TARGET

CAD to VR-tool

Introduction

The bottom line is, running Virtual Reality (VR) applications requires a lot of computing power. Having to render a high-resolution image at the target frame rate *twice* (one image for each eye) puts a strain on the hardware, especially the graphics card. We'll use the HTC Vive as an example, which means rendering a 2160x1200 (1080x1200 per eye) image that is updated 90 times per second. The exact numbers are not important here, but the very nature of VR rendering means we have to pay special attention to the content displayed to the user.

The performance limit can, to certain extent, be offset by using more powerful hardware, but merely brute-forcing our way through is not an ideal option. These performance considerations should not be ignored, though, as doing so can lead to poor user experience and even physical discomfort for the user if the frame rate drops too low. This is especially important when importing already existing CAD-models to be used in a VR environment, as they are often very detailed (the geometry is very dense and detailed, and things such as internal components are often included).

This document regards specifically high-end VR, which means a powerful PC and a high-end consumer VR headset system such as the HTC Vive or Oculus rift, and a game engine used for the visualization such as Unity3D or Unreal Engine.

If targeting for a mobile VR platform utilizing a smartphone as the hardware, the same ground rules apply, but the models will need quite a bit of extra work to run on this less powerful hardware.

Planning

Planning the wanted features and the scope of the VR environment is a necessary step, as it will lay the groundwork for the future steps! Consider at least the following:

- What platform will be used? (Mostly PC vs Mobile VR, a HUGE difference in computational power)
- Where can the user go / what can the user see?
- What are the interactions?
- What will the model look like? Does the scene need realistic materials and / or lighting?

In the end it all boils down to the complexity and scope of the model; the larger the model or environment is, the less details we can cram into that space. On the other hand a model of a single product can most likely be used as it is, requiring little or no preparation work to be brought into the game engine (for example a large industrial hall vs a single mechanical product). It should also be noted that if the model is very large or complex even after optimization, sacrifices may need to be done in other labour-heavy or performance-impacting features such as realistic materials or lighting.

Generic workflow for importing a CAD model

This is a simplified workflow outlining the steps required for bringing a CAD model to be used in a game engine. Some of the later steps will involve specific software and require people with specific skills in said tasks, but they are included for the sake of clarity to help understand the overall process. Other than that, this is a very generalized workflow that should apply to a variety of CAD programs and models.

1. Removing unnecessary parts and cleaning up

Only include the features of the model that are actually needed. This means objects or parts outside of the area of interest, helper objects or the like, as well as parts that will be completely hidden inside or obstructed by other objects. CAD models often include the internal components and mechanisms involved, so unless it is planned that the user will interact with these in some manner, they should all be left out.

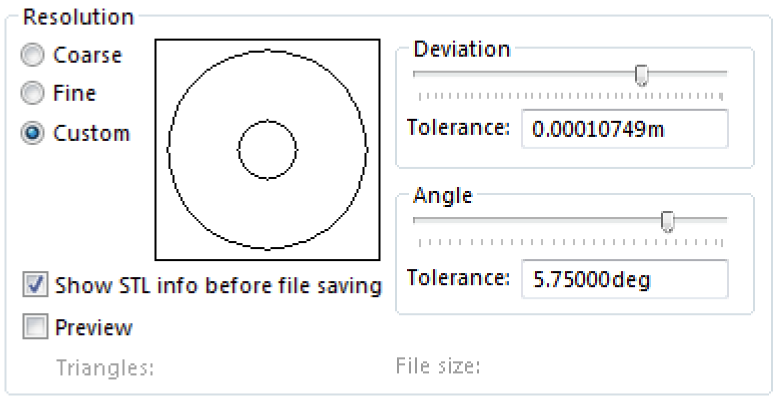
If possible, this should be done already in the CAD application, as when exporting, depending on the software and file formats used, some of the structure of the model may be lost (grouping, layers, etc). This will make it easier to work on the model further down the pipeline and helps keep the file sizes down.

Also try to make sure there are no overlapping surfaces in the model as these will otherwise need to be cleaned up later. Make sure the parts are laid out in a logical fashion, so for example, a single wall segment doesn't consist of a dozen small parts unless they are clearly separate pieces. Avoid merging everything into a single giant chunk, though.

It is likely that some parts will need to be revisited or added later after testing, so sticking to a common coordinate system and scaling across different files and parts is highly recommended as it will make making changes a lot easier.

2. Exporting the model

CAD models are parametric by nature, but to be used in a game engine they need to be converted to a polygonal mesh format. Depending on the software there should usually be a setting to adjust the resolution of the exported mesh. Here's an example of the mesh resolution options in Solidworks when exporting to STL format:



Resolution

Coarse

Fine

[✓ Custom]

[✓ Show STL info before file saving]

Preview

Triangles:

Deviation

Tolerance: 0.00010749m

Angle

Tolerance: 5.75000deg

File size:

Figure 1. *Mesh resolution settings for STL file format in Solidworks.*

The resolution used will again depend on the complexity of the model and software used, but it's often best not to go too low as it may introduce distortion in the model. It is also easier to reduce details later than try to salvage a too low-resolution mesh.

Picking a file format for the export is largely a matter of preference and should be communicated with the person responsible for working with the model further down the pipeline. A generic polygonal mesh format such as **STL**, **OBJ** or **DXF** is usually a safe choice. If for some reason these formats are not available, a well-supported generic CAD format such as **IGES** or **STEP** is the next best option, meaning the conversion to mesh will be handled later on.

If there are surface textures used with the materials, these are usually lost when exporting and should be delivered separately, if needed. Material setup usually carries over, so it is recommended to set up the material definitions before exporting if they are relevant.

3. Preparing the model for game engine usage

For a quick and dirty solution using a model directly exported to a mesh format can be used in a game engine, but usually it will be imported in a software more suited towards polygonal mesh modeling, such as 3DsMax or Blender for additional tweaking and making sure everything is working as intended.

The model will be optimized by simplifying too dense geometry and is checked for any possible problems such as overlapping surfaces, errors with surface normals or smoothing information. Possible additional tasks carried out in this step are animating moving parts or creating / cleaning up the UV layout to make sure the possible surface textures are displayed correctly. It is recommended to test the model in the VR environment early on before delving too deep into optimization to see how the model performs and to see which parts may require further editing.

Any major flaws with the model should be addressed here before continuing further. Smaller problems can be fixed by re-modeling the problematic parts by hand, but it should not be the default option to make everything twice. A special case here is mobile VR development, though, as due to the performance limitations it can actually be faster to model some parts by hand based on the CAD geometry, as automatic decimation tools can sometimes fail with very complex geometry.

Figure 2 shows a side by side-comparison of a model part that was imported from a CAD software without any optimization (on the left) and after some automatic optimization work in a modeling application (right). Even for a small part like this, the geometry is nearly 70% lighter (742 vs 286 triangles, which is an important performance metric) without sacrificing the apparent fidelity of the object.

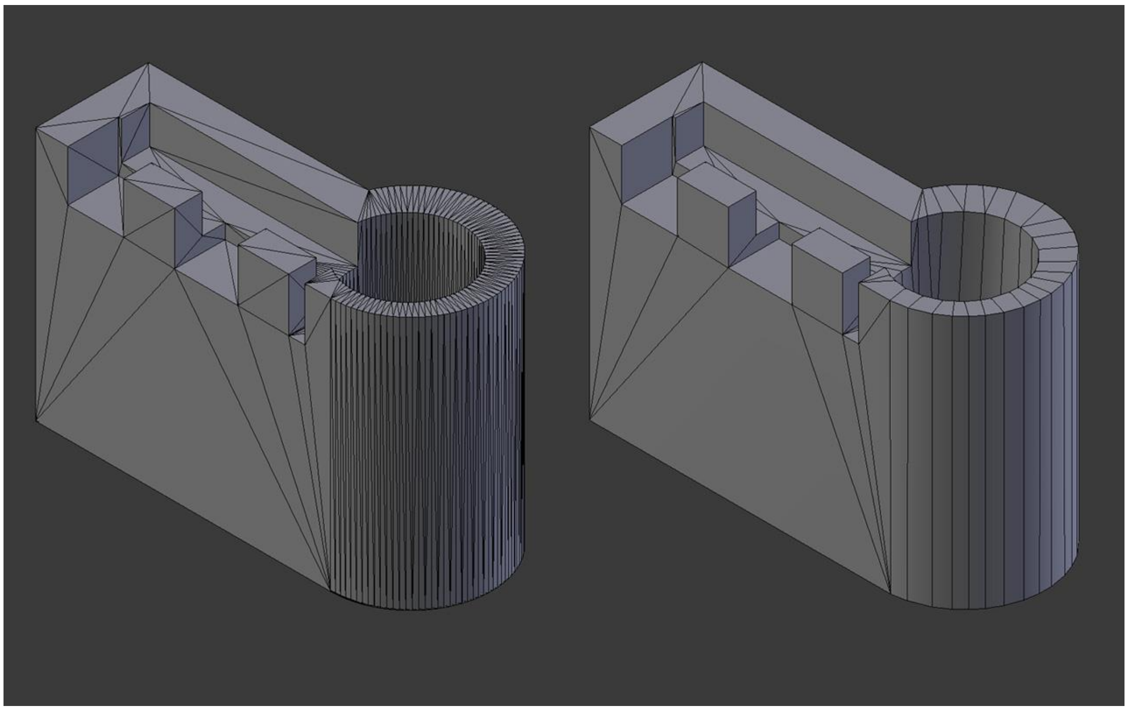


Figure 2. *The impact of even a quick optimization pass is apparent. Note the unnecessarily detailed definition of the rounded shape on the left.*

The benefit gained from optimizing the models depends heavily on the shape, but the example illustrates just how beneficial even a quick optimization pass can be. The gain could easily be tenfold or more, if the original object had some hidden internal parts that would have been removed. Spread over possibly hundreds of objects in a larger scene this difference does add up and can mean the difference between a great experience and not being able to run the scene in VR at the required frame rate.

4. Engine-specific steps

The optimized model will be brought into the game engine of choice by exporting it to a intermediary format supported by the engine, such as FBX or OBJ. Depending on the scenario, a lot will happen inside the game engine itself as it will be responsible for rendering the scene and handling the user interactions. This is also the step where some of the less obvious parts that need tweaking will be spotted as all the interactions are put together, and should be fixed by backtracking to the previous step.

Some of the typical things that happen in-engine are:

- Setting up the scene and collision geometries
- Adding textures and applying surface materials
- Creating and applying animations
- Programming interactions and adding VR functionality
- Adding the user interface
- Testing and debugging
- Compiling the final application

Additional notes

As there are a lot of variables involved when bringing a CAD model to a VR environment, sometimes things require a bit of experimentation to work. Here are a few general tips and notes that might prove helpful.

- Consult the person responsible for working with the game engine for any specific needs before preparing or exporting the model. Communication is key!
- If the model has a lot of errors or missing / distorted parts, try a different file format or adjust the mesh resolution settings higher. Although generic 3D formats are a safe bet, sometimes there are compatibility issues with certain software and certain formats.
- Test in VR as early possible. Getting to view the model in real scale and in the right environment is a great way to spot issues that might otherwise be left unnoticed.
- There is a global trend with major CAD software companies towards integrating VR as a part of the design / model drawing process, but for custom interactions and graphically intensive real-time applications game engines are the way to go.