

3P98 Project

David Ketler

January 11, 2010

Contents

0.1	The Project	1
0.2	Impressions	1
0.3	Work Process	2
0.4	Toolset used	3
0.5	Conclusion	3

Abstract

A 30s animation was done in Blender of a TARDIS from *Doctor Who* flying around in space, to a backdrop of a wormhole to the theme of the 10th Doctor. This was done using the animation suite in Blender along with some video editing tools (these tools will be listed later in the paper). Work on the animation opened some possible followup projects that could be done to the benefit of a future class or the Blender community.

0.1 The Project

For the project I chose to do a short animation in Blender; I chose this because I wanted to learn at least some of blender, and learn to work with models, even though I'm not typically artistically inclined or pursue artistic endeavours I thought it would be a good way to force myself to learn some and get some practice. I used a model licensed under the Creative Commons produced by Richard Marklew (<http://www.blenderwho.com>). This model is of a TARDIS, the spaceship 'The Doctor' uses in *Doctor Who*. The initial plan was to render a chase scene involving the TARDIS and some alien ships ('the Daleks') but due to the way the models were done and the way Blender behaves it was not viable. This will be described later.

The rendering was done with the animation tools in Blender, output to a raw AVI, encoded with 'VirtualDub' with the audio track added. The audio track is the theme song for the 10th doctor from *Doctor Who* ©BBC. The background of the render is a wormhole from the intro to *Doctor Who*.

0.2 Impressions

I ran into a few issues with the production of this animation, largely due to how Blender operates. The first, as mentioned previously with the Dalek model – to move a model properly without skewing of vertices one must either bind them to a parent object (essentially so they scale off of that parent object and do not get distorted) or join all the vertices in the mesh into one cohesive object. The models produced by Richard Marklew have controls for the lights on the top of the Dalek's head and other items, and these do not bind to the mesh properly; this caused a number of issues while attempting to animate the movement and it was not viable to continue down this path. This is more due to my inexperience with the application than a design problem.

I feel there are some issues with Blender itself, the first being the interface to the application – it runs into the same problem many other F/OSS applications do in that they do not have proper designers for the interface and you run into usability problems (much like The GIMP, KDE's settings managers, OpenOffice.org and so on), this creates a relatively steep learning curve that may not be so prohibitive to new users otherwise. I also ran into a number of crashes, both application crashes and I had Blender hardlock my system as well; hopefully the next version, v2.50 (which sees a number of interface changes in the style of Adobe's CS4 suite) will fix the stability issues; otherwise the application is not bad.

I was quite surprised by the system resources used by Blender to render the video (each successive frame, technically), I was anticipating at least some graphics library acceleration in the form of say, utilizing OpenGL to perform some of the calculations making use of the video card or shader processors, but in Blender all operation is done on the CPU. As such it can take a great deal of time to render a single frame; my final render at 800x600 resolution, 5x AA with raytracing enabled had an average render time of ~ 8 s a frame. This was with the CPU on my system maxed out: a Phenom II X4 955 (quad core 3.2GHz), a good future project would be to attempt to contribute to the project and try to make some performance improvements and get them upstream, possibly utilizing openCL to accelerate the raytracing and shadow generation, which is a large percentage of the render time. Also, Blender only has options for complex shadows and raytracing for lighting, there are no rough options for test renders, so this slows the workflow down a great deal – waiting for a full render to process to check on the progress of the animation, positioning of objects etc. another future project could be to implement some of the quick and dirty algorithms learned in class for lighting (and utilize some simpler rough shadows) and push additions to Blender to allow for quicker test renders.

An initial consideration for the project was to write the functions to import a model into OpenGL from Blender and essentially do an animation in-engine. This would be a good future project, at least to write a complete function set for the importing of Blender models (the Blender filetype is complex, but one can export to an Object file, and this would be imported). This would be a good followup project that I would like to consider, perhaps it would be of use to future courses and there does not appear to be any existing standard or opensource toolset to allow for this; this would encourage the use of Blender and OpenGL over DirectX and 3dsmax (as DirectX appears to have a .x filetype that can be directly processed by DirectX, as opposed to writing one's own import code to get a model into OpenGL from Blender).

0.3 Work Process

For this project I merely had to work in Blender in order to develop the animation with the other small programs to pair up the different files, and to compress the video. The filetypes and OpenGL importing were researched but an animation was decided upon, so making use of that time as a followup in the form mentioned above would be a good idea so that time is not wasted. Minus the application issues and the model issues there were not a great deal of snags, it was merely difficult learning the application and getting the animation done; animation: more complex than anticipated.

I feel even if there is not a desire to go into animation as a future path the knowledge and experience is useful. Practice with keyframing and animating was enjoyable and is good knowledge I believe (as I am an anime fan, and a video game fan, like most every CS student; so this is of interest).

0.4 Toolset used

- Desktop Computer:
 - AMD Phenom II X4 955 (3.2GHz)
 - 4GB DDR3 RAM
 - ATi RADEON 5770 1GB
- Blender v2.49
- Virtualdub 1.5.10 (to compress video and mux the audio in)

On this system setup rendering the 30s animation took ~ 8 s per frame, with ~ 750 frames giving a complete render time of 100 minutes = 1hr 40m approximately. Some frames render a little faster than others, closer to 7.5s, some slightly over 8 depending on the camera shot. It was quite surprising that it does in fact take that much time for a relatively simple scene, but this is due to a reliance on the CPU and software rendering.

0.5 Future Extensions

Some future work to follow would be improving the render performance of Blender with some openCL extensions using the lighting models discussed in class to speed up rendering – either for rough renders to test work, or to speed up the raytracing and other factors of rendering a frame that slow down the performance of the application. Another option is writing the framework for importing Blender Object files to allow students to make use of Blender models without needing to write the extra code to pull the model into OpenGL. The Obj filetype is relatively simple with the listed vertices, scale matrix and a surface type list; a toolset could be generalized to allow for easy use of these meshes and let students work within OpenGL.

0.6 Conclusion

Some of the work in the project may have been simplified using a graphics library like OpenGL; having direct control over some things, being able to procedurally generate content (for example, generating a wormhole from

particles for the TARDIS to fly through would have been beneficial, this is not really viable in Blender [or other 3d suites presumably, as that's not really a necessary featureset]) would have been beneficial and have greatly contributed to the quality of the project. The application also lacks control over transparencies in any easy way, I feel this would likely be easier in OpenGL and would have been a positive effect. Camera control in OpenGL may have been easier; I would have liked to achieve multiple shots but this is managed in Blender through Python scrips and Python on my system was not behaving, and the scripts are far from intuitive, this would be a good feature to be integrated into Blender; something that is quite easy in OpenGL but lacking from a suite like Blender.

Overall the project was a good experience, and I think it opened up some future projects in the form of the model handler for OpenGL and possibly contributions to the project at a later time. I went into it knowing how complex animation and rendering can be, but it was not really clear until actually working with it. The high performance requirements surprised me, even while expecting it somewhat, I did not anticipate it being to that level. It was an eye-opening experience. I do wish the render had turned out a little better, but the featureset just wasn't there to achieve the desired effect without a great deal of dirty hacks. This is where directly working in a graphics engine would have been beneficial, or writing the scripts for Blender in Python.

It was an enjoyable project that I feel was quite beneficial, and hopefully I will be able to implement at least the OpenGL import tools as a future project and allow some other students to benefit from this work.