10.2.11 Render - Blender Render Engine - Performance Consideration

Performance Considerations

Optimizing Render Performance

“A watched pot never boils” is the old saying, but you may wonder why your render takes so long to create, or worse, crashes mid-way through! Well, there is lots going on and lots you can do to speed up rendering or enable a complicated render to complete. Also, it is possible to render a very complicated scene on a mediocre PC by being “render-smart”. Here’s a “top ten” list of things to do or not do in order to speed up rendering or even avoid crashes during scene render. Some options may decrease the quality of your render, but for draft renders you may not care.

If you get the message “Malloc returns nil”, in plain English that means the memory allocator tried to get more physical memory for Blender but came back empty-handed. This means that you do not have enough memory available to render the scene, and Blender cannot continue. You will need to do one or more of the following tasks on this page in order to render.

Hardware Improvements

- Install more system memory.
- Upgrade your CPU to a multi-core/multiprocessor
- Upgrade your OpenGL video drivers
- Get faster memory, up to your PC’s motherboard limit.
- Use or set up a render farm using all available PCs in your house, or use a render farm.

Operating System Configuration

- Increase Blender’s processing priority through your OS.
- Increase your swap file space used by the OS for memory swapping. Also called virtual memory pagefile size, up to the size of your physical memory.
- Use a system-monitor to check if any other processes are using significant CPU or RAM, which can be closed.
- Render in *background mode* (from the command line), saves extra memory.

Blender Settings

- Increase the MEM Cache Limit in the User Preferences System & OpenGL tab.
- Switch to an Orthographic camera, and render your own “parts” of the scene as separate images, and then paste those parts together in GIMP. An old trick in making your own panorama with a real camera is to take three or so pictures of a very wide (beach sunset) scene, where you take one picture, rotate to the right, snap another, then another, and when you get the pictures developed, you overlap them to make a very wide landscape image. Do the same in Blender: render out one shot to a file, then move the camera to look at a different area of the scene, and render that shot. Each shot will be of a smaller area and thus take in fewer polygons/faces. Be sure that when you position your camera that you snap overlapping shots, so that you can then match them up. If you don’t want to use GIMP, you can use compositing nodes and the Translate node to match them up in Blender.
- Minimize the render window (and Blender if rendering to an internal window). ATI users report dramatic speedup on a per frame basis, which adds up over the frame range.
- Use the Big Render script to render sub-sections of the overall image, and then paste them together.

Scene and Specific Objects

- Remove lamps, or move them to unrendered layers, or tie them to layers.
- Turn off some lamp’s shadows, using only one or two main sun lamps to cast shadows. A few “shadows only” lights will render faster than every light having shadows on.
- Use Buffer Shadows rather than ray-traced Shadows
- Bake your shadows using Render Baking Full Render bake on surfaces that do not move. Use that texture for that mesh, then disable shadows for that material.
- Simplify meshes (remove polygons). The more vertices you have in camera, the more time it takes to render.
- Remove Doubles, or use the Decimator mesh edit feature.
- Remove Subsurf and Multires modifiers.
- Delete backsides of meshes (removing unseen geometry).
- Render just a few objects at a time; in the beginning of your project, render the background objects and sets that will not change and will always be in the background.
- Put the buildings on another layer, and through render layers, don’t render them. Then composite them back in later.
- Make the camera static so that you can better accomplish the above two ideas.
- Avoid use of Area lights.
- Make materials Shadeless.
- Render Bake AO and textures, and then make those materials Shadeless.
- Decrease the Clip distance for spot lights.
- Decrease the Clip distance for the camera.
- Turn off world AO.
- Turn off Material SSS.
- Use smaller image textures. A 256x256 image takes only 1% of the memory that a 2k image does, often with no loss of quality in the ultimate render.
- Reduce Subsurf. Each level quadruples (4x) the number of faces from the previous level.
- Reduce Multires.

- Make a matte render of background objects, like buildings, and put the image of them on a billboard in the scene instead of the object themselves. This will reduce vertex/face count.
- if you have lots of linked instances of an object, use DupliFaces, as these are instanced. If you have 100 of them, Blender will only store the geometry for 1 (Instances themselves take a small amount of memory).

Render Settings

- *Output Panel* - Disable *Edge* rendering. - *Save Buffers*.
    - Render to an Image Editor window, not a pop-up. *Render Window.*
    - Use multiple *Threads* on a multi-core CPU (with multiple *Parts*).
- *Render Layers Panel* - Render only the Layers of interest. - Render with all lights set to one simple spot (enter its name in the *Light:* field). - Render with one material override (enter its name in the Mat: field).
    - Disable unnecessary Render Passes, such as *Z*, or only render the pass of interest, such as *Diffuse*.
- *Render Panel* - Turn off *Shadows*. - Turn off *Environment Mapping*. - Turn off *Panoramic Rendering*. - Turn off *Raytracing*. - Turn off SSS Subsurface Scattering. - Turn off or lower oversampling/aliasing *OSA*. - Turn off or lower *Motion Blur.*
    - Render in Parts. This will also allow you to render HUGE images on a weak PC. On a multi-core PC, it will assign a thread to each part as well.
    - Increase the octree resolution.
    - Render at a percentage size of your final resolution (like 25%).
    - Turn off *Fields* rendering.
    - Use *Border* rendering to render a subset of the full image.
- *Anim Panel*
    - Decrease the frame count of the animation (and use a lower framerate for the same duration of animation). For example, render 30 frames at 10 frames per second for a 3-second animation, instead of 75 frames at 25 frames per second.
- *Bake Panel*
    - Bake Full Render - create a UV Texture that colors the objects based on materials, and then use that UV Texture shadeless instead of the material.
    - Bake Ambient Occlusion only.
    - Bake textures for objects.
    - Baking Normals or Displacement does not speed up render time, and are used for other things.
- *Format Panel* - Render at a lower resolution. Smaller pictures take less time to render. - Choose a faster CODEC or CODEC settings. - Render in black and white (*BW* button). - If using FFMPEG, do not activate *Multiplex audio*. - If using FFMPEG, *Autosplit Output* (*Video* panel button).
    - Render only RGB if you just need color; the A channel (*RGBA* button) takes more memory and is unused when saving a movie file.

Multi-Pass Compositing

Another strategy that can be used to address the problem of long (re-)render times is to structure your workflow from the ground up so that you make aggressive use of *compositing*, as described in the “Post-Production” section. In this approach, you break down each shot into components that can be rendered separately, then you

combine those separately-rendered elements to achieve the finished clip. For instance:

- If the camera isn't moving, then neither is the background: only a single frame is needed. (The same is true of any non-moving object within the frame.) These individual elements, having been generated *once,* can be re-used as many times as necessary over as many frames as necessary.
- Both shadows and highlights can be captured separately from the objects that are being illuminated or shadowed, such that the intensity, color, and depth of the effect can be adjusted later without re-rendering.
- Start by using lights that do not cast shadows. (Shadow calculations are big time-killers.) Then, use "shadow-only" lights (which cast shadows, but do not cast light) to create shadows *only* where you judge that they are actually necessary. (It is very often the case that only a few of the shadows which could exist in the scene actually matter, and that the rest of them simply won't be noticed.)
- Tricky lighting situations can be avoided by handling the objects separately, then combining the individually-rendered clips and "tweaking" the result.

This is a very familiar idea. Modern sound recordings, for example, always use a "multi-track" approach. Individual components of the song are captured separately and in isolation, then the components are "mixed" together. The "final mix" then goes through additional processing stages, called *mastering*, to produce the finished product(s). (In fact, the features and design of modern sound-processing software are directly comparable to that of Blender's node-based compositor.)

There are compelling advantages to this approach:

- You have options. If something is "not quite right," you don't necessarily have to start over from scratch.
- In practice, the deadline-killer is *re-* rendering, which ordinarily must be done (in its entirety) just because "'one little thing' about the shot is wrong." Compositing helps to avoid this, because (ideally...) only the specific parts that are found to be in error must be repeated. (Or, maybe, the error can be blocked out with a "garbage matte" and a corrected version can be inserted in its place. No one will ever know!)
- It's also possible that you find yourself saying, "okay, that's *almost* what I wanted, but now I'd like to *add* this and maybe *take away* that." A compositing-based approach enables you to do just that, and furthermore, to do so *non-destructively.* In other words, having generated the "addition" (or the "mask") as a separate channel of information, you can now fine-tune its influence in the overall "mix, " or even change your mind and remove it altogether, all without permanently altering anything.
- By and large, these stages work *two-* dimensionally, manipulating what is by that time "a raster bitmap with R, G, B, Alpha *(transparency...)* and Z-Depth information," so they're consistently fast.
- Since each discrete rendering task has been simplified, the computer can carry them out using much fewer resources.
- The tasks can be distributed among several different computers ... even less-powerful ones
- "After all, the scene doesn't actually have to be *physically perfect,* to be *convincing. "* A compositing-based approach lets you take full advantage of this. You can focus your attention (and Blender's) upon those specific aspects of the scene which will actually make a noticeable difference. It is possible to save a considerable amount of time by consciously choosing to exclude less-important aspects which (although "technically correct") probably won't be noticed.

Of course, this approach is not without its own set of trade-offs. You must devise a workable asset-management

system for keeping track of exactly what material you have, where it is, whether it is up-to-date, and exactly how to re-create it. You must understand and use the “library linking” features of Blender to allow you to refer to objects, nodes, materials, textures and scenes in a carefully-organized collection of other files. You need to have a very clear notion, *in advance*, of exactly what the finished shot must consist of and what the task breakdown must be. You must be a scrupulous note-taker and record-keeper. But sometimes this is the best way, if not the *only* way, to accomplish a substantial production.