Drake Evans
CS 46X
Fall Term
Group 35 Case Designer

Abstract
The TekBots makerspace at Oregon State University provides access to 3D printers and laser cutter for students, but many potential users are inexperienced with 3D rendering software and don't have the necessary skills to create enclosures for their projects. Online Case Designer is a tool that can generate a 3D model that can fit students' specifications to be used for fabrication. The tool will visualize the output as a 3D model that reacts to the users input. Users will also be able to specify holes in the case to accommodate the needs of their project.Successful completion of the project is a working and simple web application that can design basic cases with holes for whatever ports, cables, etc. Our final product will provide greater levels of freedom than currently available tools to support development of student projects at Oregon State University.

Role:

My primary role in the project shall be graphics programming/display and ui related to the 3d graphics that the client wants. I will also be aiding in other aspects of the project as needed.

Responsibilities:

I will primarily be responsible for the 3d graphics aspects of the projects. This is a key component of the project and must be executed correctly for the project to succeed. The 3d graphics portion of the project has 2 primary things that must be accomplished: choosing the best method of 3d graphics display for the project and proper backend coding methods. Good web interface for the 3d graphics is a secondary task, and one that will likely receive a lot of aid from the web design side of the group.

Methods for Graphics Display:

There are two methods of display that will be examined: WebGL and Sketchfab. WebGL is an application of OpenGL for use on websites. Sketchfab is a platform that allows users to publish, shop for, and share/embed 3d models.

Sketchfab is an online platform that user can register on to publish 3d models you've made as well as find other models that other users have published. The website also allows you to embed the models into webpages, which is why it's of possible interest. The website also allows the purchase of 3d models to be used however the purchaser pleases, as they are royalty free. Use of free content is subject to whatever creative commons license the uploading user applies to it. [1]

WebGL is a library that is used in javascript web programming to create 3d models in webpage's. It is more akin to OpenGL than Sketchfab, whereas Sketchfab is just a sharing platform that has a proprietary model viewer that can be embedded into webpage's. WebGL allows us to code in an interactive model that we can mold to our needs. It only requires us to learn WebGL code as opposed to designing a model in another program. As we only require a simple shape, this will be simple. WebGL is a simple yet powerful tool to have, and it will allow us to easily achieve our goals. It also is supported by all modern browsers and doesn't require any specialized programs from the user [2].

The choice between these two of which to use is simple unfortunately, as Sketchfab does not allow for one of the criteria of the project to be realized, user modification. WebGL also lines up better for accessibility, ease of use, and likely the coding language we will be using for the web interface, javascript. The choice for this project will have to be WebGL.

Coding Methods:

WebGL has plenty of coding tools for displaying 3d graphics, as well as an extensive documentation that will allow us to achieve a very strong code base for this project. As we only need to start with drawing a box, that is an easy task. The library utilizes javascript, so it can be easily integrated into the website. Utilizing various libraries in conjunction with WebGL, an interactive model should be implemented to the specification of the client. One website demonstrates the capabilities of WebGL really well, showing how WebGL can create a very competent interactive 3d display [3]. One of the greatest strengths of WebGL is its compatibility with most modern web browsers without the need for any form of plugins to be installed, allowing for increased accessibility.

Starting out with adding the spot for the model, the "Canvas" element of HTML will allow us to add a spot for our 3D model. Following that, we can enable the webgl context for the canvas element by passing the string, "webgl" to the getContext method, and then the rest of the coding comes from the WebGL tools. The tools follow a similar organization to the standard C++ OpenGL interface, making them easier for me to understand, as I have worked with OpenGL. The procedure starts with setting a lot of stuff up, such as shaders by setting up two sets of shader programs, one for the vertex shader and one for the fragment shader. The shaders don't need to be very complex, given the nature of the task we wish to achieve. Next, we set up the objects we wish to use (the cube) and the viewing options for the scene such as viewing type

(perspective or orthographic, likely orthographic for easier visualization of measurements), viewing angle (where we're looking), depth buffer (how far away from the screen is the center of the scene), colors, viewing window (the size of the canvas element), etc. Then, we draw the objects and animate them if necessary. As the user needs to be able to interact with the model, animation of the scene will come mostly from mouse click and drags on the canvas element of the webpage. The user will need to be able to modify the model, so changing options for the parameters of the object will be accomplished through the use of drop down menus and data entry fields that modify variables in the javascript, such as size and material thickness.

The second page for adding holes will be tricky, and will utilize an almost entirely different set of graphics coding, as the interaction with the model will be different. Users will need to be able to move the cube around at will, as well as be able to visualize the holes they wish to add. Visualization of the holes will be a simple css toggle, but the snap measurements will be slightly tricky. Measurements will need to be kept track of with variables that change with the mouse position, as well as checks against the current hole measurements and coordinates. Then, checks for which measurement the mouse cursor is nearest will happen. Clicking on a location will add a hole to a list of holes with all of the measurements needed for machining.

3D Interactive UI:

There will be a lot of collaboration on this element of the 3D display, in order to achieve the desired look and feel of how the user interface functions. This interface will come mostly out of HTML and CSS coding, likely utilizing drop down menus and entry fields to offer the user options to modify the shape and other facets of the model. The following page will be simple, the same canvas element along with a new set of options. As holes are placed, they will be added to a list of holes under the options on the side of the screen, and will be highlighted if the user clicks on a hole. Users will also be able to remove holes. There's not much to discuss in terms of technical review for these items, as they are mostly discussed in the other reviews for web development solutions.

Bibliography:

[1] Gamedev, D. Cao, J. van Aelst, and Oleaf, “Publish & find 3D models online,” *Sketchfab*. [Online]. Available: https://sketchfab.com/. [Accessed: 04-Nov-2019].

[2] “WebGL: 2D and 3D graphics for the web,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API. [Accessed: 04-Nov-2019].

[3] C. Delmon, “WebGL, the wow effect,” *JoliCode*, 16-Apr-2018. [Online]. Available: https://jolicode.com/blog/webgl-the-wow-effect. [Accessed: 04-Nov-2019].