**Introduction**

DSABR (the ‘D’ is silent) is a command line utility that facilitates automated rendering of computer-animated films created in the 3D computer graphics software product Blender. This rendering is done by submitting tasks to many computers, all part of a Condor cluster, using the distributed systems framework Work Queue. Work Queue manages workers, tasks, and the transfer of necessary files to remote computers. This means that even if a computer does not have the necessary application (the Blender executable), Work Queue has the capability of sending the executable. Even more impressive, using a feature of Condor calledRocking DSABR is able to Rock to the Condor pool at University of Wisconsin-Madison. This means that, in addition to the many computers here on campus, DSABR can take advantage of computers in Madison.

**How Does It Work?**

DSABR is written in the scripting language Python and uses the Work Queue API to control the submission of tasks to the Condor cluster. Here is a typical workflow for the user of the command line tool:

1. **Start**
2. **Submit Workers**
3. **Submit File to DSABR**
4. **Submit Tasks to Work Queue**
5. **Render Frames**
6. **Stitch Frames Together**
7. **Animated Movie**
8. **End**

**Step 1: Submit Workers**

- The number of workers submitted is dependent on the number of frames to be rendered, although the exact relationship is still being researched.
- Even if 100 workers are submitted for a given project, there is no guarantee that all 100 will connect. This is because there are a limited number of computers in the cluster on campus—the remaining workers need to flock to Madison, which takes much longer.

![Average Maximum Number of Connected Workers relative to the Number of Submitted Workers](image1)

- In the above chart, the average maximum number of connected workers is plotted against the number of submitted workers. The least and greatest maximum are shown with the whiskers. This clearly shows the soft limit of between 32 and 36 maximum connected workers corresponding to the workers on campus.

**Step 2: Submit the File**

- At its most basic, all DSABR needs to know is the location of the project and the number of frames to be rendered.
- Optionally, you can supply arguments that allow for debugging and logging, specifying the output file’s name and filetype, and specifying the intermediate image filetype.
- Upon submission, DSABR divides the project up into its frames. Each frame constitutes a Work Queue task which is sent off, along with the Blender executable, to the workers started in the previous step.
- Each Work Queue worker renders a given frame, sends the resulting image back to DSABR, and receives another task.
- When all images have been rendered, DSABR stitches them together into a movie.

**Step 3: Wait!**

- Although DSABR will vastly improve the rendering time of the user’s project, depending on the complexity and number of frames being rendered it could still take quite a while.
- The improvement over rendering on a single machine can be seen in the following chart:

![Average Render Time with respect to the Number of Submitted Workers](image2)

- Also noteworthy is the increase in the render time of 500 frames when 40 workers were submitted. Because it doesn’t take very long to render 500 frames, the extra time it takes to send the Blender files greatly impacts the overall render time.

![Average Render Time with respect to the Number of Submitted Workers](image3)

- The above chart illustrates how complexity can have an impact on the render time of the project. Dolphin.blend has no light sources and so all that needs to be rendered is the silhouette of the dolphin. On the other hand, CarPark.blend makes use of many light sources and reflective surfaces and so is a much more complex project.

**Future Work**

- Currently, DSABR only has a command-line front-end. One of the future goals of the project is to create a web portal where the user can upload the file she wants to render. As it is currently envisioned, the user will hit submit, the file will be processed, and the user will get an email on completion with a list of various file formats -- possibly even an auto-uploaded version on YouTube!
- The current implementation of DSABR only accepts Blender projects. It would be more useful if it could be adapted to other 3D computer graphics software programs that are more widely used in industry settings.

**Acknowledgments**

I would like to thank the Office of Research and Sponsored Programs for funding the project and a trip to UW-Madison’s Condor Week. I would also like to thank Mike McMann and Natalie Wolf for working on a 3D animated scene to test DSABR, and UW-Madison’s CH2E for letting me use their computing grid.

**Follow Me**

DSABR is open source and freely available at https://bitbucket.org/tboettcher/renderfarm. Follow the project there or at my personal blog, futureyb.org, where I make periodic updates on the project’s progress.

**DSABR**

_Distributed System for Automated Blender Rendering_

_by Travis Boettcher, Natalie Wolf, Peter Bui, and Mike McMann_  

_Computer Science Department, University of Wisconsin-Eau Claire_