list of all of the current layers patched and their start channels within any selected universe (selectable from the Viewing Universe field).

Global patch list

This follows the same format as the Patch List, however this will show us a full list of all layers patched regardless of which universe they start on.

Universe occupancy

This will show us in grid format which of our universes we currently have layers patched to.

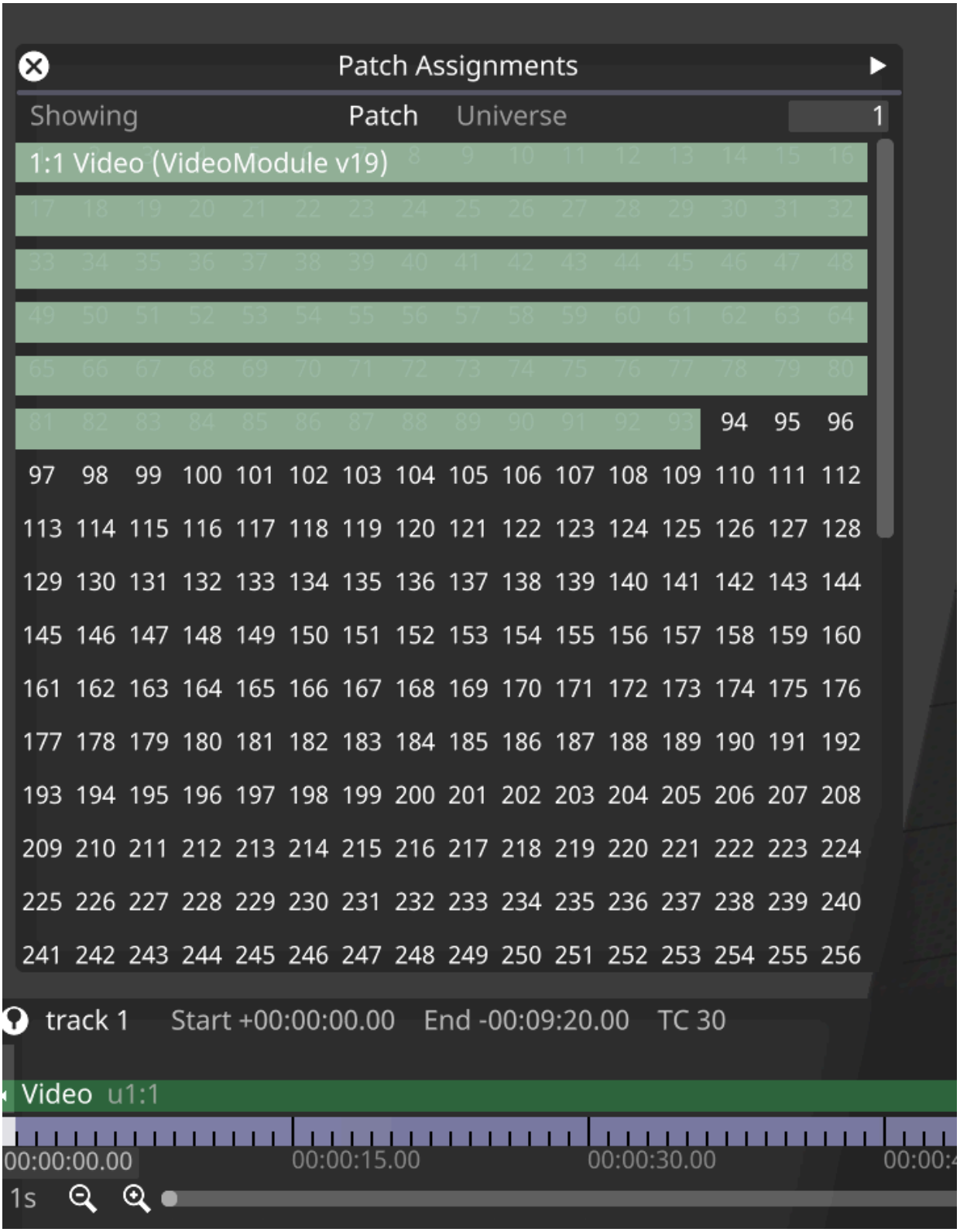
Patching layers

To patch layers within Designer ensure:

- The Patch Assignments window is open & in the patch grid view type.
- The layer you want to patch is added to the Timeline. Click here for more information on working with layers.
- Hover the cursor over the DMX channel you want to patch to in the Patch Assignments window.
- Select the DMX channel while holding down **ALT** to generate a white arrow, which should be dragged down to the layer on the Timeline that you want to patch to.

This will create a green block over the channels in our patch grid that the layer will take up, and also turn the layer on the timeline green to signify the layer is patched and only controllable through Sockpuppet DMX.

We will also be presented with some key information regarding our patch. This will include the start address, a user specified name (this will be “untitled” by default) the layer type, and the personality version.

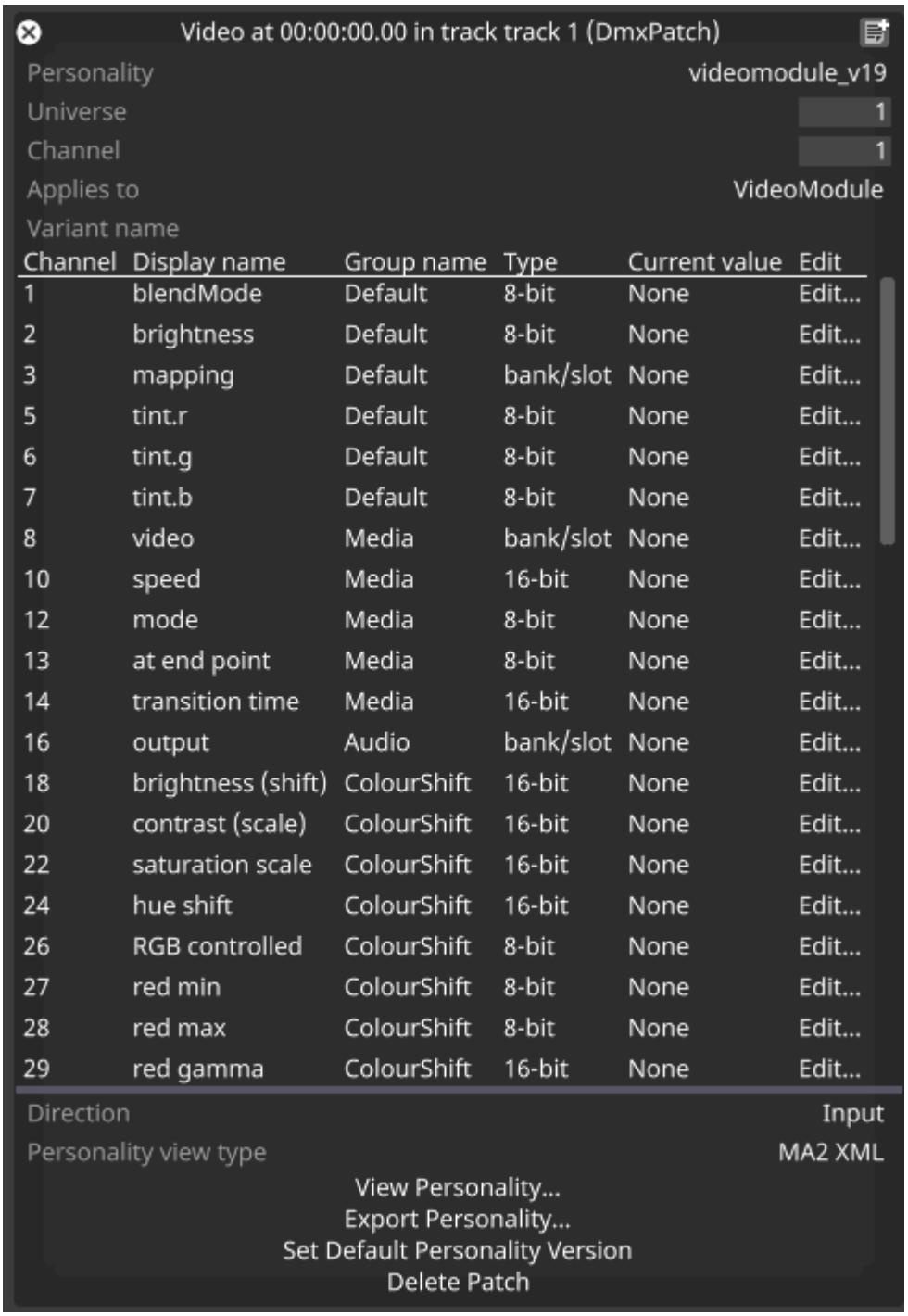


Note

Please note: Any previous keyframing / sequencing done on the layer will be overwritten and deleted by the default values sent from the desk once the layer is patched.

Patch settings

To access this patch settings menu, right click on the green patch block, this will open the patch settings.



Personality

Here we can select the version of the sockpuppet personality we would like to use. A personality profile may be updated / changed with a new release of Designer, this function will allow you to roll back to the personality version that you originally programmed your show with,

meaning you will not have to update any of the programming on your console. The current selected personality is displayed as a table below the Variant name field.

Universe

Here we can specify the universe we want to patch our layer to.

Channel

Here we can specify the start channel we want to patch our layer to.

Applies to

This displays the layer type the sockpuppet personality applies to.

Variant name

This is not relevant for sockpuppet patches, and is only used for Event Transport patches.

Direction

Here we can set whether we want input values to control the layer's values, or output the current values of the layer.

Personality view type

Here we can select whether to view and export the personality as a GDTF file, a text-formatted chart, or a (deprecated by MA) MA2 XML file.

View Personality

Opens a window displaying the personality with the chosen Personality view type.

Export Personality

Exports the personality as a file with the chosen Personality view type.

Set Default Personality Version

Sets the default version of sockpuppet personality for the current layer type.

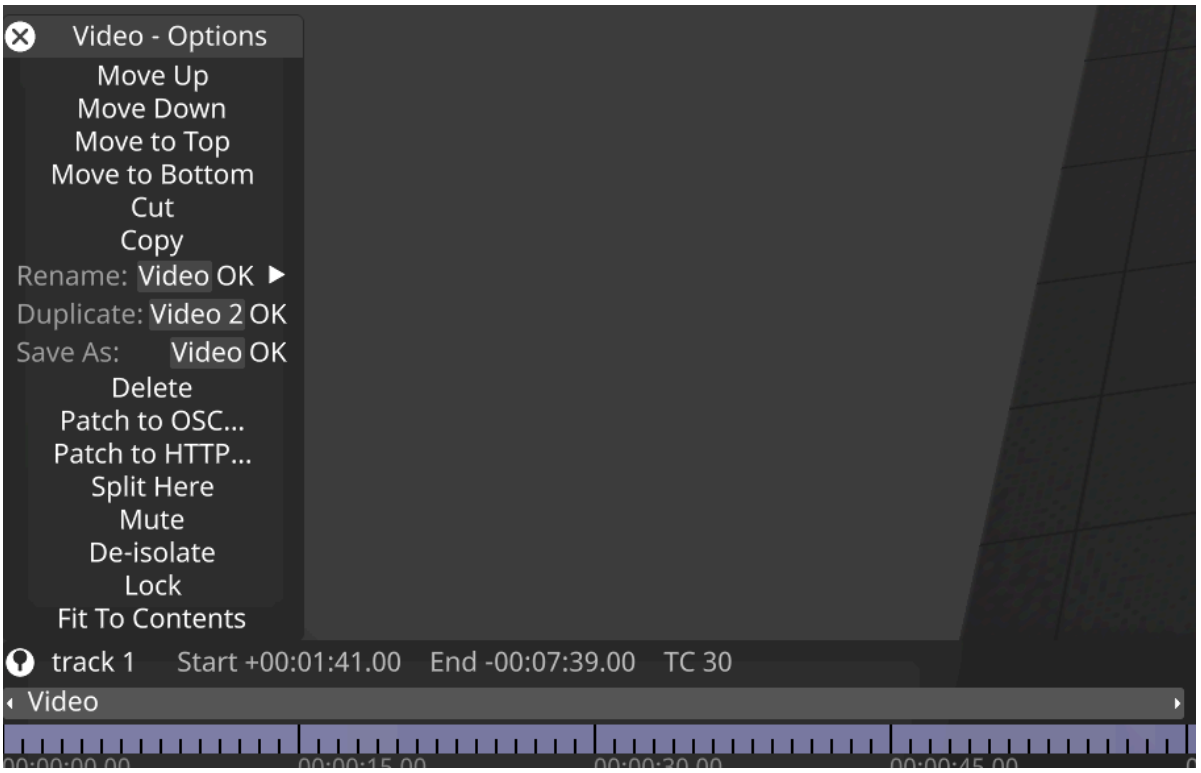
Delete Patch

By left clicking on delete patch we will remove the selected layer from our patch assignment. This will return it to its standard operating mode (controlled through the timeline & keyframable)

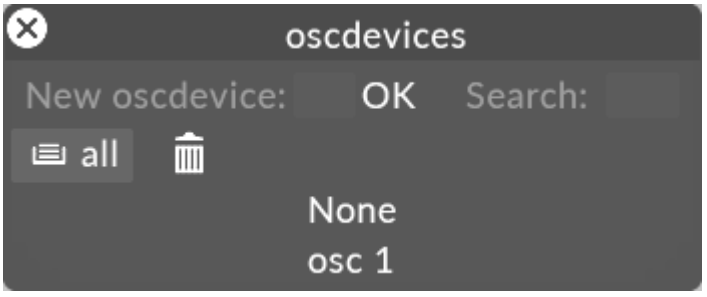
OSC

Patching a layer to OSC control

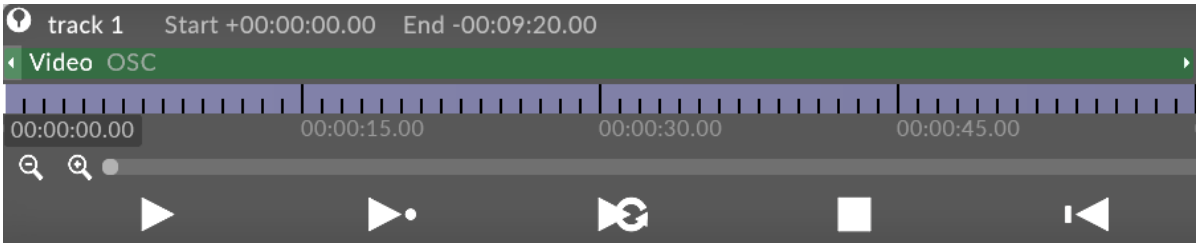
1. Right-click a layer on the timeline.
2. Left-click Patch to OSC.



3. Choose OSC device.

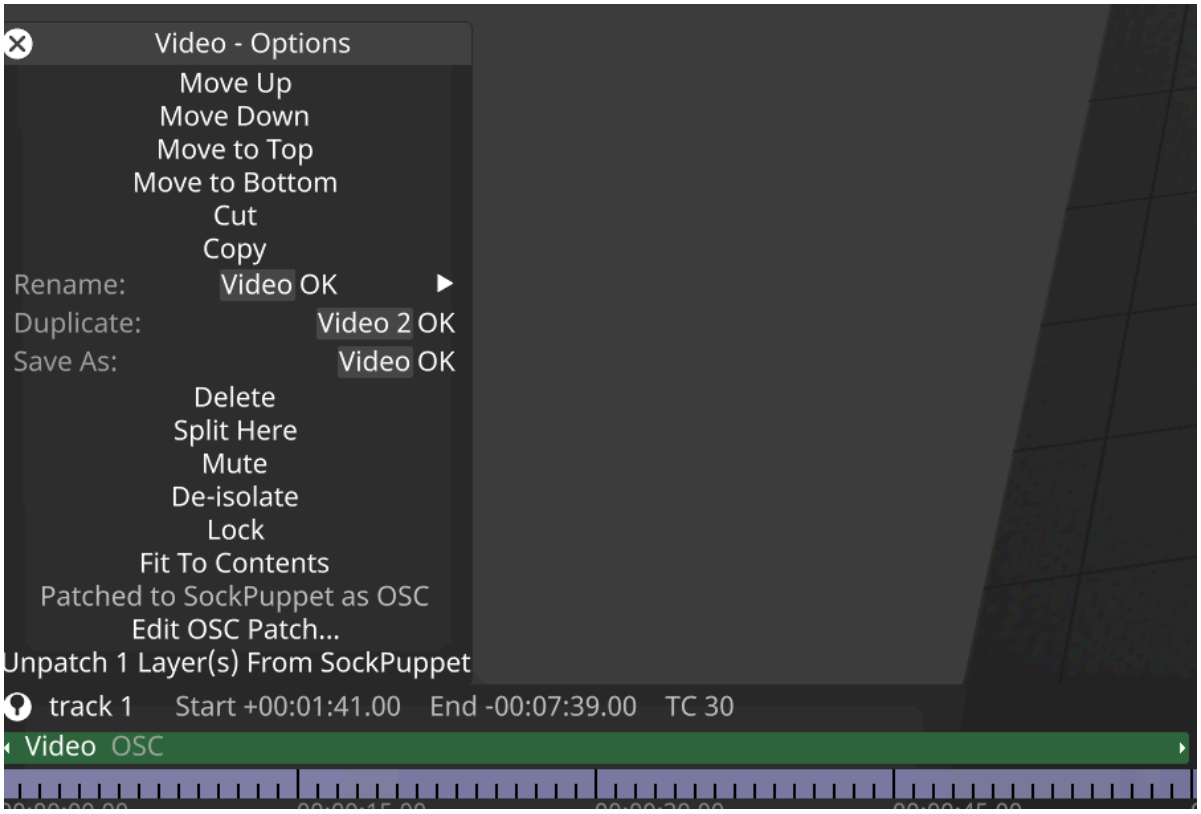


The layer will now turn green, indicating that it is controlled by an external sockpuppet system.

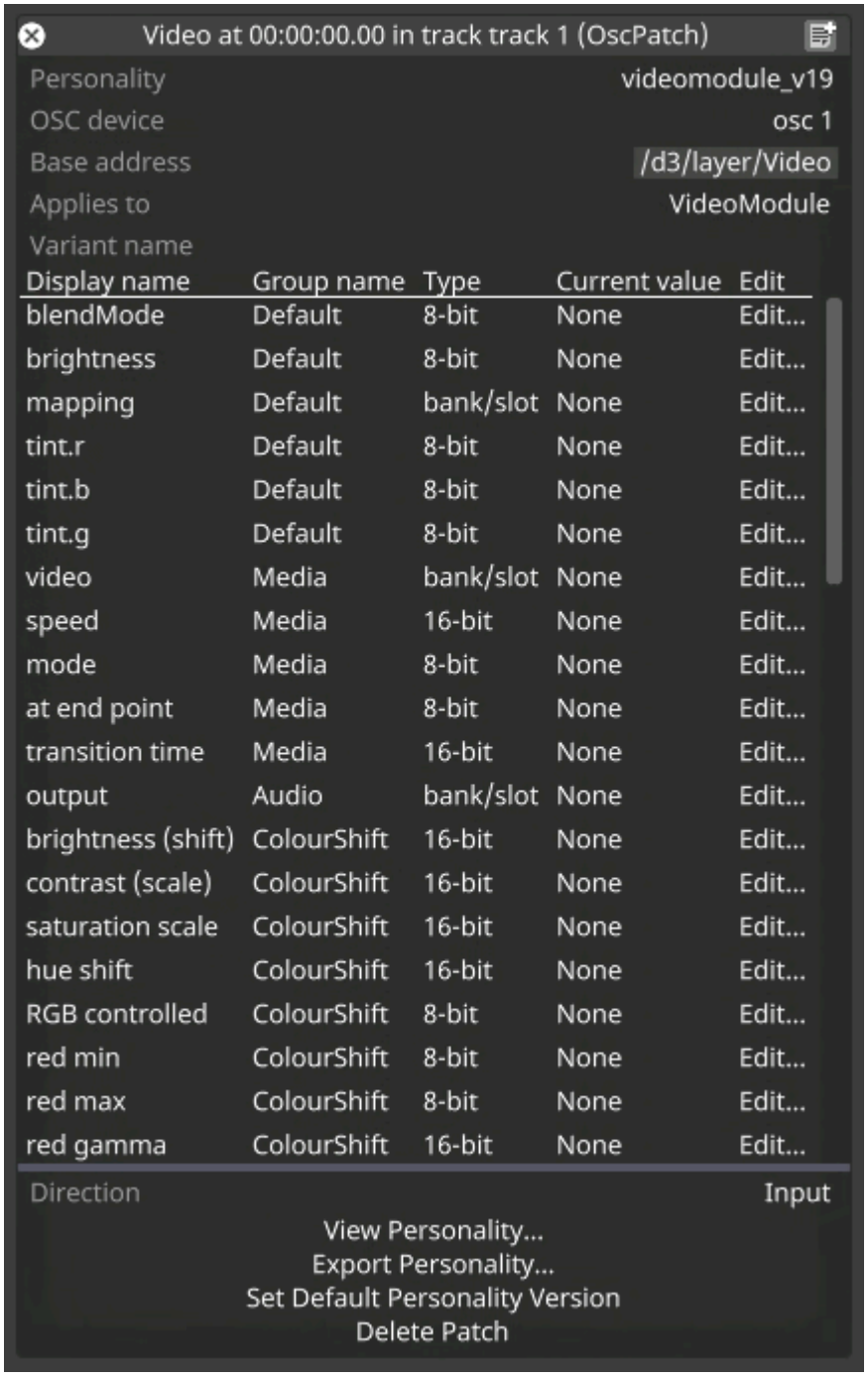


View the personality

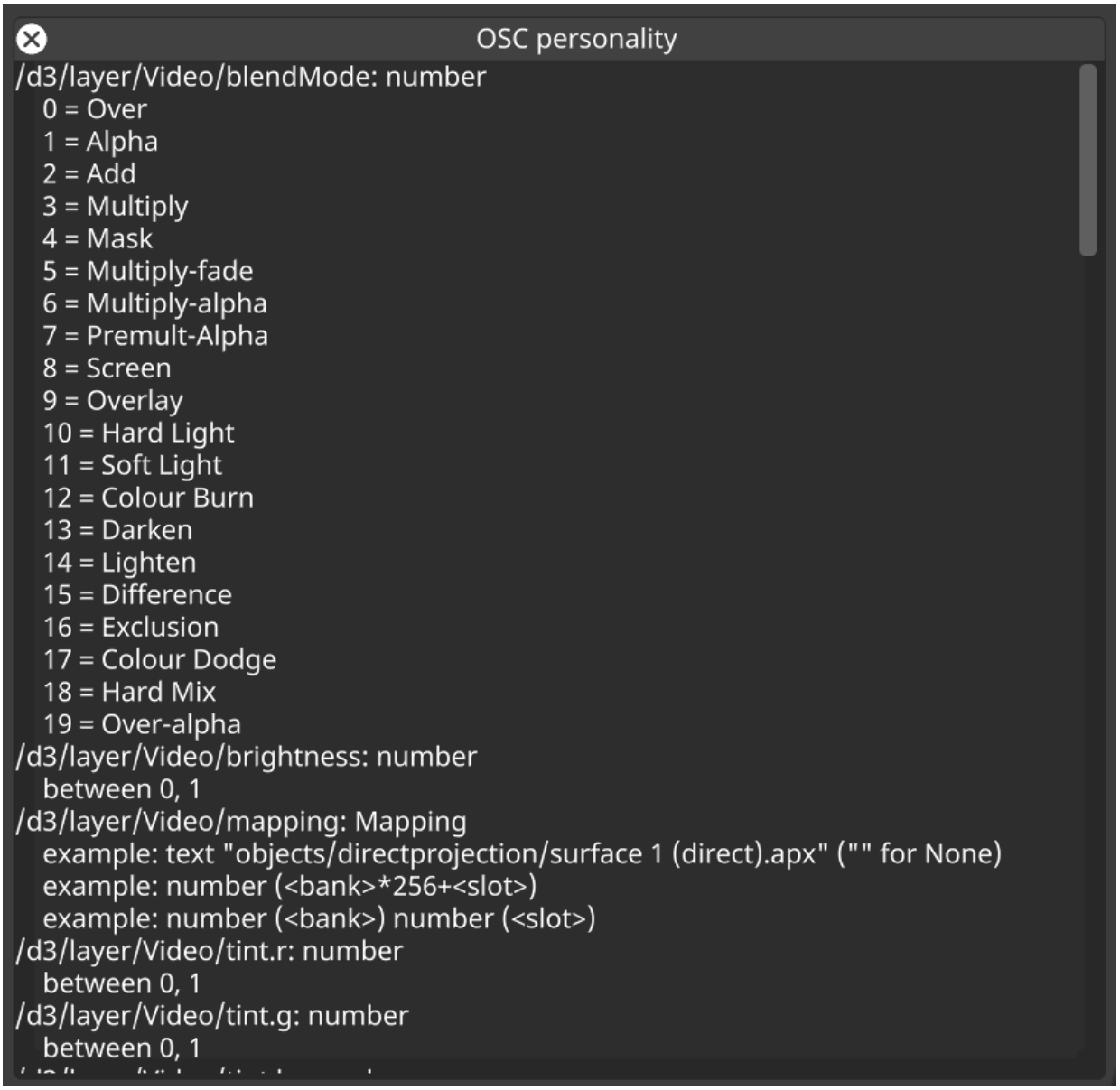
1. To view the personality of the layer for OSC, right click the layer and choose Edit OSC patch.



2. Left-click **View personality**



The default OSC personality opens.



Export the personality

1. Repeat the previous steps to view the OSC patch.
2. Once the patch is open, left click the Export Personality button at the bottom of the window to export the OSC personality as a txt file.

Bank Editor

> **Note**
>
> For best results have the console set up to work in **decimal** rather than percent.

What is the Bank Editor?

The Bank Editor enables us to assign DMX values to the various resources required within a layer. Examples of resource parameters include video or bitmap files, mapping types, or HTML bookmarks. This process is managed on a bank-and-slot basis.

Banks & slots

Within the layer personality, Sockpuppet assigns two channels to each resource parameter. The first channel is for the resource bank, and the second is for the resource slot. For example, within the video layer personality, channel 8 is for the video bank and channel 9 is for the video slot.

Example: If we have a video asset assigned to video slot 5 in video bank 3, we would need to output the following from the lighting console to recall it:

```
channel 8 @ 3, channel 9 @ 5.
```

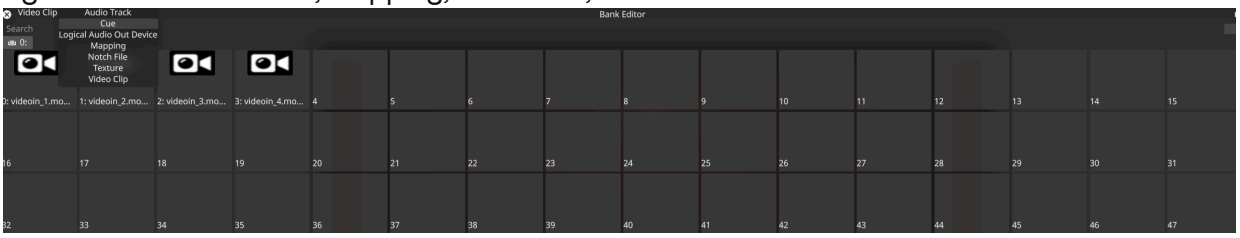
This means that we can have a maximum of 255 video banks and within each of those we can have 255 video slots, giving us a total of 65025 assignable video slots (255*255) for each resource type.

Main Interface elements

To open the Bank editor right-click on the state menu (top left) and underneath the sockpuppet tab left-click **Bank**.



At the top of the window, the currently selected resource type is displayed (in this case, Video Clip). Left-click Video Clip to open a menu of other resource types, such as Audio Track, Cue, Logical Audio Out Device, Mapping, Notch File, and Texture.



You can populate the resource bank with resources by dragging them from the resources view in the bottom of the window to empty bank slots.

Preferences

The Bank Editor preferences can be opened by left-clicking the small white triangle in the top-right corner of the window.

View type

Select how you would like to view the Bank Editor. **Grid** displays the slot window in a grid format, and **List** displays the slot window as a list.

Columns

Specifies the number of columns displayed by default.

Parameter tracking

By left-clicking on this field and setting it to **On**, we will have a visual representation of which slot we currently have selected; so as the console programmer scrolls through the slots and banks on the encoders, a red square will be drawn over the slot we currently have selected.

Auto-populate

When disabled, it stops assets being added automatically based on their file name.

Search function

Allows you to search through assets already assigned to slots. Matching results are highlighted with a light blue square.

Working with banks

Banks are configured and managed using the tabs bar above the slot layout. By default, bank 0 is already created.

To open the bank options window, right-click on an existing bank (including bank 0).

Rename

Specify a custom name for the bank, then click **OK** or press **Return**.

Duplicate

Copy the contents of the selected bank into another bank. Specify the destination bank number, then click **OK** or press **Return**.

New bank

Create a new bank by entering the desired bank number, then clicking **OK** or pressing **Return**.

Auto-populate banks from directory

Please see Adding Assets to Slots below.

Reset bank

Returns the bank to its default state and clears any populated slots.

Remove missing

Removes missing media from the selected bank.

Delete bank

Left-click to permanently delete the bank from the Bank Editor.

Adding assets to slots

Manually add assets to slots

To manually add an asset to a slot:

1. Select the resource type you wish to assign (for example, Textures).
2. Left-click the thumbnail in the library.
3. Drag and drop it onto the slot you wish to assign. Repeat this process for all required content.

If you have your content organised into files and boxes (read about organising your content here) then you can quickly populate your slots with whole boxes at a time. To do this simply hold down Alt on the keyboard and left-click over the box you wish to use, this will turn our cursor into a white arrow which we can drag from our box to the first slot we wish to use, release and it will populate the slots with the contents of that box.

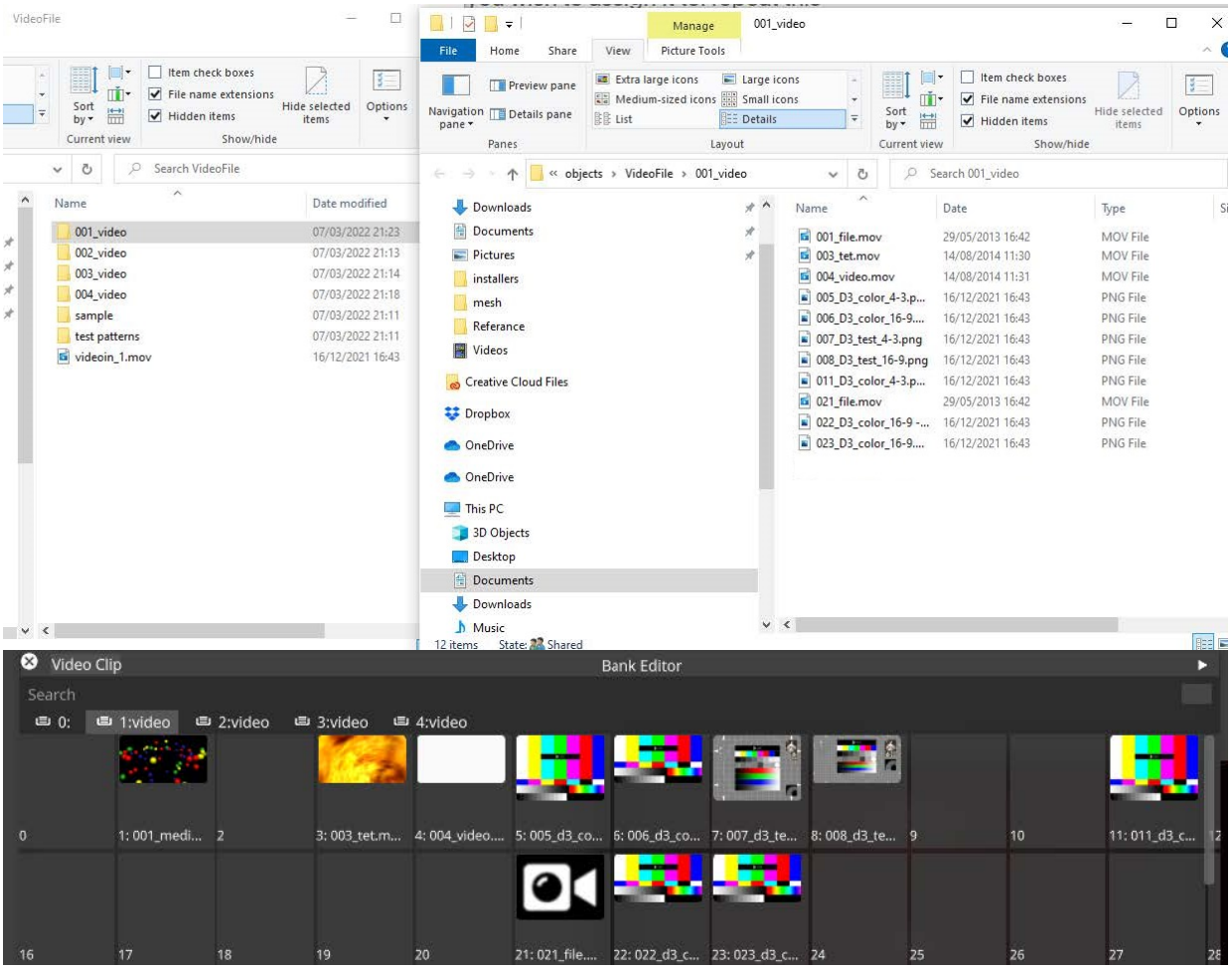
Using auto-populate to add assets to slots

Another widely used method of assigning DMX values to assets, is adding numerical values to folders and files within your assets library. Designer also supports this workflow and is managed via the auto-populate field in the bank options (see above for how to access).

Auto-populate file structure:

Each of the content folders located within the objects/videofile folder needs to be labelled numerically, for example: `001_Foldername`

Files within this folder should follow this same numeric labelling scheme: `001_filename`



Auto-populated results within the Bank Editor

Creating a directory

When using this method to populate your banks and slots, you will need to start by numbering your folders and files within the project directory in Windows.

Navigate to the resource folder you wish to use (we will be using DxTexture as an example) and number your folders within the DxTexture folder, these will become your bank numbers. Within each of these folders you will need to number your asset files, these will become your slot numbers.

Clearing assets from slots

To clear a slot, overwrite it with the **None** thumbnail from the resource library. Left-click the **None** thumbnail and drag it onto the populated slot.

Alternatively, use the **Reset Bank** option to clear the entire bank.n to clear out the entire bank.

Banks with OSC

As of r17.3, banks can also be used with OSC sockpuppet.

- The path is always the same: `/d3/layer/Typeoflayer/Nameoflayer`.
  For example: `/d3/layer/Video/video`
- It's always one message (One address/path + one or two integers)
- **Option one**: One path and one integer with bank and slot encoded into one number.
  For example: `bank 1 slot 1` is `256+1=257`.
- The formula for a single integer is `bank*255+slot`
  For example: `bank 0 slot 1` is `0*255+1`.
- **Option two**: One path and two integers with bank in the first integer and slot in the second integer.

Sockpuppet Personality Editor

The **Sockpuppet** personality editor is a feature of Designer that allows users to edit the personality of a patched layer, and export it in either GDTF, .txt chart or MA2 XML format.

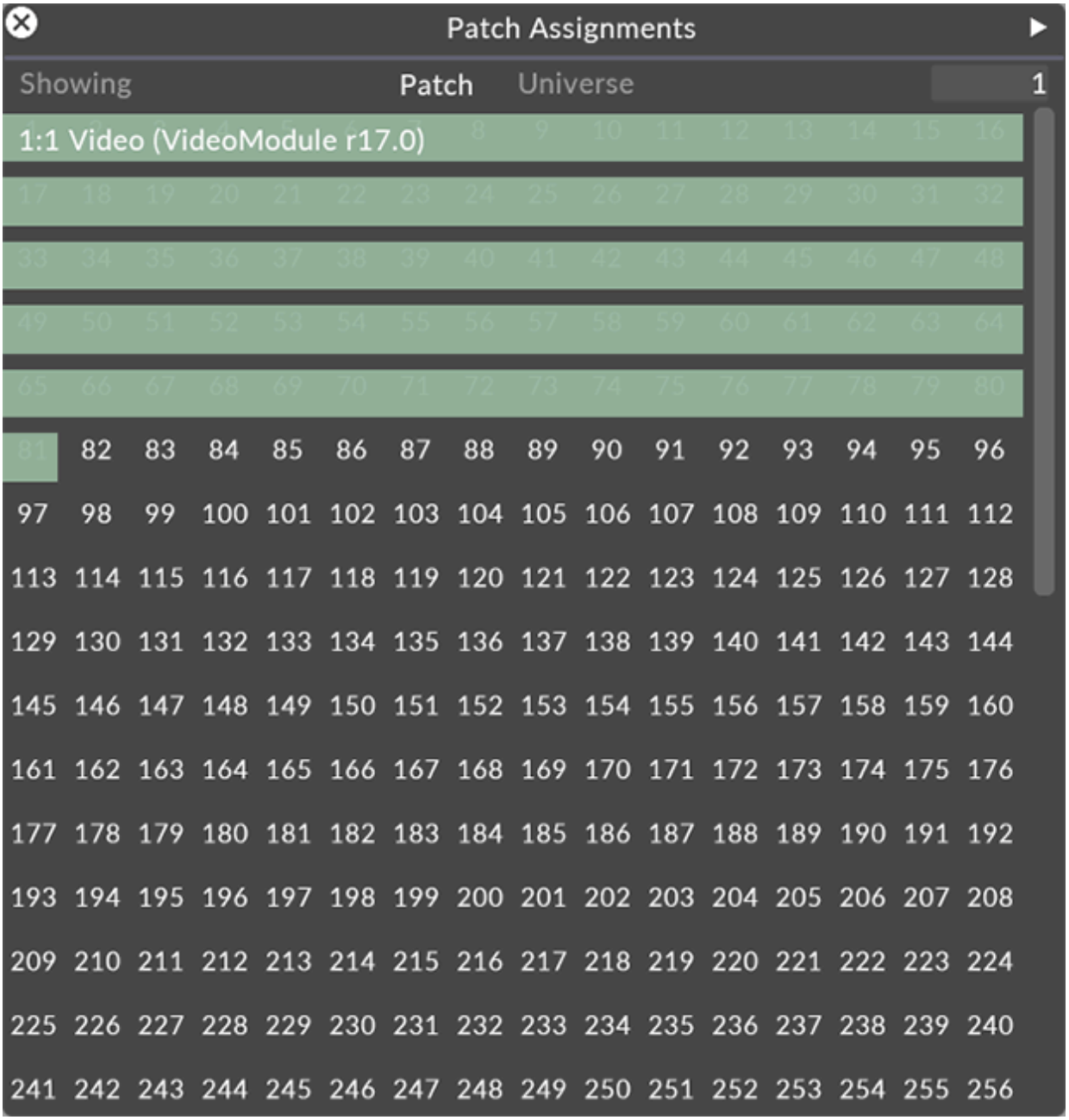
It's important to note that currently, the functionality is limited to patched layers, and you cannot store a custom personality for use with other layers of the same type.

The functionality is useful for setting 16-bit control of some control channels, as well as removing parameters from the personality.

Accessing & using the personality editor

DMX

1. Patch a layer as specified in the patch assignments topic.
2. Right-click the patched layer in the patch assignments widget.

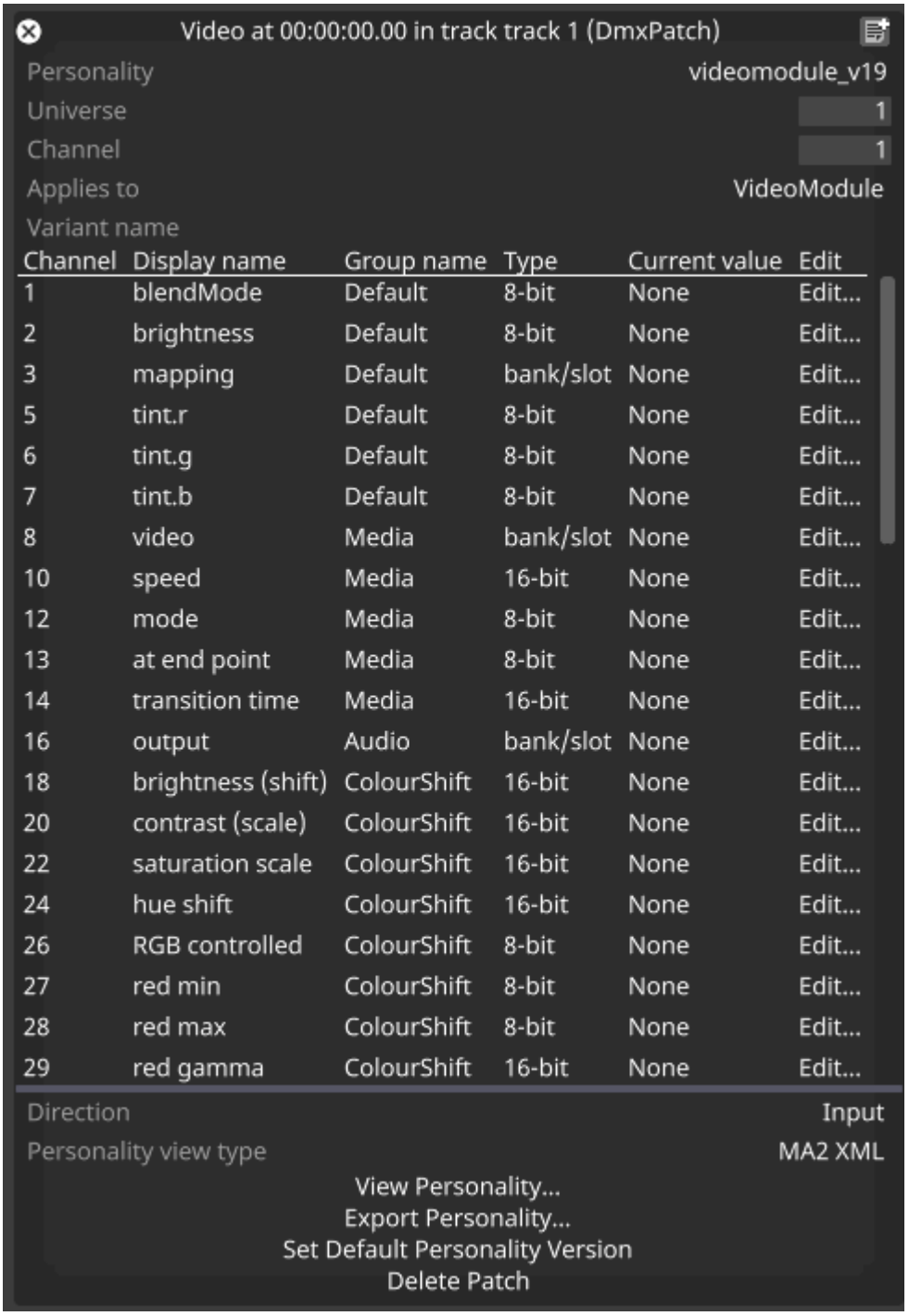


OSC and HTTP

1. Right-click the patched layer on the timeline.
2. Click Edit OSC/HTTP Patch.

Patch editor

You can now edit values in the Patch editor, which will be stored to the personality in use for that layer.



Use the **View/Export Personality** buttons to show or export the personality in either GDTF, text or MA2 XML format.

The editor automatically sorts personality properties in ascending order based on their assigned channel values.
Changing a channel value automatically moves the selected row according to its new channel value.

Editing and creating a custom DMX personality

Channels can be removed from a personality as well to create a custom DMX profile. Click on **Edit...** on the right-most column of the personality table to open the sockpuppet property editor.



Properties can be disabled and removed from the personality by unticking the **Enabled** checkbox. The Channel Offset, Display Name, Group Name, and Min/Max/Default Values can also be customised in this editor and exported.

If a property has options assigned, the corresponding table is displayed as well, allowing you to edit the values of the accepted options. Editing these values affects only the incoming ranges, not the actual value applied to the controlled layer.

Editing multiple properties

The multi-editing feature allows assigning similar values to multiple properties at once. Use the following shortcuts to control selection:

- **Ctrl + Left Mouse Button** — toggles the selection of a property.
- **Ctrl + A** — toggles the selection of all properties.
- **Shift + Left Mouse Button** — selects a range of properties, from the currently clicked property to the previously selected one.

Selection (**LMB**, **Ctrl + LMB**, **Shift + LMB**) only occurs when clicking on a read-only field or the row background.

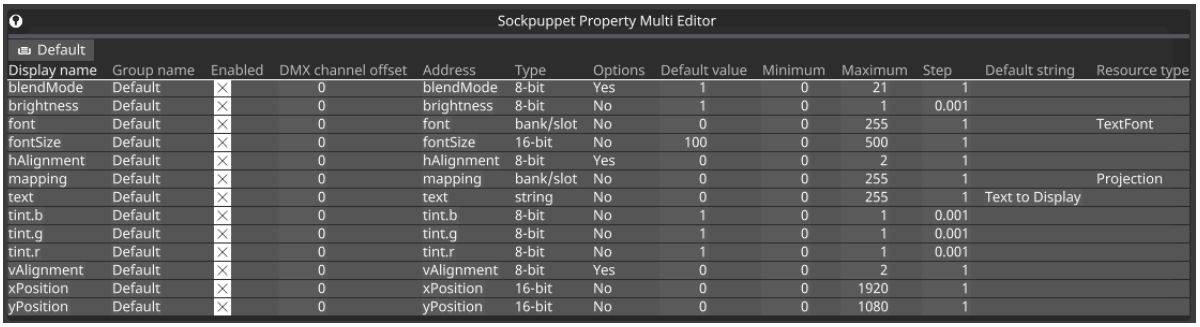
Clicking on an editable field preserves the current selection and applies the entered value to all selected properties after pressing **Enter**.

Special cases

- **Channel column:** When editing multiple properties, it assigns channel values to ensure that the edited DMX offsets align sequentially without gaps, starting from the chosen base address while preserving order.
  For example, if the selected properties have channels **7**, **15**, **23**, and **42**, typing in a channel value of **5** and pressing **Enter** assigns these properties the values **5**, **6**, **7**, and **8** respectively, preserving their order.
- **Type column:** This column is mainly read-only, except for numeric types.
  Fields with types **8-bit** or **16-bit** are editable and allow selecting a type from a dropdown menu.
  When multiple properties are selected, and a numeric type is changed, the change applies only to fields of the same type, and all others remain unchanged.

Sockpuppet property multi editor

- **Edit...** column opens a special **MultiEditor** widget when multiple properties are selected.


Sockpuppet Property Multi Editor

Default

Display name	Group name	Enabled	DMX channel offset	Address	Type	Options	Default value	Minimum	Maximum	Step	Default string	Resource type
blendMode	Default	☒	0	blendMode	8-bit	Yes	1	0	21	1		
brightness	Default	☒	0	brightness	8-bit	No	1	0	1	0.001		
font	Default	☒	0	font	bank/slot	No	0	0	255	1		TextFont
fontSize	Default	☒	0	fontSize	16-bit	No	100	0	500	1		
hAlignment	Default	☒	0	hAlignment	8-bit	Yes	0	0	2	1		
mapping	Default	☒	0	mapping	bank/slot	No	0	0	255	1		Projection
text	Default	☒	0	text	string	No	0	0	255	1	Text to Display	
tint.b	Default	☒	0	tint.b	8-bit	No	1	0	1	0.001		
tint.g	Default	☒	0	tint.g	8-bit	No	1	0	1	0.001		
tint.r	Default	☒	0	tint.r	8-bit	No	1	0	1	0.001		
vAlignment	Default	☒	0	vAlignment	8-bit	Yes	0	0	2	1		
xPosition	Default	☒	0	xPosition	16-bit	No	0	0	1920	1		
yPosition	Default	☒	0	yPosition	16-bit	No	0	0	1080	1		

The editor displays all selected Sockpuppet properties with all available parameters for editing. When first opened, all rows in the table are preselected, allowing you to edit multiple properties by typing a new value and pressing **Enter**.

The table supports sorting — simply click any header to sort properties by the corresponding field value.
By default, the table sorts properties by their display name.

Use the following shortcuts to control selection:

- **Ctrl + Left Mouse Button** — toggles the selection of a property.
- **Ctrl + A** — toggles the selection of all properties.
- **Shift + Left Mouse Button** — selects a range of properties, from the currently clicked property to the previously selected one.

Columns

- **Type column:** This column is mainly read-only, except for numeric types.
  Fields with types **8-bit** or **16-bit** are editable and allow selecting a type from a dropdown menu.
  When multiple properties are selected, and a numeric type is changed, the change applies only to fields of the same type; all others remain unchanged.
- **Options:** Read-only field indicating whether the Sockpuppet property has attached options for supported values.
  See editing a single DMX personality for details.
- **Default string:** Editable only for properties that store text. For all others, this is a read-only empty field.
- **Resource type:** Read-only field for properties that provide resource selection, such as mappings or media.

Personality Export

The Personality Editor allows you to view or export the Personality in one of three formats: GDTF, text chart or MA2 XML. The exported Personality is created from the current state of the Personality, so you can (for example) disable channels and see this change reflected in the export.

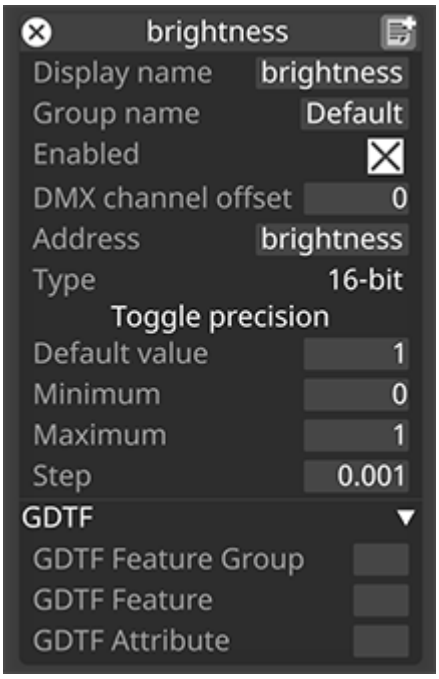
This editor also allows you to provide manual values for the Property's matching GDTF Feature Group, Feature and Attribute. If not set, Disguise will automatically match the Property to a suitable Feature Group, Feature and Attribute.

GDTF

GDTF (General Device Type Format) is a format that is designed to be interoperable between many different products and manufacturers. It does this by trying to describe every fixture using a limited number of Normative Attributes, such that two different manufacturers, for example, can treat a fixture's RGB channels the same, and additionally support things like colour palettes.

Automatic Attribute Matching

For every Property in a Personality, we use the address and Group to work out a corresponding GDTF Feature Group, Group and Attribute. Where possible, we map onto a normative Attribute (which also specifies the Feature Group and Feature), but where that's not possible, we ensure that the same address and group always results in the same Attribute definition, to keep consistency between different modules.



Both the Group and Address can be edited from the Property editor - Address is by default the same as the display name of the field.

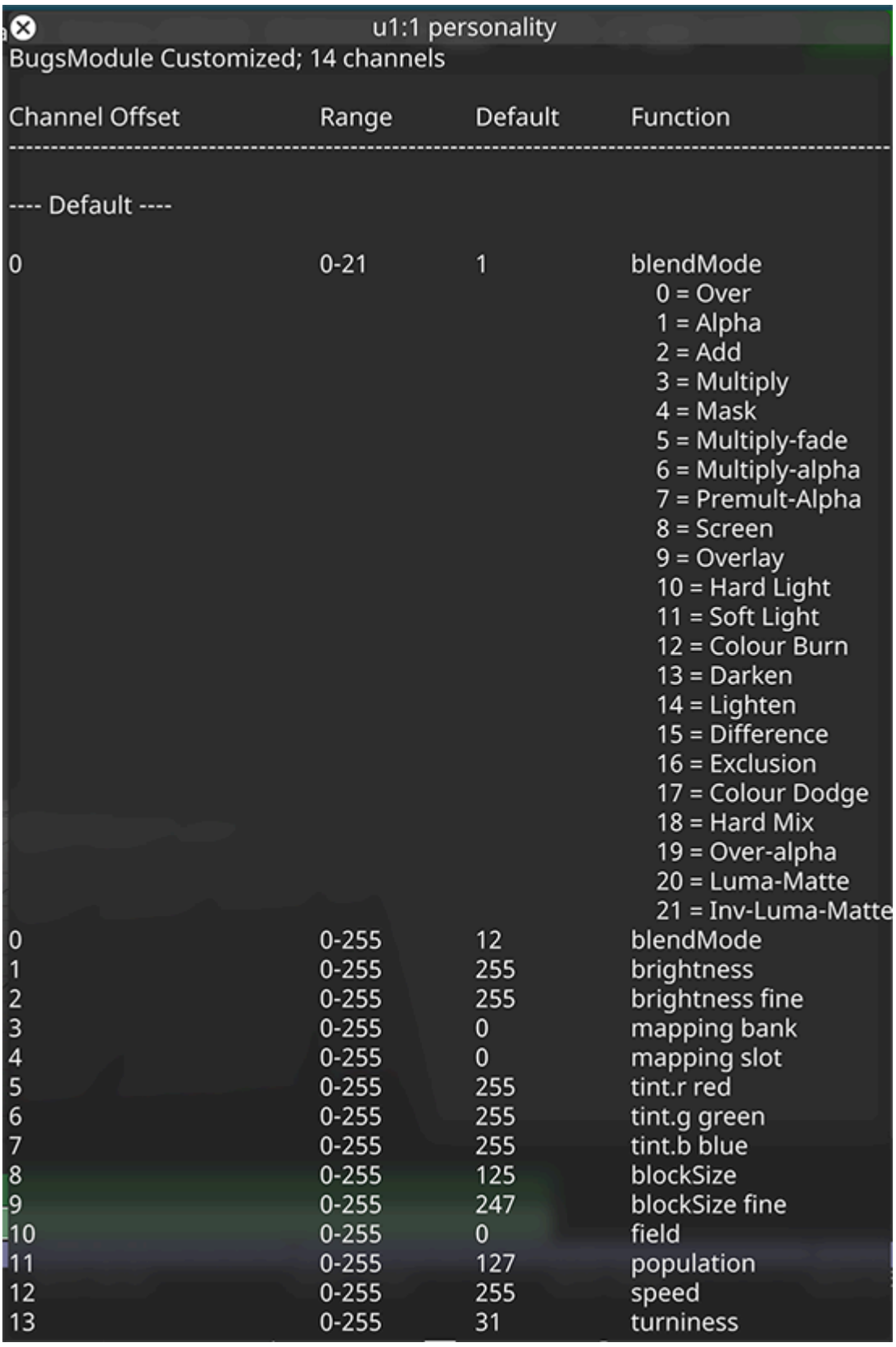
Disguise Attribute matching will always result in a normative Feature Group, but we don't promise normative Features or Attributes when there is simply no GDTF Normative Attribute that matches, for example, the Bugs module's 'turniness' property!

Manual Attribute Specification

If the automatically matched Attribute doesn't match your needs, it can be manually overridden by opening the Sockpuppet Property editor and typing in the desired Feature Group, Feature and Attribute. Any of these that are not set will continue to be matched automatically.

This feature is particularly useful for grouping things like exposed parameters in Notch, which Disguise has very limited information about.

DMX Chart



u1:1 personality
BugsModule Customized; 14 channels

Channel Offset	Range	Default	Function
---- Default ----			
0	0-21	1	blendMode 0 = Over 1 = Alpha 2 = Add 3 = Multiply 4 = Mask 5 = Multiply-fade 6 = Multiply-alpha 7 = Premult-Alpha 8 = Screen 9 = Overlay 10 = Hard Light 11 = Soft Light 12 = Colour Burn 13 = Darken 14 = Lighten 15 = Difference 16 = Exclusion 17 = Colour Dodge 18 = Hard Mix 19 = Over-alpha 20 = Luma-Matte 21 = Inv-Luma-Matte
0	0-255	12	blendMode
1	0-255	255	brightness
2	0-255	255	brightness fine
3	0-255	0	mapping bank
4	0-255	0	mapping slot
5	0-255	255	tint.r red
6	0-255	255	tint.g green
7	0-255	255	tint.b blue
8	0-255	125	blockSize
9	0-255	247	blockSize fine
10	0-255	0	field
11	0-255	127	population
12	0-255	255	speed
13	0-255	31	turniness

The DMX chart format is designed to allow you to easily reference DMX channel values for manually setting up fixtures when working with equipment that doesn't support GDTF files.

MA2 XML

We also support export into MA's proprietary MA2 XML format. Please note that MA have deprecated this format, and MA3 supports GDTF. We recommend that you only use this with legacy MA2 systems.

Sequencing with Sockpuppet

There are a number of changes to the way in which layers are programmed or sequenced when running in Sockpuppet mode. The following section will outline these differences.

Layers with Extended Sockpuppet Functionality

Below are some layers which gain additional or changed functionality when controlled by Sockpuppet:

- Video Layer
- Notch Layer
- Gradient Layer
- Chevron Layer

Dormant layers

Dormant layers is a feature to enable the stacking of large numbers of layers on the timeline. The issue with this previously was that each layer on the timeline required a certain amount of system resources in order to run, which would quickly add up when working with large numbers and result in a significant drop in performance. Now when a layer is set to be dormant it will not require any system resources, and therefore mean we can have large stacks of layers on our timeline and not have to worry about performance issues.

Setting a layer to be dormant

To make a layer dormant simply set the brightness to 0. This will change the colour of the layer to a dark green to signify the layer is now dormant. Raise the value above zero to make the layer active.

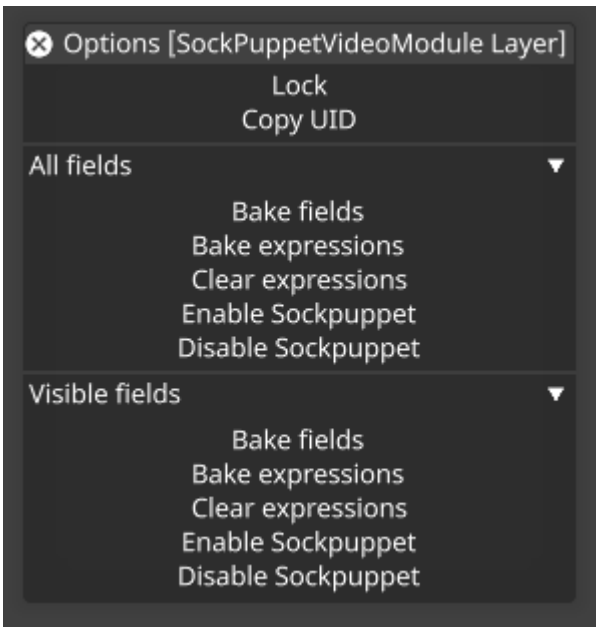
RGB colour overview

When running in Sockpuppet mode the colour pallette system with its X and Y values are ignored and instead replaced by a RGB colour system. This is to reflect the typical way of working with colour on a lighting console.

Layer Field Options

When editing a layer that is being controlled by Sockpuppet, there are additional controls available to change how each field of a layer is controlled.

Layer Options



When right clicking on the titlebar of a layer there are additional options to control the enabling or disabling of Sockpuppet across the entire layer. There are three different rollouts in which you can control, they are:

1. All fields
2. Selected fields (when editing multiple layers)
3. Visible fields (If a section/group is closed it will not be impacted)

These buttons allow users to quickly disable Sockpuppet across an entire layer and re-enable only the fields that should be controlled.

Field Options

When right-clicking on a single field there are changes to the field options when the layer is controlled by Sockpuppet.

Metafield

Metafields do not affect fields when they are being controlled by Sockpuppet, ranges and step size should be controlled through the personality editor instead.

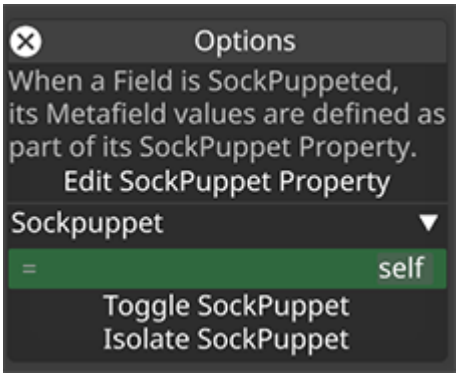
Toggle Sockpuppet

The Toggle Sockpuppet feature allows you to selectively disable the Sockpuppet functionality for any parameter within a patched layer.

For example, if you have a Bitmap layer patched and you want Sockpuppet control over all parameters except Brightness, which you want to keyframe, you can now right-click on the Brightness parameter and select “Toggle Sockpuppet”.

This action removes that parameter from Sockpuppet mode, allowing it to be animated or keyframed independently.

When a field is Sockpuppeted, its metafields are determined by the Sockpuppet property, as part of the layer's Sockpuppet personality. The widget is linked to this property, allowing you to edit the associated metafields directly.

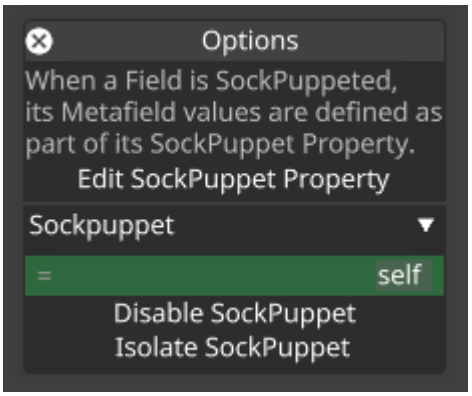


Isolate Sockpuppet

The Isolate Sockpuppet feature keeps Sockpuppet enabled only on the selected parameter, while disabling Sockpuppet functionality on all other parameters within the patched layer.

This allows you to focus Sockpuppet control exclusively on a single parameter without interference from others.

To revert this change, use De-isolate Sockpuppet, which restores the original Sockpuppet settings for all parameters in the layer.



HTTP Sockpuppet

HTTP Sockpuppet allows sockpuppet control of layers via a HTTP API. This page will give an overview of the functionality possible with HTTP sockpuppet, its limitations and how to set it up in Designer.

Patching a Layer

To control the layer via HTTP Sockpuppet, the layer must be patched.

1. Right-click the layer on the timeline.
2. Click **Patch to HTTP...**.

The layer will turn green and by all fields of the layer will be patched, turning green. When a field is patched, it can no longer be controlled with keyframes and will only be controlled by HTTP sockpuppet.

Unpatching a layer

Individual fields can unpatched by:

1. Right-click the field on the layer.
2. Click **Toggle Sockpuppet** to patch / un-patch this specific field.
3. Click **Isolate Sockpuppet** to un-patch all other fields on the layer.

Controlling the layer

Once patched the layer can be controlled with the `/api/session/sockpuppet` endpoints. Querying the available patches with `/api/session/sockpuppet/patches` will return the available patches that can be controlled, and the fields available for each one. Values can then be changed with the `api/session/sockpuppet/livechange` endpoint. For more detail see the API documentation

Animating numeric values

HTTP is not suitable for streaming values, animations are implemented to support animating values over time, rather than sending a stream of values like you would with DMX over ethernet or OSC.

Animations can be applied to a numeric field of a patch (e.g a layer) and the animation will be executed in sync across all the machines in the session. If a machine is dropping frames, you

may see issues with synchronisation.

Animations are triggered with a start value, end value, a duration of the animation and the easing method to use. The start value will be set on the field and interpolated to end value, over the course of the duration using the easing method supplied.. Examples of interpolation methods are Linear, Cubic, Quadratic.

Easing methods available can be queried with the `api/session/sockpuppet/easingfunctions` endpoint.

Considerations

HTTP sockpuppet changes are executed with a ~2 frame delay, to allow the change to be distributed across the session, and applied in a synchronised manner.

Once a value had been set via HTTP sockpuppet, it will always be set to the last value set (or final value of the animation) until d3 is restarted. There is no expiration.

The current value or a field on a patch (e.g field on a layer) is provided in the result of the `api/session/sockpuppet/patches` endpoint, but it is not possible to stream the realtime values of a field.

HTTP sockpuppet changes should be sent to the acting Director. This will depend on failover state.

Motion Capture Overview

Motion Capture includes:

- Skinned Mesh Skeleton
  - Skeleton
  - Skeletal Point Tracking Source
- Vicon
  - Vicon Emulator

Vicon Emulator

Vicon Emulator

The following information will help you configure the Vicon Virtual System with Disguise systems. The Vicon Virtual System will allow you to emulate connections to a live Vicon system.

Prerequisites

- Download and install the Shōgun software.
- Download and install the Vicon Virtual System.
- Download the Camera_Tracking_Vero_V22.zip camera calibration sample data package.

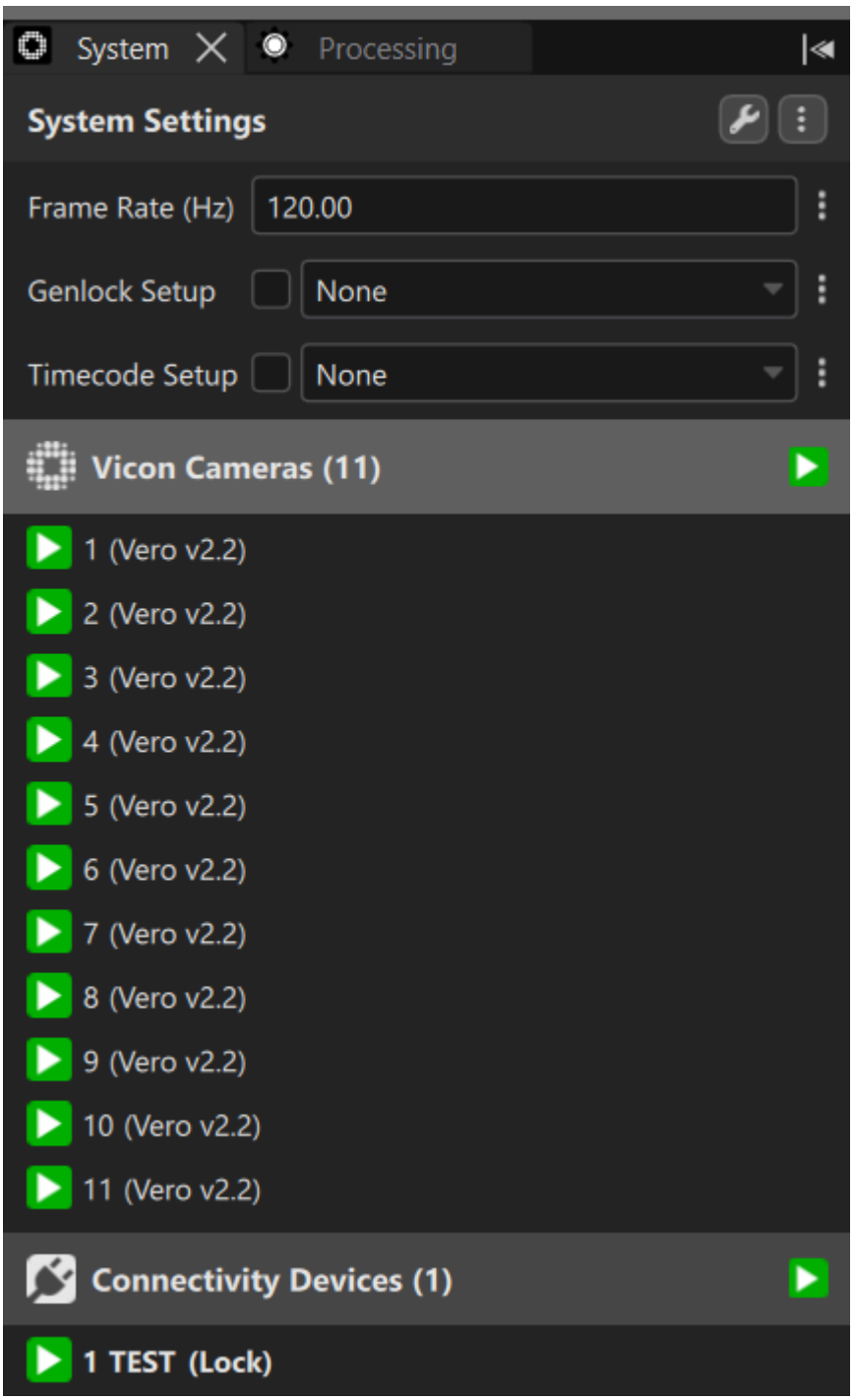
Request a license

The Vicon Shōgun Live software requires a license, which can be requested by opening the program and following the instructions for the licensing popup.

Set up the Vicon Virtual System with Disguise

1. Open the **Vicon Shōgun Live** application.
2. Open **Virtual System** and open **Camera Data File** and select oneof the **.x2d** files in the **Camera_Tracking_Vero_v22** sample data and click **Stream**.

3. In Shōgun, you will see the cameras populate the **Vicon Cameras** list.

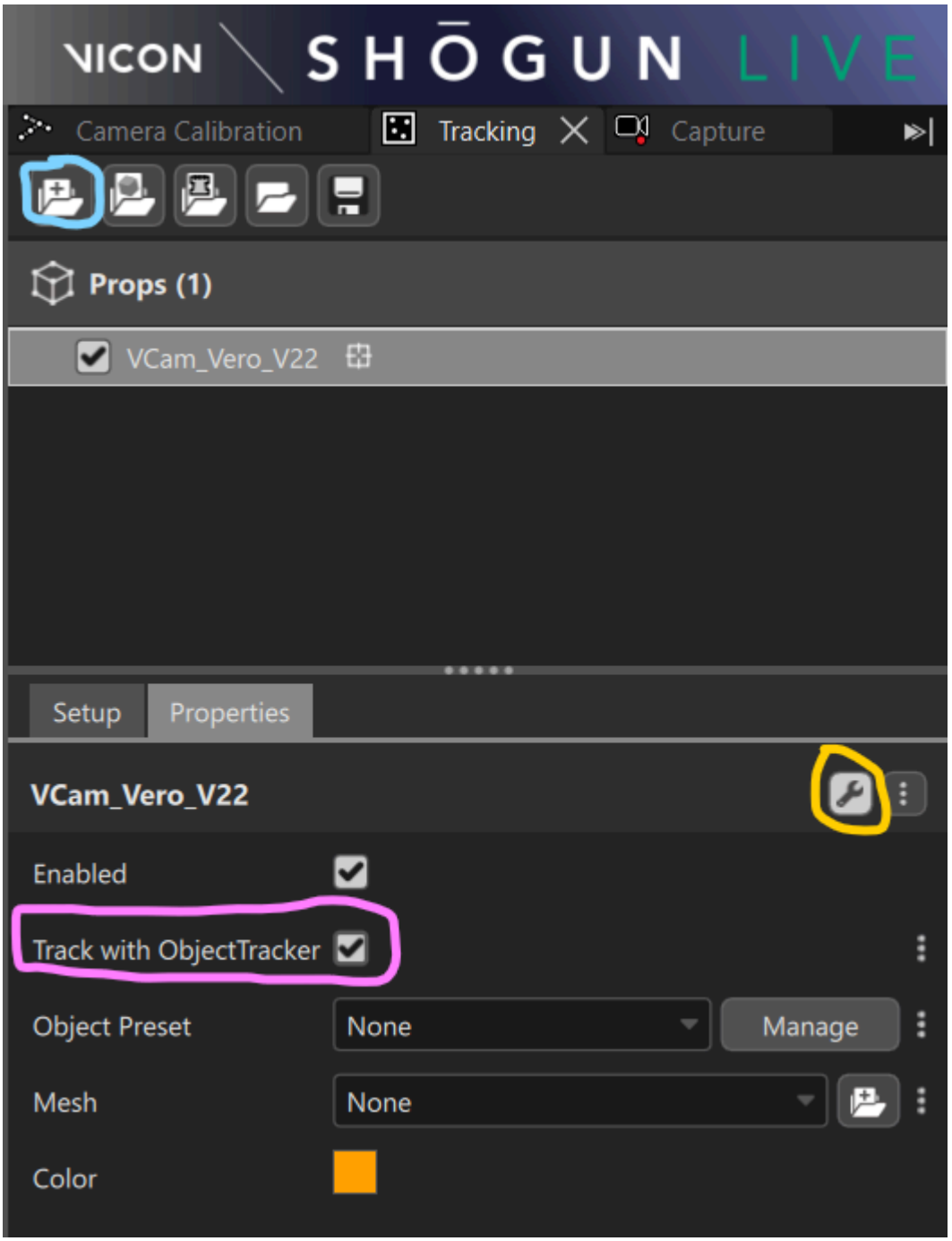


4. In the **Camera Calibration** panel, open the folder and select **Vero_V22_calibration.xcp** file in the sample data package.



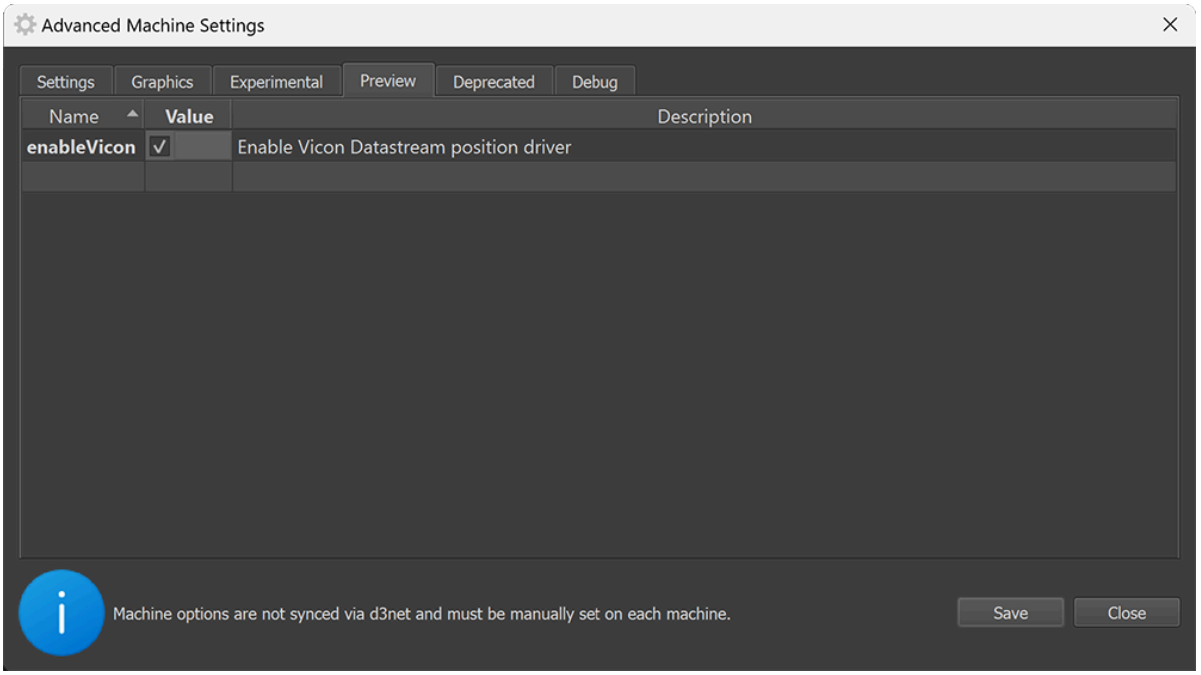
5. In the Tracking panel, click Import Subject (highlighted in blue) and select VCam_Vero_V22.vsk from the sample data.

6. Click **Advanced Parameters** (highlighted in yellow) and enable Track with **ObjectTracker** (highlighted in pink) in the **Props & Properties** panel.



7. In d3manager, navigate to **Machine Settings > Advanced Machines Settings**.

8. Click and then insert the cursor in the **Name** field of a new line and add the command **enableVicon**. All machines will need this configuring to enable the datastream.



9. Click the checkbox to enable it and then click **Save**.
10. In a Designer project, create a **Device > Position Receiver**.
11. Add a **ViconDataStreamDriver**. The default IP of 127.0.0.1 is suitable for local setups.
12. Engage the device.
13. Create a camera or a prop and set the Tracking Source to **Vicon**.

Further reading

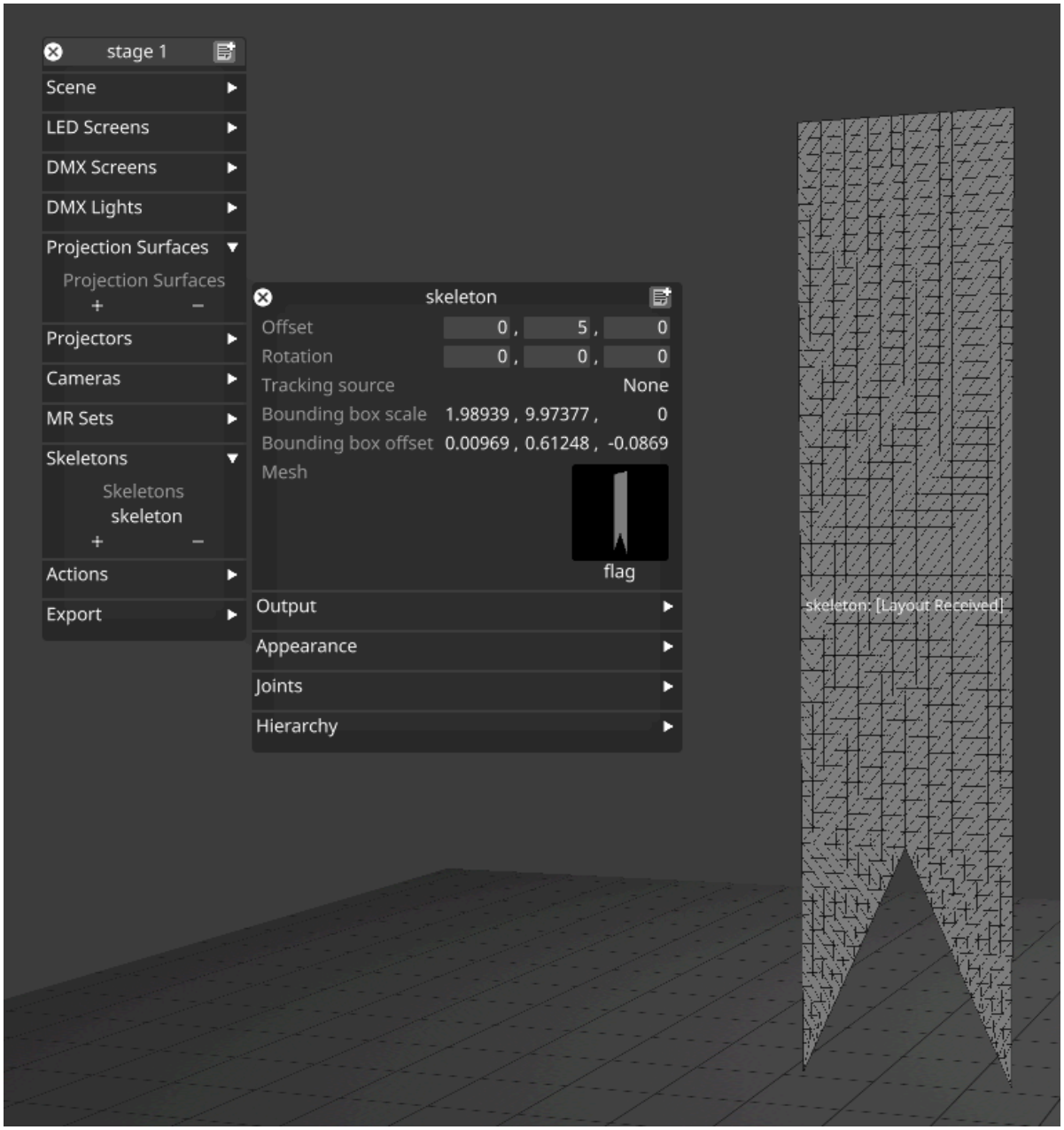
Creating PositionReceivers and adding Tracking Sources.

Skeleton

The Skeleton object is the starting point for mocap and skinned mesh workflows in Disguise. A skeleton consists of a rigged structure of bones, which is either provided by a tracking source or imported from a rigged FBX mesh. These bones can be moved to deform a skinned mesh within Disguise or sent over RenderStream to control a Skeleton in an external render engine.

Creating and editing a new skeleton

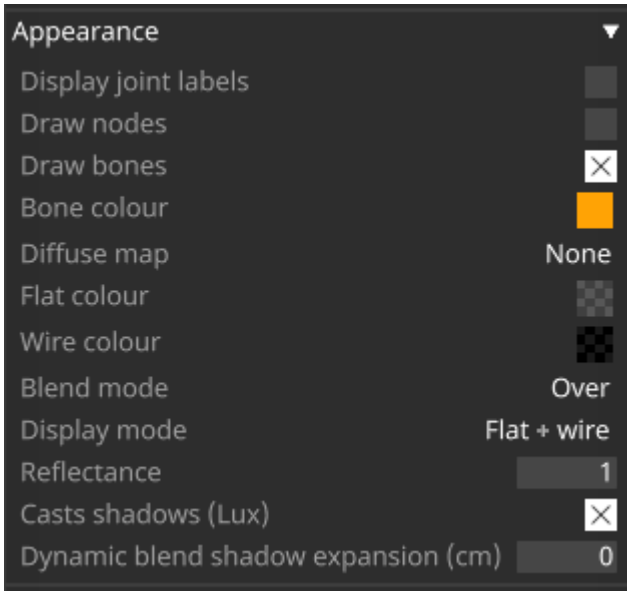
1. In the stage editor, under **Skeletons**, click the **+** button to add a new skeleton with a default bone structure.



Setting up a skeleton

2. Use the **Mesh** field to select an FBX rigged mesh for the skeleton.

3. In the **Appearance** separator options can be set to control the rendering of the skeleton in the visualiser, for example showing or hiding the bones and the mesh.



Changing skeleton appearance

4. In the **Joints** separator the skeleton can be posed manually by adjusting bone offsets and rotations.



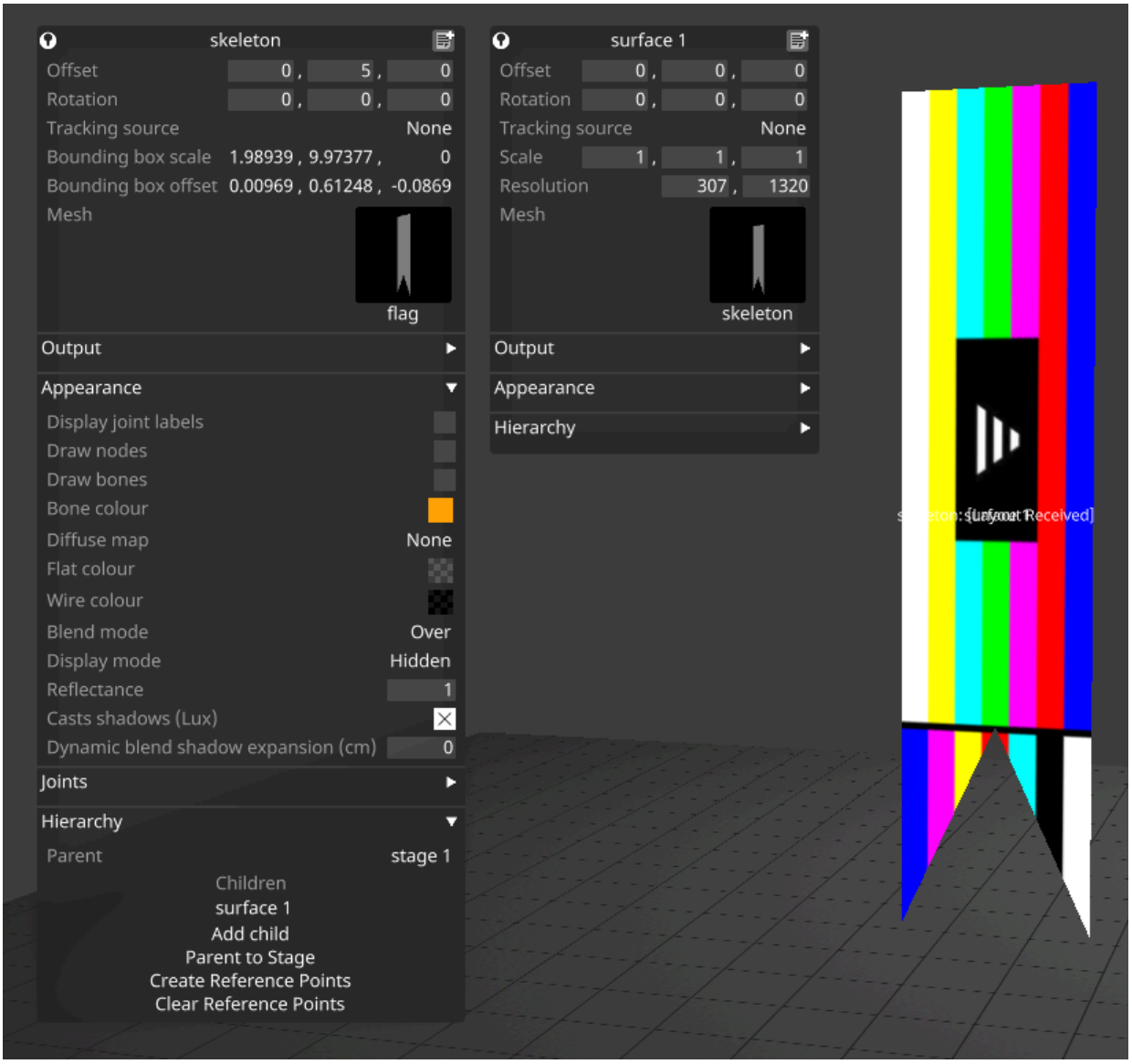
Skeleton joint posing

5. To reset the skeleton pose to zero, click the **Zero Pose** button in the **Joints** separator.

Setting up a skinned mesh for a projection surface

1. Find or create an FBX file which contains a single rigged mesh, and copy this into the project's **objects/Mesh** folder.
2. Create a new skeleton and select the FBX mesh in the skeleton's mesh field.
3. Create a new projection surface and select the mesh named after the skeleton (not the original FBX mesh).
4. Try moving the skeleton's bones using the manual posing controls in the skeleton editor. The mesh should deform in both the skeleton and the projection surface.
5. To align the projection surface exactly with the skeleton, add the projection surface as a child of the skeleton and set the surface's offset and rotation to zero.
6. If you want to hide the skeleton mesh, set the Display mode under Appearance to **Hidden**.

> **Note**
>
> **Please note**: FBX files containing multiple meshes are not currently supported.



Projection surface with skinned mesh

Playing back an animated skeleton

Skeleton animations can be played back on the timeline using the **scene animation layer**. To do this, load an animated FBX skeleton and select it as a mesh as normal. Then follow the instructions for the **scene animation layer** to play back the animation.

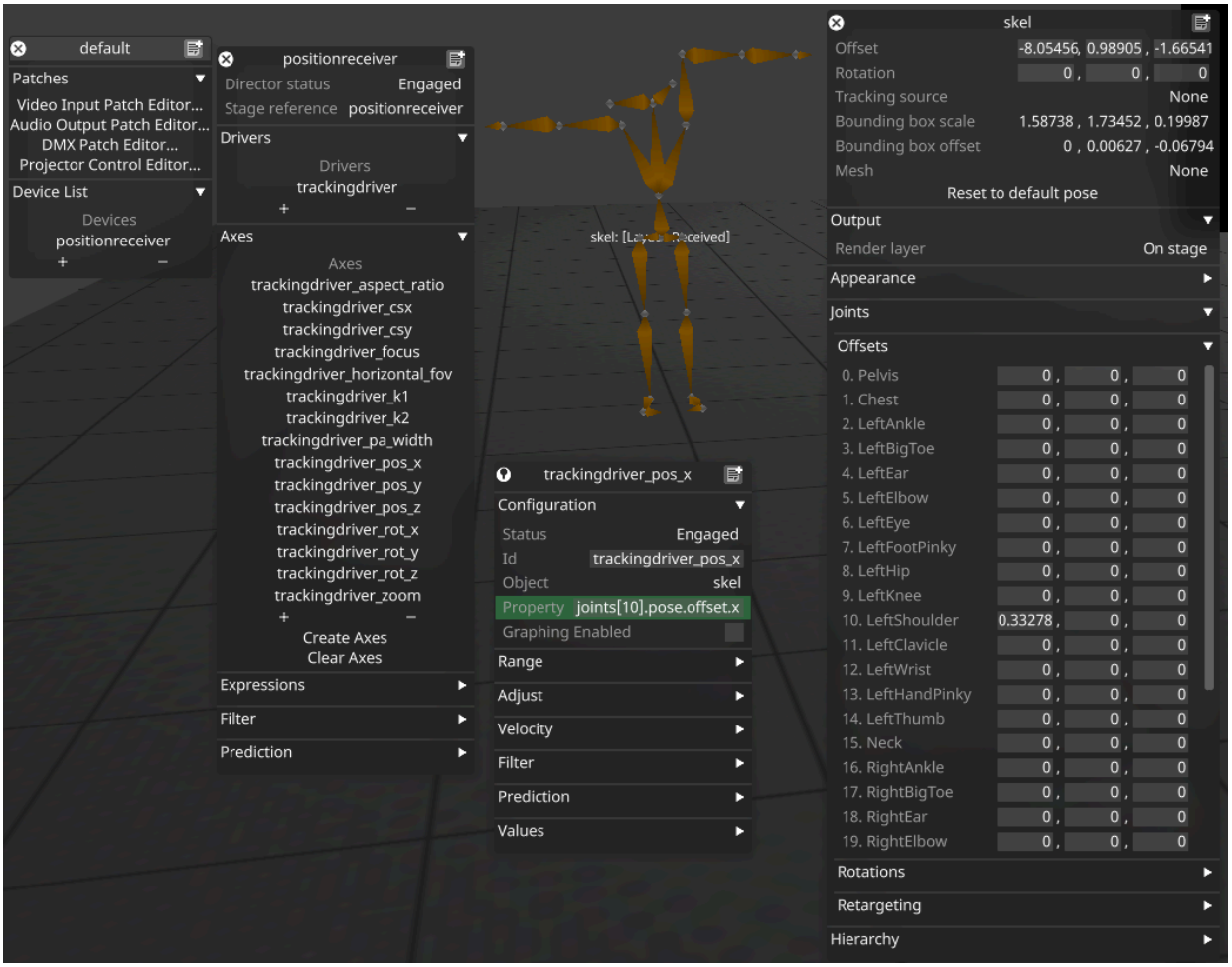
Controlling a skeleton using tracking data

To use tracking data to control a skeleton's bones, you will need to create a skeletal tracking source or driver. Available tracking sources include:

- Skeletal point tracking source

Alternatively, individual joint positions and rotations can be controlled using tracking **axes**. To control a skeleton joint with a tracking axis:

1. Create a new Position Receiver under **Devices**.
2. Open the Position Receiver editor, and add the appropriate Driver for your incoming tracking data under **Drivers**.
3. Under the **Axes** separator, click **Create Axes**.
4. Open the editor for the axis you want to use to control a Skeleton joint.
5. Under the **Configuration** separator, select your skeleton in the **Object** field.
6. In the **Property** field, type an equation to control the desired joint property, for example `joints[10].pose.offset.x`.
7. The expression is formed as follows:
   - The index inside the square brackets is the index of the joint. This can be found by expanding the joint offset or rotation separators in the Skeleton editor.
   - After `pose`, type either `offset` or `rotation`, depending on which property you want to control.
   - Type `x`, `y`, or `z` at the end of the expression to define which axis is being controlled.



Controlling skeleton joints with a tracking axis

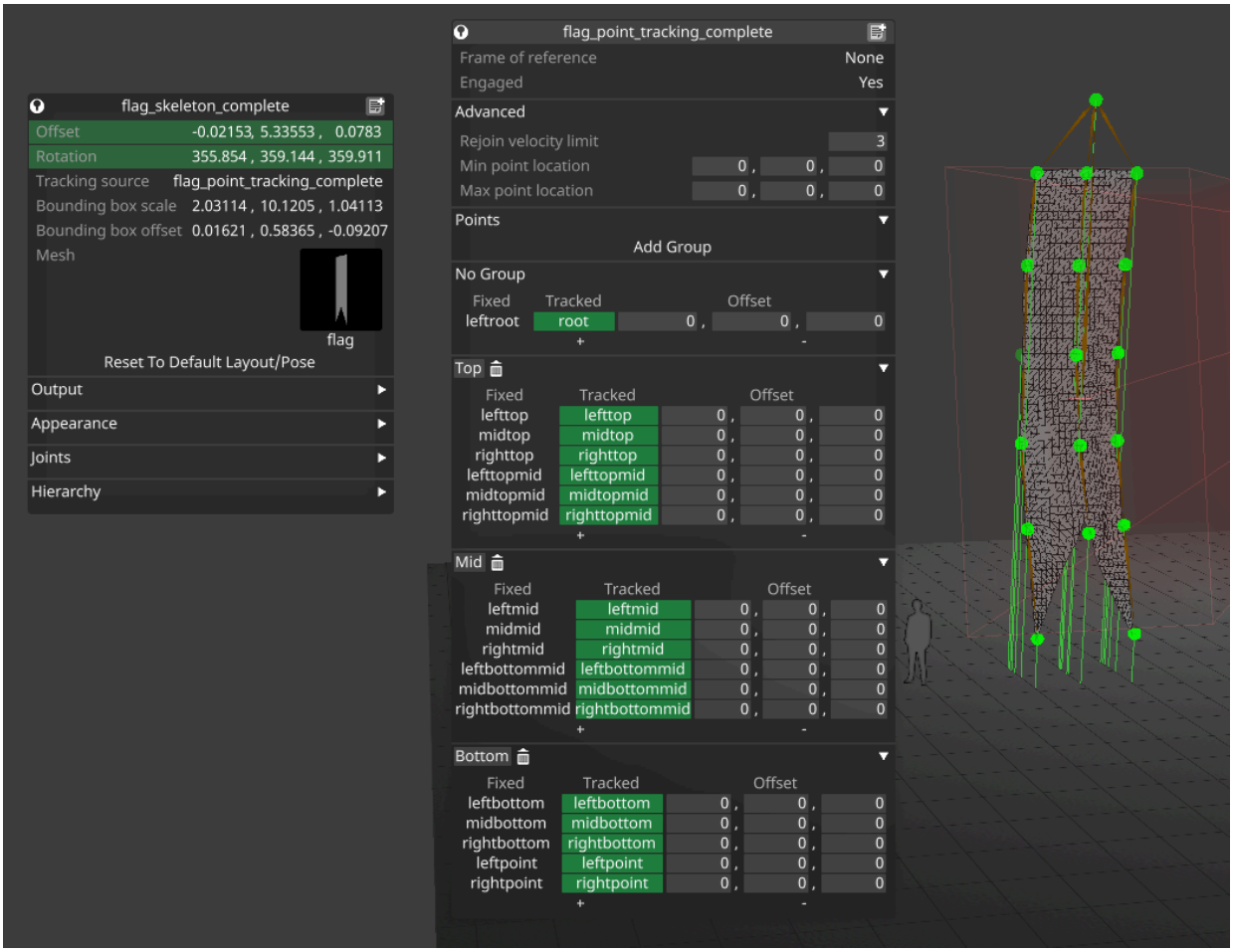
Skeletal Point Tracking Source

The Skeletal Point Tracking Source allows the joints of a skeleton to be controlled using tracked points from a marker-based tracking system. It can be used in conjunction with tracking drivers such as:

- RTTrPM Driver (BlackTrax)
- NatNet Driver (OptiTrack)
- Posi Stage Net Driver

Setting up a skeletal point tracking source to control a skeleton

1. Set up a suitable tracking system and driver to receive tracked points in Disguise
2. Create a rigged mesh with joints at each tracked marker position, export as FBX and select in a skeleton
3. Under Hierarchy in the skeleton, click **Create Reference Points** to create a reference point for each joint
4. Create a new SkeletalPointTrackingSource and set it as the skeleton's tracking source
5. In the tracking source editor, add points to link each tracked point to its corresponding reference point
6. Points can be sorted into groups. If a point stops receiving tracking data, its position will be estimated based on a rigid body transform from other receiving points in the same groups
7. Each point can also have an offset applied relative to the fixed point
8. Adjust the **Rejoin velocity limit** under **Advanced** to change the speed at which points catch up with the tracked position when tracking is received again after being lost.
9. Adjust the **Min/Max point locations** under **Advanced** to set bounds on the area in which tracked points are followed.

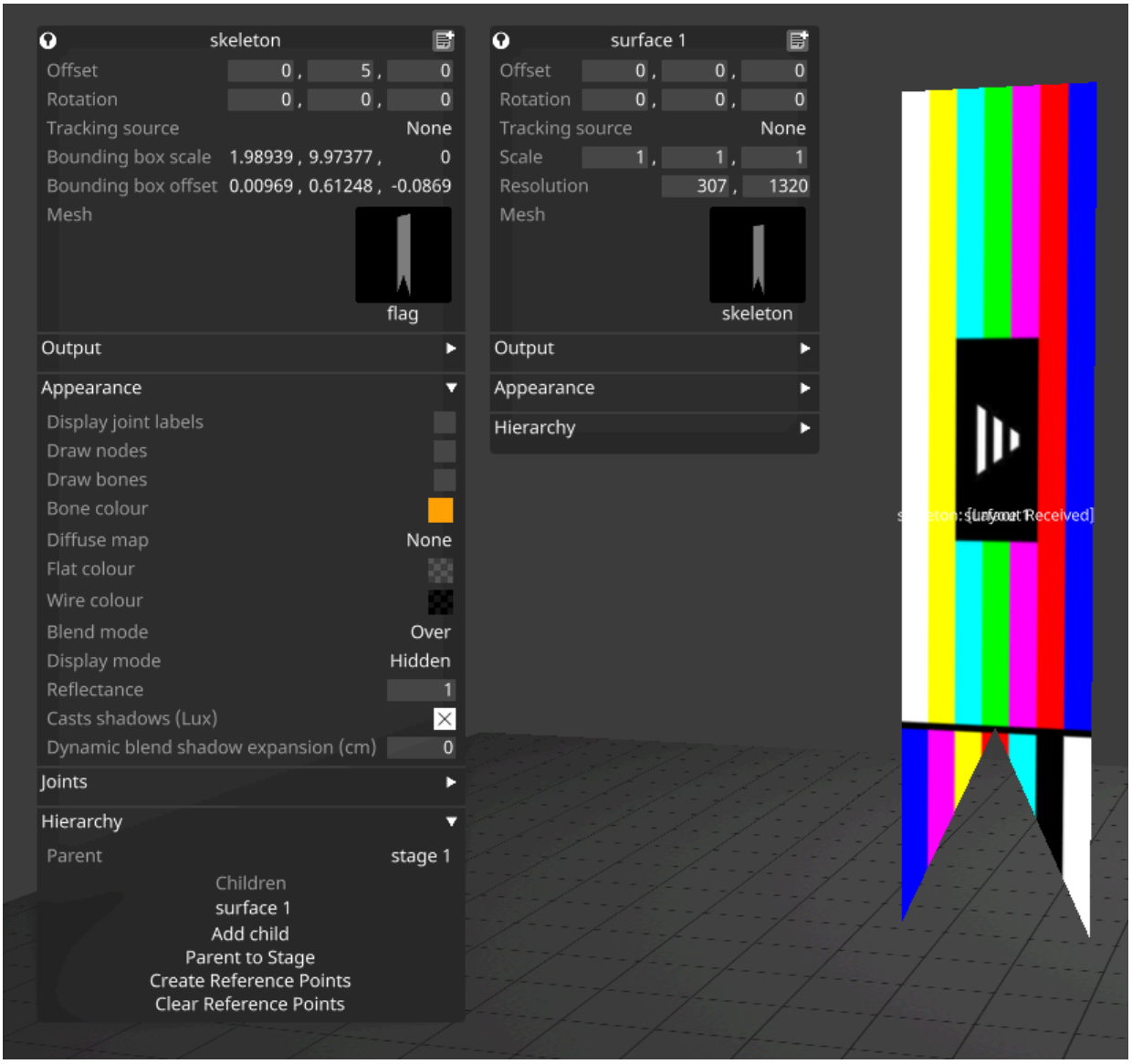


Skeletal point tracking source setup

Using a skeletal point tracking source to track a deforming surface

The skeletal point tracking source is ideal for tracking a deforming surface. To do this:

1. Create and rig a FBX skeletal mesh which matches your deforming surface. See below for tips on how to do so.
2. Set up and track a skeleton using your skeletal mesh, following the steps above.
3. Create a new surface, and select the deformable mesh which is named after the skeleton, **NOT** the original FBX mesh which is selected in the skeleton. The surface should now deform with the skeleton.
4. Make the surface a child of the skeleton and zero out its position and rotation, so that the origins are aligned.
5. Set the Skeleton **Display mode** under **Appearance** to **Hidden** if you just want to see the surface mesh.
6. Use your surface for projection or other workflows as normal.



Skeleton projection surface

Preparing your skeletal mesh

To achieve optimal tracking of a deforming surface, it is crucial to set up the skeletal mesh as accurately as possible. Below are some tips for how to model and rig your mesh:

- Create a bone structure which follows the positions of the tracked point, with a hierarchy appropriate for the structure of the surface. There should be a joint aligning with each tracked point. It is ok to have additional joints which don't correspond to tracked points at the top or bottom of the hierarchy, but it is not recommended to place untracked joints between tracked joints.
- The base layout of the skeleton should match an average/neutral pose of the real surface as closely as possible. One way to ensure this is by capturing the locations of the tracked points with the surface in place, loading these into your modelling software, and adjusting your joints to align exactly with the captured tracked points.

- The skeletal mesh should be exported with its base positioning fully captured in the layout, with a fully-zero pose. When loading the skeleton into Designer, you should see all zeros in the joint positions and rotations in the skeleton editor. If this is not the case, you may need to reset the bind pose and/or remove pose offsets and rotations in your modelling software.
- It is important to weight the vertices as accurately as possible to model the deformation of the surface as the tracked points move. The best way to do this will depend on the physical properties of the surface itself, however a good starting point is to ensure that all vertices adjacent to tracked points are fully weighted to the corresponding joint, so that they stick to the tracked point as it moves around. The remaining weightings should be applied roughly based on the distance to the corresponding joint. Note that auto-weighting algorithms often weight based on the distance to the bone rather than the joint, which can lead to the modelled surfaces diverging away from the tracking markers.
- Joints which are at the end of the hierarchy may not be able to extend along the length of the bone (e.g. to follow a stretching surface) since they do not have a bone beneath them. In this case it can be beneficial to add additional small ‘dummy’ bones at the end of the hierarchy, not weighted to any vertices. This will allow full control of the previous end joints.

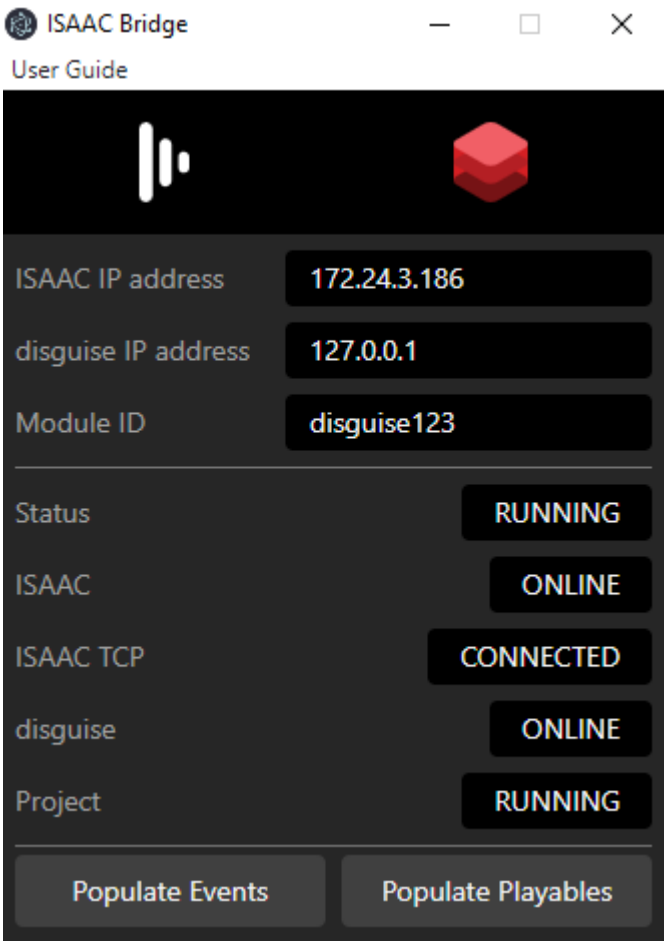
ISAAC Bridge

ISAAC is a multifaceted platform from **Smart Monkeys** that redefines the way AV systems are deployed, monitored and managed. Covering a spectrum of elements from lighting and audio to video projections and displays, it allows users to monitor and manage all elements of an AV system from a single on-premise portal without the need for internet connection. Users are also able to distribute and schedule content playback remotely using ISAAC.

Integrating ISAAC with Designer extends its functionality to both technical and non-technical users, enabling the user to remotely monitor and manage the Disguise platform alongside their wider AV system- removing the need for 24/7 on-site monitoring. ISAAC bridge creates a simplified and unified experience for users running fixed installation projects.

Learn more about ISAAC **here**.

The ISAAC Bridge app is currently still in development. For more information, please contact our **Product Team**.



Workflow

Installation

ISAAC Bridge should be installed on a machine on the d3Net network.

ISAAC Bridge installs to `Windows\Start Menu\Programs\Disguise`.

ISAAC Bridge should be installed on the Director machine when run across a multiple machine setup. When Disguise is running on a single server, ISAAC Bridge should be installed on that server.

Interface Overview

The ISAAC Bridge interface is split into three sections:

- **Configuration**
- **Status**

Configuration

On first launch, set the following:

- IP address of the ISAAC server
- IP address of the Disguise machine
- Module ID of the module in ISAAC

In a multi-machine setup the IP address of the Disguise machine will be the Director. This configuration persists between restarts.

Status

Bridge Status

Status of the ISAAC Bridge application, either `RUNNING` or `STOPPED`.

ISAAC

Status of the ISAAC server, either `ONLINE` or `OFFLINE`.

ISAAC TCP

Status of the connection to the ISAAC server, either `CREATED`, `CONNECTED`, `ERROR` or `CLOSED`. This is the connection by which scheduled events are received by Disguise.

Disguise

Status of the Disguise machine, either `ONLINE` or `OFFLINE`.

Project

Whether there is a Disguise project currently running on the specified Disguise server. Either `RUNNING` or `NOT RUNNING`.

Supported ISAAC Features

Events

ISAAC Bridge exposes two events to ISAAC:

- **Start Project** - starts the last run Disguise project
- **Stop Project** - stops the currently running Disguise project

To populate these events in ISAAC, click **Populate Events**.

Playables

ISAAC Bridge exposes Tracks on the Default Transport in Disguise to ISAAC as Playables. To populate ISAAC Playables with available Disguise Tracks, ensure that a Disguise project is RUNNING and click Populate Playables.

Variables

Variables exposed to ISAAC:

Running Project - project path of the running Disguise project.

Volinga 3DGS



Volinga provides an end-to-end pipeline for generating, optimising, and exporting volumetric assets for VFX, virtual production, broadcast and film.

Volinga transforms 2D image or video datasets into photorealistic 3D environments and provides a pipeline for utilising **3D Gaussian Splatting (3DGS)** scenes in Unreal Engine based workflows.

For more information on Volinga, refer to the **Volinga website**.

For full workflow documentation, refer to the **Volinga documentation**.

3D Gaussian Splatting (3DGS)

3D Gaussian Splatting (3DGS) is a real-time radiance field representation developed by INRIA and the Max Planck Institute.

In 3DGS, a scene is reconstructed as a collection of optimised 3D Gaussians, each containing colour, opacity, and spatial data.

For current Disguise workflows, 3DGS is the supported representation for volumetric capture, while earlier NeRF-based workflows have been deprecated.

Volinga Suite

Volinga provides the following products:

- **Volinga Suite** – for creating and editing 3D Gaussian Splatting (3DGS) content.
- **Volinga Plugin** – for integrating 3D Gaussian Splatting (3DGS) content into Unreal Engine workflows and into Disguise via RenderStream.

Volinga supports ACES and OCIO for accurate colour management.

Integration

Volinga brings 3DGS volumetric content into the Disguise ecosystem.

For all new projects, the Volinga Unreal Engine Plugin together with the Disguise Unreal Engine RenderStream Plugin is the recommended workflow. The native Volinga RenderStream plugin remains available for legacy projects but is no longer the focus of feature development or active updates.

Unreal Engine 5 Plugin

Unreal Engine via RenderStream (Recommended)

This workflow uses the Volinga Unreal Engine plugin to render 3DGS content inside Unreal Engine, with frames streamed into Designer using **RenderStream** via the Disguise Unreal Engine RenderStream plugin.

Scene Composition
Import `.nvol` or `.ply` assets into the Unreal Content Browser and place multiple volumetric actors within a single level.

Camera Integration
3DGS actors respect Unreal Cine Camera parameters, including focal length, sensor size, and camera transforms.

Production Tooling
Supports advanced relighting, optimisation controls, and access to Volinga Plugin Pro features.

This workflow aligns with standard Unreal Engine integrations within the Disguise ecosystem.

Legacy

Designer (r24.4+) supports direct playback of `.nvol` assets without a game engine.

This workflow maps volumetric content directly to LED surfaces without the need for Unreal Engine. However, it is important to note that you will not have access to advanced relighting,

the latest optimisation features, or any ongoing feature expansion. For legacy workflow information, refer to the **Volinga documentation**.

Supported asset formats

Volinga supports the following 3DGS asset formats:

- **NVOL** – Generated by Volinga Suite. Includes extended metadata, including colour information.
- **PLY** – 3D Gaussian Splatting assets generated by supported third-party pipelines.

NVOL remains the primary output format of Volinga Suite.

NeRF support

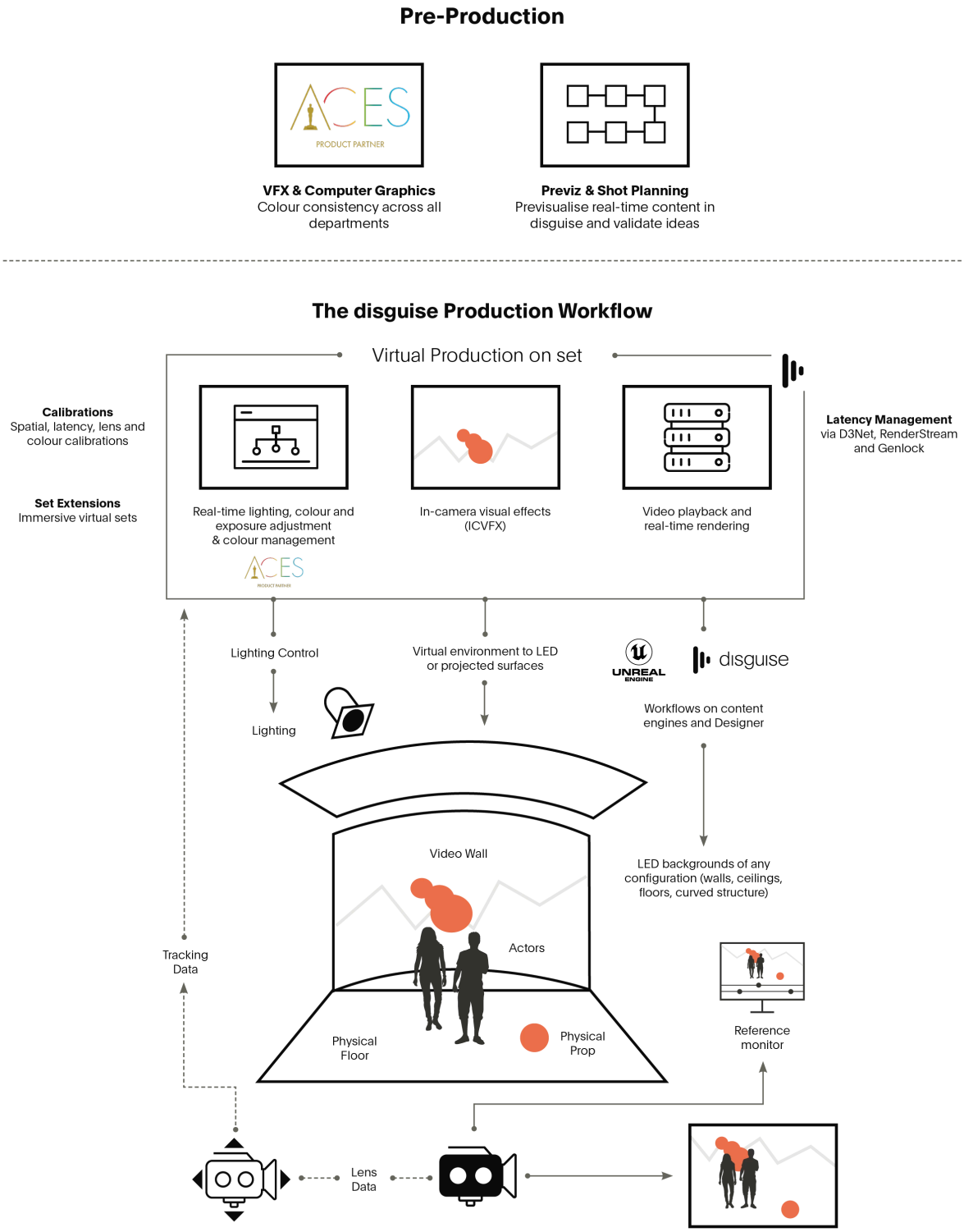
While 3DGS is the recommended standard, Volinga continues to support Neural Radiance Fields (NeRF) for specific use cases like complex refractive surfaces.

Virtual Production Overview

The Disguise virtual production workflow enables production teams to capture the creative magic of working together as they would with physical sets, while minimising the amount of time and resource needs of conventional post-production VFX.

Virtual Production Toolkit

Using state-of-the-art technology that enables real-time visuals to be shown live for actors and crew to respond to, virtual production opens up a world of opportunity. The toolset is already pushing the boundaries for filmmakers, with more exciting uses rapidly emerging.



The Disguise Production Workflow

Virtual Production Workflows

The Virtual Production Workflows at Disguise include:

- 2.5D Workflow
- Anamorphic Lenses

- ARRI Skypanels Setup
- Cuebric Integration
- Green Screen Workflow
- Mesh Mapping Workflow
- Shot Recorder Setup
- Sony's Moiré Alert in Sony Cameras

2.5D workflow

With Disguise's 2.5D workflow tools, you can achieve the parallax effect of real-time 3D content with the simplicity of video plates.

What is a 2.5D workflow?

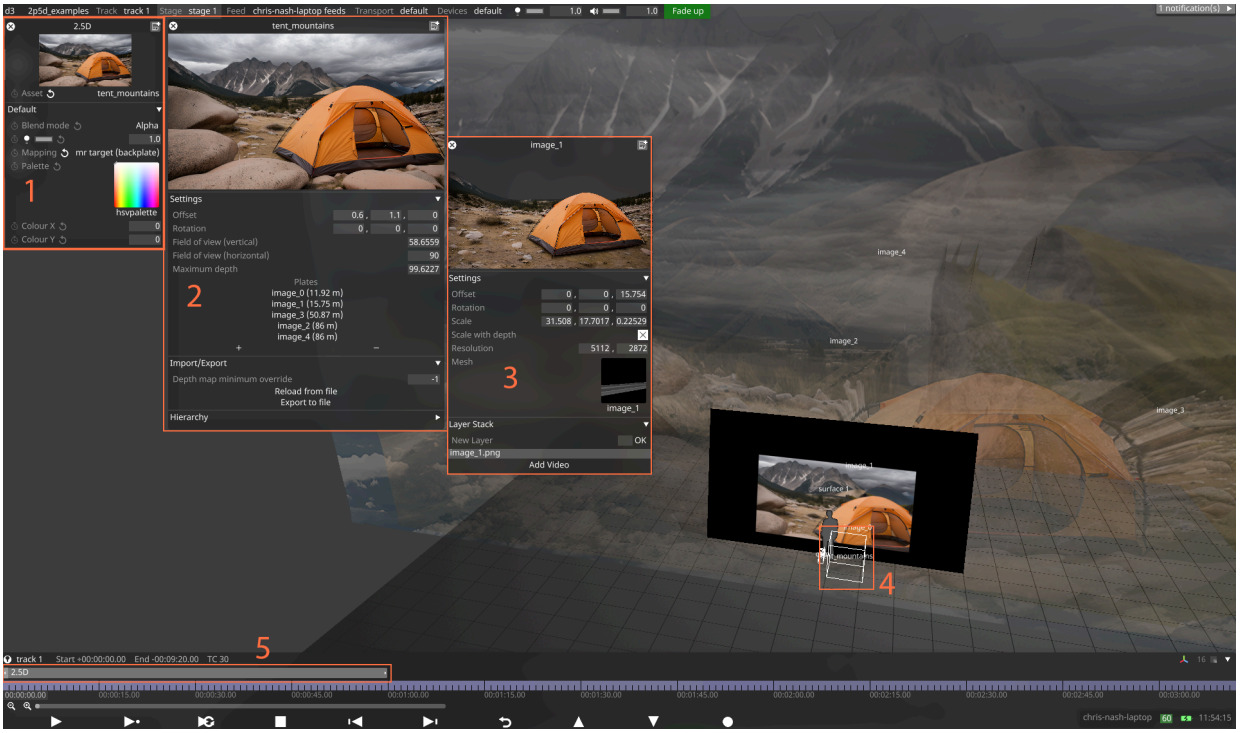
2.5D workflows bridge the gap between traditional 2D video plates and full 3D generative scenes. By building up a scene with layers of images or video plates, with the possibility to add depth or shaping to the plates, users can quickly create a realistic looking scene with parallax. 2.5D assets can be saved as .2p5d files and shared across projects, or created using external integrations such as Cuebric to harness the power of generative AI.

> **Tip**
>
> See the **Developer Documentation** site for a description of the data format used for .2p5d files.

2.5D workflow

This section details the following how-to guides:

- Set up a 2.5D asset from an existing .2p5D file
- Create a new 2.5D asset from scratch
- Modify 2.5D asset properties
- Modify 2.5D plate properties
- Advanced workflows



Key

1. 2.5D Layer editor
2. 2.5D Asset selector
3. 2.5D Plate editor
4. 2.5D Asset origin
5. 2.5D Layer

Set up a 2.5D asset from an existing .2p5D file

Use this workflow when receiving assets from Cuebric or another existing project.

1. Copy the **.2p5d** file to **objects/2p5DFile** in the project folder.
2. Add a 2.5D layer to the timeline.
3. In the **2.5D** layer editor, click on the **Asset** field. The copied 2p5d file should appear in the list of assets in the **2.5DAsset** selector.
4. Click on the asset to select it. It will then load and be visible in the stage.
5. In the 2.5D layer editor **Mapping** property, select the desired 3D mapping e.g. MR set backplate.
6. To reload the asset from file and overwrite any local changes, click **Import/Export & Reload from file** in the property editor of the 2.5D asset.

Create a new 2.5D asset from scratch

Use this workflow to build a layered parallax scene directly inside Designer.

1. Add a 2.5D layer to the timeline.
2. In the 2.5D layer editor, click on the asset field and create a new 2.5D asset.
3. Add a new 2.5D plate to the **plates** field in the 2.5D asset.
4. Under **Layer Stack** in the plate editor, click **Add Video** to add a video or image to the plate.
5. Continue adding plates and modifying their properties until the desired scene composition is achieved.
6. Select the desired 3D mapping in the 2.5D layer, e.g. MR set backplate.
7. To export the asset for use in other projects, click **Export to file** under Import/Export in the 2.5D asset editor. The exported .2p5d file will appear under objects/2p5DFile in the project folder.

Modify 2.5D asset properties

The 2.5D asset acts as the “container” for all plates.

- **Origin (asset origin box)**
  The 2.5D asset is an Object which can have its offset and rotation set by changing the values in the editor, or moving the 2.5D Asset origin (white box) in the visualiser. Doing so will move all the plates and change the origin point of the 2.5D scene. It is recommended that the origin is set roughly in the middle of the range of movement of the camera, to achieve the most realistic looking parallax.
- **Field of view** & **Maximum depth**
  The **Field of view** and **Maximum depth** fields can be adjusted to scale the total area encompassed by the 2.5D scene. This can be used to ensure that the edges of the plates are not seen when moving the camera within its expected range.
- **Managing plates**
  Plates can be added to or removed from the asset using the **+/-** buttons in the **Plates** field of the asset editor. Plates are ordered by depth, and reordering the plates in the list will update the plates’ depth to reflect their new positions.
- **Import/Export controls**
  The **Import/Export** settings can be used to load or save .2p5d files. When loading files which contain depth maps, the **Depth map minimum override** setting can be used to change the minimum distance from the origin, represented by full white in the depth map.
- **Mode: Normal vs Locked**
  The **Mode** option allows switching between **Normal** and **Locked** mode:
  - In **Normal** mode, playback of 2.5D scenes containing video runs independently
  - In **Locked** mode, playback of videos in the plate’s **Layer Stack** is synchronised to the global track playhead. This ensures the videos in the plate behave like standard video layers and respond consistently to global playback controls and modes.
- **Visibility**
  Enable **Show in stage** to make the asset always visible. If it is not enabled (ticked), the asset will be visible only when a relevant editor is open.

Modify 2.5D plate properties

Each plate acts as a layer inside the 2.5D scene.

- **Depth (z)**
  The plate depth can be changed by modifying the z offset value, or by dragging the plate in the visualiser. If the option **Scale with depth** is ticked, the scale and x/y offsets will automatically be adjusted with depth so that the plate looks the same when viewed from the asset origin. If **Scale with depth** is not ticked, the scale and x/y offsets will not change, so the plate will appear smaller as it moves further away.
- **Scale & position**
  By default, new plates will be scaled to match the total field of view of the asset. However, the scale values can be adjusted in the 2.5D plate editor to change the scale. Similarly, the x and y offsets can be adjusted to move the plate position.
- **Plate mesh**
  The plate's mesh can be modified to introduce depth to a layer, enhancing the realism of the parallax effect. Some .2p5d files will contain depth maps, which are automatically converted into meshes upon import. To remove the depth from the plate, the standard rectangle mesh can be selected instead. It is also possible to add a video depth map using a Depth Map Mesh.
- **Layer Stack**
  The plate **Layer Stack** is used to set up the layers appearing on the plate. To quickly add an image or video to the plate, use the **Add Video** button. Otherwise, layers can be created as normal and stacked to create effects such as blur or colour adjust. Note that these layers can't be keyframed, see **Advanced Workflows** for methods to keyframe layers on plates.

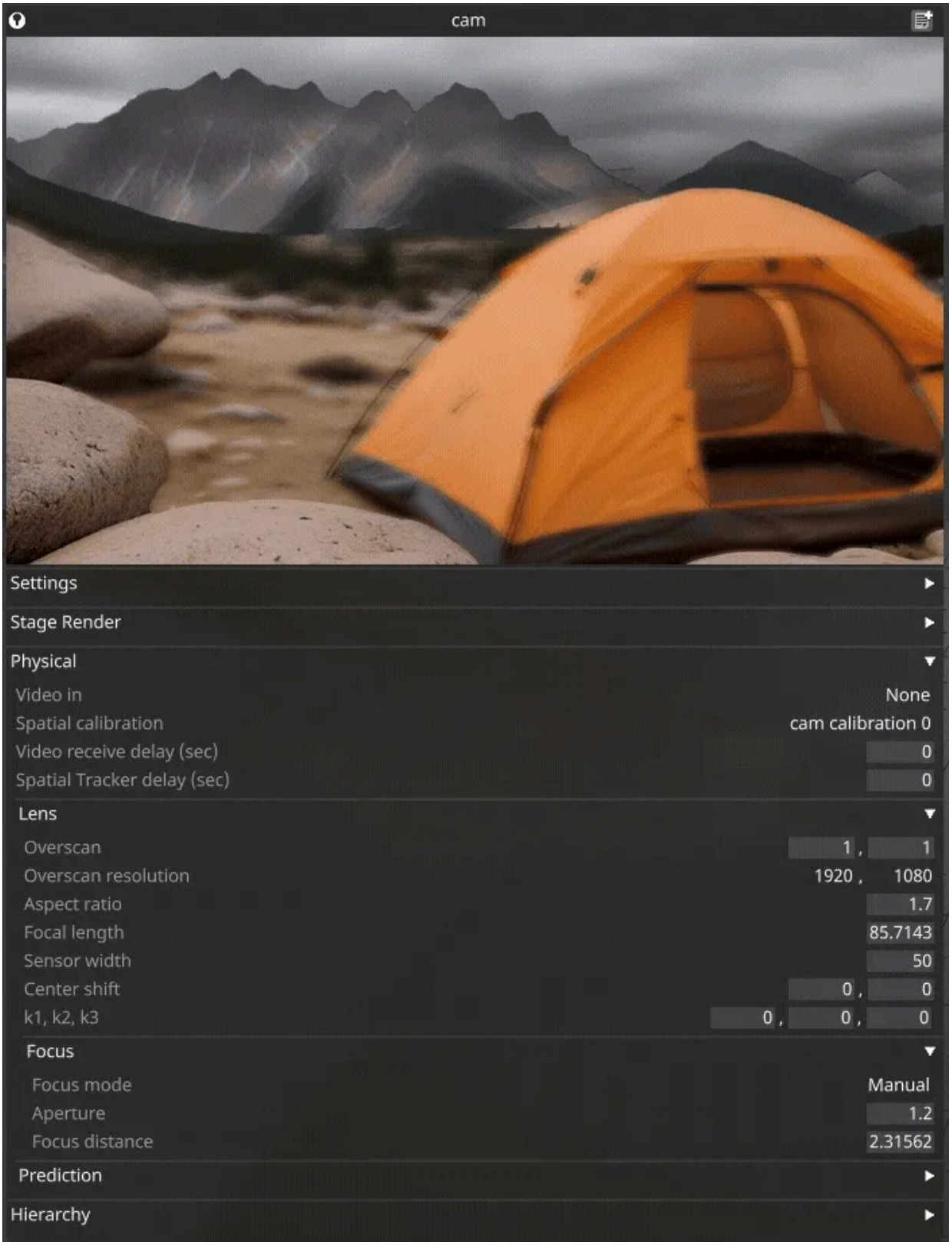
Apply depth of field to 2.5D assets

You can apply DOF to 2.5D assets on virtual cameras.

Depth of field can be applied to renders of 2.5D assets on camera frontplates and backplates, using the **Focus** properties in the camera's **Lens** settings. See Camera Properties for more information about these settings.

To use lens focus parameters with a 2.5D layer:

1. Set up a 2.5D layer as described above.
2. Map the layer to a Camera or MR set's Backplate or Frontplate.
3. Set the active camera's **Focus mode** to **Manual** or **Object tracking**.
4. Adjusting the lens focus parameters in the camera should cause the depth of field effect to change in the rendered 2.5D asset.



Advanced workflows

The 2.5D layer and assets are designed to provide a user-friendly way to quickly create 2.5D scenes which are portable between projects. The standard workflow should allow for the

majority of use-cases, however more advanced workflows can also be achieved by combining 2.5D assets with other tools in Designer.

- **Layer stack for quick composition**
  Generally images, videos and effects should be added to plates using the plate **Layer Stack**. However, this does not allow for keyframing within these layers. To keyframe content on plates, layers should be added to the timeline as normal. Content can be mapped to 2.5D plates from the timeline similarly to any other display.
- **Keyframing via the main timeline**
  To keyframe plate content (e.g. opacity, transforms, effects), add layers to the timeline instead of the Plate Layer Stack, then map those layers to the plate like any other display.
- **Animate assets and plates as Objects***
  Both 2.5D assets and plates are full Designer Objects, which means they can be animated on the timeline or parented to tracking-driven objects.
- **Legacy Stage Render workflow (pre-r26.3)**
  The **Stage render layer** which was used for 2.5D workflows prior to r26.3 still functions as it did previously, so it is still possible to use the original workflow if desired.
- **Using video-driven depth on plates**
  For dynamic depth shaping, set a plate's mesh to a Depth Map Mesh as the plate's mesh, and adding a Depth Video Layer to the plate's layer stack.

Original workflow (available prior to r26.3)

Preserved for historic compatibility.

1. Using the stage menu, add an LED surface and a camera to your project.
2. Create 3 projection surfaces and position them behind the LED screen incrementally with a gap of several metres between each.
3. Label each surface in the composition stack to reflect it's location. e.g. "foreground', "midground", and 'background".
4. Make the following changes:
   1. Set foreground and midground surfaces to **Transparent**
   2. Set the blend mode for the foreground and midground to **Alpha**.
   3. Change the render layer of each surface to **Backplate (MR)**.
5. Using the stage menu, create an MRset.
6. Assign the LED screen to the MRset and change the Camera property to **Camera override**.
7. Add a StageRender layer to the timeline above the other content layers and map it to the backplate.
8. Move the camera within the MRset to activate it.
9. The perspective projection will appear on the LED screen and tracks the movement of the camera.

Related topics

- MRset
- Cuebric Integration
- Depth Workflows

Cuebric Disguise Integration

Cuebric is an AI-powered creative production studio that provides filmmakers and other content creators with powerful and easy-to-use tools to generate and edit photorealistic, fully segmented, textured, virtual environments in seconds.

2.5D versus 3D content on volume stages

For virtual production to work, the content on the volume stage needs to be dimensionalised. For the scene to be credible, the background must also move left with the actor when the actor moves left. The Unreal Engine is used on set by CGI environment designers to render the assets they've already made in real-time. These backgrounds are three-dimensional, just like in a game. Creating realistic, three-dimensional Hollywood-quality backgrounds typically requires an army of CGI artists to work for months.

If 3D is the option that can be fully controlled, then the volume stages also let you film in what is called "two and a half dimensions," which offers less control over movement. Simply put, a 2.5D image is a two-dimensional image with a parallax effect. Cuebric was designed to enable the creation of 2-D landscapes that could be built on set in real-time. It is a quick, cost-effective alternative to labour-intensive 3-D environments and works well with them.

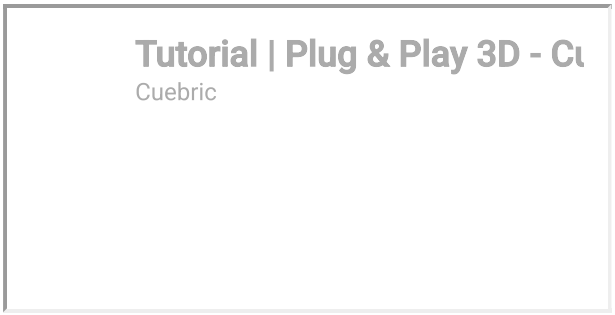
How does Cuebric work?

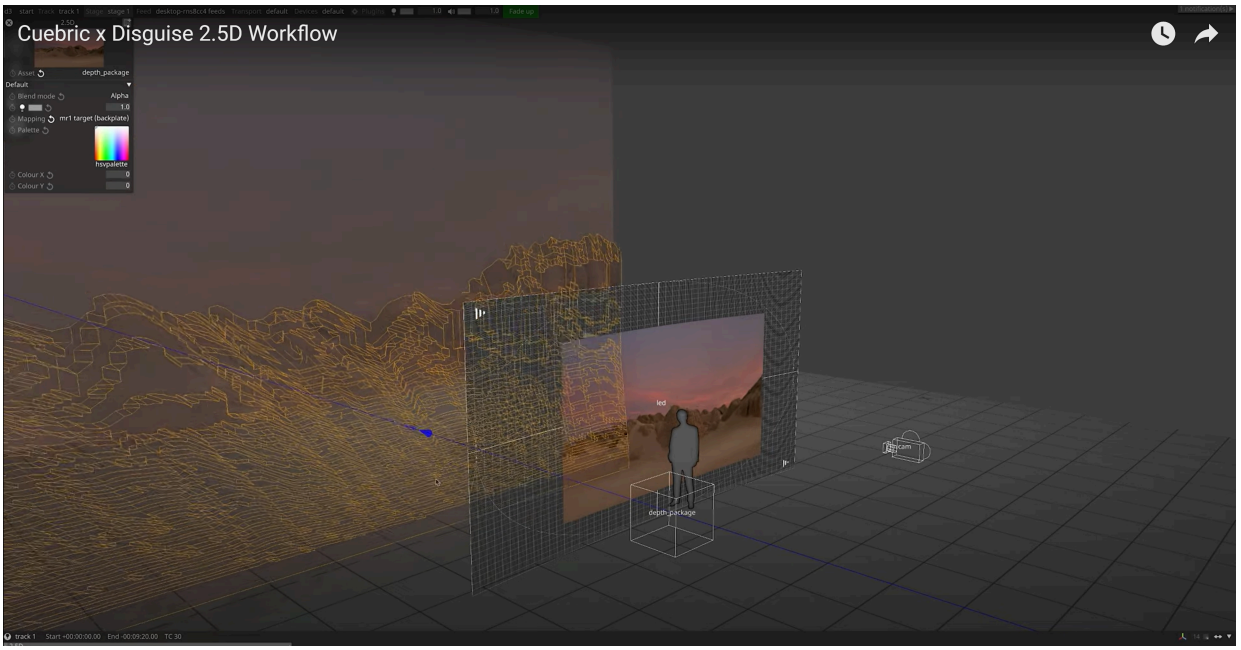
Cuebric's Plug & Play 3D solution allows you to turn your 2D assets into near-3D rapidly. Create, segment and dimensionise scenes in Cuebric, and then seamlessly export them with a single click to Disguise, where your scene's layers are automatically mapped to an LED environment, with the correct Z-depth, ready to shoot.

Cuebric includes a range of editing tools for creators to use in real-time to manifest their immediate idea or vision on a volume stage. Create, segment, and dimensionize 2.5D scenes without noise or distortion.

Cuebric's AI Depth Package enables creators to Export AI Depth Map and Export Depth Information, allowing users to rapidly layer these files as plates in Designer.

How can I use Cuebric with Disguise?





Watch online: https://www.youtube.com/watch?v=SKc3SoHT1Bg

Workflow

1. Sign up for Cuebric here.
2. Create a new project.
3. Create a prompt and generate the image.
4. Adjust the image using Cuebric's tools.
5. Use the segment tool to automatically or manually segment the image.
6. Export the scene has a depth package ready to import into Designer.
7. To set up a 2.5D asset from the Cuebric in Designer, follow the steps on this page.

Further Reading

- Sign up for Cuebric
- Cuebric website
- 2.5D Workflow

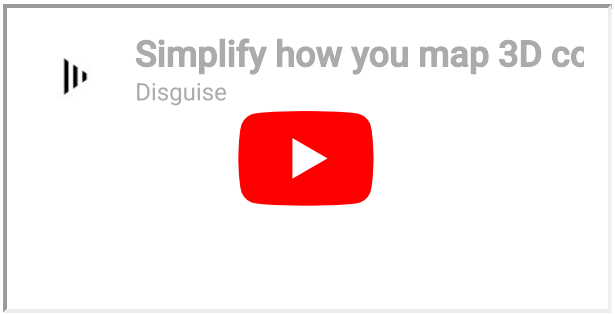
Mesh Mapping Workflow

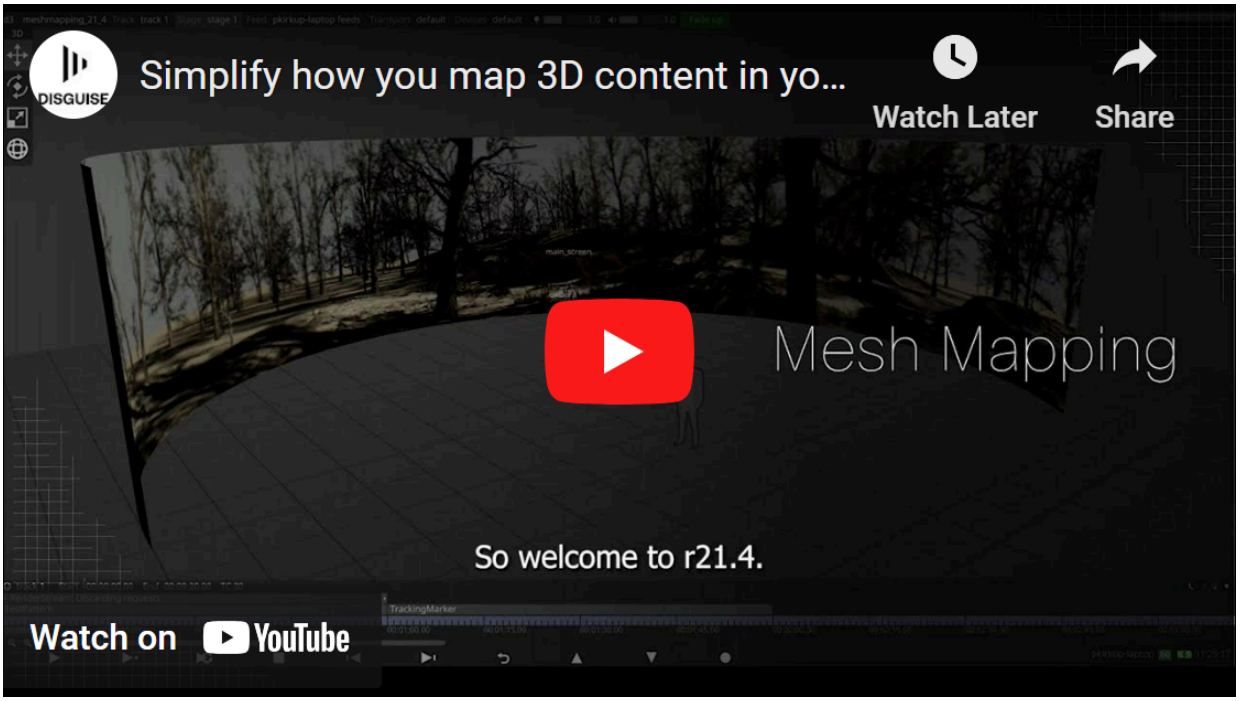
Mesh Mapping enables you to easily set up a screen projection that automatically adjusts to the extent of the surface you're targeting.

As of r21.4, experience more creative flexibility when mapping 3D content onto your volume.

This type of Mapping is aware of the surfaces that it's mapping toward even when dealing with UV mapped surfaces, screens, and things within your space.

See How it Works





Watch online: https://www.youtube.com/watch?v=owqUwk7WX4k

Workflow

Mesh mappings are in the same locations as other types of managers, from within a layer's Mapping field. For instance, in this video, the Mesh Mapping has been created from within the RenderStream Layer's Channel Mapping field.

Typical content mapped to surfaces via RenderStream is perspective projected and then cropped off by the geometry within our stage, which is inefficient. Mesh Mapping allows us to create a perspective render from an eye point that you define so that only the pixels that are relevant to that particular mesh are streamed across RenderStream.

When using a mesh mapping, the content being delivered from the render engine will be rendered from the perspective of the eyepoint that you determine in the mesh mapping before it's delivered to the target surface within Designer. This is very efficient from a bandwidth perspective as RenderStream only sends the exact pixels that the screen needs.

Increasing the pool size also allows the mesh to be subdivided into multiple perspectives that will line up to create a single contiguous scene.

Mesh mappings also support rendering to surfaces greater than 180 degrees. When the cluster assigner's channel mapping targets the Mesh mapping, Designer will determine that it needs more render instances based on available bandwidth and resolutions within the individual fragments, subdividing until it reaches the appropriate amount of fragments to match the screen resolution.

Resolution Scale is another feature available when using a mesh mapping. It allows you to reduce the scale of your content in order to decrease the number of fragments that your mesh mapping needs to provide full coverage around the volume.

Using Mesh Mapping in a Cluster Workload for an Inner & Outer Frustum Setup

1. Setup a single channel within your Cluster Workload that is targeting MR set (backplate)
2. Create another channel that targets a mesh mapping. For a pool size of 2, this will generate 2 streams for the mesh when RenderStream is started.
3. Be sure to use the same scene origin object in both so that they have the same reference point.
4. Start the RenderStream workload. You will see 3 total streams across the RenderStream network as it receives the renders for both in-frustum and out-frustum.

Best Practices for Eyepoint and Mesh Positioning

We recommend positioning the mesh and the eyepoint at the same coordinates to prevent generating a frustum with an infinite field of view.

If you need to reproduce a view at the world origin (0,0,0), you have two options:

- Slightly offset the mesh within the Disguise stage while keeping the eyepoint aligned, or
- Offset the eyepoint and use Scene Origin to re-align the resulting offset.

Both approaches maintain a valid frustum while preserving the intended viewpoint at the origin.

Shot Recorder Setup

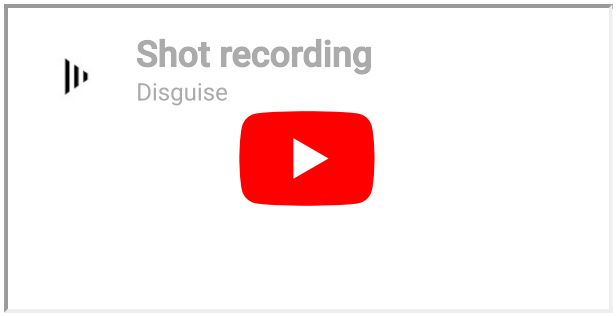
The Shot Recording system allows users to record and export parameters from within Designer, ready for use in post-production workflows.

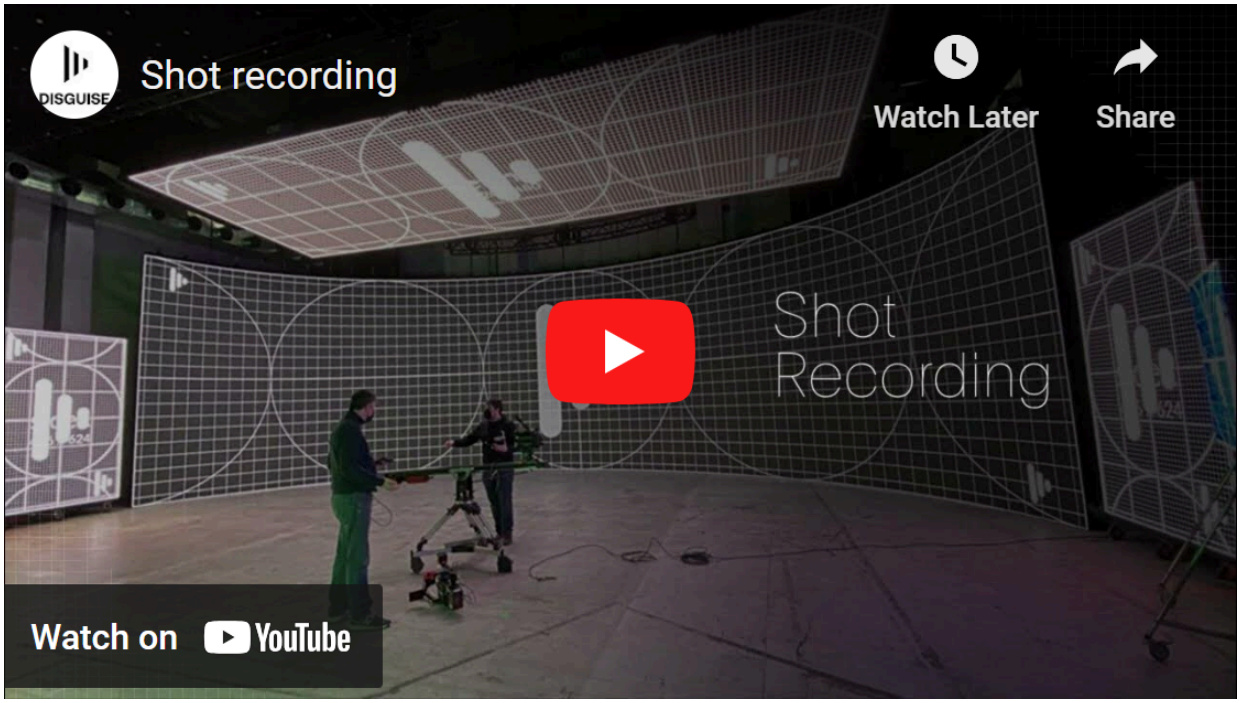
What is the Shot Recording System?

The Shot Recording system is designed as a generic system to allow users to record any fields or properties displayed in the GUI. Users can build up their own collections of parameters they want to record. In addition, the system has a quick and easy workflow for recording commonly-used sets of parameters, such as camera tracking.

Unlike device recordings, which are played back within Designer, the data recorded by the ShotRecorder Device can be used by external applications.

See How it Works





Watch online: https://www.youtube.com/watch?v=8us25d4VtBU

Components of the Shot Recorder

The main components of the ShotRecorder Device are:

- The **ParameterCollection** containing the parameters to be recorded.
- A **TimecodeTransport** used to assign timestamps to the recorded parameters.
- Recording and export settings.

Parameter Collection

ParameterCollection is a resource that allows users to create several different collections and switch between them when deciding which parameters to record. It contains an array of ParameterSources, each of which is one **parameter**, e.g. field/property to record.

ParameterCollections can also contain a list of nested ParameterCollections, allowing users to build up collections in a modular way and add them all to one outer collection which is used for recording.

Some ParameterCollections are auto-created with sets of parameters which are commonly required. For example, each Camera auto-creates a **tracking** collection containing all the parameters commonly required for recreating camera tracking data. Similarly, each **SpatialMap** also creates a similar tracking ParameterCollection - the main difference is that this accounts for offsets of the world zero point in the spatial map. It also references the currently active camera if using an MR set.

Parameter Sources

ParameterSources contain a list of field ParameterSources for compound types (e.g. Vec contains x, y, and z, Resource can contain its fields). For compound types, the value of the ParameterSource is found by populating the values of its fields. By contrast, the values for basic types are populated using the expression system, which can return floats or strings. Any limitations of the expression system, or particular fields which cannot be accessed by expressions, will also be limitations of the shot recording system.

Shot Recorder Workflows

Setting up the Shot Recording system

1. Create a **ShotRecorder Device** in the current Device Manager. This device will become the master of the shot recording system.
2. Name the **Slate** for the current shot.

3. Create a new ParameterCollection in the ***Parameters** field and add the desired parameters. ParameterCollections are a collection of different parameters that we are interested **in recording and getting the data out of i.e. camera tracking, MR set etc.
4. Add the desired Timecode provide to the '**Timecode**'** field.
5. The **Take Snapshot** and **Take Screenshot** options can be selected to take a project snapshot and/or screenshot at the end of each take.

Adding parameters to a ParameterCollection for recording

1. Open the **ParameterCollection** editor.
2. Alt+drag from the Sources **+** button to any field in the UI to add it as a parameter source
3. Several ParameterCollections are automatically created for convenience, such as camera tracking collections. These, or other user-created ParameterCollections, can be added to the ParameterCollections list in the editor to allow the parameters to be organised into smaller separate collections.
4. Click the arrow on the right hand side of a parameter to expand it and view its fields. Resource parameters do not have their fields added by default, but they can be added by clicking the plus button on the right hand side of the parameter.
5. Timecode can be set up inside the shot recorder in order to reference timecode in the recordings using **TimeCodeTransport**.

To make a recording

1. To start a recording, click the button displaying **Disengaged** to engage the recorder and start recording.
2. Click the same button again to stop the recording. The take number will automatically increment, or it can be changed manually.
3. Alternatively, use the API commands documented at `http://localhost/docs/v1/index.html` to start and stop recordings externally
4. Slates will be listed under the **Recordings** separator. Right-click on a slate to see the takes for that slate.

To export recordings

1. Select the desired formats under the **Export** tab.
2. Either click **Export all recordings**, or export recordings individually under the **Recordings** tab
3. Select **Auto-export** to automatically export in the selected formats when each take is finished.
4. Click **Export comparison table** to export a CSV table comparing the first value of each parameter for each recording
5. You can also enable to take a snapshot or a screenshot of each recording if desired.

Recordings can currently be exported in the following formats

- **JSON**: A compact JSON representation which contains the changed parameter values for each timestamp, similar to the .shot file
- **CSV (Compact)**: A compact CSV representation which contains only the changed parameter values for each timestamp
- **CSV (Dense)**: A dense CSV representation which contains all values for each parameter at each timestamp
- **FBX**: An FBX file containing meshes and pose information (position, rotation, scale etc.) of all recorded objects, where available. Any additional properties not represented within the FBX specification are recorded as custom properties.

A new sub-folder will be created in the project called `output`. Within there we have `shots` and in there will be the slate. In the slate, there will be the .shot recording in addition to the recording in the chosen format.

Recording timecode with a Shot Recorder device using Expression variables

The Recording Timecode with Expression variables and a Shot Recorder device workflow can be found here.

Limitations

- The shot recording system only writes data out of Designer. There is no capability to read recordings back into Disguise directly to replay the data. This is what device recordings should be used for.
- The shot recording system and the device recording systems do not interact with each other. They can still both be used at the same time.
- There is not currently an easy way to get all the recorded camera tracking data into Unreal Engine to re-render shots. Epic have outlined a workflow here to import and export FBX files which allows camera positions, rotations and FoV to be imported.

Depth Workflows

With Designer's depth tools, you can use black-and-white depth maps to deform meshes in real time, creating parallax and volume from flat plates or video content. The **depth video workflow** ties together **Depth Map Meshes**, **Depth Map Mappings**, and the **Depth Video Layer** into a single, repeatable setup detailed here.

What is the depth video workflow?

The depth video workflow uses a **Depth Map Mesh** as a deformable geometry, whose shape is driven by a depth map image or video. Depth can be applied in two main ways:

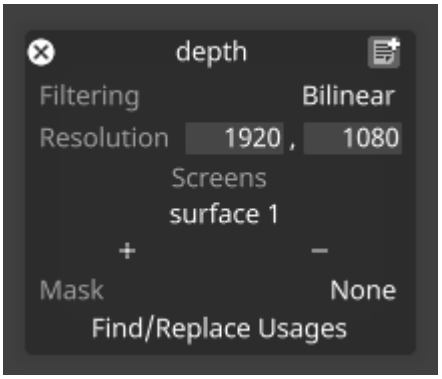
- **Depth Map Mapping** — the recommended, flexible approach that lets you drive a mesh from any layer that outputs a depth map.
- **Depth Video Layer** — a specialised content layer that sends a depth map video directly to a Depth Map Mesh, useful when mappings are not available (for example, inside a Layer Stack).

In both cases, the mesh occupies a frustum defined by **Field of view** and **Min/Max depth**, and the mesh vertices' z positions are updated dynamically based on the depth values in the content.

Depth video workflow

This section groups the how-to guides that make up a complete depth workflow:

- Set up a Depth Map Mesh
- Control the mesh using a Depth Map Mapping
- Refine depth and frustum settings



Example of a Depth Map Mesh driven by depth content

Workflow elements

1. Add **Depth Map Mesh** object to the stage.

2. Assign the **Depth Map Mapping**.
3. Create a **Depth Video Layer** on the timeline.
4. Assign **Depth map source content** such as an image or video clip.
5. Target the screen or object using the Depth Map Mesh.

Set up a Depth Map Mesh

A Depth Map Mesh is the core of the workflow: it's the geometry that will be deformed by the depth map.

1. In the **Stage** view, select the target stage object (for example, a screen or prop).
2. In the object's **Mesh** property, create a new **Depth Map Mesh**.
3. Confirm that the mesh appears in the stage with its default frustum (you can enable **Show frustum** on the mesh later if needed).

At this point, you have a mesh ready to receive depth information but it will still appear flat until you map depth content to it.

Control the mesh using a Depth Map Mapping

Use this path when you want **maximum flexibility** and the ability to reuse existing layers or complex setups.

1. Add or identify a layer on the timeline that outputs a **depth map** (for example, a Video layer playing a depth-map plate).
2. Place the depth-map media into the project structure (for example, under `objects/VideoFile`) and assign it to the layer.
3. Open the **Content Mapping** panel and create a new mapping of type **Depth Map Mapping**.
4. In the mapping, add the target screen (which uses the Depth Map Mesh) to the **Screens** list.
5. Scrub the timeline or play the track. The screen's Depth Map Mesh should now deform in real time based on the brightness values in the depth map (black/white representing relative depth).
6. Adjust any **Common Mapping Properties** (crop, scale, transforms) to align the depth map with the visible content on the screen.

This approach lets you keep your depth content in standard layers and reuse it across different meshes or screens simply by changing mappings.

Refine depth and frustum settings

Once the mesh is being driven by depth content, refine its properties to get clean parallax and minimise artefacts.

1. **Depth type**
   - Choose how the depth map is interpreted:
     - **Raw** – 0 is near plane, 1 is far plane.
     - **Relative inverse** – inverse normalised, where 0 is far plane and 1 is near plane (the default for many depth maps).
2. **Resolution**
   - Set the mesh **Resolution** to match (or broadly match) the resolution of the depth map texture. Higher resolution gives smoother deformation but uses more vertices.
3. **Field of view**
   - Adjust the mesh's **Field of view** so that the frustum encompasses the region of interest in your scene. This controls how "wide" the depth volume is relative to your stage.
4. **Min/Max depth**
   - Set **Min depth** (near plane) and **Max depth** (far plane) in metres.
   - These values convert the normalised depth in the texture into real-world distances.
5. **Depth offset**
   - Use **Depth offset** to shift the frustum's eye point relative to the mesh origin. This is useful to centre the mesh around the average depth of your scene.
6. **Blur / Dilate radius**
   - Increase the **Blur** and **Dilate radius** to soften edges and reduce artefacts around objects, especially where depth maps have noise or hard transitions.
7. **Show frustum**
   - Enable **Show frustum** to visualise the volume in the stage and confirm that the mesh's depth matches your camera and scene scale.

Advanced workflows

Once the basic depth video workflow is working, you can combine it with other Designer tools:

- **Composite with other content**
  - Map standard video plates or 2.5D assets onto the same screen or related geometry, using the Depth Map Mesh as a foreground or background element.
- **Use Layer Stacks**
  - Place a **Depth Video Layer** inside a Layer Stack for plates that already include effects (e.g. blur, colour adjustments) before or after depth deformation.
- **Animate or track the mesh**
  - Remember that the Depth Map Mesh is just another object; it can be parented, animated, or driven by tracking devices in more complex virtual production setups.

- **Match camera depth of field**
  - Combine the depth video workflow with camera **Focus** settings to add depth-of-field to your rendered scene, aligning the mesh frustum to the expected focus range of the camera.

Related information

- Depth Map Mesh
- Depth Video Layer
- Depth Map Mapping
- 2.5D Workflow

Anamorphic Lenses

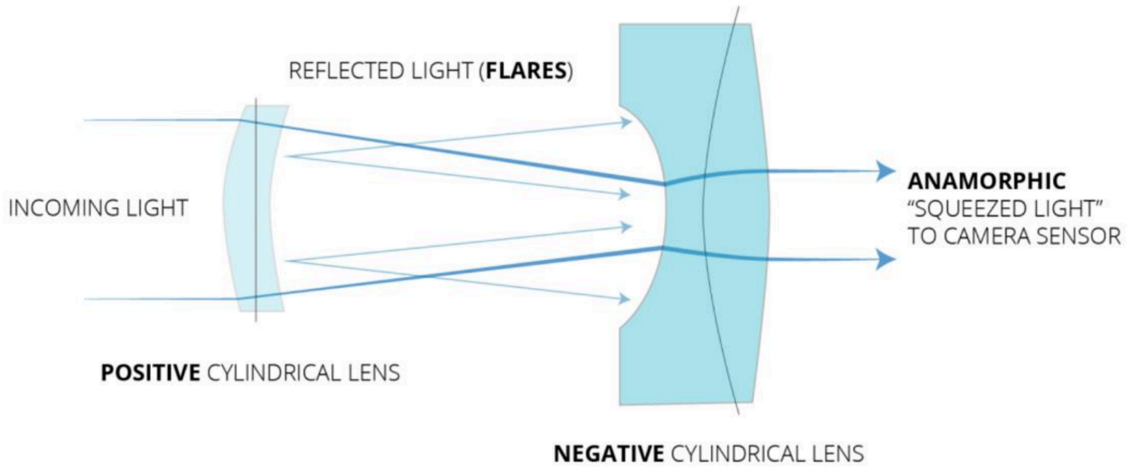
Anamorphic lenses are increasingly used in virtual production due to their unique optical characteristics, which bring both creative and technical benefits.

Anamorphic lenses compress (or squeeze) images to capture widescreen aspect ratios whilst using the same camera sensor size as spherical lenses.

Designer VP workflows fully support the calibration of these lenses.

How Anamorphic lenses work

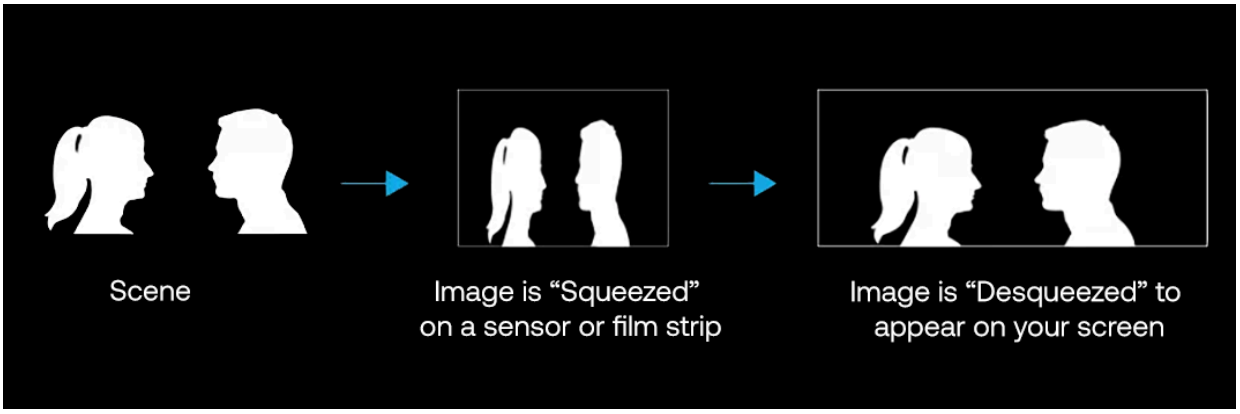
Anamorphic lenses have a unique optical design that captures images in a widescreen format by compressing the horizontal field of view. Unlike standard (spherical) lenses, which capture an image with equal proportions horizontally and vertically, anamorphic lenses compress the image only along the horizontal axis. This compression results in a “squeezed” image where objects appear thinner and taller than they are in reality.



What is De-Squeezing?

De-squeezing is the process of restoring the image to its intended aspect ratio by stretching the horizontal axis back to its correct proportions.

For example, a common anamorphic lens might have a 2x squeeze ratio, meaning it compresses the horizontal field of view by a factor of 2. So, a 2.39:1 aspect ratio scene (common in cinema) will be captured as a distorted 1.19:1 ratio image.



Video in crop feature

Frequently when data captured from non 16:9 lenses are output from camera sensors and transmitted using formats such as SDI, black bars are added to the edges of the frame to conform to the standards of the output format. To allow this data to be used in Designer in the intended aspect ratio, the video input feed can be cropped to remove these black bars. Performing this cropping on input rather than through feed mappings allows anamorphic lenses to be used for spatial calibrations like any other 16:9 spherical lens.

Further information

For more information on creating video inputs with different aspect ratios see the **setup-video-input** page.

ARRI Skypanels Setup

This guide explains how to configure ARRI Skypanels for control from the Disguise platform using Mode 24. The setup applies to Skypanel 30C, 60C, 120C, and 360C.

1. Selecting the Correct Skypanel Mode

To control any Skypanel model from Disguise

1. Set the fixture to Mode 24 – LE CCT & RGBW (8-bit). This mode provides full RGBW control.
2. Ensure Art-Net is enabled so the panel can receive DMX data from Disguise.

Mode 24 Channel Map (8-bit)

Mode 24: LE CCT & RGBW, 8 bit resolution per function

DMX Channel	Value Range	Percent	Function
1	0–255	0–100	**Dimmer** closed → open
2	0–255	0–100	**Color temperature CCT** 2,800 K → 10,000 K
3	0–10 11–20 21–119 120–145 146–244 245–255	0–4 5–8 9–47 48–57 58–96 97–100	**Green–Magenta Point** neutral / no effect full minus green −99% → −1% neutral / no effect 1% → 99% full plus green
4	0–255	0–100	**Crossfade to Color** White → RGBW color
5	0–255	0–100	**Red Intensity** 0% → 100%
6	0–255	0–100	**Green Intensity** 0% → 100%
7	0–255	0–100	**Blue Intensity** 0% → 100%

DMX Channel	Value Range	Percent	Function
8	0–255	0–100	**White Intensity** 0% → 100%

> **Note**
>
> To enable full RGBW colour mixing, Channel 1 (Dimmer) and Channel 4 (Crossfade) must be set to 255. This should be included in your driver commands.

2. Disguise Configuration

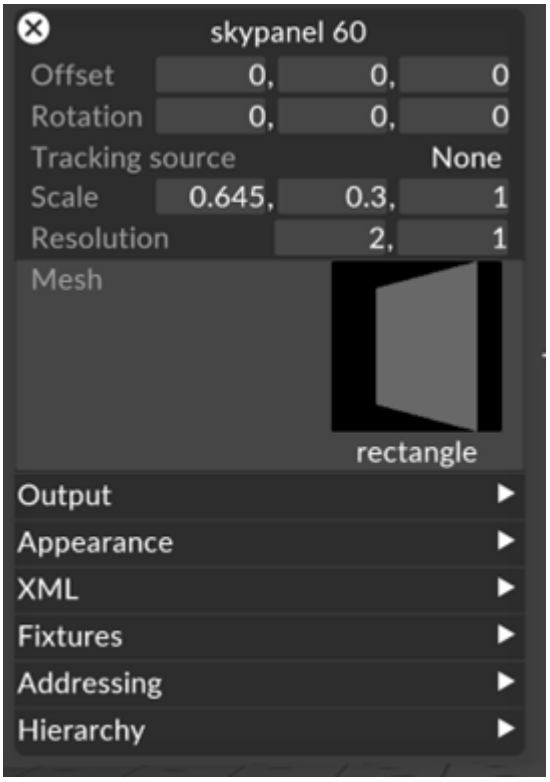
This example uses the **Skypanel 60C**, but the steps apply to all models.

Only the **fixture size** and **resolution** differ:

Model	Physical Size (approx.)	Resolution (engines)
30C	~0.35 × 0.3 m	1 × 1
60C	~0.645 × 0.3 m	2 × 1
120C	~1.29 × 0.3 m	4 × 1
360C	varies	4 × 3

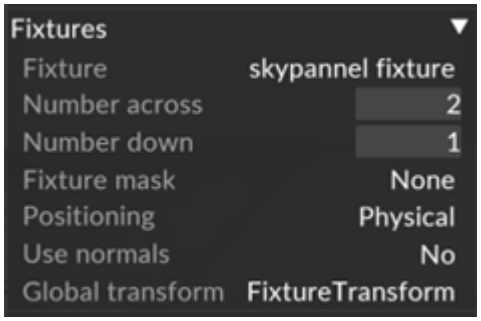
Create a new DMX Light

1. Go to **Stage** → **DMX Light**.
2. Click **+** to add a new DMX light.
3. Set the size of the **Skypanel 60C to 0.645 m × 0.3 m**.
4. Set the resolution according to your Skypanel model.

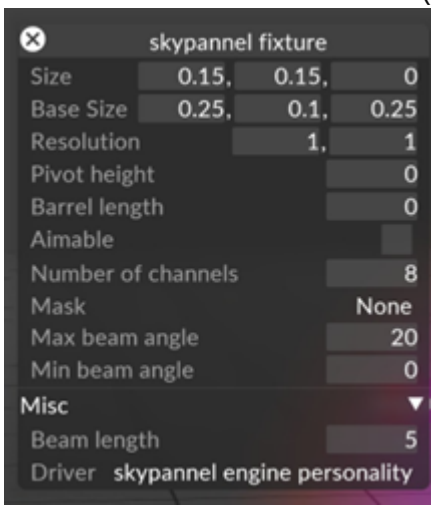


4. Fixture Configuration

1. Go to **Fixture** and create a new fixture type.



2. Set the number of channels to **8** (one RGBW engine).



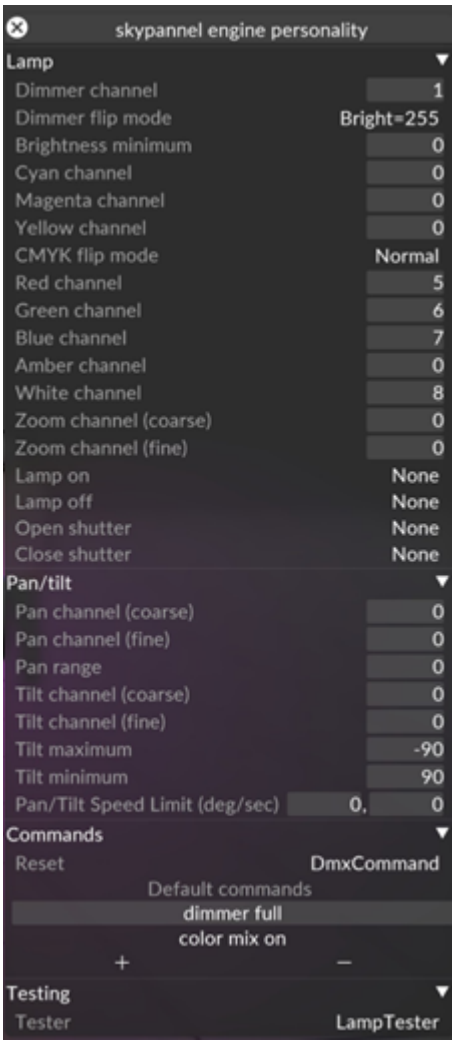
3. Set the fixture layout (number of engines):
    - 30C → **1 × 1**
    - 60C → **2 × 1**
    - 120C → **4 × 1**
    - 360C → **4 × 3**

5. Driver Setup (DMX Personality)

Create a new **driver** that maps each DMX channel:

- Channel 1 → **Dimmer**
- Channel 5 → **Red**
- Channel 6 → **Green**
- Channel 7 → **Blue**
- Channel 8 → **White**

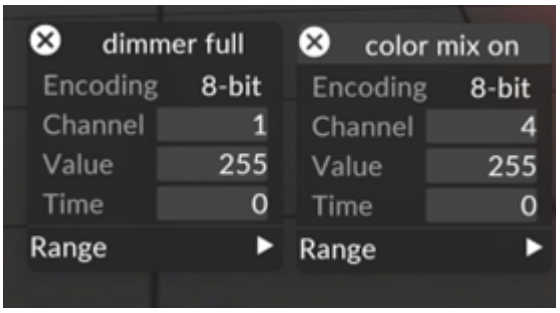
All other parameters (including pan/tilt) should be set to **0**.



Required Commands for RGBW Mixing

Create two commands in the driver:

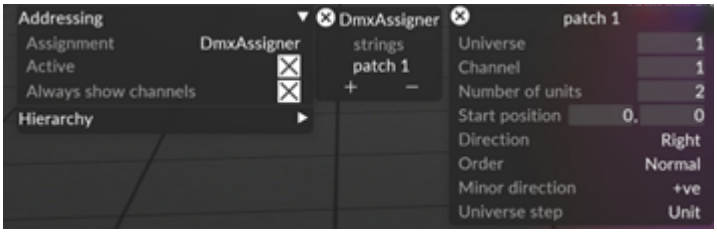
1. **Set Dimmer (Ch. 1) → 255**
2. **Set Crossfade to Color (Ch. 4) → 255**



6. Addressing

The addressing configuration allows Disguise to activate the DMX data and patch the fixture to a universe and channel.

1. Enable **Active** to output DMX data.
   If this is disabled, the fixture will receive **no DMX**.
2. Create a new **Assigner** and set:
   - Universe
   - Starting channel
   - Number of units (engines):
     - 30C → 1
     - 60C → 2
     - 120C → 4
     - 360C → 12



7. Outputting DMX data

To output DMX to the Skypanel, set up a DMX device in Disguise.
Follow the instructions in our user guide:

DMX device

Troubleshooting

Symptom	Likely Cause	Fix
Fixture stays white	Channel 4 not set to 255	Check driver commands
No output	“Active” not enabled	Enable in Addressing
Colour incorrect	Wrong Skypanel mode	Set Mode 24
Only part of fixture responds	Incorrect unit count	Verify resolution mapping

Green Screen Workflow

1. Set up the keyer for video input using one of the following two workflows below:
    1. Setting up keyers -
    2. Setting up keyers - Hardware keyer
2. Set up tracking for the camera (if not static) and perform spatial calibration. This can be one of three techniques listed below.
    1. Calibrate the tracking system outside Disguise.
    2. Or use an LED screen to perform the usual spatial calibration workflow.
    3. Or take manual observations to calibrate your camera as show at the bottom of this page.
3. Calibrate the camera's tracker delay.
4. Ensure the camera's video receive delay is set to **0**.
5. If you want to use set extension, model the green screens and add them to your project.
6. Map content to camera front and backplates as in the standard XR workflow.
7. To achieve camera switching, add your cameras to an MR set as normal.

Setting up keyers

a. Notch

1. In the video patch editor, add layer stack on the videoin.
2. Create a new notch layer in the layer stack.
3. Select chroma keyer notch block.
4. Ensure blend mode is set to Over-Alpha.
5. Calibrate notch keyer using the instructions at Chroma Key.

b. Hardware keyer

1. Connect the cameras feed into the input of the hardware keyers.
2. Connect the Key and fill feed from the keyers to d3 machines (Mustbe connected to 2 separate video inputs).
3. In the video in patch editor, create a video in or select the "new luma keyed input" opt
4. Map the chroma and luma key souces.
5. Select the luma key to Opaque white or black depending on the format coming from the hardware keyers.

Camera calibration using manual observation

1. Accurately model a real-world object within d3 to use for calibration. A flat-screen won't work very well as you need at least 6 points for calibration. A (cuboid) box is an easy way to do this.
2. Open up the spatial calibration editor from the camera.
3. Under **Manual Observations**, add your alignment object.
4. Under **Calibration**, set your calibration scope to **Registration only** if you already have calibrated lens values, or **Registration and focal length** if not. Manual observations are unlikely to contain enough points to accurately calibrate the lens distortion parameters.
5. Click **Add manual observation**. This will take a picture of the stage and capture the current tracking data. The new observation will appear under **Observations**.
6. Right-click on the new observation to open up the alignment editor, similar to Omnical's QuickAlign tool.
7. If you do not see the wireframe of your alignment object, increase the stage render opacity to see a render of the stage, and adjust the solved pose manually to get the object into view.
8. Left-click on the vertices of the alignment object wireframe to add alignment points, and drag these to the corresponding points in the captured image.
9. Once you have 6 or more points aligned, click **update calibration**. The wireframe of the alignment object should move to align much more closely with the real object.
10. If you are using a static camera, the calibration is completed.
11. If the camera is not static, similar to the XR workflow, keep moving the camera and taking and aligning observations until you are happy with the calibration over the whole movement area of the camera.
12. You can also take observations at different zoom/focus levels to calibrate the lens if required.

Using set extension

1. Create green screen objects in d3 from **Stage > MR Sets**. Model the screens as accurately as possible, although it's best to keep them slightly smaller than the real green screens to allow for small inaccuracies.
2. Ensure your camera is spatial calibrated.
3. Create a new MR set, add the green screens to the **on-stage** list, and add your camera via the override parameter or using an indirection controller.
4. Any backplate content outside your camera's view of the modeled green screens will be rendered as a set extension.

Green Screen Setup with Layer Stack

With the addition of the new Layer Stack feature in r20.0 it is now possible to do a quick Green Screen setup while applying a Notch Chroma Key asset to a VideoIn object and here is a quick

way of how to do that.

What you need for Basic Green Screen setup

- A Green Screen or a green cloth that is similar to a green screen that is positioned in front of the camera you will be using. In this basic setup we'll be using the webcam of our work laptops but any camera can be used with this workflow.
- The Notch block with the Notch keyer, which will need to be added to your d3 project's **NotchFile** folder.

How to setup Green Screen with Layer Stack

1. Launch new d3 project on any r20.0 build.
2. Add Video input to VideoIn object.
    1. Right-click on the 'Devices' tab at the top of screen
    2. Click on **Video Input Patch Editor**.
    3. Click on the black window under VideoIn_1 and then expand the **Input configuration for VideoIn_1** field to open its input configurations.
    4. Now click on **Input** and choose the camera feed of your choice. In case your don't have a real camera connected to d3, you can use your laptop's web camera.
    5. Look for names like **USB2.0 HD UVC WebCam**.
    6. Click on **Start Preview** under the VideoIn_1 to make sure the camera is working alright.
3. Create new Layer Stack.
    1. While in the input configuration field for VideoIn_1, click on **Layer Stack.**
    2. Insert the name for your new Layer Stack and click **OK**.
4. Add and configure the Notch layer in Layer Stack
    1. While in your Layer Stack, give a name to the layer you want to create and then from the list that comes up select a Notch layer.
    2. Then right-click on the newly created layer in your stack and press **Edit**.
    3. Now go to the **Effect** field and select the block with the Notch keyer.
    4. Open the **Default** tab and change the **Blend Mode** to **Over-Alpha**.
    5. Under the **Image 2D** tab, change the **Image Texture** to your VideoIn_1.
5. Add VideoIn and Content to your Surface
    1. To your timeline add a Video layer named **Camera** that is mapped to Surface 1 and as content add your VideoIn_1 to it.
6. Underneath the **Camera** layer add another Video layer, that's mapped to Surface 1 as well and to it add any content you like.
7. Observe how the content gets applied to the green screen correctly and that any real world objects in front of it appear as intended.

Using Sony's Moiré Alert with Designer

Sony's Camera and Display Plugin for Unreal Engine has a Moiré Alert feature that can warn a user if moiré is likely to be captured by a camera pointing at an LED display wall.

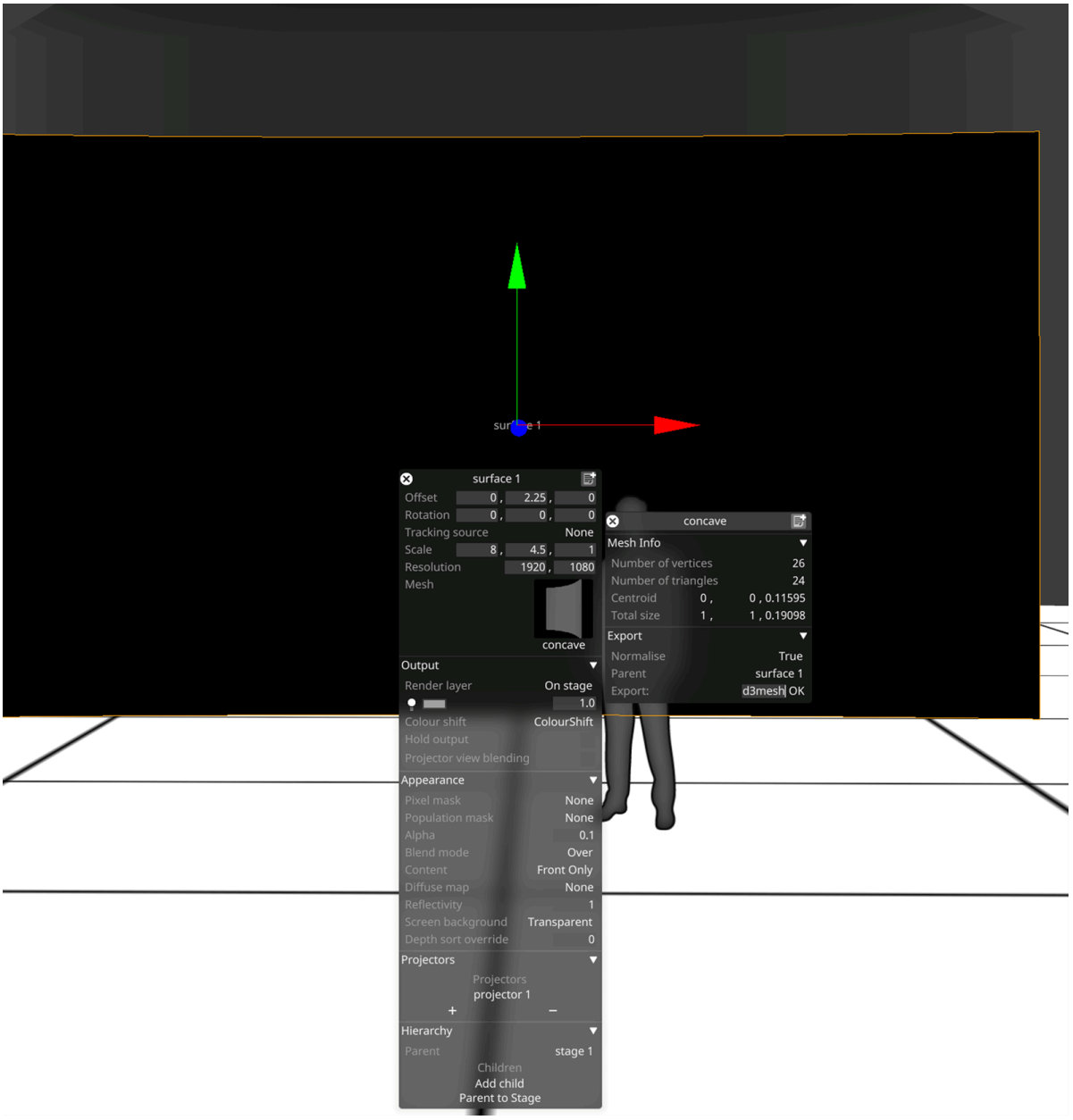
The **Sony Camera and Display** plugin can be requested and downloaded for free from the Sony website: Sony Virtual Production Toolset Downloads.

As the Sony plugin is for Unreal Engine (UE), not Disguise's Designer application, the Moiré Alert feature cannot currently be used directly inside Designer. However, by running the Unreal Editor alongside Disguise Designer the Moiré Alert can be used effectively for either On-Set or in Pre-Viz.

To do this, we need to export the mesh used as the LED wall layout from Designer and import it into Unreal Engine. This document describes the steps necessary to do this.

Export from Designer

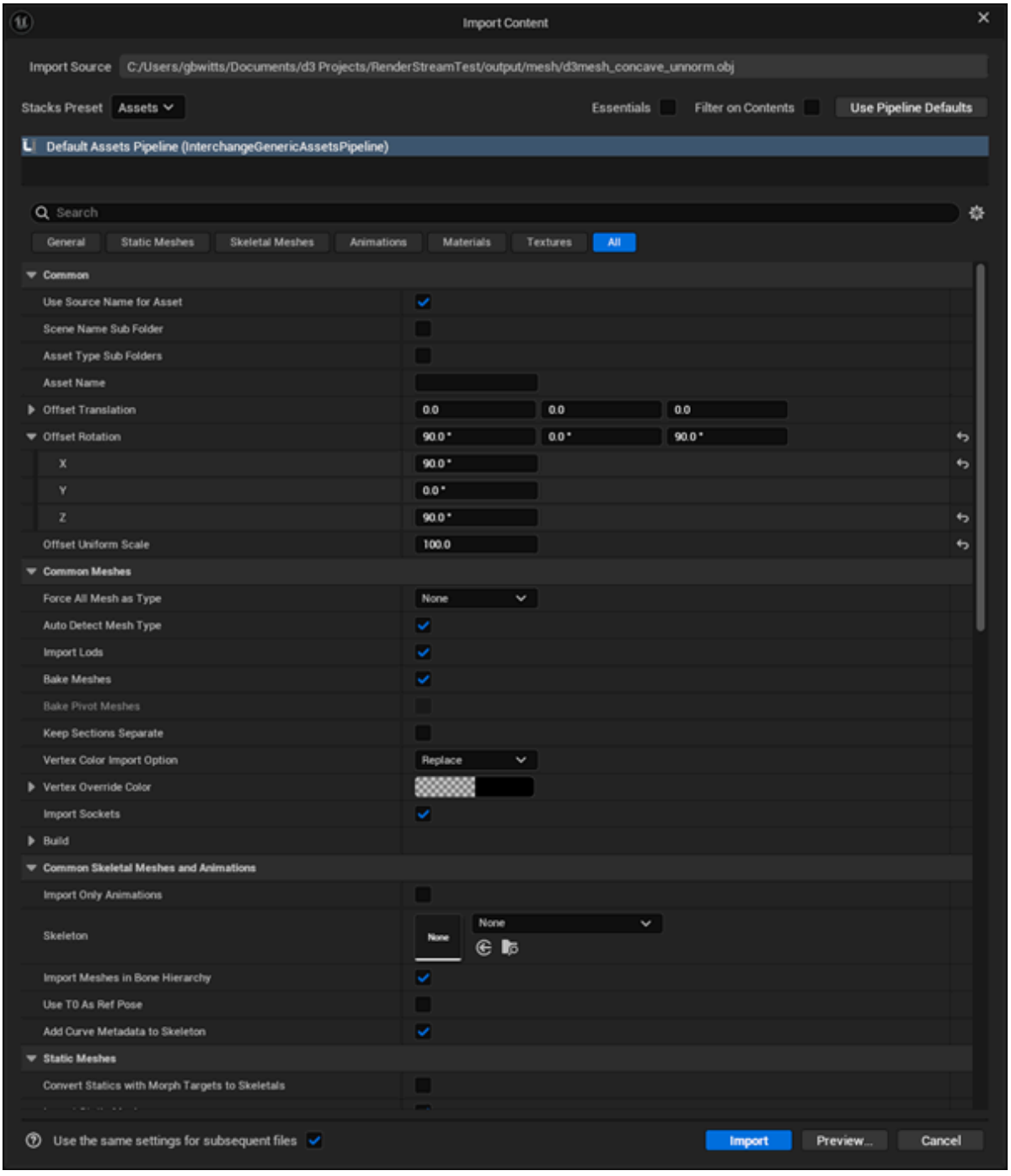
1. In Designer, right-click on the LED surface mesh, then right-click again on the Mesh thumbnail to show the menus as in the screenshot below.



2. Under **Export** select True for **Normalise** (this actually seems to do the reverse of the tooltip: setting to True maintains the current translation, rotation and scale).
3. Enter a filename in the **Export** line at the bottom, and press **Return**, or click **OK**.
4. This will save an .obj file (a text file containing the vertices etc) and a .mtl file for the material (note that changing the **Screen Background** and/or **Alpha** for the surface in Designer does not seem to affect the exported Material).

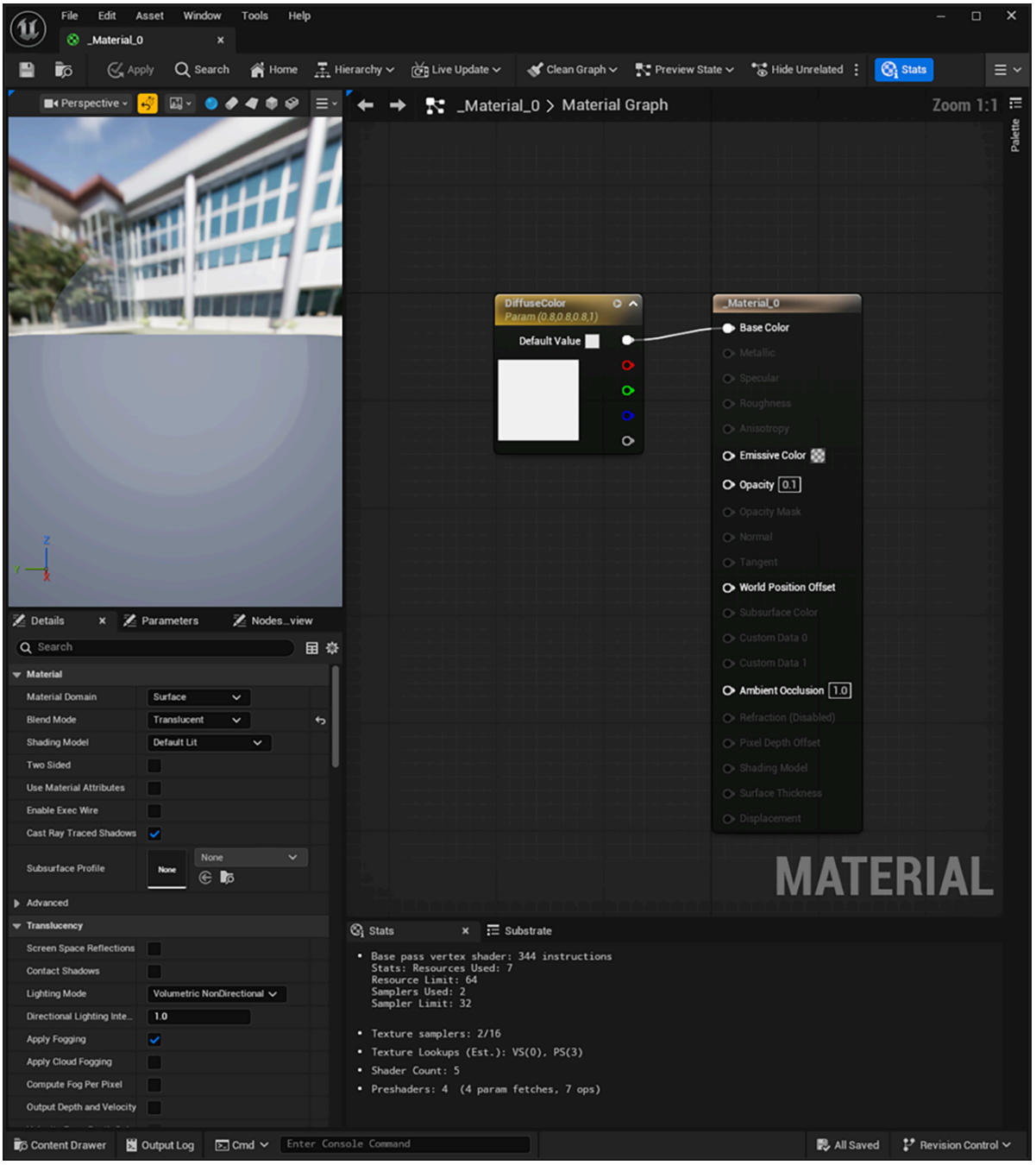
Import into Unreal Engine

1. Open the relevant project in Unreal Engine 5.7 – this could contain the 3D scene that will be rendered via Disguise's RenderStream, or it could be an empty level to reduce rendering requirements, if only the Moiré Alert is of interest.
2. Add a CineCameraActor to the level, with Sony's On-Set Camera Component attached.
3. In the Content Browser, right-click and select **Import to Current Folder**, or click on **Import**, and select the .obj file exported from Designer. The **Import Content** dialogue should appear, as shown below.



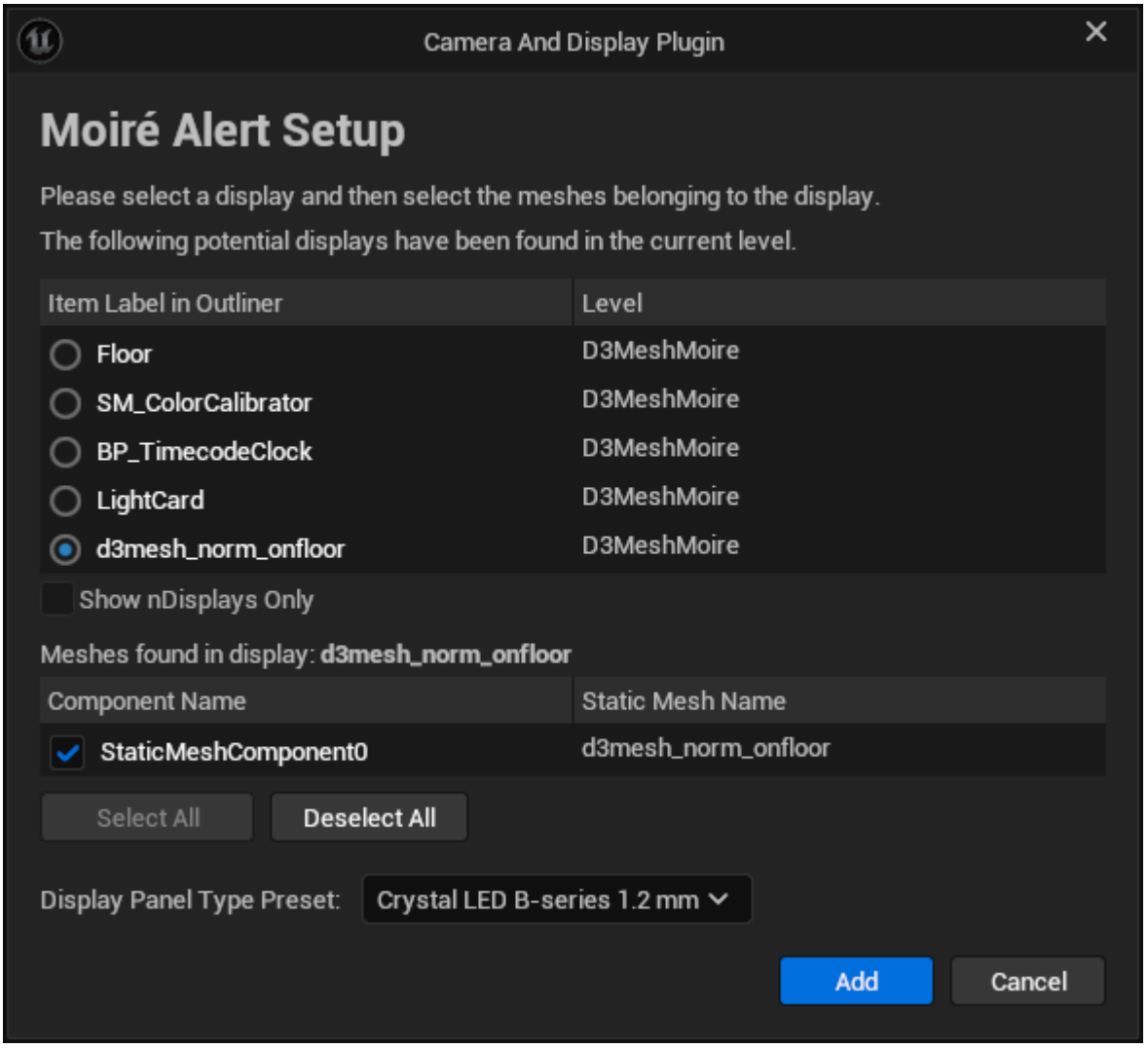
4. As Designer has units in metres, but Unreal Engine uses **cm**, we recommended to set an **Offset Uniform Scale** to 100.0 so that the scale is maintained.
5. Similarly, the XYZ axes are different between Designer and Unreal Engine, so it is recommended to set an Offset Rotation of (90, 0, 90) degrees.
6. All other Import Settings can be left as default.
7. Click the blue **Import** button at the bottom of the dialogue.
8. A Static Mesh and Material should now be visible in the Content Browser. Both need to be opened and saved.

9. If the LED screen mesh is being added to a level that has the original 3D scene in it, then it is probably preferable for this mesh to be mostly transparent when added to the level. To do this you will need to open the imported Material. Change the **Blend Mode** to **Translucent** and set the Opacity to something less than 1.0 (e.g. 0.1).

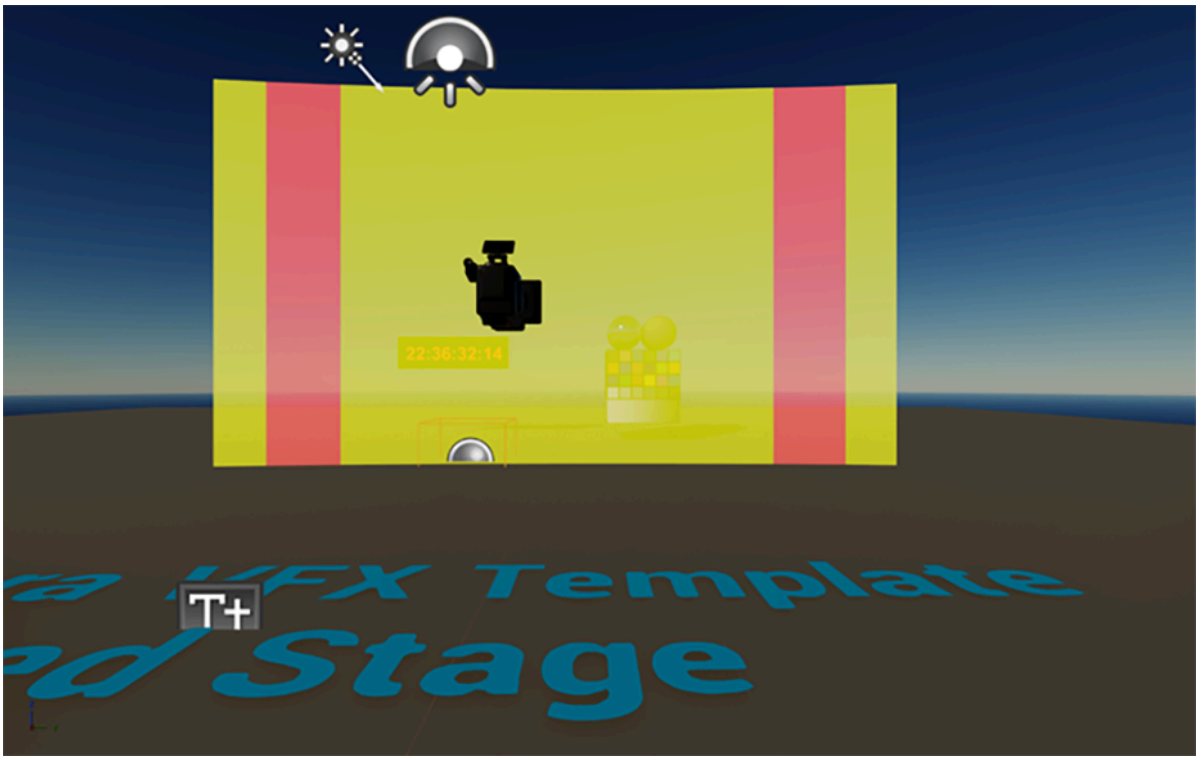


10. Drag and drop the mesh into the level, and place at the origin (location (0, 0, 0), rotation (0, 0, 0), scale (1, 1, 1)) so that its position matches that in Designer.

11. In the CineCameraActor's On-Set Camera Component, under Moiré Alert, click **Add Display** to setup the mesh up as the LED display for Moiré as shown in the screenshot below. Select the appropriate Sony LED display type.



12. Enable Moiré Alert in the On-Set Camera Component – you should see the Moiré colours appear in front of the mesh.



Camera Tracking

To ensure that the Moiré Alert is correct in a live situation, the CineCameraActor needs to be configured correctly (Camera Mode/Type, Imager Mode, etc), but also needs the correct camera position and rotation, and lens information (focal length, focus distance and aperture) from the real camera. Different Camera Tracking systems are set up in different ways, so are not detailed here, but it is recommended that the tracking data are sent to both Disguise and Unreal Engine at the same time as both systems need to know the values at the same time.

UV mapping documentation

This sub-chapter will teach you the specific principles necessary to UV map 3D models for Disguise.

Topics include:

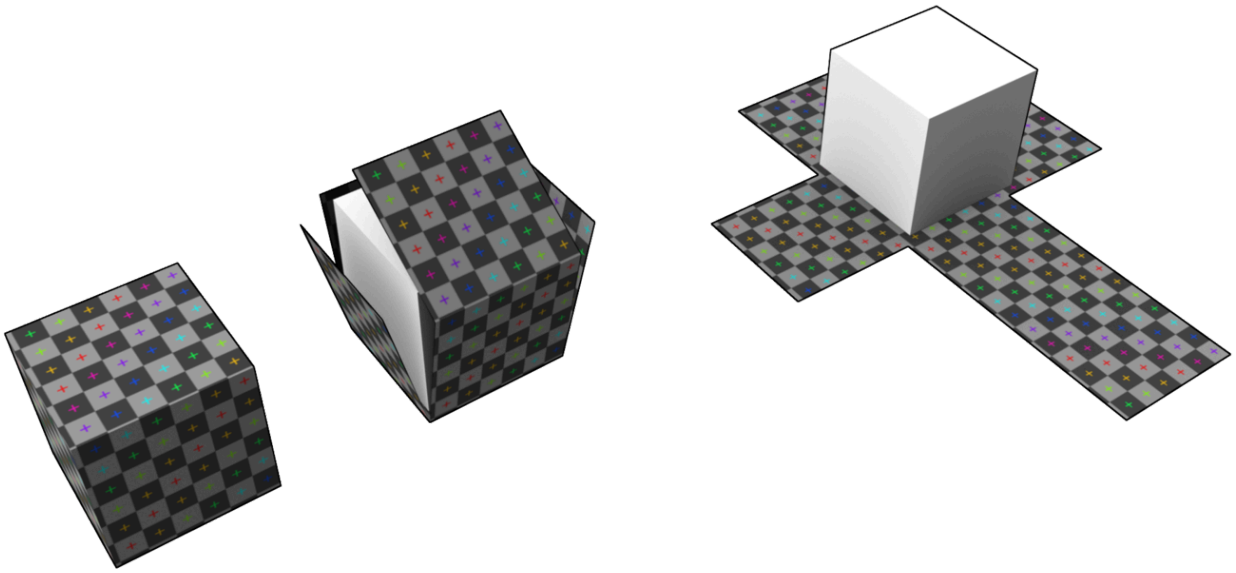
- What is UV mapping?
- How does Designer sample UV maps?
- The UV map as the hardware output
- The UV map exported as the content template
- Editing UV maps



What is UV mapping?

UV mapping is the process of generating a 2D representation of a 3D object.

This 2D representation is constructed from UV coordinates, which are commonly known as texture coordinates. U represents the horizontal axis and V represents the vertical axis. Each UV coordinate has a corresponding point in 3D space called a vertice. Together vertices form edges, edges form faces, faces form polygons, and polygons form surfaces. The image below shows how a cube in 3D space can be unwrapped into a texture in 2D space to generate a UV map.

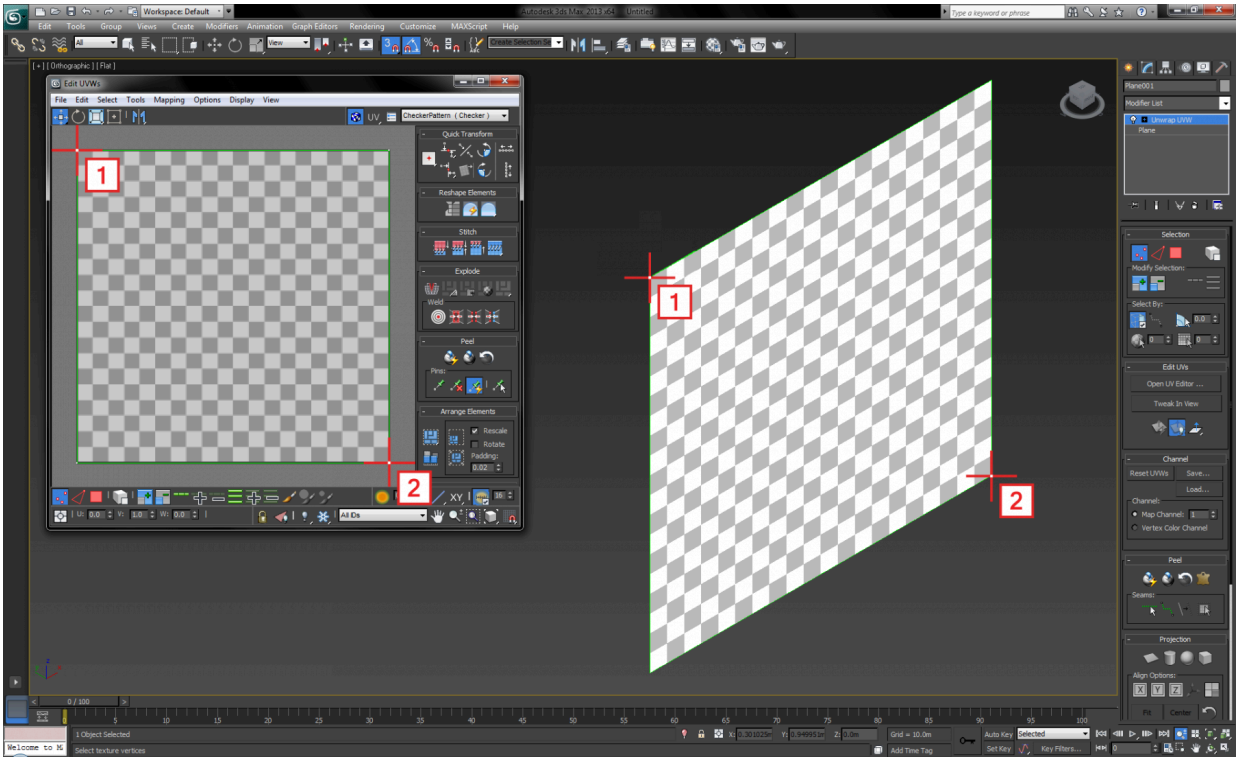


A 3D cube being unwrapped into a 2D representation to generate a UV map.

A UV map can be generated automatically by a 3D application, for example **3ds Max**, **Maya**, **Cinema 4D** and **Blender**, but normally the UV map requires editing manually. To read more about editing UV maps see the Editing UV maps

Example UV map

The image below shows a plane constructed from four vertices with four corresponding texture coordinates. The top-left vertice corresponds to the top-left texture coordinate and the bottom-right vertice corresponds to the bottom-right texture coordinate. For further information see the LED screen examples and the Projection screen examples.



A plane modelled and UV mapped in 3ds Max. The plane's top-left vertice corresponds to the top-left texture coordinate (1), and the bottom-right vertice corresponds to the bottom-right texture coordinate (2).

Helpful Resources

3ds Max

- Support and learning

Maya

- Support and learning

Cinema 4D

- Support and learning

Blender

- Support and learning

How does Disguise Designer sample UV maps?

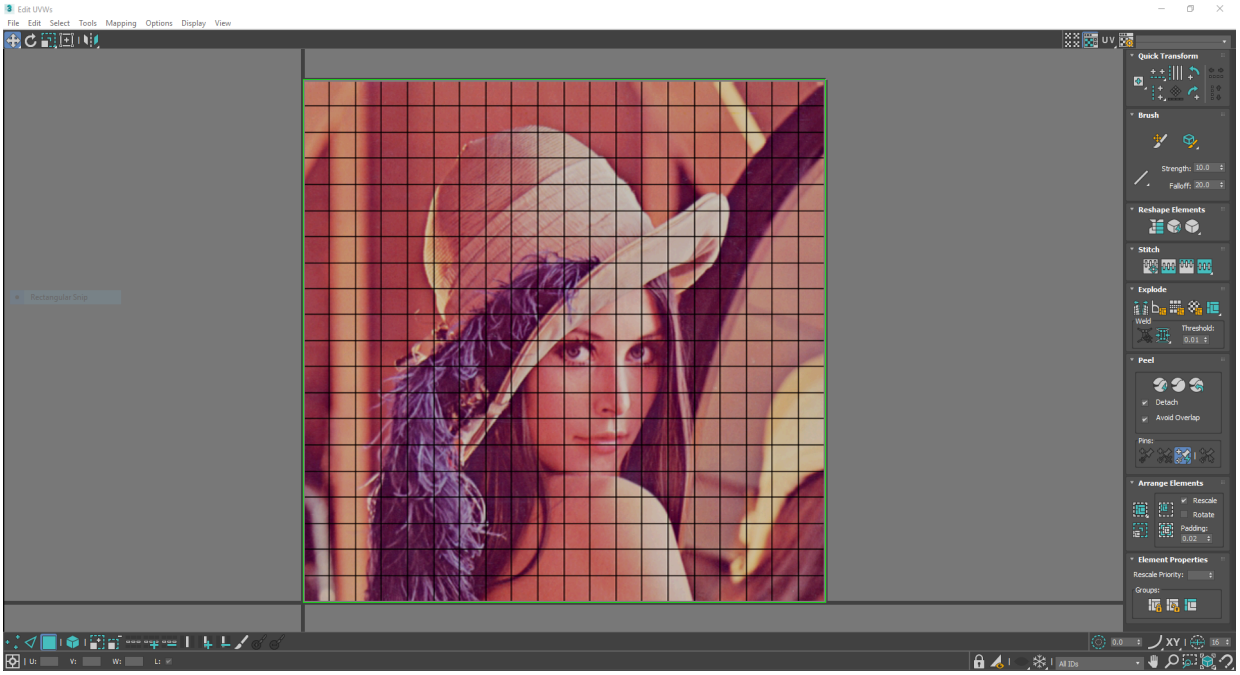
In Designer, all meshes used for displays such as LED screens or Projection surfaces require a UV map to function. This is important because if a display does not contain a UV map it will display black and cannot be used for any mapping or output.

A UV map tells Disguise how to translate back and forth between the 2D source content and the mesh's 3D polygons for visualisation and to enable Disguise's 3D content mapping & projection mapping capabilities.

A UV map does not specify the aspect ratio or resolution of the mesh. UVs are defined in what is called 'normalised coordinates', which means they must have values between 0 and 1 on both the U and V axes, and there should not be any gaps left at the edges of the UV 'box'.

This normalised property of UV maps enables you to select the appropriate resolution and aspect ratio based on artistic and technical requirements from within the d3 visualiser. The aspect ratio of the display is usually defined by the position and size of the 3D geometry of the mesh, while the resolution is determined based on the LED product or projector type and positioning within d3, depending on the display technology used. Aspect ratio is determined by the ratio between the horizontal and vertical dimensions of the display resolution.

For example, if Designer is started with a standard 'rectangle' mesh on a display, the mesh has 4 vertices in each corner with normalised UV coordinates at (0,0), (0,1), (1,0), and (1,1). The display is set to 1920x1080 resolution. The software will sample source content 1920 times across each of the 1080 lines stretching across the mesh polygons. If d3 wishes to know the 3D location of the second pixel on the second line, the UV coordinate (0.0005, 0.0009), or (1/1920, 1/1080) is found, and d3 calculates the corresponding 3D location.

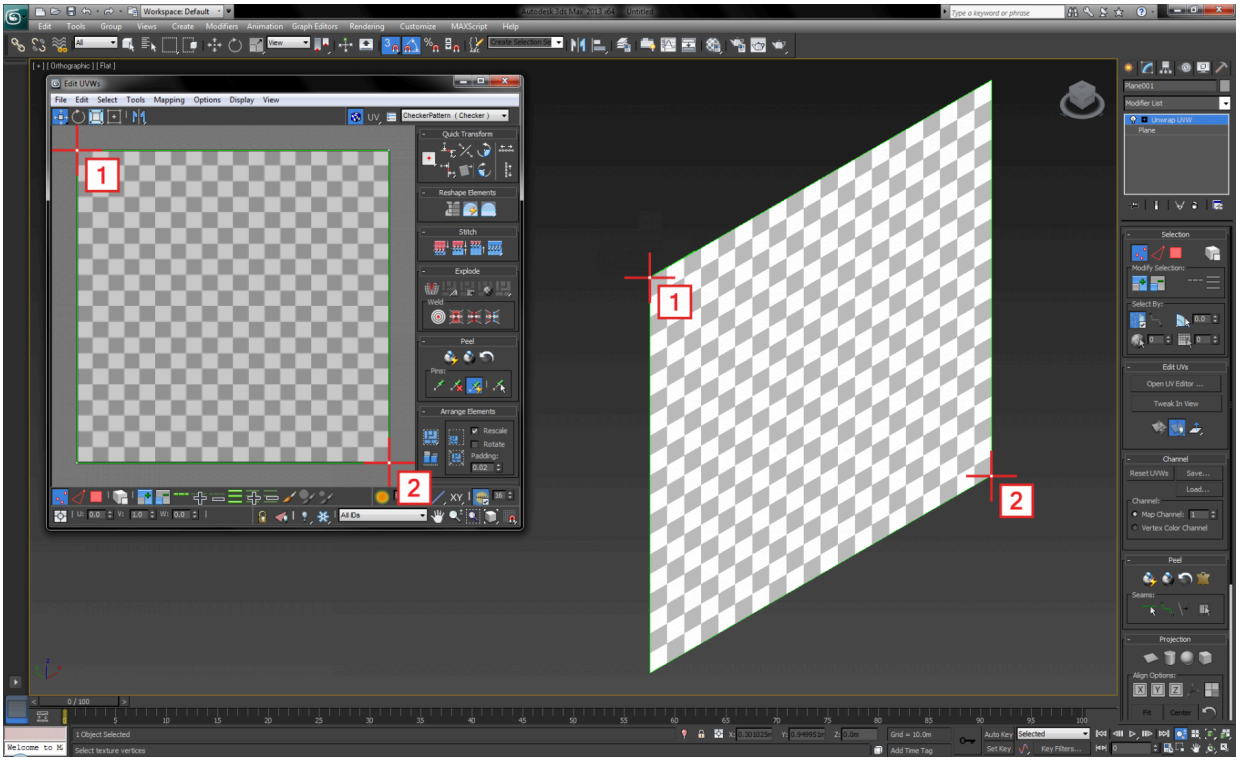


Screen in sampling the UV map shown in 3ds Max. The screen's top left pixel is sampling the UV map's top left quad.

Normalising UV maps

Normalising is the process of scaling the UV map to fill UV space entirely from a range of 0-1 in both the U and V axes. When Disguise samples the UV map it will capture the entire UV space. Therefore, it is important to normalise the UV map correctly while setting the resolution of the screen in Disguise correctly, because together these will determine how video content is mapped to the screen.

The normalised UV map is capable of generating uniform pixels across the 3D model if the resolution of the screen is set correctly in Designer. To do so, the resolution must be set to a non-square aspect, and this is possible because Disguise can read textures with different horizontal and vertical resolutions. However, a non-normalised UV map is capable of generating uniform pixels too, although in this case the resolution should be set to a square aspect in Disguise, for example 1024x1024 pixels. However, here the disadvantage is the UV map is sampling less UV space, resulting in more virtual pixels that will not be visible in Disguise. To read more about setting screen resolutions see the **Screen editor** page.



The UV map has been normalised allowing for all pixels set in Disguise's screen editor to be assigned on the screen.

Helpful Resources

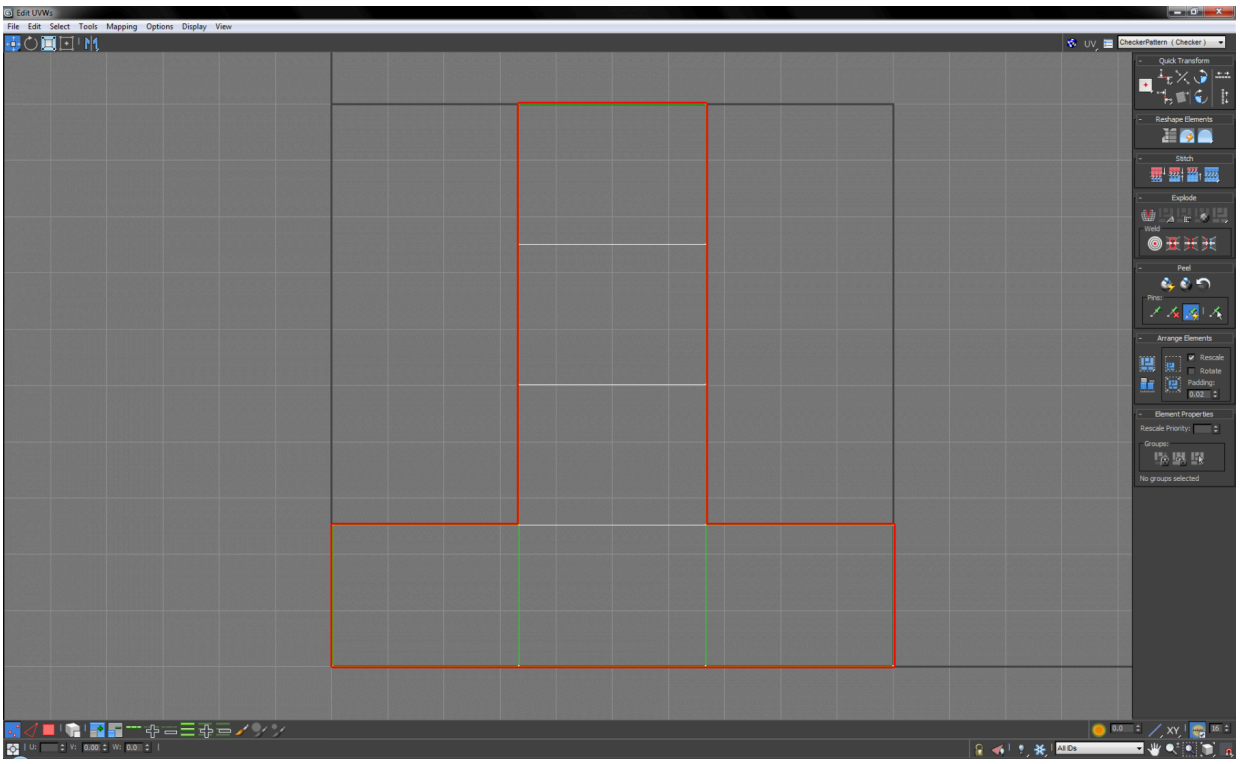
- Normalize UV's (Maya)
- PolyUnwrapper (3ds Max plugin)

Editing UV maps

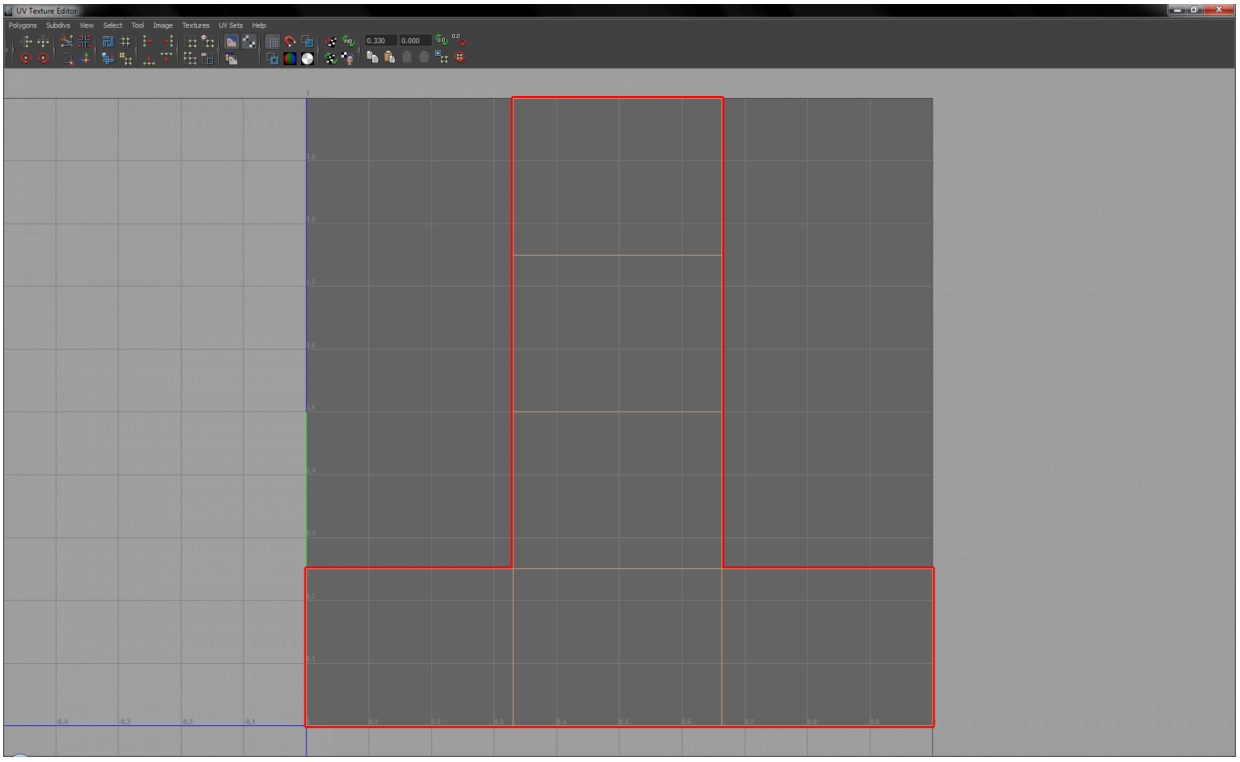
Usually after you've generated an appropriate UV map you may still need to edit the UV coordinates manually. Regardless if you have to do that or not you should always inspect the UV layout before exporting the object. 3d Max, Maya and Cinema4D offers powerful tools and plugins to do so.

The most commonly used operations for editing UV coordinates are:

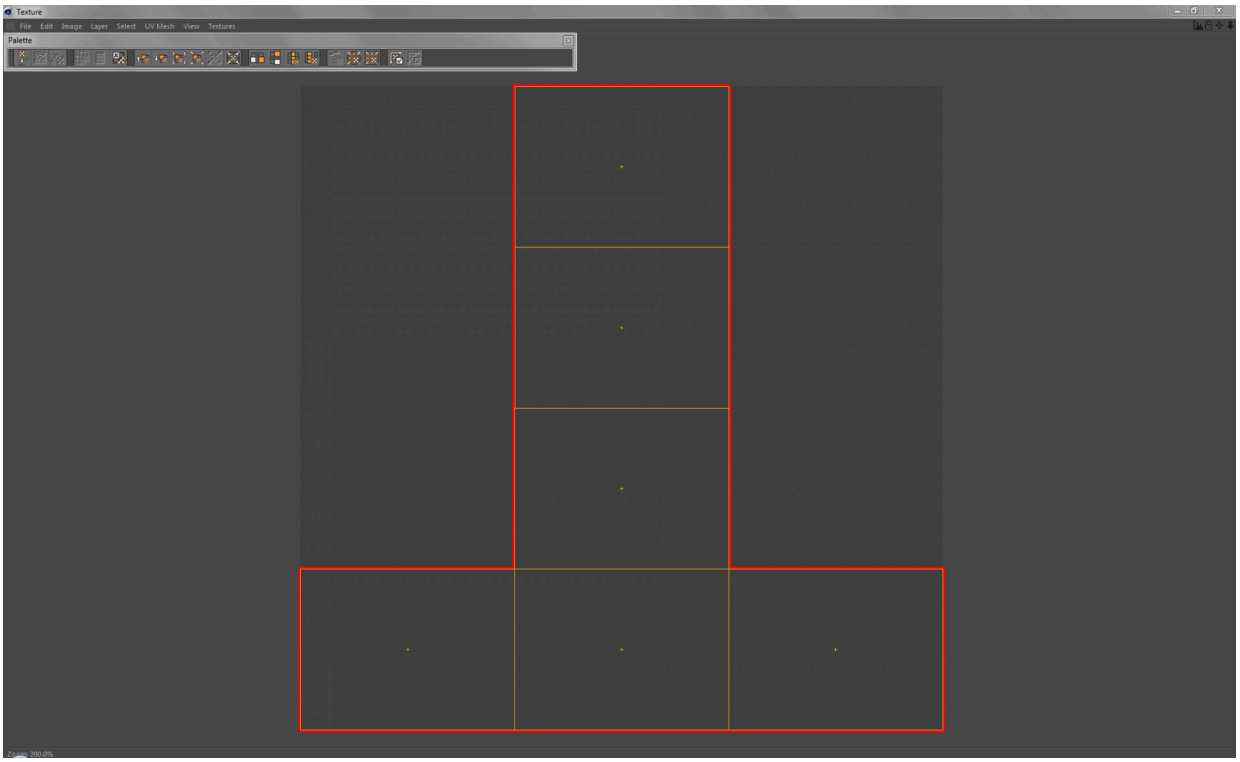
- Re-scale UV coordinates horizontally or vertically, or both.
- Move UV vertices and/or faces.
- Rotate UV coordinates.
- Mirror UV coordinates.
- Snap UV coordinates to each other.
- Relax UV coordinates.
- Normalize UV coordinates.



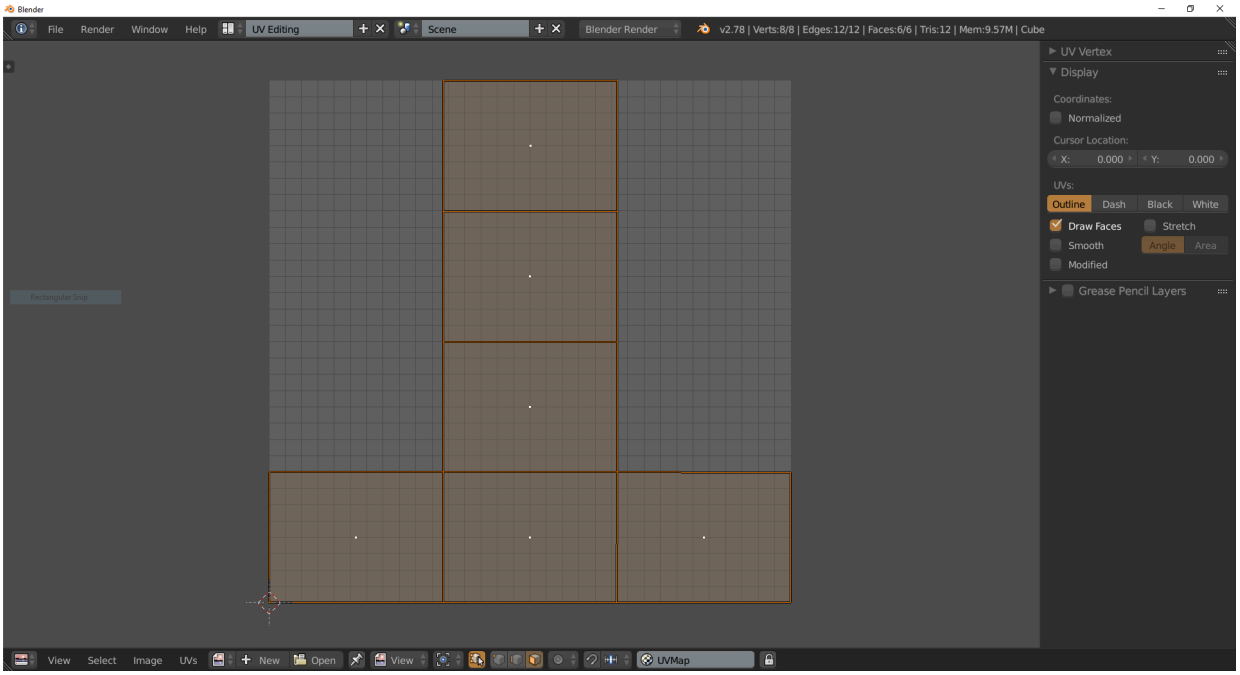
UV editor of 3ds Max



UV editor of Maya



UV editor of Cinema4D



UV editor of Blender

Helpful resources

3ds Max

- Unwrap UVW Modifier
- UVW Editor

Maya

- UV editor
- Viewing and evaluating UVs

The UV map as the Hardware Output

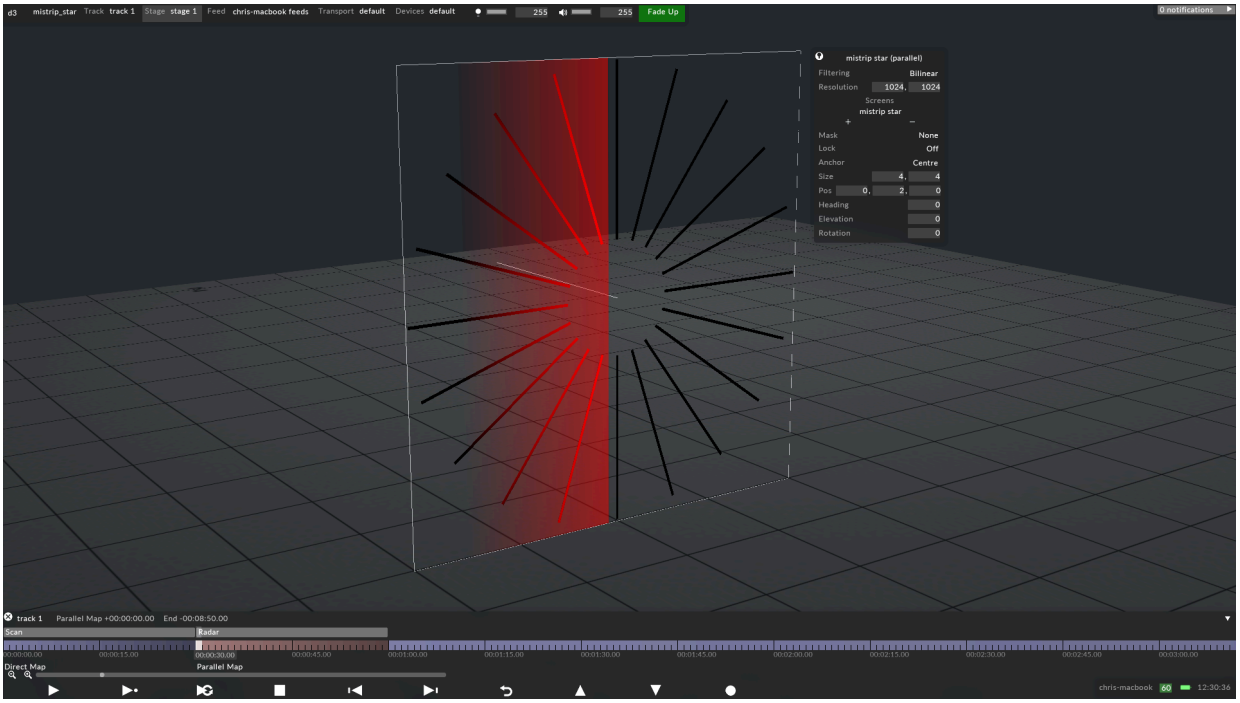
Direct mapping means that content is being applied directly onto the UV map of the screen's mesh, inside Disguise's visualiser. But the UV map itself, set to the assigned resolution by the screen editor also turns into the output that can be added to the displayhead and further sent to the LED processor. The output is captured by the processor pixel per pixel and sent to the physcial LED technology. As LED processors only can capture strict rectangular outputs it is therefore necessary to generate a precise grid-like UV map for LED screens.

Creative and Technical workflow separation

With Designer's advanced capabilities of mapping surfaces in various ways (using the different Mapping types), you are not bound not to make content only according to the layout of the UV map or resolution of the screens. Content can for example be virtually projected using a Parallel map, Perspective map, Cylindrical map, Radial map, or re-distributed using a Feed map. Only when using a Direct Map the content is applied directly onto the UV coordinates. When virtually projecting the content onto the pixels using a Parallel map the content instead gets "baked" it into the respective UV maps of all screens included in the map while the output always stays in the format of the UV maps.

With this unique system, you can use multiple content templates (i.e. mapping types) throughout your project while the outputs always stay in the same format defined by the UV map of the screens. As a result of this the content workflow and the technical output get separated; the content people can work from content templates that don't reflect the layout of the outputs.

In the example below a screen made of 20 MiStrips receives content from a Parallel map. The Parallel map virtually projects the content into the UV coordinates of the object, i.e the content gets baked into the UV coordinates as explained above. However, the output is still in the format of the UV map and is to be captured by a LED processor.

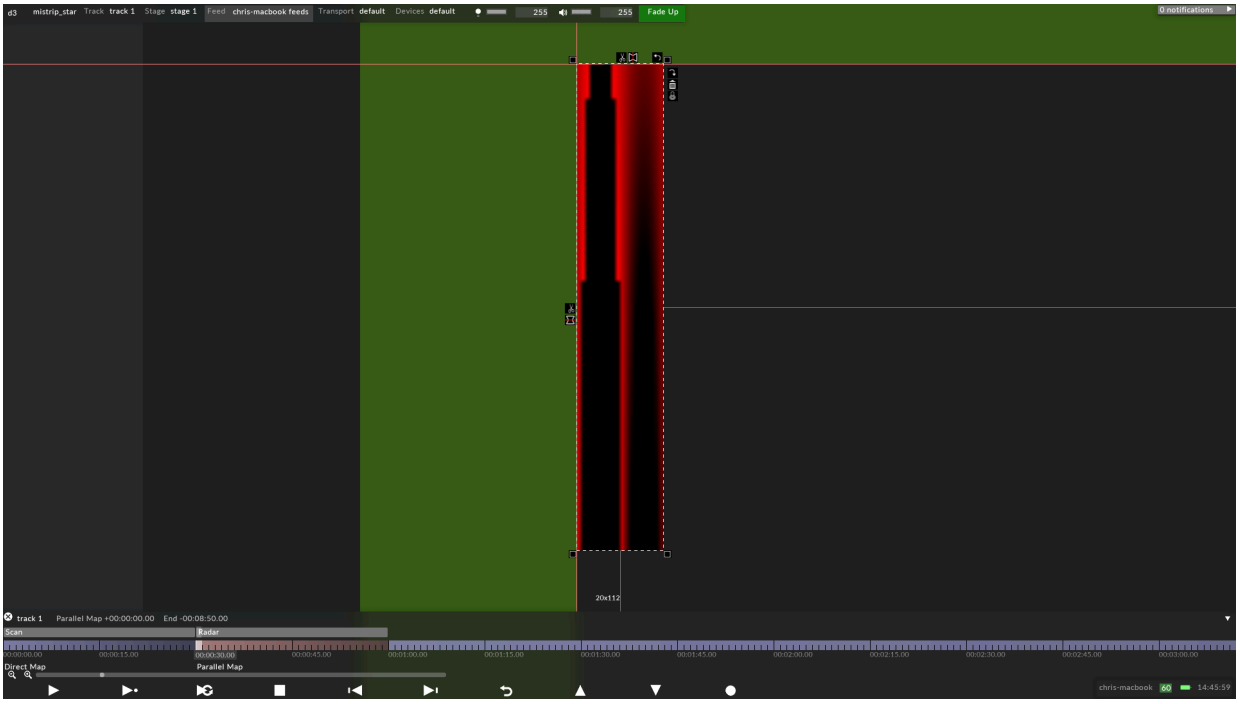


Content rendered out onto a display head for an LED processor to receive from Disguise.

Re-configure the outputs

If the output does not match the LED processor's specifications, it can be reconfigured using Disguise's **quick editing controls** including the ability to re-position, chop, mirror, and rotate the output.

This allows for even more freedom and separation between the creative and technical workflow.



Same example as above but the output has been reconfigured using Disguise's quick editing controls.

The UV map as the content template

As explained on the previous page, the UV map determines the output to the LED processor. However, preparing UV maps for projection surfaces requires a different workflow.

Normally the projection surface is unwrapped into a texture that the content creator can generate video content from, in a similar way to how artists texture 3D models in the video games industry. The UV map is rendered into a content template to be placed as a background in Adobe Photoshop, Illustrator, After Effects or any other image / video editing application. The template's resolution is calculated from two variables: aspect and density.

- The template's aspect ratio should match the UV map's to enable square pixels. This requires knowing precisely how the UV coordinates have been mapped to the 3D model. If the aspect is calculated incorrectly the rendered content may appear stretched across the projection surface.
- The template's pixel density should match, or preferably exceed, the projector's to enable a 1:1 pixel density. This requires knowing precisely how many pixels produced by the projector will hit the projection surface. If the density is calculated incorrectly the content may appear pixelated across the projection surface.

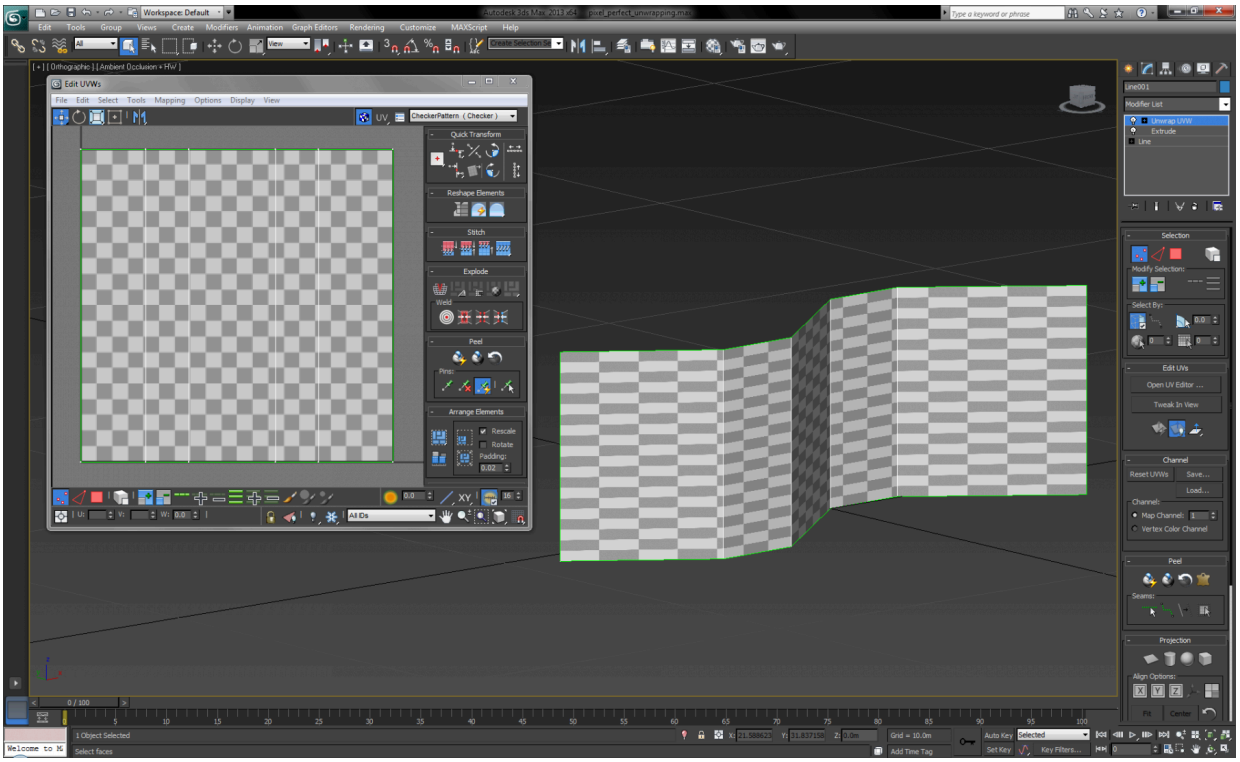
When video content has been rendered from the content template, it can then be **Direct** mapped to the screen in Designer. The content will map perfectly to the 3D model if the UV map for the template matches the UV map for the screen. In Designer, **virtual projectors** can be configured to sample content from the screen, and in realtime the sampled content is outputted to the physical projectors. Therefore, the projector outputs are independent of the content, which means the creative workflow for the content creator is separated from the technical workflow for the hardware technicians, as explained in The UV Map as the Hardware Output page.

Example

The example below demonstrates how the unwrapped UV map is rendered into a 2D image which can be used as a background template for 2D animations. When re-applied onto the screen in , the same template perfectly maps onto the 3D geometry. Virtual projectors are then sampling different parts of the screen and sends the output to the physical projectors. In this example, three HD projectors are covering the screen. This example below is identical to the example on the Pixel Perfect mapping page except the details how to create the UV map.

Step 1 - initial UV unwrap

The UV map has been unwrapped and normalised. See the Pixel Perfect Mapping page for more details how to create the UV map.

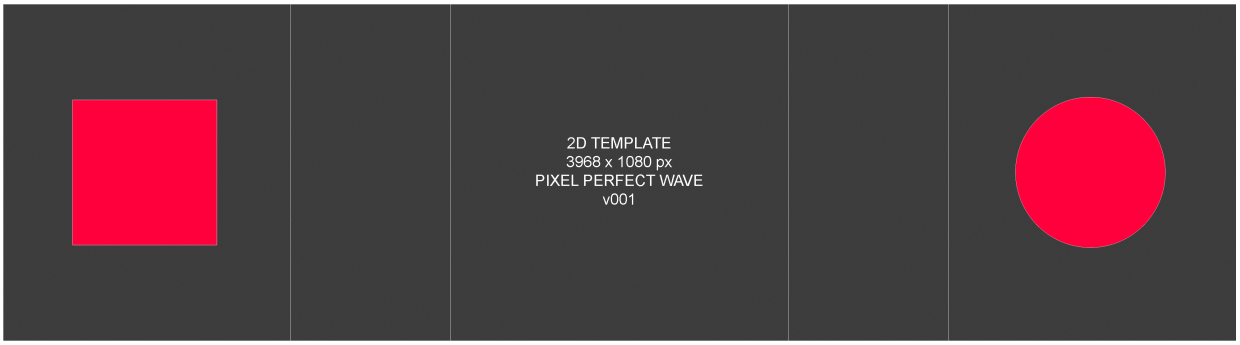


The UV map of the linear screen has been unwrapped and normalized.

Step 2 - Rendering UV maps to content templates

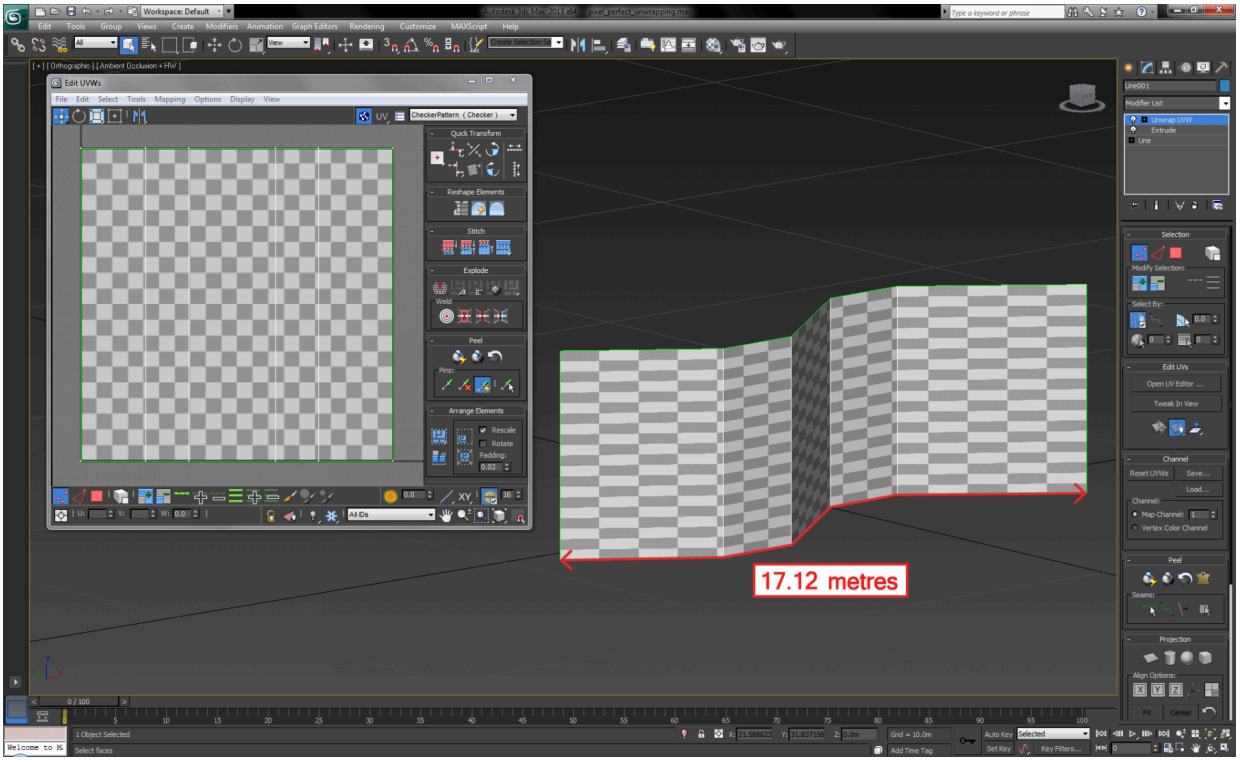
The image below shows the content template, which was rendered from the UV map. The template's resolution was calculated by:

- Knowing precisely how many pixels produced by the projector will hit the projection surface, across either the horizontal or vertical axis.
- Dividing this value by the projection surface's dimension across the chosen axis, to calculate a pixel density.
- Multiplying the pixel density by the remaining axis' dimension, to calculate its resolution.



UV map rendered to a content template.

It is important to measure the projection surface's dimensions based on the UV map. In this case, the total length of the extruded spline should be measured, because the extrusion automatically generated the normalised UV map.



By calculating both the template's horizontal and vertical resolutions from a single pixel density value, the content should enable square pixels. This is important because if the resolution is calculated incorrectly the content may appear stretched across the projection surface The template's resolution is 3698x1080 pixels. This was calculated by:

- Knowing approximately 1080 pixels will hit the projection surface vertically.
- 1080 pixels / 5 metres (the projection surface's height) = 216 pixels per metre (hitting the projection surface vertically).
- 216 pixels x 17.12 metres (the projection surface's length) = 3698 pixels (the horizontal resolution).

As a result, the aspect of the template's resolution should match that of the surface's dimensions. This can be checked by:

- Dividing the template's horizontal resolution by the vertical resolution.
- Dividing the surface's length by the physical height (remember to base the measurements on the UV map, explained earlier).

In this case:

- 3698 / 1080 = 3.424

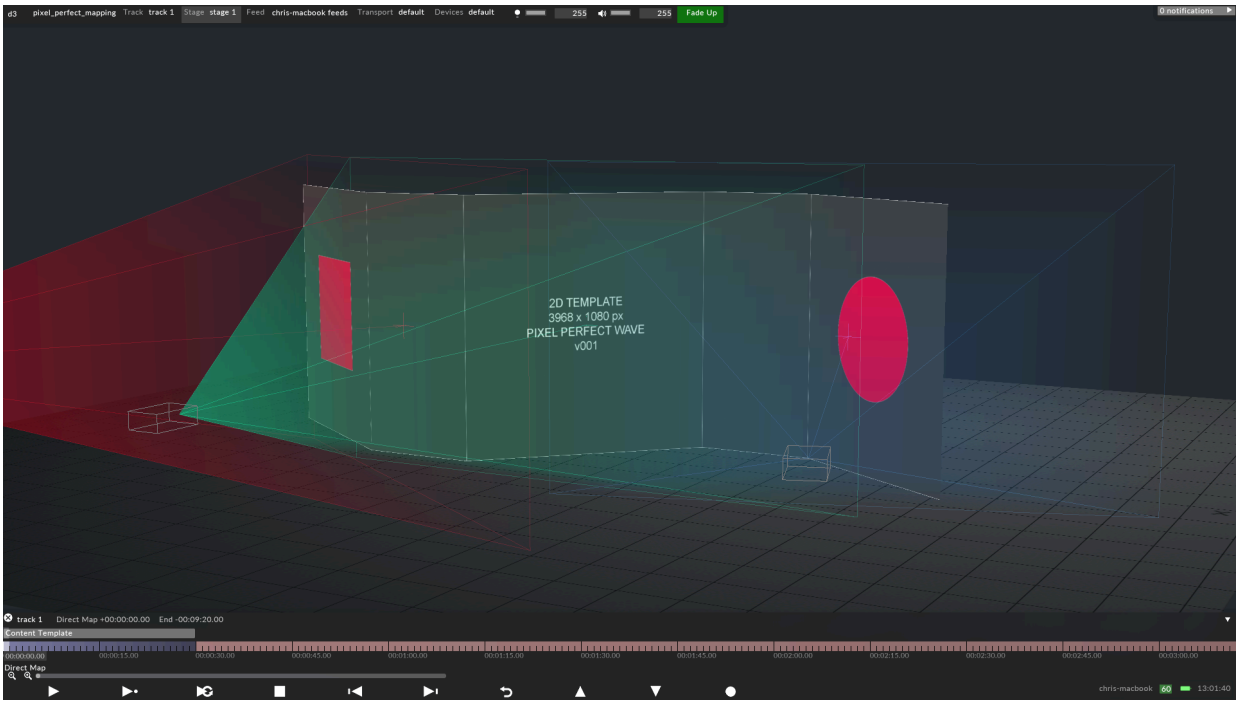
- 17.2 / 5 = 3.424

Both values are matching, which indicates the template's resolution is correct. To double-check this, a uniform square and circle should be drawn on the template in an image editing application, for example **Adobe Illustrator**, before the template is exported to . If the square and circle appear uniform across the screen in the template's resolution is correct.

Step 3 - Mapping content to screens

The image below shows the content template directly mapped to the screen in , which was exported as an .obj file from 3ds Max. In Disguise, three HD virtual projectors have been configured to sample content from the screen. Each virtual projector samples a different part of the screen, and in realtime the sampled content is outputted to the physical projectors by three unique feeds.

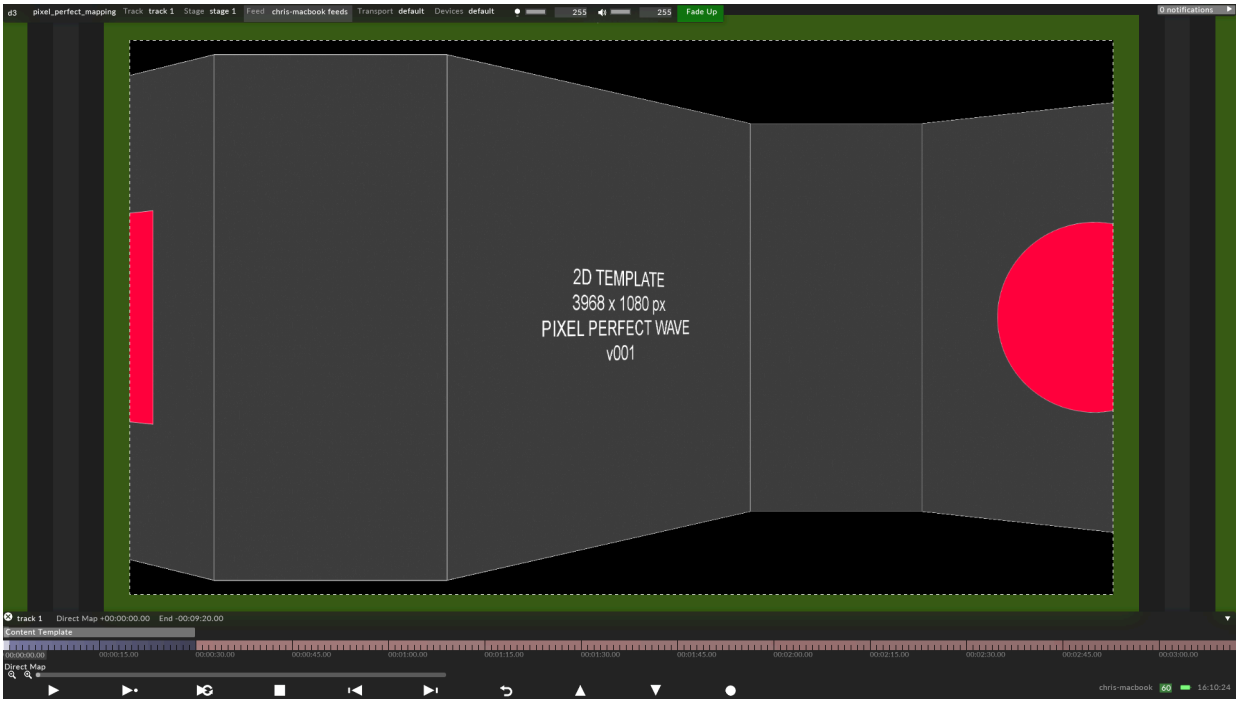
Remember to set the resolution of the screen in to match the content template by opening the **Screen editor**.



Template re-applied on the screen in Designer.

Step 4 - Outputting content to projectors

The image below shows the three unique feeds to be outputted to the physical projectors. If a virtual projector changes position its corresponding output will automatically update in realtime to show the virtual projector's updated point of view. Therefore, if a virtual projector's position changes onsite, content does not require rerendering, because the outputs are independent of the content. To read more about Disguise's projector simulation toolkit see the **Projector simulation** chapter, which explains how to **quick calibrate**, **warp**, and **blend** projectors.



The automatically generated outputs from the three virtual projectors.

Helpful Resources

- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Flattening UVs in 3ds Max
- Unfolding UVs in 3ds Max
- Rendering content templates in 3ds Max

LED Screen Examples Overview

This sub-chapter will teach you **how** to UV map **LED** screens for Designer, which are controlled either by **DVI** or **DMX** signals. The following examples revisit the UV mapping principles covered in the previous sub-chapter. Therefore, it is **recommended** to read this first.
Topics include:

Example 1: MiSTRIP star

Example 2: OLite column

Example 3: DNA spiral

Example 4: Millenia Tower



Example 1 - MiSTRIP star

This case study shows how to prepare a star shaped LED screen constructed from Barco MiSTRIPS for Designer. The first step is to UV map the 3D model by **unwrapping** the mesh into a **grid**. As explained on The UV Map as the Hardware Output page, **the UV map should comprise a grid because this is the output to the LED processor from the software, and LED processor's can only capture outputs composed from rectangles**.

How the UV map is generated

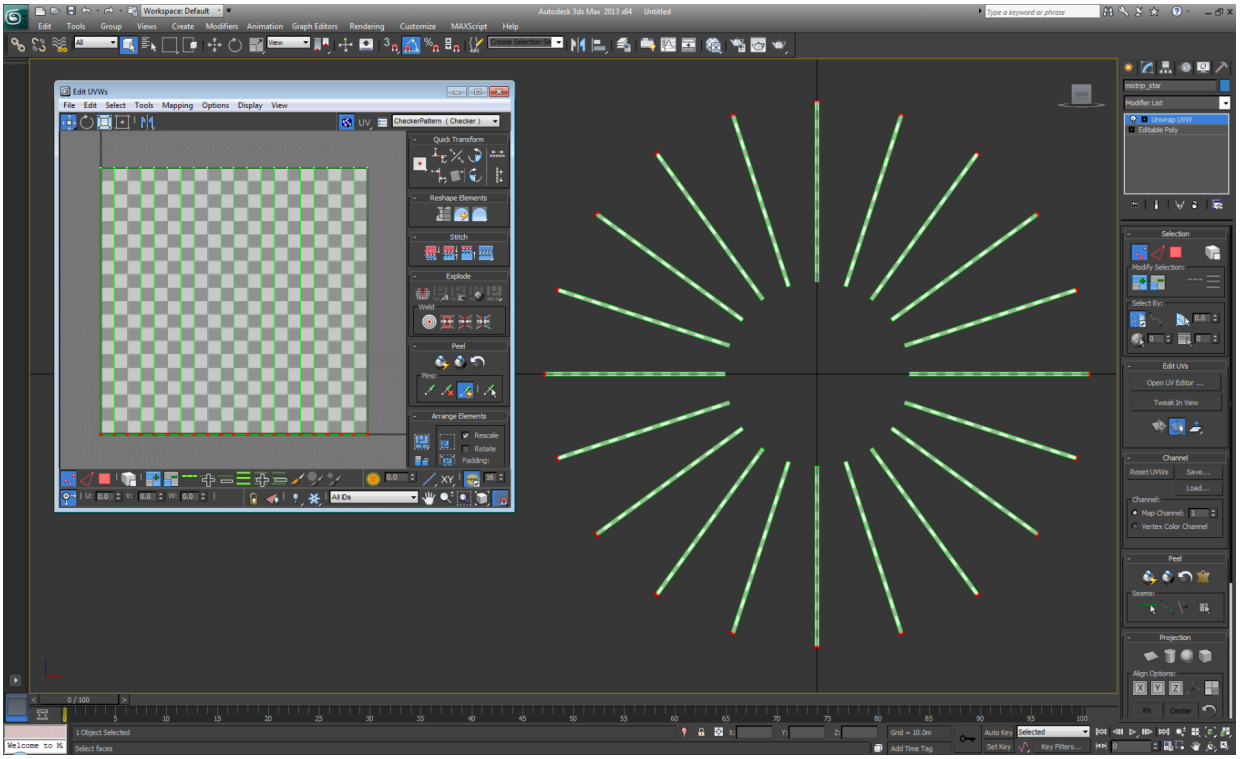
A plane can be **arrayed around a local pivot point** to recreate the surface with the UV map **automatically generated**. However, the UV map will have **overlapping** UV shells and therefore will require editing, either **manually** using UV editing tools or **automatically** using a script, depending on the level of complexity.

Step 1 - Generating the UV map (3ds Max)

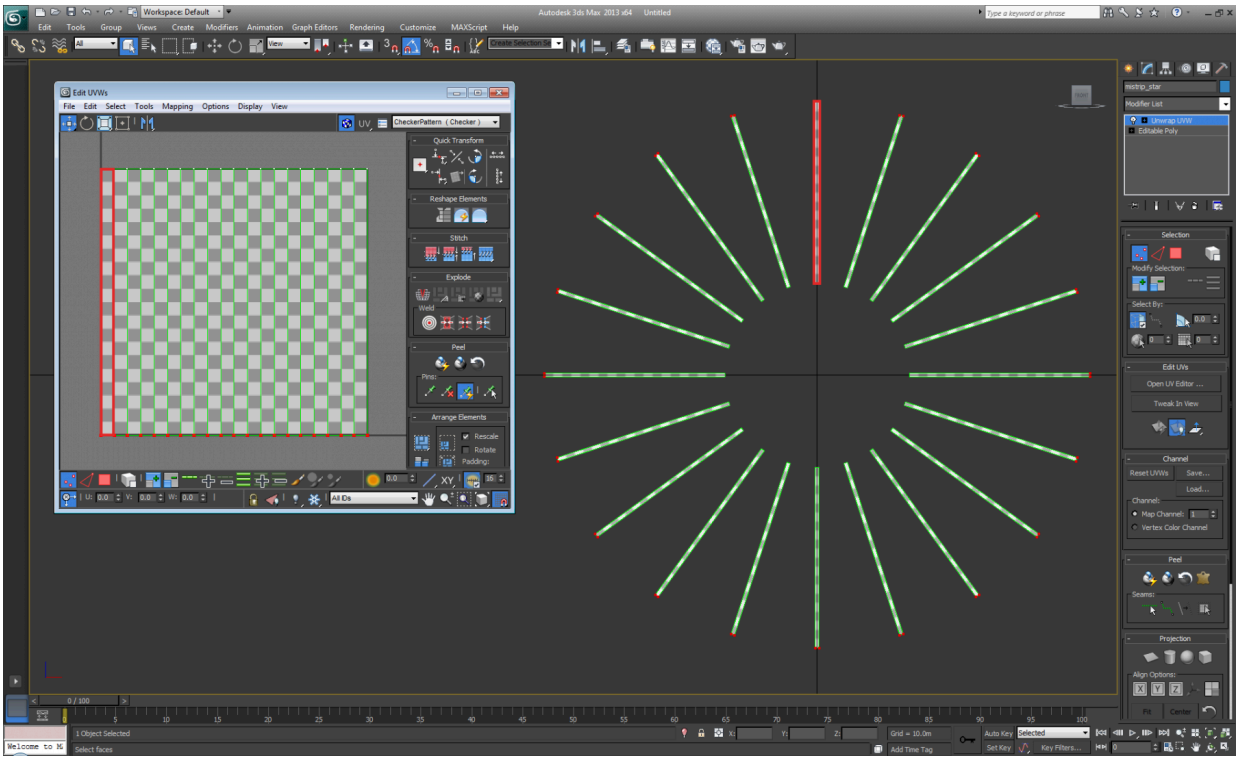
1. Draw a plane that matches a MiSTRIP's dimensions, 0.063x1.484 metres.
2. Ensure you have the box called **generate mapping coordinates** checked. Doing so will enable the plane to automatically generate a normalised UV map.
3. Position the plane's local pivot point to the scene's origin. The origin is 0,0,0 in 3D space corresponding to the X,Y,Z axes.
4. With the plane selected, select the **Array** tool.
5. **Array** the plane by a count of **20** around **360** degrees. This will array twenty planes around the local pivot point.
6. **Attach** all the plane's together into one **Editable Mesh / Editable Poly**.
7. Select the**Unwrap UVW**modifier.
8. Open the **Edit UVWs** window to view the UV map.
9. At this stage, the UV map will have overlapping UV shells. To **unwrap** the UV map, reposition the UV shells next to each other, either **manually** using UV editing tools or **automatically** using a script.
10. It is important to **snap** the UV shells next to each other into a **grid**. Doing so will remove gaps in the UV map, which in turn will remove virtual pixels from Disguise's output to the LED processor.
11. **Normalise** the UV map by rescaling it to fill UV space entirely from a range of 0-1 in both the U and V axes.

The image below shows the MiSTRIPs arrayed in a 360 degree **arc**, but in UV space they are arrayed in a horizontal **row** as a **grid**. Importantly, the surface's outer ring of vertices correspond to the UV map's bottom row of UV coordinates, highlighted in red.

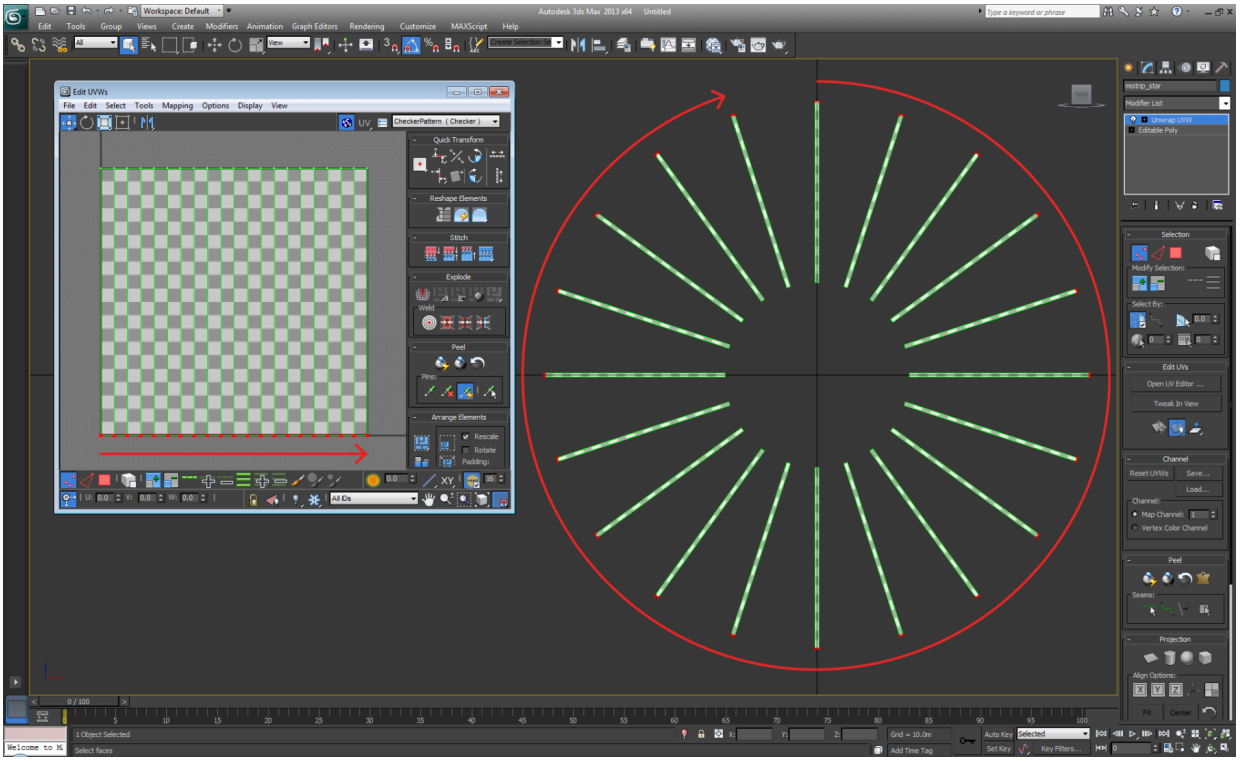
This indicates the surface's UV coordinates are **uniformly** positioned.



The image below shows how each MiSTRIP corresponds to a UV shell, highlighted in red. Importantly, each UV shell is *equal* in scale to each MiSTRIP, which indicates the UV map has an *even* distribution of UV coordinates.



The image below shows how the MiSTRIPs are arrayed in UV space. Importantly, the MiSTRIPs are arrayed in a *clockwise* direction, and this order follows the UV map, where the UV shells are arrayed in a *left to right* direction.



In order to set a resolution of this screen made of 20 Mistrips it is crucial to turn each strip into a rectangle in UV space.

Step 2 - Mapping content to screens

The left-image below shows a **Radar** layer **Parallel** mapped to the screen in Designer, which was exported as an **.obj** file from **Autodesk 3ds Max**. The right-image shows the **Output Feed** from Disguise to the LED processor, which is being generated in **realtime** by the Parallel map applied in Disguise's **Stage simulator**. The feed output is **independent** of the video content, which means the output can be a **different** resolution. In this case, the content has a resolution of 1024x1024 pixels, but the output has a resolution of 20x112 pixels. This is possible because **the Parallel map does not apply content directly to the screen's UV coordinates**; enabling the content creator to design content using templates that are **independent** of the LED outputs. Therefore, the creative workflow for the content creator is **separated** from the technical workflow for the hardware technicians, as explained on The UV Map as the Hardware Output page.

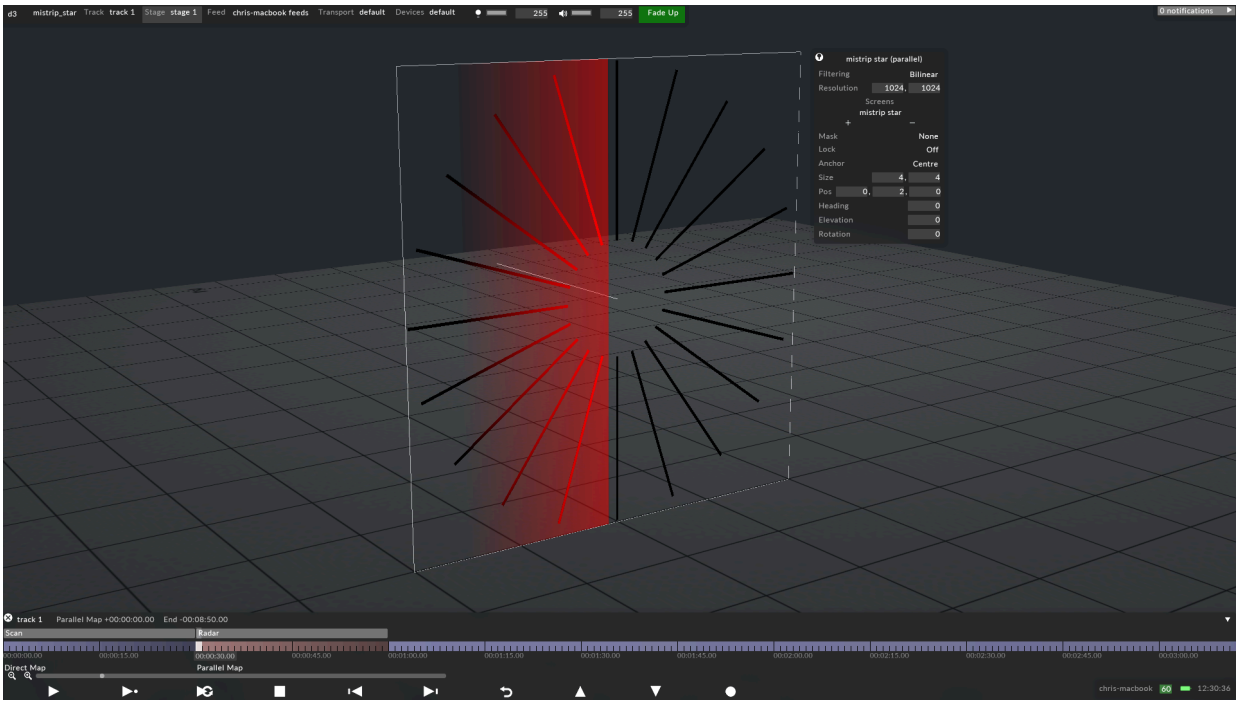
The resolution of the screen in Designer should **exactly** match the physical screen's number of pixels. The resolution of 20x112 pixels was calculated by:

- Knowing precisely how many pixels each MiSTRIP contains **horizontally**.
- Multiplying this value by the **total** number of horizontal MiSTRIPs, to calculate a horizontal resolution.
- Knowing precisely how many pixels each MiSTRIP contains **vertically**.

In this case:

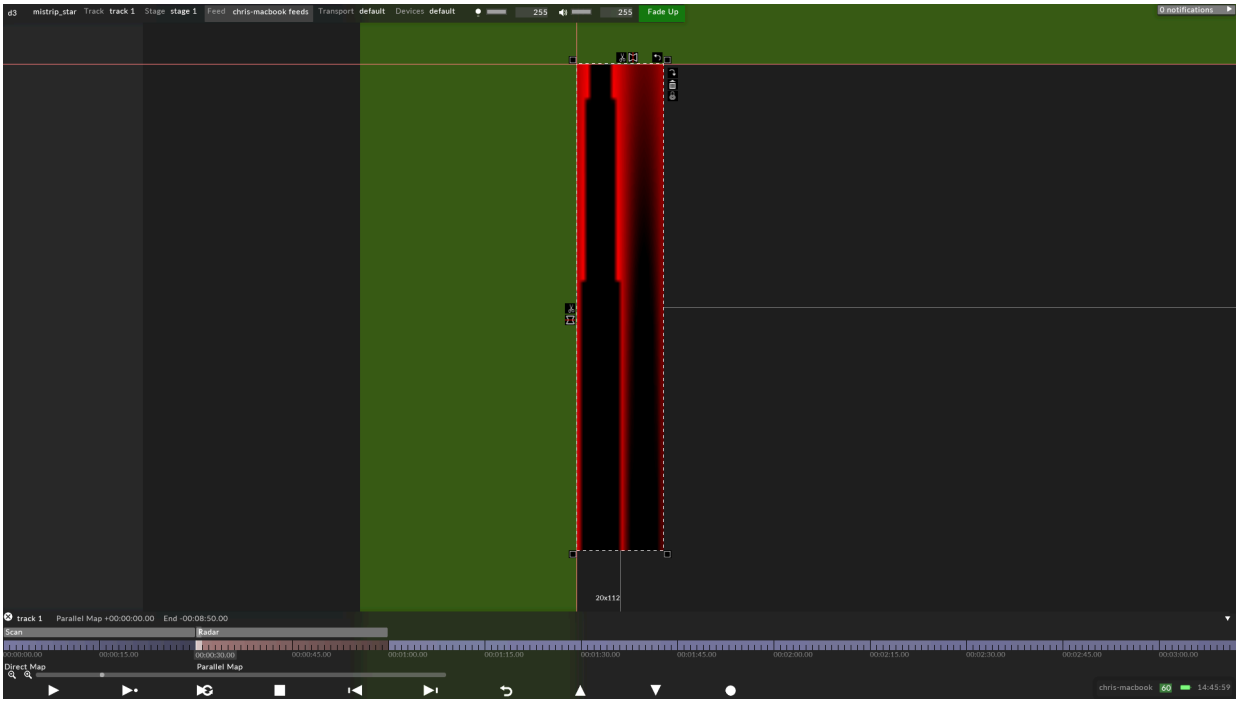
- 1 MiSTRIP = 1 pixel **horizontally**.
- The total number of horizontal MiSTRIPs is 20. Therefore, 1 pixel x 20 MiSTRIPs = 20 pixels horizontally in **total**.
- 1 MiSTRIP = 112 pixels **vertically**.

> **Note**
>
> **Please note**: When setting the resolution of a screen in the software, the UV map is **divided** into a **grid** of the same resolution, and in turn each of the grid's quads are **sampled** by the screen and **outputted** to the LED processor. Therefore, it is important to **match** the resolution of Disguise's screen to the physical screen.



The UV map also represent the output sent to the physcial LED strips.

The image below shows a Population mask applied to the screen in Designer, which was exported as a **.png** from **Adobe Photoshop**. The Population mask has removed MiSTRIP sections from the screen and, in turn, from the output to the LED processor, which maybe necessary if the screen's geometry requires updating. To read more about generating Population masks, see the Population masks page.


mistrip_star
track 1
stage 1
chris-macbook feeds
default
default
255
255
Fade Up
0 notifications
20x112
track 1
Parallel Map +00:00:00.00
End -00:08:50.00
Scart
Radar
Direct Map
Parallel Map
chris-macbook
60
14:45:59

Example 2 - OLite column

This case study shows how to prepare a column shaped LED screen constructed from Barco OLites for for Designer. As explained on the MiSTRIP Star page, the first step is to UV map the 3D model by **unwrapping** the mesh into a **grid**.

How the UV map is generated

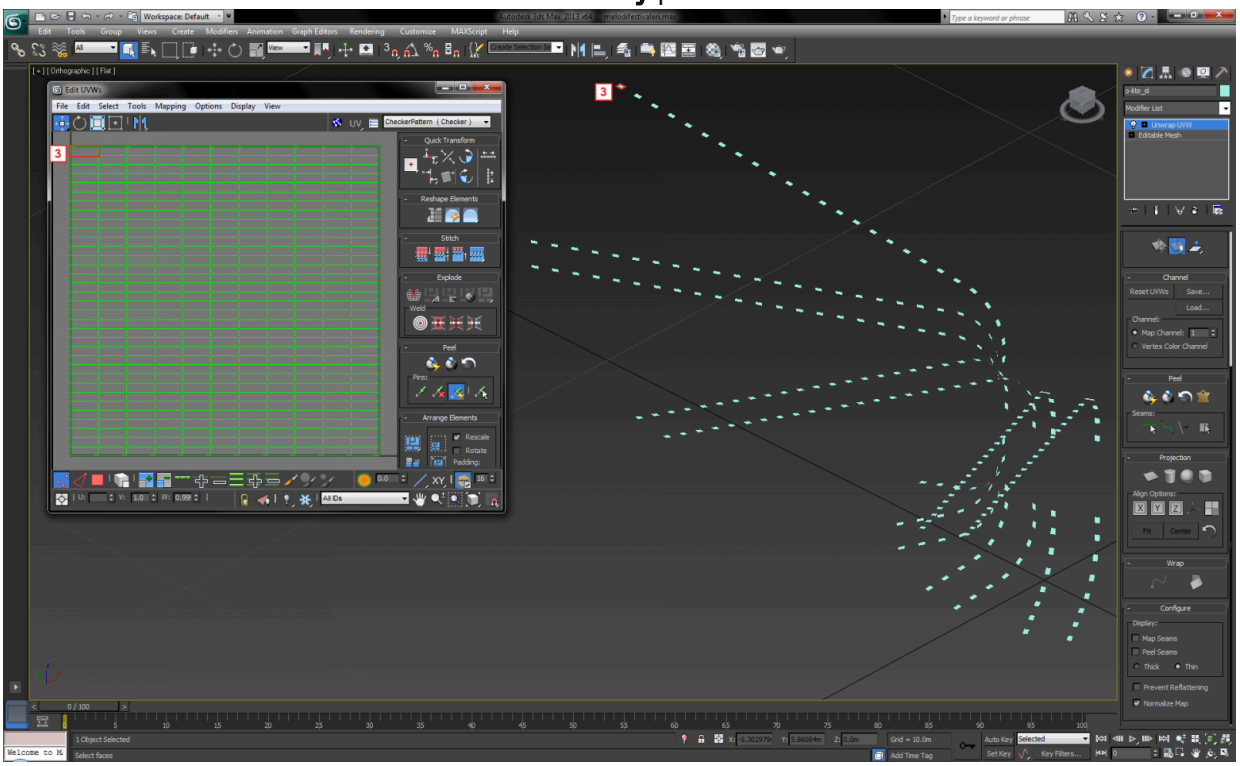
A plane can be **arrayed across a spline** to recreate the surface with the UV map **automatically generated**. However, the UV map will have **overlapping** UV shells and therefore will require editing, either **manually** using UV editing tools or **automatically** using a script, depending on the level of complexity.

Step 1 - Generating the UV map (3ds Max)

1. Draw a plane that matches an OLite's dimensions, 0.084x0.112 metres.
2. Ensure you have the box called **generate mapping coordinates** checked. Doing so will enable the plane to automatically generate a normalised UV map.
3. Draw a spline that snaps to the outer edge of one OLite column.
4. With the plane selected, select the **Spacing** tool.
5. From the spacing tool's parameters select **pick path**, and select the spline.
6. From the spacing tool's parameters select **count**, and enter a value of **34**. This will array thirty four planes across the spline.
7. **Attach** all the plane's together into one **Editable Mesh / Editable Poly**.
8. **Clone** the column by a count of **11**.
9. Align each column into position using the **move** and **rotate** tools.
10. **Attach** all the column's together into one **Editable Mesh / Editable Poly**.
11. Select the **Unwrap UVW**modifier.
12. Open the Edit UVWs window to view the UV map.
13. At this stage, the UV map will have overlapping UV shells.To **unwrap**the UV map, reposition the UV shells next to each other, either **manually** using UV editing tools or **automatically** using a script.
14. It is important to **snap** the UV shells next to each other into a **grid**. Doing so will remove gaps in the UV map, which in turn will remove virtual pixels from Disguise's output to the LED processor.
15. **Normalise** the UV map by rescaling it to fill UV space entirely from a range of 0-1 in both the U and V axes.

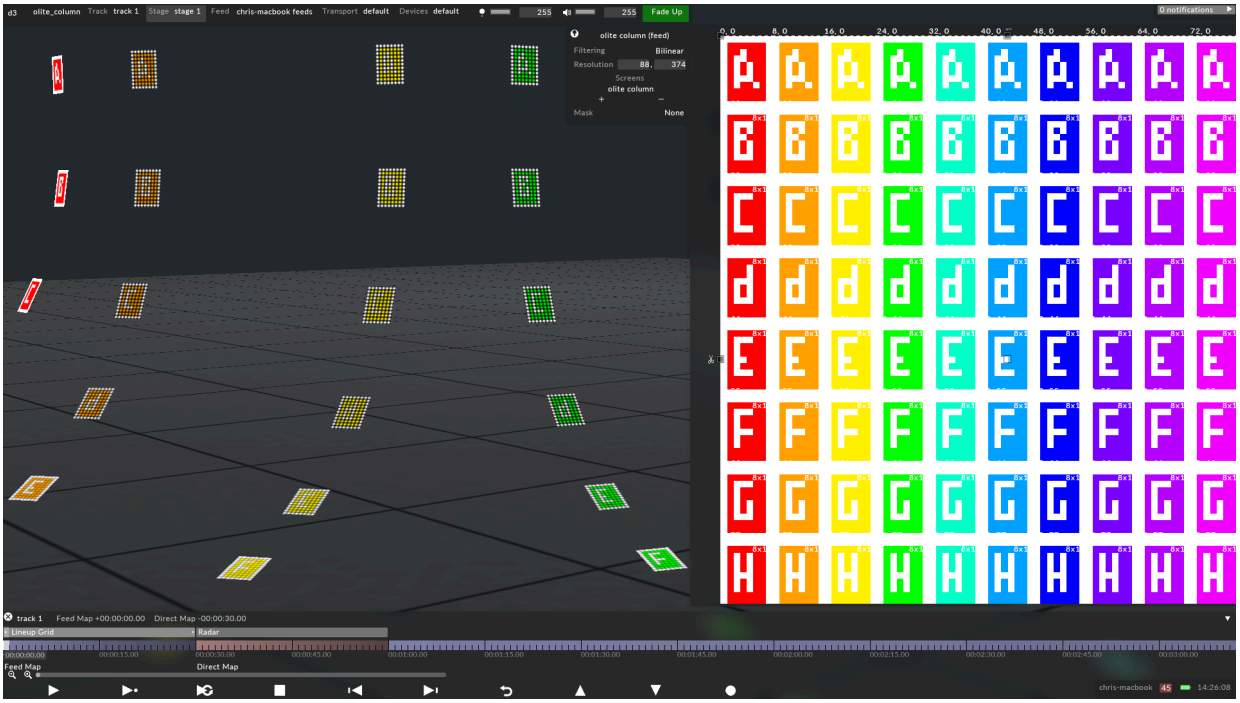
The image below shows the OLites arrayed in a **curved** direction, but in UV space they are arrayed in a **linear** direction as a **grid** Importantly, the surface's outer ring of

polys correspond to the UV map's top row of UV shells, highlighted in red.
This indicates the surface's UV shells are **uniformly** positioned.



Step 2 - Mapping content to screens

The image below shows a content template **Feed** mapped to the screen in Disguise, which was exported as an **.obj** file from **Autodesk 3ds Max**. The template was rendered from the UV map, and therefore contains no information about the **physical gaps** between the OLites. If video content is to be mapped to the screen with the physical gaps included, then a **Feed** map could be generated in d3. The Feed map **samples** different areas of the template, and in turn the sampled areas are **mapped** to the OLites. The advantage here is the sampled areas can be **reconfigured**, allowing for content to be moved, cropped, scaled and rotated, in the same way the Output Feeds can be reconfigured.



A **Parallel** or **Perspective** map can be generated from Designer. As explained on the MiSTRIP Star page, these mappings**do not apply content directly to the screen's UV coordinates**; enabling the content creator to design content using templates that are **independent** of the LED outputs. Therefore, the creative workflow for the content creator is **separated** from the technical workflow for the hardware technicians, as explained on The UV Map as the Hardware Output page.

The resolution of the screen in Designer should **exactly** match the physical screen's number of pixels. The resolution of 80x374 pixels was calculated by:

- Knowing precisely how many pixels each OLite contains **horizontally**.
- Multiplying this value by the **total** number of horizontal OLites, to calculate a horizontal resolution.
- Knowing precisely how many pixels each OLite contains **vertically**.
- Multiplying this value by the **total** number of vertical OLites, to calculate a vertical resolution.

In this case:

- 1 OLite = 8 pixels **horizontally**.
- The total number of horizontal OLites is 11. Therefore, 8 pixels x 11 OLites = 88 pixels horizontally in **total**.
- 1 OLite = 11 pixels **vertically**.
- The total number of vertical OLites is 34. Therefore, 11 pixels x 34 OLites = 374 pixels horizontally in **total**.

> **Note**
>
> **Please note**:When setting the resolution of a screen in Designer, the UV map is **divided** into a **grid** of the same resolution, and in turn each of the grid's quads are **sampled** by the screen and **outputted** to the LED processor. Therefore, it is important to **match** the resolution of Disguise's screen to the physical screen. For further information see the How Does Sample UV Maps page.

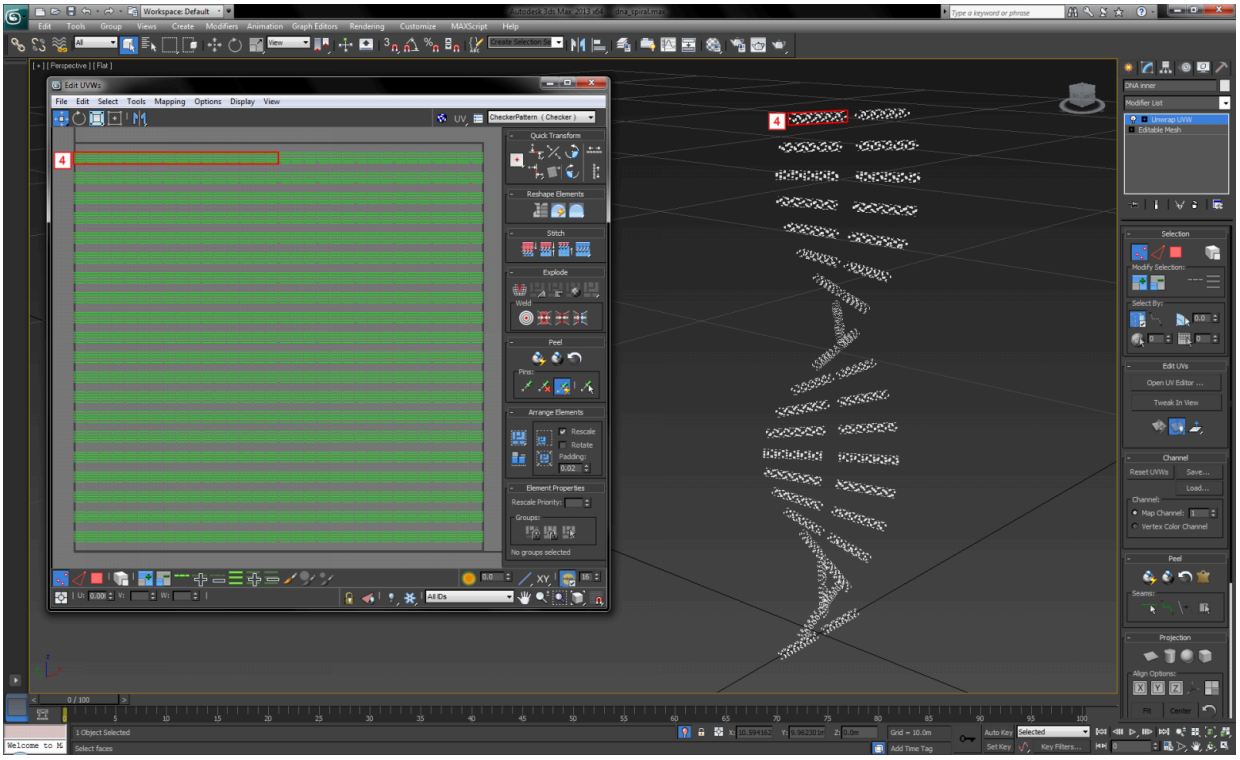
Example 3 - DNA Spiral

This case study shows how to prepare a spiral shaped LED screen constructed from pucks for Designer. As explained in the previous case studies, the first step is to UV map the 3D model by **unwrapping** the mesh into a **grid**.

How the UV map is generated

The DNA spiral's UV map was generated using a combination of techniques applied to the MiSTRIP Star and the OLite Column.

However, the key difference here is the UV map contains **gaps**. This is because the content creator required a 2D template that **accurately** represented the 3D model. In this case, **virtual gaps** were included to represent the **physical gaps** between the pucks.



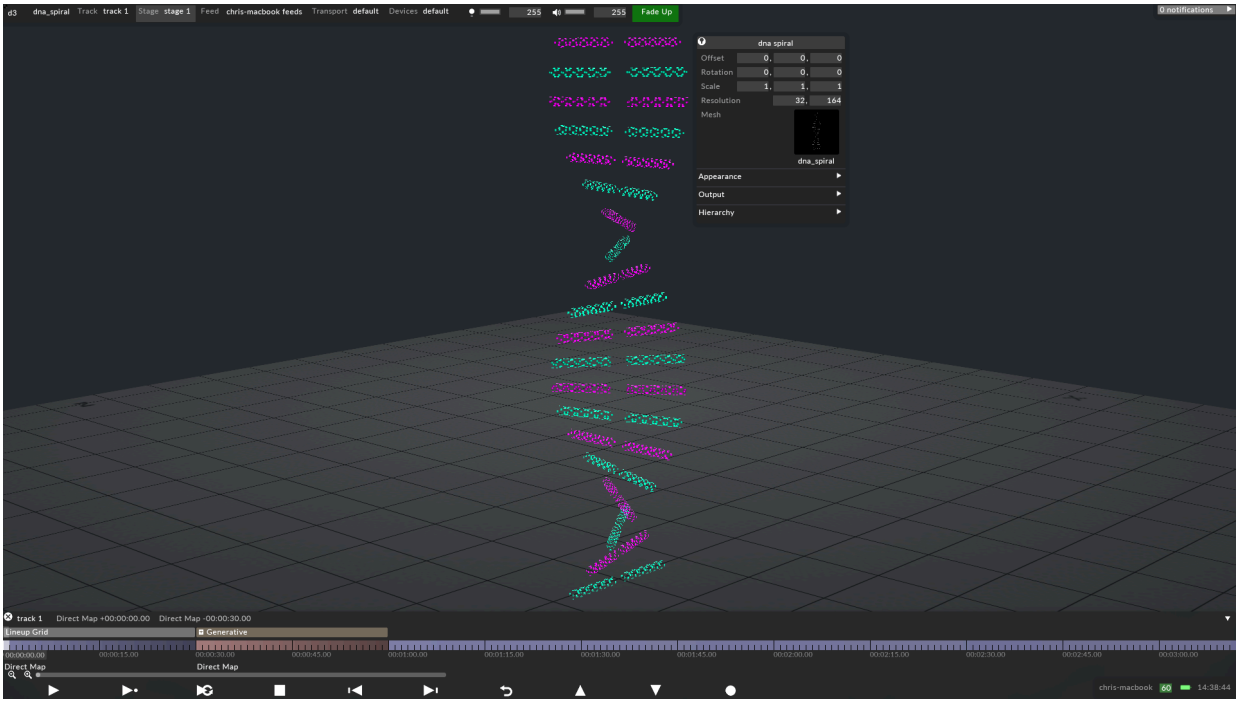
The UV map of the DNA spiral

The left-image below shows a content template **Direct** mapped to the screen in Designer, which was exported as an **.obj** file from **Autodesk 3ds Max**. The template was rendered from the UV map, and therefore includes **virtual gaps** to,represent the **physical gaps** between the pucks.

The Direct map samples the **entire** UV map including the **virtual gaps**, but only the UV coordinates **assigned** to the 3D model will display video content; enabling the content creator

to design content using a template which **accurately** represents the 3D model. However, here the disadvantage is Disguise's screen will require a **higher** resolution than the physical screen, **which may affect video performance**.

The right-image below shows the **Output Feed** from Designer to the LED processor. The output is **independent** of the content, which means the output can be a **different** resolution. In this case, the content has a resolution of32x164 pixels, but the output has a resolution of 32x82. This is possible because the Output Feeds can be **reconfigured**, allowing for virtual pixels to be **removed** from the output, and for the output to be **edited** to match the LED processor's pixel map. Therefore, the creative workflow for the content creator is **separated** from the technical workflow for the hardware technicians, as explained on The UV Map as the Hardware Output page.



Normally, the resolution of the screen in Disguise should **exactly** match the physical screen's number of pixels. However, in this case the resolutions can **differ**. The resolution of 32x164 pixels was calculated by:

- Knowing precisely how many pixels each puck contains **horizontally**.
- Multiplying this value by the **total** number of horizontal pucks, to calculate a horizontal resolution.
- Knowing precisely how many pixels each puck contains **vertically**.
- Multiplying this value by the **total** number of vertical pucks, and **doubling** the sum, to calculate a vertical resolution.

In this case:

- 1 puck = 1 pixel **horizontally**.
- The total number of horizontal pucks is 32. Therefore, 1 pixel x 32 pucks = 32 pixels horizontally in **total**.

- 1 puck = 1 pixel **vertically**.
- The total number of vertical pucks is 84. Therefore, 1 pixel x 82 pucks = 82, and 82 x 2 = 164 pixels horizontally in **total**.

The vertical resolution was **doubled** because the UV map contains **gaps**. **Importantly, the gaps were spaced evenly to match the heights of the UV shells**. Doing so ensures Disguise's screen will **precisely** sample the UV map. If the resolution is incorrect the screen will sample virtual pixels, which in turn will display **black** across the output. Therefore, it is important to generate a **pixel-perfect** content template to render video content from.

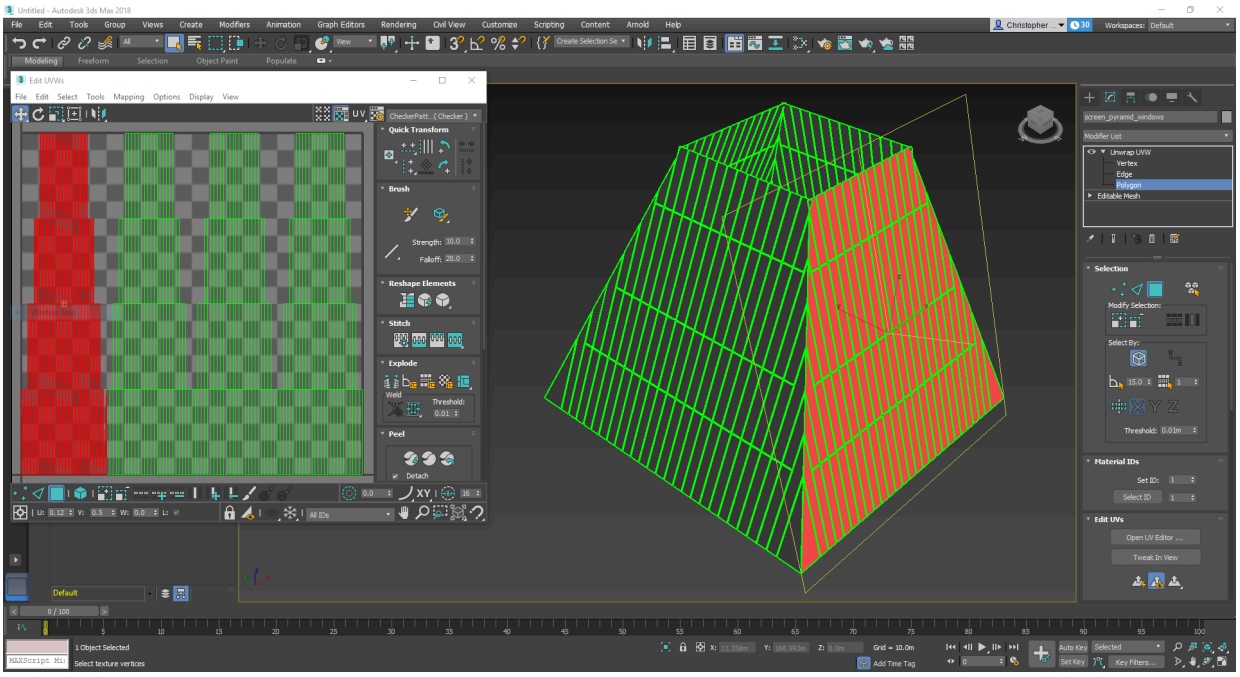
> **Note**
>
> **please** note**: When setting the resolution of a screen in Designer , the UV map is **divided** into a **grid** of the same resolution, and in turn each of the grid's quads are **sampled** by the screen and **outputted** to the LED processor. Normally, it is important to **match** the resolution of the screen to the physical screen but, in this case, Disguise's screen should have a **higher** resolution to allow for virtual pixels. Therefore, it is **highly recommended** to test playback performance before running a live show. For further information see the How Does Sample UV Maps page.

Example 4-Millenia Tower

Example 4: Millenia Tower

This case study shows how Singapore's Millenia Tower was prepared for . The key difference here is the screens used Pulsar Chroma lights, which are **DMX** controlled fixtures. The previous case studies explained how to prepare **DVI** controlled fixtures for , although **maps** DMX pixels in the *same* way it does DVI pixels. Therefore, the 3D modelling and UV mapping techniques covered earlier apply here. However, it is important to understand 's DMX screen sends RGB data over Art-Net and other supported protocols, instead of the Output Feeds. To read more about DMX screens, see the Creating DMX screens sub-chapter.

How the UV map is generated

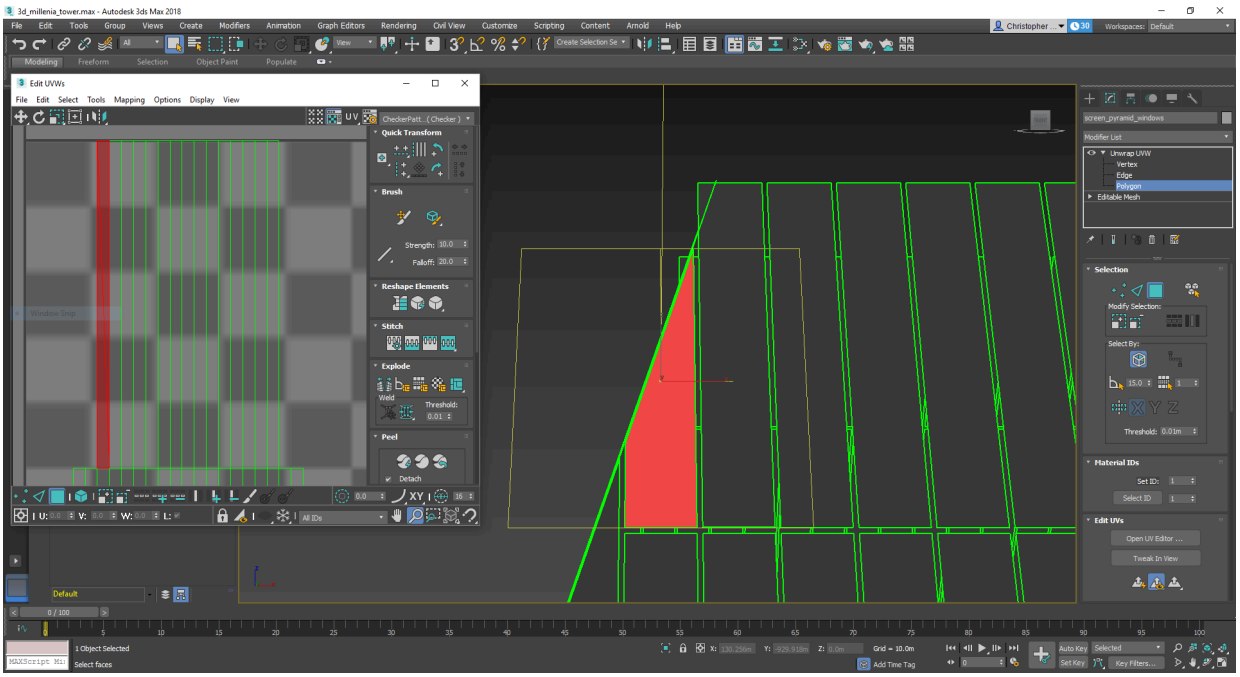


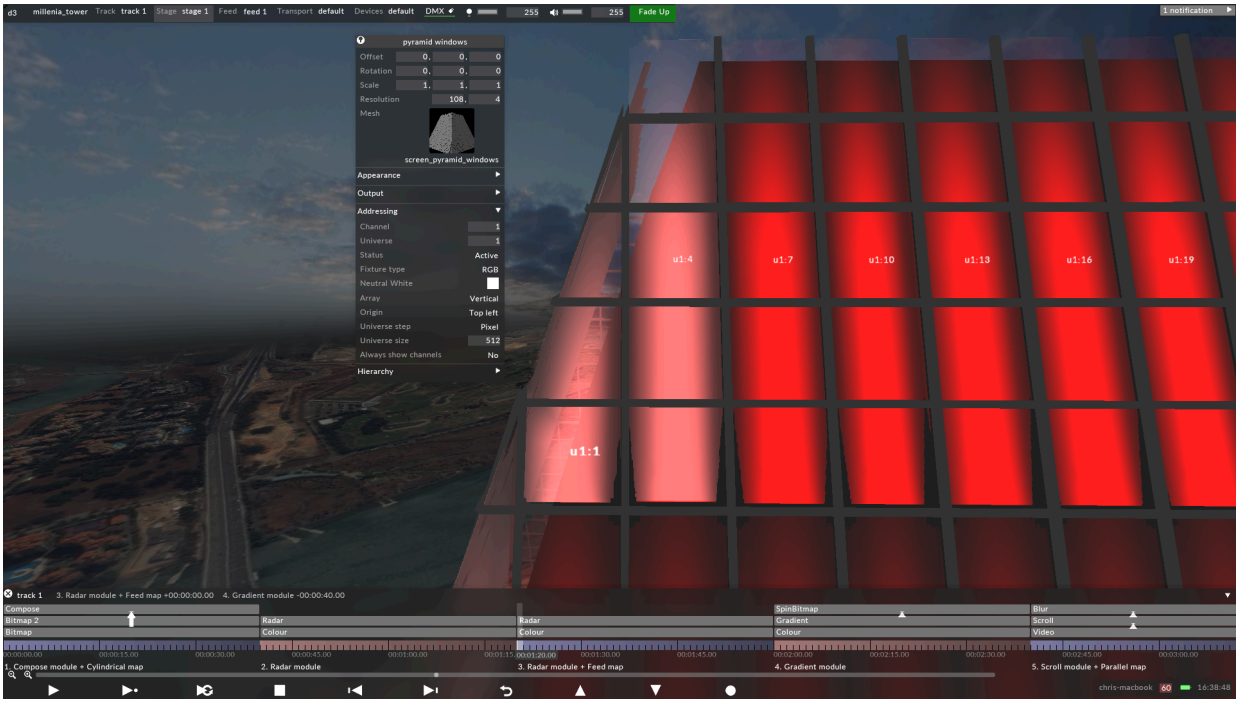
The Millenia Tower's UV map was generated using a combination of techniques applied previously, which explained how to **unwrap** a 3D model into a **grid**.

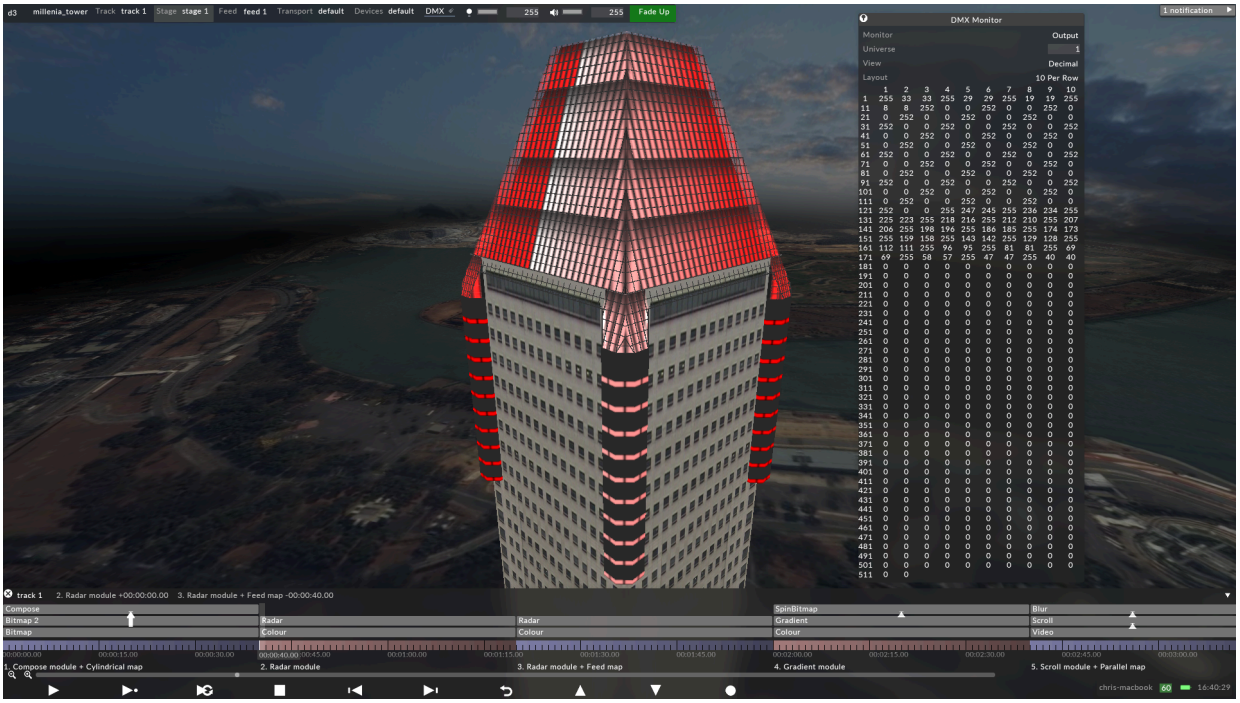
Similar to the DNA Spiral, this UV map contains gaps **separating** the tower's four sides, although in this case the 3D model does **not**contain the same gaps.

The UV map was designed on the basis that each Chroma light would be mapped in to a resolution of 1x1 pixels. Consequently, in **Autodesk 3ds Max** each Chroma light was assigned one UV shell, all of which **matched** in scale, because this ensured 's **Direct** map **precisely** sampled the UV map. Because the tower's upper levels contain less Chroma lights than the

lower levels, the UV map's upper levels contained less UV shells than the lower levels, resulting in **virtual gaps**.






DMX Monitor
1. Compose module + Cylindrical map
2. Radar module
3. Radar module + Feed map
4. Gradient module
5. Scroll module + Parallel map

Projection Examples Overview

This sub-chapter will teach you how to UV map Projection screens for Designer. The following examples revisit the UV mapping principles covered in the previous sub-chapter. Therefore, it is recommended to read this first.

Topics include:

Several mapping types in Designer:

- Cylindrical Mapping
- Pelt Mapping
- Perspective Mapping
- Pixel-perfect Mapping
- Planar Mapping
- Shrink Wrapping
- Spline Mapping

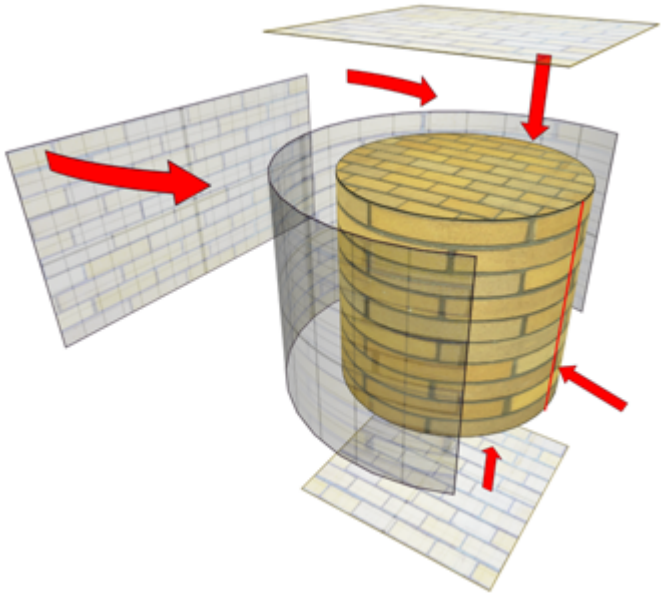


Projection Examples- Cylindrical mapping

Cylindrical mapping is suitable for projection surfaces **wrapping** around **360** degrees, for example a cylinder or a bottle.

How the UV map is generated

The Cylindrical map will **cylindrically** project UV coordinates to the surface, by **arraying** the map around 360 degrees. This will enable **symmetrical** content to flow across the surface **smoothly**. However, because the UV map is joined at a seam, **asymmetrical** content will **not** flow around the surface smoothly.

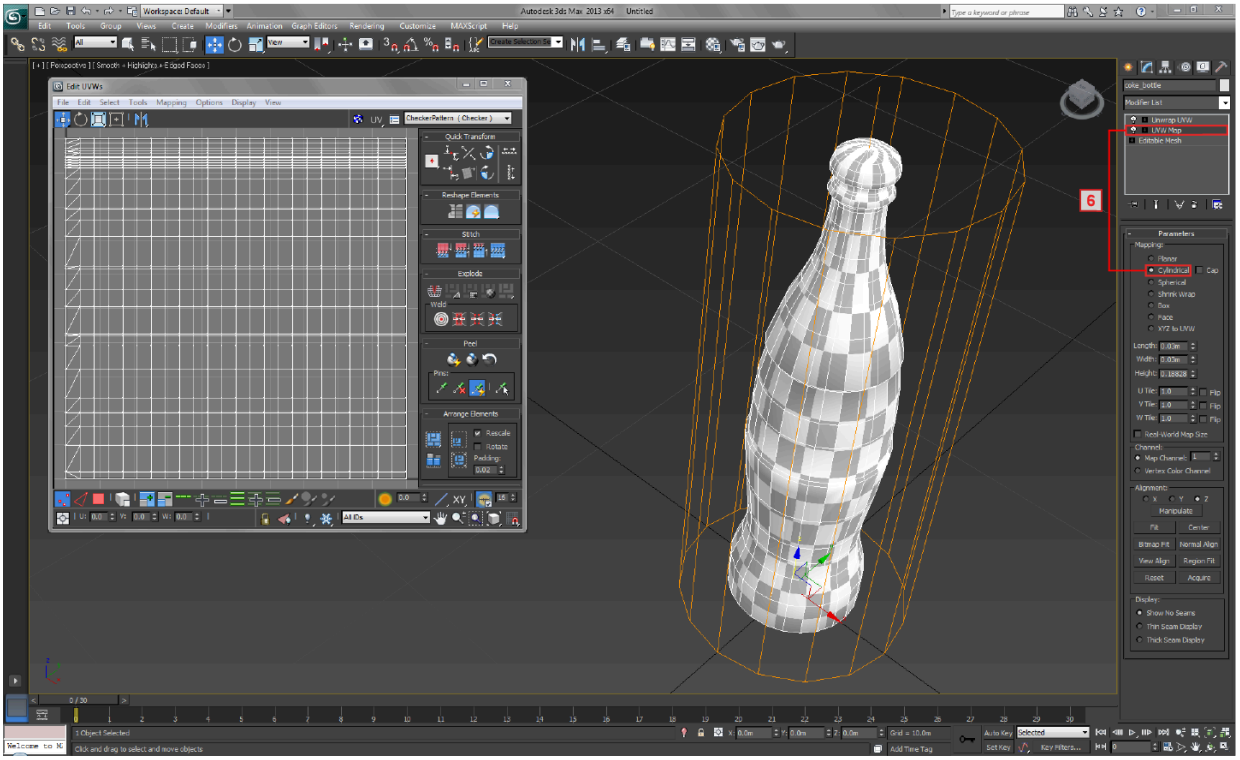


Step 1 - Generating the UV map (3ds Max)

1. Select the UVW Map modifier.
2. From the list of projection types select **cylindrical**.
3. The cylindrical map will project UV coordinates from its normals to the surface. Therefore, it is necessary to fit the cylindrical map's position, orientation and scale to the surface.
4. Select the Unwrap UVW modifier.
5. Open the Edit UVWs window to view the UV map.

The image below shows a **non-uniform** checker pattern rendered to the surface, which indicates the UV map has an **uneven** distribution of UV coordinates. To fully unwrap the

surface the UV map should be ***relaxed***.

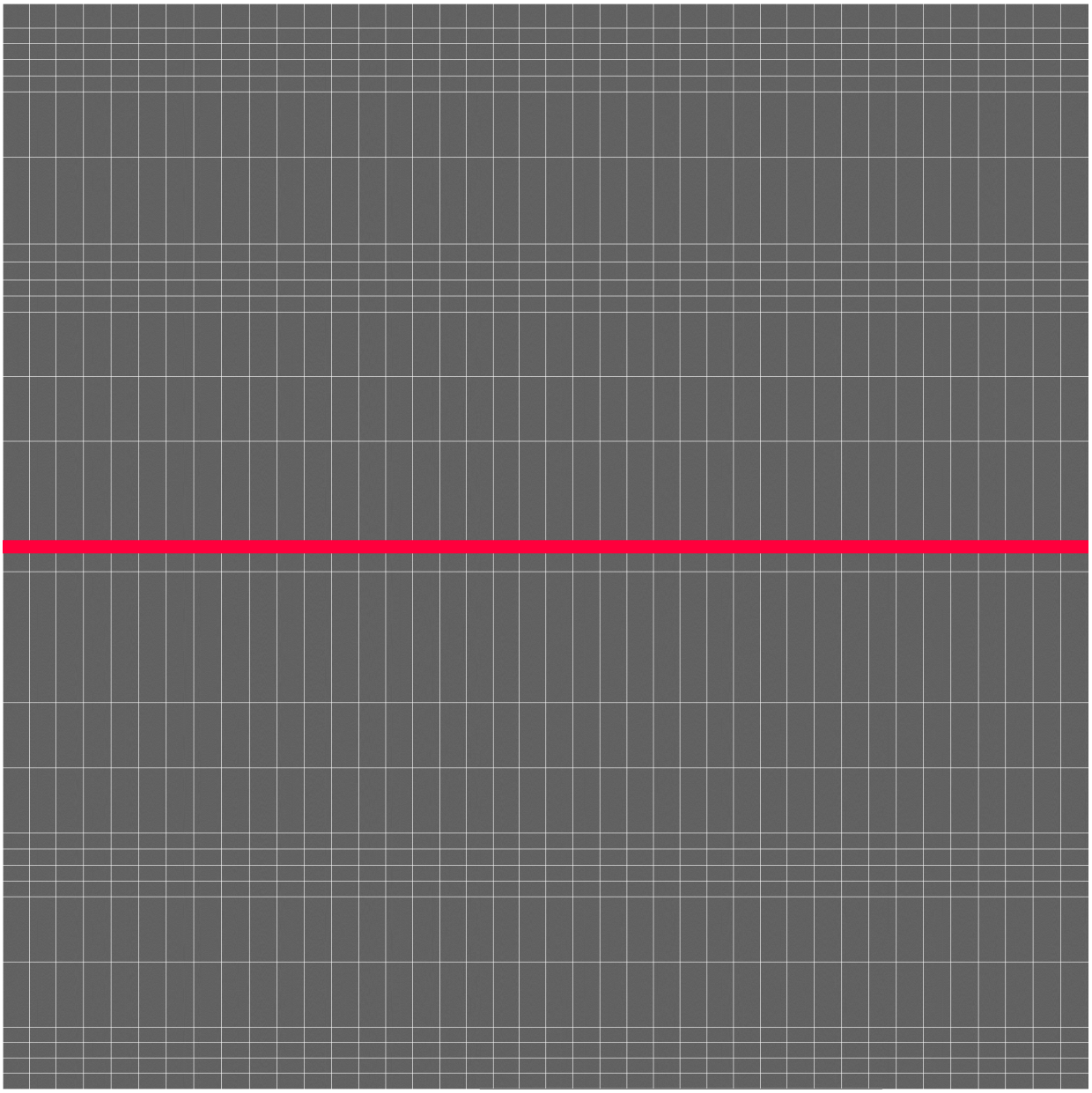


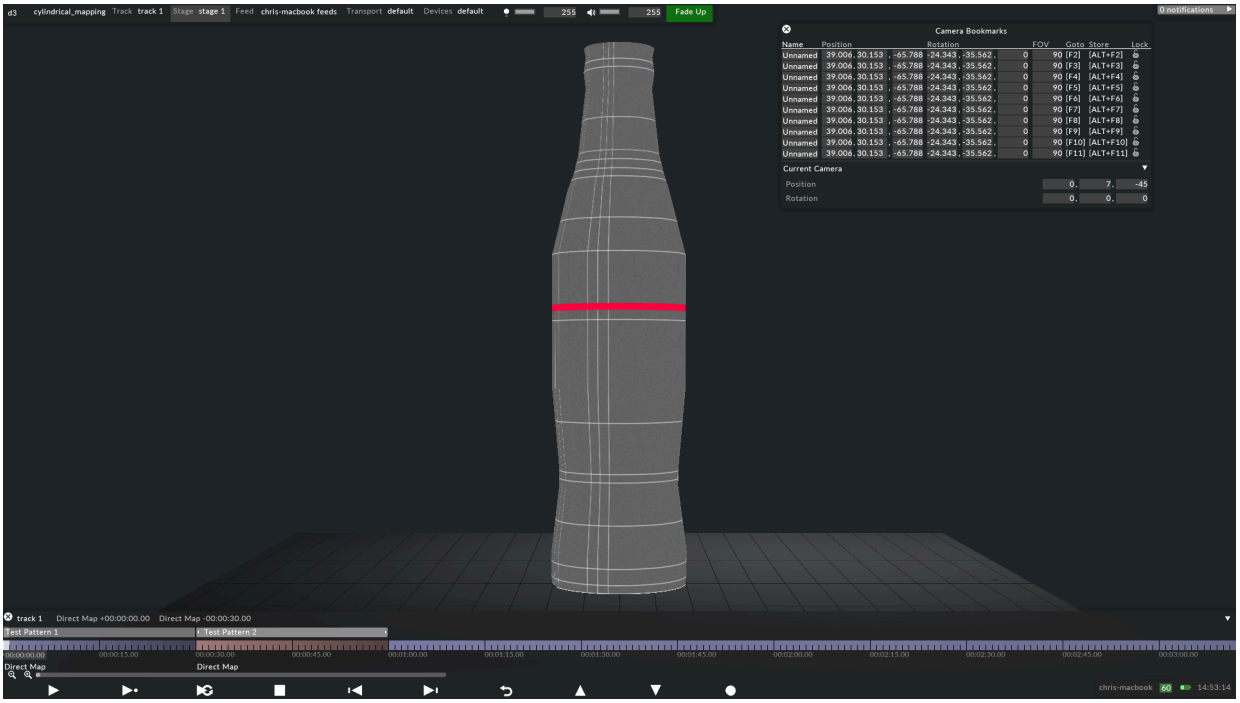
A 3D model of a coke bottle cylindrically mapped in 3ds Max.

Step 2 - Rendering UV maps to content templates

The images below show two content templates mapped to the screen in Designer. The template's resolution was calculated by following the same technique used for Shrink wrapped projection surfaces, because when a surface is Cylindrically mapped it is problematic to calculate the resolution **mathematically.**

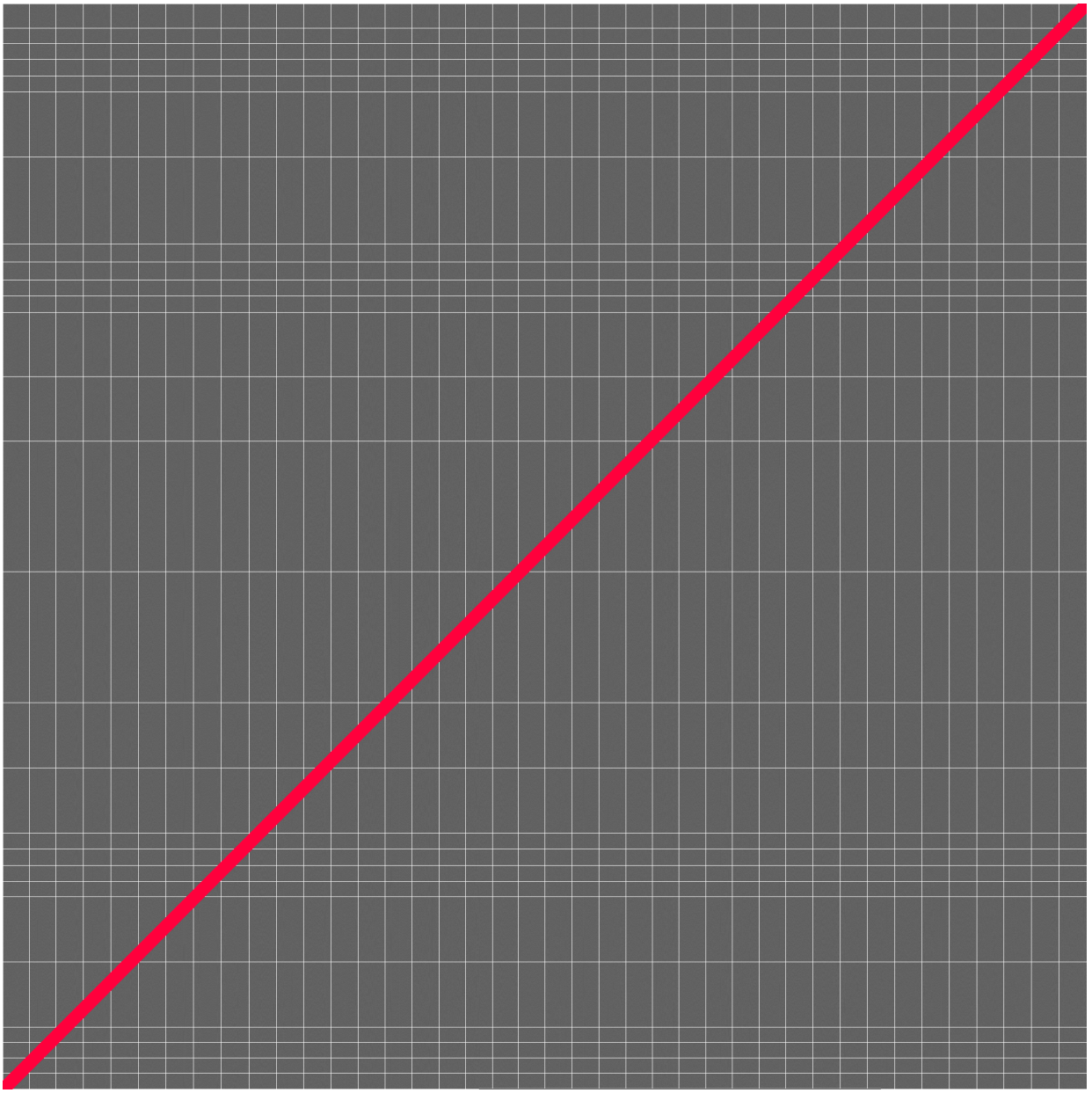
If a **horizontal** line is drawn across the template, in it will seamlessly map to the screen.

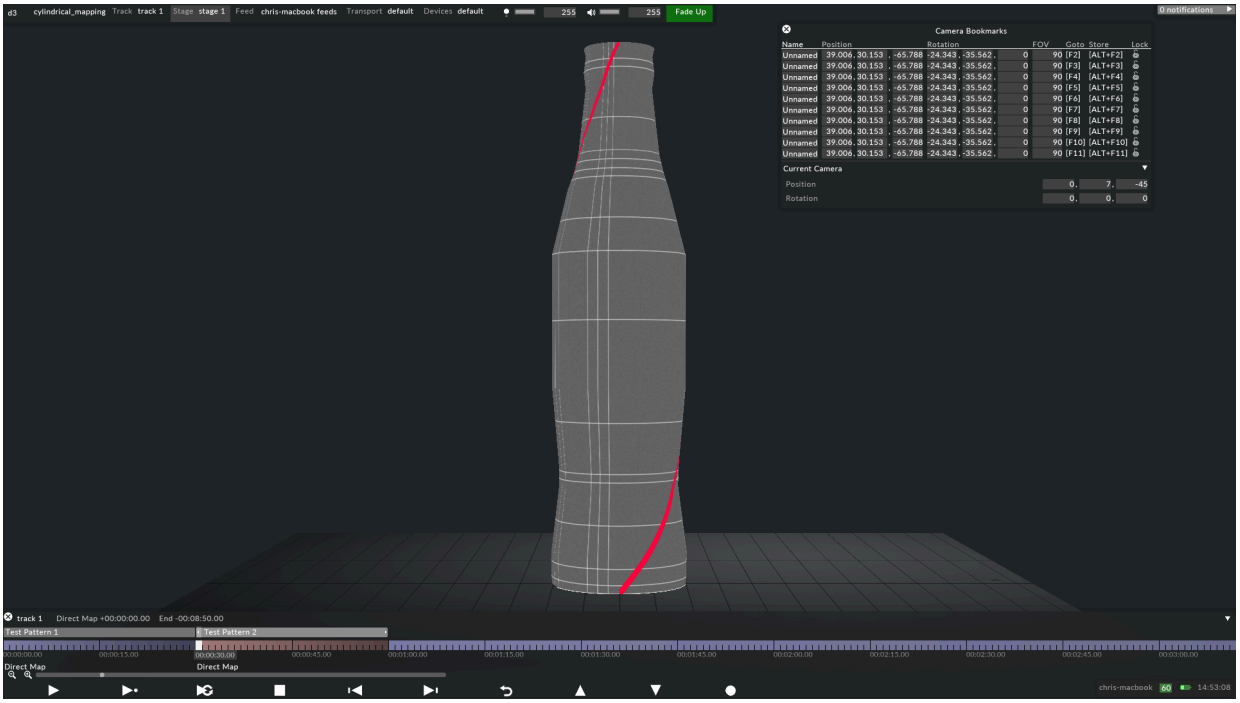




A 2D content template rendered from 3ds Max and applied to the coke bottle in . Notice how the horizontal line maps onto the 3D model seamlessly.

If a **diagonal** line is drawn across the template, in it will reveal a seam when mapped to the screen.





A 2D content template rendered from 3ds Max and applied to the coke bottle in . Notice how the diagonal line reveals a seam at its start and end points.

Helpful resources

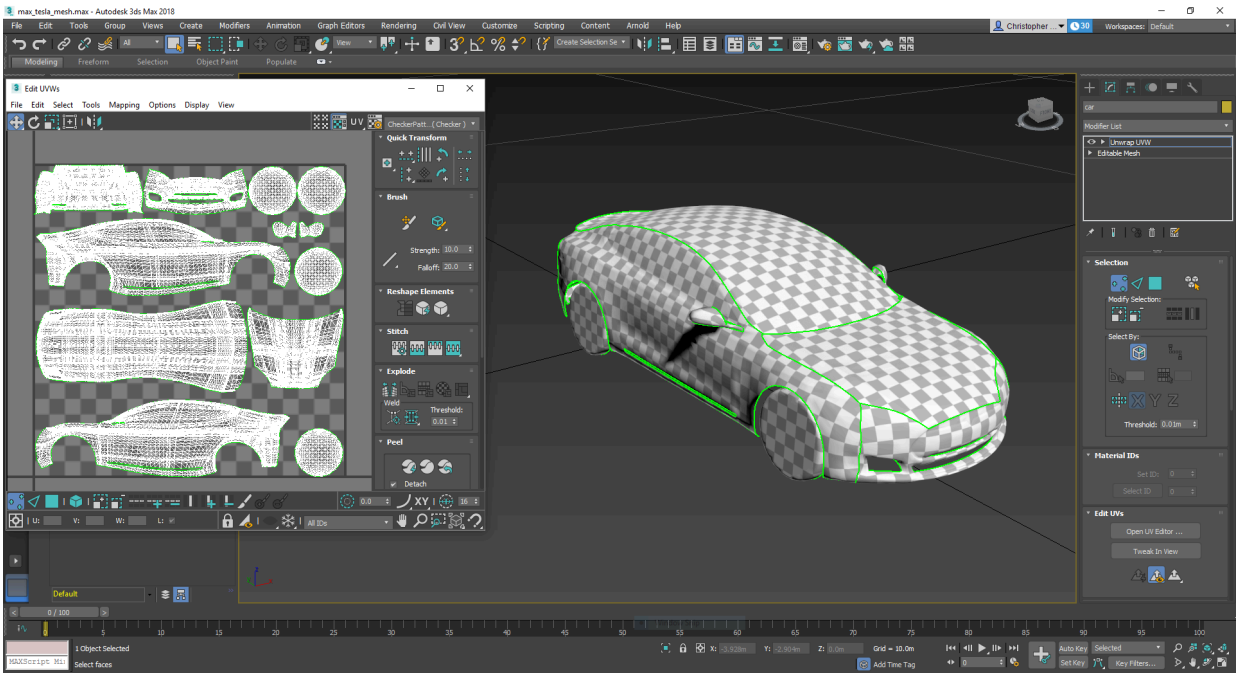
- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Flattening UVs in 3ds Max
- Unfolding UVs in 3ds Max
- Rendering content templates in 3ds Max

Projection Examples - Pelt mapping

Pelt mapping is suitable for projection surfaces **curving** across **multiple** axes, which areproblematic to unwrap using other mapping types, for example a body or a head.

How the UV map is generated

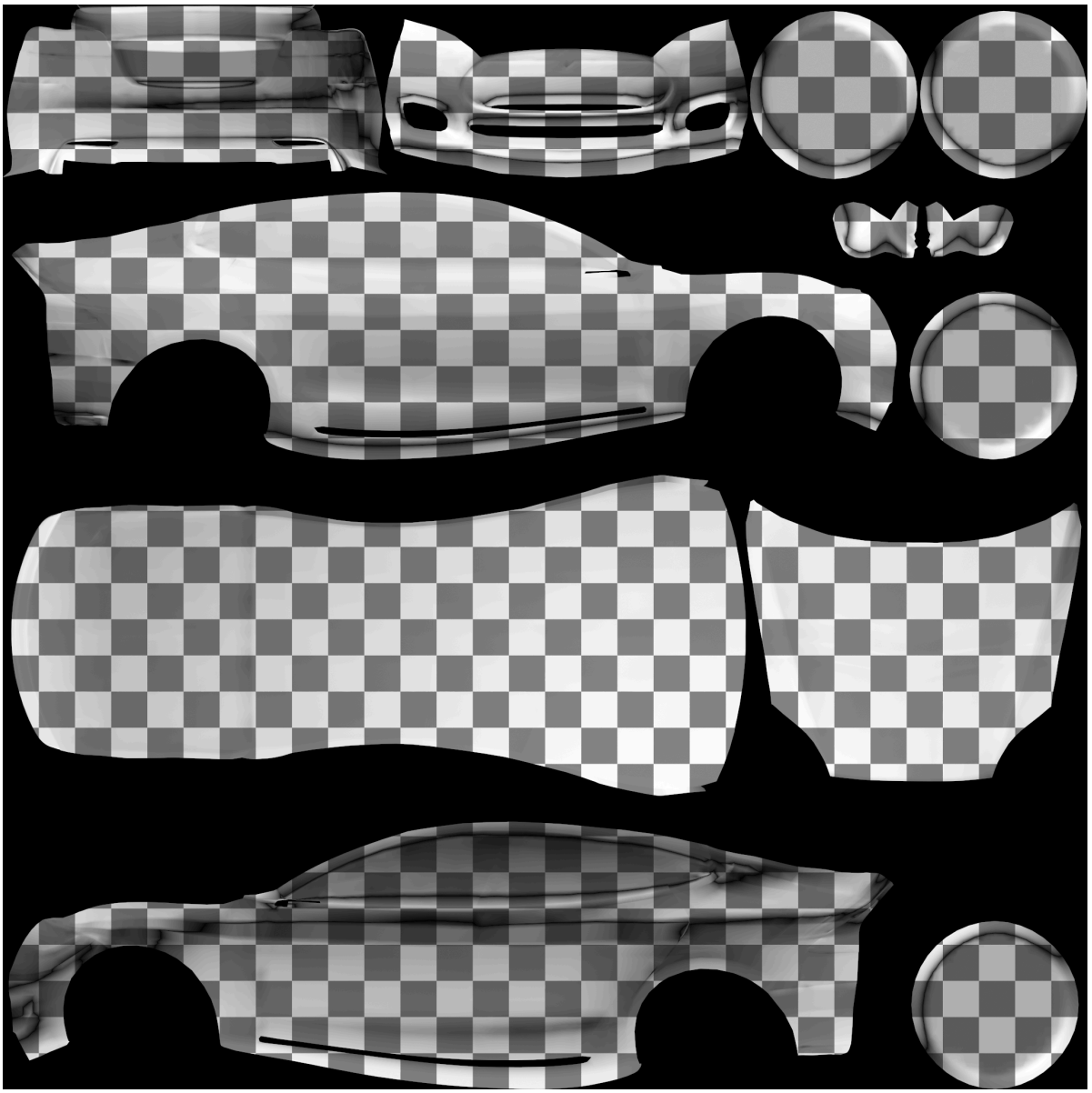
The Pelt map features a **stretcher** that can be manipulated by **springs** to **pull** the UV map outwards. This enables **precision** control over the UV coordinates to ensure they are **evenly** distributed across the **most** complex surfaces, which other mapping types are incapable of **completely** unwrapping.



Step 1 - Generating the UV map (3ds Max)

1. Select the **Unwrap UVW** modifier.
2. From the list of projection types select **pelt mapping**.
3. Open the **Edit UVWs** window to view the UV map. The image below shows a **non-uniform** checker pattern rendered to the surface, which indicates the UV map has an **uneven** distribution of UV coordinates.

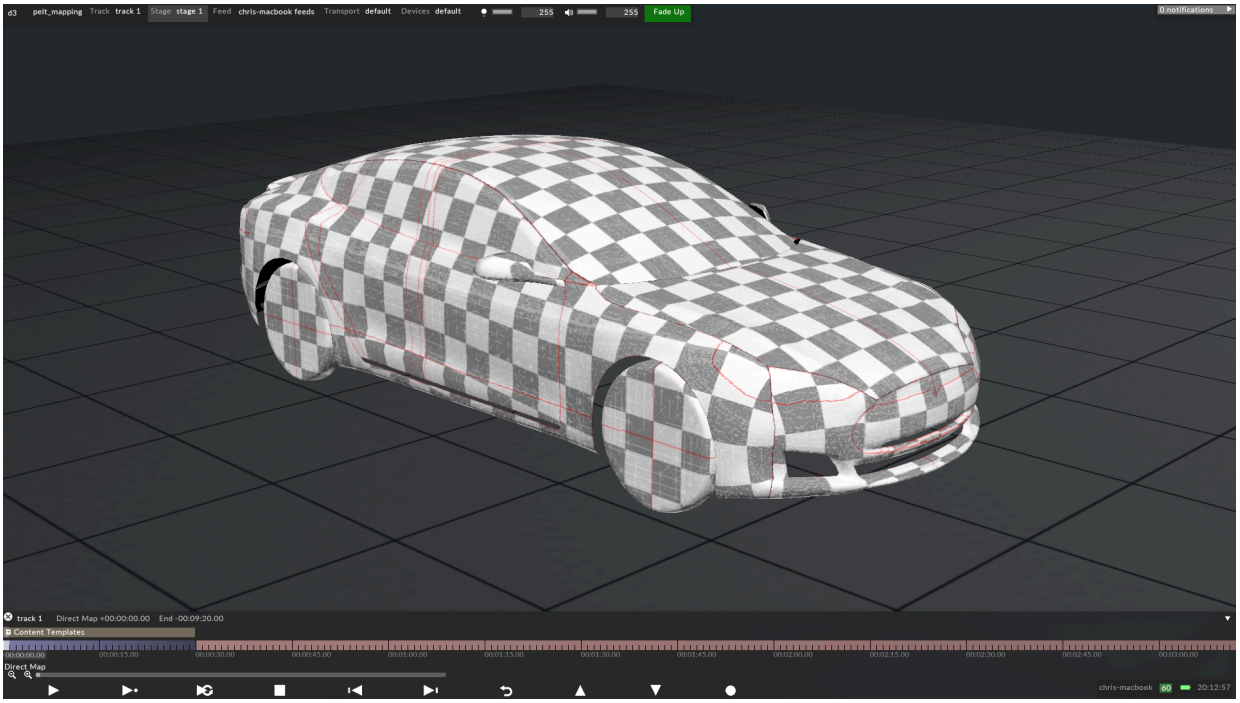
4. To fully unwrap the surface the pelt map's stretchers should be **pulled** outwards until the UV map **accurately** represents the **true** shape of the projection surface.



A 3D model of a car pelt mapped in 3ds Max.

Step 2 - Rendering UV maps to content templates

The image below shows the content template mapped to the screen in Disguise. The template's resolution was calculated by following the same technique used for Shrink wrapped / Cylindrical mapped projection surfaces, because when a surface is Pelt mapped it is problematic to calculate the resolutio **mathematically**.



For more information

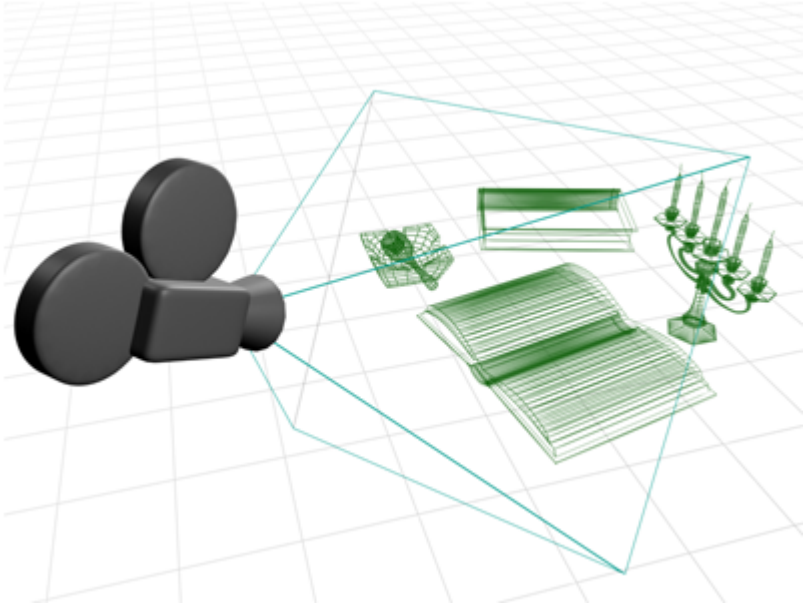
Autodesk 3ds Max Learning center

Projection Examples - Perspective mapping

Perspective mapping is suitable for projection surfaces to display content rendered to a single point of view, for example building blocks animating into a building. However, perspective mapping is also suitable for 2D content, as explained below.

How the UV map is generated

A camera can perspective project UV coordinates from its point of view to the surface. The UV map can then be rendered to a content template to be placed as a background in Adobe After Effects. Here the content creator can render 2D content and in Designer the content will map perfectly to the screen. The 2D content will appear correct from any point of view.



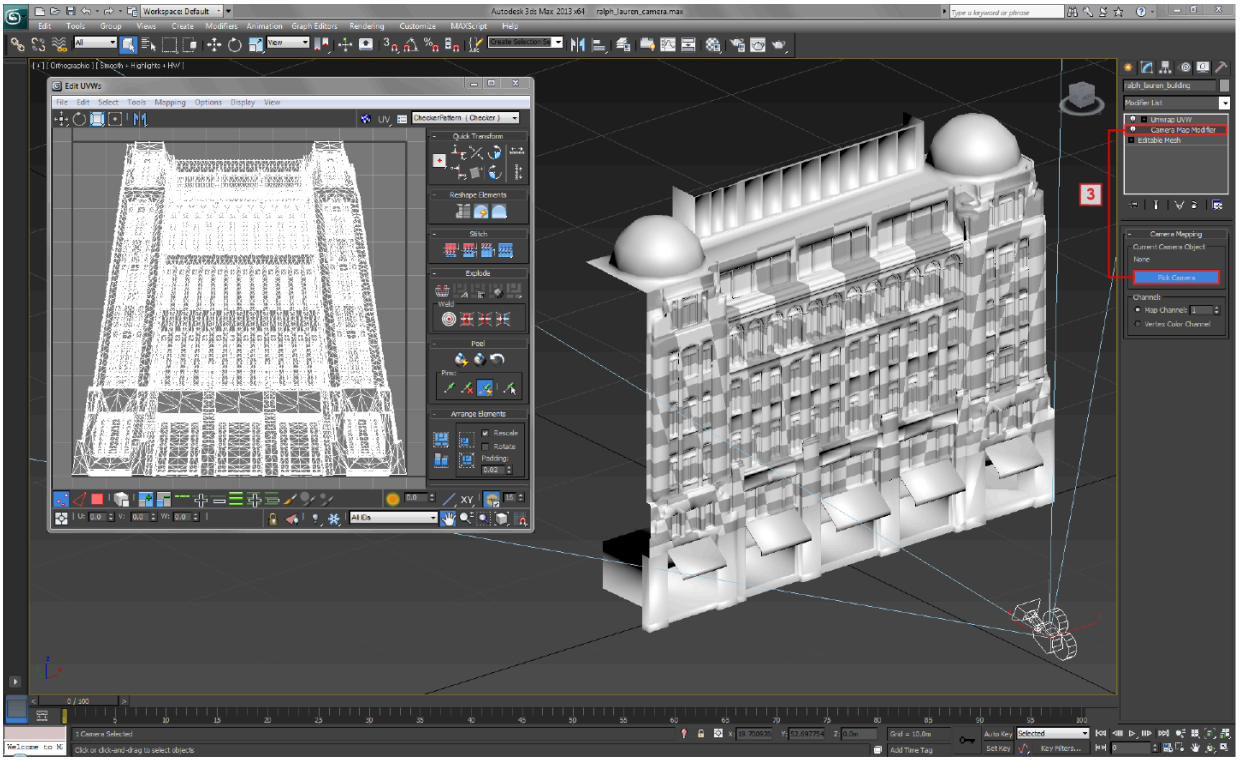
In a 3D application the content creator can also render 3D content by rendering 3D effects from a camera's point of view. The render camera should match the position of the previous camera, and the 3D content should be rendered to a resolution that matches the 2D content. The mapped 3D content will only appear correct from the render camera's point of view. Therefore, both cameras should be positioned to the point of view from which the audience will view the projection surface.

Step 1 - Generating the UV map (3ds Max)

1. Create a camera in your 3D scene.

2. Adjust its position, field of view and zoom, to capture the surface's outer edge. In the camera's viewport safe frames can be enabled to show the captured area more clearly.
3. Select the surface and apply a Camera Map modifier.
4. From the Camera Map modifier pick camera. The camera will perspective project UV coordinates onto the surface.
5. Select the Unwrap UVW modifier.
6. Open the Edit UVWs window to view the UV map.

The image here shows a non-uniform checker pattern rendered to the surface. However, from the camera's point of view the UV map will appear to have an even distribution of UV coordinates.



Step 2 - Rendering UV maps to content templates

- The images below show the content template and how its resolution was calculated. The template's resolution was calculated from the render settings set to generate 3D content, because when a projection surface is perspective mapped the UV map can be used to generate both 2D and 3D content. Therefore, the 2D and 3D content should match in resolution.
- The images below show the content template and how its resolution was calculated. The template's resolution was calculated from the render settings set to generate 3D content, because when a projection surface is perspective mapped the UV map can be used to generate both 2D and 3D content. Therefore, the 2D and 3D content should match in resolution.

- The UV map can be rendered to a template to be placed as a background in Adobe After Effects, to generate 2D content.
- The UV map can have animated geometry baked into it in a 3D application, for example 3ds Max, to generate 3D content.

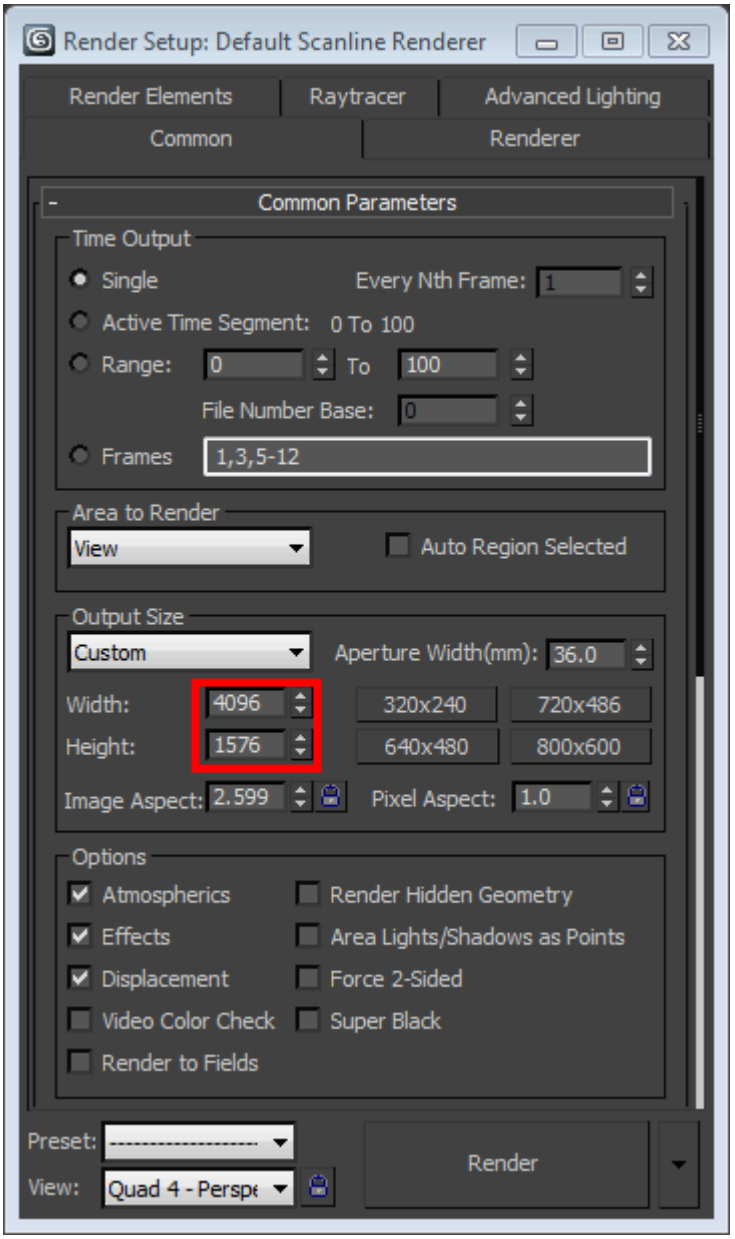
The template's resolution is 4096x1576 pixels. This was calculated by:

- Knowing the aspect ratio of the camera's output (this is automatically generated by the camera output's width and height, shown in the render settings).
- Knowing precisely how many pixels produced by the projector will hit the projection surface, across either the horizontal or vertical axis..
- Multiplying or dividing this value by the aspect ratio, to calculate a resolution for the remaining axis.

Therefore, the following calculations apply:

- The camera's output has an aspect ratio of 2.599.
- Approximately 4096 pixels will hit the projection surface horizontally.
- 4096 pixels / 2.599 = 1576 pixels (the vertical resolution).

Render settings from 3ds Max



The images below show the content template and examples of 2D and 3D content rendered from the UV map.



A 2D content template of the Ralph Lauren London Flagship Store renderedvfrom 3ds Max.



Helpful resources

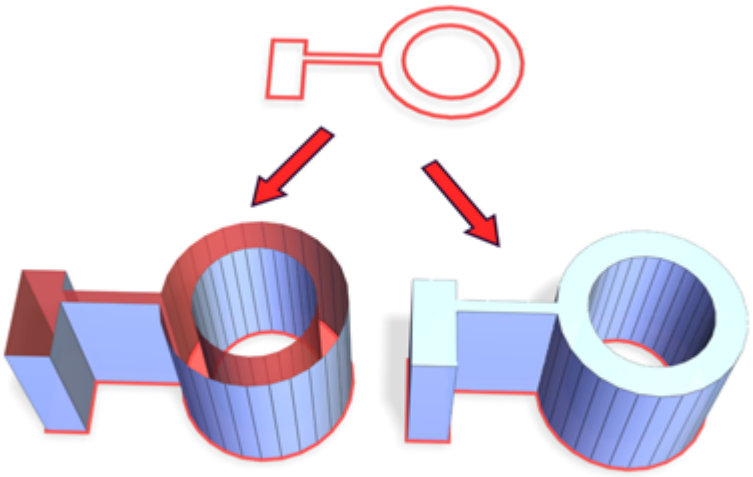
- Creating cameras in 3ds Max
- Projecting camera maps in 3ds Max
- Rendering content templates in 3ds Max

Projection Examples - Pixel-perfect mapping

Pixel-perfect mapping is suitable for projection surfaces **bending** across **one** axis, for example a wall constructed from multiple angled surfaces.

How the UV map is generated

A spline can be **extruded** to recreate the surface with the UV map **automatically generated**. Because the UV coordinates are generated by the extrusion, the UV map will have a **perfectly** even distribution of UV coordinates.

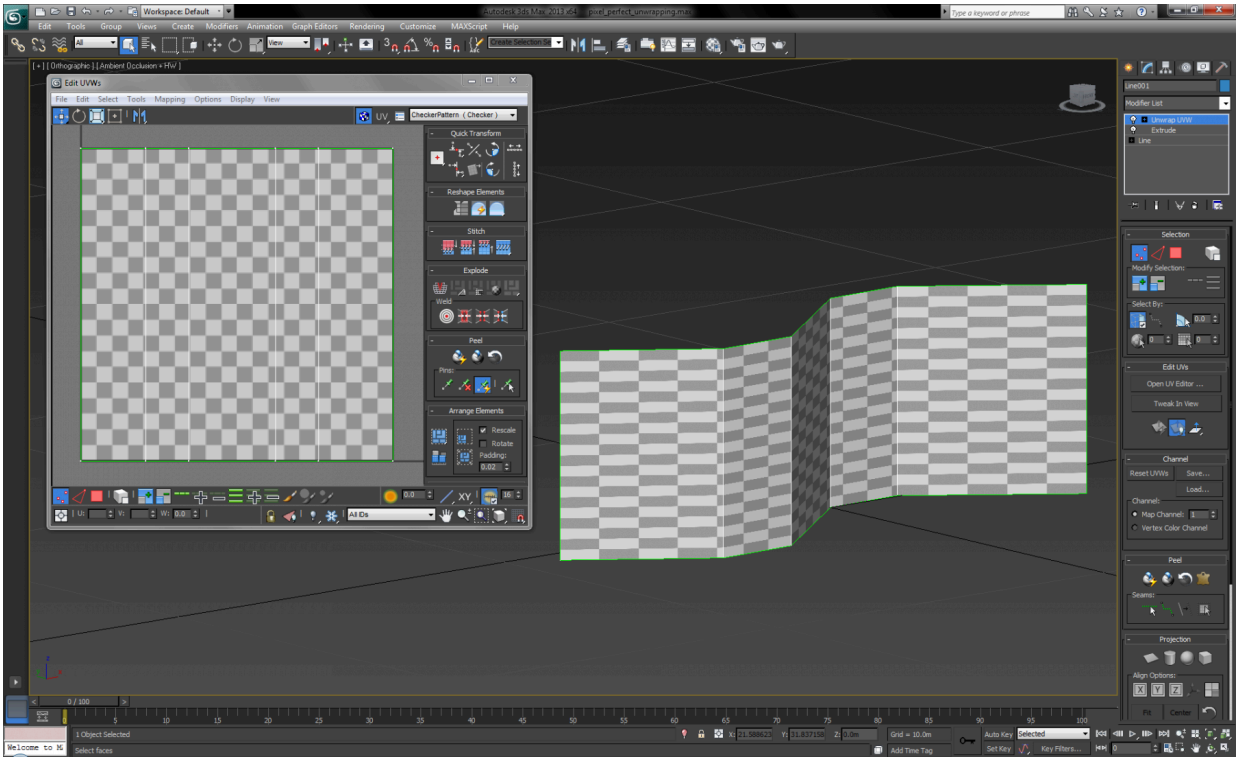


Step 1 - Generating the UV map (3ds Max)

1. Draw a spline that traces the surface's bottom edge.
2. Apply an Extrude modifier to extrude the spline to match the surface's height.
3. Ensure you have the box called **generate mapping coordinates** checked when extruding the spline. Doing so will enable the extrusion to automatically generate a normalised UV map.
4. Select the Unwrap UVW modifier.
5. Open the Edit UVWs window to view the UV map.

6. Check the UV coordinates are correctly **flipped** by comparing the positions of the surface's vertices to the UV map's texture coordinates. Before performing the check, ensure the surface is orientated and viewed in a south-north (front-back) direction. The image below shows the surface's top-left vertice corresponds to the UV map's top-left texture coordinate. This is correct. If the UV coordinates were incorrectly flipped, in , the video content would be **mirrored** across the screen when Direct mapped, for example a **Radar** layer would scan the screen in the **opposite** direction.

The image below shows a **uniform** checker pattern rendered to the surface, which indicates the UV map has an **even** distribution of UV coordinates.If the checker pattern appeared non-uniform, i.e. the checkers were differently sized, this would indicate the UV map had an **uneven** distribution of UV coordinates.

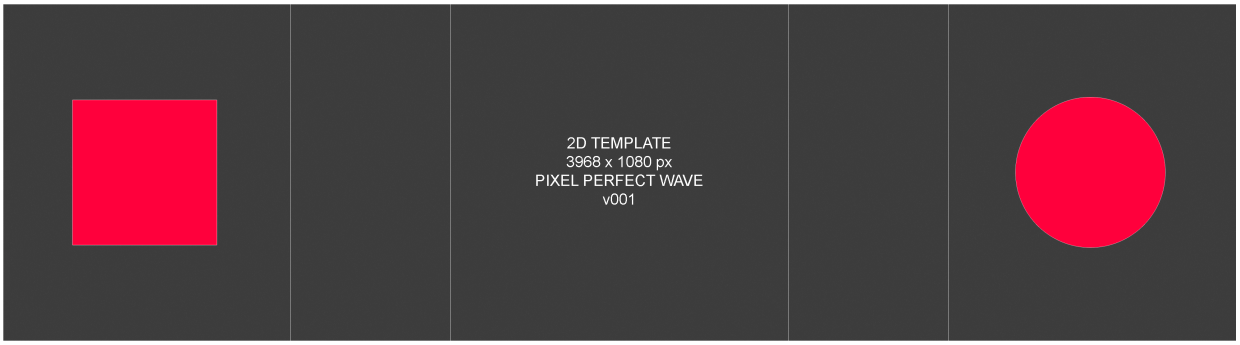


A pixel-perfect UV mapped surface in 3ds Max.

Step 2 - Rendering UV maps to content templates

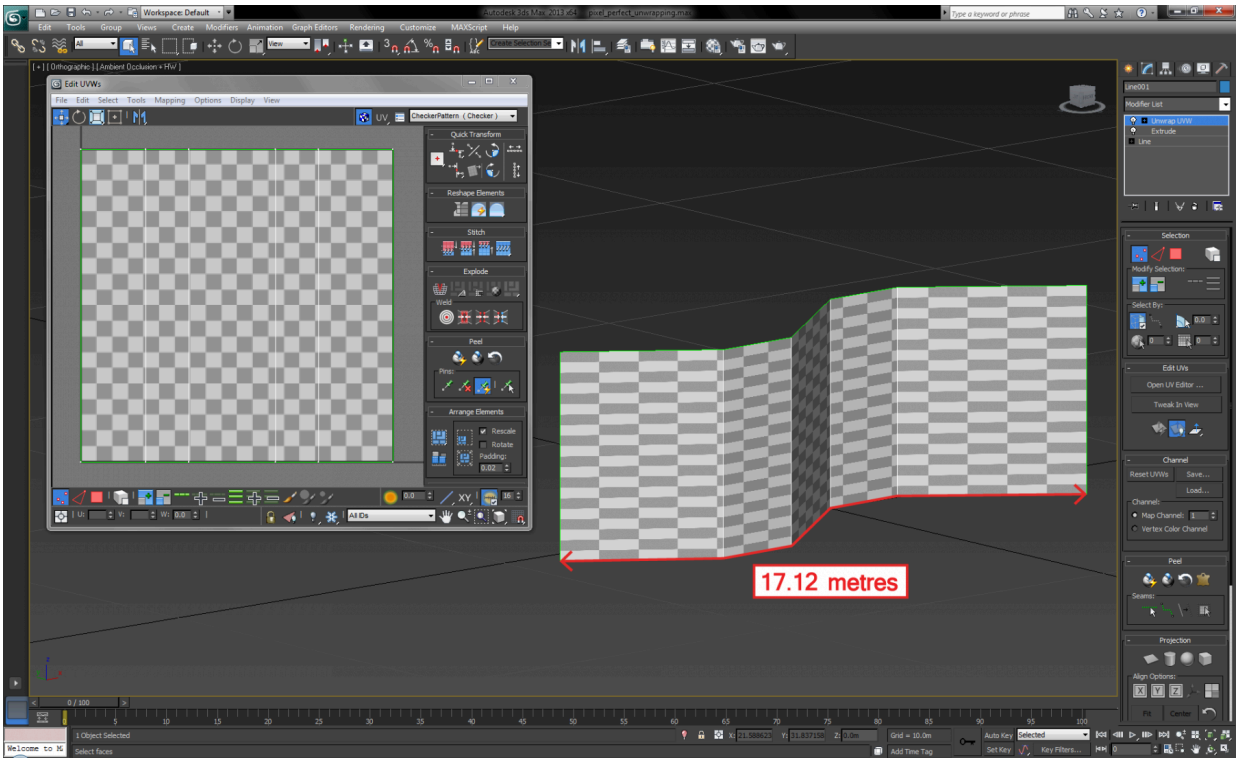
The image below shows the content template, which was rendered from the UV map. The template's resolution was calculated by:

- Knowing precisely how many pixels produced by the projector will hit the projection surface, across **either** the horizontal or vertical axis.
- Dividing this value by the projection surface's dimension across the chosen axis, to calculate a **pixel density**.
- Multiplying the pixel density by the remaining axis' dimension, to calculate its resolution.



UV map rendered to a content template

It is important to measure the projection surface's dimensions based on the UV map. In this case, the total length of the extruded spline should be measured, because the extrusion automatically generated the normalised UV map



By calculating both the template's horizontal and vertical resolutions from a single pixel density value, the content should enable square pixels. This is important because if the resolution is calculated incorrectly the content may appear stretched across the projection surface. The template's resolution is 3698x1080 pixels. This was calculated by:

- Knowing approximately 1080 pixels will hit the projection surface vertically.
- 1080 pixels / 5 meters (the projection surface's height) = 216 pixels per meter (hitting the projection surface vertically).
- 216 pixels x 17.12 meters (the projection surface's length) = 3698 pixels (the horizontal resolution).

As a result, the aspect of the template's resolution should match that of the surface's dimensions. This can be checked by:

- Dividing the template's horizontal resolution by the vertical resolution.
- Dividing the surface's length by the physical height (remember to base the measurements on the UV map, explained earlier).

In this case:

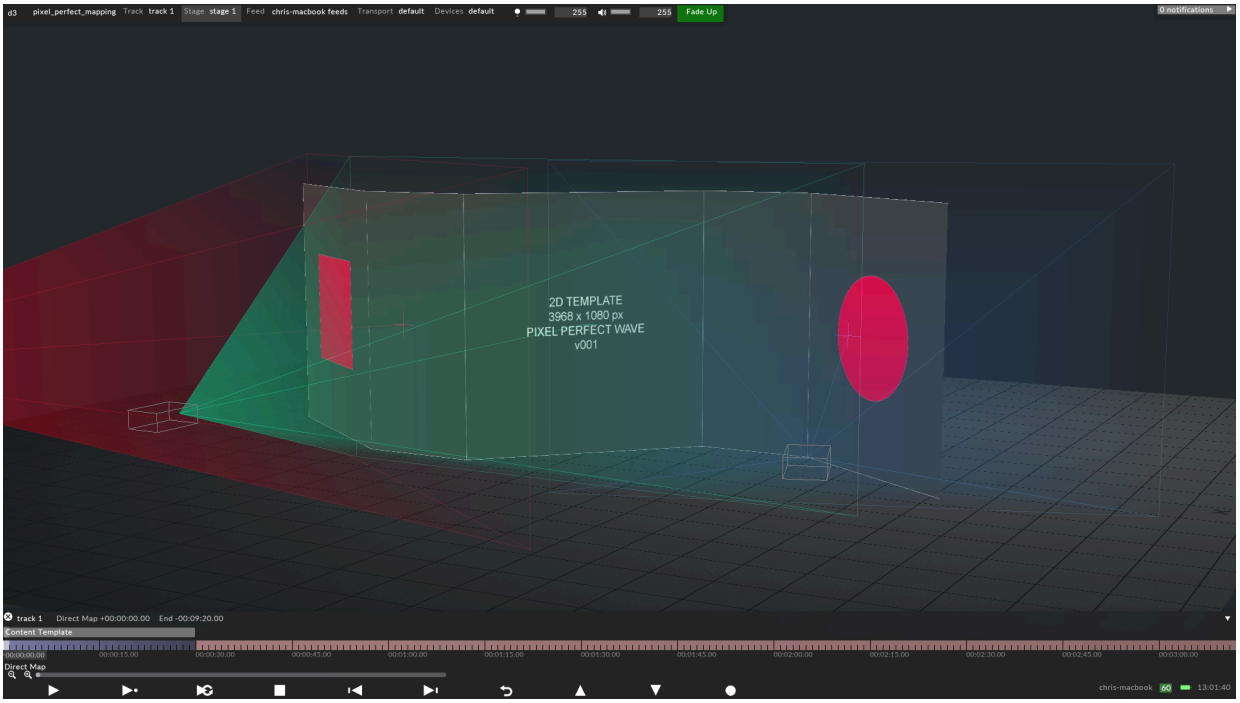
- 3698 / 1080 = 3.424
- 17.2 / 5 = 3.424

Both values are matching, which indicates the template's resolution is correct. To double-check this, a uniform square and circle should be drawn on the template in an image editing application, for example ***Adobe Illustrator***, before the template is exported to . If the square and circle appear uniform across the screen in the template's resolution is correct.

Step 3 - Mapping content to screens

The image below shows the content template directly mapped to the screen in Disguise, which was exported as an `.obj` file from 3ds Max. n Designer three HD virtual projectors have been configured to sample content from the screen. Each virtual projector samples a different part of the screen, and in realtime the sampled content is outputted to the physical projectors by three unique feeds.

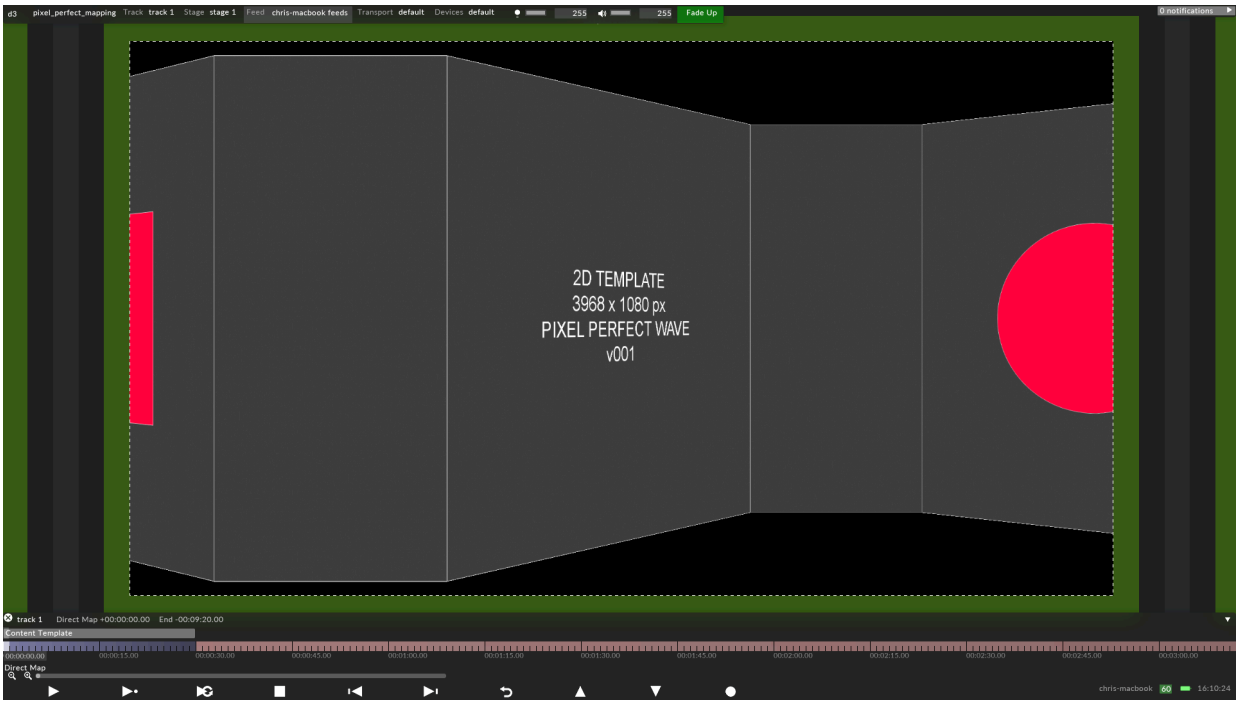
Remember to set the resolution of the screen in to match the content template by opening the **Screen editor**.



Template re-applied on the screen in Disguise.

Step 4 - Outputting content to projectors

The image below shows the three unique feeds to be outputted to the physical projectors. If a virtual projector changes position its corresponding output will automatically update in realtime to show the virtual projector's updated point of view. Therefore, if a virtual projector's position changes onsite, content does not require re-rendering, because the outputs are independent of the content. To read more about Disguise's projector simulation toolkit see the **Projector simulation** chapter, which explains how to perform quick calibration, **warp**, and **blend** projectors using Disguise.



The automatically generated outputs from the three virtual projectors.

Helpful resources

- Drawing splines in 3ds Max
- Extruding splines in 3ds Max
- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Rendering content templates in 3ds Max

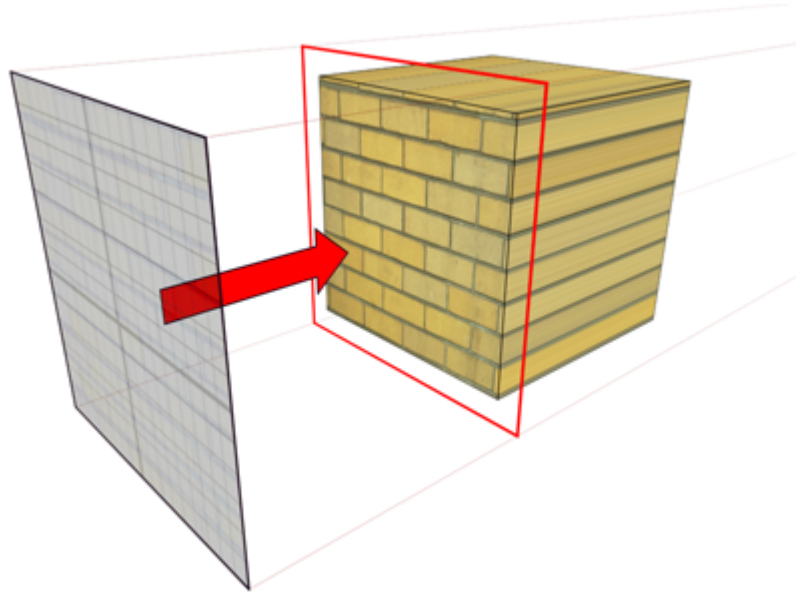
Projection Examples - Planar mapping

Planar mapping is suitable for projection surfaces that have:

- flat surfaces requiring one side to be UV mapped
- shallow surfaces requiring multiple oblique sides to be UV mapped
- for example a plane or a facade

How the UV map is generated

The Planar map will parallel project UV coordinates to the surface. Therefore, it is important to align the Planar map's position, orientation and scale to the surface.

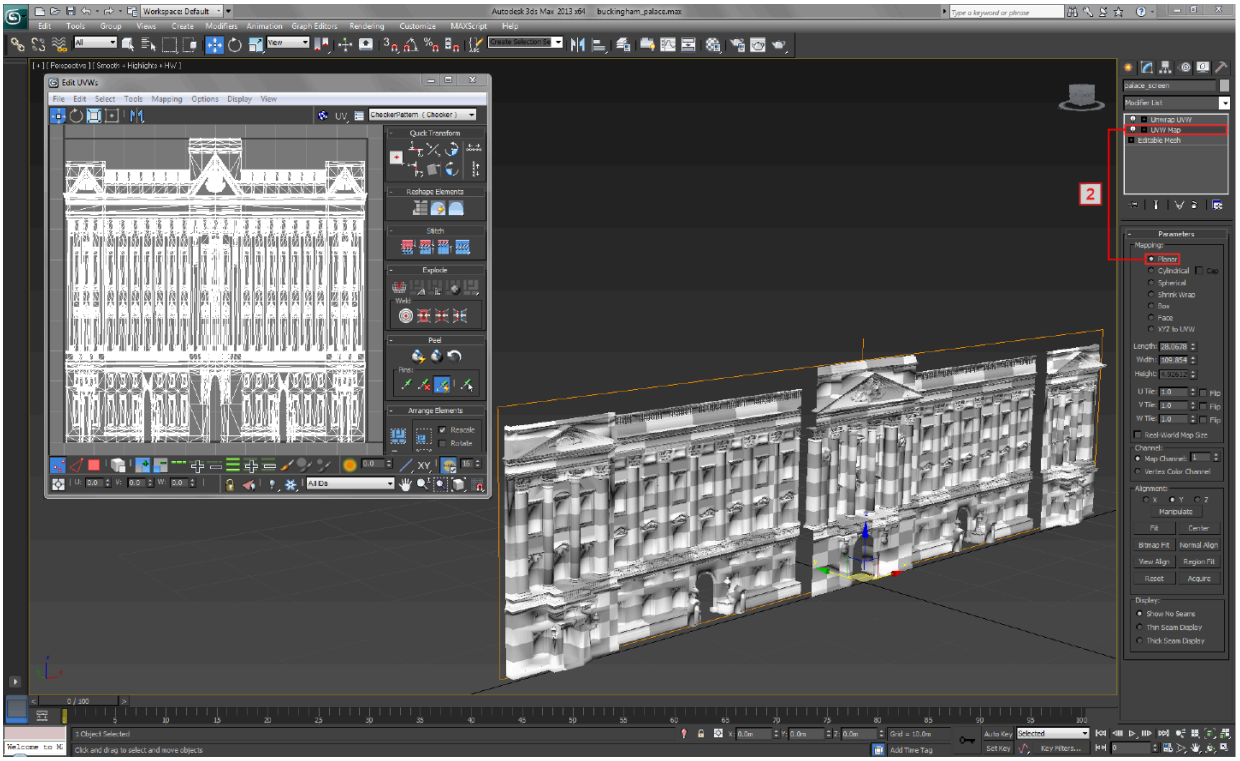


Step 1 - Generating the UV map (3ds Max)

1. Select the UVW Map modifier.
2. From the list of projection types select planar.
3. The Planar map will project UV coordinates from its normals to the surface. Therefore, it is necessary to fit the planar map's position, orientation and scale to the surface.
4. Select the Unwrap UVW modifier.
5. Open the Edit UVWs window to view the UV map.

The image here shows a uniform checker pattern rendered to the surface, which indicates the UV map has an even distribution of UV coordinates, for all areas facing towards the Planar

map. As a result, those areas have a higher UV density than the areas facingaway from the Planar map. For example, the windows' side faces have a lower UV density, which will result in content appearing stretched. To avoid this the UV map should be **relaxed**.



A 3D model of the Buckingham Palace planar mapped in 3ds Max.

Step 2 - Rendering UV maps to content templates

The image below shows the content template, which was rendered from the UV map. The template's resolution was calculated by following the same technique used for pixel-perfect projection surfaces. However, the key difference is this template was calculated by measuring the dimensions of the Planar map object, because in this case the Planar map automatically generated the normalised UV map.





A 2D content template of the Buckingham Palace rendered from 3ds Max.

Helpful Resources

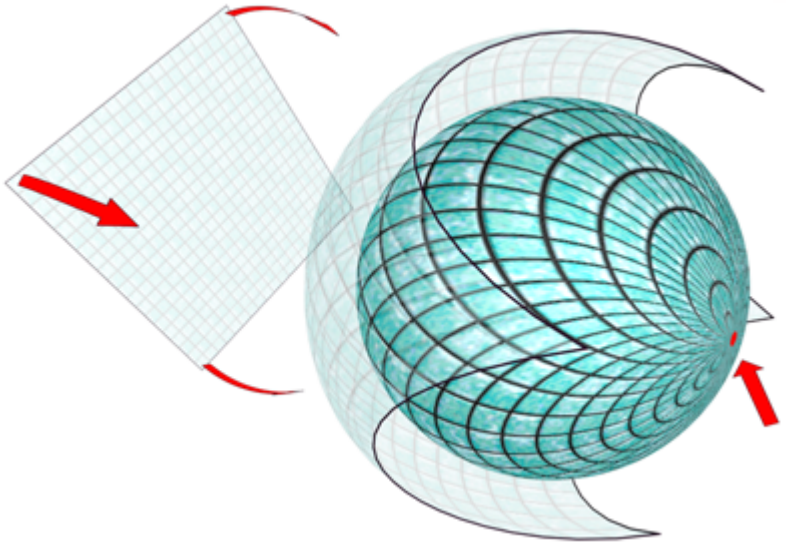
- Projecting planar maps in 3ds Max
- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Relaxing UVs in 3ds Max
- Rendering content templates in 3ds Max

Projection Examples - Shrink wrapping

Shrink wrapping is suitable for projection surfaces curving across multiple axes, for example a sphere or a car. However, depending on the creative requirements, a Pelt map may be more suitable.

How the UV map is generated

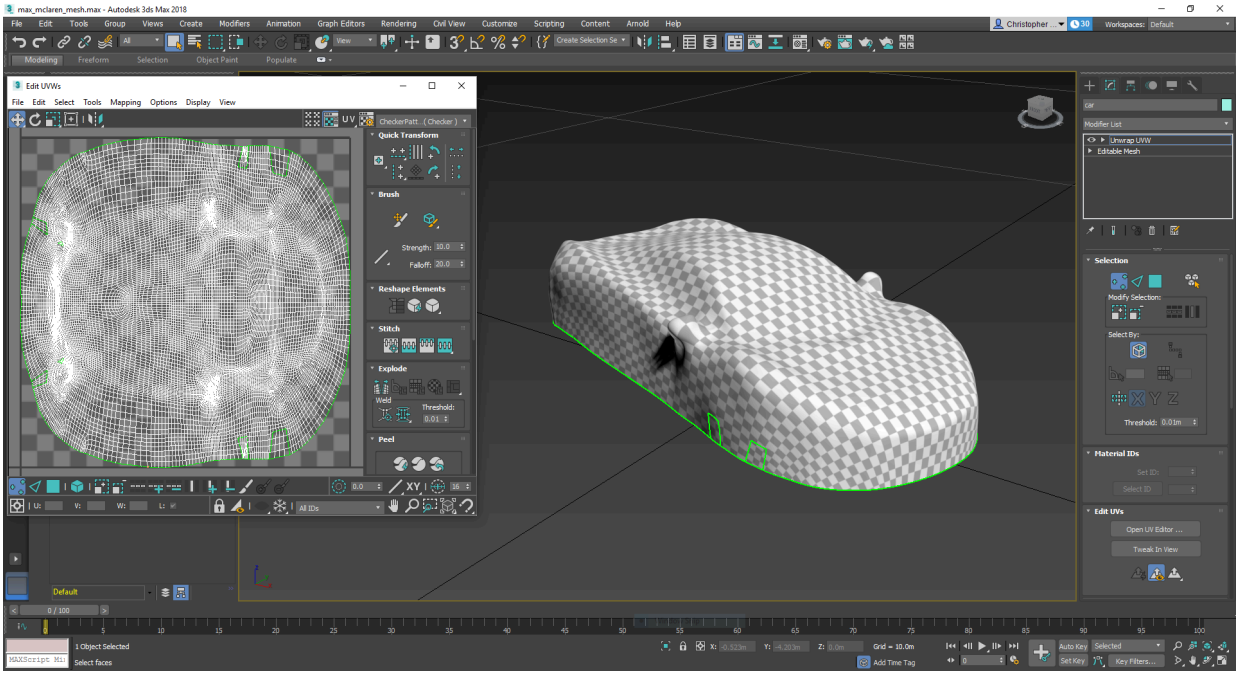
The Shrink wrap will spherically project UV coordinates to the surface, by truncating the corners of the map and joining them at a single pole. This will enable content to flow across the surface smoothly, because the UV map will only contain one singularity.



Step 1 - Generating the UV map (3ds Max)

1. Select the UVW Map modifier.
2. From the list of projection types select shrink wrap.
3. The shrink wrap will project UV coordinates from its normals to the surface. Therefore, it is necessary to fit the shrink wrap's position, orientation and scale to the surface.
4. Select the **Unwrap UVW modifier**.
5. Open the Edit UVWs window to view the UV map.

The image below shows a non-uniform checker pattern rendered to the surface, which indicates the UV map has an uneven distribution of UV coordinates. To fully unwrap the surface the UV map should be relaxed.



A 3D model of the Audi A7 mesh shrink wrapped in 3ds Max.

Step 2 - Rendering UV maps to content templates

The images below show the content template and how its resolution was calculated. When a projection surface is shrink wrapped, it is problematic to calculate the resolution mathematically. However, it is possible to calculate the resolution visually. The template's resolution is 3448x1424 pixels.This was calculated following the steps below:

1. Export the projection surface as an .obj from the 3D application to Designer. Remember to export the .obj's texture coordinates, as explained in the Exporting UV maps page.
2. In Disguise, add the .obj to the Stage as a Screen.
3. Create a **Bitmap** layer on the Timeline.
4. From the layer's DxTexture library, select a texture with a square aspect.
5. **Direct** map the texture to the screen.
6. Set the layer's scale mode to pixel-perfect.
7. From the **screen's editor**, set a resolution for the horizontal or vertical axis that matches or preferably exceeds, the projector's pixel density. This requires knowing precisely how many pixels produced by the projector will hit the projection surface.
8. Increase / decrease the screen's resolution across the remaining axis until the texture appears square. This will ensure square pixels are enabled.
9. Offset the texture using the pos x and y parameters, to check square pixels are visible across the entire surface.



A square Bitmap Direct mapped to the car screen in Disguise.



A content template rendered from the Audi A7 UV map with a paint texture baked into the shrink wrap.

Helpful Resources

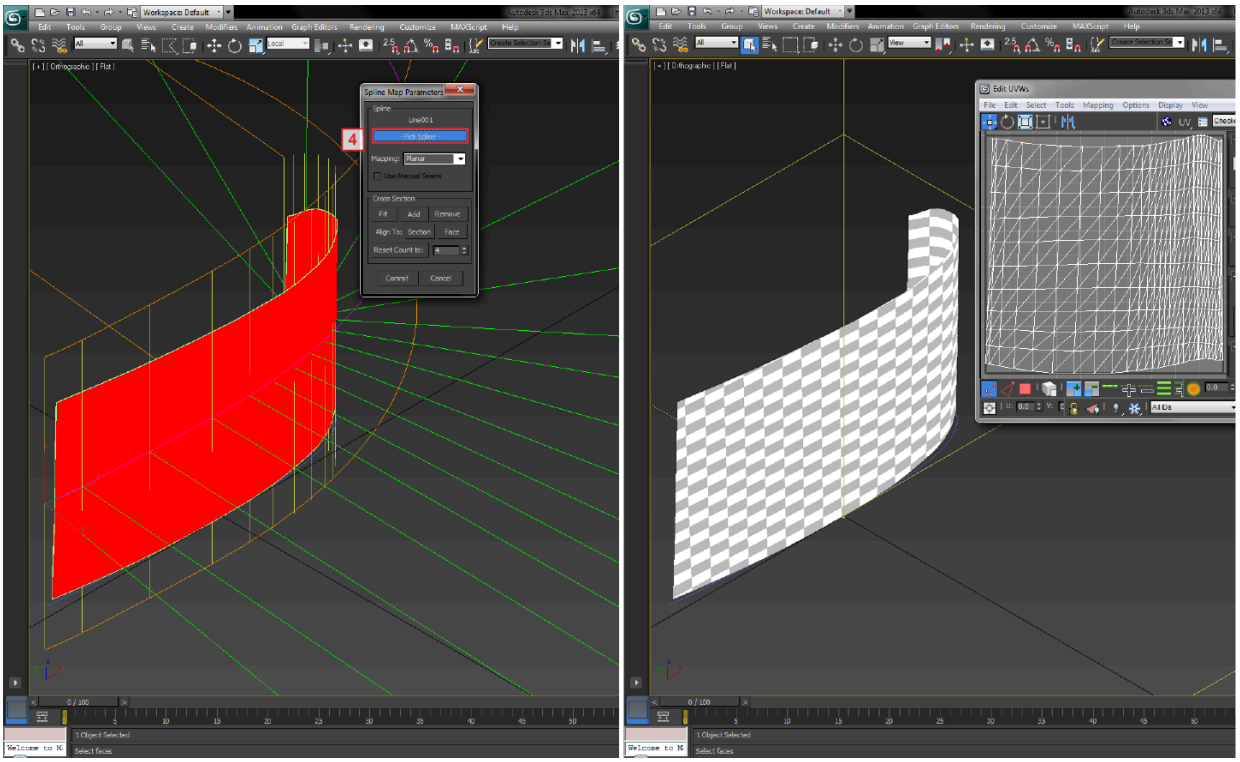
- Projecting shrink wraps in 3ds Max
- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Flattening UVs in 3ds Max
- Unfolding UVs in 3ds Max
- Rendering content templates in 3ds Max

Projection Examples - Spline mapping

Spline mapping is suitable for projection surfaces curving across multiple axes, for example a coiled snake or a winding road.

How the UV map is generated

The Spline map will extrude UV coordinates from a selected spline to be projected to the surface. This will enable content to flow across the surface smoothly, because the UV coordinates will match the surface's curvature.



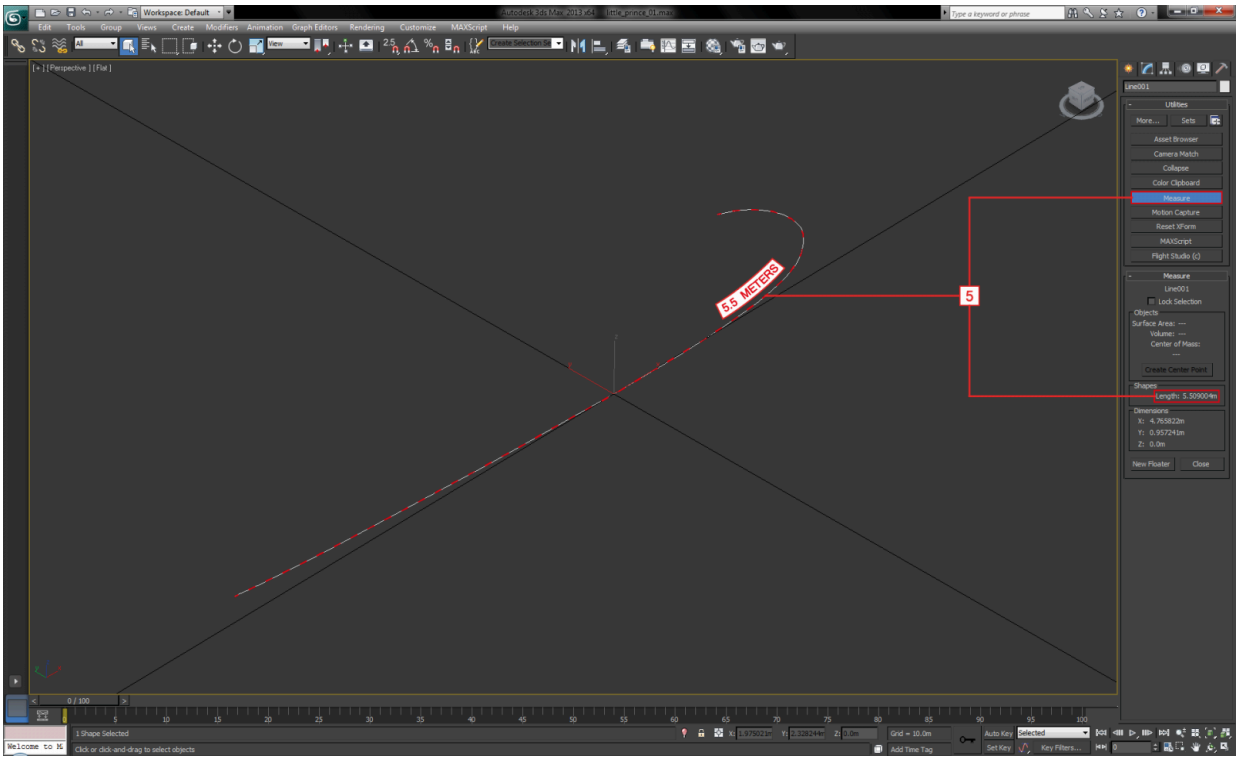
Step 1 - Generating the UV map

The following steps are instructions for generating UVs in 3ds Max.

1. Draw a spline that snaps to the surface's bottom edge.
2. Plot all the vertices in the same vertical position to ensure the UV projection is orientated correctly.
3. Plot the spline across the surface's widest points to ensure the UV projection is assigned to the entire surface.
4. Select Unwrap UVW modifier.

5. Select all of the surface's faces.
6. From the list of projection types select **spline mapping**.
7. From the spline map's parameters select pick spline.
8. Open the **Edit UVWs** window to view the UV map.
9. Rotate and normalise the UV map, if necessary.

The image here shows a uniform checker pattern rendered to the surface, which indicates the UV map has an even distribution of UV coordinates.

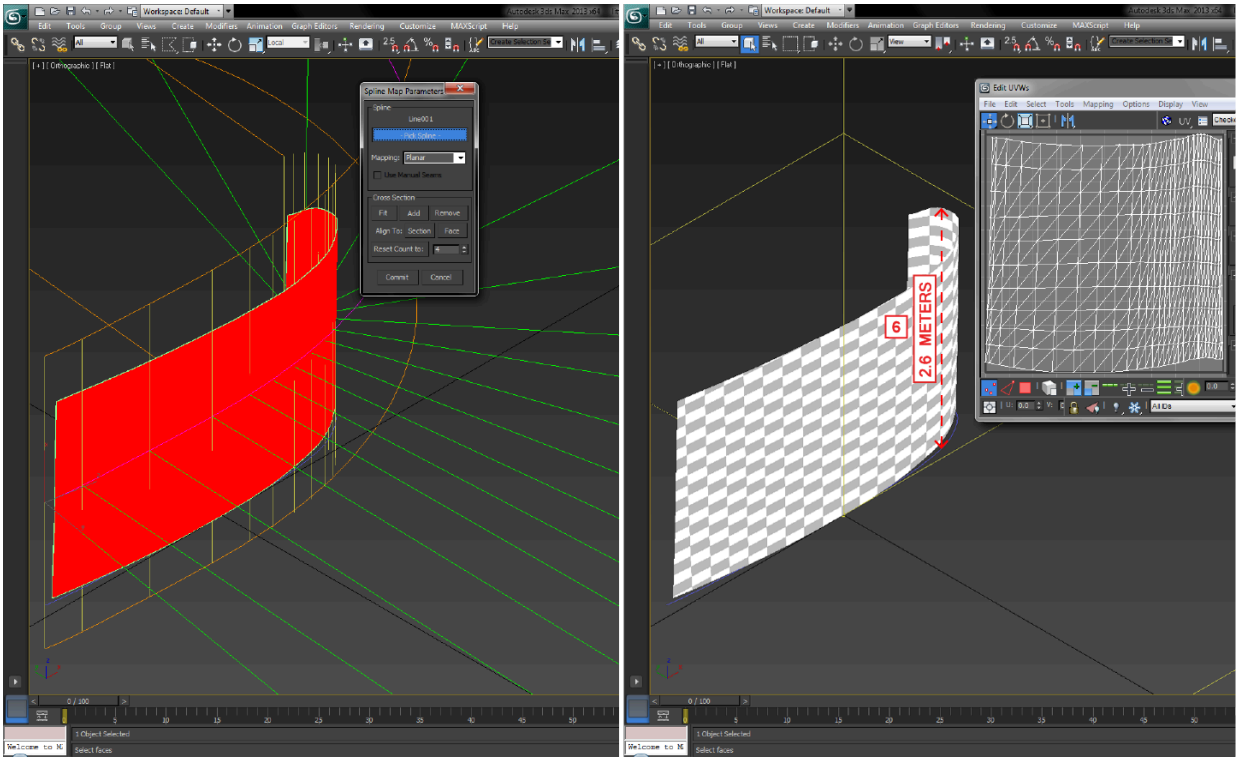


A 3D model of a screen from the Little Prince musical spline mapped in 3ds Max.

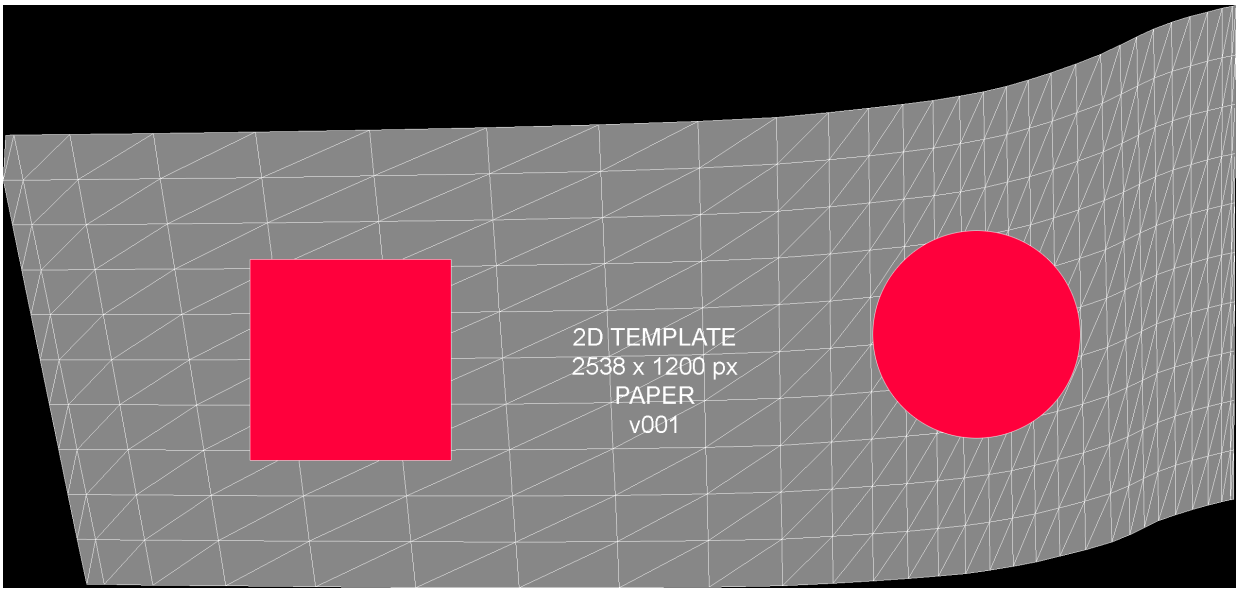
Step 2 - Rendering UV maps to content templates

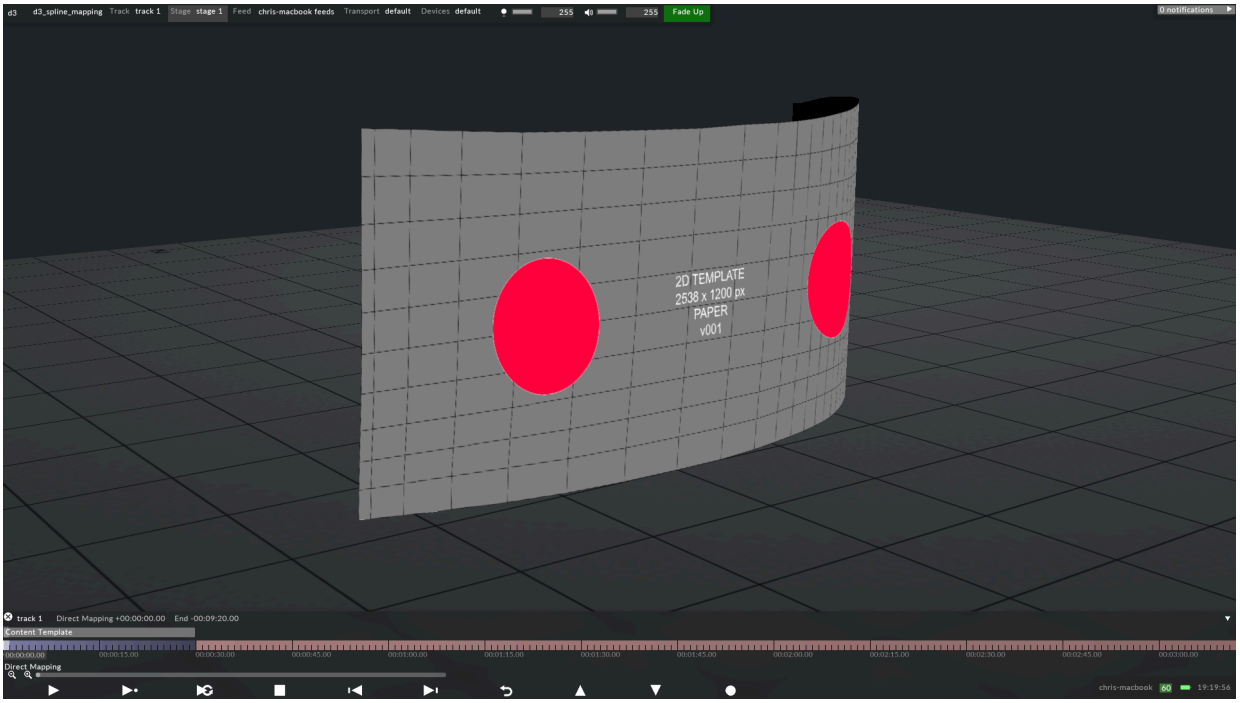
The images below show the content template and how its resolution was calculated. The template's resolution was calculated by following the same technique used for pixel-perfect projection surfaces. However, the key difference is this template was calculated by measuring the total length of the extruded spline, because in this case the extrusion automatically generated the UV map.

The 3D application should calculate the total length of the spline automatically. In 3ds Max select the spline, **Command Panel > Utilities > Measure**.



The length of the spline can be measured by selecting it and then selecting Utilities > Measure.





A 2D content template of a screen from the Little Prince musical rendered from 3ds Max.

Helpful Resources

- Projecting spline maps in 3ds Max
- Unwrapping UVs in 3ds Max
- Editing UVs in 3ds Max
- Flattening UVs in 3ds Max
- Unfolding UVs in 3ds Max
- Rendering content templates in 3ds Max

Virtual Reality

Virtual Reality visualiser is an OpenVR / Steam compatible visualiser mode allowing the user to enter and navigate through the scene.

VR was designed to primarily work with Designer machines, utilising GPU's such as GTX 1060 & higher (or other manufacturer equivalent).

Supported headsets

We have tested HTC and Meta headsets on Designer, which are known to work. We have also used Apple Vision Pro with Designer, through the ALVR system.

> **Note**
>
> **Any headset that supports Steam VR can be used with Disguise.**
>
> Please check the manufacturer's specifications of the headset type you would like to use to confirm if it supports Steam VR. Some headsets, such as **Apple Vision Pro**, support this via third-party plugins. For example, for **Apple Vision Pro**, this functionality is provided via the **ALVR** application, which must be installed on the host machine alongside SteamVR *and* on the Vision Pro (via the App Store) to enable VR previsualisation.

HTC Vive

- The machine needs to be restarted after the first installation, otherwise, the headset will not be detected
- All EDID's need to be de-emulated otherwise the headset will not work
- The box needs to be plugged into a USB 3 port.

You need to have the SteamVR system running; you should be able to get to the point where the headset and one controller are reporting green and working, and the headset should show the Steam basic scene. Once that's working, if steam is installed, and the VR equipment is connected - when starting a Disguise project, you will automatically be placed into VR mode.

- The visualiser shows you the position and orientation of the headset in stage space (small white cuboid about the same size as a headset, with an orange line down to the floor). It also shows you where the controller is pointing (white line when it hits the floor level, blue line otherwise).

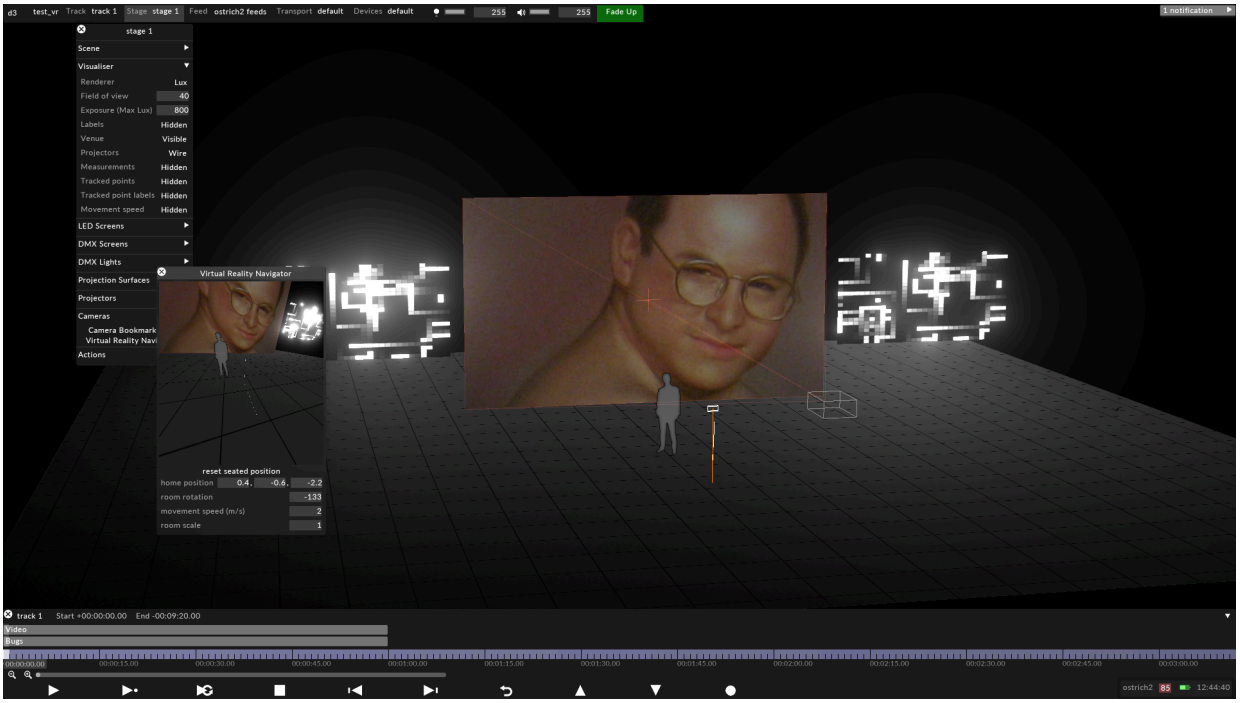
- Navigation - the VR user navigates by pointing the controller at the floor location they want to go to; it goes white; they click the button and they jump there. The property movement speed in the VR Navigator window can be set to zero (jump immediately) or some non-zero number, which moves the VR world in a straight line on a linear path. This gives you a little 'jerk' at start and end but doesn't seem to result in nausea. It's less confusing for new users, but may be annoying for experienced users.

The Virtual reality navigator also lets you set up the position and orientation of the scene relative to the physical world. The VR experience keeps this orientation constant to avoid users getting lost. They still get lost of course, and when they do you can hit the reset button to bring them back to the center.

Opening the virtual reality navigator

1. Open the stage editor by right-clicking **Stage** from the dashboard or by right-clicking the floor in the visualiser.
2. Select the **Cameras** tab.
3. Left-click Virtual reality navigator and then you can see what the VR user sees.

> **Warning**
>
> Navigation is restricted to floor level. We need to be able to move to the nearest horizontal surface on whatever venue / stage you're on.
>
> ⚠️ There's no **reverse** button; the workaround is just to point behind you and hit the button.



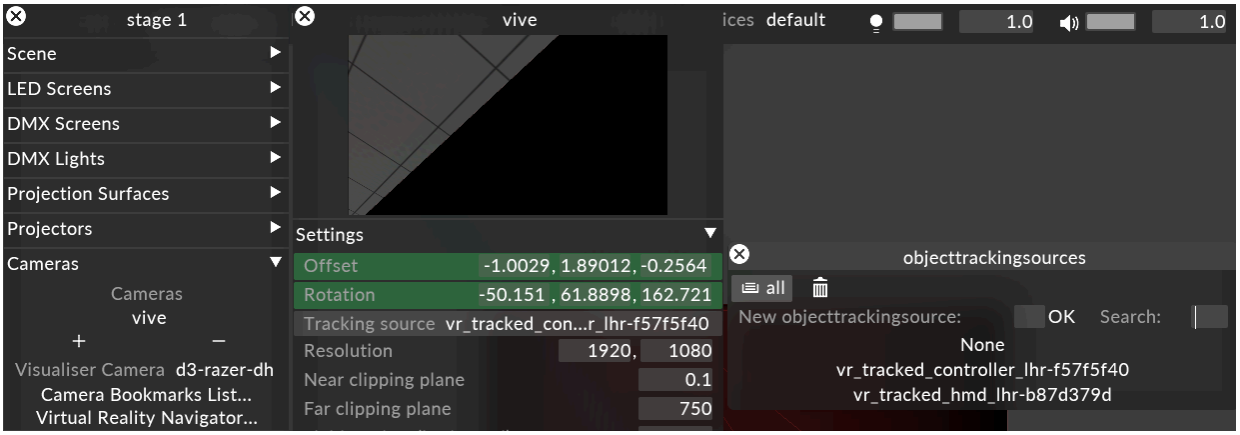
The image shows the basic elements of VR functionality.

Using a VR device for camera tracking

Cameras and other stage objects can be tracked by the headset or controller of a compatible VR headset

To assign a VR device as a tracking source for a camera

1. Connect your VR headset to your machine and ensure the device is working in SteamVR.
2. Launch your Disguise project (version r17.2 or later).
3. Create a new virtual camera in your project file by right clicking on the Stage menu.
4. Hit the + icon under the Cameras tab and input the name and specifications of your camera.
5. Open the camera's menu by right clicking on its name from the Stage menu, or on the icon of the object in the visualiser.
6. Under the first tab called Settings, right click on the Tracking source field.
7. The VR's headset and any active controllers will be listed here; left clicking on one will assign the position of the camera to it.
8. The Offset and Rotation fields of the virtual camera's menu will turn green when receiving a working signal.



To use an animated object as a moving VR platform

In some situations, the VR viewpoint may need be a part of a moving element, such as a ride or some vehicle. In this case, the VR floor position can be **parented** to an animated object.

1. Have the VR system set up and working in Designer, with your project loaded.
2. Create the animated object you want to parent the VR camera to.
3. Open the virtual reality navigator from the Stage menu.
4. In the VR navigator window, left-click on the “User parent” field.
5. From the list of objects, left-click on the animated object you want to parent the VR camera to.
6. The VR camera will now move along with the animated object.

To reset the VR camera back to the stage floor, left-click on the “User parent” field again and select “Stage” from the list.

xR Workflow Overview

This workflow guide is intended to give users an end to end example of how to get up & running with the Disguise xR workflow.

The Disguise xR workflow consists of the following key topics:

1. xR Project Setup
2. xR Stage Setup
3. MR Set
4. Video Receive Delay
5. Observations
6. Spatial Calibration
7. Spatial Tracker delay
8. Colour Calibration
9. Inverse Lut Method
10. API xR Calibration
11. Set Extension
12. Virtual Cameras
13. Camera Switching
14. Cluster Rendering
15. Frustum Workflows
16. Multi-frame Remapping
17. Depth of Field
18. xR for VR Workflow

xR Project Setup

This topic will explain the basic principles of setting up a project for use with the xR workflow.

1. Create a new **Disguise project** if required.
2. Configure your **Director/Actor setup** if required.
3. Create an accurate representation of your **LED setup** in the stage.
4. Configure your cameras.
5. Configure **Genlock**
6. Ensure project refresh rate and output refresh rates match.
7. Ensure feed **output resolutions** are configured.
8. Perform a spatial calibration as required.
9. Configure **RenderStream** content if required.

xR Stage Setup

This topic covers setting up your virtual stage and cameras in Designer for xR calibration of an LED stage.

Workflow

1. Add LED screens to the stage. For the xR workflow, LIDAR scanned and UV unwrapped OBJ meshes are best.
2. Add virtual cameras to the stage.
3. Connect the video output of the physical stage camera to a video input on the Disguise server.
4. Check and configure physical camera settings, including white balance, framerate, and genlock status.
5. Patch the video input(s) to the virtual cameras in the stage in the Video Input Patch Editor.
6. Run preview in the video input patch editor to ensure the camera feed is being received at the correct signal.
7. Check & configure camera tracking system.
8. Create a position receiver to link to the camera tracking system.
9. Add tracking drivers dependent on your camera tracking system to the position receiver.
10. Engage the camera tracking drivers.
11. Assign the driver in Designer to a virtual camera.
12. Monitor the incoming data from the camera tracking system to ensure it is being received in time.
13. Create an MR set.
14. Optionally, add a set extension mesh.
15. Add LED screens to the MR set.
16. Add camera to the MR set if required.
17. If using multiple cameras, configure and test the indirection controller.

Further Reading

- Creating Screens
- Virtual Cameras
- Video Input
- Automation - Creating a new Motion Controller
- MR Set

- Indirections

MR Set

An MR set contains all the components of your virtual set and the camera(s) you will be using to capture that virtual set for transmission.

> **Warning**
>
> **Warning**: xR will not work properly if the configuration is not completed properly. For more information on the configuring xR, please visit the xR workflow page.

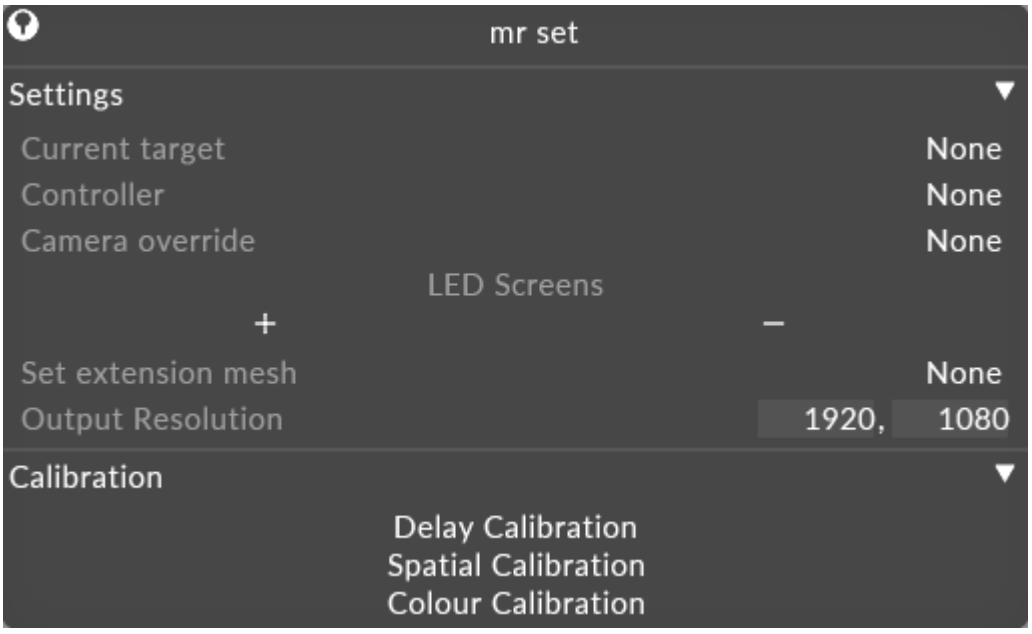
Workflow

To create an MR set

1. Open the Stage properties widget and navigate to the MR Set tab. It can be useful to pin this widget open.
2. Add LED screens to the MR set.
3. Add an Indirection Controller (if needed).

Properties

Settings



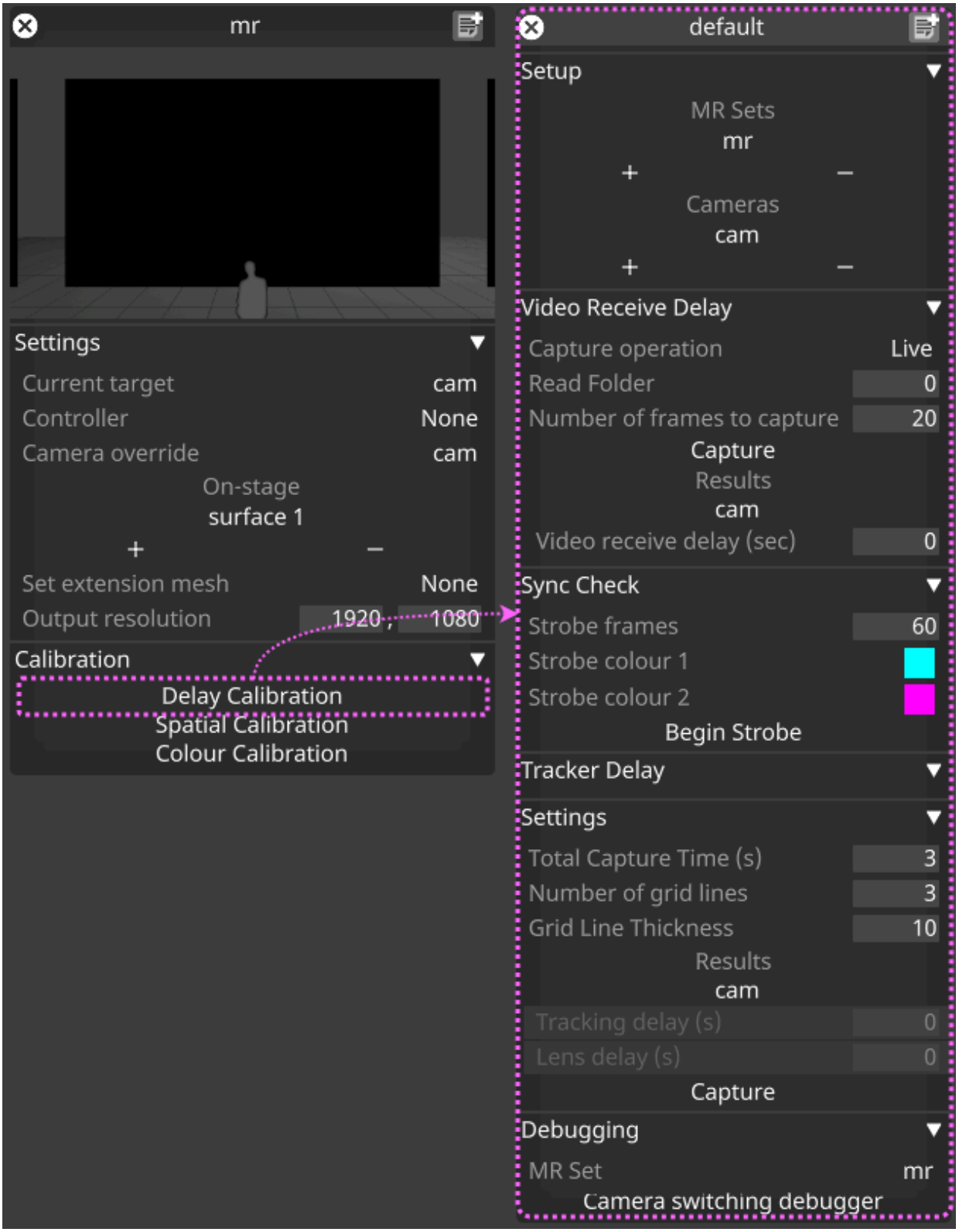
- `Current target` - Defines the camera that the content is being rendered from.

- `Controller` - Defines the indirection controller that controls the camera switching.
- `Camera override` - Allows you to set a camera override for whatever the current controller is set to.
- `LED Screens` - Defines which LED screens are included in the MR Set.
- `Set extension mesh` - Defines the mesh used for the set extension. This property reads from the mesh folder of your project folder.
- `Output resolution` - Defines the final output resolution of the MR Set.

Calibration

Delay calibration

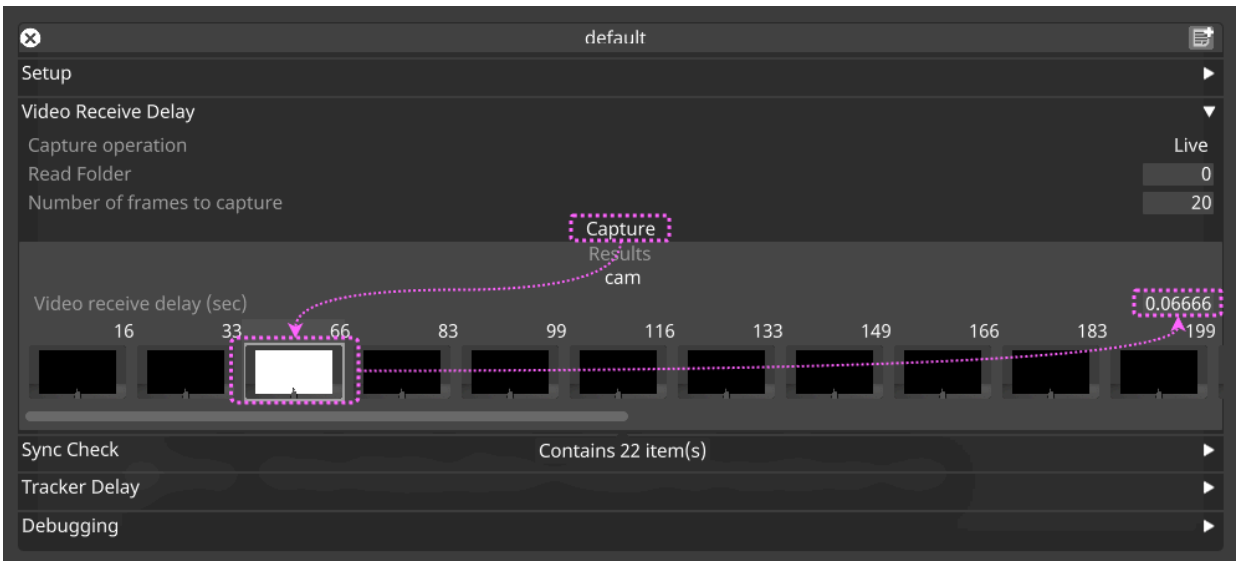
Left-click this button to open the calibrate displays editor.



Setup

- `MR Set` - defines which MR Sets are used.
- `Cameras` - defines which cameras are included.

Video Receive Delay



Workflow

1. Click **Capture**.
2. The LED wall displays one white frame among black frames.
3. The captured frames appear below the **Video receive delay (sec)** field.
4. Click the white frame.
5. The **Video receive delay (sec)** value updates accordingly.
    - This is the capture delay of the camera.

Capture operation. There are three options

- `Live` - The live setting will use the live captured images to do the calibration.
- `Write` - The write setting will write the images captured to disk so they can be used for debugging later.
- `Read` - The read setting will read back the images you previously wrote to disc, this is useful for debugging.

Sync check

- `Strobe frames` - defines the FPS for the sync test.
- `Strobe colour 1` - Defines the colour for part one of the strobe alternation.
- `Strobe colour 2` - Defines the colour for part two of the strobe alternation.

Tracker Delay

Settings

- `Total capture time` - Defined in seconds. This is the total capture time of the calibration.

- `Number of grid lines` - Defines the number of grid lines.
- `Grid line thickness` - Defines the thickness of the grid lines. Increase or decrease thickness based on specific stage setup.
- `Results` - Left-click Results to open the results dialog.
- `Capture` - Left click capture to run a calibration.

Debugging

- `MR set` - Defines the MR Set to debug.
- `Camera switching debugger` - Left click to open the camera switching debugger.

Spatial calibration

Left-click this button to start a spatial calibration.

Video Receive Delay

Calibrating delays is an important part of the xR workflow. Failure to complete this configuration correctly will result in the set extension and LED screen output not being in sync.

There are two user calibrated delays in every system that must be determined.

The first is the amount of time between the Disguise hardware rendering a frame and receiving the corresponding video frame. This is called the Video Receive Delay.

The second is the time between when the tracking system calculates a camera's position and when it is received and processed by Designer. This is called the Tracker Delay.

MR is a game of latency - all components of the system are moving independently, and the preservation of a unified image requires compensating for distortions after an image is taken. These delay values allow Disguise to catch up to the camera sensors and tracking sources.

Workflow

Setting the Video Receive Delay will allow the camera to capture the blobs that are outputting to the screen correctly. If the delay is too high or low, the correct frame will not be captured, which could mean that the structured light pattern is not recognised.

To set the video receive delay

1. Open the MR set editor.
2. Navigate to the calibration tab.
3. Add MR set to the calibrate delay tool.
4. Click **Delay Calibration**.

> The calibrate delay function is pre-populated with the current camera target, but you can add or remove different combinations of cameras.

5. Click **Capture**.

> Designer will output one frame of white to all outputs assigned to the MR set.

6. Look at the captures:

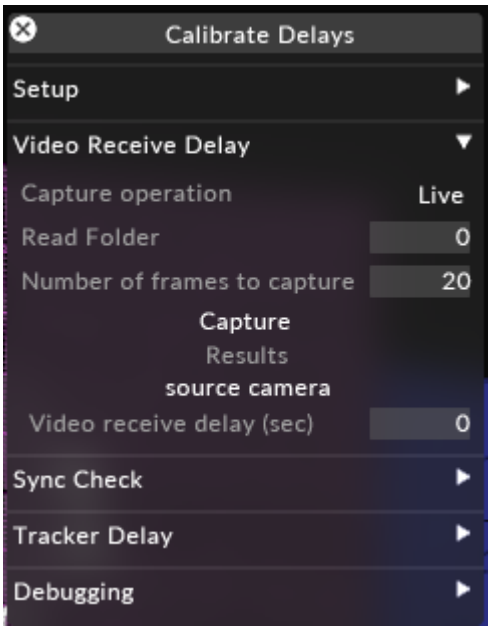
> - If the white frame is splitting across two frames, you may need to adjust the phase offset of the LED processors.
> - If the white frame is fully present in multiple frames, adjust the shutter angle of the camera. This will close the camera shutter faster resulting in only one frame of white to be captured.

7. When a single white frame has been captured, left-click the white frame; the delay value will autopopulate in the editor.
8. Test your screens are in sync:

> At this point you should check that your screens are in sync with each other; you can use the Sync Check tool found in the calibrate delay widget to accomplish this.

9. Expand the Sync Check tab to ensure that the set extension and LED screens flash at the same frame.
10. Begin Strobe and view on the xR transmission output if the set extension and LED screens seem in or out of sync.

Properties



Capture operation

Determines if the captured images during testing are archived to the debug folder of the Windows project file structure, or captured and assessed Live.

Number of frames to capture

The user defined number of frames to be captured for the capture to be registered.

Capture

Clicking will capture the set number of frames on output. Video Receive Delay (sec): The delay value of the incoming video frame that will be autopopulated by clicking on the correct white frame.

Sync Check

The tool used to ensure the Video Receive delay value is correct.

Observations

An observation is a set of images of the stage captured by the camera of white dots on a black background (called structured light.) The number of dots and their size/spacing is determined by the user.

Observations are used as data points within the spatial calibrator, which is the predefined algorithm Disguise will use to align the tracked camera with the stage and set up the lens characteristics.

Observations are split between primary and secondary. Primary observations let Disguise determine the position of the camera and the position of the screens which make up the MR Set. We also determine the lens characteristics in that primary configuration.

Secondary observations contribute to Disguise's understanding of variable zoom and focus settings on the camera lens.

Adding observations is a cumulative process, and each "primary" observation will equally affect the overall calibration results. This can mean that one bad observation will spoil the lot, but that is something we want to hear about to fix it. It is critical to review each observation after it has been taken to determine if it will improve or worsen the overall primary calibration.

Workflow

1. Check camera tracking data.
2. Set your first camera position.
3. Select the target camera you wish to calibrate.
4. Select spatial calibration.
5. Use live blob preview.
6. Set your first zoom position.
7. Focus the camera.
8. Lock zoom & focus values (optional).
9. Set adjust screen position to on or off in the calibration widget.
10. Add your first observation.
11. Troubleshoot any issues with the observation.
12. Delete observations (if necessary).
13. Reset observations (if necessary).
14. Add more observations from new positions.
15. Repeat process for secondary observations.
16. Test zoom interpolations.

Example

Check camera tracking data

1. Check that the camera tracking system can see its location dots, and that it is outputting good data.
2. Check that zoom and focus encoder information is reaching Disguise. Wiggle the encoders on the camera and see the change in the automation monitor.
3. Check that you are receiving the correct movement in each axis: move the camera along the x y and z axes and pan, tilt and roll (where possible) making sure that the movement of the virtual camera in Designer matches the real world movement.

Set your first camera position

1. Choose where you want to take your first observation from on the floor. Pick a position and lock the camera off.

> Try to remove as much negative space from the frame and fill the monitor with as much of the LED as possible.

2. Select The Target Camera You Want To Calibrate.
3. Select Spatial Calibration in the MR set.
4. Use the Live Blob Preview tool to visually ensure the blobs are on the stage.
5. Set your first zoom position by zooming the camera out to the first level of zoom you need.
6. Focus the camera.
7. Set whether the calibration should set both the camera and LED screen positions (on by default), or just camera position (off by default). This can be done in the calibration editor.

> For stages that comprise of multiple screens, the screen at the top of the list in the mr set will be the main screen, and will remain unchanged while the other screens and cameras move around it. If you monitor the MR Set on your monitor, you should see the virtual line up layer pop into place and align with the test pattern the LED is outputting.

> **Note**
>
> You may see the edges of the screen looking distorted, this is due to lens distortion. You can add overscan in the camera widget to see 100 % of the virtual line up layer alignment with the test pattern.

8. Click Add Observation.

Designer will display a blob pattern on stage and capture the blobs as they appear. When the observation is finished, it will appear in the observation list.

Add more observations from the first position

As you add more observations they will appear in the observation list; Disguise automatically calibrates all the observation in a list.

- For A/B testing the effect an observation has on a calibration you can mute or delete observations from the observation list.
- After taking your first observation then take additional observation from that same location pointing the camera at different places on the set.
- Try panning left, right and tracking up, down to cover all the pixels you will use in your show.

Add more observations from new positions

- Now move your camera to other positions and start at 0 level of zoom and 0 focus, try to cover all the pixels from other angles.
- Monitor your MR Set Output notice when the virtual line up layer appears out of place, take observations at these points.

Secondary observations

- With our primary observation calibrated, we now want to repeat the above steps at the next zoom and focus level you require for your show.
- Disguise will calculate the interpolation between zoom and focus levels automatically.
- Repeat these step for as many zoom and focus level you require.
- Check after each observation that the zoom and focus values match those of other observations taken at the same zoom / focus level. If they do not, edit the zoom / focus value to match.

> **Note**
>
> **Please note**: A common issue is lens encoder jitter. This means that while optically nothing has changed in the lens, the encoders may have ‘jittered’ very slightly, giving a marginally different value. This different encoder value will result in another pose, despite the lens not having changed zoom or focus. This can adversely affect camera registration.

Testing zoom interpolations

To test if Designer is interpolating between different zoom positions, zoom in and out on the camera and watch the MR Set. If Designer is interpolating correctly, the virtual line up layer should zoom in with the camera and stick to the test pattern on the stage.

Troubleshooting an observation

Understanding why an observation is bad can be achieved with the observation debugger.

The Current Observation parameter allows you to cycle through the captured observations. Different colour squares are overlaid on the blobs which help Designer differentiate which screen the blob was seen on.

You can rectify what blobs Designer is able to see by increasing or decreasing the dot size and grid spacing parameters in your spatial calibration settings.

Users can change the **Tracker Distortion Compensation** to **Matrix**. This will attempt to resolve the calibration using another method.

Deleting an observation

Deleting an observation will remove it from the observation list.

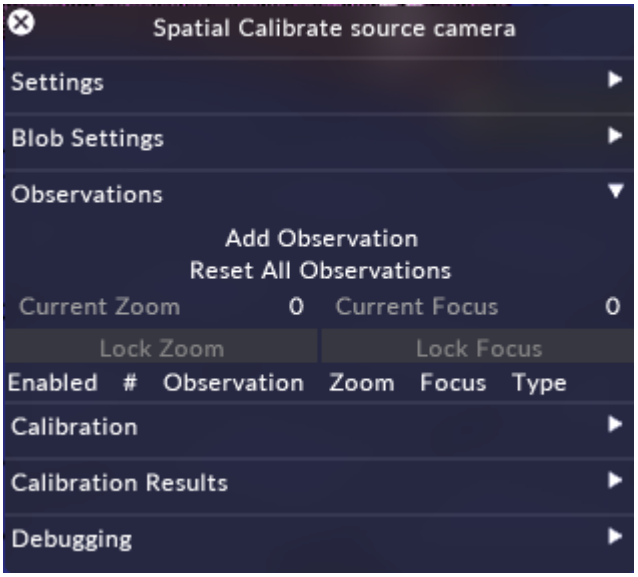
To delete an observation

1. Left-click the trash icon next to the observation to delete it.

Resetting all observations

It is possible to reset all observations, rather than deleting them one by one. **Reset all Observations** will completely reset the spatial calibration of the camera. Confirmation is required when clicking this button.

Properties



Spatial Calibration

Here we will cover the basic steps of aligning the physical and virtual worlds within Designer for xR and VP workflows. For this guide we will focus on xR workflows because the tolerance for error is much lower in xR than in VP. The same steps are suitable for both.

A well calibrated xR stage will reveal no seams or visual artifacts that break the seamless blend between the real and virtual environments, and with adequate preparation, can be fully calibrated in less than a few hours.

Prior to beginning spatial calibration, ensure that:

1. A camera tracking system is set up and receiving reliable data.
2. The xR project has been configured with an MR set, accurate OBJ models of the LED screens, and **tracked cameras** with **video inputs assigned**.
3. The cameras, LED processors, and all servers are receiving the same **genlock** signal.
4. The **feed outputs** have been configured and confirmed working.
5. The Delay Calibration has been completed.

Concepts

A **calibration** is a mathematical equation that converts from the tracking system's coordinate space to Designer's coordinate space. Designer will figure out this equation for you if you give it **observations**.

An **observation** is created by a set of images of the stage captured by the camera of white dots on a black background (called structured light). Designer uses the structured light to work out where the camera actually is relative to the LED surfaces. At the same time, Designer captures the position of the camera according to the tracking system. These two positions form a data point that Designer uses to create a calibration.

There are two types of observations: **Primary and Secondary (P and S)**.

- **Primary observations** are used to align tracking space to Designer space. A minimum of five primary observations are required for the solving method to compute, so aim for five good observations when you begin your calibration process. Sometimes this alignment is referred to as "Registration".
- **Secondary observations** are used to align zoom and focus encoder data to Designer's lens model.

Observations are automatically marked as Primary if they have the same Zoom and Focus values as the majority of the observations.

There is no need to assign a zero point in Designer. Minimal offsets and transforms can be applied to align the tracking data and Disguise origin point prior to starting the Primary calibration, but this is not generally required.

Workflow

In this section we will outline the basic spatial and lens calibration workflows.

Pre-calibration setup

Prior to beginning spatial calibration, ensure that:

1. Your LED screens are configured with an accurate 3D model loaded in Designer. The closer your model is to real life, the better your calibration will be.
2. The cameras, LED processors, and all servers are genlocked to the same signal.
3. The outputs have been configured and confirmed working.
4. Video receive delay has been calibrated. You can find instructions here
5. A tracking system has been attached to the camera and configured both outside and inside of Designer. This is not required if you do not intend to move the camera.
6. Optional: To proceed with lens calibration, a lens encoder should be attached to the lens and configured both outside and inside of Designer.

Virtual set preview setup

Before calibrating, it is good practice to set up your stage to make it easy to assess the quality of your calibration.

To do this follow the following steps:

1. Create a **test pattern layer**
2. Assign a Direct mapping to this layer containing all screens that are used within the MR set being calibrated.
3. Configure your **feed outputs** to send content to the LED screen.
4. Create a **virtual line up layer**.
5. Assign a spatial mapping set to frontplate as the mapping for this layer. This will show the representation of the virtual set and will move/deform during the calibration process.
6. Right-click on the MR set widget to open it.
7. Optional: Use Ctrl + left-click on the header of the editor to pin the window to the GUI.

The MR set preview will show the current active camera view, the AR Virtual Lineup overlay, and the test pattern mapped to the LED screen outputs. You're now ready to calibrate!

Spatial Calibration Workflow

The spatial calibration process aligns the tracking world with the Designer world. It only uses primary observations which are automatically defined as any observations that share the most numerous pair of zoom/focus values within the list of all observations.

To carry out spatial calibration you should take the following steps:

1. Open the spatial calibration editor by left clicking spatial calibration button on the MR set.
2. Ensure the correct MR set is selected and the camera being calibrated is the current target.
3. Verify the camera tracking system is outputting the correct data and there is no scaling applied from the tracking source.
4. Set the base/most consistent shot for the project. This is called the “hero” shot.
5. Adjust zoom and focus values to their most consistently used levels for the show (If this changes later in your shoot, don’t worry, you won’t need to re-do your calibration).
6. In the calibration editor, left-click **Lock Zoom** and **Lock Focus** to fix the current zoom and focus values. This will ensure that all the observations you’re taking will be marked as Primary and used for spatial calibration.
7. Begin taking primary observations. Primary observations will be notated in the list with a **P** indicator.
8. Take a minimum of 5 good observations from different camera angles/positions that will be used in the show.
9. After each observation, the alignment should look good from the current camera position. If the alignment begins to fail, review all observations and remove suboptimal ones.
10. Move the camera around to a representative range of positions that you expect to move it near during the shoot. If you leave a part of the room without any observations near it, the calibration the camera uses when in that region may be inaccurate.

> **Tip**
>
> Don’t take too many observations too close together. This will result in Designer “over-fitting” the calibration which makes the results poor. A good rule of thumb is that each observation should be about a meter apart.

Lens Calibration Workflow

Lens calibration does the same thing for lens and focus data that spatial calibration does for position and rotation data. If you plan to only use a single zoom or focus setting on your lens then you don’t need to do this step at all.

Focus calibration is specifically intended to account for the problem of “lens breathing”. If you do not have this problem you can simply ignore the steps about focus calibration below.

If you do plan to use zoom and/or focus you should follow the following steps to calibrate them:

1. Return the camera to a “hero” or central camera position.
2. Unlock Focus.
3. Adjust the focus value to the end of the range in one direction.
4. Capture a Secondary observation. Secondary observations will be categorized with an **S** indicator.
5. Repeat steps 3 and 4. Each new observation with a new focus value will create a new Lens pose. You should normally expect to use 3 observations, one at each end of the focus range, and one in the middle. If you need more you can take them but be careful not to take them too close together to avoid overfitting the curve.
6. Unlock Zoom.
7. Adjust the zoom by 10% of the range you expect to use.
8. Lock Zoom.
9. Repeat steps 3-8 until each zoom level has several focus levels captured as observations.
10. Zoom the camera in and out and adjust the focus as needed. View the MR transmisison output to see if there are obvious points where the virtual zoom and focus of the stage elements do not match the real world. At those values, add more zoom and focus observations as needed.

Spatial Calibration Properties

Here we will give an overview of all of the settings within the spatial calibration widget. Some of these are utilized in the standard workflows above and some are for more advances usage or troubleshooting.

Settings

In this section you can configure which MR sets you want to calibrate. You can add multiple MR sets for advanced ghost-frame workflows.

Blob Settings

This allows you to configure the spacing and size of the dots displayed on the LED walls. These dots needs to be clearly detected by the camera in order for spatial calibration to give accurate results.

The optimal size and spacing of the dots generally depends on the size of the volume and the lens of the camera, so it often differs between positions based on the camera’s distance from the stage. There is no consistent range of values that works for all stages, so it is critical to find time to experiment with large and small dots to compare the results on your system.

Tips and Tricks

Here are some tips and tricks for setting up good blob settings:

- The default blob size and spacing is sufficient to get a good calibration in most scenarios.
- Calibration does not require that dot size and/or spacing be consistent between observations.
- Sometimes when you zoom in very close to an LED wall, the individual pixels of the LED may have visible dark-space between them which can be detected as separate blobs. This leads to poor spatial calibration. You can reduce blob size and change blob spacing to avoid this.
- You can use the observation debugging tools to look at the images captured in the observation and see if the blobs line up with reality as you expect.

Observations

Settings under this heading allow you to take and manage observations.

It has the following options:

Use Legacy Capture

This allows the user to toggle on a slower capture process which is more tolerant of dropped frames. This is not selected by default.

Pre-Calibration

This exposes a Capture button which will send a fixed set of colours to the LED wall in a certain order. The widget will display captured frames in order. If the images do not capture what they should (based on the description) then you are not receiving frames consistently and observations will fail to capture.

You can try the following troubleshooting steps:

1. Check if you are dropping frames. Calibration will not succeed when dropping frames.
2. Validate that video receive delay has not changed or drifted. You can find instructions on how to do this here
3. Validate that your servers and camera are still genlocked. If they are not then observations may fail.

If you cannot solve the problem, you can try ticking Use Legacy Capture which is slower but more tolerant of frame drops and other configuration issues.

Observation Controls

Here there are two buttons:

1. Add Observation - Creates a new observation
2. Reset All Observations - Deletes all observations (asks for confirmation first)

Observations List

A list of all observations that can contribute to your spatial and lens calibrations.

The listed data for each observation includes:

1. A toggle to enable and disable the observation
2. The index of each individual observation.
3. The tracked position of the camera when the observation was taken. (You can right-click on this to access the observation debugger)
4. Zoom encoder value of the observation. (user editable)
5. Focus encoder value of the observation. (user editable)

> **Note**
>
> Zoom and Focus are user editable because sometimes you cannot get the zoom or focus back to exactly where you had them when doing primary observations. Editing them manually allows you to achieve this.

5. RMS Error of the observation. This is a numeric representation of how close the observation is to the true location of the camera after calibration. Smaller values are better and you can use this to help detect outliers in your observations.
6. Type of the observation: **P** for Primary or **S** for Secondary.
7. A button to delete the observation.

Manual Observations

This section allows you to calibrate your camera manually for purely AR workflows. You can read more about this workflow in a guide coming soon.

Calibration

Adjusts global settings regarding the calibration.

- Observations image source: Live, Write, and Read.
  - Live (default): Images captured in the observation process are stored within the observation object itself and cannot be recovered if the observation is deleted.
  - Write: Backs up all captures in the observation process to a newly created folder within the Windows project folder, called /debug. (Typically only used for troubleshooting)
  - Read: Will read captured images from disc to recreate a calibration offline. (Typically only used for troubleshooting). When in Read mode you can also select the index of the folder you want to read from with the `Read images from` field.
- Adjust screen positions
  - This property defines whether the calibration solver can slightly adjust screen positions while solving. You can try using this if you don't have a very accurate 3D model to see if it improves results.

- Detect outlier blobs
  - This property is on by default and will ignore blobs that don't fall nicely within the observation model. You can disable this for some advanced troubleshooting workflows.
- Tracker Distortion Compensation
  - This property defines how the calibration system will attempt to match the tracking system's space to Designer's space and which factors it is allowed to change in order to get a better calibration. In general this should be left as default unless you know something special about your tracking system. You can also experiment with different options to see if one gives you better results. The available settings are:
    - **None**: Only accounts for the physical offsets (tracker -> focal point and tracker origin -> Disguise origin) between the tracking system and Disguise. In theory in a perfect setup this should be all you need.
    - **Single gain**: Allows for a single scaling factor between the tracker and Disguise measurements.
    - **Gains**: This is the default setting. It allows for different scaling factors in X/Y/Z axes.
    - **Gains and Skews**: Also adds skews, roughly equivalent to the tracker axes not being perpendicular.
    - **Matrix**: This is able to account for the most distortions but also is the most liable to "Over-fit" the calibration which can lead to bad extrapolated results in places outside where you've actively taken calibrations.
- Calibration scope
  - This property lets you define which parameters are solved for in the calibration. The available settings are:
    - **Lens and Registration**: Optimise both the lens intrinsics and the camera-tracker registration.
    - **Lens Only**: Optimise the lens intrinsics, take the position/rotation directly from the tracker.
    - **Registration Only**: Optimise the registration, take lens intrinsics from the tracker (if provided) or camera.
    - **Lens and Manual Registration**: Use regular observations to calibrate lens intrinsics and manual observations to calibrate registration.
    - **Registration and Focal Length**: Optimise the registration and camera focal length, but not distortion parameters.

> **Note**
>
> Here "Registration" refers to the spatial calibration.

Calibration Results

Provides a list of all lens poses created by secondary observations. You can click on each one to manually edit it. This is sometimes used in advanced troubleshooting or to correct for a change in incoming data format.

Debugging

Following the standard calibration process correctly and consistently is very reliable and typically results in very good calibrations. Despite this, errors can and do still occur. If you are getting poor calibration results this is the section for you!

The Debugging sections offers 3 tools to help with debugging bad spatial calibrations.

- Observation Debugger - the most commonly used tool
- Plot Calibration Errors - advanced
- Show 3D Observations

Observation Debugger

This is the tool you will most often used when debugging a bad calibration, because most of the time a bad calibration is caused by a bad observation. The observation debugger gives you options to view each individual observation to see if they are of good quality or not. You can look at each captured image to determine if the blobs were properly identified, clearly visible, and matched with the right virtual blobs. You can see if the image had a lot of light spill, reflection, poor masking, etc.

You can view Primary observations, either in “Solved” pose mode or “Tracked” pose mode. Tracked displays the values that we get from the tracker and from the camera, Solved displays the values transformed in to Designer space by the calibration and so is dependent on all the observations currently active that contribute to that calibration.

Secondary observations do not have a “Solved” mode to view because they are not used for the spatial calibration.

Troubleshooting a bad calibration with the observation debugger

If you have a bad calibration you will see the virtual lineup layer does not match the actual LED of the set in the background. This problem can have multiple causes, so we recommend followin this step-by-step troubleshooting guide:

1. Carry out initial checks:
    1. Press the ‘Re-run calibration’ button to ensure the set is in a fully calibrated state. This is especially important when re-opening or updating the project file.
    2. Check the primary/secondary observations are labelled as expected. It’s possible that a zoom or focus value has changed and caused them to be misinterpreted as the incorrect kind.
    3. Check the desired observations are enabled in the list editor.
2. Check whether the ‘solved’ results look good in the **observation debugger**

If not, this indicates a problem during the blob detection stage. Possible causes could be:

- Blobs have been detected in the wrong places, e.g. due to reflections. Check in the observation debugger or viewer for any detected blobs that look wrong.
- The camera moved during the observation.
- The stage model is not accurate, or the UVs are incorrect (e.g.pointing the wrong way so the screen is flipped).
- Feed output mappings are incorrect

4. Check whether the ‘tracked’ results look good in the **observation debugger**

If not, this indicates a problem with the tracking system registration. Possible causes could be:

- The tracking system is not engaged and receiving reliable data.
- The tracking system has physically moved, or something in the setup has changed between observations.
- The tracking system coordinate system is wrong, e.g. flipped axes or incorrect rotation order. Some of the debugging tools may help diagnose this.
- The tracking system is encoder-based, and physical components are mis-measured or bending.
- Try changing the solving method of the calibration.
- If none of the above help, a normal diag with the bad observations in should be enough to look into the issue.

5. If the observation debugger looks good, but Virtual Lineup layer/content looks bad, this indicates that something in tracking or registration is not being applied properly.

Possible causes could be:

- The calibration is not up to date. Press the ‘Re-run calibration’ button to be sure.
- The tracking system has physically moved, or something in the setup has changed.
- The camera has moved into a position where it is not well calibrated. Try taking another observation in this position.- The zoom/focus has changed.

6. If none of these issues are found, go through the following steps to create a project diagnostic with the information necessary for disguise support to debug the problem at HQ:
    1. Take an observation which shows this issue (for example, it looks fine in the observation debugger but the Virtual Lineup Layer is not aligned).
    2. Leave the camera in the same position that the observation was taken.
    3. Take a short device recording of the tracking data for the camera.- Take a screenshot of the camera feed, with the test pattern on the screens but without the Virtual Lineup Layer overlay
    4. Take a screenshot of the camera feed with the blob pattern preview displayed on the screens.
    5. Take an observation in Read mode which will write the images to the projects output folder and be included in a diagnostic.
    6. Export a diagnostic of the project, and along with screenshots of the MR set preview, send to **support@disguise.one**

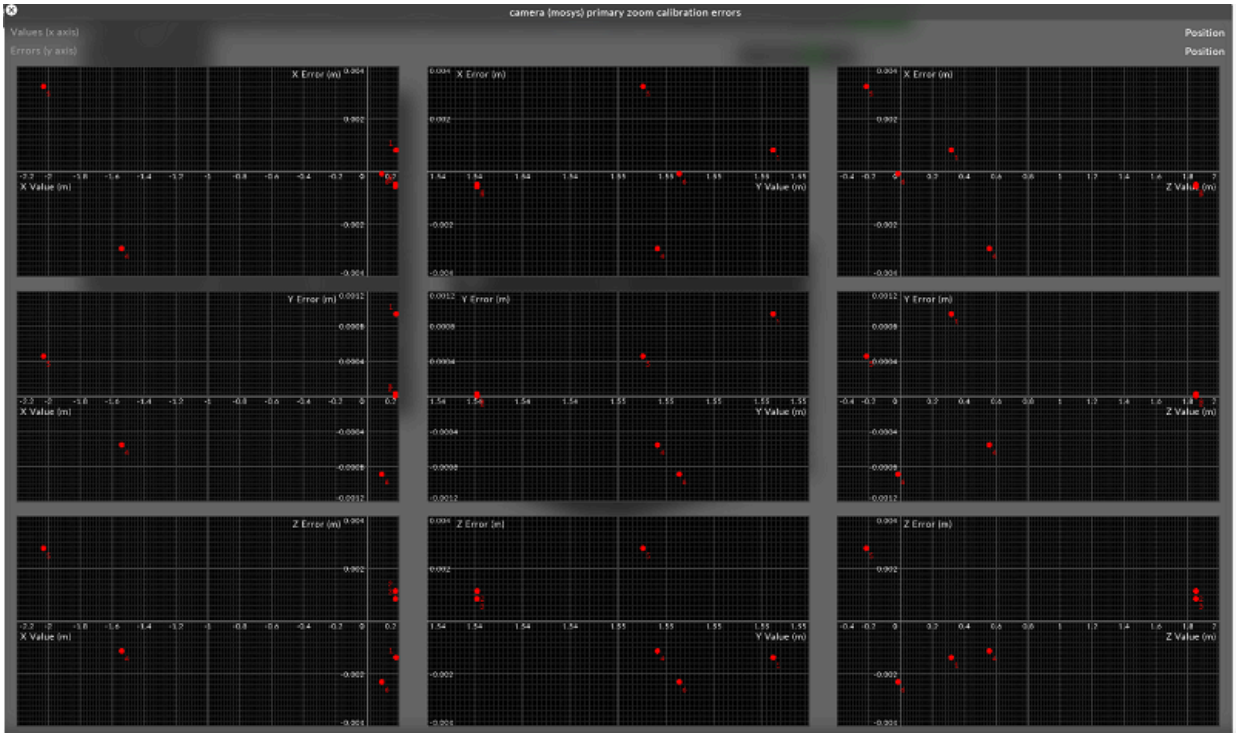
Plot Calibration Errors

This shows you plots of each spatial dimention X/Y/Z and rotations Pitch/Roll/Yaw plotted against every other dimension and rotation. This allows you to detect biases in the tracking coordinate space. You would expect each graph to look like a roughly scattered set of points with little to no correlation.

If you see an obvious pattern like a clear linear or quadratic correlation between two of the axes, this may indicate that they are unexpectedly correlated. For example, when I move the camera left and right, my tracker is also sending me movements forwards and backwards.

The Designer calibration system can account for these kinds of correlations, however this can give a very good indication of where problems within the tracking system may exist if you're having trouble with camera moves.

An example where this might be important is a mechanical tracking system like a jib. In mechanical systems we receive tracking data based on what the motors tell us, but if the jib arm flexes as it extends, the tracking system is not aware of this flex. In this case we will see a correlation between the extension of the job arm the its vertical height which can be viewed in this debug widget.



Plot calibration errors interface

Show 3D Observations

This will render 3D arrows on stage where each of the observations were taken in 3D space. This is useful if you have a part of your stage where you may have very few observations or you're trying to take your observations in an evenly spaced manner.

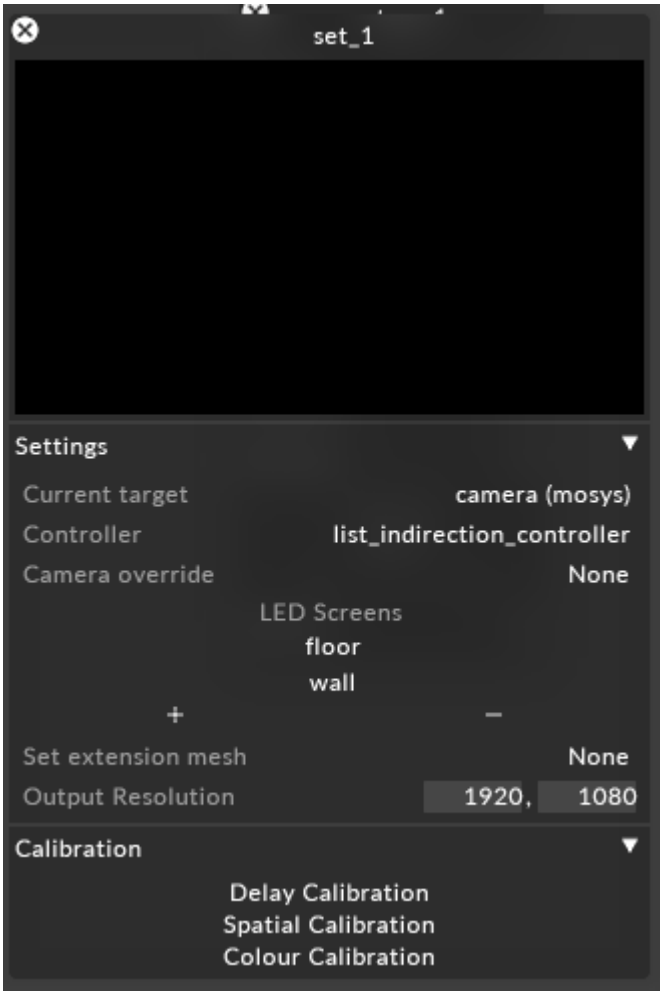
It can also be useful for visualising the alignment of the tracking coordinate system with the Designer coordinate system. If there is a systematic error between the two tracking systems then you can see the areas where they align and mis-align in 3D space.

Spatial Tracker Delay

The “Spatial Tracker Delay” field refers to the time difference between Disguise receiving an image from the camera and receiving the associated tracking data from the tracking system for a given frame

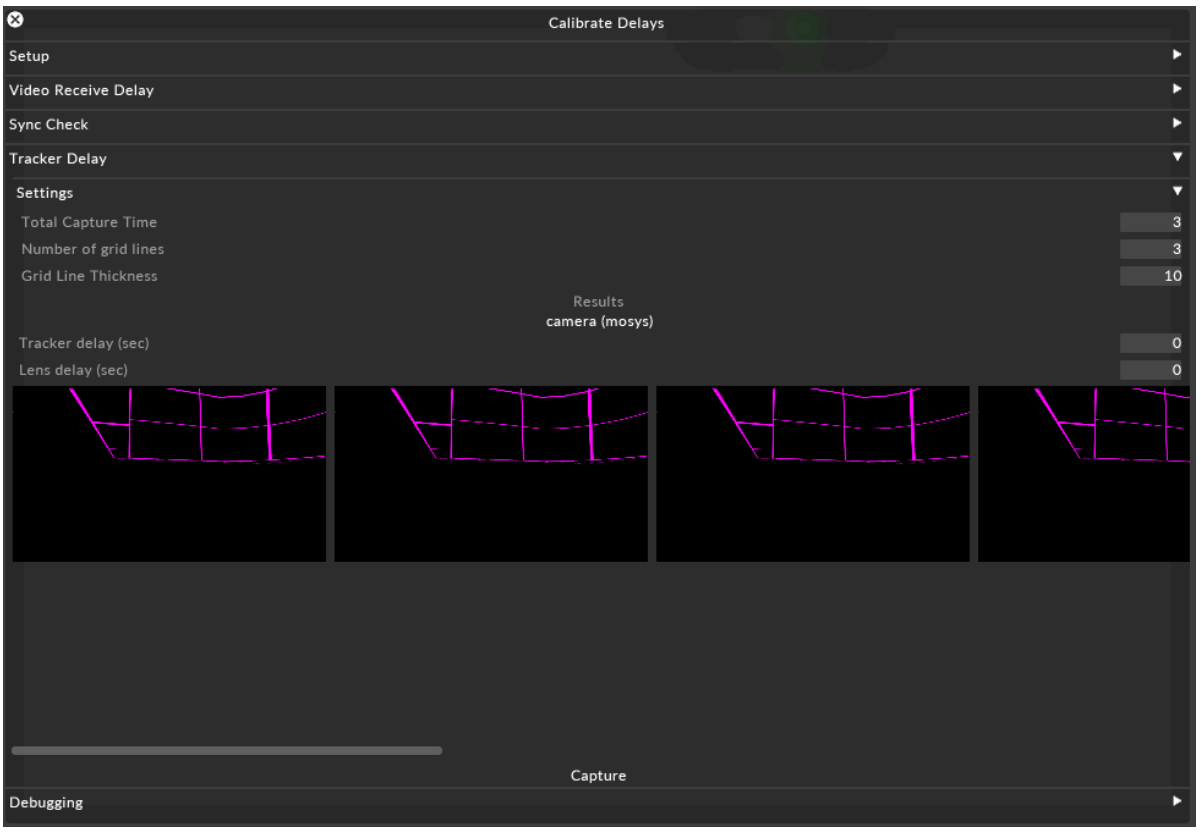
Workflow

1. Open the properties editor.



2. Navigate to **Calibration** and left click on **Delay Calibration**

3. Open **Tracker Delay > Settings**.



4. Left-click on **Capture** while moving the camera along a single axis to capture the camera movement
5. Select the image in which the grids are most closely aligned
6. Adjust the Tracker Delay field as needed to align the remaining grids (this is a subjective decision based on your visual preference).

Properties

- Total Capture Time: Total time in seconds captured during the tracker delay calibration capture
- Number of Grid Lines: The number of gridlines displayed on the backplate during the tracker delay calibration capture
- Grid Line Thickness: Thickness of lines displayed on backplate during the tracker delay calibration capture
- Tracker delay (sec): Amount of time between receiving a video frame and receiving the corresponding tracking data:
  - If negative, tracker data arrives before the video frame
  - If positive, tracker data arrives after the video frame

- Lens delay (sec)Seconds in between spatial tracking data arriving and lens data arriving.
  - If negative, lens data arrives before the spatial frame
  - If positive, lens data arrives after the spatial frame

  Calibrating the Spatial Tracker Delay typically requires a second person to operate the camera while the calibration process is running. For this reason, Lens delay can be useful when performing the Spatial Tracker delay calibration process alone as it allows you to delay the start of the Spatial Tracker Delay calibration capture.

Colour calibration

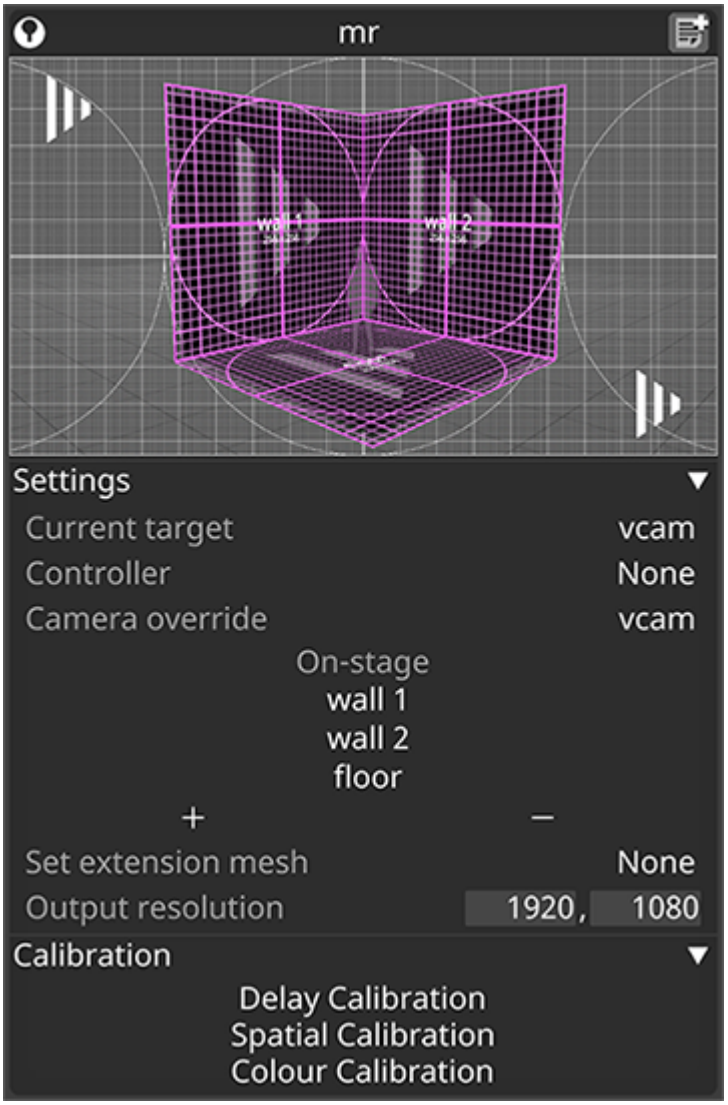
Disguise Colour Calibration is a process used to standardise colour representation across physical LED panels and virtual set extensions in an xR environment.

Variations in LED panel output, due to differences in manufacturer specifications, panel age, or batch inconsistencies, can result in perceptible mismatches between physical and virtual elements. The colour calibration workflow compensates for these discrepancies by generating a 3D LUT (Look-Up Table) that characterises and adjusts the LED output to align with the intended virtual colour space.

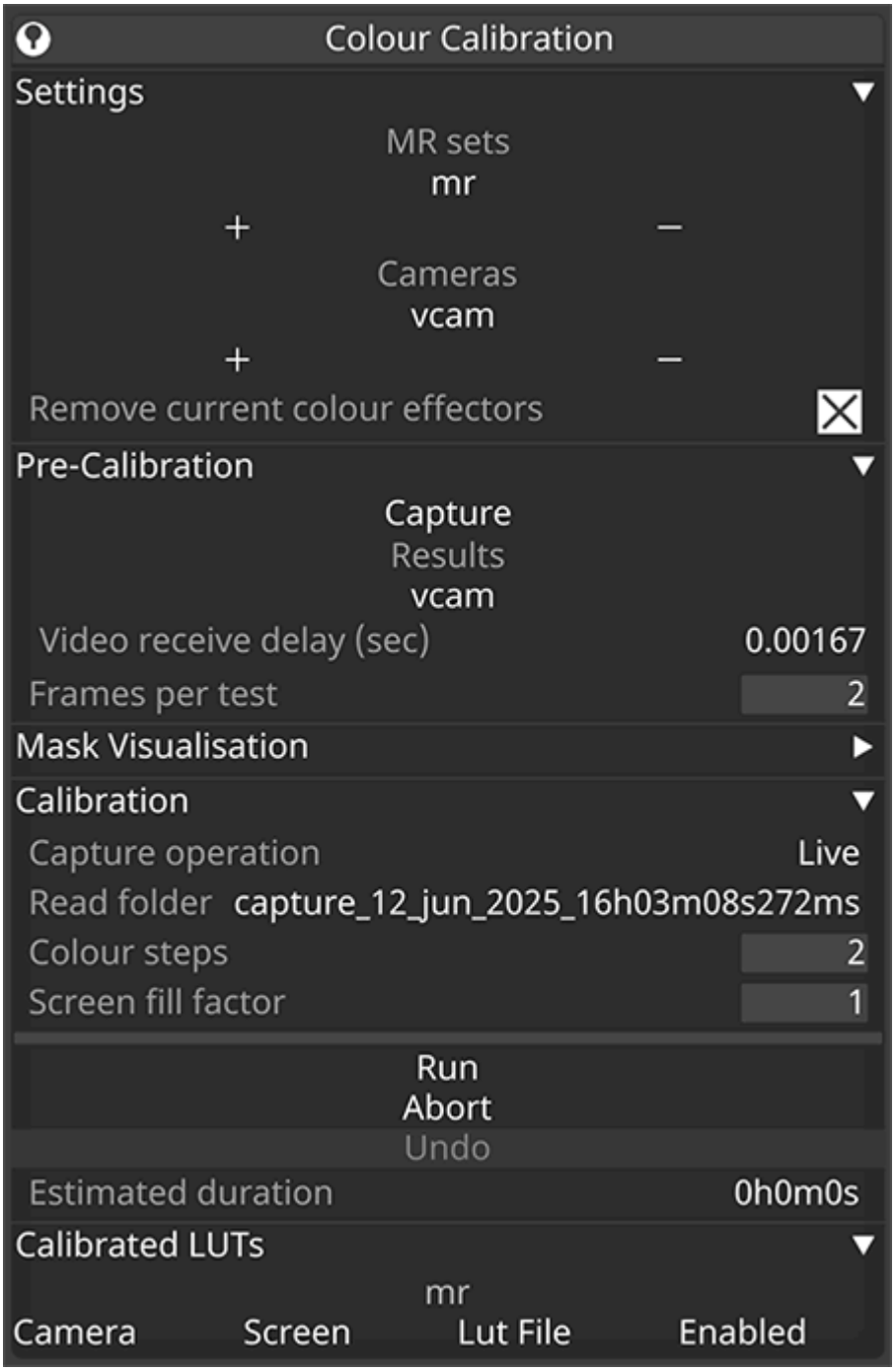
The calibration process measures the LED display's colour response and reconciles it with the system's reference values, ensuring consistency across all visual surfaces. This enables accurate blending of content between real and virtual components, even when using LED panels from different vendors or with different physical properties.

Workflow

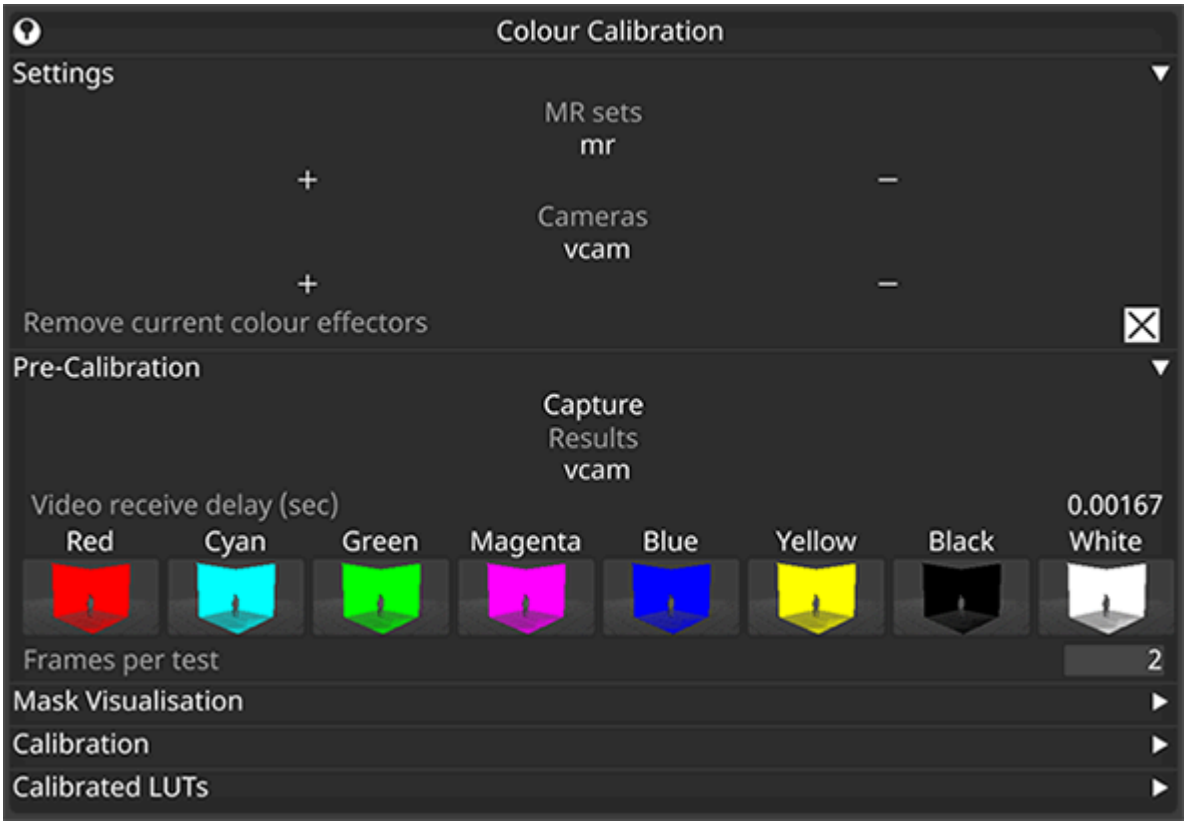
1. Complete spatial and delay calibration in full.
2. Bring all lighting, LED and cameras to optimal settings and clear the stage of any physical items.
3. Fill the camera frame with LED pixels.
4. Open the MR set.



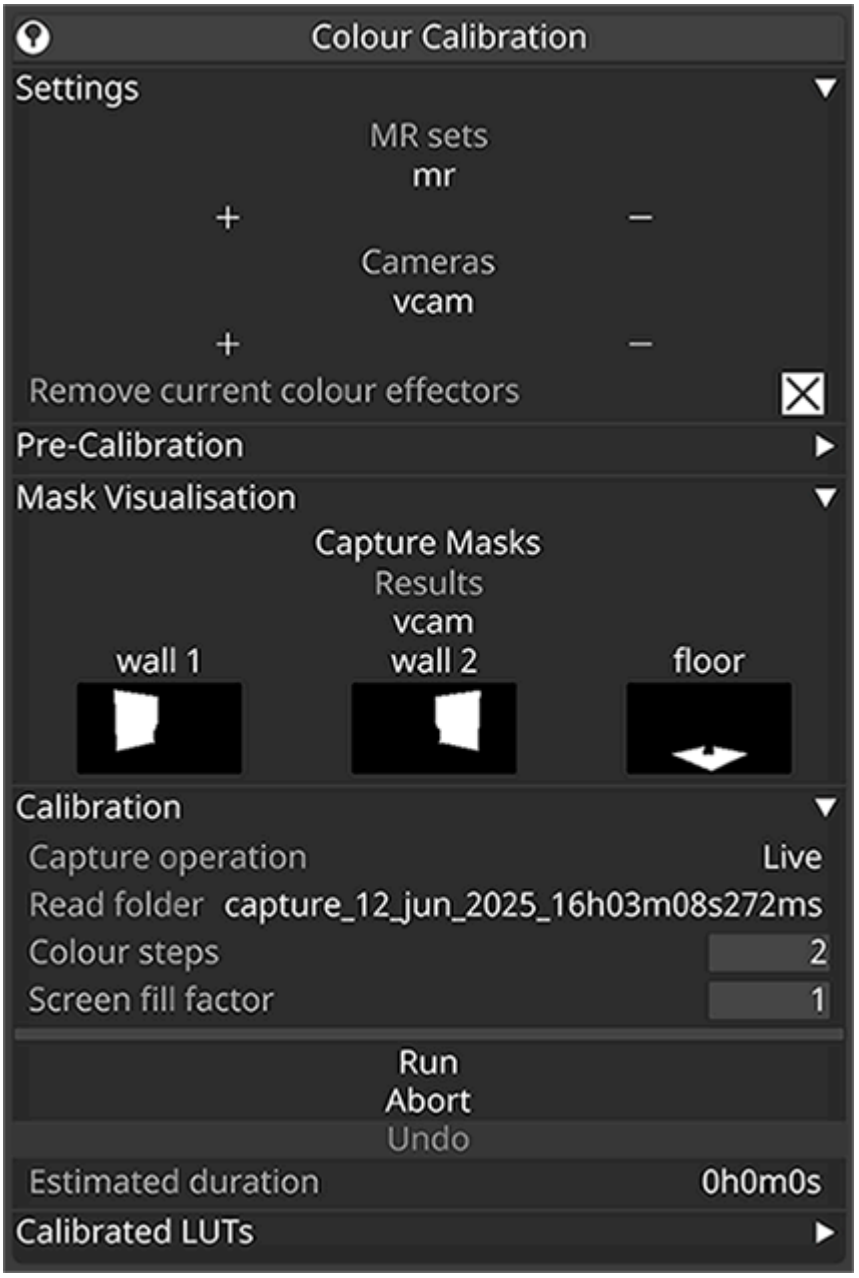
5. Open the **Colour calibration** menu.



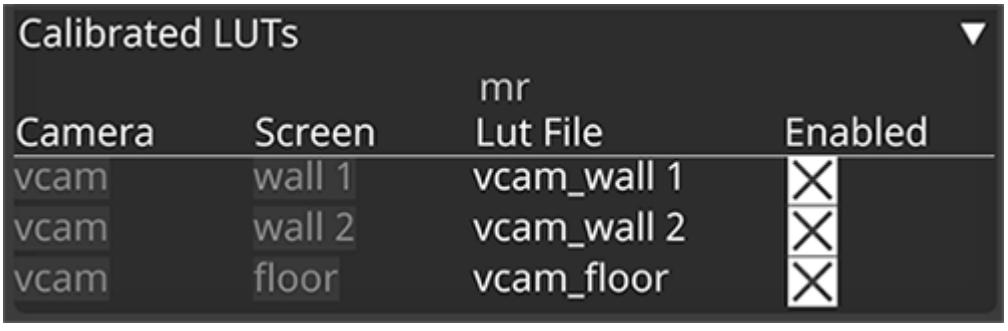
6. Ensure the correct MR sets to be calibrated have been populated in the **Settings** tab.
7. Expand the **Pre-Calibration** tab. The Video receive delay should be autopopulated from the Delay calibration. The camera will be automatically taken from the active camera of the MR set. For multicam setups, include all cameras to be calibrated.
8. Click **Capture**.



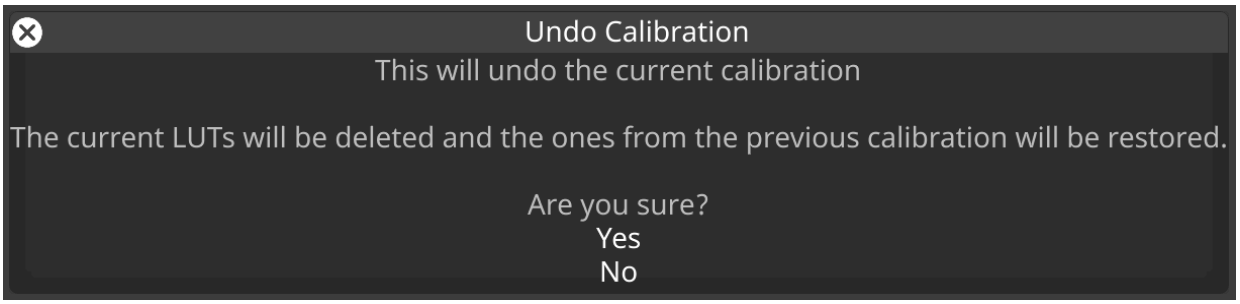
9. Step up incrementally in the **Frames per test** field until the colours in the preview are saturated and appear to be true representations of each colour.
10. This ensures that all steps of each colour range are correctly captured by the camera.
11. Optionally, expand the **Mask Visualisation** tab.
12. Hit **Capture Masks** to generate a preview of the calibration mask for each LED-screen/camera pair. Review that any occlusions are correctly masked in black and that the white regions cover only the screen surfaces used for calibration. To make this pre-flight check meaningful, ensure the setup remains unchanged between previewing masks and performing the actual colour calibration. – These preview masks are for inspection only; fresh masks will be generated during the actual colour calibration.



13. Expand the **Calibration** tab.
14. Hit **Run** to start the calibration. Ensure the camera does not move and lighting levels do not change. Once complete, a mapping will be created for each combination of camera/LED screen/LUT file as determined by the elements included in the MR set object.
15. Expand the **Calibrated LUTS** tab to toggle on and off each LUT if needed.



16. Increase the value of the **Screen Fill Factor** if lighting from each LED is spilling onto the edges of the floor and affecting the calibration result. When the value is set to 1, the colour output will fill 100% of the screen.
17. If a previous colour calibration was in place before running the current one, it will be automatically backed up. You can restore the LUTs from the previous calibration by clicking the **Undo** button under the **Calibration** tab and selecting **Yes** when the confirmation prompt pops up.



Each physical camera will require its own colour calibration, the LUTs are switched automatically by the indirection controller when it switches cameras.

If the **Capture operation** mode is set to **Write**, thumbnails of each colour calibration step will be written to the debug captures folder within your project. This can be useful for sending to the support team to reproduce any issues you may encounter.

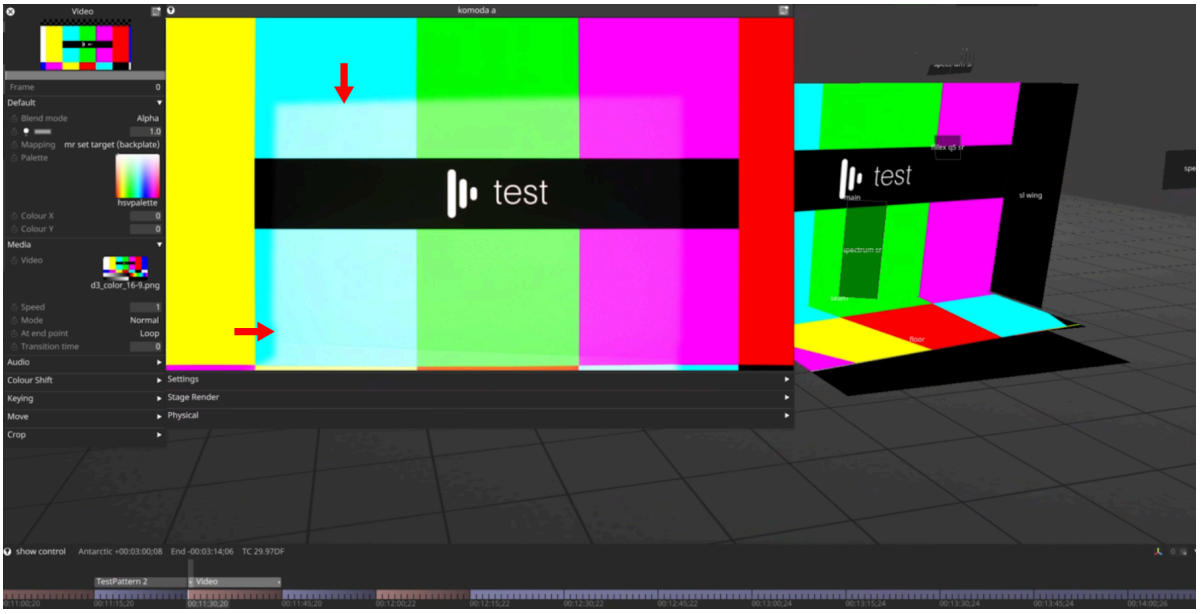
For further colour adjustment to the set extension, a **Colour Adjust** layer can be created and assigned a CamPlate Mapping set to backplate with all cameras used for the MR transmission added as the mapping objects.

Inverse LUT Method

The Inverse LUT colour calibration method builds upon the existing colour calibration workflow, with the key difference being that instead of using the calibration to adjust the colors going to the LED to match a specific color space that may not be possible with the LED, the generated Lut will be able to adjust the color of the set extension to match the LED.

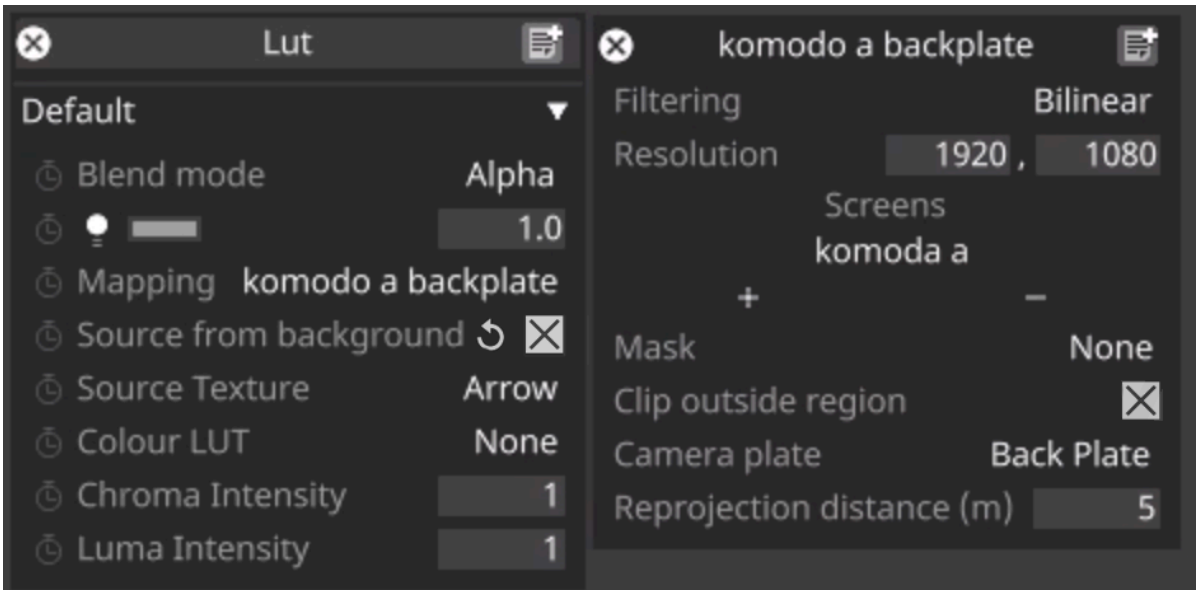
Workflow

1. Set up your LED screen(s), camera, and MR set as normal and continue with the regular delay and spatial calibrations.
2. To test the effectiveness of this process you can create a video layer with a colour test pattern selected, mapped to the MR set target “backplate”. When positioning the camera to show the set extension you will see a clear difference in color between the LED and set extension.

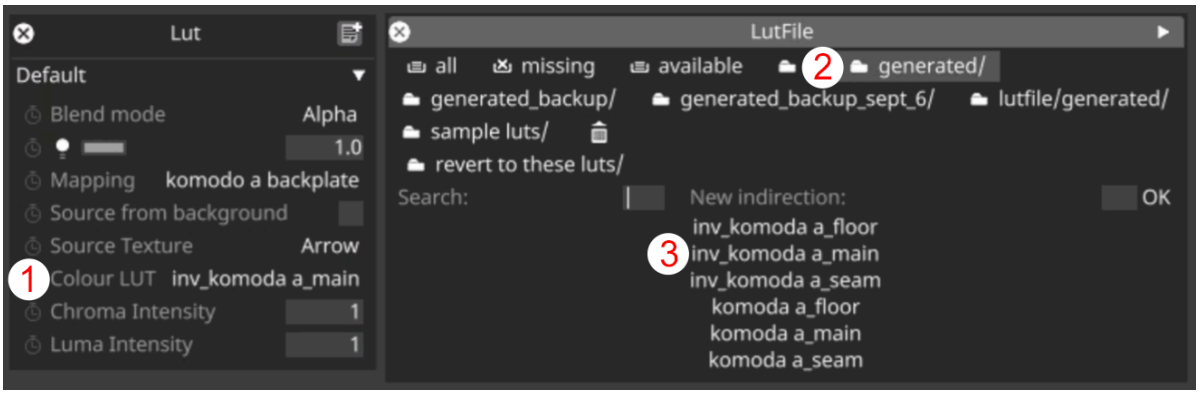


3. Run the colour pre-calibration and calibration like normal but when completed it will automatically enable the generated LUTs to the LED screens, you will need to disable these.

4. On the timeline, create a LUT layer, you will need this layer to extend anywhere where content with set extension or AR is used.

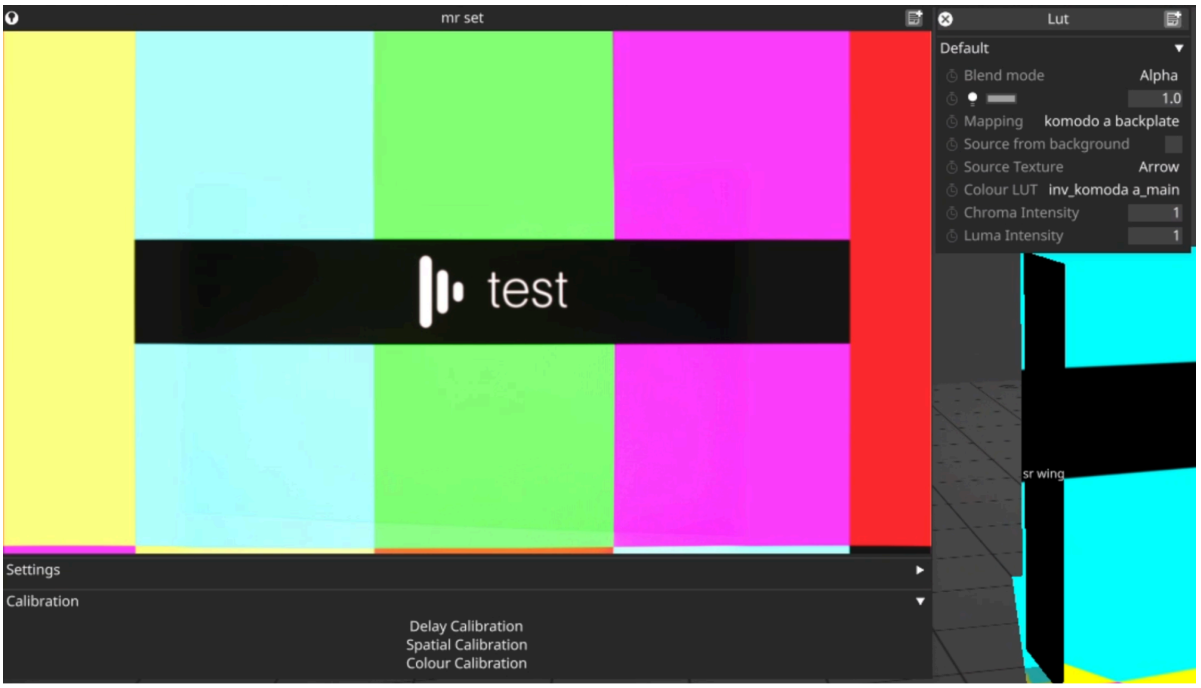


5. In the LUT layer create a “camera plate” mapping with the camera added in the “screens” section, set the camera plate to back plate and enter the camera’s resolution.



6. In the “Colour LUT” section, go to the “generated” tab and for every lut created there will be an inverted LUT as well. Select the LUT labeled “Inv_(camera name)_(main LED screen)”

7. Now the set extension will match the same colors at the LED screen. You can also play with the “Chroma and Luma Intensity” variables to fine tune your results.



8. You can also create a second Lut layer with the camera plate mapping to the camera but instead selecting “Camera Plate” to “Front Plate” this will affect any AR composite elements.

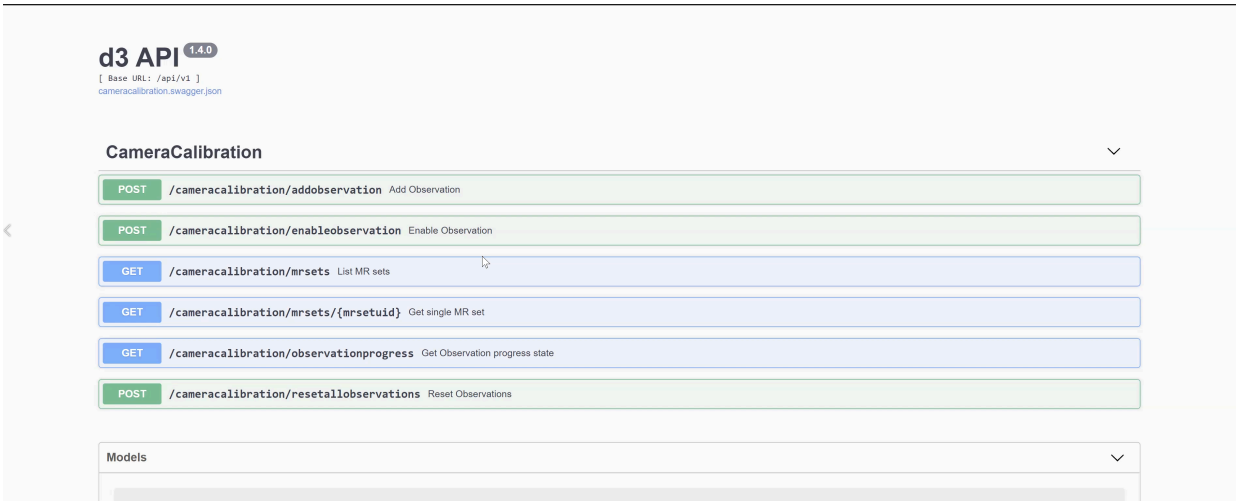
API xR Calibration

This topic covers the basic steps of aligning the physical and virtual worlds within Designer, using the API.

What is the API?

The xR API is an application programming interface accessible via d3Net that allows direct access to specific software functionality within the Designer software.

Designer version r20 introduced the ability to perform a spatial calibration directly from the API, allowing remote access to the functionality while reducing the number of people required to execute this process to one.



To access the API, navigate to the base URL `http://localhost/docs/v1/index.html`

Workflow

A well-calibrated stage will reveal no seams or visual artifacts that break the seamless blend between the real and virtual environments, and with adequate preparation, can be fully calibrated in less than a few hours.

Preparation

Prior to beginning spatial calibration, ensure that:

1. A camera tracking system is set up and is receiving reliable data.
2. The xR project has been configured with an MR set, accurate OBJ models of the LED screens, and **tracked cameras** with **video inputs assigned**.
3. The cameras, LED processors, and all servers are receiving the same **genlock** signal.

4. The feed outputs have been configured and confirmed working.
5. The **Delay Calibration** has been completed.

Calibration

A **Calibration** is a set of data that is contained within an individual camera object. The calibration process uses tracking data as a base against structured light patterns to determine the reality in relation to the raw tracking data.

Observations

An **observation** is a set of images of the stage captured by the camera of white dots on a black background (called structured light). The user determines the number of dots and their size/spacing.

- **Observations** are used as data points for the predefined algorithm Disguise will use to align the tracked camera with the stage and set up the lens characteristics.

There are two types of observations used in the process: **Primary and Secondary (P and S)**.

- **Primary observations** are the positional or spatial observations used to align the real world and virtual cameras. A minimum of five Primary observations is required for the solving method to compute, so aim for five good observations when you begin your calibration process. Primary or secondary status is defined by the most common zoom and focus values in the pool of observations.
- **Secondary observations** comprise the zoom and focus data that create the lens intrinsics file. Each new zoom and focus position will create a new **Lens Pose**.

Taking observations

There is no need to assign a zero point in Designer. Minimal offsets and transforms should be applied to align the tracking data and Disguise origin point prior to starting the Primary calibration.

Start by taking primary observations at a single locked zoom and focus level. This Primary Calibration will calibrate the offsets between the tracking system and Disguise's coordinate system.

- After each observation, the alignment should look good from the current camera position. If the alignment begins to fail, review all observations and remove suboptimal ones.
- The Secondary calibration calibrates the zoom and focus data. This will align the virtual content and the real-life camera zoom and focal changes.

Lens Pose

A **Lens Pose** is the result of the data captured in the observation process. They are different checkpoints that Disguise will intelligently interpolate between. The number of lens poses you will end up with will be dependent on the range of your camera's lens.

- A new lens pose is created for every new combination of zoom and focus values. The most common zoom/focus combination will be the **Primary** lens pose, attributed to the **Primary** calibration.

Set Extension

A set extension is a virtual extension of the digital content being displayed on the LED screens in the MR set. It is used to fill the in-camera space surrounding the LED screens.

Workflow

1. Configure set extension mask.
2. Configure set extension addition.
3. Configure set extension feather.
4. Configure set extension mesh.

Example

Set extension mask

For using objects in the visualiser to mask to areas of the set extension, perform the following steps:

1. Add an LED screen to the stage.
2. Set this LED screen render target to Set extension mask.
3. Define the screen's mesh to whatever masking shape you want.
4. Add the LED screen to the MR set.

This screen will mask the set extension, leaving only live video and AR effects.

Set extension addition

This functionality allows objects on the stage to mask out live content and show set extension in their place.

1. Add an LED screen to the stage.
2. Set this LED screen render target to Set extension addition.
3. Set this screen's mesh to whatever masking shape you want.
4. Add the LED screen to the MR set.

This screen will mask the live content, set extension is displayed instead

Set extension feather

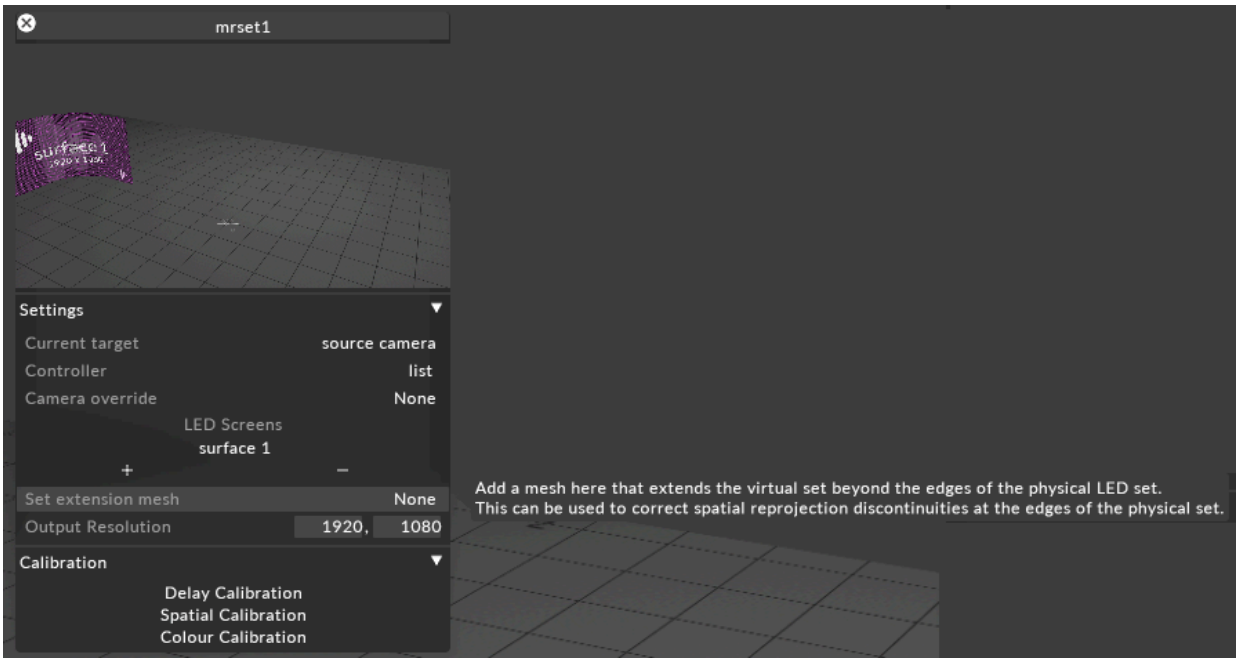
To adjust the feather of your set extension, navigate to the target camera and adjust the Set Extension Feather in the settings.

Set extension mesh

Set extension meshes were added to prevent tearing between the set extension and the LED screens when the camera moves. Content is reprojected based on the position of the camera so as to maintain the correct perspective as the camera travels.

Previously, the reprojection was done using a 2D plane at a reprojection distance that was a best fit for the varying reprojection distances required. Set extension mesh is used to virtually extend the geometry of the LED screens, so that content can be reprojected to the correct planes at the correct distances. Set extension meshes should be extrusions from the outer edges of the screens only.

1. Create the set extension mesh object.
2. Copy it into meshes folder of your **project folder**.
3. Open your MR set editor.
4. Set the Set extension mesh to be the relevant mesh.



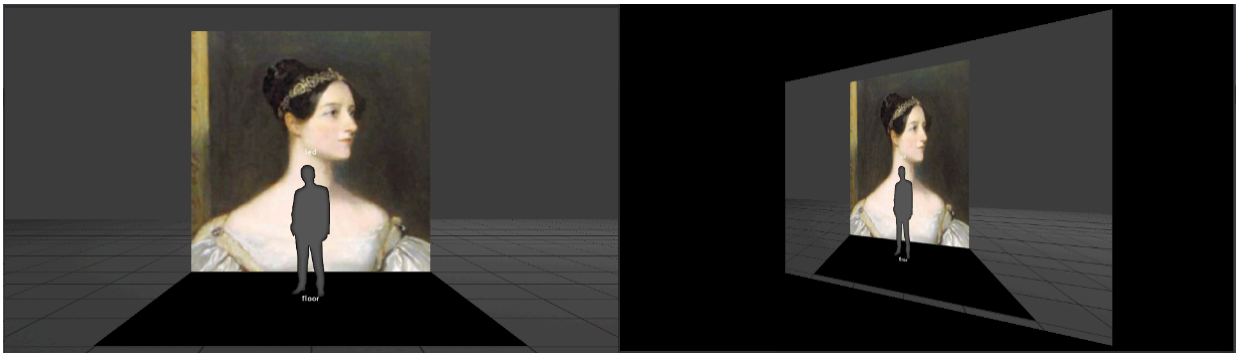
Virtual Cameras

Virtual cameras can be used as part of MR or green screen workflows to allow shots to be captured from positions outside of the range of the physical cameras. By reprojecting the filmed action and virtual content to a new viewpoint, wide shots can be captured for small stages with limited space.

A virtual camera can also be used to traverse a large virtual scene, such as a cityscape. It is possible to start by moving through the streets rendering the virtual scene, and end by transitioning into the view of the real camera on an xR stage, capturing the people on the stage within your virtual scene.

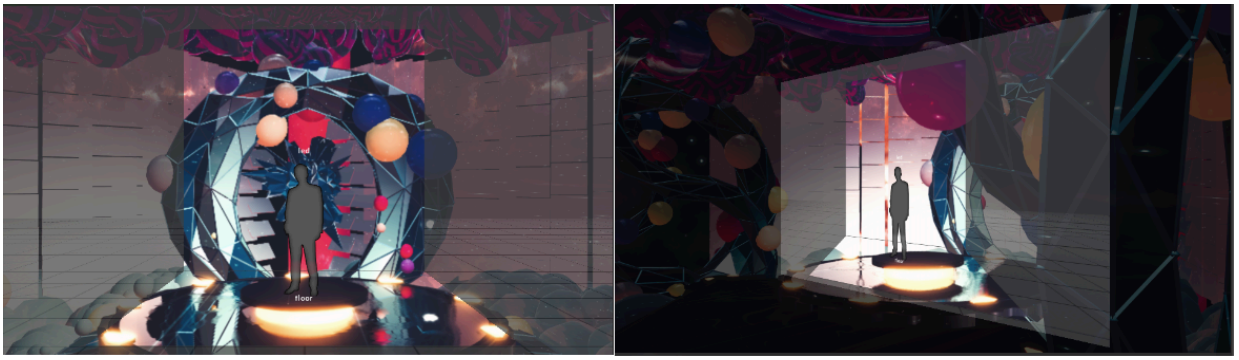
Background

Virtual cameras are designed to be interchangeable with real cameras within a MR or green screen setup. They should be interacted with in similar ways to a real camera. Virtual cameras work by reprojecting the filmed content from a parent camera to the virtual camera's viewpoint:

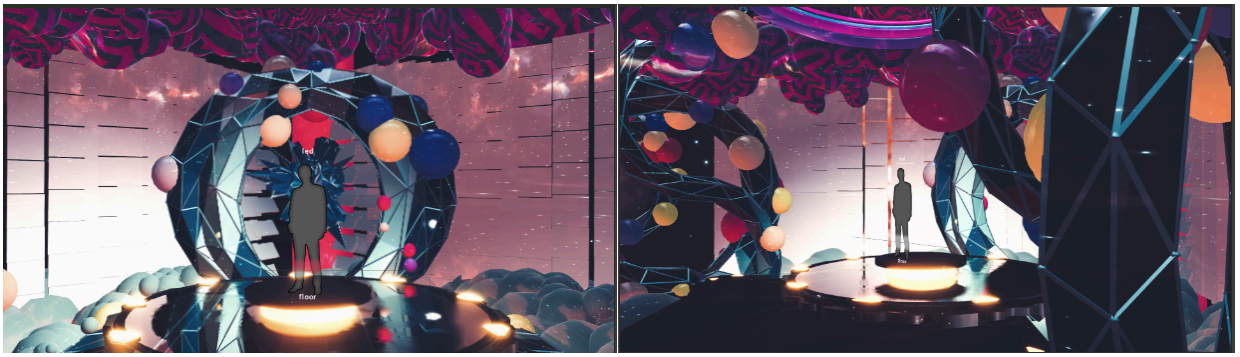


Reprojection of the content filmed by the parent camera

We render the virtual scene from the virtual camera's viewpoint, add set extension to the outsides of the reprojected plane, and account for the reprojection in the content we send to the screens. This gives us the virtual camera viewpoint:



Reprojection with translucent set extension



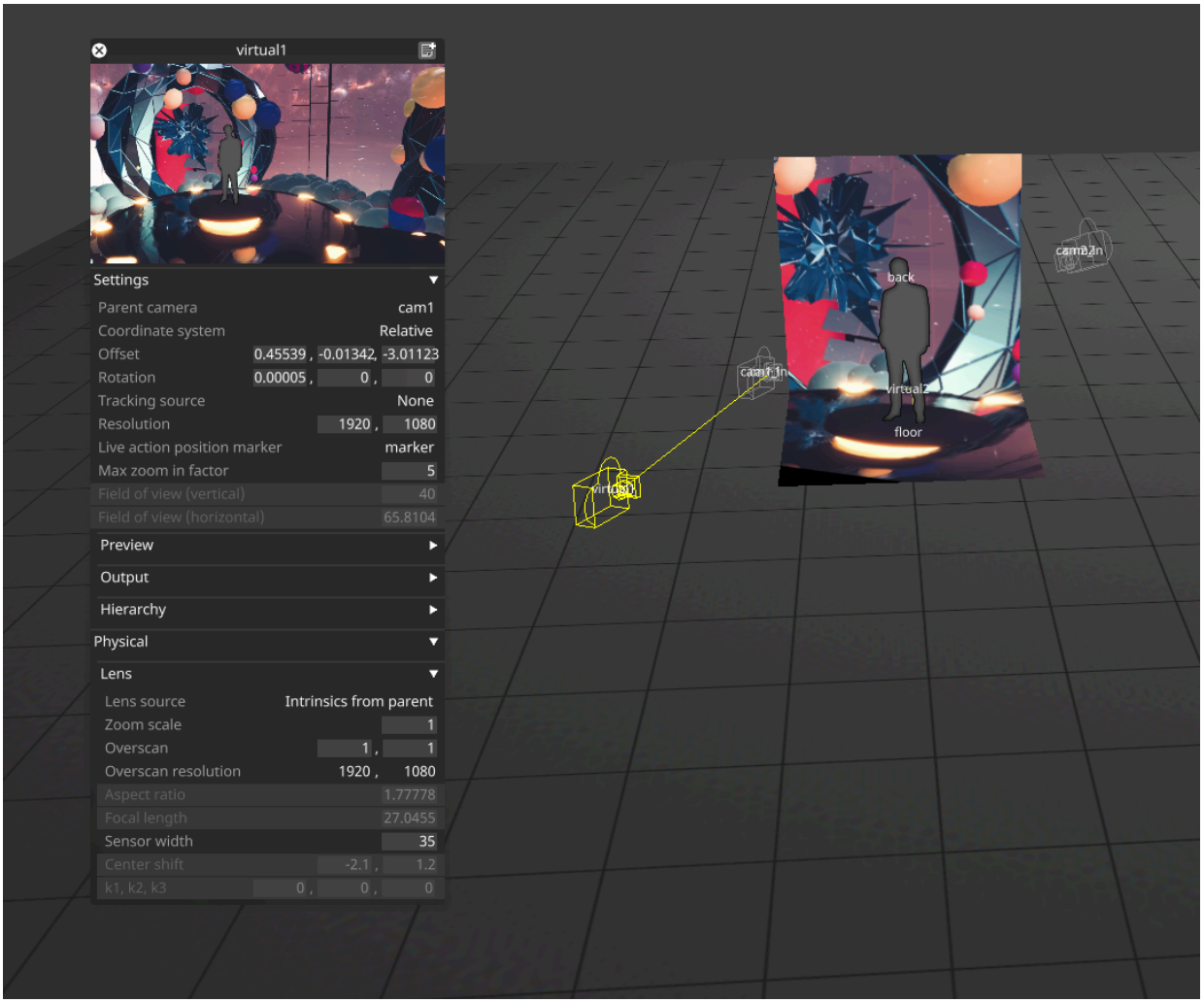
Final composite, from initial and reprojection viewpoints

> **⚠ Caution**
>
> If the angle between the viewing directions of the parent and virtual camera is too large, the people and props on stage can look flat, as seen in the images above.

Workflow

To set up a virtual camera

1. Create a new virtual camera in the Cameras list in the stage.
2. Set the **Parent camera** property to the desired real camera.
3. Add a **Live action position marker** located at the average position of live content on the stage (e.g. people). This will ensure that the live content appears with the correct perspective relative to the virtual content when the virtual camera moves.
4. Set the virtual camera as the target camera of the MR set.
5. Map content to the MR set front and backplates as normal.



Virtual camera setup

Virtual camera properties

There are several properties than can be set on virtual cameras to control their behaviour. Many of these are the same as regular **cameras**. Other notable properties to be aware of for virtual cameras specifically include:

- **Max zoom in factor** can be set to restrict how much the filmed content can be zoomed into before purely virtual content is displayed. This prevents excessive pixelation when moving the virtual camera towards the live action.
- Lens settings can be set under **Physical→Lens**. These can be defined locally in the virtual camera, set to follow the zoom of the parent camera (with an optional scaling applied), or set to follow all the intrinsics of the parent camera (with an optional zoom scale).
- **Output type** under **Output** can be set to **Spherical** to output spherical content for viewing in VR. See xR for VR for more details about this workflow.

To control a virtual camera

There are various ways for controlling the position of the virtual camera. You can either:

- Use the 3D object manipulators to move the virtual camera object around in the stage.
- Edit the values in the virtual camera editor. Note that values can be defined in a **Global** coordinate system, or **Relative** to the parent camera.
- Use a tracking source: A tracking source can be used to control the pose and lens intrinsics for a virtual camera in the same way as a real camera.
- Keyframe camera animations using AnimateCameraPreset or AnimateCameraControl.

To keyframe virtual camera animations

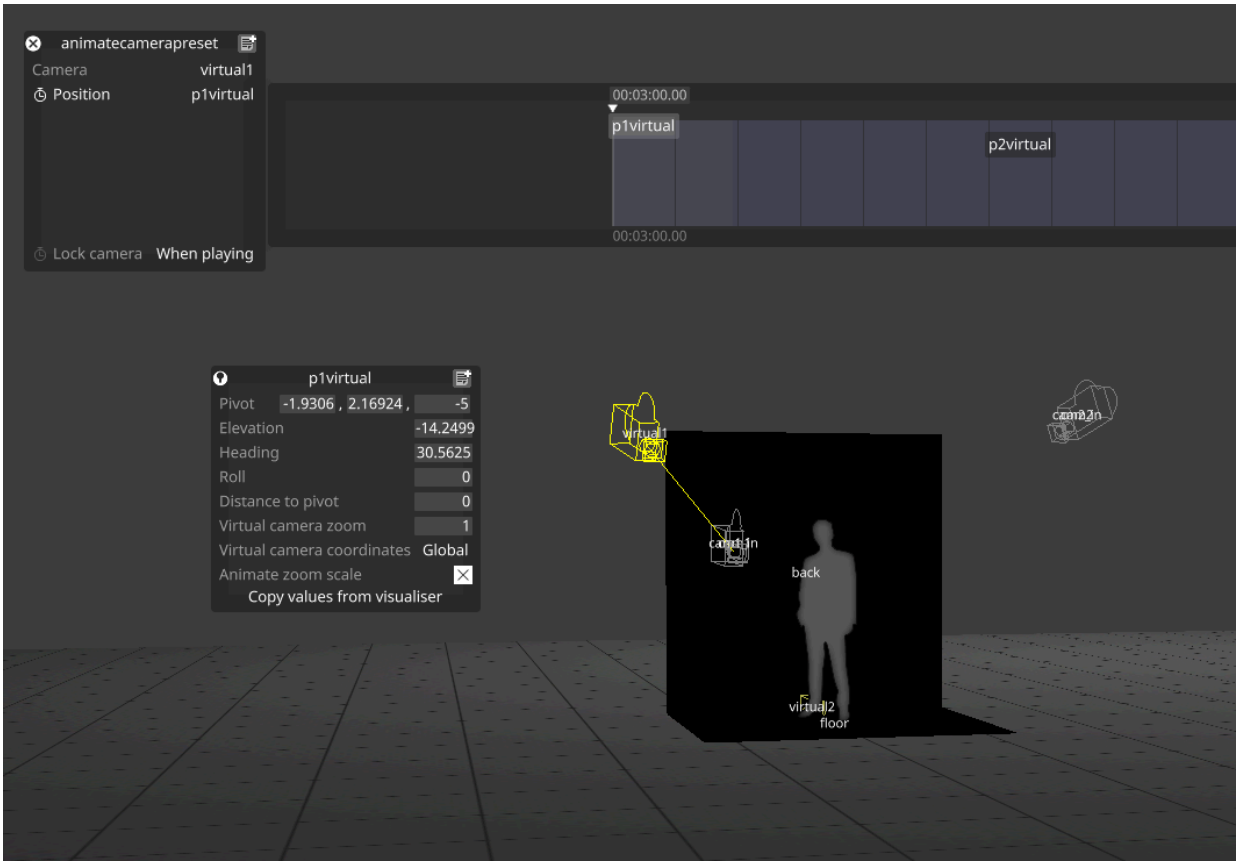
Virtual camera animations can be keyframed similarly to standard cameras using the AnimateCameraControl or AnimateCameraPreset layers.

AnimateCameraControl:

1. Add an **AnimateCameraControl** layer.
2. Set the camera to the virtual camera you wish to animate.
3. Set the coordinate system to **Global** or **Relative** as desired.
4. Keyframe virtual camera properties. In addition to the standard properties for a normal camera, zoom scale can also be keyframed.

AnimateCameraPreset:

1. Add an **AnimateCameraPreset** layer.
2. Set the camera to the virtual camera you wish to animate.
3. Add new virtual camera positions to the **Position** property.
4. Set the coordinate system to **Global** or **Relative** as desired.
5. In addition to the standard properties for a normal camera, the virtual camera's zoom can be set. Toggle the **Animate zoom scale** option to choose whether to animate zoom scale or global field of view.
6. Keyframe several virtual camera positions to animate the camera between these positions. Positions in Global and Relative coordinates can be mixed, and the camera will move between the coordinate systems.



Keyframing virtual camera positions

Limitations

- If the virtual camera is significantly off-axis from the real camera, the live objects (e.g. people) will become skewed. The projection can be thought of as looking at a 2D cardboard cutout from a different angle.
- All live action must be completely within the frame captured by the real camera. If not, it will be cut off during the reprojection.
- If the lens is heavily distorted, there maybe be distortion artifacts. A radial mask can be set in the editor to remove these effects, which are most pronounced at the edges of the image.
- If you attempt to move the virtual camera forwards towards the live action, the picture will become progressively more pixellated. This is because you are zooming in on the captured image at fixed resolution. This can be mitigated by setting the **Max zoom in factor**.
- If the virtual camera moves behind the live action, it will render virtual content only.
- If the parent camera moves quickly, there may be some noticeable discontinuity between filmed content and the set extension. This may be more noticeable if the virtual camera is near to the parent camera. For fast camera moves it is recommended to use a real camera rather than a virtual camera.

Legacy Workflow

Virtual camera animations could previously be keyframed using additional options in the AnimateCameraControl or AnimateCameraPreset layers. The new workflow supersedes this method, however for backwards compatibility the old workflow can be enabled by setting the option switch **enableLegacyVirtualCameraWorkflow**.

Camera Switching

While there can only be one target camera in an MR set at any given time, an MR Set can contain multiple cameras that can be switched between using an Indirection Controller.

The target camera of an MR set determines the video input being routed to the MR set. However, multiple cameras can be added into an MR set.

Camera switching in xR requires all delays in the system to be compensated for perfectly, so that all parts of the output switch view in the same frame. This requires introducing a delay between the user requesting a switch, and the switch being visible in the output.

> **Note**
>
> **Please note**: Delays must be perfect to then get in sync switching.

Controllers

Inside Disguise Controller we have 6 Indirection Controller Types:

- DMXIndirectionController
- ListIndirectionController
- MachineListIndirectionController
- ManualIndirectionController
- SequencedIndirectionController
- OSCIndirectionController
- UDPIndirectionController

These allow multiple protocols to be used to switch the active camera inside Disguise, as well as non-external trigger based.

Workflow

DMXIndirectionController

This allows for Cameras to be selected in a Bank and Slot mechanism.

1. Ensure you have set up Disguise to receive DMX.
2. Create and assign a DMXIndirectionController to the MR Set.
3. In the Sockpuppet bank editor, a Camera bank list is now made available.
4. Populate banks and slots with Cameras you wish to select.

5. Enter the Universe and channel you wish to select the Camera bank with into the controller.
6. The slot is the next channel immediately after the bank, so you must specify 2 channels in your DMX sender.
7. From the DMX sending device, send the appropriate 2 channel identifier to the universe and channel you selected.

ListIndirectionController

This allows for Cameras to be selected on a 0 Based index

1. Create a ListIndirectionController.
2. Add the Cameras as Resource.
3. The order of the Cameras as the Index Number.
4. Scroll through the Index to then choose the active cam.

The selected index is 0-based, meaning the first item in the list is selected by index 0.

Right-clicking on the selected index allows the user to enter an expression to control the index (e.g. allowing control from an **external protocol**.)

If the selected index is less than 0 or greater than or equal to the number of elements in the list, then the controller selects `None` as the Resource.

MachineListIndirectionController

This allows for certain machines to only output the perspective of specified cameras. A use case for this is when splitting up Front and Back plate rendering over multiple machines.

1. Create a ManualIndirectionController
2. Select a Camera.

SequencedIndirectionController

This is controlled by keyframes in an IndirectionControl layer. The indirection control module allows indirections to be sequenced on the timeline. One Indirection control module controls one indirection.

To sequence an indirection:

1. Add a SequencedIndirectionController in the controller field of the indirection.
2. Add an **IndirectionControl** layer to the timeline.
3. Select the controller in the Indirection control module.

OSCIndirectionController

This allows you to force a camera to be active over OSC.

1. Create an OSCIndirectionController
2. Create an OSC Device
3. Add the OSC device to the Controller.
4. Remove the $ sign from the OSC address field.
5. The string to recall the Camera to be active is the camera's name inside Disguise. Currently the OSC indirection controller requires you to use the full path of the object you are selecting such as:

path: `/d3/indirection/osc`

value: `objects/camera/cam1.apx` **

> **Note**
>
> **Please note**: Confirm that the labels for camera objects and the OSC indirections match and are named using the same format.

UDPIndirectionController

This allows you to force a camera to be active over UDP.

1. Create a UDPIndirectionController.
2. Add the number of Resources needed. This will be the number of cameras in the scene.
3. Add a Key to trigger that active camera, and add a resource which would be the Virtual Camera.
4. Add the receive port number from the UDP sender.
5. Send the command to Disguise and see the Current Target change. Turn on Verbose Logging to check incoming strings in the console (alt+c).

Inner-Outer Frustum Workflow

Not only can content be mapped to the within the camera's frustum, another stream of content can simultaneously be mapped to the outside of the camera's frustum.

Inner - Outer Frustum workflow

This workflow allows natural lighting and colour from content streaming via RenderStream to illuminate talent or objects within the performance area while the camera tracks through the space.



Use these steps to set up this workflow:

1. Create an MR set and add the LED surface(s).
2. Create a new camera in the Stage menu, move it back from the default position, and assign as the Current target in the MR set.
3. Create a RenderStream Layer.
4. Right-click on Workload.
5. Add an asset to the Cluster Workload and select the desired stream from the autopopulated Asset list.
6. Confirm that the RenderStream Layer is assigned to the Backplate for the MR set.

7. Select the desired scene level. At this point content mapped to the inner frustum will be visible on the LED screen.
8. Open another instance of the Unreal Engine project in order to create a second stream.
9. In Designer add another camera to the stage. Rename this camera “outer frustum camera” or similar label.
10. Position this camera in approximately the same location as the first camera and change the field of view to a larger value such as 120 so that the entire LED can be seen within the camera’s thumbnail.
11. Create a second MR set that references the second camera, and select the same LED screen as the target.
12. Duplicate the RenderStream Layer.
13. Create a new Cluster Workload.
14. Add an asset and choose the second instance of RenderStream in the asset list.
15. Change the mapping to the second MR set.
16. Set the camera plate of the second MR set to Backplate At this point you will have content mapped to the LED screen outside of the camera’s frustum.
17. In order to add some feathering to the edge of the inner frustum stream, you may wish to add a slight amount of overscan such as 1.2 to the first camera in its property window to make it slightly wider.
18. Then, add a soft edge mask to the MR set mapping on the first RenderStream Layer as well. Open the RenderStream Layereditor and right-click on the MR set (spatial) mapping. Once open, you can create a new softedge mask for this layer. To read more about creating masks, visit this **link**.



The Inner frustum highlighted with different machine inputs identified

Further Reading

Read the **Disguise Whitepaper** to learn more about how the Disguise architecture works with RenderStream.

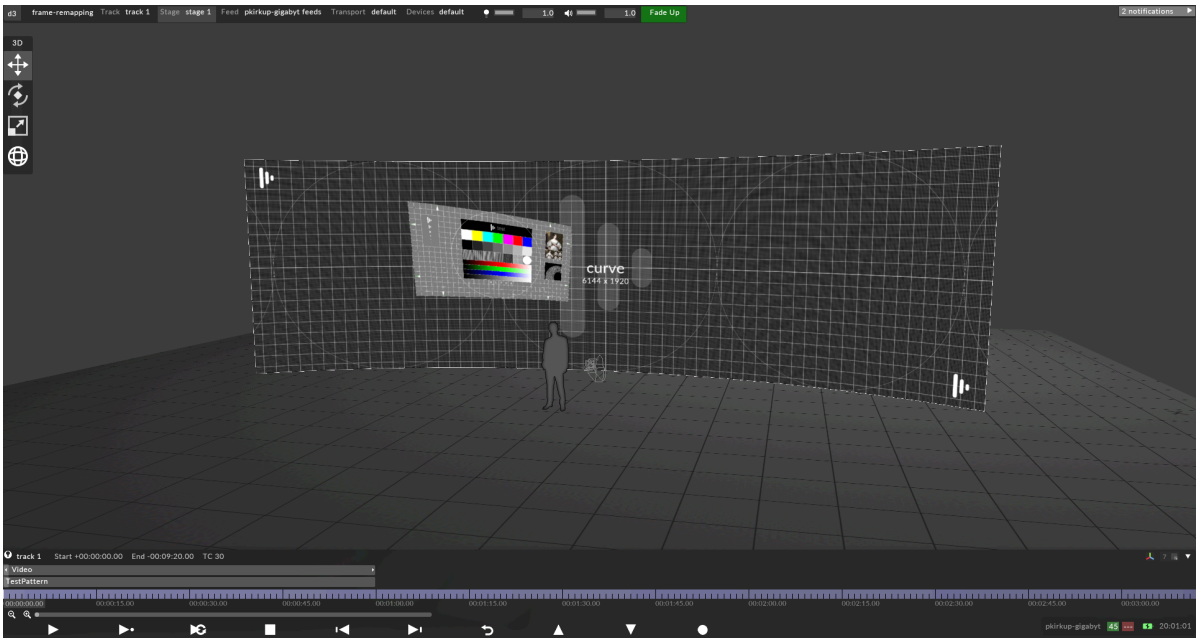
Multi Frame LED Processor Mapping

Frame Remapping (Brompton) and GhostFrame (Megapixel) enable multiple frames from the server to be perceived by multiple cameras shooting out of phase with each other.

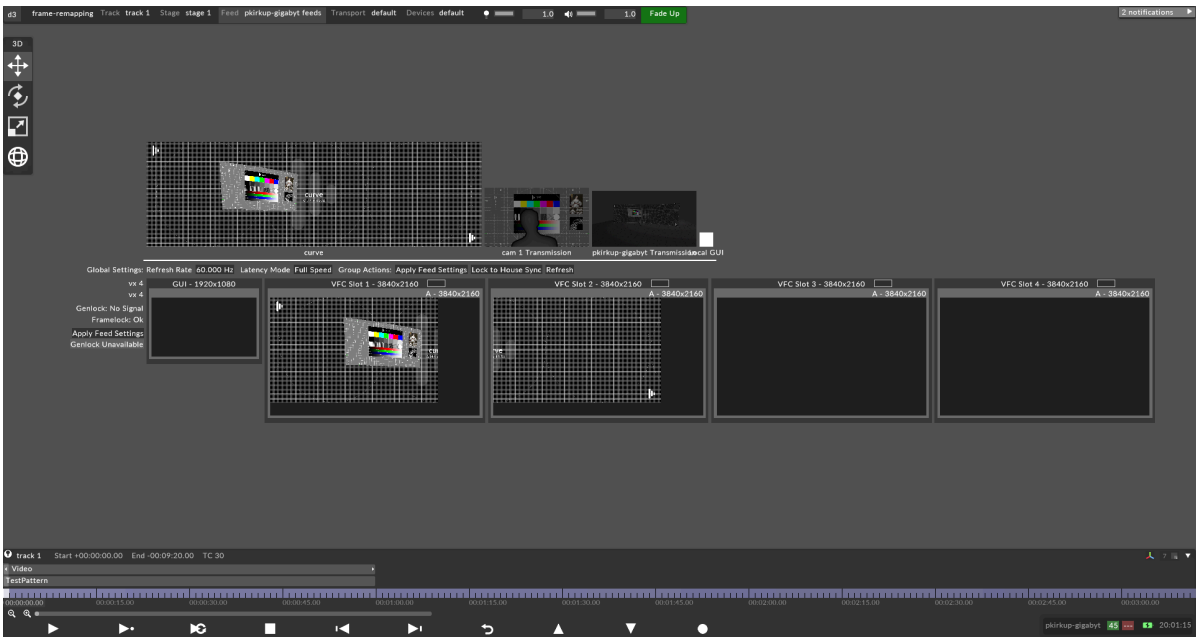
From Disguise's perspective, we are still sending a 4K60 signal to the processor - the processor is doing the job of sending multiple frames one after another, with the frames sourced from different regions within the input raster.

Workflow

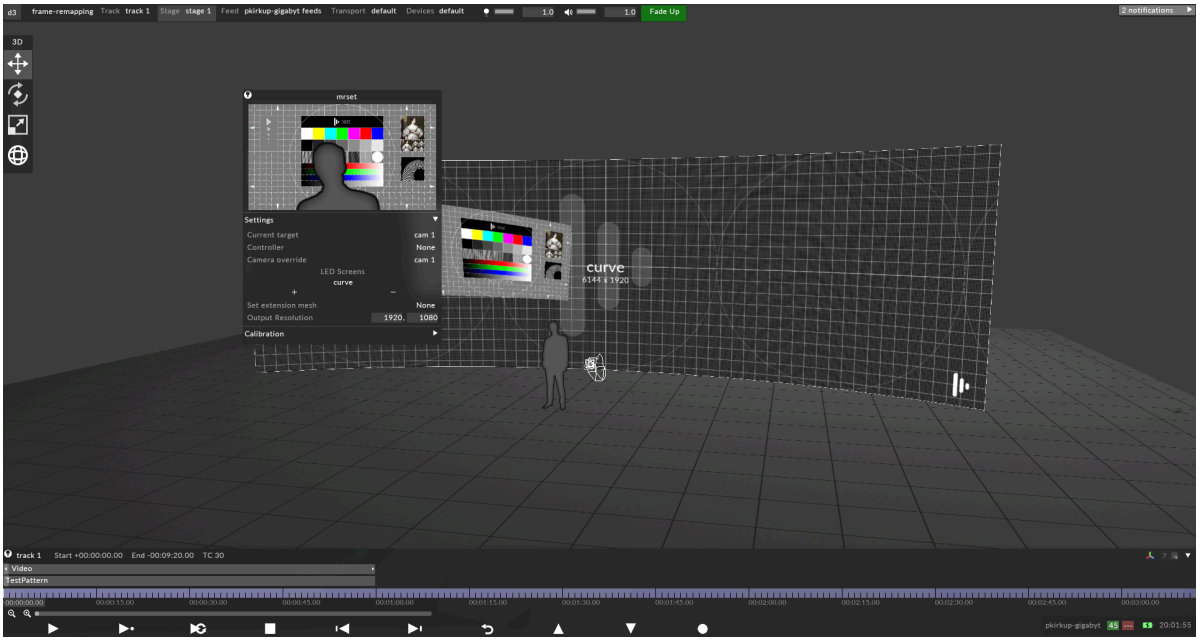
1. Begin with a basic (non multi frame) setup. In our example we've used a large curved LED screen and a single camera:



- This is mapped out of a vx 4 to 2x 4K capable LED processors:

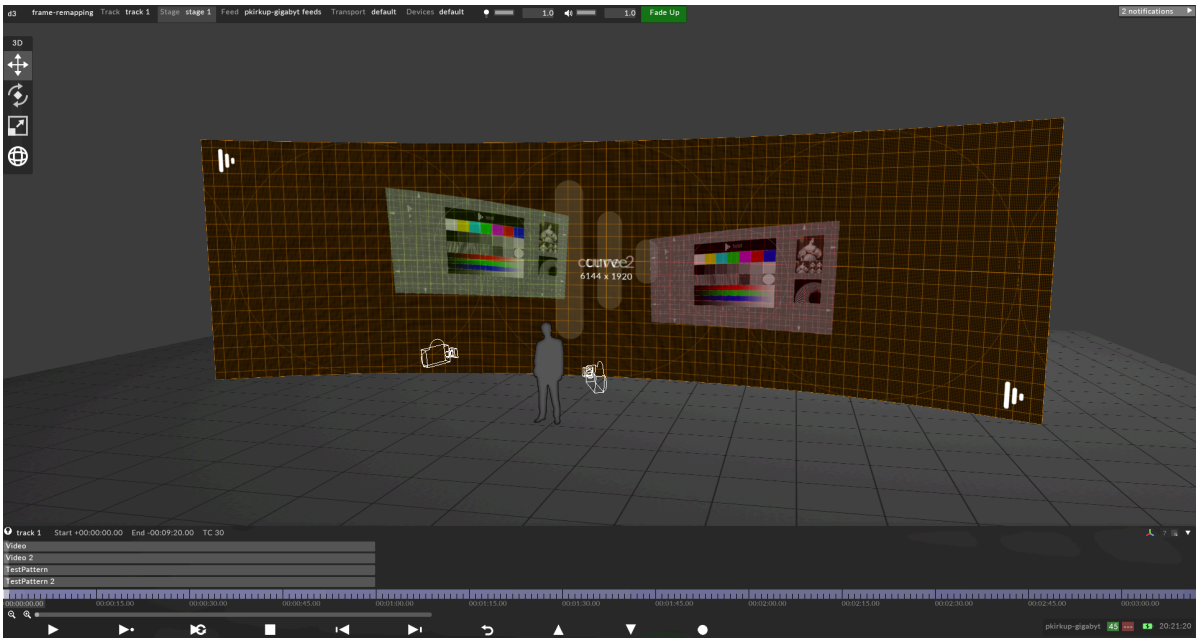


- The LED screen and camera are connected together in a single MRset object. This is the standard workflow, and everything so far is normal:



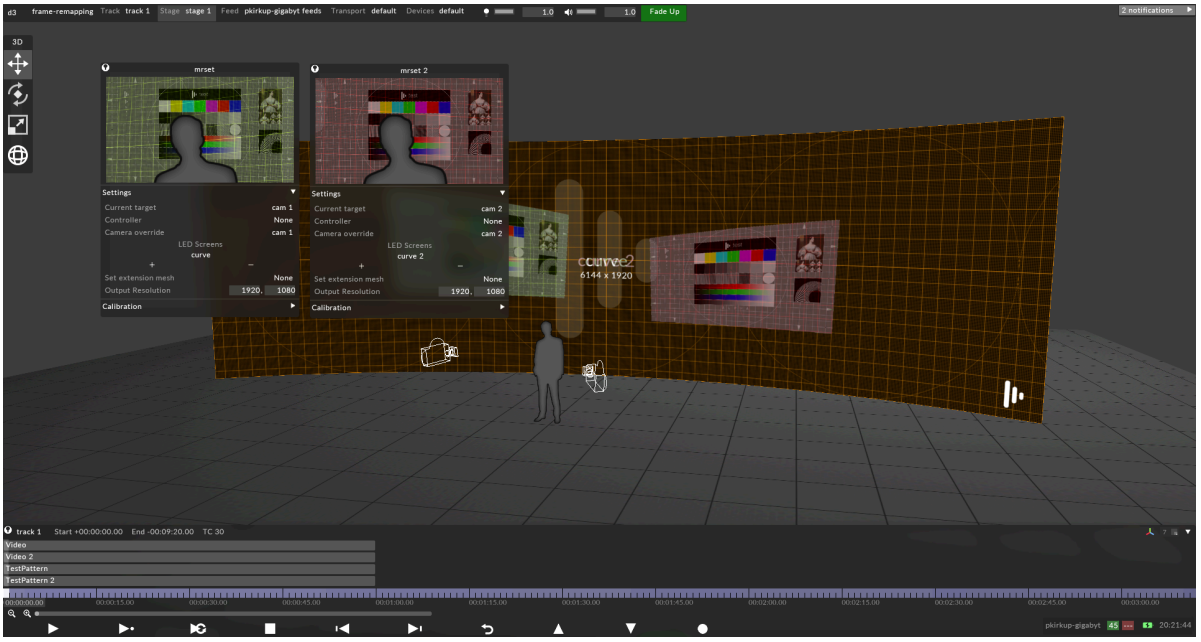
2. Next, place a second LED screen on top of the first one (set to Alpha blend, and with 50% alpha).

- This allows us to pre-viz the overall effect of the frame remapping. Two LED screens placed on top of each other:

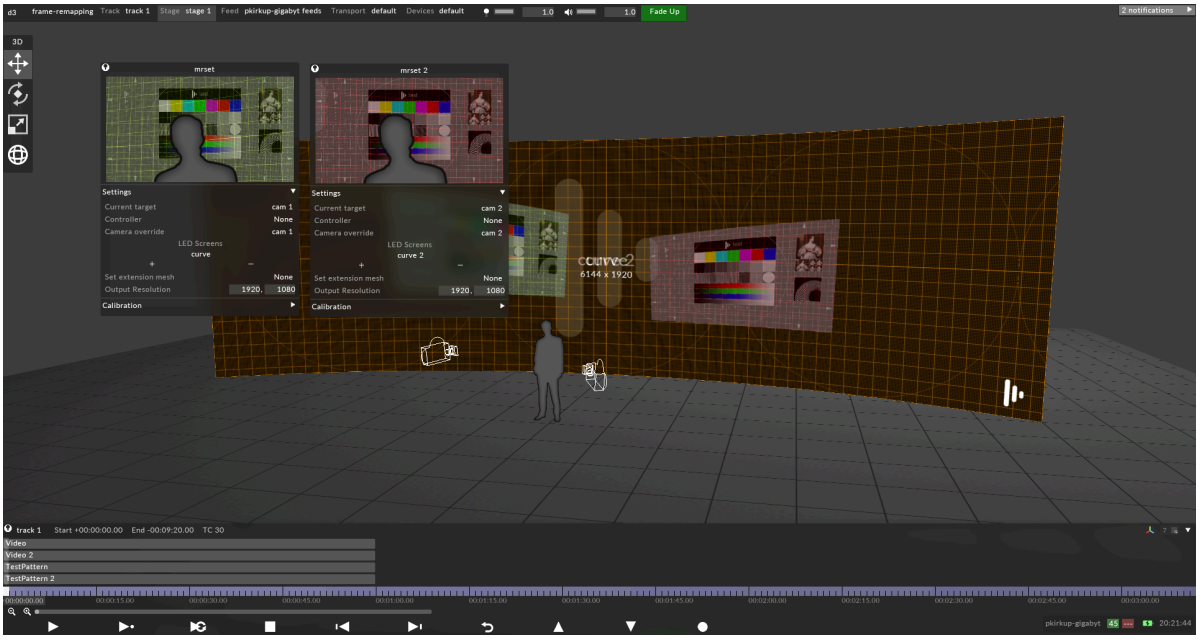


3. Create a second MRset with the new LED screen and a new camera.

- Since we can use two MRsets, the two cameras will only see their own content (although it will be at 50% alpha)::

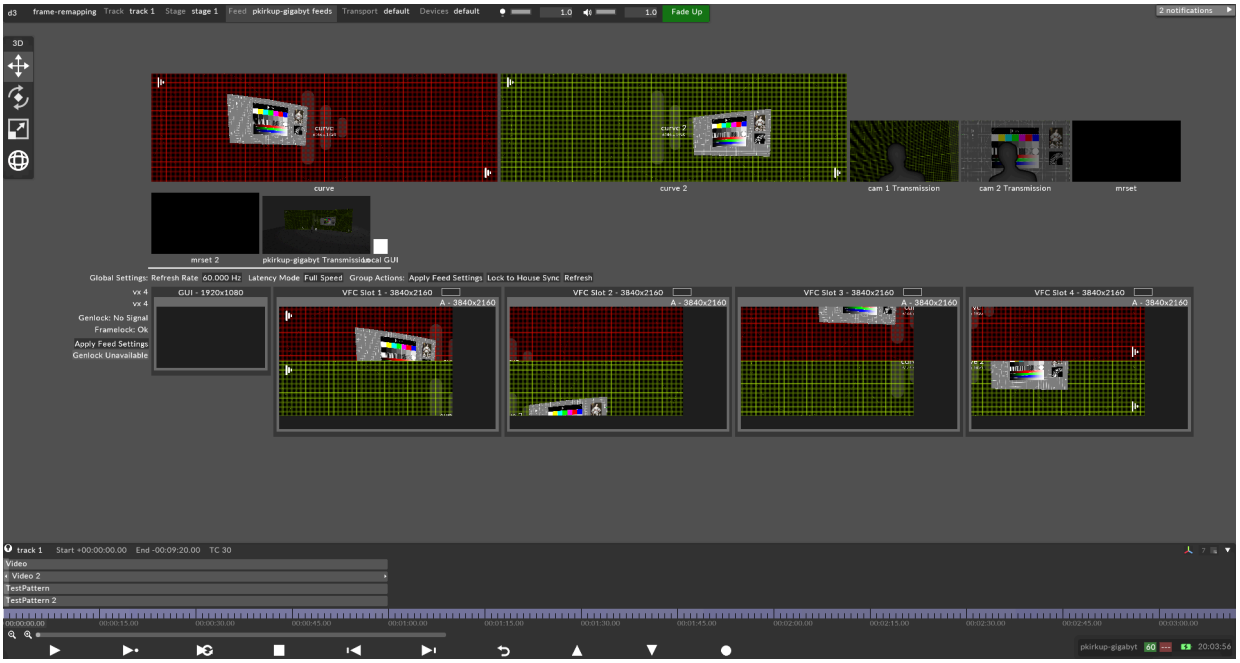


4. **Example#1: Brompton Frame Remapping** Once configured in the Feed output window, the video will be output on top and bottom fragments across 4 outputs on the VX 4 -with the top representing LED screen 1 and the bottom representing LED screen 2.



Example#2: Megapixel GhostFrame

Once configured in the Feed output window, the video will be output on left and right fragments across 4 outputs on the VX 4 -with left representing LED screen 1 and right representing LED screen 2.



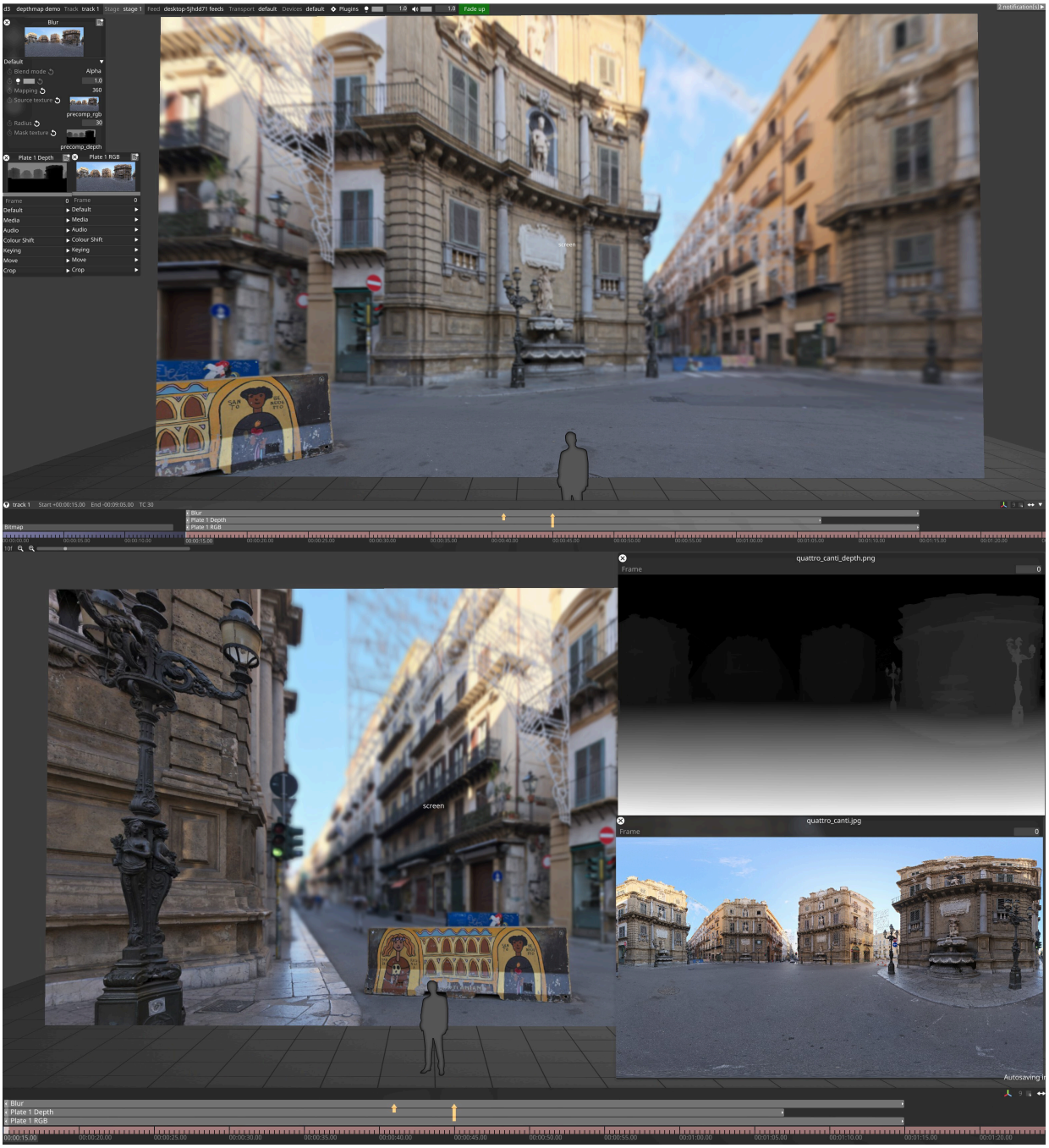
In the LED processor software, the regions sampled for the multiple frames should match the Disguise output feed rectangles.

> **Note**
>
> **Please note**: Different LED processors sample multi-frames differently and the above are two examples. Consult your LED processor manufacturer/user guide for more information about specific workflows. For example: **Megapixel user guide** for HELIOS LED processor (GhostFrame) **Brompton user guide** for Tessera LED processor (Frame Remapping)

Depth of Field Workflow

Depth of Field Using a Blur Layer and Depth Map

This guide provides the direct steps to set up depth-based blur on 2D content using a depth map and the Blur layer in Disguise Designer.



Example images

Workflow

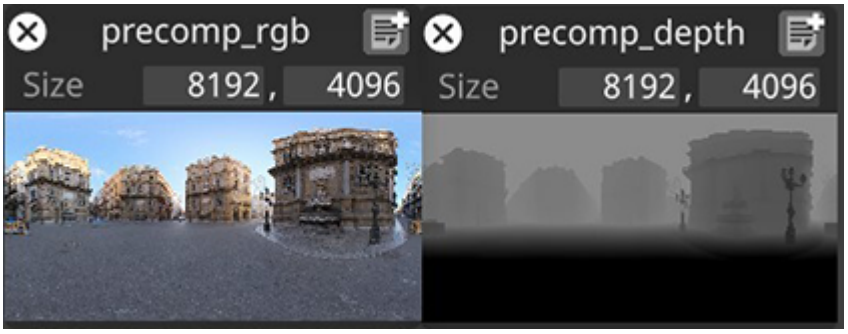
Prerequisites

- RGB content (video/still) imported into Disguise.
- Corresponding Depth Map content (video/still) imported into Disguise.

1. Create 2 Pre-Comp Textures

Also see the PreComp page.

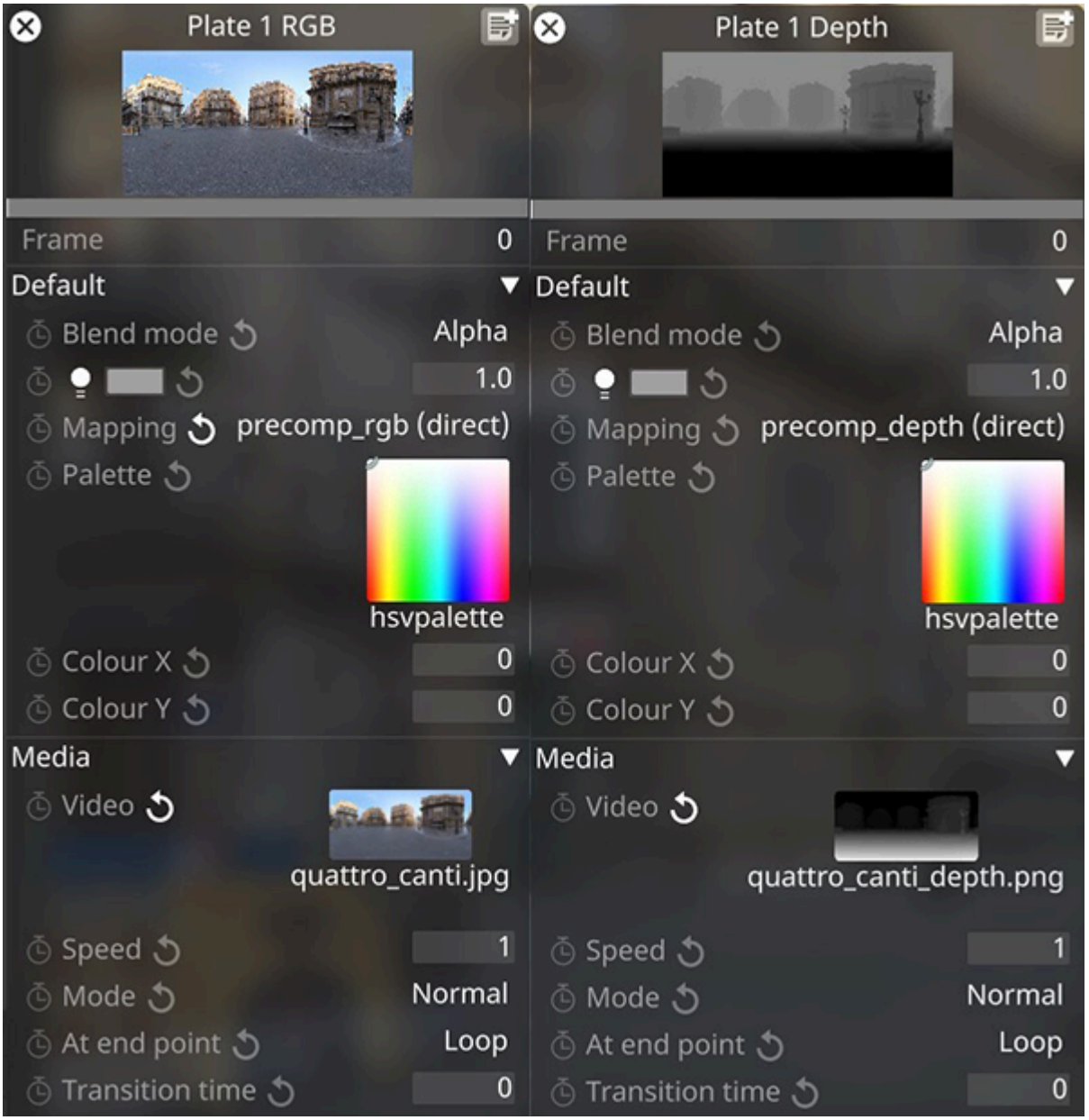
1. Create a **Bitmap Layer**
2. Open up the Media drop down, open the Texture window.
3. Create a **PreCompTexture** (e.g. `precomp_rgb`).
4. Set the **PreCompTexture** Size/Resolution to the Source content Resolution.
5. Create a second **PreCompTexture** (e.g. `precomp_depth`).
6. Set the **PreCompTexture** Size/Resolution to the Source content Resolution.
7. You can now delete the **Bitmap Layer**.



Precomp image

2. Create Video Layers

1. Add a **Video Layer** for your RGB content. Assign the RGB media.
2. Set the Mapping to `precomp_rgb`.
3. Add a second **Video Layer** for your Depth Map content. Assign the Depth Map media.
4. Set the Mapping to `precomp_depth`.

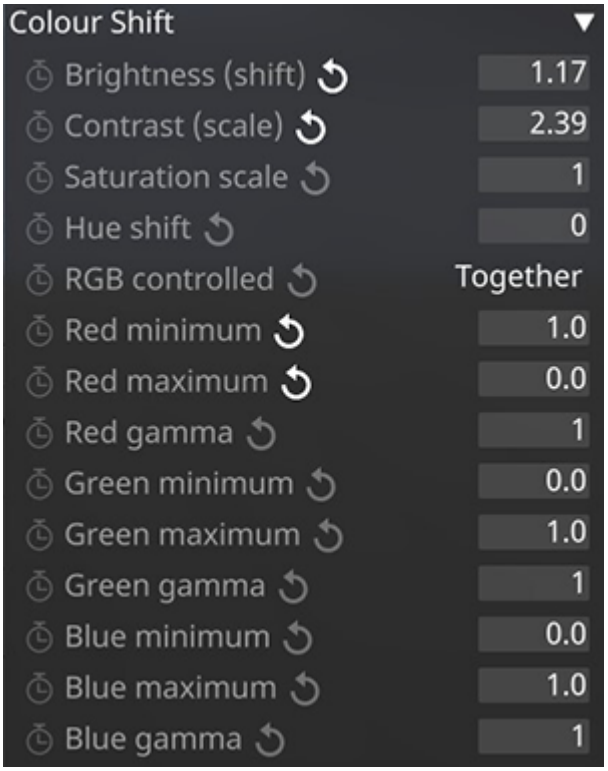


Video layers

3. Configure Depth Map Layer

1. Select the **Depth Map Video Layer**.
2. Go to its **Colour Shift** settings.
3. **Invert** the colours so close objects are black and far objects are white. Use RGB controls to invert.
   - `RGB controlled: Together`
   - `Red minimum 1.0`
   - `Red maximum 0.0`

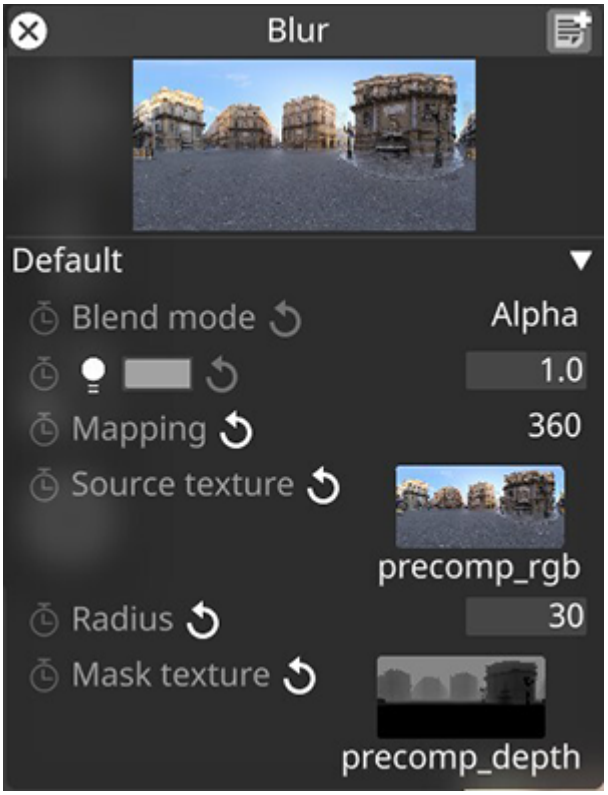
4. **Adjust Brightness/Contrast** in the Colour settings to fine-tune the blur falloff.
    - `Brightness (shift) - Controls the 'distance' of the blur`
    - `Contrast (scale) - Controls the size of the min-max blur`
5. **Fine-tuning** the brightness/contrast of the depth map is essential to achieve the desired look and requires artistic judgment.



Depth Colour Shift

4. Apply and Configure Blur Layer

1. Add a **Blur Layer** effect to your timeline.
2. Creating Mapping (e.g., Direct, Spherical, etc.)
3. Set `Source Texture` to `precomp_rgb`.
4. Set `Mask Texture` to `precomp_depth`.
5. Adjust the Blur Layer's `Radius` parameter to control the maximum blur for the 'furthest' elements.



Blur layer

Result

The RGB content will now have blur applied based on the depth map, with the amount controlled by the Blur Layer settings and the depth map's contrast/brightness adjustments.

xR for VR Workflow

The xR for VR workflow enables the production of Spatial Video (VR180 or VR360) content for consumption in Spatial Computing devices from a typical LED xR production stage, using standard broadcast cameras and lenses.

Once the stage and project are set up correctly for xR, a spherical output can be produced in just a few clicks, using a virtual camera.

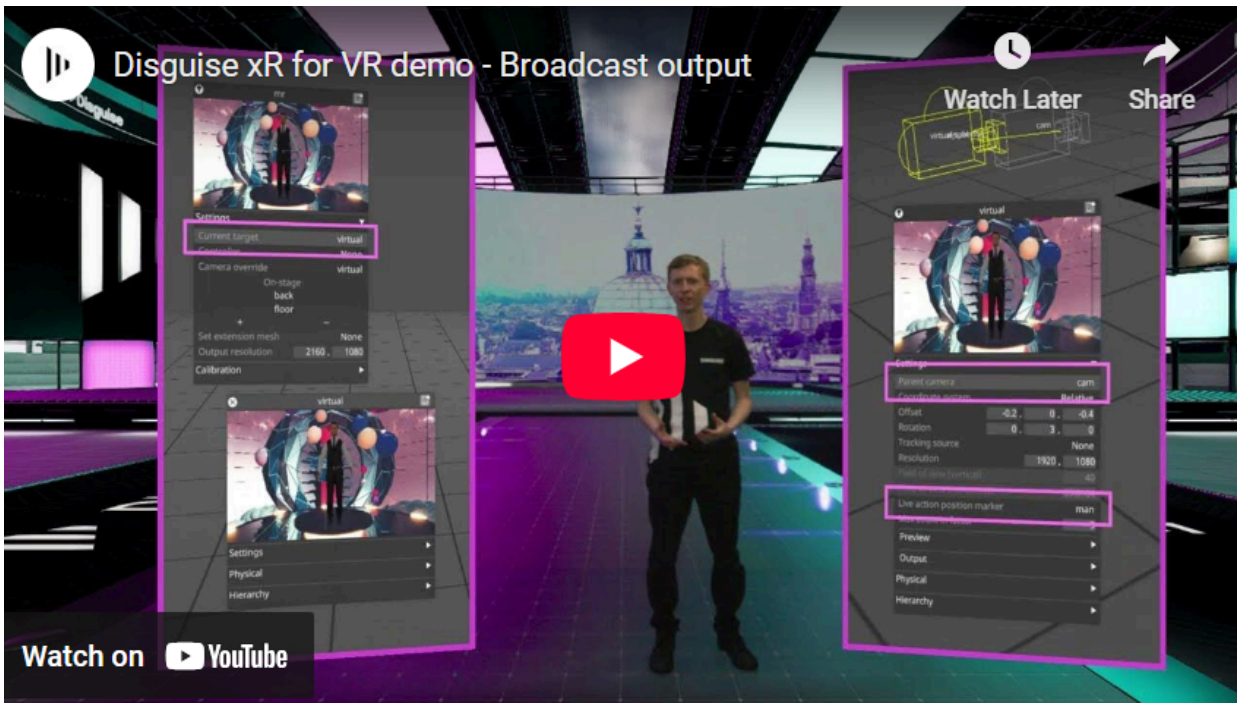
Demo videos

Check out our demonstration of the xR for VR workflow in the two videos below. We recorded simultaneous VR and broadcast outputs, following the process described here.

VR output video



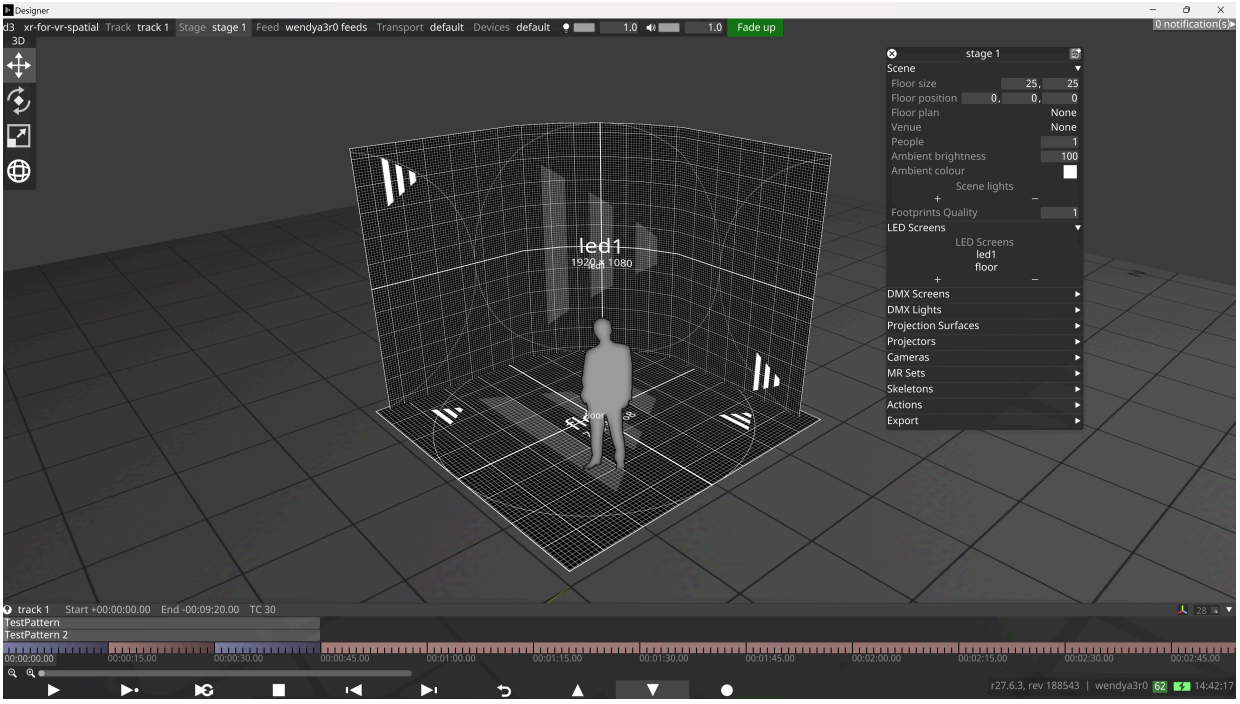
Broadcast output video



Stage setup

The stage must be set up correctly for xR. See xR stage setup for more detailed instructions. In brief, the steps are:

1. Create a standard xR LED stage in the Disguise software. Add an LED wall and floor to the stage.
2. Add a broadcast camera. In this example, it is called “d3cam”.
3. Create an MR set, and add your screens and camera to it.
4. Setup and calibrate the stage ensuring lens, colour, and delays are fully calibrated.
5. Position your broadcast camera so that it covers all participants on the stage from head to toe. You can preview this in the stage visualiser and use the mouse to pan, rotate, and zoom the stage into view.



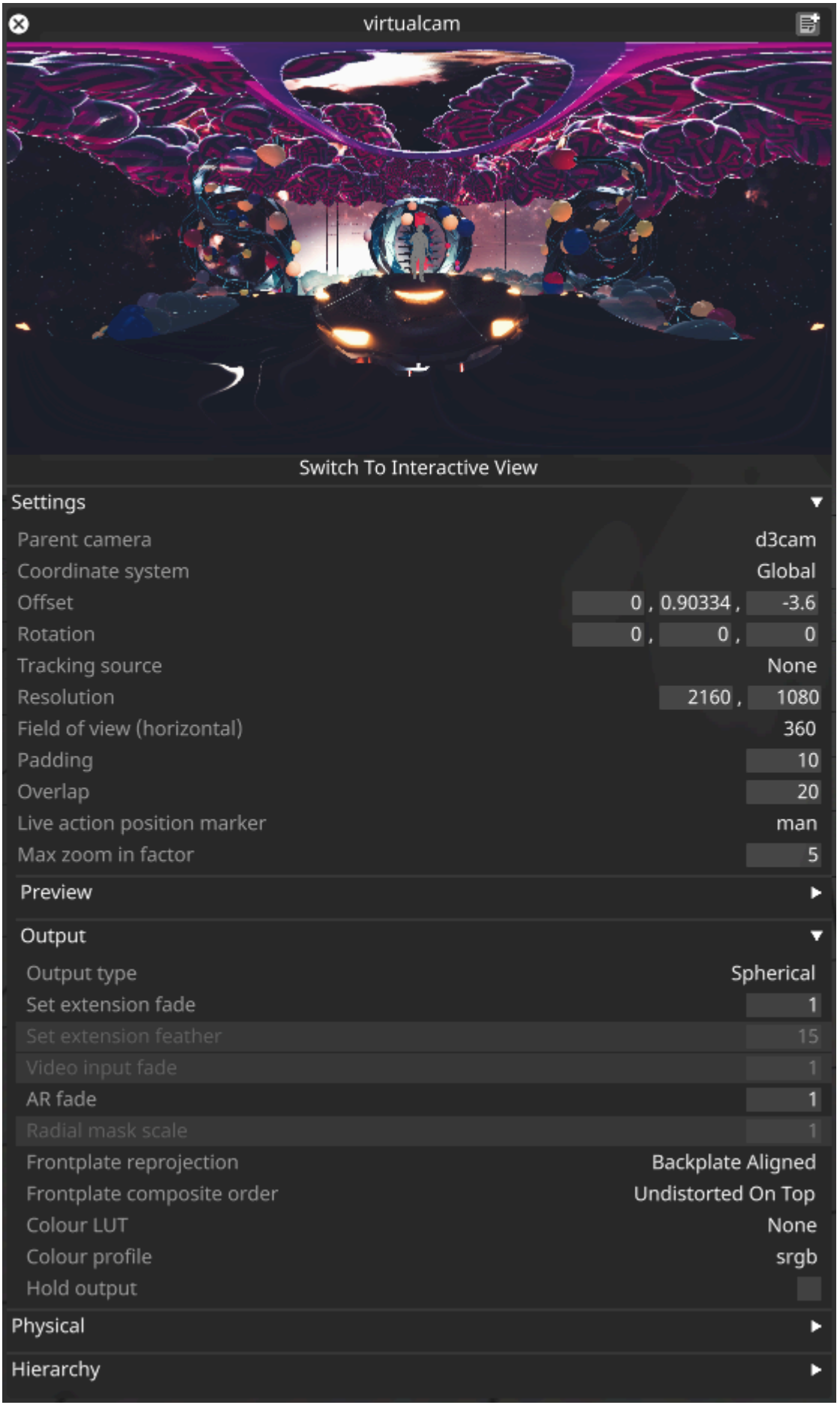
The xR Stage with Test Patterns

Add and configure virtual camera

In Designer, add a virtual camera and configure it as follows:

1. Add a new virtual camera to the list of cameras in the stage.
2. Set the virtual camera as the target of the MR set. This will allow you to see how changing the virtual camera properties affect the mapped content.
3. Set the **Parent camera** of the virtual camera to the Camera object representing the broadcast camera (d3cam in this example).
4. Set the **Live action position marker** to an object in the stage positioned at the average position of real on-stage people and props, for example at the position of the main presenter. This is necessary to to avoid sliding of the people relative to the virtual scene as the virtual camera moves.
5. Set the **Coordinate system** to **Global**, and zero out the pitch (X) and roll (Z) rotations. **It is important that the virtual camera pitch and roll rotations are zero in global space, otherwise the camera horizon won't be aligned with the centre of the output, which may cause the scene to appear misaligned when viewed on VR devices.**
6. Adjust the **Offset** and yaw (Y) **Rotation** of the virtual camera to the desired position. This can be anywhere, although if it is positioned at too large an angle relative to the parent camera the people captured on-stage may appear flat.
7. Expand the **Output** separator, and change the **Output type** to **Spherical**. The camera will now output a VR360 spatial video feed.
8. If desired, the **Field of view (horizontal)** can be set to 180 for a VR180 output.
9. Set the **Resolution** of the virtual camera to the desired output resolution of the camera.

> **⚠ Caution**
>
> By default, the MR set will have a resolution which is more suitable for standard camera feeds. It is likely you will want to set the MR set resolution to match the virtual camera resolution.


virtualcam
Switch To Interactive View
Settings
Parent camera
d3cam
Coordinate system
Global
Offset
0 , 0.90334 , -3.6
Rotation
0 , 0 , 0
Tracking source
None
Resolution
2160 , 1080
Field of view (horizontal)
360
Padding
10
Overlap
20
Live action position marker
man
Max zoom in factor
5
Preview
Output
Output type
Spherical
Set extension fade
1
Set extension feather
15
Video input fade
1
AR fade
1
Radial mask scale
1
Frontplate reprojection
Backplate Aligned
Frontplate composite order
Undistorted On Top
Colour LUT
None
Colour profile
srgb
Hold output
Physical
Hierarchy

Example setup for the virtual camera

Avoiding seams in content

Rendering of the spherical content is achieved under the hood by rendering the 6 faces of a cube, then combining and warping them to the correct spherical output. Certain screen-space effects used when rendering the virtual scene can cause visible seams between the faces. See **UE scene optimisation** for tips on how to avoid this in Unreal Engine.

If some screen-space effects are required for creative reasons, it is possible to mitigate their effects and avoid seams by setting **Padding** and **Overlap** in the virtual camera:

- **Padding**: This value sets a number of additional pixels which are rendered outside each face, then cut away when combining the faces. This can be used to mitigate effects such as blur which exhibit edge effects on the extremities of the rendered image.
- **Overlap**: This value also renders additional pixels outside each face, then smoothly blends between the pixels of adjoining faces on each edge. This reduces the appearance of seams between faces by smoothing over the join, so you don't get a hard line.

> **Caution**
>
> Both padding and overlap will increase the render resolution of each face, and therefore will increase the GPU resources required to render the content. It is recommended to set these to the smallest values required to remove visible seams.

Add VR180 or VR360 Content

Using RenderStream

1. Add a RenderStream Layer to your track.
2. In the RenderStream Layer Editor, select your VR180 or VR360 asset, such as an Unreal project.
3. Add a Cluster Pool of render nodes on the layer.
4. Add a Channel, and set the mapping to the backplate of the MR set.
5. Start the Workload.

> **Note**
>
> The spherical camera output requires 6 viewpoints to be rendered. The number of render nodes required for this workflow depends on the resolution of the mapping, and the complexity of the virtual scene.

Using other 3D modules

Other modules which render 3D content can also be used for spherical camera outputs, such as locally hosted Notch blocks, or the Stage Render or 2.5D layers. Simply set the module mapping to the backplate of the MR set while the virtual camera in Spherical mode is set as the MR set's target.

> **⚠ Caution**
>
> Using local 3D modules rather than RenderStream will require all 6 viewpoints to be rendered on the same local machine. This may impose a large GPU load on the machine, depending on the setup of the module and mapping.

Using a video layer - static camera

If you are not planning on moving the virtual camera, then it is simple to map video content to the camera:

1. Add the video to the project folder. The video should be equirectangular content, formatted in the same way as the intended spherical camera output.
2. Add a Video layer, and select your media.
3. Set the video layer mapping to the backplate of the MR set.

Using a video layer - moving camera

If you want to add video content while also having the flexibility to move the virtual camera, you will need to map the video to a spherical mesh, and use the stage render layer to map this to the MR set backplate.

Add a spherical mesh:

1. In the Stage editor, create a virtual projection surface, name it "sphere", and select the **puffersphere mesh**.
2. Scale the sphere to a large value, e.g. (10, 10, 10)
3. Set an appropriate resolution for the video content
4. Set the virtual projection surface to **Backplate (MR)** in the Render layer.
5. Set the position of the sphere to the desired eye point of the content (not too far from the position of the camera)

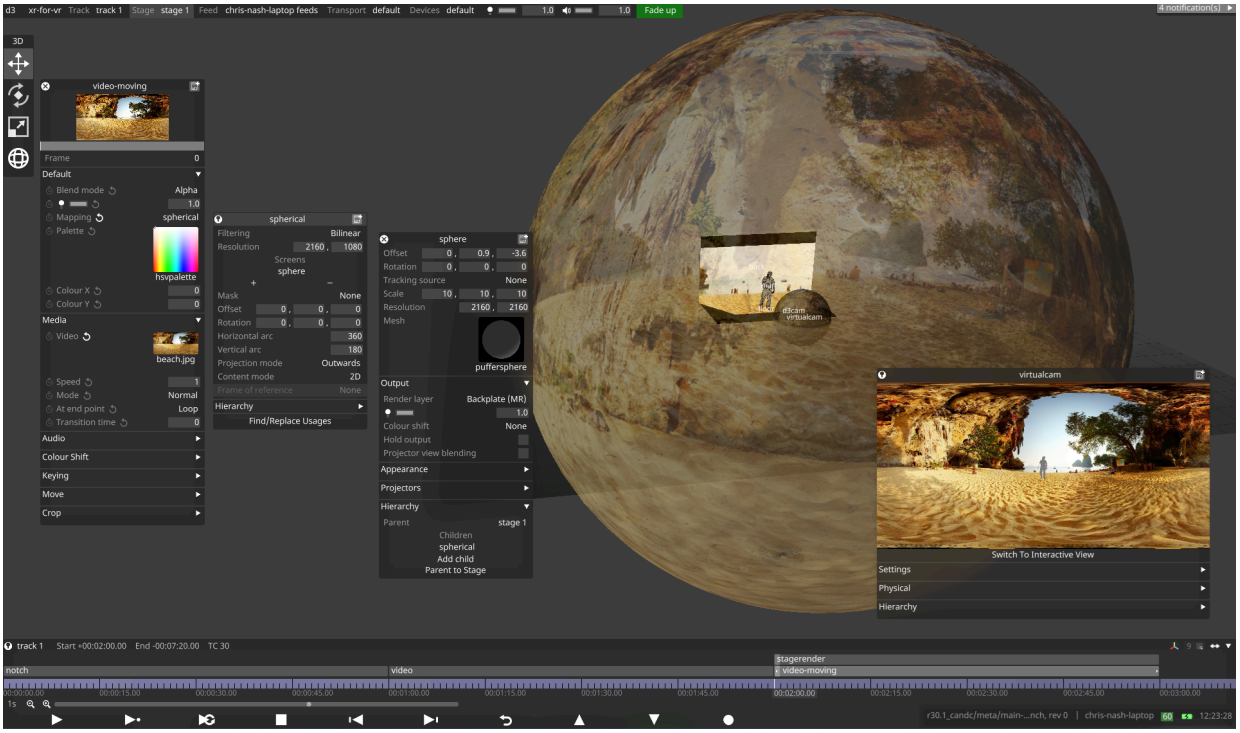
Map the video to the spherical mesh:

1. Add a video layer and select the desired content.
2. Create a new spherical mapping, and set this as the mapping for the video layer
3. Set the resolution of the spherical mapping appropriately for the content

4. Add the sphere projection surface to the list of screens in the spherical mapping
5. Set the spherical mapping as a child of the sphere projection surface, under **Hierarchy** in the projection surface editor

Add a stage render layer:

1. Add a StageRender layer to the timeline.
2. Set the **Render layer** to **Backplate**.
3. Map the StageRender layer to the MR set backplate.

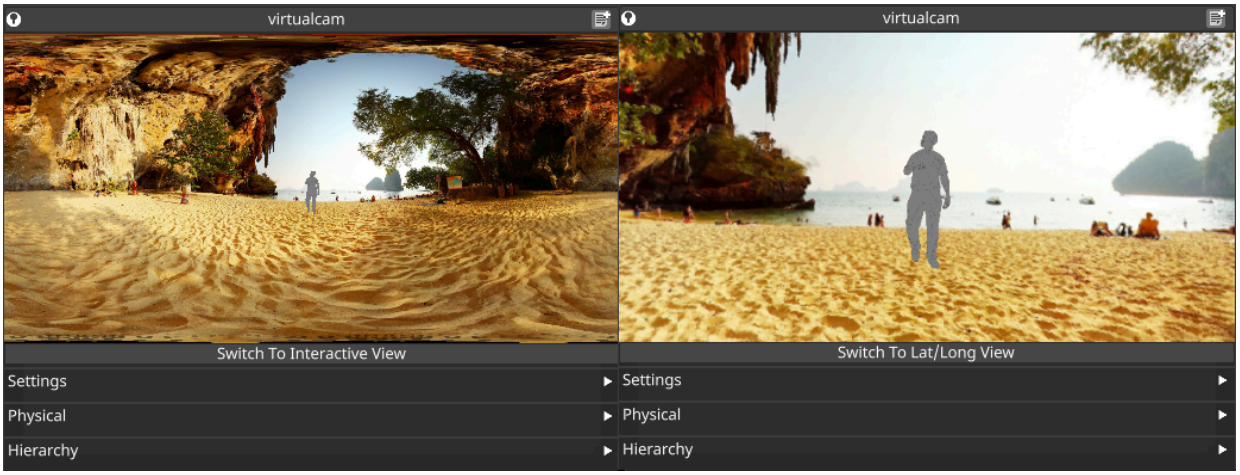


Setting up spherical video content for a moving camera

> **Note**
>
> If you don't want to see the virtual sphere projection surface in the stage, open the visualiser camera editor from the Stage menu. Open the editor for the camera's **Render settings**, under **Stage Render**. Then set the **Virtual object alpha** to zero.

Viewing the content

To preview the spherical camera output within Designer, you can use the 'interactive view' in the camera widget. When the **Output type** of a virtual camera is set to **Spherical**, A button will appear below the camera preview allowing you to **Switch To Interactive View**. By default the camera preview will show the equirectangular Lat/Long output image. Switching to the interactive view gives an interactive perspective view of the scene, which can be navigated by holding the left mouse button within the preview and moving the mouse.



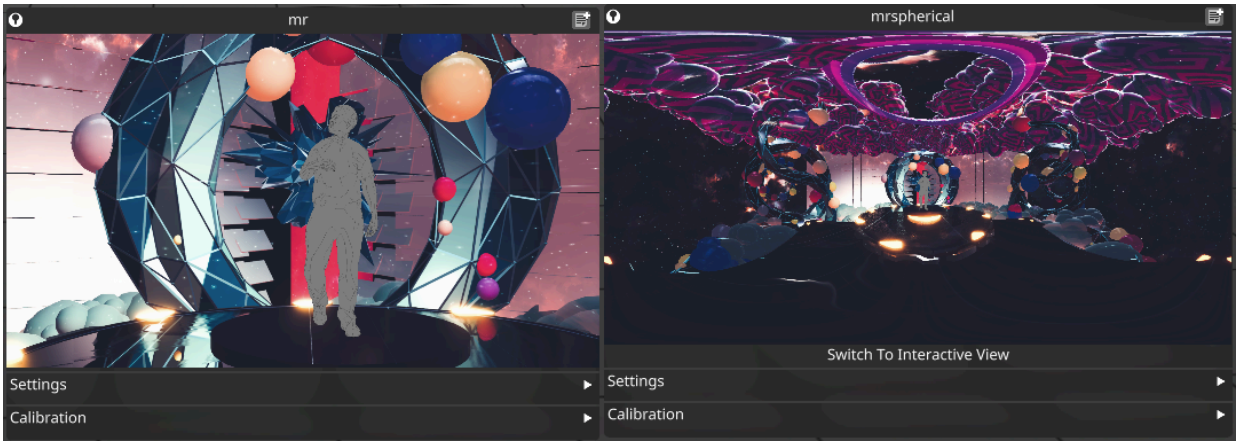
Interactive and Lat/Long spherical camera preview modes

Simultaneous spherical and regular outputs

Sometimes it may be desirable to output a regular perspective camera output alongside a spherical output, to produce a traditional broadcast feed alongside a feed for consumption in VR. Since spherical and regular cameras both project onto the screens from the same eye point, it is possible to produce both types of output simultaneously from a single broadcast camera.

To output spherical and regular outputs simultaneously:

1. Set up the stage and configure a virtual camera in **Spherical** output mode, as described above.
2. Create a second virtual camera, in **Regular** output mode.
3. Set the **Parent camera** and **Live action position marker** in the second virtual camera to match those in the first.
4. Make the second virtual camera a child of the first (from the **Hierarchy** separator), so that both cameras have the same position and rotation.
5. Set the other properties of the second virtual camera as desired.
6. Creating a second MR set, add the LED screens, and set the second virtual camera as the target.
7. Map any content to the backplates of both MR sets. This may require duplicating layers to enable mapping the content to two mappings with different resolutions. However, in the RenderStream layer this can be achieved by adding a second channel.
8. Ensure the content mapped to the Regular output is above the content mapped to the Spherical output. This is because the content projected onto the screens is likely to appear at higher resolution for the Regular output.



Simultaneous spherical and regular outputs