<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RenderStream Layer</td>
<td>60</td>
</tr>
<tr>
<td>RenderStream and Unreal Engine</td>
<td>74</td>
</tr>
<tr>
<td>RenderStream and Unity</td>
<td>75</td>
</tr>
<tr>
<td>RenderStream and Notch</td>
<td>87</td>
</tr>
<tr>
<td>RenderStream Failover</td>
<td>90</td>
</tr>
</tbody>
</table>
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. Video receive delay
3. Taking observations
4. Spatial Calibration
5. Spatial Tracker delay
6. Colour Calibration
7. Managing set extension
8. Virtual Cameras
9. Camera switching
10. Cluster Rendering
xR Project Setup

This topic will explain the basic principles of setting up a project for use with the xR workflow.

Overview

This topic will explain the basic principles of setting up a project for use with the xR workflow.

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

Overview
This topic covers setting up your virtual stage and cameras in the disguise software 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 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 inputs to the 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 the disguise software to a the virtual camera.
12. Monitor the incoming data from the camera tracking system to ensure it is being received in time.
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**.
MR Set

The MR Set object is used as the ‘container’ for the stage elements used in the xR workflow.

Overview

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:** 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

   📌 **Please note:** 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.

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

**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

Overview
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 the disguise software. 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
   1. Click Delay Calibration
2. The calibrate delay function is pre-populated with the current camera target, but you can add or remove different combinations of cameras.

4. Click Capture

5. The disguise software 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

Overview

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 the disguise software 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


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.

   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.

   The disguise software 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.
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 the disguise software is interpolating between different zoom positions, zoom in and out on the camera and watch the MR Set. If the disguise software 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 the disguise software differentiate which screen the blob was seen on.

You can rectify what blobs the disguise software 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

For additional tips for taking observations, please visit the xR hub on the disguise community
xR Spatial Calibration

Overview

This topic covers the basic steps of aligning the physical and virtual worlds within the disguise software, the defining attribute in the xR workflow.

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.

Please note: For information on calibration tips & tricks, please visit the xR Hub of the disguise community portal.

Concepts

A Calibration is a set of data that is contained inside of 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.

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 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 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 creates the lens intrinsics file. Each new zoom and focus position will create a new Lens Pose.

There is no need to assign a zero point in the disguise software. 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.

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.

**Please note:** For information on calibration tips & tricks, please visit the xR Hub of the disguise community portal.

**Workflow**

Prior to beginning spatial calibration, ensure that:
The xR project has been configured with an MR set, accurate OBJ models of the LED screens, and tracking system if being used.

The cameras, LED processors, and all servers are receiving the same genlock signal.

The outputs have been configured and confirmed working.

The Delay Calibration has been completed.

**Virtual Set Preview Setup**

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 feed output to send content to the LED screen
4. Create a virtual line up layer
5. Assign a Spatial mapping set to Frontplate for this layer.
   This will show the representation of the virtual set and will move/deform during the calibration process.
7. Use CTRL+Left Click on the header of the MR set editor to pin the window to the GUI. The preview will show the current active camera view, the AR Virtual Lineup overlay, and the test pattern mapped to the LED screen outputs.
8. For calibration of a range of focus levels, take multiple observations at each zoom level for each of the focus levels. Make sure the zoom is locked for each set of focus observations.

**Primary Calibration**

The Primary calibration is the set of observations that define the virtual world's positional and rotational offsets to accurately match the real world, as interpreted by the camera lens.

The Primary observations are defined by the most consistent pair of zoom/focus values within the data set.

1. Open the spatial calibration editor by left clicking spatial calibration from 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 in show.

6. In the calibration editor, left click Lock Zoom and Lock Focus to fix the current zoom and focus values.

7. All primary observations will be grouped based on the most consistent zoom/focus combination in the list.

8. Use the Live Blob preview to set blob size and spacing proportional to your camera lens and LED resolution.

9. Begin taking primary observations. This will calibrate the offsets between the tracking system and coordinate system within disguise. Primary observations will be notated in the list with a P indicator.

10. Take a minimum 5 good observations from different camera angles/positions that will be used in show.

11. Utilize the tools under the Debugging tab to determine if an added observation is good or bad.

**Please note:** For more in depth information on how to take quality primary observations, please refer to our advanced guide on the [disguise community portal](https://community.disguise.com).

**Secondary Calibration**

The Secondary Calibration is a set of checkpoints along the zoom and focus ranges of the camera lens. Each checkpoint is an individual Lens pose comprised of a specific zoom and focus value. Disguise will interpolate between the defined lens poses as the lens zoom and focus values are changed in show.

1. Return to the "hero" or most base camera position.

2. Unlock Focus.

3. Adjust the focus value so it is a new value.

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.
6. Unlock Zoom.

7. Zoom in at a predefined interval, for example 10% in.

8. Lock Zoom.

9. Capture several different observations of varying focus values.

10. Repeat steps 7-9 until 100% zoom has been achieved.

11. Zoom the camera in and out and adjust the focus as needed. View the MR transmission 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.

**Please note:** For more in-depth information on how to take quality primary observations, please refer to our advanced guide on the [disguise community portal](#).

---

**Properties**

**Settings**

The MR set to be calibrated, which will contain:

- All LED surfaces that will display virtual content.

- The indirection controller containing the current camera that is being calibrated.

**Blob Settings**

- Adjust the settings of individual observations.

- Adjustments to the size in pixels include:

  - each blob that will be displayed for that observation.

  - how many pixels apart they will be.

- The option to exclude specific screens from individual observations.

For example, many stages are calibrated with the floor excluded from the calibration due to the steep viewing angle.

**Please note:** The 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.

**Observations**

A list of all observations saved within the camera object’s calibration.

The data includes:

- The tracked zoom, focus, positional and rotational data of the camera at the time of the observation capture, as sent by the tracking source assigned to the camera object.

- The categorization of Primary or Secondary.

- A status indicator of if the observation is enabled or currently active within the calibration.

- The list number of each individual observation.

**Calibration**

Adjusts global settings regarding the calibration.

Includes:

- 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.

- Read: Will read captured images to recreate a calibration offline.

**Calibration Results**

Provides a list of all lens poses created by differing sets of primary and secondary observations.

**Debugging**

- Observation Debugger

- Plot Calibration Errors

- Show 3D Observations
Tracker Distortion Compensation

This property allows for the potential correction of non-physical errors in the tracking system.

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**: 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**: Before this setting was added to the UI, the tracker distortion compensation method was Matrix, which was hard coded in. A byproduct of Matrix is it also allows for more distortions.

Debugging

There are many contributing factors that may result in a poor spatial calibration. Below is a recommended workflow for troubleshooting the possible causes.

**Primary Calibration**

1. Carry out initial checks:
   - 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.
   - 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.
   - 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).

The lens distorts content in a way not captured by our model. For example, anamorphic lenses are not currently supported.

Feed output mappings are incorrect

**Please note:** Please see our [disguise community portal](#) for our guide on tips for taking Observations.

3. 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.

4. 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.

5. If none of these issues are found, go through the following steps to create a project diagnostic:

- Take an observation which shows this issue (for example, it looks fine in the observation debugger but the Virtual Lineup Layer is not aligned)

- Leave the camera in the same position that the observation was taken

- 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

- If possible, take a screenshot of the camera feed with the blob pattern preview displayed on the screens

- Export a diagnostic of the project, and along with screenshots of the MR set preview, send to support@disguise.one

Please note: For information on calibration tips & tricks, please visit the xR Hub of the disguise community portal.
xR Stage Alignment Overview

Overview
This topic covers the basic steps of how to align your stage and run calibrations for the xR workflow.

Workflow
1. Test pattern layer setup
   a. Add test pattern layer using Direct mapping to LED surface
   Configure feed output to send content to LED screen

2. Virtual lineup layer
   a. Add a virtual line up layer mapped to the MR set via a spatial mapping set to front plate.

3. Calculate your video delay
   a. Determine the video receive delay value using the delay calibration editor in the MR set.
   b. Add MR set and cameras to the delay calibration editor.
   c. Click Capture and once completed select the single white frame from the delay calibration editor, if this does not happen, check the system is correctly Genlocked.
   d. Test the screens are in sync using the Check sync tool in the delay calibration editor.

4. Configure observations.
   a. Open the spatial calibration editor by left clicking spatial calibration from the MR set.
   b. Ensure the correct MR set is selected and the camera you’re calibrating is the current target.
c. Verify the camera tracking system is outputting the correct data as you test camera movements.

d. Set a real world camera position by moving the physical camera on the stage.

e. Ensure the screens you wish to calibrate are included in the observation.

f. Use the Live Blob preview to set blob size and spacing proportional to your camera lens and LED resolution.

g. Set your base zoom position.

h. Focus the camera then lock zoom & focus if desired.

i. Add observations. Repeat for multiple angles & positions.

5. Calculate tracking delay

   a. Open delay calibration editor

   b. Expand the tracker delay tab.

   c. Open the tracking delay tool and click settings.

   d. Left click Capture to initiate a capture. This captures a set of frames whilst the camera is moving.

   e. Move the physical camera along one axis.

   f. The captured frames are displayed with the captured grid lines and the virtual overlay.

   g. Adjust delay tracker by selecting the numeric value until the virtual overlay and real content align.

6. Colour calibration

   Please note: Perform standard colour calibration and balancing setup routines for all LED screens, cameras and lighting as required prior to running the colour calibration process in The disguise software.

   a. Open the MR set.

   b. Expand the calibration tab.
c. Left click Colour Calibration to open the Colour Calibration editor.

d. Add cameras to the calibration. It is possible to calibrate a single camera, or multiple cameras together.

e. Left click **Capture**, in the pre-calibration tab.

f. Increase frames per test until pure colours are represented in the preview.

g. In the calibration tab, select the number of colour steps to capture.

h. Left click run.
Spatial Tracker Delay

Overview

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 MR set 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.
Please note: 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

The disguise Colour Calibration function is a tool for aligning how colour is represented across the system.

Overview

The goal of xR is to create a complete environment that includes real elements and virtual elements.

The disguise colour calibration is a tool for aligning how colour is represented across the system. If the colour between the physical LED panels and the virtual set extension do not match, the illusion of a single, visually-cohesive environment will be broken.

This process is aimed at reconciling the differences across all screens: it will create and apply a LUT file of the LED screens to match with the Set Extension colour. Different products across LED manufacturers will vary slightly in their representation of assigned colours - the disguise colour calibration allows them to come together seamlessly by determining the minute color differences of each LED surface, even between varying LED manufacturers.

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. Enter the Colour calibration menu.
6. Ensure the correct MR set to be calibrated have populated in the Settings tab.
7. Expand the Pre-Calibration tab.

Please note: For information on white balancing cameras, LED product and lighting ahead of the calibration, please visit the xR Hub of the disguise community portal.
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. Hit capture.

9. Step up incrementally in the Frames per test field until the colours in the preview read and saturated true representations of each colour. This ensures that all steps of each colour range are correctly captured by the camera.

10. Expand the Calibration tab.

11. 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.

12. Expand the Calibrated LUTS tab to toggle on and off each LUT if needed.

13. 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.

   **Please note:** Each physical camera will require its own colour calibration, the LUTs are switched automatically by the indirection controller when it switches cameras.

   **Please note:** 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.

   **Please note:** 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.
Properties

---

**Settings**

MR sets: The MR set to be calibrated. One should be selected.

Cameras: The cameras to be calibrated. All to be calibrated should be selected.

Remove Current Colour Effectors: Will remove all applied LUTs and software colour adjustments from camera objects.

---

**Pre-Calibration**

Capture: Clicking will capture the set number of frames on output.

Video Receive Delay (sec): The delay value of the incoming video frame that is determined in the Delay calibration process.

Frames per test: The user defined number of frames to be captured for each displayed colour value.

---

**Calibration**

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.

Colour steps: The range of values each colour will be captured at. The larger the number, the more tonal values of each colour assessed. Thus, the longer the calibration. A known good value is 14 but for testing, a low value can be used for a faster calibration result. For newly constructed stages it is recommended to experiment with multiple step values.
Screen fill factor: The area of the screen that is displaying colour. A value of 1 is 100% fill.

Calibrated LUTs: This tab will show all created LUT files for each camera/screen pair and can be toggled on and off using the Enabled button.
Set extension

Overview
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 this 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.

Properties
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.

Overview

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.

Please note: Delays must be perfect to then get in sync switching

Controller Type

Inside disguise Controller we have 6 Indirection Controller Types:

- DMXIndirectionController
- ListIndirectionController
- MachineListIndirectionController
- ManualIndirectionController
- 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.
   - The slot is the next channel immediately after the bank, so you must specify 2 channels in your DMX sender.
6. 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 camera
**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

**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: osc/objects/camera/cam1.apx

   **Please note:**
   Confirm that the labels for camera objects and the OSC inductions match and are named using the same format.

**UDPIndirectionController**

This allows you to force a camera to be active over UDP.
1. Create 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; 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 the disguise software 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 Layer editor 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.
Multi Frame LED Processor mapping

Frame Remapping (Brompton) and Ghost Frame (Megapixel) allows multiple frames from the server to be perceived by multiple cameras shooting out of phase with each other.

Overview
Multi Frame LED processor mapping

The concept of Multi Frame LED processor mapping is to introduce multiple frames from the media 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:

   ![Image of LED screen setup]

   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 top representing LED screen 1 and bottom representing LED screen 2.
Example#2: MegapixelVR Ghost Frame

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.
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.
Virtual Zoom

Overview

Using Virtual Cameras allow xR to overcome the limitation of only being able to show content from the position of a real camera, which can be somewhat restrictive, particularly for small xR stages.

Assuming that the performers/real content are captured within the frame, using a virtual camera allows you to control the position, rotation, and zoom of the virtual camera, which then reduces the size of the captured frame rendered to the set extension around the outside.

A dolly movement away from the subject in a similar manner can be created as well when using a virtual camera, allowing you to recreate shots from as far away as you like even with a small stage.

The idea is to have your real camera inputs adapt/move and shrink further than the physical zoom of the camera itself. This effect greatly increases the possible range of shots and effects from small xR stages.

Workflow

1. Ensure you have an MR set created
2. Ensure delays are calibrated
3. Add an object to the stage
4. Set live action position reference
5. Map content
6. Add an AnimateCameraPreset layer
7. Set Camera positions
8. Configure lens distortion
Please note: Virtual camera moves should only be done when the physical camera is static. Reprojection errors will occur when the physical camera is moving, and may create a "wobble" or other artifacts in the content.

Example

1. Add an object to the stage representing the average position of the real objects in the scene such as a person.

2. Assign this object to be the live action position reference in the virtual camera's editor.

3. Add an AnimateCameraPreset layer to the timeline.

4. Untick Animate camera option in the layer. This will stop the layer overwriting the real camera position received from the tracker.

5. Set camera positions by making changes to the virtual camera within the AnimateCameraPreset layer. You can choose whether the virtual camera coordinates are applied relative or global.

6. If there are significant lens distortions, these might be more apparent for the virtual camera moves. If so, consider reducing the Radial mask scale in your Camera Settings.

Additional information:

When using virtual zoom and while moving the real camera, you may see the set extension detach due to latencies in the system as well as the camera's movement. This will not be as noticeable when using a local Notch block. However, when using RenderStream there is an extra delay we have to account for: the Camera Plate Reprojection Distance. Normally we reproject the content to match the new position of the camera, but with virtual zoom we don’t do this because we want to keep the set extension in the same place.
RenderStream

RenderStream is the proprietary disguise protocol for controlling third party render engines from the disguise software.

Overview

RenderStream allows for the sequencing and sharing of content from a third party render engine to the disguise software, much like how Notch works onboard a gx, albeit running on the rx, our proprietary external render node.

This topic is intended to outline the basic steps involved in configuring RenderStream for use with Unreal Engine, Unity, and Notch. Once the configuration steps explained here are complete, sequencing of the third party render engine can be accomplished via the RenderStream Layer within the disguise software software..

Uncompressed vs. Compressed

RenderStream Uncompressed requires the use of a 25G Mellanox network interface to stream uncompressed, 10bit video data. To test RenderStream Uncompressed appropriately, access to multiple machines and the networking equipment specified is required. Testing RenderStream Uncompressed also requires both a RenderStream Send License and a RenderStream Uncompressed License.

RenderStream Compressed provides consistent content quality with advanced H.265 compression. You can now choose high frequency compression, and ensure that even the finest details of your content are shown via our superior, reliable networking.
Plugins

In order to communicate with disguise, both Unreal Engine and Unity require the installation of a plugin on the render node. Visit the disguise Github for the latest plugins.

Cluster Rendering

**Warning:** For cluster rendering it's recommended to use render nodes from the same disguise product range, e.g. all rx series machines. Mixing of machines from different product ranges is not recommended and is unsupported. It is acceptable to mix rx and rxII types however.

Here are just some of the benefits of using Cluster Rendering:

1. Cluster Rendering allows you to span your render engine content over more than one disguise server by scaling out real-time content up to an unlimited capacity.

2. Each machine will let you render a fragment of your final content frame to increase the render power and get your content onto your displays at your desired quality.

3. Use Cluster Rendering to render real-time content of the highest quality, detail and framerate without worrying about GPU power.

Cluster Rendering is configured within the disguise software using the RenderStream Layer.
RenderStream Layer

The RenderStream Layer is used to control third party render engines running externally to The disguise software.

Overview

The RenderStream Layer is used to sequence remote instances of Notch, Unity, and Unreal Engine.

Layer Properties

There are four main sections within the RenderStream Layer:
Media Thumbnail

This area will display an image of the active stream being received. Note that in a multi-channel environment, this will only show the Camera in UE that has the RS component attached to it.

![RenderStream: Running](image)

Config

This section of the RenderStream Layer is where you will configure properties such as Asset and Cluster Pool, as well as monitor workload and engine health in real time.

<table>
<thead>
<tr>
<th>Config</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Framerate fraction</td>
<td>1</td>
</tr>
<tr>
<td>Workload</td>
<td>Not started</td>
</tr>
<tr>
<td>Receive health</td>
<td>No streams</td>
</tr>
</tbody>
</table>

Framerate Fraction

The framerate of each stream is determined by the global refresh rate set within d3. The Framerate Fraction setting indicates the fraction of the framerate this layer renders at.
The options are 1, 1/2, and 1/3.

Cluster Workload

The Cluster Workload window contains the elements required for configuring a RenderStream Cluster.
It contains the following sections:

--- Configure

The Configure section contains these elements of a RenderStream Cluster:

- **Asset**
- **Cluster Pool**
- **Cluster Assigner**
- **Channel Mapping**
Bandwidth Estimate

---

Control

Asset

This is the Render Engine project or Notch block that will be controlled by the layer. In order for the asset to be visible here, it must be present in the RenderStream projects folder.

Right click on the asset name to open the Asset properties editor:

Asset Properties include:

- **Asset Discovery Name**: (name of UE folder/project)
- **Target Engine**: name of the Render Engine
- **Engine Settings**
- **Colour shift**: apply color correction to entire asset
- **Content Source Machine**: The render node that is running the render engine project. This is the machine that the project will be synced from.
- **Available**: Status indicator displaying if the Asset is available. Not a tick box.
- **Channels**: displays active channels defined by the CameraActors in your UE project
Cluster Pool

The **Cluster Pool** will automatically detect all available machines on the same network. Once detected, you can assign as many machines to the **Cluster Pool** as needed and inspect each machine in the pool to see network status, IP address, current Streams and available Assets.

Cluster Pool Properties include:

- **Render Machines:** The list of render nodes that the content will be distributed across
- **RenderStream Understudies:** backup nodes in case of failover

**Please note:** Pools should be named as to purpose or the node group assigned. For example, "Cluster" is not as descriptive to its purpose as "Main_Wall_Nodes"

Cluster Pool Machine Properties

Right click on the name of a machine in the **Cluster Pool** to edit additional properties related to that machine:
**Machine** properties include:

- **Load Factor**: Relative load weight this machine can handle. Weight is relative to other nodes in pool. The workload can be spread evenly across the cluster by the user using a load factor.
This allows you to take into account each machine’s spec and assign loads accordingly.

---

**Online status**

---

**Preferred Synchronisation Adapter**: Choose the network you want sync to occur on.

Only used to control cluster communication and content syncing; it has no impact on how the streams are communicated to disguise.

---

**Network Adapters and settings**: Non-editable list of network adaptors available on the node.

To configure the system for Cluster Rendering, all machines within a Cluster Pool must be on the same subnet;

If a machine is reporting an unexpected IP address (e.g. loopback address), make sure to select the specific network adapter within the ‘Network’ tab of d3manager.

---

**Streams**: non-editable list of RS output from the node.

---

**Assets**: non-editable list of assets resident on the node.

---

**Cluster Assigner**

The **Cluster Assigner** is where we control the settings for the distribution of the content across all the machines in the **Cluster Pool**.

Cluster Assigners are used to create definitions as to how channels are delivered from the nodes. You can create as many Cluster Assigners as needed.
**Cluster Assigner** Properties include:

### Splitting Strategies

These are used to generate the sub-regions needed for distributing content across a set of render nodes.

There are 3 types of splitting strategies:

- **Tiles** - lets the disguise software split up the content as it sees fit to the mapped channels. Note: this splitting strategy DOES NOT work with Mesh Policy.

- **Full frame** - Sends the entire stream to the mapped channel.

- **Manual** - splits each Viewport according to user specified instructions. When using the ‘Manual’ splitting strategy, the options for specifying how to split frames can be found within the Advanced workload settings.
- **Splitting Overlap**: The Splitting Overlap option can be used to define a blend region at each split that hides imperfections at the seams.

- **Padding (Pixels)**: The Padding option enlarges the rendered area at each split, but does not stream the padded area, cutting off edge artifacts.

- **Load Weight**: 'Load Weight' allows you to set a priority for the Cluster Assigner. The higher the weight of the assigner, the more machines will be assigned to render the content and thus more splits will be made.


- **Alpha**: The option to enable or disable the Alpha channel on the content is available.

- **Scale factor**

- **Preferred network**

Once the cluster has been configured, the stream's playback is managed via the Cluster Workload controls.
The options for managing the workload are:

- **Instances**: there can be only one workload instance running at a time in the same cluster pool. When started, the Instances table will be populated with the machines that have been sent the start signal.

- **Start**: sends signal to all machines in the pool to launch the Asset according to config settings.

- **Stop**: Sends signal to cease outputting a stream and quit the process sending it. The workload log can be opened by left clicking on the status within the Instances table.

- **Sync**: begins the process of content sync. Content is synced from the source machine to all other machines available within the cluster pool; copies only the necessary directories and files needed.
Advanced: right click to open the workload's advanced options:

Please note: Any changes to the Assigner require a restart of the workload

Receive Health

Status of entire cluster receiving streams, dynamically reported while stream is running.

Right-click on Receive health to open machine specific window:
Default

In the Default section of the RenderStream Layer, there are four common layer properties:

- **Blend Mode**: Premult-Alpha is needed for content that contains alpha data
- **Brightness**: Layer intensity
- **Mode**: This is used to determine the timeline's playback behavior as the layer is playing

Properties Section

The last section of the RenderStream layer is asset specific and will display all exposed parameters from the asset as well as their keyframe editors
Workflow

- Set up your RenderStream environment including installation of the Render Engine and required plugin.

- Use the RenderStream Layer to control the third party render engine.

  For Notch, visit this link
  
  For Unity, visit this link
  
  For Unreal Engine, visit this link

**Please note:** The preferred sync adapter should not be left set to 'any', it should be set to the network of the highest bandwidth that is available to all machines in the workload, plus the disguise machines. This network will then be the one that is used to do the project sync and for all workload comms.
RenderStream and Unreal Engine

RenderStream is the proprietary disguise protocol for controlling third party render engines from The disguise software. This set of topics covers the steps needed for configuring Unreal Engine for use with RenderStream.

This list includes the essential set of topics to help guide you through the RenderStream-UE workflow.

- UE Project Setup
- Scene Optimization
- Scene Levels
- Exposed Parameters
- Remote Texture Sharing
- 3D Object Transforms
- Remote Text Parameters
- Level Sequencing
- Multi-user Editing
RenderStream and Unity

RenderStream is the proprietary disguise protocol for controlling third party render engines from The disguise software. This topic covers the steps to configure Unity with RenderStream.

Warning: For cluster rendering it’s recommended to use render nodes from the same disguise product range, e.g. all rx series machines. Mixing of machines from different product ranges is untested and unsupported.

Unity is a third-party game engine that can be used to create 3D scenes. Unity projects can be adapted such that they create RenderStream output streams using disguise’s Unity plugin. This plugin supports both camera and non-camera based mappings.

A professional license is required when working in Unity. Please visit the Unity website for more information on purchasing licenses and training on the Unity software.

Plugins

- In order to communicate with the disguise software, Unity requires the installation of a plugin on the render node.

- The RenderStream pre-packaged plugins for Unity are available in the disguise github.

- For the most up to date Unity plugin, you can compile the plugin from the source code here

- Place the plugin into this folder : Unity: PROJECT_ROOT/Assets/DisguiseUnityRenderStream.
When adding a plugin to a Unity project, it is important that it is placed in the correct location and that the folder containing the plugin files is named correctly otherwise unexpected errors may occur.

Unity project setup

1. Launch Unity, navigate to Projects and select **New**.

2. Select a project template (e.g. 3D)

3. Name the project, set the location to: “C:\Users\USERNAME\Documents\Renderstream Projects” or “D:\Renderstream Projects” if using a system with a media drive (e.g. RX) and select ‘Create’.

4. Open the project folder and place the plugin inside the ‘Assets’ folder.

5. Select **File** followed by **Build Settings**:
   a. Set **Architecture** to **x86_64**.

6. Navigate to **Player Settings...** and **Other Settings**:
   a. Set **Api Compatibility Level** to **.NET 4.x**.

7. Optionally, set game object Channel visibility:

8. a. **Edit** then navigate to **Project Settings...**
   b. **Tags and Layers** and **Layers**.
   c. Give any empty **User Layer** a name.
   d. Select any object from the scene
   e. Set **Layer** to your newly defined Layer from the Inspector panel.
f. Select your Camera(s)

g. Select whether or not you want the Camera(s) to see the objects in your newly defined Layer by opening the **Culling Mask** dropdown from the Inspector panel.

9. Build the Unity project.

10. Save and close Unity.

Please note: Ensure the correct version of the disguise software is installed on all machines in network, including render nodes.

**RenderStream Layer configuration**

1. Create a new RenderStream Layer.

2. Select **Asset** and choose your Unity executable.

3. Right-click the Asset to open its editor:

   a. Set **Source Machine** to the machine with the asset (used for syncing content).

4. Select **Cluster pool** and create a new cluster.

5. Within the Cluster Pool, add the desired machines.

6. Select **Cluster assigner** and create a new **MultiChannelClusterAssigner**.

7. Within the **MultiChannelClusterAssigner**:

8. a. Select the Asset.

    b. Select **Create Channels**.
c. Within each Channel’s Cluster Assigner and/or the default Cluster Assigner:

   i. Select the distribution strategy and video transport options (changes to transport settings require the workload to be restarted).

9. Expand the Default separator and right-click MultiChannelMap to open it’s editor:

   a. Select Create Channels.

   b. Assign one or more Mapping (regular disguise mapping) to each Channel (e.g. frontplate). Each unique Channel-Mapping combination creates a new workload.

10. Right-click Workload to open it’s editor:

    a. If the machines in the cluster pool do not have the content or the project has changed, press Sync.

    b. Ensure all Sync Tasks are marked completed.

    c. Press Start.

    d. Wait for Workload status to switch to Running.

**Warning:** When launching a Unity executable for the first time, a Windows Firewall popup will appear. If the executable is not allowed through the firewall, the disguise software will not be able to receive the RemoteStream.

**Warning:** Unity assets can only be split across multiple nodes when using 3D Mappings (i.e. Camera Plate or Spatial). Attempting to split using a 2D Mapping will not work; all nodes will render the entire frame.
Exposed parameters

The Unity plugin allows you to expose certain options for each component of an object available within the scene. These options will be presented as parameters within the RenderStream layer in the disguise software. Modifying the value of a parameter in turn changes the value of the corresponding option in Unity thus altering the options of the selected object within the scene.

1. Exposing a parameter in Unity:
   a. Select an object from within the scene (e.g. any light source).
   b. Select ‘Add Component’ at the bottom of the Inspector panel.
   c. Add the ‘Remote Parameters’ component.
   d. Drag and drop the component you wish to expose (e.g. Light) into the ‘Exposed Object’ field.
   e. Expand the ‘Fields’ separator.
   f. Select all options you wish to expose (e.g. Colour and Intensity).

2. Build the Unity project.

3. Save and close Unity.
4. Open the RenderStream layer in the disguise software and start the workload.

5. Modify parameter value(s).

Remote Texture Parameters

The Unity plugins offers support for sharing textures remotely through the use of exposed parameters. This allows a two-way flow of video content between the disguise software and the Unity engine.
1. [Optional] Add a Plane (or any other 3D game object) to the scene.

2. Create a new Render Texture: Select 'Assets' in the Project panel. Right-click inside and select Create → Render Texture.

3. Drag and drop the new Render Texture onto your Plane (or 3D game object of your choice) in the scene. Confirm that the new Render Texture has been added as a 'Material' component to your game object. Confirm that the new Render Texture has been set as a material element in the 'Mesh Renderer' component of your game object.

4. Expose the Render Texture as a remote parameter: Add a 'Remote Parameters' component to the Plane (or 3D game object of your choice). Drag and drop the new Render Texture (Material) component into 'Exposed Object'. Open the 'Fields' separator and ensure that "Main Texture" is enabled (no need to enable any other fields).

5. Build the Unity project.

6. Save and close Unity.

7. Open the RenderStream Layer in the disguise software and start the workload.

8. Create a new layer (e.g. Video) and assign media (e.g. Ada).

9. Move the new layer underneath the RenderStream Layer with Ctrl+Alt+down arrow.

10. Alt+drag to arrow the newly created layer into the RenderStream Layer.

11. Confirm arrowed input appears in the RenderStream content.
Screenshot showing how to create a Render Texture and expose it on an object.

Remote Parameters-3D Object Transforms

You can also expose a Unity GameObject's transform parameters; this will allow you to control the object's movements in two ways which is defined by the field you expose. When exposing a GameObject's Transform, you will have the options to expose the following fields:

**Transform:** This allows you to control the full 3D transform (translation, rotation, scale) of the GameObject using a null object in the disguise software. This workflow is known as **3D Object Transform.** The null object acts as a 'proxy', allowing you to move objects in 3D via the on-screen transform handles, or by linking it to a dynamic transform data source, e.g. a tracking device.

**Local Rotation, Local Position, Local Scale:** If you expose the local options these will give you the ability to keyframe the rotation, positions and scale in the disguise software allow for quick and easy manipulation of a Unity object on the disguise timeline.

Note: if you expose all of the fields, the Local Rotation, Local Position and Local Scale will override the Transform in the disguise software. This will result in the null object not controlling the Unity GameObject.
Remote Parameters-Text Parameters

The Exposed Parameter workflow can also be used to expose live text parameters using a “3D text” actor in the Unity engine. You can use the Remote Parameters workflows to expose the text input field in the disguise software allowing you to edit the text in real time.

Scenes

Scenes in Unity can be composed of any number of game objects which are unique to that scene (by default). The Unity plugin offers two forms of multi-scene support:

**Manual** - this option restricts the disguise software’s control of scenes and instead merges all Channels and remote parameters into a single scene.

**Selection** - this option allows scenes to be controlled from inside the disguise software; Channels are merged into a single list (duplicates removed) and remote parameters are per-scene.

1. Create a new scene:
   a. Open the Kebab menu for your current scene in the Hierarchy panel.
   b. Select *Add New Scene*.
   c. Optionally, populate scene with your desired game objects and exposed parameters.

2. Select **Resources** then **DisguiseRenderStreamSettings** from the Project panel.

3. Set **Scene Control** option accordingly.

4. Build the Unity project.

5. Save and close Unity.
6. Open the RenderStream Layer in the disguise software and Start the workload.

7. Modify the **Scene** parameter as part of your normal sequencing.

---

**Time control**

The Unity plugin offers Timecode support. This means that if any game object has a ‘Playable Director’ component and is animated using timeline functionality, it will be reflected in the disguise software timeline.

1. Adding time control to an object in Unity:

   a. Select **Window** followed by **Sequencing** followed by **Timeline**.

   b. Optionally, move the Timeline Window from the main panel to the bottom panel (i.e. drag and drop Timeline Window next to Console Window).

   c. Select an object from within the scene (e.g. any user placed prop).

   d. Click the **Create** button within the Timeline Window.

   e. Save the new Timeline.

   f. Hit the Record button on the newly created Animator for the chosen object within the Timeline.

   g. Add an initial keyframe by right-clicking on any animatable property of the object (e.g. Position) and select **Add Key**.
h. Move the Playhead along the Timeline.

i. Modify your chosen property either by using the 3D controls within the Scene or by updating the value directly from within the Inspector panel (a keyframe will be added automatically when a value is changed).

j. Hit the Record button again to stop recording.

k. Return the Playhead to the beginning of the Timeline and play the sequence to confirm your animation is correct.

l. With the object still selected, select Add Component at the bottom of the Inspector panel.

m. Add the Time Control component.

2. Build the Unity project.

3. Save and close Unity.

4. Open the RenderStream Layer in the disguise software and Start the workload.

5. Play your timeline within the disguise software.

Useful Unity information

There is no need to attach the disguise RenderStream script to the cameras in Unity. Cameras will auto-configure on asset launch.
When using the ‘Manual’ scene selection option in Unity, game objects from all built scenes will not appear in the “Default” scene. In order to merge scenes and/or dynamically load/unload them, a custom script must be used.

The “Default” scene will be the first indexed scene from within the ‘Scenes In Build’ table in Unity’s build options.

The exposed parameters from all scenes will still show within the disguise software, even if game objects from all built scenes are not merged into the “Default” scene.

When using the ‘Selection’ option in Unity, game objects and exposed parameters are unique to each scene. There is no shared object scene similar to the “Persistent” level in Unreal Engine.

When launching a Unity executable for the first time, a Windows Firewall popup will appear. If the executable is not allowed through the firewall, the disguise software will not be able to receive the RemoteStream.

Game objects included as part of a Unity template may not be able to be controlled via Timecode. This is a Unity issue rather than one with the disguise script.

If attempting to use the Unity High Definition Render Pipeline (HDRP):

Both the ‘Windows 10 SDK’ and ‘Game Development with C++’ modules must be installed as part of your Visual Studio installation.

The ‘Scripting Backend’ must be set to “Mono” when building the executable.

Since Unity Assets are built executables, disguise will not recognise the ‘Engine’ of them; they will simply be reported as “Custom”. disguise will not be able to report the Unity plugin version used within built executables. If an incompatible disguise-Unity plugin combination is used, no explicit notification will be shown.
RenderStream and Notch

RenderStream is the proprietary disguise protocol for controlling third party render engines from The disguise software. This topic covers the steps specific to configuring the RenderStream workflow with Notch.

⚠️ Warning: For cluster rendering it’s recommended to use render nodes from the same disguise product range, e.g. all rx series machines. Mixing of machines from different product ranges is untested and unsupported.

Notch is the third party render engine available from Notch.one. Notch can be configured to run either internally via a Notch layer or externally via RenderStream. This topic will cover the RenderStream-Notch workflow. For more information on using a Notch layer, visit this link.

Notch Host
Notch_Host is a secondary application that is included as part of the main disguise installation process. The purpose of this application is to load Notch blocks externally the disguise software and create RenderStream output streams which can be accessed by all machines across the same network.

Notch_Host testing can be conducted locally as data is delivered through NDI and therefore no special network hardware is required.

Please note: Ensure the correct version of the disguise software is installed on all machines in network, including render nodes.

Notch Assets

Place Notch blocks in: C:\Users\USERNAME\Documents\Renderstream Projects or D:\Renderstream Projects if using an rx hardware system.

The Asset discovery system searches all sub-directories so you can place all Notch blocks in an aptly named "Notch" folder.

Setup RenderStream Layer

1. Launch the disguise software and create a new RenderStream layer.
2. Select Asset and choose your Notch block.
3. Right-click the Asset to open its editor.
4. Set ‘Source Machine’ to the machine with the asset (used for syncing content).
5. Select Cluster pool and create a new MultiChannelClusterAssigner.

Within the Cluster Pool:

1. Add the desired machines.
2. Select Cluster Assigner and create a new MultiChannelClusterAssigner.
Within the MultiChannelClusterAssigner:

1. Select the Asset.

2. Select Create Channels (if required).

Within each Channel's Cluster Assigner and/or the default Cluster Assigner:

1. Select the distribution strategy and video transport options (changes to transport settings require the workload to be restarted).

2. Expand the Default separator and right-click MultiChannelMap to open its editor:

3. Select Create Channels.

4. Assign one or more Mapping (regular disguise mapping) to each Channel (e.g. frontplate). Each unique Channel-Mapping combination creates a new workload.

5. Right-click Workload to open its editor.

6. If the machines in the cluster pool do not have the content or the project has changed, press Sync.

7. Ensure all Sync Tasks are marked completed.

8. Left click Start.

9. Wait for Workload status to switch to Running.

Please note: Notch blocks can only be split if they contain an exposed camera. If you attempt to split a block that does not have an exposed camera across multiple render nodes, the content will simply be duplicated.
RenderStream Failover

Active backups are crucial for live events. With the RenderStream Failover functionality, you can set up rules and switch inputs to allow your servers to switch to an Understudy if a stream isn’t working.

Workflow

Follow these steps to configure RenderStream for failover:

1. Create a RenderStream Layer
2. Create a Cluster Pool
3. Assign the RenderStream Understudy machine within the Cluster Pool
4. Create a Cluster Assigner
5. Create a new Channel mapping
6. Start the workload; all render machines and Understudy will launch their instances of the project
7. Use the d3net Network Session widget to monitor and control the render machines and Understudy; switch to the Understudy within the Network Session widget if needed using 'Mark as Failed' on the stream that has failed and disguise will automatically switch to the Understudy.