An Asset Pipeline for Creating Immersive Experiences: From CAD Model to Game Engine

Blake A. Schreurs

ABSTRACT

When an object is designed for manufacture and use, it is vital that various members of the design team, as well as the object's end users, be able to fully comprehend the purpose and value of the designed object. Engineers at the Johns Hopkins University Applied Physics Laboratory (APL) call this "experiencing design intelligence." When designing objects for use by people, following human-centered design practices will help the design team leverage the experience of those people to inform the object being created. Computer-aided design (CAD), in use for decades, has revolutionized the design and manufacturing processes for objects ranging from toys to buildings to spacecraft. CAD tools provide a wealth of information that is essential for many modern operations, and today CAD data can be combined with newer technologies to create immersive experiences that provide even more information and lead to even greater design intelligence for both design teams and end users. This article presents a nominal asset pipeline, including best practices, for taking a CAD model into a game engine to create an immersive experience.

INTRODUCTION

The field of engineering has always been aimed at aiding someone in performing a task through the use of applied scientific principles. Many organizations, including APL, have championed human-centered design as a means to expand their focus beyond the technical challenge of building an object to include gaining a full understanding of how that object will be used within its operational context. Computer-aided design (CAD) has been an indispensable tool of design engineers for decades. Historically, advances in CAD have generally been driven by a desire to increase the utility of a product, reduce the time to deliver the product, or both.

Although they are fantastically powerful, models built using CAD tools exist in a space separated from the physical world, forcing engineers to make the conceptual leap between the digital and physical domains. With the advent of advanced prototyping techniques such as additive manufacturing (3-D printing), design teams have additional tools to help them more quickly iterate through the design process. Today, by taking advantage of augmented and virtual reality (AR and VR, respectively, or XR collectively), engineers can immerse themselves in the digital world to experience a design before it is manufactured in the physical world. Experiencing this design intelligence enables design teams to more quickly and

effectively communicate impactful information, such as complex manufacturing instructions, and to fully consider the object's end user during the design phase, before an object is manufactured and it is too late or too costly to make modifications. Table 1 presents a brief comparison of various methods for engaging with a design.

While CAD tools excel when used for designing and mathematically analyzing models, they are not focused on showing CAD models in a human-centric context. In contrast, the electronic gaming industry has primarily focused on creating new human-centric experiences and empowering developers to create novel ways of interacting with digital content. Combining these domains allows operators to explore and understand CAD-based objects in an intuitive and informative way. To do that effectively, they need a pipeline to move models (often called content or assets) from CAD tools to game engines.

A game engine consists of the main game program containing the game logic; a rendering engine for generating animated 3-D graphics; an audio engine that includes the algorithms related to sounds; a physics engine that "enforces" the laws of physics so that physical interactions within the system are realistic; and artificial intelligence designed specifically for the game.[1,2] Traditionally, game engines have been used to develop video games, but their use is expanding into many other fields such as film, automotive design, architecture design, and construction.

Engineers today can create immersive experiences by importing their CAD models into a game engine, enabling them to review and iterate over designs much more quickly than they could with physical prototypes; once the workflow is established, reviews can be done in minutes. Furthermore, unlike traditional on-screen or printed CAD designs, immersive technology can provide an accurate sense of scale and can tailor an experience of the design to the specific user. As physical designs increase in size (human-size and bigger), the ability to understand the full scope and scale becomes even more challenging and important.

BUILDING A CUSTOM EXPERIENCE

Immersive technology is perhaps the most personal form of computing devised to date. Instead of being designed as a simple interface to a computer or an output from a computer, immersive tools are designed to be an extension of the operator. For example, when a user puts on a headset, the headset knows where that user's head and hands are. It knows which direction the user is facing, and in some cases, the headset's built-in eye tracking knows precisely what the user is looking at.

Many factors, including the user's age, height, weight, and clothing, affect how the user will experience an application. For many in the entertainment/gaming industry, adapting to the physical needs of the user is a new challenge. For those authoring immersive tools for use in design and manufacturing processes, this challenge becomes an invaluable asset, enabling teams to rapidly ideate, experiment, and communicate. Instead of talking about human-centered design with the focus on people in aggregate, immersive technology makes it practical to perform human-centered design for an individual, providing tools that are custom tailored to the user. In addition to their value in design analysis, custom immersive experiences can provide value to design teams and end users through personalized training and rehearsals of difficult tasks (especially if the users can wear equipment similar to what they would wear in the field).

IMMERSIVE TECHNOLOGY PROCUREMENT

As with all software, there are three options for getting needed functionality from immersive applications. The first is open-source software. Since this type of software is available for free on the internet, the value proposition is unbeatable if the software meets project requirements. Unfortunately, very few open-source CAD visualization tools have enough functionality and maturity to meet the needs of real-world use cases. The second option is commercial off-the-shelf (COTS) products. Immersive COTS software products do exist, and many of them provide a lot of useful capabilities. However, COTS software is generally limited to the functionality the vendor chooses to develop, and the source code is almost always unavailable. If the source code of a COTS product cannot be obtained, there could be security concerns when procuring foreign-developed applications. The third option is custom software.

Table 1. Comparison of various methods for viewing or experiencing a design

	Speed of Production	Relative Cost	Fidelity to Final Design
CAD on screen	Extremely fast	Inexpensive	Limited sense of scale and mass
Immersive experience	Fast	Inexpensive	No limitations on physical size; no sense of mass
3-D print	Slow	Moderately inexpensive	Limits on physical size that can be produced
Prototype	Slow	Expensive	Fully conveys scale and mass of final design
Final product	Very slow	Very expensive	Finished product

Note that blueprints are intentionally omitted from this comparison. They are an effective means of communicating specifications to manufacturing experts but are often a poor way of communicating with individuals who are using objects in the field.

Organizations can use available game engines to develop their own software to create immersive experiences (these game engines can also be open source, COTS, or custom built). Although developing a custom experience is slow and expensive, custom applications will do precisely what is needed while eliminating concerns about security issues related to the origin of the software. Given the expense of building a custom experience, however, teams should perform an analysis of alternatives before initiating a development effort: procurement is very often preferable to custom development so long as the project's requirements are sufficiently met.

The next section describes a nominal pipeline for moving a CAD model into to a game engine for the purpose of designing a custom experience, regardless of the specific game engine.

A NOMINAL PIPELINE: FROM CAD MODEL TO GAME ENGINE

Because of variations in software packages, no processing pipeline fits all approaches for developing an immersive experience. The nominal nine-step pipeline described below and shown in Figure 1 includes the major areas a design team should consider when planning to move a model from CAD into a game engine.

Much of the pipeline is designed around model geometry and graphical performance. In 3-D graphics, triangles are the atomic shape from which all models are created (much like pixels in an image) and are the metric used to determine model detail and computational complexity. In most designs, a large degree of accurate detail ensures that the design can be fully understood and is of high quality. For desktop CAD systems, poor rendering performance is often an unfortunate side effect of a complex and high-quality model. In an immersive experience, this side effect is both unfortunate and unacceptable: a system that does not render images quickly not only results in a subpar experience, but it can also make users physically ill. Different immersive devices have different performance needs, so the design team must define performance requirements up front. For example, a high-end graphics workstation with a powerful GPU will provide the ability to render more complex geometry (2,000,000+ triangles) than a stand-alone device (which will often have recommended polygon limits in the 50,000- to 100,000-triangle range).

Unlike text or image editing tools, which have generally accepted interchange formats, there is no accepted interchange format between the CAD and interactive media worlds. Common mechanical 3-D CAD formats, such as Inventor, Creo, NX, Catia, and SolidWorks files, are not natively supported by game engines. Media and game formats, such as 3DS, FBX, and glTF files, are not often available for CAD systems without additional cost. Establishing a content creation pipeline from professional design tools to interactive media tools remains one of the biggest hurdles to creating turnkey immersive experiences. Developers in both academia and industry are working on simplifying this workflow, but for now there is no cost-effective or de facto standard that can be recommended for a broad range of applications. Design teams should take care to address these interchange issues during project planning to ensure that enough time, staff, and funds are available to resolve this critical part of the effort.

Responding to the demand for tools to help engineers move their models into game engines, the industry has started to produce applications to automate some of this processing. APL has assessed many of these applications for various projects and has found that while some improve processing in certain cases, none provide a reliable pipeline that can fully automate the process. In time these products will likely become robust enough that engineers will not need to understand much of the underlying processing. Until then, it is important that engineers understand the overall process so they can design a content pipeline that meets the needs of their team and project.

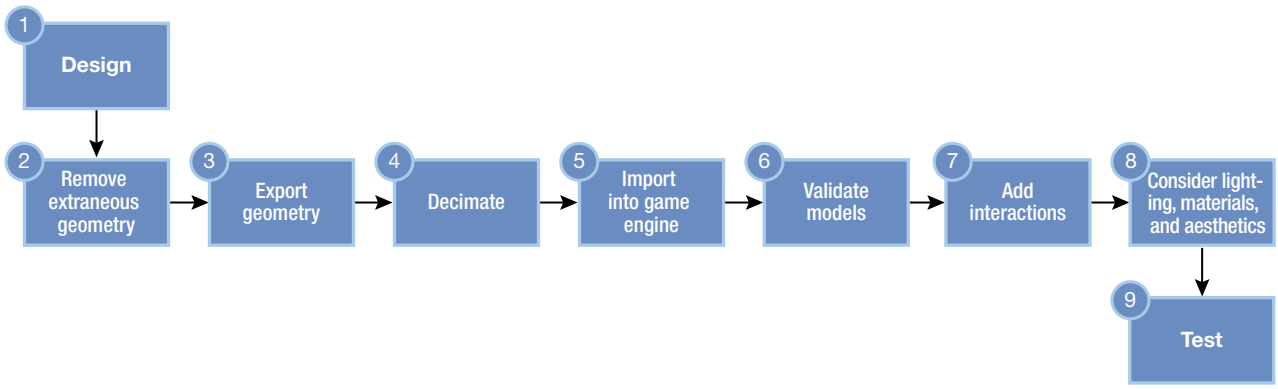

Figure 1. Notional asset pipeline for moving from a CAD model to a game engine. There is no one-size-fits-all approach to creating an immersive technology, but this pipeline includes the nine major steps a team will need to consider regardless of which software application they are using.

Step 1: Design

In most cases, exporting to a game engine for viewing in XR is not the end goal, but rather a step along the development process. Therefore, designers should not build their parts around viewing them in XR. However, they should follow good design practices to make some of the stages of the pipeline easier. Having well-structured component hierarchies and the ability to identify and remove fasteners on export are two design practices that can significantly reduce the amount of labor required for XR development teams.

Step 2: Remove Extraneous Geometry

One of the advantages of CAD models is that they can provide incredible levels of detail, but that detail can result in an experience that does not meet performance requirements. Very often the first step in reducing the model complexity is to remove extraneous geometry such as fasteners. Rendering these small details can be as computationally demanding as rendering large parts. In some tools it is also possible to render tiny details into a texture (a process called texture baking) to make a low-polygon approximation appear similar to a high-resolution model.

Step 3: Export Geometry

In this phase, the content that will be imported into the game engine is produced from the CAD tool directly or from a CAD plug-in, a conversion tool, or a model optimization tool. As mentioned, unfortunately, unlike for images or audio, there are no well-adopted standards to transfer geometry/textures between CAD and game engines. The team should consider these transfer challenges at the start of the project, as the limitations of certain file formats may inform how designs or interactive experiences are built. For example, STEP 242 is a popular neutral format that works well between CAD systems but is not supported by the Unreal or Unity game engines. OBJ is an older standard that is supported by many tools but inefficiently stores geometry, which can cause problems when working with high-fidelity models. OBJ has the additional disadvantage of requiring separate files for textures and materials. FBX is more efficient in terms of storage, but it is not supported in some CAD tools, and those tools that do support FBX often use different versions, leading to significant compatibility issues. The more recently introduced glTF format (a royalty-free format provided by the Khronos Group) shows promise, but it has not garnered broad adoption yet.

Step 4: Decimate

Once the extraneous geometry has been removed, designers or developers of real-time engines will have to determine whether the remaining geometry is too complex to be rendered at acceptable performance rates. If it is too complex, the geometry will need to be simplified before the model is imported into the game engine. Some tools, such as Blender, offer a built-in feature for decimation, and mesh optimization tools are available on the market. As a last resort, a low-polygon stand-in may need to be developed. Once the mesh is

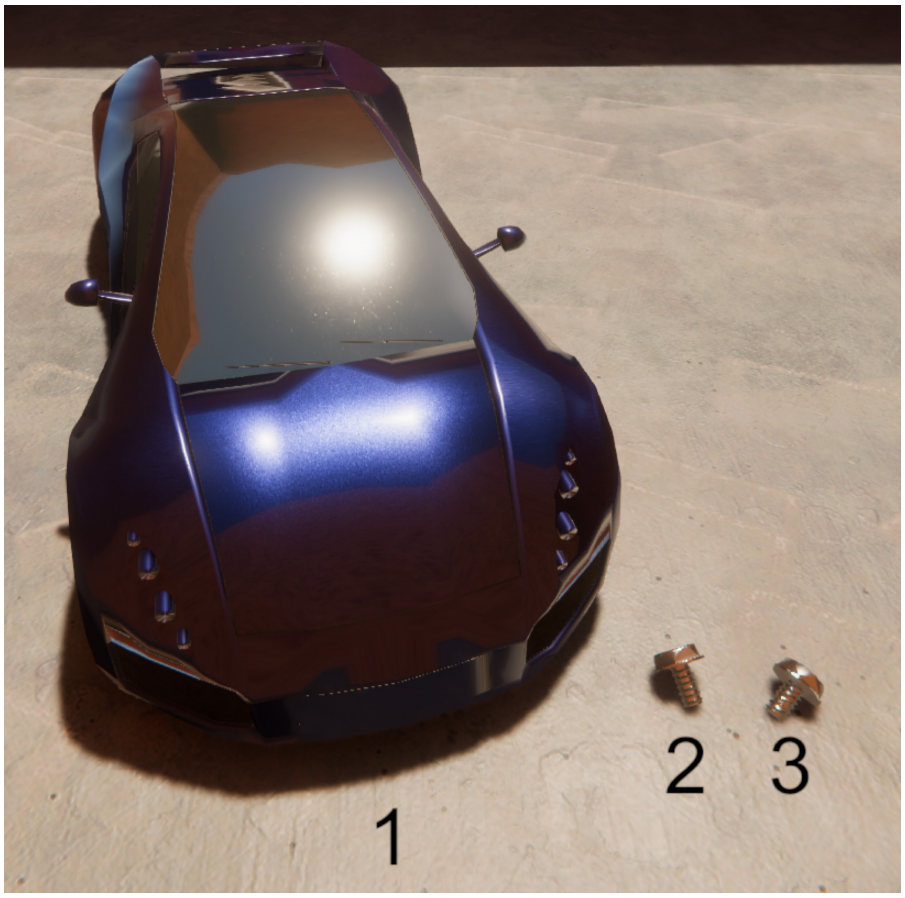

Figure 2. Models showing decimation. The models of the car (labeled 1) and the screw labeled 2 both contain approximately 14,000 triangles. The model of the screw labeled 3 has been decimated to fewer than 500 triangles. Designers should remove fasteners when feasible and then decimate geometry to reduce polygon counts enough to reach target performance metrics.

decimated, the designers should be consulted to make sure the output mesh is of sufficient quality to represent the design. Figure 2 shows the difference decimation can make.

Step 5: Import into Game Engine

The corollary to exporting data from one program is to import it into another. Once the geometry (and textures, if desired) are in an appropriate transfer format, the data will need to be imported into the game engine. Most game engines will handle their preferred formats automatically, but additional tools may be required for nonpreferred file formats. Many game-engine-specific settings and optimizations can be applied here (such as draw call batching[3] or ensuring proper parts of the model are made static[4,5]).

Step 6: Validate Models

Caution: Overlooking this vital step could result in significant consequences. Scale issues can arise when using tools that work in unit-less space, tools that assume different units, tools that work at different orders of magnitude (meters versus millimeters), or models that use both metric and US customary units. If the scale of an immersive experience is erroneous, designers will use bad data to make decisions to adapt the experience to the specific audience. Once someone has an immersive experience, it cannot be un-experienced, and it can be difficult to shake the first impression of a model when viewing it at human scale for the first time. Scale validation does not have to be complex: a stand-in cube of the expected size, acting as a fiducial marker, will help determine whether there are scaling issues. However scale is checked, it should be part of the regular development pipeline to ensure consistency throughout the development process. Tools commonly orient their coordinate systems differently, leading to models that are unintentionally rotated or inverted (most often along the y and z axis). An unintended rotation is obvious for most models, but highly symmetric objects should be carefully checked for unintended rotation.

Step 7: Add Interactions

Once the model has been imported into the game engine at the proper scale, interactions can be added. Moving hinges, lights that turn on or off, buttons, levers, and similar elements can add a level of realism that conveys what it would be like to interact with the physical object. Creating interactions usually requires a significant amount of code and can be labor intensive. However, when interactions are correctly authored, many can be reused across versions of a model, and sometimes across projects, allowing some of the development expense to be amortized over time. For AR applications, spatial logic (calibration with the real world, world sensing, object recognition, camera control) is handled in this step.

Step 8: Consider Lighting, Materials, and Aesthetics

Lighting is extremely valuable when making an environment feel realistic. Light conveys a lot of experiential information, affecting everything that can be seen and how it is seen (after all, without light nothing can be seen). Some understanding of various light sources and their uses is valuable for artists, designers, and developers working to create immersive experiences. It is important to remember that different lighting might be used for different purposes, as shown in Figure 3, where the model on the left is lit to provide visual clarity for a designer and the model on the right is lit to provide a more realistic experience. Lighting tools and techniques vary by engine, so some tool-specific training is recommended. Materials are properties that are attached to a model and provide visual characteristics to help objects appear

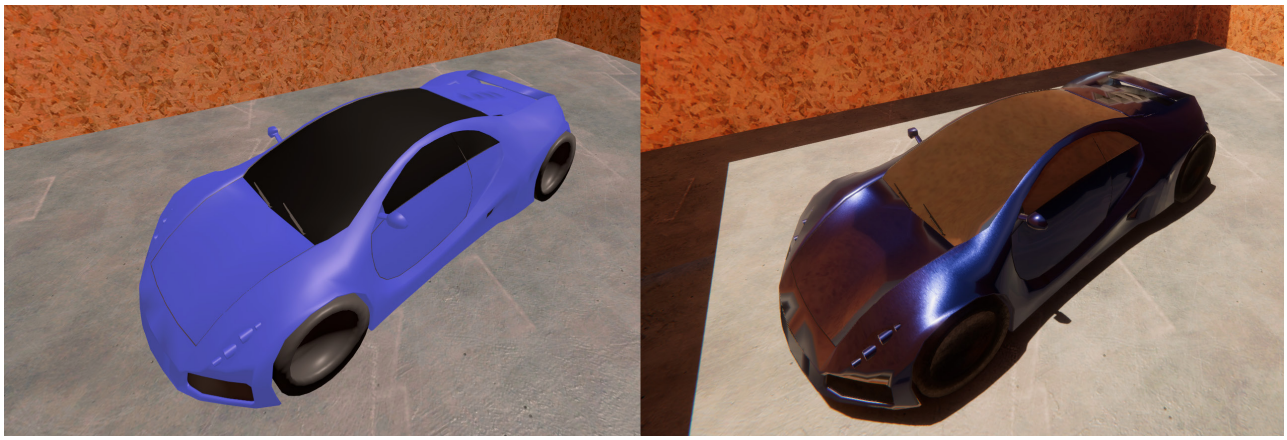

Figure 3. The difference lighting makes in an immersive experience. Identical models rendered in the Unity game engine with different lighting and material settings. The model on the left is lit to provide visual clarity for a designer. The model on the right is lit to provide a more realistic experience.

more realistic, such as reflectivity, smoothness, and similar properties. These materials are what make a model appear to be made of metal, plastic, cloth, wood, or some other substance. While materials are often used in quantitative analysis of CAD parts, during visualization these materials provide a wealth of qualitative information. Adding aesthetic touches, such as a context-appropriate sky and background, for example, is a fast and easy way to help boost the user's feeling of immersion.

Step 9: Test

The final step in the process is testing the imported design. Position, orientation, and scale of components should be checked, as well as the visual quality of the render. Testers should also assess the system from a performance perspective, taking note of any time there is a reduction in frame rate or visual clarity. Finally, interactive components should be tested. Depending on the level of interaction, this can be tested manually, with a record/playback mechanism, or with a fully automated testing tool/service. A formalized test plan is often required to ensure that tests provide sufficient coverage of the application and are performed consistently throughout development.

CONCLUSION

Immersive technology gives design teams a robust tool to increase their own design intelligence as well as that of their end users. This article presented a nominal pipeline for taking a CAD model into a game engine to create an immersive experience that can provide value in numerous ways, such as reducing the time it takes for design teams to review and iterate on designs, providing personalized training and rehearsals of difficult tasks, and imparting complex information quickly and realistically.

REFERENCES

1. "How do game engines work?" *Interesting Engineering*, Nov. 2, 2016, https://interestingengineering.com/how-game-engines-work.
2. "Game engine," *Wikipedia*, last edited Apr. 28, 2020, https://en.wikipedia.org/wiki/Game_engine.
3. "Draw call batching," *Unity User Manual (2019.3)*, San Francisco, CA: Unity, updated Oct. 26, 2017, https://docs.unity3d.com/Manual/DrawCallBatching.html.
4. "Static meshes," *Unreal Editor Manual*, Cary, NC: Epic Games, https://docs.unrealengine.com/en-US/Engine/Content/Types/StaticMeshes/index.html (accessed Jun. 8, 2020).
5. "Static GameObjects," *Unity User Manual (2019.3)*, San Francisco, CA: Unity, published Jun. 5, 2020, https://docs.unity3d.com/Manual/StaticObjects.html.

Blake A. Schreurs, Information Technology Services Department, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Blake A. Schreurs is a Senior Professional Staff member in APL's XR Collaboration Center. He has a BS in computer science from Davis and Elkins College and an MS in computer science from George Mason University. Blake is a virtual reality (VR) and augmented reality (AR) expert, with a focus on using commodity tools and equipment in novel ways to drive positive impact for sponsors. He has expertise in technology readiness assessment, ideation and design, application development, software architecture, and performance tuning. He has extensive knowledge of interactive and real-time visualization technology, with a focus on improving situational awareness for safety, site protection, and Blue force protection. He is one of the cofounders of APL's Immersion Central, an XR community of practice with over 180 members. His email address is blake.schreurs@jhuapl.edu.