Criterion E: Evaluation

Criteria for Success (Defined during Planning Phase):

1. Open a file dialog box, open an image file, and show the image
    a. **Works**
2. Convert Image to a 2D array of doubles
    a. **Works (3D array)**
3. Calculate vertexes for each triangle
    a. **Works**
4. Calculate normals for each triangle
    a. **Works (had to adjust the order to make sure they pointed the right way)**
5. Display a button to generate the output and ask where to save to
    a. **Works (also added overwrite to save to last location)**
6. Write triangles to a .stl file, test output on MakerBot
    a. **Works (see videos for printed example)**
7. Add sliders to adjust size of the model
    a. **Works**
8. Add ability to use 2 pictures, one for the underside, the other for the top

a. **Works**

9. Add a rotate-able 3D rendered solid of the model as a preview

    a. **Works using Java3D (Java3D recomputes normals which takes longer than our algorithm; crashes with an image 2300*2300)**

10. Export as executable jar file

    a. **Works**

Conclusion

On the March 30th 2016, I had a post-development interview with my client to talk about the final product, of which a transcript can be found in Appendix 2. We concluded that I successfully implemented all important features and had them running effectively and efficiently. I added other features not discussed originally in the criteria for success, but which came up throughout talks during the development process, which led to the addition of smoothing, as well as the ability to quickly reload images with a single press of a button. Together we also decided we didn't need a no distort check-box.

My client an I were very pleased with the final product, but even during the design phase, it was planned to be easily maintainable and extensible. One of the ways this can be seen, is that hot-keys could easily be implemented with a key listener, and they would only need to call completed methods. Along with this also comes the idea that the product could, with a good bit of work, also be converted to do the inverse of what it currently does, such that a 3D model is converted into an Image. Last but not least, other filters or modifiers could be implemented to manipulate the final model. I implemented one such filter, smoothing, but others could easily be added. Also, tools to edit the image and/or model directly within the program to give the user further control. Later down the road, one could also easily transition away from Java3D to use another 3D preview such as GL11.

The one issue that remained was the requirement that future developers must have Java3D properly installed on their system to compile it. However, the client does not need to install those libraries, as seen in the Windows64 and Linux64 runners. I would also like to explore how I could make the program work with higher resolution images, such that it can run more quickly and maybe support some sort of compression in the final file. I was able to implement features that I believed would be very difficult to implement correctly, like smoothing, but was able to achieve this with help from my teacher. I really believe this project was a success and would like to continue to develop it further.

Word Count: 381