Turning Real World Objects into Photorealistic One Poly Models

Darren Flowers-Finley
Master of Technical Art
University of Utah,
Senior Tech Engineer, Collins Aerospace
Salt Lake City, Utah
Darren.Flowers-Finley@rockwellcollins.com

Jonathan Bishop
Master of Technical Art
University of Utah,
Synthetic Environment Artist, Collins Aerospace
Salt Lake City, Utah
jonathan.bishop@rockwellcollins.com

ABSTRACT

The simulation industry currently needs 3D models for visualization. Photogrammetry and impostors can improve the visual standard of those models while reducing polygon count and texture space requirements. These techniques produce models that display real world accuracy. Current methods for building models take a lot of time and resources. Additionally, displaying the highest fidelity version of each model can be very intensive for the computer. Level of Detail (LOD), which creates multiple versions of the model for different distances from the eyepoint can still burden the machine's performance. If the simulation industry achieved the same or better visual fidelity as other industries and maintained current performance, while reducing development costs and time, a new era of simulation would emerge. Our method accomplishes this.

The simulation industry today creates, models and their LODs with aging methods, which are more time consuming and visually less appealing compared to similar industries. Armed with modern software (Agisoft, Meshroom, Houdini, etc.), and a clever use of physical and virtual cameras, we have created photorealistic models and the LODs to complement them. This method of creating photorealistic models is called photogrammetry. The newer LOD method, being adopted by game development companies, uses a technique called impostors. Photogrammetry creates a hyper realistic model when the eyepoint is close, impostors maintain that visual standard when the eyepoint is far away from the object for a fraction of the performance cost allowing more objects to be drawn at any given time. We have created and tested photorealistic models from scratch in a matter of hours versus weeks. Additionally, this allows for quicker iterations on newer projects and more efficient ways of altering models down the road.

ABOUT THE AUTHORS

Jonathan Bishop holds a Master in Technical Art and earned his bachelor's in game art from the University of Utah. Mr. Bishop served 10 years as a Joint Terminal Attack Controller and held the position of both an Evaluator and a Squadron Chief Tactical Instructor. Mr. Bishop conducted numerous combat missions in support of Operation Iraqi Freedom where he earned a Bronze Star for combat operations during the Basra crisis alongside British Armed Forces.

Darren Flowers-Finley holds a Master in Technical Art and a bachelor's Under Study in Environment Art Production from the University of Utah. Mr. Flowers-Finley has earned multiple scholastic awards for his performance in school such as the National Society of Leadership and Success. These awards were a product of his upbringing as a military brat traveling across the world meeting new people and learning about different lives and cultures. Those experiences shaped him into the man he is today: a hard worker, a committed person and someone you can trust.

Turning Real World Objects into Photorealistic One Poly Models

Darren Flowers-Finley
Master of Technical Art
University of Utah,
Senior Tech Engineer, Collins Aerospace
Salt Lake City, Utah
Darren.Flowers-Finley@rockwellcollins.com

Jonathan Bishop
Master of Technical Art
University of Utah,
Synthetic Environment Artist, Collins Aerospace
Salt Lake City, Utah
jonathan.bishop@rockwellcollins.com

INTRODUCTION

In today's age we have seen a great change in the way we create, develop and maintain 3D models (Digital School, 2020). There has been a steady increase in the realism and quality of 3D models in the entertainment industries. The simulation and training industry has made great strides in developing mechanics and real-world comparability. This is because simulation and training customers care more about whether their simulated plane, helicopter, tank, train or car reacts and behaves like its real-world counterpart rather than how realistic it looks. The entertainment (gaming, multimedia, Hollywood etc.) industries' tactic is to develop cinematic and emotion-driven experiences which may require reflecting reality. Both industries have come up with creative and useful ways of solving similar issues. Methods such as system optimization, modeling techniques, research & development, visual consistency in models and environments have progressed visual fidelity greatly over the last few years. The entertainment industry typically focuses more time on their visual standards instead of real-world applications. They will have techniques or processes that may help the simulation and training industry and vice versa. This paper describes how we implemented innovative technologies from the entertainment industry into simulation and training for improved visual fidelity.

One limitation the simulation industry faces today is the lack of visually realistic models. This is due to the limitations of specifically built hardware that powers and runs simulation software. Realistic models can hinder simulations because of file size, model polygon restriction and texture space. This becomes compounded by the expansiveness of the environments needed for some simulation companies compared to entertainment industries; this issue can create many complex problems such as hard triangle limits. The engines that are used vary company to company and their software needs may be restricted based on the engine's requirements. For example, "Grand Theft Auto (GTA) 5", an open world game created by Rockstar Games, has a map size of 49 square kilometers (Carpenter, 2020). In many AAA open world game environments, there may be up to hundreds of thousands of polygons in a level for it to run without any reduction in the frame rate at its highest setting. For instance, Kratos from, "God of War" was created with over 32K polygons in just his face and 80K polygons for his body (Shuman, 2019). GTA's engine is built to run with these factors in mind on modern PCs and consoles. They have optimized it to fit their visual standard of realism. GTA 5 has multiple visual settings to allow the end user to have the best gaming performance based on their personal computer's specifications. Because of the competitive nature of the entertainment and game industry, there is a need to make ever increasing realistic looking games each year. These game companies have invested in researching and developing tactics to get their game in as many hands as they can, while being as visually superior, fun and entertaining as possible. According to Randy Lee, head business development for Tencent America LLC, *"Content is the most important factor in the U.S., but in China, distribution is king." (Game Marketing Genie, 2018).*

Comparatively, a typical flight simulation must render 323,548km² of land at 60 frames a second and can only have 80-100K polys on a screen at a time to prevent dip in framerate. Simulations must also be able to render different sensors, such as FLIR (Forward Looking Infrared Radar), NIR (Near Infrared), I2 (Intelligent Sensor), CDSA (Color Day side assembler), OTW (Out the Window) and I2 blend. This means running multiple scenes at once, requiring each sensor to have its own files, textures, models and associated hardware parallel and seamlessly to each other; These standards are nonnegotiable. For this training and simulation system example, the company would have to greatly increase the database capabilities, change their software and/or engine preferences or substantially change the hardware used if the result were to look like blockbuster movies or AAA game releases, while maintaining the current capabilities.

This paper describes a method and workflow for creating realistic models quickly with highly optimized LODs that have a high degree of visual fidelity for the flight simulation and training industry. These methods were created using photogrammetry and LOD impostors. This follows system and software standards, while upgrading without investing too heavily in new technology and hardware. This document will explain how to (1) integrate the methods of photogrammetry to create realistic 3D models by taking photos of real-world objects and turning them into models (2) convert those models into a one polygon stamp called an impostor that will be used to represent the realistic models at a distance. LOD impostors maintain the same visual quality at a fraction of the triangle count cost.

WHAT IS PHOTOGRAMMETRY

Photogrammetry is the process of creating high fidelity 3D models from multiple images of an object taken from different angles and distances. These pictures are paired together and aligned based on key points and depth maps identified in the photogrammetry software, an example of which is seen in Figure 1. Key points are gathered from multiple images that share the same location and color info. Depth maps are color coded representations (red meaning close, blue meaning far) of the distance of an object from the camera. Key points are color coded and placed in their corresponding spot based on the depth maps to build an object. This allows software to create a cloud of points in the shape of the object. The point clouds are used to generate a highly dense polygonal mesh which is optimized, to a manageable size for the required system standards. The next step is to project the color from the point cloud onto the new mesh to create a diffuse texture. To maintain the micro detail from the high polygonal mesh we create a normal map through a process called baking. This is the process of taking a high poly mesh (typically millions of polys) and projecting its normal detail on to a texture that can used to simulate high detail onto a lower poly mesh (1-80K depending on system standards). This where the Physically Based Rendering (PBR) texture workflow would be incorporated if required. PBR is the science of rendering materials to look as close to reality as possible (Russell, J, 2018). This is achieved by basing the computer graphic renderer on the principles of physics and utilizing separate texture maps to represent the color, normal, roughness, metalness and ambient occlusion values of an object.

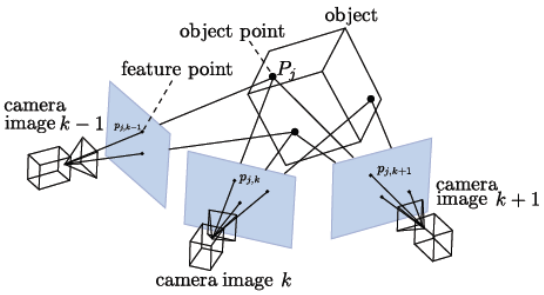


Figure 1. Photogrammetry Method Example Stuart Attenborrow (2018, July)

WHY IS THIS BETTER THAN TRADITIONAL MODELING?

The Entertainment industry is moving toward more realistic models and environments (Unreal Engine, 2020). Photogrammetry is becoming a prominent solution because of its quick process of creating models compared to traditional box modeling. While traditional methodologies required 60-80 hours to create a complex model, our investigations revealed that through photogrammetry, the effort to create the same model can be reduced to 16-40 hours, a significant savings in both time and cost. The cost savings can be as much as 50 percent in some cases.

Figure 2. Photo Data Gather Example. Nick Lievendag (2018, June)

Using photogrammetry to create models is much faster than creating it by hand. Dice, the company behind Star Wars Battlefront and pioneers of photogrammetry in games, was polled on the speed of traditional modeling versus Photogrammetry. They concluded that photogrammetry was usually two times faster than traditional modeling (Hall, C. L, 2016).

WHAT ARE IMPOSTORS

Impostors are a LOD technique in which you take many rendered images of a textured 3D model, compile them into a texture atlas called a sprite sheet and project those individually rendered images onto a single polygon. Sprite sheets are typically used for animation as the real time engine cycles through each rendered image on the texture sheet to imitate motion. In this case, Impostors use sprite sheets to cycle through the images of the selected object based on the eyepoint angle to the center of the object to give the illusion of a 3D shape. For this technique to work in a real time engine, import the sprite sheet and have a single stamp that is always facing the camera. Each camera angle is tied to an image on the sprite sheet and will show that dedicated image on the polygon when the camera angle changes. The polygon will blend between each corresponding image to give the illusion that the poly is 3D as the eyepoint moves. The more camera angles that are setup, the smoother the impostor looks when transitioning between images on the sprite sheet. Impostors are an effective option for optimizing the LOD while maintaining the same visual fidelity as photogrammetry and freeing up the number of polys used.

METHOD & PROCESS

Why are photogrammetry and impostors becoming necessary in today's simulation industry? To start with, using these technologies provide a larger return on investment. When companies spend money building models, the time spent should be efficient and produce a good product. When creating impostors, once the photo data is collected for a photogrammetry project, the process is quick to create a usable and realistic 3D model that is both believable and efficient to use. Secondly, by utilizing newer photogrammetry and impostor technologies in their content development processes, companies can gain an edge over competitors.

Photogrammetry and impostors are quickly becoming the standard in other industries. Games such as Star Wars Battlefront I and II utilize photogrammetry heavily and almost exclusively for their model database (Hall, C. L, 2016). Companies like Quixel Megascans have dedicated themselves to building the biggest photogrammetry library in the world. Quixel Megascans have also partnered with companies like Epic Games to make their photogrammetry library more accessible to gaming companies. Games like Fortnite, one of the most optimized multiplayer games on the market (capable of playing their game on PC, mobile and console and allowing players to play with each other across multiple platforms) use impostors and have pioneered some of the most popular and recent techniques in optimization in the gaming industry.

Photogrammetry and impostor LOD technologies are a great way to make high fidelity and optimized 3D models when incorporated into your workflow correctly. In order to understand what it takes to incorporate these technologies into your workflow, this document will lay out the basic information and how it affects overall performance with some examples. There are both negatives and positives to using photogrammetry and impostor LODs and it is not meant for every project or product. Instances where this would not be recommended would be when realism is not required or when viewing objects always from the same eyepoint angle. It is important to keep in mind as newer software packages and better photo data capture techniques become available, the process becomes smoother and even better results may be acquired.

Selection of Photogrammetry Subject

Just about any object can be photo-captured and turned into a 3D model. The object selected will determine what photo-taking process will be needed to capture it. For this document, the object that will be used is a L7A2 general purpose machinegun trainer. The process used to capture this gun was a turntable set up in a green room with controlled lighting. Using a turntable with controlled lighting helps to avoid many of the pitfalls in the photo capturing process compared to capturing data outside, such as bad lighting and complex or busy backgrounds. Bad lighting is described as lighting that shows shadows on the object that delineates what direction light is coming from. Avoiding this is important because hard shadows on an asset will conflict with the simulated shadows in the real time engine and will add more work in post-production. Additionally, avoiding specular highlights and overly bright areas on the object is ideal. These brighter areas can produce a lack of key point data or no data at all for those areas causing the program to not build a successful model. Regarding busy backgrounds, heavy clutter such as having buildings or big bright objects that stand out in the background can confuse the photogrammetry software when creating key points for the main object. Photogrammetry programs only have a set amount of key points per image, and you want 90 to 100 percent of your key points to be your main object. If there are too many things in the background the program will

waste key points on them. This is typically caused by not having your object take up at least 70 percent of the image space resulting in the system wasting key points on background objects because it does not know what to focus on. Objects in the background that move or change position from photo to photo may also confuse the software resulting in mis-aligned photos, which in turn generates a point cloud that does not match the volume of the actual object. It is important that your object of choice remains one hundred percent still until the completion of the photo-capturing process.

Photo Data Collection of Subject

Knowing how to take pictures is one of the most important parts of the photogrammetry process. The object must be well lit without any hard shadows and each photo should have the subject take up a minimum of 70 percent of the photo space. Each photo should have at least 60 percent overlap of the details from the prior picture in an image sequence, (Reference figures 3 and 4). These photos should have a consistent white balance, so the images do not have slight hue changes.

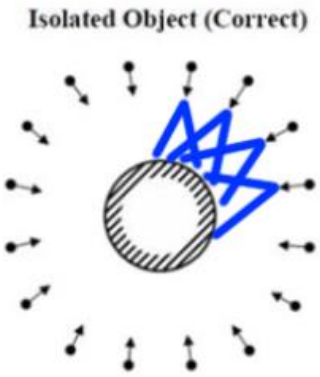


Figure 3. Photo Overlap Example Agisoft Manual (2020, p.7)

Figure 4. Photo Data Overlap Example

Turning Photo Data into a 3D Model

After photo data has been gathered and compiled, the data is fed into software that can digest and create a point cloud. For this example, Agisoft was used, though there are many software packages to choose from. First, add photos into the software and align them. This can be done procedurally or by hand if you want more control on how the software interprets photos. The software will build depth maps and key points for each picture. Sparse clouds will then be created using the depth maps and key points, (see Figure 5). Sparse clouds are a loose collection of points that are generated using key points and depth maps from the images. This helps the software determine the basic shape of the object when aligning the photos. It is also used for editing and correcting issues in your photo alignments and key point allocation quickly. Think of it as a quick preview for the shape of the object. If the sparse cloud looks bad, do not waste anytime going through the full process until it looks correct. These pictures show the depth map (see Figure 5) and key points used to build a sparse cloud (see Figure 6).

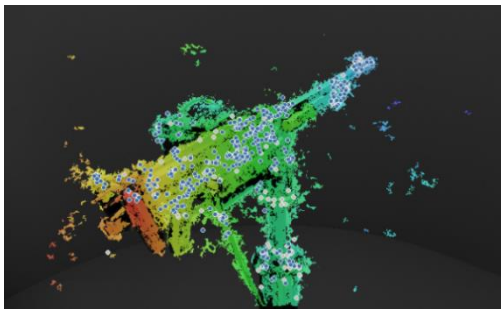


Figure 5. Depth Map of Gun

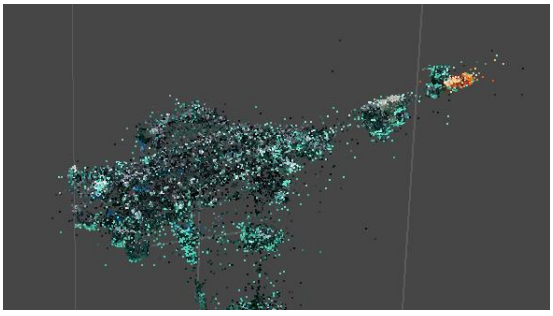


Figure 6. Sparse Cloud of Gun

Once the sparse cloud (see Figure 6) is generated a dense cloud (see Figure 7) needs to be built to generate a mesh. Dense clouds are exactly what they sound like, a denser version of the sparse cloud. After the dense cloud is created, the high poly mesh (see Figure 8) can be established and the texture built from the dense cloud in conjunction with the photo data.

Figure 7. Dense Cloud of Gun

Figure 8. High Poly Mesh of Gun

In the final stages of the mesh-building process there will be problem areas, (circled in red in figure 9). These problem areas are identified by the user during the build process after the high poly dense mesh is made. These areas can be cleaned up in any 3D modeling program. The gun then needs to be decimated and have any color correction issues fixed due to lighting anomalies or user error. Decimating a mesh is the process of reducing the polygon count of the model. Decimating can be done by hand in any 3D modeling software or through procedural means via software like Houdini. The purpose of decimating the mesh is to provide an optimized model, regarding UV and polygonal structure for general use in the real time engine. Usually, this version of the object would be represented as LOD 0, or the LOD which is always shown when closest to the eyepoint versus impostors which would be utilized as LODs that are farther away. The mesh above, (see Figure 8), is made of four million polys. This mesh will be decimated to about four thousand polygons (see Figure 10).

Figure 9. Identified Problem Areas

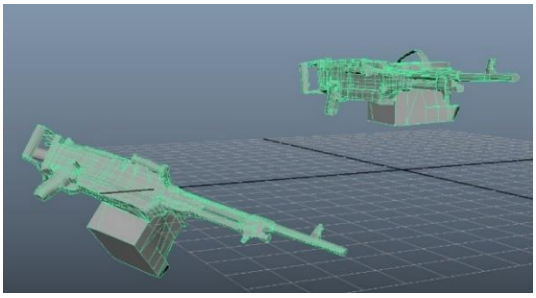


Figure 10. Lower Poly Mesh of Gun

When the mesh is finished being decimated, it is ready to be brought back into Agisoft and have the diffuse texture and the normal texture projected from the high-fidelity model to the decimated one. After the textures have been baked, the green hue needs to be taken out of the mesh texture. This can be done by editing the texture in any photo

editing software like Photoshop. Once the texture is fixed, create any additional texture maps that need to be created based on the asset requirements (normal, bump, roughness, metallic, ambient occlusion maps etc.). Package all the new textures and mesh together (see Figure 11) then begin the impostor process.

Figure 11. Finished Model of Gun

Prep Photogrammetry Model for Impostor Process

Import the photogrammetry model and associated textures into the software of choice for the impostor conversion. Some common software packages used are Maya, 3ds Max, Blender and Houdini. Once you import the required files, position your model in the scene, create a material and set up desired lighting. Creating/choosing your camera setup should be determined by the type of impostor you want. Below are impostor camera setups to generate a sprite sheet with the intended viewing angles of the object.

The hemi-octahedron (see Figure 12) camera render setup is designed to maximize blending between images on the impostor. This setup captures renders of the object and then in turn uses them create a sprite sheet. This setup is best suited when viewing the impostor from a downward angle. This is the standard for static objects that are only seen from the top of the object down to the base.

The full-octahedron (see Figure 13) camera render setup is designed maximize blending between images on the impostor with a view from any angle. This setup captures renders of the object and then in turn uses them create a sprite sheet. This is the standard for static objects that require the impostor to be seen from any angle.

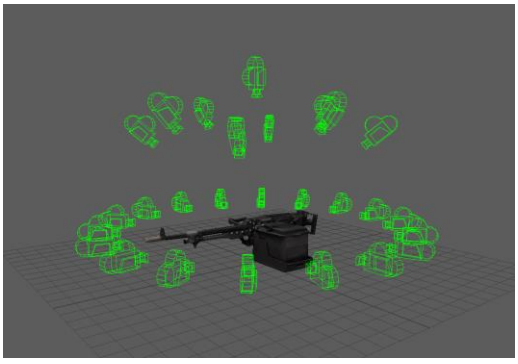


Figure 12. Hemi-Octahedron Camera Example

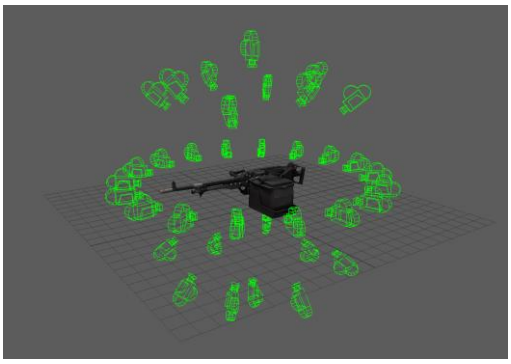


Figure 13. Full-Octahedron Camera Example

The hemi-octahedron and full-octahedron impostor are both designed to easily convert 3D objects into a 2D space. The way the camera is setup allows for maximum use of the texture sheet, in conjunction with the UV space, set up on the impostor polygon.

The full 360 (see Figure 14) camera render setup takes renders of the object at set intervals around an object with a 360° view. This has limited uses since it is not as optimized as full octahedron. This method is still useful if your engine does not support octahedron impostors or if you are not able to generate octahedron rendered sprite sheets.

The single rotation axis (see Figure 15) camera render setup that takes renders of the object on one axis all the way

around the object. This method is best for an impostor that is locked to a single axis.

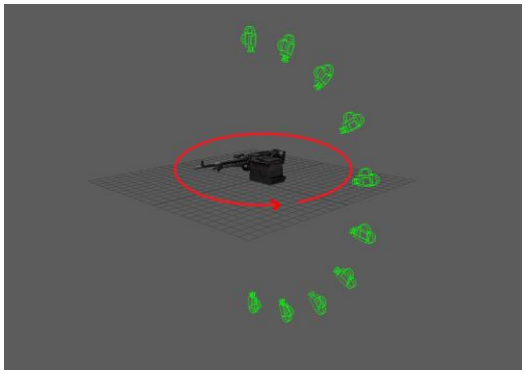


Figure 14. Full 360 Camera Example

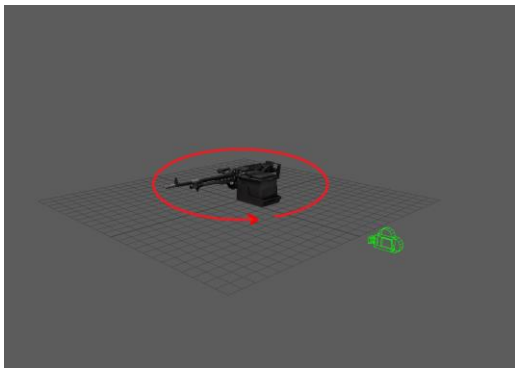
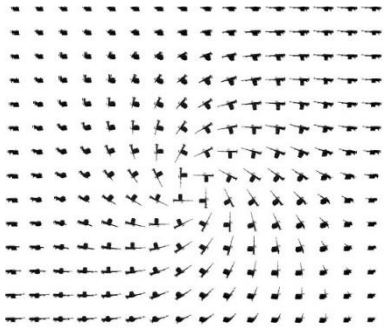


Figure 15. Single Rotation Axis Camera Example

Render & Results

Rendering out a texture sheet should look something like Figure 16. Additionally, some 3D modeling software have automated impostor sprite sheet and polygon creation capabilities. Software like Houdini, which is a procedural visual effects software package that has many functions including a renderer, auto generates the associated impostor polygonal Filmbox (FBX) file at the same time as the texture. FBX files are great for compiling model data into one file that can be imported into a real time engine with all the setting, animation and texture info from the modeling software. This saves time and allows for quicker iteration if needed. When having to create your own polygon, create it in your 3D geometry editor of choice such as Maya, 3ds Max or Blender.



16. Impostor Texture Sheet.

Real Time Engine Setup

Import the sprite sheet and the polygon impostor geometry into the real time engine. The setup for the scene is driven by the method of impostor chosen in the beginning stages. Create an object in the real time engine using the polygon impostor (see Figure 17). For impostors that require viewing the object from any angle, have the polygon always facing the camera. When having issues with the polygon flipping or rotating when viewed from the top, the solution is dependent on which real time engine you are using. Locking one of the axes of rotation or detailing a specific direction for the impostor when viewing from the top may be needed.

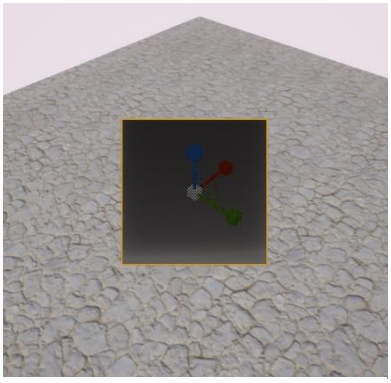


Figure 17. Impostor Polygon.

For the impostors that only require a view angle for one rotation axis, such as from the side, be sure to lock down the other two axes of rotation. These methods depend on requirements from the project director and should be tailored to individual objects. For instance, if flying over vehicles objects never requires a side or bottom view, use a hemi-octahedron impostor. If viewing the same object from the side or below is required, a full-octahedron or full 360 impostor might work best.

Once the polygon is moving and facing the camera (also known as billboarding), tie the respective images on the sprite sheet to a specific camera angle. Depending on the amount of camera renders that were processed, there may be many. The intended outcome is to have a specific image from the sprite sheet be viewed on the polygon at specified camera angles in the real time engine from the eyepoint to the center of the impostor. After all the images are tied to specific viewing angles, the images should transition smoothly from one to the next as the eyepoint viewing angle changes (see Figure 18). Without a smooth transition, the images will pop and seem out of place. It is important to note that the more camera angle images that are setup the smoother the transition of the images will be.

ENHANCING THE LEVEL OF DETAIL END RESULT

The created impostor should be integrated into a real time asset's LOD process. Whether it be a road sign, vehicle or human character, the actual model should be set up as LOD 0 and your impostor should be LOD 1. If more LODs of the model are required, always ensure that the impostor is the highest LOD. In most cases, one or two model LODs with one impostor LOD will do the trick. This saves the system from drawing too many polys on the screen and allows for more objects to be drawn at any given time. Depending on what engine you are using, you may or may not have shadows for your impostors. Note that the shadow uses the silhouette of the polygon and not the fake 3D object being represented.

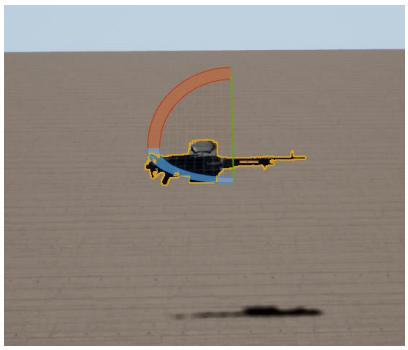


Figure 18. Impostor with Texture.

Results Breakdown for Test Object: Gun

To test the results, we used the 3D photogrammetry model of the gun and five different texture size impostors (see Figure 19). We used 8x8 sprite sheets. This means eight images of specific camera angles across and eight images of specific camera angles down as the texture. These were tested on a Razer Blade laptop with 16 GB of RAM, NVIDIA GeForce GTX 1060 and an Intel i7 in the Unreal Engine 4.22. Below is the collected data covering how the 3D model and the impostors affect the memory usage (see Figure 20).

Figure 19. Impostors & 3D Model (256, 512, 1k, 2k, 4k, 3D Model).

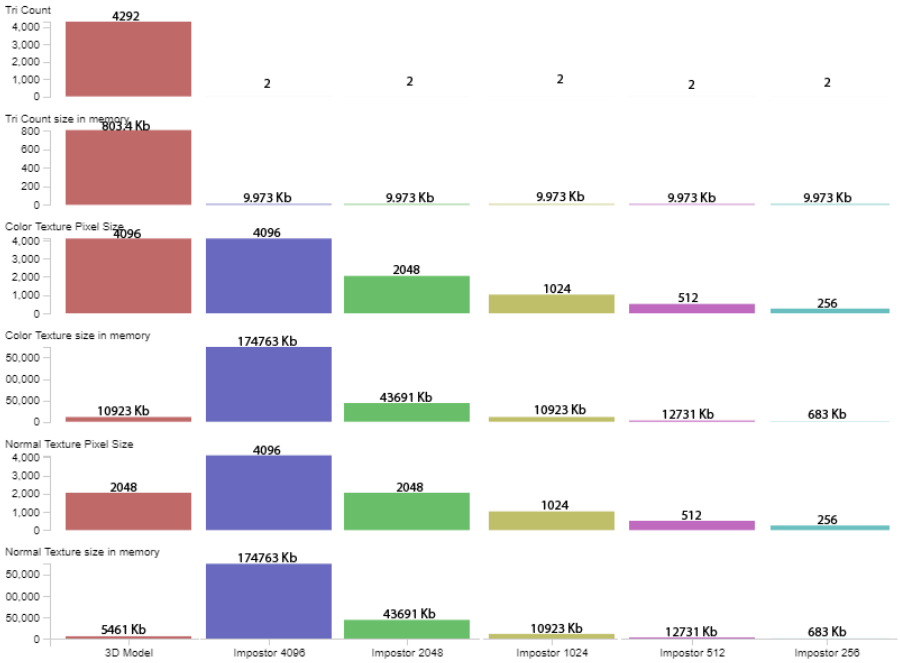


Figure 20. Impostors vs 3D Model Memory Usage Per Model

SUMMARY

Photogrammetry is useful, but there are shortcomings. Gathering enough photo data of the object to generate an accurate model is difficult in some circumstances. No blurry or excessive glare or reflections in the photos is a must. For larger objects, such as trees and buildings, this may involve the use of drones which entails following specific local and federal requirements on top of the skill and equipment. A well-lit object with minimal or soft shadows is optimal for creating usable textures. This will require proper planning for time of year and aligning with workable weather that is conducive to taking usable photo data. There may be additional costs in the form of equipment required to gather some photo data, such as drones, or a trip on site for data collection. Impostors also face shortcomings. They reduce overall poly load on the system but require a substantial increase in texture memory. If the system being used for simulation has a fixed amount of memory and is unable to upgrade that texture memory, impostors may not be the right solution. For scenes with adverse weather or fog, impostor objects will not have visual degradation as it protrudes into the distance like normal 3D objects do. Since the impostor is only one poly and has no physical dimension, it may look out of place and unnatural under those conditions. Creating the impostor textures and the impostor polygon requires a bit of expertise and knowledge of the processes. Even so, there are many resources available on the internet.

Photogrammetry gives the user the flexibility to create high-quality models at a standard that is at the top of many industries. These models are then capable of being modified to fit multiple system standards. The impostor LODs can be iterated upon to meet the ever-evolving standards which vary from system to system. With current and future military simulators, this technology allows for adaptation and flexibility within a model asset library. The ease at which it takes to create these model packages allows for quick pivots when necessary while keeping quality and trainability important. When used under the right conditions, photogrammetry and impostors are powerful technologies that optimize polygon budgets while maintaining model appearance from a distance on most military simulation systems.

ACKNOWLEDGEMENTS

Darren and Jonathan would like to thank the following for their support: Jeanette Ling, Abhishek Verma, Rishabh Kaushik, Larry Seppi, Triston Thorpe, Ankur Rathore, John Haraldstad, Danilo Groppa, Chris Longhurst, Shin Wang, Steve Stapley, Ross Vellinga, Brad Southwick, Suzanne Jones, Lance Moss, Misty Dawson, Eduardo Garcia and Carla Cropper.

Jonathan would like to thank his beautiful wife Rebecca and his father, C.W Bishop-Perera, for their motivation.

Darren would like to thank his parents for showing him how to pursue and follow his dreams.

REFERENCES

Agisoft Metashape User Manual Standard Edition, Version 1.6., from (n.d.).https://www.agisoft.com/pdf/metashape_1_6_en.pdf

Attenborrow, Stuart. "The Pointy End of Camera Matching." The Starry Expanse Project, 9 July 2018, from http://www.starryexpanse.com/2018/04/18/the-pointy-end-of-camera-matching/

Brucks, R. (2018, March 18). Octahedral Impostors. Retrieved May 7, 2020, from https://shaderbits.com/blog/octahedral-impostors/

Carpenter, A. (2020, April 20). How Big is "Grand Theft Auto V" Really? Retrieved May 22, 2020, from https://www.complex.com/pop-culture/2013/09/grand-theft-auto-v-size-scale

Engine, Unreal. "A First Look at Unreal Engine 5." Unreal Engine, 13 May 2020, from www.unrealengine.com/en-US/blog/a-first-look-at-unreal-engine-5

Hall, C. L. (2016, March 17). To Ship Stars Battlefront, Developers had to build their own AT-AT foot. Retrieved May 7, 2020, from https://www.polygon.com/2016/3/17/11251930/star-wars-battlefront-gdc-photogrammetry-at-at-foot

Lievendag, Nick. "The Beginners Guide to 3D Scanning & Photogrammetry on a Budget." 3D Scan Expert, 14 June 2019, from https://3dscanexpert.com/beginners-guide-3d-scanning-photogrammetry/

Lievendag, Nick. "Updated for 2018: RealityCapture Photogrammetry Software Review." 3D Scan Expert, 15 Mar. 2018, from https://3dscanexpert.com/realitycapture-photogrammetry-software-review/

Marques, N. F. (2019, August 15). Tutorial: Lighting in Photogrammetry. Retrieved May 8, 2020, from https://sketchfab.com/blogs/community/lighting-in-photogrammetry/

Ruisheng Wang (2013) 3D building modeling using images and LiDAR: a review, International Journal of Image and Data Fusion, 4:4, 273-292, DOI: 10.1080/19479832.2013.811124

Russell, J. (2018, November 1). Basic Theory of Physically-Based Rendering. Retrieved May 7, 2020, from https://marmoset.co/posts/basic-theory-of-physically-based-rendering/

Vertex Library. (n.d.). 15 Outdoor Photogrammetry Tips. Retrieved May 8, 2020, from https://www.vertexlibrary.com/guide-to-3d-scanning-outdoor-photogrammetry-tips