WEB-BASED MODEL SLICING FOR 3D PRINTERS

BY

MICHAEL U.B. MEDING
B.S., UNIVERSITY OF MASSACHUSETTS LOWELL (2015)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Author .....
April 29, 2016

Certified by .....
Fred G. Martin
Professor
Thesis Supervisor

Certified by .....
Jeff Brown
Associate Professor
Thesis Reader

Accepted by .....
Jie Wang
Department Chair

Web-Based Model Slicing for 3D Printers

by

Michael U.B. Meding

Abstract of a thesis submitted to the faculty of the
Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Master of Science
University of Massachusetts Lowell
2016

Thesis Supervisor: Fred G. Martin
Title: Professor

Thesis Reader: Jeff Brown
Title: Associate Professor

Abstract

In the field of 3D printing, there is currently a large gap between hardware and software. A hobbyist machine can now be purchased for less than $500, but optimal software can be difficult to find, and the only free options available are Cura and Repetier Host. Cura may be run on the Windows, Mac, and Linux operating systems, while Repetier Host is intended only for Windows. This leaves a considerable software gap, which could be filled by a web-based application unbound to a particular system. The purpose of this research was to construct an open source, web-based slicing software to facilitate the use of 3D printers, making 3D printing more approachable for users who have limited experience with technology. Additionally, this application creates opportunities for STEM educators to teach students about 3D printing in a simple and practical way, thereby beginning to bridge the gap between hardware and software.

Acknowledgments

The work in this thesis was only possible because of all the helpful and guiding members of the UMass Lowell Computer Science community. Many people have helped me not just in research effort, but in introducing me to new thoughts and ideas that became foundational to this project.

This research would not have been possible without the helpful guidance of Professor Fred Martin. His passion for education has been a constant source of inspiration. I would like to thank Jeff Brown for serving as my thesis reader. I would especially like to thank Kelsea Gildawie for the late nights spent helping me put my thoughts on paper. It would not have been possible to finish this research without her help.

I also wish to acknowledge my parents Uwe and Sally, who have always encouraged me to pursue my education.

Contents

1 Introduction **1**

1.1 The RepRap Idea 3

1.2 Sudden Growth of 3D Printing 3

1.3 Purpose of Research 3

1.4 Research Objectives 4

1.4.1 CuraEngine Integration 4

1.4.2 Design Intuitive Web Interface 4

1.4.3 Run Beta Testing with Actual Users 4

1.5 Related Work 5

1.6 Thesis Map 5

2 Libraries & Existing Code **6**

2.1 CuraEngine 6

2.2 Client Side Libraries 7

2.2.1 Bootstrap 7

2.2.2 AngularJS 7

2.3 JavaEE 8

2.4 OctoPrint 8

2.5 G-Code Visualizer 9

3 Methodology **10**

3.1 Research Design 10

3.2 Working Procedure 10

3.2.1 Web Interface 10
3.2.2 Slicing Engine 12
3.2.3 Web Tool Path Viewer 13
3.3 Review and Usability Testing 14

4 Client Side **15**
4.1 AngularJS 15
4.1.1 Controllers & Data Binding 15
4.1.2 Factories & Services 16
4.1.3 Directives 16
4.2 WebSlicer AngularJS Structure 16
4.2.1 app.js 16
4.2.2 index.html 17
4.2.3 Settings 18
4.2.4 G-code Visualizer 19
4.3 Key Challenges 20
4.3.1 Visualizer Integration 20
4.3.2 Interpolating Settings 20
4.4 Other Planned Integrations 21
4.4.1 OctoPrint 21
4.4.2 Thingiverse & YouMagine 21
4.5 Issues & Known Bugs 22

5 Server Side **24**
5.1 JavaEE Structure 24
5.2 ProcessBuilder 25
5.3 REST API 26
5.4 Key Challenges 27
5.4.1 ProcessBuilder Deadlock 27
5.4.2 FileTracker Revamp 28
5.5 Future Improvements 29

5.6 Issues & Known Bugs 30

6 Discussion **31**

6.1 Usability Testing 31

6.1.1 Task Description 31

6.1.2 Results 32

6.2 Printer Build 33

6.3 Future of WebSlicer 33

A IRB Compliance Documents **40**

A.1 IRB Exempt Form 40

A.2 IRB Consent Form 47

A.3 WebSlicer Recruitment Email 50

A.4 WebSlicer Survey 52

A.5 WebSlicer Tasks 56

B Server Side Code Appendix **65**

B.1 config/ApplicationConfig.java 65

B.2 config/CORSFilter 65

B.3 cura/CuraEngine.java 66

B.4 cura/CuraEngineOptions.java 69

B.5 cura/StreamDrainer.java 72

B.6 rest/SlicerAPI.java 73

B.7 util/CuraEngineException.java 79

B.8 util/FileTracker.java 79

B.9 util/PlatformExecutable.java 83

C Client Side Code Appendix **85**

C.1 app.js 85

C.2 index.html 88

C.3 visualizer/Worker.js 93

C.4 visualizer/gcodeFactory.js 104

C.5 visualizer/renderFactory.js 110
C.6 visualizer/viewer.js 120
C.7 model/files/filesController.js 122
C.8 model/files/filesDirective.js 123
C.9 model/files/filesTemplate.html 123
C.10 model/octoprint/octoprintController.js 125
C.11 model/octoprint/octoprintDirective.js 127
C.12 model/octoprint/octoprintService.js 127
C.13 model/octoprint/octoprintTemplate.html 130
C.14 model/profile/settingsController.js 131
C.15 model/profile/settingsTabDirective.js 133
C.16 model/profile/settingsDirectiveTemplate.html 134
C.17 model/profile/json/basic.json 134

List of Figures

1-1 (a) High level view of the normal 3D printing process. (b) High level view of proposed new process using WebSlicer. 2

3-1 High level view of how WebSlicer functions and how users will interact with it 11

4-1 Full AngularJS structure breakdown 17

4-2 The flow of data through the application from beginning to end of client side user interaction 23

5-1 The structure of the server side of WebSlicer 25

5-2 Diagram of a deadlock issue that took weeks to resolve 28

6-1 This is what the output from the visualizer looks like to the user after successfully slicing a model. 34

6-2 This is the 3D model that Figure 6-1 is representing in G-code. . . . 35

6-3 RepRap style 3D printer that was built for this research (front view) 36

6-4 RepRap style 3D printer that was built for this research (side view) . 37

List of Tables

5.1 Documentation of all exposed endpoints of the RESTful API 27

6.1 The tasks given to participants of the usability study. The participants were asked to complete the tasks in numerical order. Average easiness is the average of all participants responses where 5 was the easiest and 1 was the hardest. 32

Listings

3.1 A sample of G-code that results from slicing using CuraEngine. . . . 12

4.1 A sample from a static settings file in JSON format. 18

4.2 An example of a ng-repeat looping construct in HTML5. 18

5.1 An example of running CuraEngine C++ executable directly from the command line. 26

5.2 WebSlicer's underlying file structure supported by FileTracker. 29

B.1 A required file in JavaEE for declaring the default application path when deployed. 65

B.2 As CuraEngine allows cross origin responses a filter is requred to allow access by default in WildFly. 65

B.3 CuraEngine.java is the main code that preforms slicing throgh the CuraEngine C++ executable. 66

B.4 All arguments to the CuraEngine C++ executeable must be compiled before it can be executed. This code is the compiler for that argument list. 69

B.5 For ProcessBuilder to run correctly its output stream must be read from. This code spawns a separate thread to read and log this output stream. 72

B.6 The Wildfly application server hosts a public RESTful API which is where requests for slicing and data processing get sent to their corresponding methods. 73

B.7 This exception class gets thrown when any issue arises while CuraEngine is preforming a slice. 79

B.8 FileTracker tracks all of the files uploaded to the server to a series of hash maps. It also issues unique identifiers for each file that gets uploaded so that they may be tracked and accessed simply. 79
B.9 The PlatformExecutable class verifies if the local C++ executable is available and sets up the input and output pipes for CuraEngine if it is. 83
C.1 Main app code where the angular module WebSlicer is defined. This is similar to a main function in C++. 85
C.2 As this is a single page web application this is the only HTML file. It contains all of the libraries needed to run all subsequent sections. . . 88
C.3 Workerjs is the core gcode to D3 line processor. 93
C.4 gcodeFactory.js is the main gcode header parsing code. 104
C.5 When render frames are requested this file compiles the lines of gcode from the worker and sets up the rest of the canvas frames. 110
C.6 This file takes in all requests for parsing gcode and for canvas frames to the main page. 120
C.7 filesController.js handles all requests for uploading files to the server. 122
C.8 This directive links together the template and controller for ease of use when integrating into the index.html page. 123
C.9 HTML snippet which controls the structure of the file upload dialog. 123
C.10 The controller for the octoprint integration which was cut late in development. 125
C.11 This directive bound the controller and the template for easy and multi-use integration into the index page. 127
C.12 This file wrapped the octoprint API so that it could be easily integrated with the controller. 127
C.13 HTML structure which represents the view of the octoprint integration. 130
C.14 This code is the main controller which brings in the static settings files and converts them into settings items. 131
C.15 Each tab in the UI is represented as one of these directives. 133

C.16 This HTML snippet decides what kind of setting needs to be displayed and shows it. 134
C.17 A full static settings file. There were five more files like this but one is sufficient to show the structure of this code. 134

Chapter 1

Introduction

Over the past few years, 3D printing, has become immensely popular due in part to its reduction in cost and its easy availability. An enthusiast can now purchase a do it yourself (DIY) 3D printer kit for less than $500. However, one major pitfall that remains is the software support which leaves much to be desired. Most of the big name companies that used to hold all of the patents for 3D printers have retained the software patents but not the hardware patents. This creates a gap in knowledge between building the printer and running it. This research has served as an exploration of what software existed in the open source for 3D slicing and using that software to build a web based 3D print slicer. This will, effectively "democratize" the world of 3D printing, much in the same way that Google has democratized the way we search the web. In an article written by Harvard Business review, they theorize that this rise in the popularity of 3D printing will spur an industrial revolution as manufacturing becomes more personalized and decentralized (D'Aveni, 2015).

To print something on a 3D printer, there is a multi-step process that can be daunting to many first time users. The first step in the process is to either create or download a model. Creating a model can be achieved with any standard 3D CAD software, such as AutoCAD or SolidWorks. Downloading pre-existing models from an online repository can be done from websites such as Thingiverse or YouMagine. Once a model file is obtained, it is time to "slice" the model. Slicing is the act of taking

this model file and splitting it up into many thin layers that the 3D printer can then convert into motor movements. This process can only be executed by a dedicated slicing software which can often be complicated to use and difficult to install. Once the file has been sliced, the resulting file is a G-code file, which is a set of movement instructions that the printer head must follow. This file is then loaded to an SD card or sent via a print server similar to the way that a normal 2D printer is networked. Once the G-code file has been loaded, all that is left is to hit print either manually using the printer's interface for the SD card or by hitting print on the network interface for the file that was uploaded.

Figure 1-1: (a) High level view of the normal 3D printing process. (b) High level view of proposed new process using WebSlicer.

1.1 The RepRap Idea

The Replication Rapid-Prototyper Project (RepRap) is a movement with the goal of providing open-source, DIY 3D printers at low cost (Jones et al., 2011). RepRap printers are 3D printers with the additional ability to produce most of the parts necessary to assemble another identical printer. This idea also extends outside of hardware but to the software as well. Much of the software available for RepRap style printers are open source projects put together by the community. Unfortunately, community based software development is slow and leads to hardware that outpaces its software.

1.2 Sudden Growth of 3D Printing

In an article written by Forbes, the market trend for 3D printing was analyzed. They determined that 3D printing is becoming one of the fastest growing emerging markets, stating that "The worldwide 3D printing industry is now expected to grow from $3.07 billion in revenue in 2013 to $12.8 billion by 2018, and exceed $21 billion in worldwide revenue by 2020" (Columbus, 2015). 3D printing may be reaching more people as a result of the availability of the RepRap designs. Additionally, building a 3D printer from a kit only requires basic hand tools and a moderate knowledge of electronics, which means that it has been opened up to a much broader audience.

1.3 Purpose of Research

The purpose of this research is to construct a web based slicer and simplify it for users who not well versed in 3D printing, so they are able to slice models and run their 3D printers. This will make 3D printing much more approachable for occasional users who are not prepared to spend thousands of dollars on a professional machine and expensive software. Additionally, this opens up opportunities for educators in STEM programs to teach students about 3D printing in a simple and practical way. Many of the technologies required to complete this project exist in varying states of

completeness. Thus, combining them into a cohesive application is the subject of this study.

1.4 Research Objectives

This research encountered several milestones which had to be met for it to be considered complete. These milestones are listed in no particular order; however, several of them must be completed before allowing other milestones to be completed as detailed below.

1.4.1 CuraEngine Integration

CuraEngine is an open-source slicing engine designed to take model files in .stl file format and convert them into G-code for 3D printing. For this project, wrapping this code and making it callable from the web is the heart of this application.

1.4.2 Design Intuitive Web Interface

This project requires both an intuitive and easy to way to slice a model file for 3D printing. This would be completed using Bootstrap and AngularJS because they are both scalable and flexible for almost any web design. These technologies also allow for small scale or even mobile use.

1.4.3 Run Beta Testing with Actual Users

No research would be complete without testing the software in question. Running a closed beta test of this software is a milestone that must be met as one of the major aspects of this application is its usability. The goals of this test were to see if WebSlicer is both easy to use and useful to those who decide to use it with their 3D printer.

1.5 Related Work

AstroPrint is an all-inclusive cloud operating system for 3D printers. It attempts to encompass the entire package of 3D printing to a dedicated Raspberry Pi, or similar computer system. It is also one of the only software platforms that currently exists that attempts to do web based slicing. Unfortunately, the AstroPrint software tries to accomplish too many tasks at once and has become somewhat like a Swiss army knife, being capable of many functions but only complete a few tasks well. Additionally, their cloud-based slicing software, while being reasonably fast, lacks any support for reviewing the sliced model. This review stage is critical for anyone who is printing something that will take more than a few hours to complete.

1.6 Thesis Map

Chapter 2: Background and design choices. This chapter discusses many of the external resources used and why they are relevant to this research.

Chapter 3: Software architecture review. Compiled here is a complete overview of the architecture of WebSlicer. It follows the user through the entire process of slicing while explaining briefly some of the details of what is really happening.

Chapter 4: Client side in detail. Here is where all of the possible client interactions and application specific details are explained.

Chapter 5: Server side in detail. Details of exactly how a C++ executable is tied into a Java process are explained here as well as the exposed web API created for this research.

Chapter 6: Discussion about usability testing and further improvements. Included in this chapter are the formal results from closed beta testing and a discussion of the printer which inspired this research.

Chapter 2

Libraries & Existing Code

This chapter discusses many of the external resources used and why they are relevant to this research. All of the resources discussed in this chapter are open source projects and libraries.

2.1 CuraEngine

CuraEngine is the server side of a larger application called Cura. Cura is an open source 3D print slicer designed by Ultimaker that is part of the software suite for their Ultimaker line of 3D printers (Kuipers, 2016). As it is the main portion of the server side of WebSlicer, Cura will be discussed extensively in latter portions of this paper.

The reason for choosing to use CuraEngine as the slicing engine is that it has the most clear separation between application and interface. Another option was to use Slicer, which is much older and has better documentation, but is unfortunately written on Windows and would require a large amount of extra effort to get working properly on a linux server. CuraEngine is also platform agnostic, as it is written in C++ and uses one library called protobuf, which is its main interface library for the Cura application.

2.2 Client Side Libraries

When structuring a website for optimal layout and scalability, there are few better options than using the combination of Bootstrap and AngularJS. Bootstrap is a client side CSS library which includes most commonly used CSS options such as buttons and input fields and styles them all accordingly. AngularJS creates a client side environment to support higher level language constructs and features that are normally reserved for complex server side applications.

2.2.1 Bootstrap

Bootstrap is one of the most popular CSS and JS frameworks for developing responsive and mobile projects on the web (Mark Otto, 2016). Being responsive means that Bootstrap is capable of being dynamically displayed on screens of varying sizes. When the screen size changes, Bootstrap is capable of moving and resizing elements on the page without losing any content or overlapping items (Mango, 2013). In addition, it speeds up the process of developing web based applications, as it removes the need to write heavy amounts of CSS to make an application look and preform nicely. Bootstrap also includes many standard features which most web developers use every day, further simplifying the process of building a website.

2.2.2 AngularJS

AngularJS is a framework that allows for easy development of dynamic web pages (Brat Tech LLC, 2016). It contains many ways to logically separate code similar to popular object oriented programming languages like Java or C++. Freeman (2014) states, “the goal of AngularJS is to bring the tools and capabilities that have been available only for server-side development to the web client and, in doing so, make it easier to develop, test, and maintain rich and complex web applications” (p. 48). Therefore, the use of Bootstrap for the unified look, responsiveness, and mobile support combined with the power of AngularJS as a framework, make it the easy choice for a web based application such as WebSlicer.

2.3 JavaEE

JavaEE is a layer built on top of JavaSE the standard Java development environment (Gosling, 2016). This additional layer provides many important architectural interfaces, such as the ability to encapsulate code in EJB containers or web containers for deployment of large applications which may have many of such containers (Pilgrim, 2013). JavaEE also provides frameworks for working with RESTful web services which streamlines the creation of a useful application API.

There are many server side frameworks which have support for RESTful web services, but most lack real scalability like that provided in JavaEE. Furthermore, JavaEE provides the ability to easily distribute computing as more resources are needed. As this research may eventually scale to supporting many users, it was logical to choose a framework which allowed for this kind of growth. The use of JavaEE as a framework also provides the full platform cross compatibility which is standard in Java. Allowing for cross platform compatability would facilitate the process of linking many computers that are running on different operating systems together, further simplifying the prospect of scaling.

To run a JavaEE based application, it must run in a web capable application container. This container must be placed on an application server which exposes it to the web under some context. For WebSlicer, the application server WildFly was chosen for this task as it is well known for its reliability (Fleury, 2016).

2.4 OctoPrint

OctoPrint is a small server that connects a Raspberry Pi to a 3D printer and serves printer info and files to the printer remotely (Häußge, 2016). The Raspberry Pi serves a small webpage which allows the user to easily keep track of the temperature of the printer and the current print progress. This interface allows for a web based connection between your remote g-code files and the local printer (Horvath, 2014).

OctoPrint also contains its own API which makes for easy integration into other web applications.

The fact that OctoPrint is so easily integrated into other web based applications made it a logical choice for integration with WebSlicer. Integrating with OctoPrint would allow for one touch printing by allowing me to send g-code files directly to a connected OctoPrint server. The only interaction that is required of the user is to connect their OctoPrint server which is done by simply indicating the address at which the server can be reached at. Additionally, users are required to include an API key to allow for other applications to directly access the servers API.

2.5 G-Code Visualizer

When building the G-code visualizer, it seemed logical to find one which already existed and worked well. gCodeViewer is an open source project originally written by Nils Hitze (Hitze, 2015). This project was written in pure JavaScript and would be seemingly easy to integrate with WebSlicer.

Unfortunately, many of the libraries that were needed to make this code needed to be updated. Additionally, integrated into the existing AngularJS framework which was more challenging than anticipated. Thus, what ultimately remained from the existing open source project was very small.

Chapter 3

Methodology

This chapter discusses the architecture of WebSlicer. It also serves to justify the design choices that were made.

3.1 Research Design

This thesis is a mixture of both research and design implementation. The research portion of this project focused on linking an existing C++ application (CuraEngine) into a larger JavaEE based project. This research also included running a closed beta test of the software and logging the results.

3.2 Working Procedure

As shown in Figure 3-1, the application has 3 major components that all need to work together in a cycle until the user decides that the output is what they desire.

3.2.1 Web Interface

The web interface includes a set of forms for collecting the user's settings for their printer and the settings as they relate to printing the model itself. This interface also includes a method so the user may retain the files they have uploaded for future use.

Figure 3-1: High level view of how WebSlicer functions and how users will interact with it

This does not include the actual slicing engine, which must be driven and accessed independently.

3.2.2 Slicing Engine

To comprehend the slicing engine it is vital to understand what G-code is. G-code consists of sequential machine instructions that command a 3D printer to move in a sequence of patters to build a three dimentional part (Thornton, 2012). Listing 3.1 shows a sample of G-code that results from slicing using CuraEngine. Comments in G-code are preceded by a semicolon and the first line denotes the type of G-code, we are only focusing on RepRap G-code in this case. Commands in G-code follow a standard pattern of a letter, either M or G, and a number which denotes the type of command. All subsequent items that are separated by spaces are arguments to the command.

Listing 3.1: A sample of G-code that results from slicing using CuraEngine.

```
FLAVOR:RepRap
M190 S110.000000
M104 S245.000000
M109 S245.000000
G21 ;metric values
G90 ;absolute positioning
M82 ;set extruder to absolute mode
M107 ;start with the fan off
G28 X0 Y0 ;move X/Y to min endstops
G28 Z0 ;move Z to min endstops
G1 Z15.0 F9000 ;move the platform down 15mm
G92 E0 ;zero the extruded length
G1 F200 E3 ;extrude 3mm of feed stock
G92 E0 ;zero the extruded length again
G1 F9000
```

```
;Put printing message on LCD screen
M117 Printing...
;Generated with Cura_SteamEngine master
;LAYER_COUNT:158
;LAYER:0
M107
G0 F7200.000000 X99.430 Y80.760 Z0.300
G0 X102.806 Y78.243
;TYPE:SKIRT
G1 F1800.000 X103.337 Y78.033 E0.01074
G1 X103.441 Y77.996 E0.01282
G1 X103.842 Y77.875 E0.02070
G1 X104.142 Y77.800 E0.02651
```

The slicing engine rests at the core of this project as most of the computation time is spent preforming geometry calculations and converting them to G-code. This process of converting from model to G-code is known as slicing. An application which is encapsulated and preforms slicing is known as as a slicing engine. The engine which carries out the raw geometry calculations for WebSlicer is CuraEngine which is written in C++. Thus, this portion of the project required deploying the CuraEngine application on a remote server and creating a RESTful API to interface with it.

3.2.3 Web Tool Path Viewer

After configuring and generating the G-code representation for a 3D model, there must be a way to visually review how the slicing engine will divide the model. This is executed by loading the resulting G-code from the slicing engine into a web based visualizer. The user is then able to view each of the layers and the steps involved in creating each one in detail. As displayed in Figure 3-1, this process of changing settings, slicing, and reviewing can occur an arbitrary number of times before the user

decides they are satisfied with the result.

3.3 Review and Usability Testing

After preforming all steps of the working procedure, it was then necessary to test WebSlicer by running a small beta test. This beta test consisted of a small group of users with varying familiarity with 3D printing to test how simple WebSlicer is to use through a series of simple tasks. These simple tasks ranged from uploading a file to finding and adjusting the correct setting for a printer.

Chapter 4

Client Side

This chapter discusses the client side of WebSlicer in detail. It also includes some necessary background about AngularJS as it will be crucial to understanding latter sections of this chapter.

4.1 AngularJS

In order to understand the structure of WebSlicer a few things must be known about how AngularJS applications are structured and how AngularJS itself works.

4.1.1 Controllers & Data Binding

The controller structure in AngularJS that sits between the view and the JavaScript world acts like the glue between the two. From a controller the user is allowed access to "scope" variables, which are similar to normal JavaScript variables, but they have a special two-way data binding property. Two-way data binding is one of the most interesting features of AngularJS, as it gives the user real time updates as the user interacts with a view. In traditional JavaScript, a developer must identify key events that occur in the browser as the trigger for a sequence of events. AngularJS also has a sequence of events but no trigger is required as the sequence of events is triggered automatically. This action is known as a digest cycle. AngularJS has a watch function

for each variable attached to its scope. When any changes to this variable occur, they propagate that change to the functions that are associated with that variable (Freeman, 2014).

4.1.2 Factories & Services

JavaScript has many pitfalls, including the lack of any kind of larger design pattern support, such as Object Oriented Programming (OOP). The current trend with web based applications is to create single page applications with the same functionality as prior designs that have many pages. AngularJS is the answer to this with Factories and Services which mimic the design of a POJO (Plain Old Java Object) and Singleton objects in Java.

4.1.3 Directives

A directive is one of the most powerful structures in AngularJS. It allows the programmer the power to write their own HTML5 tag with its own parameters and rules. Directives also contain support for templates which are HTML snippets that replace the main tag when the code is loaded. Combining this with the use of the aforementioned design patterns creates a very powerful tool to create and organize dynamic content.

4.2 WebSlicer AngularJS Structure

This section discusses the entire client side structure of WebSlicer when it is broken down into its respective Controllers, Factories, and Directives.

4.2.1 app.js

The full graphical AngularJS structure of WebSlicer is shown in Figure 4-1. Flow through this diagram starts with the app.js node which represents the main controller of the application. This can be thought of as a main function in C++. The main

Figure 4-1: Full AngularJS structure breakdown

control variables are also inside of app.js which are similar to global variables. Variables in this controller are used to store the current settings, file pointer, and output G-code.

4.2.2 index.html

Figure 4-1 also shows that index.html is large hub and, as WebSlicer is a single page web application, this is the only static HTML file. Index.html has several other functions, such as bringing in all libraries and custom directives.

4.2.3 Settings

As shown in Figure 4-2, settings have a long path that they must travel before they are submitted to be used while slicing. This data flow starts with loading a static JavaScript Object Notation (JSON) file, which describes the settings in a pattern as shown in Listing 4.1.

Listing 4.1: A sample from a static settings file in JSON format.

```
1 {
2     "setting": "layer_height",
3     "default": 0.1,
4     "type": "float",
5     "category": "Quality",
6     "label": "Layer Height (mm)",
7     "description": "Layer height in millimeters. This is
          the most important setting to determine the
         quality of your print. Normal quality prints are
         0.1mm, high quality is 0.06mm. You can go up to 0
         .25mm."
8 }
```

Listing 4.2: An example of a ng-repeat looping construct in HTML5.

```
1 <table class="table">
2   <tr><td>Action</td><td>Done</td></tr>
3   <tr ng-repeat="item in todos">
4     <td>{{item.action}}</td>
5     <td>{{item.done}}</td>
6   </tr>
7 </table>
```

A directive called curasettings takes this static JSON file and divides it so an input field exists in the template for each setting object. AngularJS provides this functionality through the use of an ng-repeat, which is written in a similar fashion to

that of a for-loop in Python which is shown in Listing 4.2. Using a template for a directive in this case also means that the user can have control over many different fields, such as drop-downs and number-specific inputs. The curasettings directive also gave the logical separation of one static JSON file per tab of settings in the UI, which made it simple to find and modify the settings as needed.

Settings in WebSlicer have a complex method of being tracked after loading to the client. In most applications such as this, there would be an individual watch on each field or a submit button which would trigger grabbing all the fields. WebSlicer, however, uses a single object to track all of the settings for the application and uses a dynamic method of AngularJS to map from the input field to a field of a single object. Thus, when the settings are submitted, there is no more interpolation needed, as the settings object already has the current state as shown in the Figure 4-2 as "Pull Current Settings."

4.2.4 G-code Visualizer

The G-code visualizer is a layer-by-layer G-code analyzer to aid in model slicing. It is written using a combination of D3 and a JavaScript web worker. From Figure 4-1, it can be seen that the controller for the visualizer is sent in a G-code file, which it splits up to two separate services. The controller completes an initial parse of the file, which places each one of the lines into its own entry in an array before exchanging it with gCodeFactory and Worker. Both of these files parse through this entire array, but do so at roughly the same time to expedite the process of visualizing. The worker takes the array of lines and ignores the header to just focus on converting the raw movement commands into D3 lines so that they may be rendered. The gCodeFactory takes the headers from the array and uses them to do analytics of the G-code file, such as total print time.

The final steps in the process of visualizing the G-code require splitting the G-code file into layers for rendering. This task which is made simple by following the given

tags in the G-code which signal a layer change. Once all of the cumbersome tasks of parsing and splitting up the G-code file are finished, it is simply a matter of returning layer requests with canvas frames. Each time a layer is requested, a progress for that layer is also sent. The controller then indexes to the layer height in the array and renders a frame with the number of lines that are described by the current progress. As array indexing is nearly instantaneous, the visualizer once parsed and loaded runs very quickly to display layers.

4.3 Key Challenges

Discussed in this section were the most notable challenges that had to be overcome when developing the client side of WebSlicer.

4.3.1 Visualizer Integration

The visualizer starter code for this was originally written by Nils Hitze as an open source project which had many other features (Hitze, 2015). This code, however, required a lot of work to integrate properly with the rest of the application. AngularJS, despite its many features does not mix well with other projects. Ultimately, the only code that remained from the original was the JavaScript web worker and some of the parser code.

4.3.2 Interpolating Settings

A method was required to be designed for the application to handle a lot of input fields. A simpler method would have been to create a series of fields, each with their own variable, and submit methods and triggers; however, if settings ever change format it would require refactoring many files. Thus, spending the extra time to design an intelligent method of handing large amounts of input data seemed logical. This, however, took much more time and effort than anticipated. At one point, this even required contacting the original developers of CuraEngine discuss the meaning of

some of their settings. Documentation for many of the settings was dissatisfactory and in many instances was non existent which further slowed down development.

4.4 Other Planned Integrations

During the development process often many items do not make it into the final product for various reasons. Included in this section are some items which were cut from WebSlicer before beta testing.

4.4.1 OctoPrint

A feature that was removed at a late state in the process of building this application was an integration with OctoPrint. OctoPrint has a exceptional API that allows for external applications to integrate easily, making it an ideal choice for this application. This integration was to allow a user who was running an OctoPrint server to be able to send files directly to their server. This would eliminate the need to download the G-code from WebSlicer only to be uploaded to the print server seconds later. It was decided at the last moment that this feature need not be in the minimum viable product and that time was best allocated to finishing more crucial features of the application.

4.4.2 Thingiverse & YouMagine

Another planned integration was the ability to import from a web based repository such as Thingiverse or YouMagine. These repositories are public sites where users can upload their 3D designs so others can 3D print them. Thingiverse in particular has a effective API for accessing models from their site, which would make it an easy integration for a web based slicing software; however, this feature was terminated early on, as it would have required too much unnecessary development time to finish.

4.5 Issues & Known Bugs

As mentioned in prior sections, WebSlicer was designed with a minimum viable product in mind. Developing a working 3D print slicer for the web was the primary task and all other features needed to support this or extend this functionality. For this reason, there is no login or user database, which would normally be the first item to be developed for an application such as this. There is also no way to view any of the models in 3D which, as discussed in latter sections of this thesis, makes the software harder to use.

Figure 4-2: The flow of data through the application from beginning to end of client side user interaction

Chapter 5

Server Side

This chapter discusses the server side of WebSlicer in detail. Included with this is some necessary background about JavaEE and some of its functions.

5.1 JavaEE Structure

The server side of WebSlicer was written in JavaEE, the structure for which is shown in Figure 5-1. JavaEE was the optimal choice for this application, as it allowed for the easiest deployment and was also the easiest to scale (Pilgrim, 2013). Additionally, JavaEE has a excellent code packaging mechanism for web and non web based applications alike. The web container which is in use for this application exposes a RESTful API on a privately hosted server.

To further simplify the development process, Maven was also used. Maven is a build tool for Java and has support for deploying complex applications such as those in JavaEE (Vincent Massol, 2005). Thus, when a build was completed, it was automatically deployed and ready for testing.

Figure 5-1: The structure of the server side of WebSlicer

5.2 ProcessBuilder

At the core of the server side application is an executable called CuraEngine. It is the main executable which is compiled from the open source slicing platform Cura which is written in C++. This presented a problem, as all of the server side code in this application is written in Java. ProcessBuilder was the solution to this problem as it is capable of redirecting the input and output streams of a local executable process into the Java server application. CuraEngine from Figure 5-1 uses a ProcessBuilder and the PlatformExecutable to create a runnable Java method that is capable of executing like a C++ executable. CuraOptions feeds the CuraEngine class with all of the parameters that it needs from the API. It gathers the path to the appropriate settings file and includes all of the parameters needed to run the CuraEngine executable.

Listing 5.1: An example of running CuraEngine C++ executable directly from the command line.

```
1 CuraEngine slice -v -j {settings.json} -g -e -o {output.
    gcode} -l {model-file.stl}
```

An example of running the CuraEngine C++ executable from the command line is shown in Listing 5.1. When the ProcessBuilder class of WebSlicer receives a slice command from the API it gathers the arguments listed in brackets and sends them to PlatformExecutable. PlatformExecutable then spawns a native process and pipes its input and output streams into the respective Java streams. In the meantime, StreamDrainer spawns a new thread and waits for the output stream that was created by PlatformExecutable. StreamDrainer's task is to take the unneeded output from stdout and pipe it into a log file for debugging.

After CuraEngine has finished slicing the current file and PlatformExecutable has returned the REST API, which has been waiting, it unblocks and starts reading the output gcode file. This file is then packaged and sent back to the client as the response of the "/slice/{clientId}/{modelId}" command as shown in Table 5.1.

5.3 REST API

The goal of the client is to satisfy all the requirements needed to run the "/slice" command from Table 5.1. A typical call pattern to the API in begins with creating a user if one does not already exist. Creating a new user requires calling "/setupClient" and recording the returned unique user identifier. Using the recorded unique identifier the client must then upload a model file and settings file to the corresponding "/importStl" and "/importSettings" functions. This returns unique identifier strings for all settings and model files that the client uploads to the server. Finally, to run a slice, the client takes all acquired unique identifiers and places them into the call to "/slice" as shown in Table 5.1. Upon successful completion of all aforementioned calls the user will be returned a G-code file in raw textual form.

Type	Address	Description
GET	/ping	A simple ping endpoint used for testing.
POST	/setupClient	Sets aside all needed files for a new client and return its unique ID.
POST	/importStl/{clientId}	Takes a MIME type file stream and imports the file to the clientId specified in the URL. It also returns a unique identifier for the file.
POST	/importSettings/{clientId}	Similar to importStl this endpoint takes a settings JSON file and imports it to the specified clientId
POST	/slice/{clientId}/{modelId}	This is the main slice function of the API. It combines all of the parameters specified by the calls before and returns a gcode file to the user.
POST	/testSlice	A test endpoint that requires no parameters and simply returns some arbitrary gcode to the user.
GET	/getFiles/{clientId}	Returns all the model file names and their tracking ID's that are associated with a clientId.

Table 5.1: Documentation of all exposed endpoints of the RESTful API

5.4 Key Challenges

5.4.1 ProcessBuilder Deadlock

One considerable bug encountered while developing this project was a thread deadlock issue. The server side code uses Java's ProcessBuilder, which builds a system native call to an executable and then pipes the input and output into the corresponding pipes of Java's stdio as shown in Figure 5-2. This is suitable for small platform executables with limited I/O, but can become problematic when complex native calls such as CuraEngine are used.

ProcessBuilder executes its normal writes to stdout and the drainer pipes them into a file; however, the drainer must wait for a file pointer using the fp.available() function. This is a non-blocking function that only estimates the buffer size that it has for the file. The check for file pointer availability was determining whether

Figure 5-2: Diagram of a deadlock issue that took weeks to resolve

this function returned something greater than 0 as an estimate before notifying the ProcessBuilder that it was ready; however, the buffer size would often start as zero before allocation and, as this check was not part of a loop, it would stay stuck forever as the notify was missed.

This problem was solved by using the correct blocking file pointer available check. Occasionally, the buffer size was larger than 0 and the application ran suitably but, with some models, it would consistently fail, as the buffer had not been allocated yet. This solution is seemingly obvious yet it took many days to find and correct because the application did not fail consistently.

5.4.2 FileTracker Revamp

The first iteration of FileTracker was crude and not well planned out. It tracked two hash maps: one for model files and the other for settings files with no mind for the client who required access to those files. This worked for testing, but incurred

many pitfalls including the inability to reuse files that already existed. As soon as the client closed their session, those files were lost, which is a major inefficiency.

The fdmprinter.json file within the unique client folder is symbolically linked to the fdmprinter.json file within the common folder. The CuraEngine executable requires that all of the settings files rest within the same directory when performing a slice as shown in Listing 5.2. Unfortunately, this leaves the potential result of having this file copied for many clients. Thus, symbolically linking the file to the rescue.

The output.gcode and settings.json files are dynamically overwritten for every iteration so their existence here is only for the sake of running CuraEngine through file arguments. The user has no access to these files and is only able to obtain their content through the web interface, which parses in and out of files.

Listing 5.2: WebSlicer's underlying file structure supported by FileTracker.

```
webslicer/
- b1a2a69e-5893-4d7c-aa1f-d639fa3b4ed1/
  - fdmprinter.json -> /tmp/webslicer/common/fdmprinter.
     json
  - models/
     - balanced_die_version_2.stl
     - raldrich_planetary.stl
  - output.gcode
  - settings.json
- common/
  - fdmprinter.json
  - presets/
    - prusa_i3.json
    - ultimaker2.json
```

5.5 Future Improvements

Currently, FileTracker does not take advantage of the presets within the "common/presets/" folder as described by Listing 5.2. These files contain the default settings for the corresponding printer, which are the ultimaker2 and a basic configuration of a prusa i3 variant. Optimally, the user would select from one of these starting presets

and then modify and save their own. This would allow users an optimal starting point while lowering the amount of starting knowledge and increasing the usability of WebSlicer.

This new file structure also allowed for an easy client index. In the future, the unique folder ID will become the client's identification number, which will be attached to their login. Additionally, simplifying the login process with Google's OAuth 2.0 system was also planned.

5.6 Issues & Known Bugs

Currently there is no way for the server to import existing user files into its structure. Thus, when the server is restarted for any reason, the supporting file structure with all user files is lost. Resolving this is just a matter of writing an initial import function that indexes all of the existing files. It was removed from the initial version due to time constraints.

Chapter 6

Discussion

This chapter summarizes the experience of running usability testing of WebSlicer and the construction of a RepRap 3D printer that was used during software trials.

6.1 Usability Testing

Good design goes through many design iterations before it is considered production ready. Between iterations feedback is given through usability testing. For this research the time line was short thus, only one cycle of usability testing was completed.

6.1.1 Task Description

WebSlicer was evaluated by giving six research participants a series of tasks to preform in a controlled environment. This environment consisted of a lab computer with a small collection of model files and a web browser. The tasks that were given to participants are formalized in Table 6.1. Participants were also asked to fill out a questionnaire when they had completed the study. This questionnaire asked participants to rate the level of difficulty of each task from 1 to 5 and to include comments of what they thought of each task. Results of the average difficulty reported by users is shown in Table 6.1. As an incentive to participate, participants were given a small 3D printed figure of their choosing.

Task #	Task Name	Average Easiness	Description
1	Create a new user	4.66 out of 5	Participants were simply asked to click a button which generated a unique ID for them.
2	Upload your file	5 out of 5	Uploading a model file of the participants choosing from a given USB drive.
3	Adjust slicer settings	4.33 out of 5	Participants were presented with a table of settings and their value. They were instructed to find these settings and change them while ignoring all other settings.
4	Analyze and download G-code	4.66 out of 5	This task simply required the user to toggle the visualizer to see the output of their actions in prior tasks.

Table 6.1: The tasks given to participants of the usability study. The participants were asked to complete the tasks in numerical order. Average easiness is the average of all participants responses where 5 was the easiest and 1 was the hardest.

6.1.2 Results

The expected result of usability testing was to expose some minor flaws of the user interface (UI) design; however, the results showed that more than minor changes would be required before the next iteration of WebSlicer. G-code output in textual form exposed to the user was found to be a bug in certain cases during usability testing. The output when slicing certain models was so large that the browser crashed when trying to load the text to the screen. This bug could be resolved by filtering the amount of text displayed at one time to the user.

Another flaw that was exposed during testing was the functionality of the G-code visualizer. Most participants found that both the controls, and explanation of what it was meant to accomplish was lacking as shown in Figure 6-1. One proposed solution would be to replace the numerical input fields with sliders which gives more natural control. Concluded from this is that more textual explanations and user guidance was required in further iterations of WebSlicer.

6.2 Printer Build

For this research a 3D printer was required to test the output of the slicer. Instead of purchasing an expensive pre-built machine, a RepRap style 3D printer was constructed using commonly available materials as shown in Figures 6-3 and 6-4. Building the printer took approximately 3 months and incurred many issues along the way such as faulty electronics and bad wiring. During the research when testing parameters were required for the slicer those from this printer were used. Additionally, all the incentive parts that were given to research participants for their service during usability testing were printed on this machine.

The design of the 3D printer is that of a Cartesian RepRap Prusia i3 variant. The frame is built of extruded aluminum bar and all complex mechanical parts were 3D printed on another machine. The electronics consisted of an old computer power supply, Raspberry Pi, Arduino Mega, and a salvaged PID controller in addition to a standard set of heaters and motors. Octoprint was installed on the Raspberry Pi and was used during the Octoprint integration with the client side of WebSlicer.

6.3 Future of WebSlicer

The goal of future versions of WebSlicer are to lay the groundwork for a larger manufacturing software tool. This manufacturing software will link together many 3D printers and allow for unified control from a web based hub. Unified control such

Figure 6-1: This is what the output from the visualizer looks like to the user after successfully slicing a model.

as this would allow for users to preform small scale manufacturing which would fill a void that currently exists in the manufacture of plastic parts.

Figure 6-2: This is the 3D model that Figure 6-1 is representing in G-code.

Figure 6-3: RepRap style 3D printer that was built for this research (front view)

Figure 6-4: RepRap style 3D printer that was built for this research (side view)

References

Brat Tech LLC, G. e. a. (2016). *AngularJS*. Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043. Retrieved from https://angularjs.org/.

Columbus, L. (2015). 2015 roundup of 3d printing market forecasts and estimates. *Forbes*.

D'Aveni, R. (2015). The 3-d printing revolution. *Harvard Business Review*.

Fleury, M. (2016). *Wildfly Application Server*. Red Hat. Retrieved from www.wildfly.org.

Freeman, A. (2014). *Pro AngularJS*. Apress.

Gosling, J. (2016). *Java Enterprise Edition*. Sun Microsystems, Oracle. Retrieved from http://www.oracle.com/technetwork/java/javaee/overview/index.html.

Hitze, N. (2015). gcodeviewer. https://github.com/hudbrog/gCodeViewer.

Horvath, J. (2014). *Mastering 3D Printing*. Apress.

Häußge, G. (2016). *OctoPrint*. Impressum, Goethestraße 5 63179 Obertshausen Deutschland. Retrieved from https://www.octoprint.org.

Jones, R., Haufe, P., Sells, E., Iravani, P., Olliver, V., Palmer, C., and Bowyer, A. (2011). Reprap – the replicating rapid prototyper. *Robotica*, 29:177–191.

Kuipers, T. (2016). *CuraEngine*. Ultimaker. Retrieved from https://github.com/Ultimaker/CuraEngine.

Mango, A. (2013). *Mobile First Bootstrap*. Packt Publishing.

Mark Otto, J. T. (2016). *Bootstrap*. Twitter. Retrieved from http://www.getbootstrap.com.

Pilgrim, P. A. (2013). *Java EE 7 Developer Handbook*. Packt Publishing.

Thornton, J. (2012). Linuxcnc g-code. http://gnipsel.com/linuxcnc/g-code/gen01.html.

Vincent Massol, T. O. (2005). *Maven: A Developer's Notebook*. O'Reilly Media.

Appendix A

IRB Compliance Documents

Included in this appendix are all the IRB compliance documents that were requited for preforming usability testing.

A.1 IRB Exempt Form

The following pages are the IRB Exempt Form which contains the offical summery of the research to be undertaken.

IRB EXEMPT STATUS APPLICATION

IRB No.: 16-061-MAR-EXM Rev. No./Date: 2/4-11-16

Submit all documents to IRB@uml.edu Date Submitted to IRB: 3/29/16

(This form is ONLY for minimal risk research where no identifiers are collected.)

A. GENERAL INFORMATION

Project Title: WebSlicer Usability Testing	
PI: Fred Martin	Email:fredm@cs.uml.edu
Department: Computer Science	Work Address (Bldg and No.): Olsen Hall 306
Phone: 978-934-1964	Emergency Phone: 978-934-2705
Co-PI(s):	Co-PI(s) Contact Info:
Student Researcher: Michael Meding	Student Researcher Contact info: 214-334-1905

1. Sponsor Information- Check One (Double left click on each check box to access tool.)
(x) Not funded.
( ) Internal funding. Type:
( ) Government/Federal funding. List agency name:
( ) Subcontract. List organization name and include contact name, telephone no., and address:
( ) Other. List organization/company name and include contact name, telephone no., and address:

The proposal to the funding source noted has also been submitted to the IRB: ( )Yes ( )No

2. Project Personnel: Include the PI and all personnel who may interact with subjects or access identifiable human subject data. Training certification should be submitted with the application.

Name and Title(Check one)	Email Address	Training Completed
Fred Martin (x)Faculty ( )Staff ( )Student	fredm@cs.uml.edu	(x) CITI Basic, Date: 06/27/2014 ( ) NIH, Date:
Michael Meding ( )Faculty ( )Staff (x)Student	mike@mikemeding.com	(x) CITI Basic, Date: 03/01/2016 ( ) NIH, Date:
( )Faculty ( )Staff ( )Student		( ) CITI Basic, Date: ( ) NIH, Date:
( )Faculty ( )Staff ( )Student		( ) CITI Basic, Date: ( ) NIH, Date:
( )Faculty ( )Staff ( )Student		( ) CITI Basic, Date: ( ) NIH, Date:
( )Faculty ( )Staff ( )Student		( ) CITI Basic, Date: ( ) NIH, Date:

41e16d377dc500ffbda8e27ba2e36b55

( )Faculty ( )Staff ( )Student		( ) CITI Basic, Date: ( ) NIH, Date:

Additional personnel or other information:

a. Is this a student research project? (x)Yes ( ) No
If yes, (x) Graduate or ( ) Undergraduate, please specify below:
( ) Dissertation (x) Thesis ( ) Directed Study ( ) Class Project ( ) Other:

B. SCREENING QUESTIONS

1. Conflict of Interest Disclosure:

a. Do you or any family members have a financial interest in this research activity (such as an equity position or outside consulting arrangement with the company whose drug, procedure, device or product is used or tested in this study)? ( )Yes (x) No
If **yes**, explain the nature of the relationship and the conflict(s):

b. Do other faculty or staff involved with this research have a financial interest in this research activity?
( )Yes (x) No
If **yes**, indicate the nature of the relationship and the conflict(s):

c. To your knowledge, does the University have a financial interest in the company whose drug, procedure, device or product is used or tested in this study (such as patent rights, equity)? ( )Yes (x)No
If **yes**, indicate the nature of the relationship and the conflict(s):

2. Will the research expose participants to discomfort or distress beyond that normally encountered in daily life?
( )Yes (x)No

3. Could disclosure of participants' responses outside the research reasonably place them at risk of criminal or civil liability or be damaging to their financial standing, employability, or reputation? ( ) Yes (x)No

4. Does any part of the research require deception or incomplete disclosure of information to participants?
( )Yes (x)No

5. Will prisoners (or their data and/or specimens) be participants in the research? ( )Yes (x)No

6. For research proposed under categories 1-5, is the research subject to FDA regulations? ( )Yes (x5)No

Note: a **YES** for **questions 2-6** above indicates your research does NOT meet exempt criteria. Submit an Application for Expedited or Full Review.

C. EXEMPT CATEGORY CLAIMED (check all that apply):

( ) 1. Research conducted in established or commonly accepted educational settings, involving normal educational practices, such as research on regular and special education instructional strategies, or research

on the effectiveness of or the comparison among instructional techniques, curricula, or classroom management methods.

If you checked this category, to do the research under this exempt category, it must be conducted in commonly accepted educational settings and not deviate from normal educational practices.
Is this true? ( )Yes ( )No If no, please submit an application for expedited or full review.

(X) 2. Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior, **unless**: Information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

If you checked this category, the activity may **not** involve any interactions of the researcher with children, if they are participants. Does it involve interactions with children? ( )Yes (X)No
If yes, submit an application for expedited or full review.

( ) 3. Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior that is not exempt under paragraph (2) of this section, if: the human subjects are elected or appointed public officials, or candidates for public office; or federal statute(s) require(s) without exception that the confidentiality of the personally identifiable information will be maintained throughout the research and thereafter.

( ) **4.** Research involving the collection or study of existing data, documents, records, pathological specimens, or diagnostic specimens, if these sources are publicly available or if the information is recorded by the investigator in such a manner that subjects cannot be identified, directly or through identifiers linked to the subjects.

For research under this category, will any of the data, documents, records, or biological specimens be collected or created **after** the date of this application? ( )Yes ( )No
If yes, submit an application for expedited or full review.

For research under this category, will any of the information obtained from private sources of data, documents, records, or biological specimens be recorded by the investigator in such a manner that subjects could be identified directly or through identifiers linked to the subjects? ( )Yes ( )No
If yes, submit an application for expedited or full review.

( ) **5**. Research and demonstration projects conducted by or subject to the approval of department or agency heads, and which are designed to study, evaluate, or otherwise examine: public benefit or service programs; procedures for obtaining benefits or services under those programs; possible changes in or alternatives to those programs or procedures; or possible changes in methods or levels of payment for benefits or services under those programs.

( ) **6**. Taste and food quality evaluation and consumer acceptance studies, if wholesome foods without additives are consumed or if a food is consumed that contains a food ingredient at or below the level and for a use found to be safe, or agricultural chemical or environmental contaminant at or below the level found to be safe, by the Food and Drug Administration or approved by the Environmental Protection Agency or the Food Safety and Inspection Service of the U.S. Department of Agriculture.

D. **RESEARCH ACTIVITIES-Check all that apply:**

( ) Internet or email data collection
( ) Existing data, publicly available
( ) Existing data, NOT publicly available
( ) Focus groups
( ) Audio recordings
( ) Other:

(x) Observation of participants
( ) Record review
( ) Research using existing specimens
(x) Surveys or questionnaires
( ) Interviews

41e16d377dc500ffbda8e27ba2e36b55

E. **RESEARCH SUMMARY**

1. Describe the research purpose and objectives: In this research study, we are evaluating a web-based 3D print slicer. Participants will choose a small model or figure from a set of pre selected models and follow the steps provided in the "Task Instructions" document to take that model from design to printable file. The objective of this study is to gain insight into the usability of this software so that it may be evaluated and improved as part of the thesis study.

2. Describe the research methods: Participants will be recruited via email. Upon response from recruitment email participants will choose a time to join us in our lab Olsen 306, From there, participants will be asked to review the consent form (which will be provided as a web link) and then indicate that they have read the form and consent to the study in an online survey form. If they indicate they have not read the consent form, or decline consent, the session will end there. Sessons will last between 20 to 30 minutes depending on the amount of prior knowledge that the participant has of 3D printing and its surrounding technologies. Prior knowledge is not required to participate in this study. Upon consent, participants will be asked to complete a pre-activity survey on a computer in our lab Olsen 306. Then participants will be provided a printed version of the Task Instructions, and asked to use the web-based software to perform the set of tasks described therein. While each participant is completing the tasks, the researcher will record brief observations on a observations form (attached). After the tasks are complete, participants will be asked to take at post-study survey (which is provided as a continuation of the pre-survey). During this time, they can watch the 3D printer output a model. At the conclusion of the study, participants will receive a previously-printed model as a token of appreciation for their participation. The researcher will match surveys and observation notes by numbering the first participant "1", the second "2", and so on, on the observation form. No personally identifiable data will be collected during this study.

3. Describe the participant population: Students and faculty at UMass Lowell whom have little to no knowledge of 3D printing or its surrounding technologies.

4. Recruitment Information

a. Describe how the participants will be recruited:
Students will be recruited via email from the Computer Science email message board. The recruitment email will be sent from the PI's email and responses will be sent to the student researcher's email as included in the instructions in the email.

b. Indicate the anticipated number of participants: 6 to 8

c. Will any participants be under 18 years of age? ( )Yes (x)No

If yes, justify and describe how you will meet the exemption requirements:

5. Estimate the duration of the study: 1 Month

6. Will all of the research activities be conducted at UMass Lowell? (x)Yes ( )No

a. If no, list the site/collaborator(s):

b. A letter(s) has been submitted to the IRB from the collaborator to document how they intend to support the research. ( ) Yes ( )No

7. Describe how participants will provide consent: Participants will be asked to read a consent form and then asked to click yes or no on the survey page.

41e16d377dc500ffbda8e27ba2e36b55

8. Does research involve the use of publicly available or currently existing data? ( )Yes (x)No

a. If yes, list source of the data or specimens:

b. Indicate whether the data is currently de-identified or how it will be de-identified:

9. Describe any potential risk to participants from participating in the research: There is no more than minimal risk to participants since no identifying information will be collected. Participants who experience discomfort from using a computer will be advised in the consent form to refrain from particpating.

10. Indicate how you intend to minimize any risks to participants: Participants will be informed that they may leave the study at any time should they feel uncomfortable and no identifying information will be collected.

11. Describe procedures to protect participants' privacy and confidentiality: Personally identifiable data will not be collected.

12. Describe the potential benefits from the research: Design improvements of the software being researched.

13. Check all of the supporting materials submitted with this application:

(x) Questionnaires, surveys

( ) Standard Research Tools (published testing materials, etc.)

(x) Recruitment Materials

(x) Consent Documents

(x) Other, list: Observation rubric to be used by researcher while participant completes tasks; Task Instructions document.

F. PRINCIPAL INVESTIGATOR ASSURANCE AND SIGNATURE

(X) I understand that, as the PI, I am ultimately responsibility for the protection of the rights and welfare of human participants and the ethical conduct of research under this protocol. I agree to conduct the study in accordance with the approved protocol and ensure that all personnel involved in the research will do the same.

(X) I agree to follow the UMass Lowell IRB Policies and Procedures.

(X) I certify that the information provided in this application is complete and correct, and believe that my project qualifies as Exempt from the Federal Regulations.

(X) I agree to personally conduct or supervise the described investigation(s).

(X) I agree to maintain copies of all questionnaires, survey instruments, interview questions, data collection instruments, and information sheets for human participants for three years following termination of the project,

(X) I understand all investigators associated with this research must renew their human participant research training every 3 years.

(X) I understand it is my responsibility to resubmit an application to the IRB if I need to make any changes that alter the exempt status determination and approval.

(X) I understand this project will be closed by the IRB one year from the date of approval and records will be retained in the IRB office for 3 years after that date.

Project Title: WebSlicer Usability Testing

The signature page only may be submitted as a scanned document, faxed to x6012, or sent by intercampus mail to the IRB Administrator* at *Wannalancit, 2nd Floor. The entire application should be emailed as a word document to IRB@uml.edu

SIGNATURE: **Note: Students are not eligible to sign this page.**

PI Signature: /s/ Fred G. Martin	Date:3/29/16
Printed Name of PI: Fred G. Martin	

OR (X) Check here if submitted electronically from the PI's email account.

41e16d377dc500ffbda8e27ba2e36b55

A.2 IRB Consent Form

The document that follows is the consent form which was distributed to participants before beginning the usability study.

WEBSLICER USABILITY TESTING
INFORMED CONSENT

INTRODUCTION

You are invited to join a research study under the direction of Fred Martin and conducted by Michael Meding, “Webslicer Usability Testing.” This study will involve the process of bringing a 3D model from digital design to physical world using our new slicing software. Please take whatever time you need to discuss the study with your family and friends, or anyone else you wish to. The decision to join, or not to join, is up to you.

In testing WebSlicer, we will be evaluating a web 3D print slicer. A print slicer is a piece of software which takes 3D models and divides them into virtual layers in preparation for being 3D printed. As a participant your role will be to choose a small model or figure from a set of pre-selected models. From there the participant will take this model through all of the steps necessary to prep it for being printed on a supplied 3D printer.

From this focused study I hope to gain insight into some of the strengths and weaknesses of my software design so that I may improve its final design.

WHAT IS INVOLVED IN THE STUDY?

If you decide to participate you will be asked to sit at a computer in one of our labs and answer questions from a pre-study survey. The questions will be about your general knowledge of 3D printing and should take no more than 5 minutes.

Participants will then be asked to perform a set of tasks to the best of their ability using our software. The tasks are short and simple, such as inputting given information into text fields or uploading a file.

Finally, participants will be asked to take a short post-study survey which asks them to rank the difficulty of given tasks from the prior exercise. Participants will also be able to add their own notes about their experience using our new software.

This research will take you about 30 min from start to finish.

You can stop participating at any time.

RISKS

There are no anticipated risks to participating in this study. However, if you experience any discomfort with using a computer then it is not recommended that you participate in this study.

BENEFITS TO TAKING PART IN THE STUDY?

There are no direct benefits to you however, participating in this research you may develop a better understanding of 3D printing and its surrounding technologies.

CONFIDENTIALITY

We will not be collecting any personally identifiable data during this study.

The results from this study will be published as a part of the student researcher's master's thesis and will only be shown in the form of what has been gained by the insight that you as a participant tell me is wrong with my software.

INCENTIVES

The only incentive in this study is the ability to take home a 3D printed model of your choice at the end of the study.

YOUR RIGHTS AS A RESEARCH PARTICIPANT

Participation in this study is voluntary. You have the right not to participate at all or to leave the study at any time. Deciding not to participate or choosing to leave the study will not result in any penalty or loss of benefits to which you are entitled, and it will not harm your relationship with WebSlicer and any University of Massachusetts Lowell staff.

CONTACTS FOR QUESTIONS OR PROBLEMS?

Call Fred Martin at 978-934-1964 or email at fredm@cs.uml.edu if you have questions about the study, any problems, unexpected physical or psychological discomforts, any injuries, or think that something unusual or unexpected is happening.

A.3 WebSlicer Recruitment Email

A recruitment email was needed to find a sufficient number of participants to preform usability testing.

Subject Line: 3D print slicer software prototype study

Professor Fred Martin and I, Michael Meding are looking for a small group of students to participate in our study, “Webslicer Usability Testing.”

You will use our new software called WebSlicer. WebSlicer is a 3D print slicer designed to run entirely on the web. Its purpose is to take a model from a 3D modeling software such as AutoCAD and “slice” it into layers such that a 3D printer can understand how to build it.

Your participation in this study will take about 30 minutes upon arrival you will be asked to give your consent for the study. The study will take place in our lab (Olsen 306) and will be in the form of an online survey given on one of our lab computers. As a participant you are free to stop the study at any time with no consequence. As a token of our appreciation you will be able to go home with a 3D printed model of your choice.

If you choose to participate please send an email to Michael_Meding@student.uml.edu with the subject “WebSlicer Study”. Please include your time availabilities and we will try our best to schedule you a time to meet with us.

Requirements:

- The ability to comfortably use a lab computer
- Some knowledge of the process of 3D printing is helpful but not required to participate

Please reply to Mike Meding, mikeymeding@gmail.com, if interested.

Thanks for your help!

A.4 WebSlicer Survey

The document that follows is the survey that was given to participants of the study. It consists of a few short questions and open ended comments.

WebSlicer Pre-Study Survey

Consent and questions while testing WebSlicer

* Required

1. **Have you read and signed the Consent Form and hereby give consent to participate in this study?** *
   Mark only one oval.
   - Yes
   - No *Stop filling out this form.*

Pre-Study Question 1

2. **Have you ever used a 3D printer before?**
   Mark only one oval.
   - Yes
   - No *Skip to question 5.*

Pre-Study Question 2

3. **Do you know what a slicer is or what slicing software does?**
   Mark only one oval.
   - Yes
   - No *Skip to question 5.*

Pre-Study Question 3

4. **What slicing software have you used or know about?**
   Check all that apply.
   - [ ] Cura
   - [ ] Repetier Host
   - [ ] KISSlicer
   - [ ] Other: _____

Post-Study Survey

5. **How would you rate the difficulty of Task 1 (Create a new user)?** *
Mark only one oval.

	1	2	3	4	5	
difficult	◯	◯	◯	◯	◯	easy

6. **Comments about Task 1**

7. **How would you rate the difficulty of Task 2 (Upload a file)?** *
Mark only one oval.

	1	2	3	4	5	
difficult	◯	◯	◯	◯	◯	easy

8. **Comments about Task 2**

9. **How would you rate the difficulty of Task 3 (Adjust slicer settings)?** *
Mark only one oval.

	1	2	3	4	5	
difficult	◯	◯	◯	◯	◯	easy

10. **Comments about Task 3**

11. **How would you rate the difficulty of Task 4 (Analyze and download Gcode)?** *
Mark only one oval.

	1	2	3	4	5	
difficult	◯	◯	◯	◯	◯	easy

12. **Comments about Task 4**

13. **What is one thing that you would change about this software?**

14. **Did you run into any problems while preforming the tasks given?**

15. **Was anything difficult to use or hard to find when you needed it?**

16. **Additional comments**

Powered by
Google Forms

A.5 WebSlicer Tasks

Participants in the usability study were given a series of tasks as described by the document that follows.

WebSlicer

181 Bacon St
Natick, MA 01760
(214) 334-1905

Task Instructions

March 03, 2016

Overview

As a participant your goal today is to follow all the steps involved in taking a model from the virtual world into the physical world with the power of 3D printing. Starting by choosing from a small subset of models which I have pre-selected, you will download this and load it into the slicing software which I have developed. The models which you will be using for this study have been pre printed and you will be able to take one home at the end of the study.

2

Choose Your Model

1

2

3

4

5

6

3

Task 1: Create a new user

User Goal: Get a new login ID.

This task is pretty simple. All you need to do is click the orange button labeled “generate new id.” If you wish to return to your work later please copy this number as it is your reference number to your file.

4

Task 2: Upload your file

User Goal: Take your chosen model file and upload it to WebSlicer.

In this task you simply need to take the file which you have chosen and upload it using the upload button under slicer controls. The file will be provided to you on a USB stick. You upload to Webslicer in the same way that you would by selecting a file in Word.

Your output should look something similar to the picture above.

5

Task 3: Adjust slicer settings

User Goal: Adjust settings so that the model will print correctly.

Basic Settings

The only settings that you will need to adjust will be under the basic tab. The settings that you need to find and change are as follows,

- Layer Height: 0.2
- Infill: 80%
- Filament Diameter: 1.75
- Shell Thickness in Lines: 2
- Printing Temperature: 245
- Bed Temperature: 110
- Print Speed: 40

The default values for the rest of the settings should do fine for your model.

6

Your window will look different from the one above. The picture is simply as a reference.

Task 4: Analyze and download Gcode

User Goal: Use visualizer to see the resultant Gcode and download it for printing.

When you have completed task 4 you have to press the slice button next to the model you wish to slice. The slice button should change into a spinning cog to indicate that it is working on slicing your model in the cloud. When it completes you should see some output that looks something like the following image,

7

You should also see a visualizer window like the one in the next picture. This is where you can step through all of the layers of your model and see exactly what it will do while it is printing.

When you are satisfied with this output you can press the “download gcode” text as it appears in the photo above. You will then get your Gcode file ready to be printed.

8

And that's it! You have successfully sliced your first model using WebSlicer. All that is left to do is grab the person conducting the research and tell them that you are done and you can watch as your model will get printed for you.

Thank you for your participation.

Appendix B

Server Side Code Appendix

Included in this appendix is the full source code written to run WebSlicer on a Wildfly application server. The section names are the file locations as they exist from the top level directory.

B.1 config/ApplicationConfig.java

Listing B.1: A required file in JavaEE for declaring the default application path when deployed.

```
1 package com.partbuzz.slicer.config;
2
3
4 import javax.ws.rs.ApplicationPath;
5 import javax.ws.rs.core.Application;
6
7 /**
8  * Created by mike on 1/9/16.
9  */
10 @ApplicationPath("")
11 public class ApplicationConfig extends Application {
12 }
```

B.2 config/CORSFilter

Listing B.2: As CuraEngine allows cross origin responses a filter is requred to allow access by default in WildFly.

```
1 package com.partbuzz.slicer.config;
```

```java

import java.io.IOException;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.ext.Provider;

/**
 * @author mike
 */
@Provider
public class CORSFilter implements ContainerResponseFilter {

    @Override
    public void filter(final ContainerRequestContext requestContext,
                       final ContainerResponseContext cres) throws IOException {
        cres.getHeaders().add("Access-Control-Allow-Origin", "*");
        cres.getHeaders().add("Access-Control-Allow-Headers", "origin, content-type, accept, authorization");
        cres.getHeaders().add("Access-Control-Allow-Credentials", "true");
        cres.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS, HEAD");
        cres.getHeaders().add("Access-Control-Max-Age", "1209600");
    }

}
```

B.3 cura/CuraEngine.java

Listing B.3: CuraEngine.java is the main code that preforms slicing throgh the CuraEngine C++ executable.

```java
/*
 * Copyright (c) 2016 Michael Meding -- All Rights Reserved.
 */
package com.partbuzz.slicer.cura;

import com.partbuzz.slicer.util.CuraEngineException;
import com.partbuzz.slicer.util.PlatformExecutable;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Logger;

/**
 * Invoke the CURA slicing engine.
 * <p>
 * SAMPLE BASH
```

```
19  * ../../CuraEngine-master/build/CuraEngine slice -v -j ultimaker2
        .json -g -e -o "output/test.gcode" -l ../../models/
        ControlPanel.stl
20  *
21  * @author mike
22  */
23 public class CuraEngine extends PlatformExecutable {
24
25     private static final Logger log = Logger.getLogger(CuraEngine.
           class.getName());
26     private static final String CURAENGINE_PROG = "/usr/local/bin/
           CuraEngine";
27     private final CuraEngineOptions options;
28
29     public CuraEngine() {
30         this.options = new CuraEngineOptions();
31     }
32
33     public CuraEngineOptions options() {
34         return options;
35     }
36
37     public String output = "";
38
39     /**
40      * Setup a stream drainer.
41      *
42      * @param fp the input stream
43      * @return the stream drainer
44      * @throws java.io.IOException
45      */
46     protected static StreamDrainer setupStreamDrain(InputStream fp
           ) throws IOException {
47         final StreamDrainer drainer = new StreamDrainer(fp);
48         // wait until we are draining
49         synchronized (drainer) {
50
51             log.info("starting thread");
52             Thread t = new Thread(drainer);
53             t.setDaemon(true);
54             t.start();
55
56             while (!drainer.hasStarted()) {
57                 try {
58                     log.info("waiting for drainer thread");
59                     drainer.wait(500);
60                     log.info("drainer wait expired");
61                 } catch (InterruptedException ex) {
62                     Thread.interrupted();
63                     throw new IOException("Unable to setup stream
                           drainer");
64                 }
65             }
66
67             log.info("drainer started");
68         }
69
70         return drainer;
71     }
72
```

```
73      /**
74       * Run the CURA engine.
75       *
76       * @throws CuraEngineException
77       */
78      public void execute() throws IOException {
79          checkPlatformExecutable(CURAENGINE_PROG);
80
81          List<String> arguments = new ArrayList<>();
82          arguments.add(CURAENGINE_PROG); // add base executable
83          arguments.addAll(options.getOptions()); // add all options
                in the correct order
84
85          for (String arg : arguments) {
86              log.info("cura argument: " + arg);
87          }
88
89          // Process builder options for command line execution
90          ProcessBuilder pb = new ProcessBuilder();
91 //     pb.directory(basePath);
92          pb.redirectErrorStream(true);
93          pb.command(arguments);
94
95          // Create new thread and start execution
96          Process p = pb.start();
97          try {
98
99              // Setup drains for both output and error streams
100             StreamDrainer stdout = setupStreamDrain(p.
                    getInputStream());
101 //              StreamDrainer stderr = setupStreamDrain(p.
        getErrorStream());
102
103             int status = p.waitFor(); // wait for slice to finish
104
105             if (status == 0) {
106
107                 // gather everything that is sent back from the
                        command.
108                 StringBuilder sb = new StringBuilder();
109                 InputStream fp = p.getInputStream();
110                 byte[] buffer = new byte[512];
111                 while (fp.available() > 0) {
112                     int n = fp.read(buffer);
113                     sb.append(new String(buffer, 0, n));
114                 }
115                 output = sb.toString();
116                 log.info(output); // just log the output for now
117                 log.info("STDOUT:" + stdout.getText());
118                 return;
119             } else {
120                 throw new CuraEngineException("unable to parse
                        gcode file");
121             }
122
123 //              if (options.haveIgnoreErrors()) {
124 //                  log.warning(sb.toString());
125 //              } else {
126 //                  throw new CuraEngineException(sb.toString());
```

```
//            }


        } catch (InterruptedException e) {
            throw new IOException(e);
        } finally {
            cleanupResources(p);
        }
    }

    /**
     * After execute is called the standard out of the call is
        concatenated into output.
     *
     * @return
     */
    public String getOutput() {
        return output;
    }
}
```

B.4 cura/CuraEngineOptions.java

Listing B.4: All arguments to the CuraEngine C++ executeable must be compiled before it can be executed. This code is the compiler for that argument list.

```
package com.partbuzz.slicer.cura;

import com.partbuzz.slicer.util.CuraEngineException;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by mike on 3/30/16.
 */

public class CuraEngineOptions {

    private String settingsFileName;
    private String outputFileName;
    private String modelFileName;
    private boolean verbose = false;
    private boolean p_option = false;
    private boolean g_option = false;
    private boolean e_option = false;
    private boolean ignoreErrors = false;

    /**
     * Set the verbose mode.
     *
     * @return the options
     */
    public CuraEngineOptions verbose() {
        this.verbose = true;
        return this;
```

```
31      }
32
33      /**
34       * Set the settings file. This is a JSON formatted file which
           includes all needed information for slicing the file.
35       *
36       * @param filename the settings filename
37       * @return the options
38       */
39      public CuraEngineOptions settingsFilename(String filename) {
40          this.settingsFileName = filename;
41          return this;
42      }
43
44      /**
45       * Set the output filename. The resulting gcode file name/
           location.
46       *
47       * @param filename the output filename
48       * @return the options
49       */
50      public CuraEngineOptions outputFilename(String filename) {
51          this.outputFileName = filename;
52          return this;
53      }
54
55      /**
56       * Set the model filename. This is the model to be sliced.
57       *
58       * @param filename
59       * @return options
60       */
61      public CuraEngineOptions modelFilename(String filename) {
62          this.modelFileName = filename;
63          return this;
64      }
65
66      /**
67       * This option is similar to verbose but instead of just
           logging the output to the screen it logs the output to
           the CuraEngine log files.
68       *
69       * @return the options
70       */
71      public CuraEngineOptions logProgress() {
72          this.p_option = true;
73          return this;
74      }
75
76      /**
77       * Switch setting focus to the current mesh group only.
78       * Used for one-at-a-time printing.
79       *
80       * @return the options
81       */
82      public CuraEngineOptions currentGroupOnly() {
83          g_option = true;
84          return this;
85      }
```

```
86
87      /**
88       * Adds a new extruder train for multi extrusion. For every
             time -e is included another extruder is added as on
             option to the slicer.
89       *
90       * @return the options
91       */
92      public CuraEngineOptions extruderTrainOption() {
93          e_option = true;
94          return this;
95      }
96
97      /**
98       * For debugging.
99       *
100      * @return the options
101      */
102     public CuraEngineOptions ignoreErrors() {
103         this.ignoreErrors = true;
104         return this;
105     }
106
107     public boolean haveIgnoreErrors() {
108         return ignoreErrors;
109     }
110
111     /*
112      * Get the options in the right sequence. To build a correct
             CuraEngine command line executable.
113      */
114     public List<String> getOptions() throws CuraEngineException {
115         List<String> list = new ArrayList<>();
116         list.add("slice"); // the main parameter of the CuraEngine
                 executable.
117         if (verbose) {
118             list.add("-v");
119         }
120         if (p_option) {
121             list.add("-p");
122         }
123
124         if (settingsFileName == null) {
125             throw new CuraEngineException("no setting file defined
                    ");
126         } else {
127             list.add("-j");
128             list.add(settingsFileName);
129         }
130         if (g_option) {
131             list.add("-g");
132         }
133         if (e_option) {
134             list.add("-e");
135         }
136         if (outputFileName == null) {
137             throw new CuraEngineException("no output file defined"
                    );
138         } else {
139             list.add("-o");
```

```
            list.add(outputFileName);
        }
        if (modelFileName == null) {
            throw new CuraEngineException("no model file defined")
                ;
        } else {
            list.add("-l");
            list.add(modelFileName);
        }

        return list;
    }
}
```

B.5 cura/StreamDrainer.java

Listing B.5: For ProcessBuilder to run correctly its output stream must be read from. This code spawns a separate thread to read and log this output stream.

```
package com.partbuzz.slicer.cura;

import java.io.IOException;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Drain an input stream into a buffer.
 */
class StreamDrainer implements Runnable {

    private static final Logger log = Logger.getLogger(
        StreamDrainer.class.getName());
    private final InputStream fp;
    private final StringBuilder sb;
    private boolean started;

    public StreamDrainer(InputStream fp) {
        this.fp = fp;
        this.sb = new StringBuilder();
        this.started = false;
    }

    public String getText() {
        return sb.toString();
    }

    boolean hasStarted() {
        return started;
    }

    @Override
    public void run() {

        // tell the invoker that we are draining
        log.info("starting drainer thread");
        synchronized (this) {
```

```
38                  log.info("drainer thread  sync");
39
40                  started = true;
41                  notifyAll();
42
43                  byte[] buffer = new byte[512];
44                  try {
45                      boolean reading = true;
46                      int n;
47                      do {
48                          n = fp.read(buffer);
49                          if (n > 0) {
50                              sb.append(new String(buffer, 0, n));
51                          }
52                      } while (n >= 0);
53                  } catch (IOException ex) {
54                      Logger.getLogger("PlatformExecutor").log(Level.
                           SEVERE, null, ex);
55                  }
56                  log.info("done with the drainer");
57              }
58          }
59      }
```

B.6 rest/SlicerAPI.java

Listing B.6: The Wildfly application server hosts a public RESTful API which is where requests for slicing and data processing get sent to their corresponding methods.

```
1  package com.partbuzz.slicer.rest;
2
3  import com.partbuzz.slicer.cura.CuraEngine;
4  import com.partbuzz.slicer.util.CuraEngineException;
5  import com.partbuzz.slicer.util.FileTracker;
6  import os.io.FileHelper;
7  import os.util.ExceptionHelper;
8  import os.util.StringUtils;
9  import os.util.json.DefaultJSONFactory;
10 import os.util.json.JSONException;
11 import os.util.json.JSONObject;
12
13 import javax.mail.internet.MimeBodyPart;
14 import javax.mail.internet.MimeMultipart;
15 import javax.mail.util.ByteArrayDataSource;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.ws.rs.*;
18 import javax.ws.rs.core.Context;
19 import javax.ws.rs.core.MediaType;
20 import javax.ws.rs.core.Response;
21 import java.io.*;
22 import java.util.HashMap;
23 import java.util.logging.Level;
24 import java.util.logging.Logger;
25
26 /**
27  * Created by mike on 1/9/16.
28  */
```

```
29 @Path("slicer")
30 public class SlicerAPI {
31     private static final Logger log = Logger.getLogger(SlicerAPI.
           class.getName());
32     private final DefaultJSONFactory JSONFactory =
           DefaultJSONFactory.getInstance();
33
34     /**
35      * Simply returns 200 OK if called.
36      * <p>
37      *
38      * @param req
39      * @return
40      */
41     @GET
42     @Path("ping")
43     @Produces({MediaType.TEXT_PLAIN})
44     public Response ping(@Context HttpServletRequest req
45     ) {
46         log.info("ping called");
47         return Response.ok("PONG", MediaType.TEXT_PLAIN).build();
48     }
49
50     /**
51      * Add a new client to our database and file structure.
52      * This will set aside all of the files needed for a new
           client and link them correctly
53      *
54      * @param req
55      * @return The new uuid for that client so that we may track
           files
56      */
57     @POST
58     @Path("setupClient")
59     public Response setupClient(@Context HttpServletRequest req) {
60         try {
61             String uuid = FileTracker.setupNewClient();
62             JSONObject payload = new JSONObject();
63             payload.put("clientId", uuid);
64
65             return Response.ok().entity(payload).build();
66         } catch (CuraEngineException e) {
67             e.printStackTrace();
68             return Response.serverError().build();
69         }
70     }
71
72
73     /**
74      * Import an STL model file from the client
75      *
76      * @param req is the http request
77      * @return the names of the uploaded files or an error message
78      */
79     @POST
80     @Path("importStl/{clientId}")
81     public Response importStl(@Context HttpServletRequest req,
82                               @PathParam("clientId") String
                                     clientId) {
83         try {
```

```
84
85              ByteArrayDataSource bads = new ByteArrayDataSource(req
                    .getInputStream(), req.getContentType());
86              MimeMultipart mmp = new MimeMultipart(bads);
87
88              // Read the raw file
89              String filename = "";
90              for (int i = 0; i < mmp.getCount(); i++) {
91                  MimeBodyPart mbp = (MimeBodyPart) mmp.getBodyPart(
                        i);
92
93                  byte[] buffer = new byte[mbp.getSize()];
94                  mbp.getInputStream().read(buffer);
95
96                  // calculate a filename
97 //                 String md5Hex = StringUtils.encodeHexString(
       StringUtils.md5(buffer));
98 //                 filename = StringUtils.encode(md5Hex + "-" + mbp
       .getFileName());
99                  filename = StringUtils.encode(mbp.getFileName());
100
101                 // Save the file to disk
102                 File file = new File(FileTracker.getModelPathById(
                        clientId) + FileTracker.delimiter + filename);
103                 FileOutputStream fp = new FileOutputStream(file);
104                 fp.write(buffer);
105                 fp.close();
106
107             }
108
109             // register our new model file
110             String fileId = FileTracker.registerModelFile(clientId
                    , filename);
111
112             // build response message
113             JSONObject response = new JSONObject();
114             response.put("fileId", fileId);
115
116             return Response.ok().entity(response.toString()).build
                    ();
117         } catch (Exception e) {
118             return Response.ok().entity(ExceptionHelper.
                    getStackTrace(e)).build();
119         }
120     }
121
122     /**
123      * Import a settings file from the client
124      *
125      * @param req is the http request
126      * @return the names of the uploaded files or an error message
127      */
128     @POST
129     @Path("importSettings/{clientId}")
130     public Response importSettings(@Context HttpServletRequest req
            ,
131                                    @PathParam("clientId") String
                                           clientId) {
132         log.log(Level.INFO, "importing settings ");
```

```
133         StringBuilder sb = new StringBuilder();
134         try {
135             String line;
136 
137             // parse input stream into a string
138             try (BufferedReader br = new BufferedReader(new
                    InputStreamReader(req.getInputStream()))) {
139                 while ((line = br.readLine()) != null) {
140                     sb.append(line);
141                 }
142             }
143             String data = sb.toString(); // finish parse
144 
145             // Save to file
146             String settingsPath = FileTracker.getSettingsFullPath(
                    clientId);
147 
148             // write/overwrite file
149             try (PrintWriter out = new PrintWriter(settingsPath))
                    {
150                 out.print(data);
151             }
152 
153             return Response.ok().build();
154         } catch (IOException e) {
155             e.printStackTrace();
156             return Response.serverError().entity("File write error
                    ").build();
157         }
158     }
159 
160     /**
161      * Slice is the main function of this API. Given the
             parameters below it will return a formatted .gcode file
162      * to the user.
163      * <p>
164      * The parameters required are a .stl file ID and a .json
             settings file ID.
165      * These parameters are given when the above API methods are
             called. It is up to the client to track these ID's
166      * <p>
167      * SAMPLE JSON FORMAT
168      * {"modelId":"UUID","settingsId":"UUID"}
169      *
170      * @param req is the http request
171      * @return the names of the uploaded files or an error message
172      */
173     @POST
174     @Path("slice/{clientId}/{modelId}")
175     public Response slice(@Context HttpServletRequest req,
176                           @PathParam("clientId") String clientId,
                                  @PathParam("modelId") String modelId
                                  ) {
177         try {
178             // SAMPLE COMMAND RESULT
179             // ../../CuraEngine-master/build/CuraEngine slice -v -
                    j ultimaker2.json -g -e -o "output/test.gcode" -l
                    ../../models/ControlPanel.stl
180 
```

```
181             // create new CuraEngine platform executable
182             CuraEngine ce = new CuraEngine();
183             ce.options() // add options. order is irrelevant.
184                     .verbose()
185                     .currentGroupOnly()
186                     .extruderTrainOption()
187                     .logProgress()
188                     .settingsFilename(FileTracker.
                           getSettingsFullPath(clientId))
189                     .outputFilename(FileTracker.getOutputFilePath(
                           clientId))
190                     .modelFilename(FileTracker.getModelFullPath(
                           clientId, modelId));
191             ce.execute();
192
193             // build response from output file and return it.
194             JSONObject response = new JSONObject();
195             byte[] raw = FileHelper.readContent(new File(
                   FileTracker.getOutputFilePath(clientId)));
196             String gcode = new String(raw);
197             response.put("gcode", gcode);
198
199             return Response.ok().entity(response.toString()).build
                   ();
200         } catch (IOException e) {
201             return Response.serverError().entity(e.getMessage()).
                   build();
202         }
203     }
204
205     /**
206      * This function is to run a full test slice with a sample
            model and return its gcode.
207      * Eliminates the need for a UI to run backend tests.
208      * @param req is the http request
209      * @return the names of the uploaded files or an error message
210      */
211     @POST
212     @Path("testSlice")
213     public Response testSlice(@Context HttpServletRequest req) {
214         try {
215             // create new CuraEngine platform executable
216             CuraEngine ce = new CuraEngine();
217             ce.options() // add options. order is irrelevant.
218                     .verbose()
219                     .currentGroupOnly()
220                     .extruderTrainOption()
221                     .logProgress()
222                     .settingsFilename("/tmp/webslicer/test/
                           prusa_i3.json")
223                     .outputFilename("/tmp/webslicer/test/output.
                           gcode")
224                     .modelFilename("/tmp/webslicer/test/testModel.
                           stl");
225             ce.execute();
226
227             // build response from output file and return it.
228             JSONObject response = new JSONObject();
229             byte[] raw = FileHelper.readContent(new File("/tmp/
                   webslicer/test/output.gcode"));
```

```
230             String gcode = new String(raw);
231             response.put("gcode", gcode);
232 
233             return Response.ok().entity(response.toString()).build
                    ();
234         } catch (IOException e) {
235             return Response.serverError().entity(e.getMessage()).
                    build();
236         }
237     }
238 
239     /**
240      * Get all of the model file names and tracking ids associated
                 with a client uuid.
241      *
242      * @param req
243      * @param clientId
244      * @return
245      */
246     @GET
247     @Path("getFiles/{clientId}")
248     @Produces("application/json")
249     public Response getFiles(@Context HttpServletRequest req,
250                              @PathParam("clientId") String
                                     clientId) {
251         HashMap filesMap = FileTracker.getAllModelFiles(clientId);
252         return Response.ok().entity(filesMap).build();
253     }
254 
255     /**
256      * A util function to parse an input JSON stream and parses to
                 a JSONObject
257      *
258      * @param req
259      * @return
260      */
261     private JSONObject parseJSONStream(HttpServletRequest req) {
262         StringBuilder sb = new StringBuilder();
263         try {
264             String line;
265 
266             // parse input stream into a string
267             try (BufferedReader br = new BufferedReader(new
                    InputStreamReader(req.getInputStream()))) {
268                 while ((line = br.readLine()) != null) {
269                     sb.append(line);
270                 }
271             }
272             String data = sb.toString();
273 
274             log.info(data);
275 
276             return JSONFactory.jsonObject(data);
277 
278         } catch (IOException | JSONException e) {
279             log.severe(e.getMessage());
280             throw new JSONException("Stream parse error.");
281         }
282 
283     }
```

```
}
```

B.7 util/CuraEngineException.java

Listing B.7: This exception class gets thrown when any issue arises while CuraEngine is preforming a slice.

```
/*
 * Copyright (c) 2016 Michael Meding -- All Rights Reserved.
 */
package com.partbuzz.slicer.util;

import java.io.IOException;

/**
 * A CURA engine exception.
 *
 * @author mike
 */
public class CuraEngineException extends IOException {

  public CuraEngineException() {
  }

  public CuraEngineException(String message) {
    super(message);
  }

  public CuraEngineException(String message, Throwable cause) {
    super(message, cause);
  }

  public CuraEngineException(Throwable cause) {
    super(cause);
  }

}
```

B.8 util/FileTracker.java

Listing B.8: FileTracker tracks all of the files uploaded to the server to a series of hash maps. It also issues unique identifiers for each file that gets uploaded so that they may be tracked and accessed simply.

```
package com.partbuzz.slicer.util;

import os.io.FileHelper;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
```

```
8  import java.nio.file.Path;
9  import java.nio.file.Paths;
10 import java.util.HashMap;
11 import java.util.Map;
12 import java.util.UUID;
13
14 /**
15  * For tracking the associated files with a particular client id.
16  * FILE STRUCTURE
17  * /main---/(clientUUID)---output.gcode
18  * -settings.json
19  * -/models---(uuid-filename.stl)
20  *  -(...stl)
21  *  -...
22  * <p>
23  * -/common---fdmprinter.json
24  * -/presets---ultimaker.json
25  * -prusai3.json
26  *  -...
27  * <p>
28  * Structure dictates file limit and will throw exceptions if # of
       files goes above limit.
29  * <p>
30  * Created by mike on 1/14/16.
31  */
32 public class FileTracker {
33     //TODO: This really needs to come from a properties file (JNDI
           path to properties file)
34     public static String delimiter = File.separator;
35     public static String basePath = delimiter + "tmp" + delimiter
           + "webslicer";
36     public static String common = basePath + delimiter + "common";
37     public static String modelDir = "models";
38     public static String settingsFileName = "settings.json";
39     public static String fdmprinterFile = "fdmprinter.json";
40     public static String outputFile = "output.gcode";
41     private static Map<String, HashMap<String, String>>
           clientFilesMap = new HashMap<>();
42
43     /**
44      * Same as above only for the model file.
45      *
46      * @param clientId The id of the client in question
47      * @param fileId   The tracked filename which must be managed
           by client
48      * @return
49      */
50     public static String getModelFileName(String clientId, String
           fileId) {
51         return clientFilesMap.get(clientId).get(fileId);
52     }
53
54     /**
55      * Get the full path to a model file
56      *
57      * @param clientId
58      * @param fileId
59      * @return
60      */
```

```
61      public static String getModelFullPath(String clientId, String
            fileId) {
62          return basePath + delimiter + clientId + delimiter +
                modelDir + delimiter + getModelFileName(clientId,
                fileId);
63      }
64
65      /**
66       * Get the path to the model file directory
67       *
68       * @return
69       */
70      public static String getModelPathById(String clientId) {
71          return basePath + delimiter + clientId + delimiter +
                modelDir;
72      }
73
74      /**
75       * Return the entire hashmap of model files associated with a
             client
76       *
77       * @param clientId
78       * @return
79       */
80      public static HashMap<String, String> getAllModelFiles(String
            clientId) {
81          return clientFilesMap.get(clientId);
82      }
83
84      public static String getOutputFilePath(String clientId) {
85          return basePath + delimiter + clientId + delimiter +
                outputFile;
86      }
87
88      /**
89       * Register a new model file with our registry
90       *
91       * @param fileName
92       * @return
93       */
94      public static String registerModelFile(String clientId, String
             fileName) {
95          String uuid = UUID.randomUUID().toString(); // generate
                new tracking uuid
96          clientFilesMap.get(clientId).put(uuid, fileName); // store
                 this filename using that new tracking uuid for that
                client
97          return uuid;
98      }
99
100     /**
101      * Gets the fully qualified path to the settings file for a
             specified client
102      *
103      * @param clientId
104      * @return
105      */
106     public static String getSettingsFullPath(String clientId) {
107         return basePath + delimiter + clientId + delimiter +
                settingsFileName;
```

```
108     }
109
110     /**
111      * Remove all files associated with a client and remove its
             hashmap tracking in our server
112      *
113      * @param clientId
114      */
115     public static void removeClient(String clientId) {
116         try {
117             FileHelper.delete(new File(basePath + delimiter +
                    clientId));
118             clientFilesMap.remove(clientId);
119         } catch (IOException e) {
120             e.printStackTrace();
121         }
122     }
123
124     /**
125      * Completely remove model file from our disk
126      *
127      * @param clientId
128      * @param id
129      */
130     public static void removeModelFile(String clientId, String id)
            {
131         try {
132             FileHelper.delete(new File(getModelFullPath(clientId,
                    id)));
133             clientFilesMap.get(clientId).remove(id);
134         } catch (IOException e) {
135             e.printStackTrace();
136         }
137     }
138
139
140     /**
141      * Register and setup a new client with our registry.
142      *
143      * @return the new clients UUID.
144      */
145     public static String setupNewClient() throws
            CuraEngineException {
146
147         // generate new client uuid
148         String uuid = UUID.randomUUID().toString();
149
150         // Create empty file as clients home directory
151         if (!new File(basePath + delimiter + uuid + delimiter +
                modelDir).mkdirs()) {
152             throw new CuraEngineException("Could not create file
                    path correctly");
153         }
154
155         // Symbolically link fdmprinter.json from common to this
                clients home directory
156         try {
157             Path targetPath = Paths.get(basePath + delimiter +
                    uuid + delimiter + fdmprinterFile);
```

```
            Path sourceFile = Paths.get(common + delimiter + fdmprinterFile);
            Files.createSymbolicLink(targetPath, sourceFile);
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Give our new client a file tracking hashmap
        clientFilesMap.put(uuid, new HashMap<String, String>());// initalize client with new empty hashmap

        return uuid;
    }

}
```

B.9 util/PlatformExecutable.java

Listing B.9: The PlatformExecutable class verifies if the local C++ executable is available and sets up the input and output pipes for CuraEngine if it is.

```
/*
 * Copyright (c) 2016 Michael Meding -- All Right Reserved.
 */
package com.partbuzz.slicer.util;

import java.io.Closeable;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;

/**
 * See if an executable is present and runnable.
 *
 * @author mike
 */
public abstract class PlatformExecutable {

  /**
   * Check if a particular executable exists
   *
   * @param path is the path
   */
  protected static void checkPlatformExecutable(String path) {
    File file = new File(path);
    if (!(file.exists() && file.canExecute())) {
      throw new UnsupportedOperationException(path + " is not supported on this platform");
    }
  }

  /**
   * See if a port is in the valid range 0-65535
   *
   * @param port the port number
   */
```

```
35    protected static void checkPortNumberLegal(int port) {
36      if (port < 0 || port > 65535) {
37        throw new IllegalArgumentException("TCP/IP Port " + port + "
            illegal");
38      }
39    }
40
41    /**
42     * Gather all the output a command may have sent back. Used if a
          system
43     * command did not finish successfully.
44     *
45     * @param p is the process
46     * @return the string
47     */
48    protected static String gatherCommandOutput(Process p) throws
        IOException {
49      StringBuilder sb = new StringBuilder();
50      InputStream fp = p.getInputStream();
51      byte[] buffer = new byte[512];
52      while (fp.available() > 0) {
53        int n = fp.read(buffer);
54        sb.append(new String(buffer, 0, n));
55      }
56
57      return sb.toString();
58    }
59
60    /**
61     * Cleanup the external process resources
62     *
63     * @param p is the process
64     */
65    protected static void cleanupResources(Process p) {
66      if (p == null) {
67        return;
68      }
69
70      close(p.getOutputStream());
71      close(p.getInputStream());
72      close(p.getErrorStream());
73      p.destroy();
74    }
75
76    private static void close(Closeable c) {
77      if (c != null) {
78        try {
79          c.close();
80        } catch (Throwable t) {
81          // ignore
82        }
83      }
84    }
85  }
```

Appendix C

Client Side Code Appendix

Included in this appendix is the full source code for the client side WebSlicer application.

C.1 app.js

Listing C.1: Main app code where the angular module WebSlicer is defined. This is similar to a main function in C++.

```
/**
 * Created by mike on 9/25/15.
 */
(function () {
    var app = angular.module("WebSlicer", ["ui.router", "ui.bootstrap", "ngAnimate", "ngFileUpload"]);

    app.controller("Main", ["$http", "$scope", "$window", "$rootScope", function ($http, $scope, $window, $rootScope) {

        // enable all bootstrap tooltips
        //$(function () {
        //    $('[data-toggle="tooltip"]').tooltip()
        //});

        // the main servers base url for making REST calls
        $rootScope.baseUrl = "http://localhost:8080/WebSlicer/slicer";
        $rootScope.clientId = "";
        $rootScope.busy = false;

        // environment vars
        //$rootScope.baseUrl;
        $scope.title = "Web Slicer";
```

```
22          $scope.displayOctoprintData = false;
23          //$scope.clientId = "";
24          $scope.modelFiles = {};
25          $scope.gcode = {"gcode": ""};
26
27          // make gcode available for download
28          var blob = new Blob([$scope.gcode.gcode], {type: 'text/
               plain'});
29          $scope.gcodeDownload = (window.URL || window.webkitURL).
               createObjectURL(blob);
30
31          /**
32           * Generate a new client id and layout all of the files
                for this new client on the server
33           */
34          $scope.generateClientId = function () {
35              $http({
36                  method: 'POST',
37                  url: $rootScope.baseUrl + "/setupClient"
38              }).then(function successCallback(response) {
39                  console.log(response);
40                  $rootScope.clientId = response.data.clientId;
41              }, function errorCallback(response) {
42                  console.error(response);
43              });
44          };
45
46          //DOWNLOAD GCODE FUNCTION
47          $scope.downloadGcode = function (gcode) {
48              var blob = new Blob([gcode], {type: "application/json;
                   charset=utf-8;"});
49              var downloadLink = angular.element('<a></a>');
50              //var downloadLink = angular.element(elem.
                   querySelector(".downloadGcode"));
51              downloadLink.attr('href', window.URL.createObjectURL(
                   blob));
52              downloadLink.attr('download', 'output.gcode');
53              downloadLink[0].click();
54
55              //$window.open("data:application/json;charset=utf-8,"
                   + encodeURIComponent(gcode));
56          };
57
58
59          /**
60           * The main slice function. This will only operate if the
                client has both uploaded a settings file and a model
                file.
61           * Both of these files will be given an id that the client
                 must track in order to slice a file properly.
62           *
63           * @param modelId
64           */
65          $rootScope.slice = function (modelId) {
66              if (modelId) {
67                  $rootScope.busy = true;
68
69                  // run import settings to update settings on
                       server BEFORE SLICING
```

```
70                  $rootScope.importSettings()
71
72                      .then(function successCallback(response) {
73                          console.log(response);
74
75                          // SEND SLICE REQUEST!
76                          $http({
77                              method: 'POST',
78                              url: $rootScope.baseUrl + "/slice/" +
                                    $rootScope.clientId + "/" +
                                    modelId
79                          }).then(function successCallback(response)
                                {
80                              // capture our gcode as a response.
81                              console.log(response);
82                              $scope.gcode = response.data;
83                              $rootScope.busy = false;
84
85                              // render our gcode in the 2D viewer
86                              $rootScope.render(response.data.gcode)
                                    ;
87
88                          }, function errorCallback(response) {
89                              console.error(response);
90                              $rootScope.busy = false;
91                          });
92
93                      }, function errorCallback(response) {
94                          console.error(response);
95                          console.error("import settings before
                                slice");
96                      });
97
98              } else {
99                  console.error("model or settings id missing");
100             }
101         };
102     }]);
103
104     app.config(["$stateProvider", "$urlRouterProvider", "$
            controllerProvider", "$compileProvider", function
            appConfig($stateProvider, $urlRouterProvider, $
            controllerProvider, $compileProvider) {
105
106         // default route
107         $urlRouterProvider.otherwise("home");
108
109         // states
110         $stateProvider
111             .state("home", {
112                 url: "/home",
113                 templateUrl: "public/index.html"
114             });
115
116         // controller config
117         $controllerProvider.allowGlobals();
118
119         // post compile config
120         $compileProvider.aHrefSanitizationWhitelist(/^\s*(https?|
                ftp|mailto|tel|file|blob):/);
```

```
    }]);

    app.directive('fileModel', ['$parse', function ($parse) {
        return {
            restrict: 'A',
            link: function (scope, element, attrs) {
                var model = $parse(attrs.fileModel);
                var modelSetter = model.assign;

                element.bind('change', function () {
                    scope.$apply(function () {
                        modelSetter(scope, element[0].files[0]);
                    });
                });
            }
        };
    }]);
})();
```

C.2 index.html

Listing C.2: As this is a single page web application this is the only HTML file. It contains all of the libraries needed to run all subsequent sections.

```
<!DOCTYPE html>
<html ng-app="WebSlicer">
<head lang="en">
    <meta charset="UTF-8">
    <title>Web Slicer</title>

    <link rel="icon" type="image/png" href="img/logo22-plain.png"/>

    <!--libraries-->
    <script src="bower_components/jquery/dist/jquery.js"></script>
    <script src="bower_components/bootstrap/dist/js/bootstrap.js"></
        script>
    <script src="bower_components/angular/angular.js"></script>
    <script src="bower_components/angular-animate/angular-animate.js"
        ></script>
    <script src="bower_components/angular-ui-router/release/angular-
        ui-router.js"></script>
    <script src="bower_components/angular-bootstrap/ui-bootstrap.js">
        </script>
    <script src="bower_components/angular-bootstrap/ui-bootstrap-tpls
        .js"></script>
    <script src="bower_components/ng-file-upload/ng-file-upload.js">
        /script>

    <link rel="stylesheet" href="bower_components/font-awesome/css/
        font-awesome.css">

    <!--css-->
    <!--<link rel="stylesheet" href="bower_components/bootstrap/dist/
        css/bootstrap.css">-->
    <link rel="stylesheet" href="bower_components/bootstrap/dist/css/
        bootstrap-cosmo.css">
```

```
24      <link rel="stylesheet" href="css/custom.css"/>
25
26      <!-- MAIN CONTROLLER AND CONFIG -->
27      <script src="app.js"></script>
28
29      <!--FILE CONTROLS-->
30      <script src="model/files/filesController.js"></script>
31      <script src="model/files/filesDirective.js"></script>
32
33      <!--SETTINGS-->
34      <script src="model/profile/settingsController.js"></script>
35      <script src="model/profile/settingsTabDirective.js"></script>
36      <script src="model/profile/settingsBuilderService.js"></script>
37
38      <!--OCTOPRINT-->
39      <script src="model/octoprint/octoprintService.js"></script>
40      <script src="model/octoprint/octoprintDirective.js"></script>
41      <script src="model/octoprint/octoprintController.js"></script>
42
43      <!--VISUALIZER-->
44      <script src="viewer/lib/zlib.min.js"></script>
45      <script src="viewer/gcodeFactory.js"></script>
46      <script src="viewer/renderFactory.js"></script>
47      <script src="viewer/viewer.js"></script>
48
49  </head>
50  <body ng-controller="Main">
51
52  <div class="container" ng-controller="SettingsController">
53  <div class="row">
54      <div class="col-lg-6 col-lg-offset-3">
55          <h1 class="title"><img src="img/logo22.svg" width="150px">{{
                title}}</h1>
56      </div>
57  </div>
58  <div class="row">
59  <div class="col-lg-6 col-lg-offset-3">
60      <div class="panel panel-default">
61          <div class="panel-heading">
62              <h3>Login</h3>
63          </div>
64          <div class="panel-body">
65          <form>
66          <div class="form-group">
67          <label for="clientId">Client ID</label>
68          <input class="form-control" type="text" id="clientId" ng-
                model="$root.clientId">
69          <button class="btn btn-warning" ng-click="generateClientId()"
                >generate new id</button>
70          </div>
71          </form>
72          </div>
73      </div>
74  </div>
75  </div>
76
77  <div class="row">
78
```

```
<!--CONTROLS-->
<div class="col-lg-6">

<!--<filecontrols></filecontrols>-->
<div ng-controller="FilesController">

<div class="panel panel-primary">
    <div class="panel-heading">
        <h3>Slicer Controls</h3>
    </div>
    <div class="panel-body">
        <ul class="nav nav-pills nav-stacked">

            <!--file chooser-->
            <li role="presentation">

                <button class="btn btn-primary" type="file" ngf-select="uploadFile($file,$invalidFiles)"
                        ngf-max-height="1000" ngf-max-size="2MB">
                    Upload File
                </button>

                <div ng-show="f.progress >= 0">
                <h3>{{f.name}} {{errFile.name}} {{errFile.$error}} {{errFile.$errorParam}}</h3>
                <div class="progress">
                <div class="progress-bar" role="progressbar" aria-valuenow="{{f.progress}}"
                  aria-valuemin="0"
                  aria-valuemax="100" style="width: {{f.progress}}%;">
                    <!--<div class="progress-bar" role="progressbar" aria-valuenow="{{f.progress}}" aria-valuemin="0"-->
                    <!--aria-valuemax="100">-->
                    {{f.progress}}% Complete
                </div>
                </div>
                </div>

            </li>

            <!--current file list-->
            <li role="presentation" ng-if="$root.clientId">
                <p>{{modelFileId}}</p>

                <table class="table table-hover table-condensed">
                    <tbody>

                    <tr ng-repeat="item in modelFiles">
                    <td>{{item.value}}
                    <button ng-if="!$root.busy" class="btn btn-primary btn-sm pull-right"
                            ng-click="$root.slice(item.key)">
                        Slice
                    </button>
                    <i ng-if="$root.busy" class="fa fa-3x fa-cog fa-spin pull-right"></i>

```

```
130
131                     </td>
132                     </tr>
133
134                     </tbody>
135                 </table>
136
137             </li>
138
139             <!--slice button-->
140             <li role="presentation">
141                 <!--<button class="btn btn-primary" ng-click="
                        getFileList()">Refresh file list</button>-->
142             </li>
143
144         </ul>
145
146     </div>
147 </div>
148
149
150     </div>
151 </div>
152
153 <!--<div class="col-lg-6">-->
154 <!--<octoprint></octoprint>-->
155 <!--</div>-->
156
157 </div>
158
159 <div class="row" ng-controller="SettingsController">
160
161 <!--SETTINGS PANEL-->
162 <div class="col-lg-6">
163
164     <div class="panel panel-default">
165         <div class="panel-heading">
166             <h3>Printer Settings</h3>
167         </div>
168         <div class="panel-body">
169
170             <uib-tabset>
171             <uib-tab heading="basic">
172                 <curasettings src="model/profile/json/basic.json"></
                        curasettings>
173             </uib-tab>
174             <uib-tab heading="advanced">
175                 <curasettings src="model/profile/json/advanced.json">
                        </curasettings>
176             </uib-tab>
177             <uib-tab heading="fine-tune">
178                 <curasettings src="model/profile/json/fine-tune.json"
                        ></curasettings>
179             </uib-tab>
180             <uib-tab heading="machine">
181                 <curasettings src="model/profile/json/machine.json"><
                        /curasettings>
182             </uib-tab>
```

```
183             <uib-tab heading="start/end gcode">
184                 <curasettings src="model/profile/json/start-end-gcode
                        .json"></curasettings>
185             </uib-tab>
186             </uib-tabset>
187
188         </div>
189     </div>
190
191 </div>
192
193 <!--GCODE OUTPUT PANEL-->
194 <div class="col-lg-6" ng-if="gcode.gcode">
195     <div class="panel panel-success">
196         <div class="panel-heading">
197             <h3>Gcode Output Window</h3>
198             <a ng-href="{{gcodeUrl}}" ng-click="downloadGcode(gcode.
                    gcode)">download gcode</a>
199         </div>
200         <div class="panel-body">
201             <button class="btn btn-sm" ng-click="showGcode = !
                    showGcode" data-toggle="tooltip" data-placement="
                    bottom" title="WARNING! This may freeze browser!">
                    press to view gcode</button>
202             <textarea ng-if="showGcode" cols="30" rows="50" ng-model=
                    "gcode.gcode"
203                         class="form-control"></textarea>
204
205         </div>
206     </div>
207 </div>
208 </div>
209
210 <!--2D GCODE VISUALIZER-->
211 <div class="row" ng-controller="gcodeController" ng-show="gcode.gcode
        ">
212 <div class="panel panel-success">
213     <div class="panel-heading">
214         <h3>Gcode Visualizer</h3>
215     </div>
216     <div class="panel-body">
217         <canvas id="canvas" width="650" height="620"></canvas>
218         <form>
219             <div class="form-group">
220                 <label for="layerNumber">layerNumber</label>
221                 <input type="number" id="layerNumber" ng-model="
                        layerNumber">
222                 <label for="progress">progress</label>
223                 <input type="number" id="progress" ng-model="progress
                        ">
224             </div>
225         </form>
226     </div>
227 </div>
228 </div>
229
230 <div class="footer row">
231     <p class="text-center">&copy; Michael U.B. Meding 2016</p>
```

```
232 </div>
233
234 </div>
235 </div>
236 </body>
237 </html>
```

C.3 visualizer/Worker.js

Listing C.3: Workerjs is the core gcode to D3 line processor.

```
1 /**
2  * User: hudbrog (hudbrog@gmail.com)
3  * Date: 10/24/12
4  * Time: 12:18 PM
5  */
6
7 var gcode;
8 var firstReport;
9 var z_heights = {};
10 var model = [];
11 var gCodeOptions = {
12     sortLayers: false,
13     purgeEmptyLayers: true,
14     analyzeModel: false
15 };
16 var max = {x: undefined, y: undefined, z: undefined};
17 var min = {x: undefined, y: undefined, z: undefined};
18 var modelSize = {x: undefined, y: undefined, z: undefined};
19 var filamentByLayer = {};
20 var filamentByExtruder = {};
21 var totalFilament = 0;
22 var printTime = 0;
23 var printTimeByLayer = {};
24 var layerHeight = 0;
25 var layerCnt = 0;
26 var speeds = {extrude: [], retract: [], move: []};
27 var speedsByLayer = {extrude: {}, retract: {}, move: {}};
28 var volSpeeds = [];
29 var volSpeedsByLayer = {};
30 var extrusionSpeeds = [];
31 var extrusionSpeedsByLayer = {};
32
33
34 var sendLayerToParent = function (layerNum, z, progress) {
35     self.postMessage({
36         "cmd": "returnLayer",
37         "msg": {
38             cmds: model[layerNum],
39             layerNum: layerNum,
40             zHeightObject: {zValue: z, layer: z_heights[z]},
41             isEmpty: false,
42             progress: progress
43         }
44     });
45 };
46
```

```
var sendMultiLayerToParent = function (layerNum, z, progress) {
    var tmpModel = [];
    var tmpZHeight = {};

    for (var i = 0; i < layerNum.length; i++) {
        tmpModel[layerNum[i]] = model[layerNum[i]];
        tmpZHeight[layerNum[i]] = z_heights[z[i]];
    }

    self.postMessage({
        "cmd": "returnMultiLayer",
        "msg": {
            model: tmpModel,
            layerNum: layerNum,
            zHeightObject: {zValue: z, layer: tmpZHeight},
            isEmpty: false,
            progress: progress
        }
    });
};


var sendSizeProgress = function (progress) {
    self.postMessage({
        "cmd": "analyzeProgress",
        "msg": {
            progress: progress,
            printTime: printTime
        }
    });
};

var sendAnalyzeDone = function () {
    self.postMessage({
        "cmd": "analyzeDone",
        "msg": {
            max: max,
            min: min,
            modelSize: modelSize,
            totalFilament: totalFilament,
            filamentByLayer: filamentByLayer,
            filamentByExtruder: filamentByExtruder,
            printTime: printTime,
            layerHeight: layerHeight,
            layerCnt: layerCnt,
            layerTotal: model.length,
            speeds: speeds,
            speedsByLayer: speedsByLayer,
            volSpeeds: volSpeeds,
            volSpeedsByLayer: volSpeedsByLayer,
            printTimeByLayer: printTimeByLayer,
            extrusionSpeeds: extrusionSpeeds,
            extrusionSpeedsByLayer: extrusionSpeedsByLayer
        }
    });
};

var purgeLayers = function () {
    var purge = true;
    for (var i = 0; i < model.length; i++) {
```

```
107         purge = true;
108         if (!model[i])purge = true;
109         else {
110             for (var j = 0; j < model[i].length; j++) {
111                 if (model[i][j].extrude)purge = false;
112             }
113         }
114         if (!purge) {
115             layerCnt += 1;
116         }
117     }
118 //         self.postMessage('LayerCnt: ' + layerCnt);
119 };
120
121
122 var analyzeModel = function () {
123     var i, j;
124     var x_ok = false, y_ok = false;
125     var cmds;
126     var tmp1 = 0, tmp2 = 0;
127     var speedIndex = 0;
128     var type;
129     var printTimeAdd = 0;
130 //         var moveTime=0;
131
132     for (i = 0; i < model.length; i++) {
133         cmds = model[i];
134         if (!cmds)continue;
135         for (j = 0; j < cmds.length; j++) {
136             x_ok = false;
137             y_ok = false;
138             if (typeof(cmds[j].x) !== 'undefined' && typeof(cmds[j
                ].prevX) !== 'undefined' && typeof(cmds[j].extrude
                ) !== 'undefined' && cmds[j].extrude && !isNaN(
                cmds[j].x)) {
139                 max.x = parseFloat(max.x) > parseFloat(cmds[j].x)
                    ? parseFloat(max.x) : parseFloat(cmds[j].x);
140                 max.x = parseFloat(max.x) > parseFloat(cmds[j].
                    prevX) ? parseFloat(max.x) : parseFloat(cmds[j
                    ].prevX);
141                 min.x = parseFloat(min.x) < parseFloat(cmds[j].x)
                    ? parseFloat(min.x) : parseFloat(cmds[j].x);
142                 min.x = parseFloat(min.x) < parseFloat(cmds[j].
                    prevX) ? parseFloat(min.x) : parseFloat(cmds[j
                    ].prevX);
143                 x_ok = true;
144             }
145
146             if (typeof(cmds[j].y) !== 'undefined' && typeof(cmds[j
                ].prevY) !== 'undefined' && typeof(cmds[j].extrude
                ) !== 'undefined' && cmds[j].extrude && !isNaN(
                cmds[j].y)) {
147                 max.y = parseFloat(max.y) > parseFloat(cmds[j].y)
                    ? parseFloat(max.y) : parseFloat(cmds[j].y);
148                 max.y = parseFloat(max.y) > parseFloat(cmds[j].
                    prevY) ? parseFloat(max.y) : parseFloat(cmds[j
                    ].prevY);
149                 min.y = parseFloat(min.y) < parseFloat(cmds[j].y)
                    ? parseFloat(min.y) : parseFloat(cmds[j].y);
```

```
150                 min.y = parseFloat(min.y) < parseFloat(cmds[j].
                        prevY) ? parseFloat(min.y) : parseFloat(cmds[j
                        ].prevY);
151                 y_ok = true;
152             }
153
154             if (typeof(cmds[j].prevZ) !== 'undefined' && typeof(
                    cmds[j].extrude) !== 'undefined' && cmds[j].
                    extrude && !isNaN(cmds[j].prevZ)) {
155                 max.z = parseFloat(max.z) > parseFloat(cmds[j].
                        prevZ) ? parseFloat(max.z) : parseFloat(cmds[j
                        ].prevZ);
156                 min.z = parseFloat(min.z) < parseFloat(cmds[j].
                        prevZ) ? parseFloat(min.z) : parseFloat(cmds[j
                        ].prevZ);
157             }
158
159             if ((typeof(cmds[j].extrude) !== 'undefined' && cmds[j
                    ].extrude == true) || cmds[j].retract != 0) {
160                 totalFilament += cmds[j].extrusion;
161                 if (!filamentByLayer[cmds[j].prevZ])
                        filamentByLayer[cmds[j].prevZ] = 0;
162                 filamentByLayer[cmds[j].prevZ] += cmds[j].
                        extrusion;
163                 if (cmds[j].extruder != null) {
164                     if (!filamentByExtruder[cmds[j].extruder])
                            filamentByExtruder[cmds[j].extruder] = 0;
165                     filamentByExtruder[cmds[j].extruder] += cmds[j
                            ].extrusion;
166                 }
167             }
168
169             if (x_ok && y_ok) {
170                 printTimeAdd = Math.sqrt(Math.pow(parseFloat(cmds[
                        j].x) - parseFloat(cmds[j].prevX), 2) + Math.
                        pow(parseFloat(cmds[j].y) - parseFloat(cmds[j
                        ].prevY), 2)) / (cmds[j].speed / 60);
171             } else if (cmds[j].retract === 0 && cmds[j].extrusion
                    !== 0) {
172                 tmp1 = Math.sqrt(Math.pow(parseFloat(cmds[j].x) -
                        parseFloat(cmds[j].prevX), 2) + Math.pow(
                        parseFloat(cmds[j].y) - parseFloat(cmds[j].
                        prevY), 2)) / (cmds[j].speed / 60);
173                 tmp2 = Math.abs(parseFloat(cmds[j].extrusion) / (
                        cmds[j].speed / 60));
174                 printTimeAdd = tmp1 >= tmp2 ? tmp1 : tmp2;
175             } else if (cmds[j].retract !== 0) {
176                 printTimeAdd = Math.abs(parseFloat(cmds[j].
                        extrusion) / (cmds[j].speed / 60));
177             }
178
179             printTime += printTimeAdd;
180             if (typeof(printTimeByLayer[cmds[j].prevZ]) === '
                    undefined') {
181                 printTimeByLayer[cmds[j].prevZ] = 0;
182             }
183             printTimeByLayer[cmds[j].prevZ] += printTimeAdd;
184
185             if (cmds[j].extrude && cmds[j].retract === 0) {
186                 type = 'extrude';
```

```
187                 } else if (cmds[j].retract !== 0) {
188                     type = 'retract';
189                 } else if (!cmds[j].extrude && cmds[j].retract === 0)
                        {
190                     type = 'move';
191                 } else {
192                     self.postMessage({cmd: 'unknown type of move'});
193                     type = 'unknown';
194                 }
195                 speedIndex = speeds[type].indexOf(cmds[j].speed);
196                 if (speedIndex === -1) {
197                     speeds[type].push(cmds[j].speed);
198                     speedIndex = speeds[type].indexOf(cmds[j].speed);
199                 }
200                 if (typeof(speedsByLayer[type][cmds[j].prevZ]) === '
                        undefined') {
201                     speedsByLayer[type][cmds[j].prevZ] = [];
202                 }
203                 if (speedsByLayer[type][cmds[j].prevZ].indexOf(cmds[j
                        ].speed) === -1) {
204                     speedsByLayer[type][cmds[j].prevZ][speedIndex] =
                            cmds[j].speed;
205                 }
206
207                 if (cmds[j].extrude && cmds[j].retract === 0 && x_ok
                        && y_ok) {
208                     // we are extruding
209                     var volPerMM = cmds[j].volPerMM;
210                     volPerMM = parseFloat(volPerMM).toFixed(3);
211                     var volIndex = volSpeeds.indexOf(volPerMM);
212                     if (volIndex === -1) {
213                         volSpeeds.push(volPerMM);
214                         volIndex = volSpeeds.indexOf(volPerMM);
215                     }
216                     if (typeof(volSpeedsByLayer[cmds[j].prevZ]) === '
                            undefined') {
217                         volSpeedsByLayer[cmds[j].prevZ] = [];
218                     }
219                     if (volSpeedsByLayer[cmds[j].prevZ].indexOf(
                            volPerMM) === -1) {
220                         volSpeedsByLayer[cmds[j].prevZ][volIndex] =
                                volPerMM;
221                     }
222
223                     var extrusionSpeed = cmds[j].volPerMM * (cmds[j].
                            speed / 60);
224                     extrusionSpeed = parseFloat(extrusionSpeed).
                            toFixed(3);
225                     var volIndex = extrusionSpeeds.indexOf(
                            extrusionSpeed);
226                     if (volIndex === -1) {
227                         extrusionSpeeds.push(extrusionSpeed);
228                         volIndex = extrusionSpeeds.indexOf(
                                extrusionSpeed);
229                     }
230                     if (typeof(extrusionSpeedsByLayer[cmds[j].prevZ])
                            === 'undefined') {
231                         extrusionSpeedsByLayer[cmds[j].prevZ] = [];
232                     }
```

```
233                 if (extrusionSpeedsByLayer[cmds[j].prevZ].indexOf(
                        extrusionSpeed) === -1) {
234                     extrusionSpeedsByLayer[cmds[j].prevZ][volIndex
                            ] = extrusionSpeed;
235                 }
236             }
237
238
239         }
240         sendSizeProgress(i / model.length * 100);
241
242     }
243     purgeLayers();
244
245     modelSize.x = Math.abs(max.x - min.x);
246     modelSize.y = Math.abs(max.y - min.y);
247     modelSize.z = Math.abs(max.z - min.z);
248     layerHeight = (max.z - min.z) / (layerCnt - 1);
249
250     sendAnalyzeDone();
251 };
252
253 var doParse = function () {
254     var argChar, numSlice;
255     model = [];
256     var sendLayer = undefined;
257     var sendLayerZ = 0;
258     var sendMultiLayer = [];
259     var sendMultiLayerZ = [];
260     var lastSend = 0;
261     //              console.time("parseGCode timer");
262     var reg = new RegExp(/^(?:G0|G1)\s/i);
263     var comment = new RegExp()
264     var j, layer = 0, extrude = false, prevRetract = {
265         e: 0,
266         a: 0,
267         b: 0,
268         c: 0
269     }, retract = 0, x, y, z = 0, f, prevZ = 0, prevX, prevY, lastF
            = 4000, prev_extrude = {
270         a: undefined,
271         b: undefined,
272         c: undefined,
273         e: undefined,
274         abs: undefined
275     }, extrudeRelative = false, volPerMM, extruder;
276     var dcExtrude = false;
277     var assumeNonDC = false;
278
279     for (var i = 0; i < gcode.length; i++) {
280         x = undefined;
281         y = undefined;
282         z = undefined;
283         volPerMM = undefined;
284         retract = 0;
285
286
287         extrude = false;
288         extruder = null;
289         prev_extrude["abs"] = 0;
```

```
290         gcode[i] = gcode[i].split(/[\(;]/)[0];
291
292         if (reg.test(gcode[i])) {
293             var args = gcode[i].split(/\s/);
294             for (j = 0; j < args.length; j++) {
295                 //                          console.log(args);
296                 //                          if(!args[j])continue;
297                 switch (argChar = args[j].charAt(0).toLowerCase())
                        {
298                     case 'x':
299                         x = args[j].slice(1);
300
301                         break;
302                     case 'y':
303                         y = args[j].slice(1);
304
305                         break;
306                     case 'z':
307                         z = args[j].slice(1);
308                         z = Number(z);
309                         if (z == prevZ)continue;
310                         if (z_heights.hasOwnProperty(z)) {
311                             layer = z_heights[z];
312                         } else {
313                             layer = model.length;
314                             z_heights[z] = layer;
315                         }
316                         sendLayer = layer;
317                         sendLayerZ = z;
318                         prevZ = z;
319                         break;
320                     case 'e':
321                     case 'a':
322                     case 'b':
323                     case 'c':
324                         assumeNonDC = true;
325                         extruder = argChar;
326                         numSlice = parseFloat(args[j].slice(1)).
                                toFixed(6);
327
328                         if (!extrudeRelative) {
329                             // absolute extrusion positioning
330                             prev_extrude["abs"] = parseFloat(
                                    numSlice) - parseFloat(
                                    prev_extrude[argChar]);
331
332                         } else {
333                             prev_extrude["abs"] = parseFloat(
                                    numSlice);
334                         }
335                         extrude = prev_extrude["abs"] > 0;
336                         if (prev_extrude["abs"] < 0) {
337                             prevRetract[extruder] = -1;
338                             retract = -1;
339                         }
340                         else if (prev_extrude["abs"] == 0) {
341                             //

                                    if(prevRetract <0 )prevRetract=
                                    retract;
```

```
342                             retract = 0;
343                         } else if (prev_extrude["abs"] > 0 &&
                                prevRetract[extruder] < 0) {
344                             prevRetract[extruder] = 0;
345                             retract = 1;
346                         } else {
347                             //

                                    prevRetract = retract;
348                             retract = 0;
349                         }
350                         prev_extrude[argChar] = numSlice;
351
352                         break;
353                     case 'f':
354                         numSlice = args[j].slice(1);
355                         lastF = numSlice;
356                         break;
357                     default:
358                         break;
359                 }
360             }
361             if (dcExtrude && !assumeNonDC) {
362                 extrude = true;
363                 prev_extrude["abs"] = Math.sqrt((prevX - x) * (
                        prevX - x) + (prevY - y) * (prevY - y));
364             }
365             if (extrude && retract == 0) {
366                 volPerMM = Number(prev_extrude['abs'] / Math.sqrt
                        ((prevX - x) * (prevX - x) + (prevY - y) * (
                        prevY - y)));
367             }
368             if (!model[layer])model[layer] = [];
369             //if(typeof(x) !== 'undefined' || typeof(y) !== '
                    undefined' ||typeof(z) !== 'undefined'||retract
                    !=0)
370             model[layer][model[layer].length] = {
371                 x: Number(x),
372                 y: Number(y),
373                 z: Number(z),
374                 extrude: extrude,
375                 retract: Number(retract),
376                 noMove: false,
377                 extrusion: (extrude || retract) ? Number(
                        prev_extrude["abs"]) : 0,
378                 extruder: extruder,
379                 prevX: Number(prevX),
380                 prevY: Number(prevY),
381                 prevZ: Number(prevZ),
382                 speed: Number(lastF),
383                 gcodeLine: Number(i),
384                 volPerMM: typeof(volPerMM) === 'undefined' ? -1 :
                        volPerMM
385             };
386             //{x: x, y: y, z: z, extrude: extrude, retract:
                    retract, noMove: false, extrusion: (extrude||
                    retract)?prev_extrude["abs"]:0, prevX: prevX,
                    prevY: prevY, prevZ: prevZ, speed: lastF,
                    gcodeLine: i};
387             if (typeof(x) !== 'undefined') prevX = x;
```

```
388                 if (typeof(y) !== 'undefined') prevY = y;
389             } else if (gcode[i].match(/^(?:M82)/i)) {
390                 extrudeRelative = false;
391             } else if (gcode[i].match(/^(?:G91)/i)) {
392                 extrudeRelative = true;
393             } else if (gcode[i].match(/^(?:G90)/i)) {
394                 extrudeRelative = false;
395             } else if (gcode[i].match(/^(?:M83)/i)) {
396                 extrudeRelative = true;
397             } else if (gcode[i].match(/^(?:M101)/i)) {
398                 dcExtrude = true;
399             } else if (gcode[i].match(/^(?:M103)/i)) {
400                 dcExtrude = false;
401             } else if (gcode[i].match(/^(?:G92)/i)) {
402                 var args = gcode[i].split(/\s/);
403                 for (j = 0; j < args.length; j++) {
404                     switch (argChar = args[j].charAt(0).toLowerCase())
                        {
405                         case 'x':
406                             x = args[j].slice(1);
407                             break;
408                         case 'y':
409                             y = args[j].slice(1);
410                             break;
411                         case 'z':
412                             z = args[j].slice(1);
413                             prevZ = z;
414                             break;
415                         case 'e':
416                         case 'a':
417                         case 'b':
418                         case 'c':
419                             numSlice = parseFloat(args[j].slice(1)).
                                    toFixed(3);
420                             extruder = argChar;
421                             if (!extrudeRelative)
422                                 prev_extrude[argChar] = 0;
423                             else {
424                                 prev_extrude[argChar] = numSlice;
425                             }
426 //                                prevZ = z;
427                             break;
428                         default:
429                             break;
430                     }
431                 }
432                 if (!model[layer])model[layer] = [];
433                 if (typeof(x) !== 'undefined' || typeof(y) !== '
                        undefined' || typeof(z) !== 'undefined')
434                     model[layer][model[layer].length] = {
435                         x: parseFloat(x),
436                         y: parseFloat(y),
437                         z: parseFloat(z),
438                         extrude: extrude,
439                         retract: parseFloat(retract),
440                         noMove: true,
441                         extrusion: 0,
442                         extruder: extruder,
443                         prevX: parseFloat(prevX),
444                         prevY: parseFloat(prevY),
```

```
                    prevZ: parseFloat(prevZ),
                    speed: parseFloat(lastF),
                    gcodeLine: parseFloat(i)
                };
        } else if (gcode[i].match(/^(?:G28)/i)) {
            var args = gcode[i].split(/\s/);
            for (j = 0; j < args.length; j++) {
                switch (argChar = args[j].charAt(0).toLowerCase())
                    {
                    case 'x':
                        x = args[j].slice(1);
                        break;
                    case 'y':
                        y = args[j].slice(1);
                        break;
                    case 'z':
                        z = args[j].slice(1);
                        z = Number(z);
                        if (z === prevZ)continue;
                        sendLayer = layer;
                        sendLayerZ = z;//}
                        if (z_heights.hasOwnProperty(z)) {
                            layer = z_heights[z];
                        } else {
                            layer = model.length;
                            z_heights[z] = layer;
                        }
                        prevZ = z;
                        break;
                    default:
                        break;
                }
            }
            // G28 with no arguments
            if (args.length == 1) {
                //need to init values to default here
            }
            // if it's the first layer and G28 was without
            if (layer == 0 && typeof(z) === 'undefined') {
                z = 0;
                if (z_heights.hasOwnProperty(z)) {
                    layer = z_heights[z];
                } else {
                    layer = model.length;
                    z_heights[z] = layer;
                }
                prevZ = z;
            }
//                  x=0, y=0,z=0,prevZ=0, extrude=false;
//                  if(typeof(prevX) === 'undefined'){prevX=0;}
//                  if(typeof(prevY) === 'undefined'){prevY=0;}

            if (!model[layer])model[layer] = [];
//                  if(typeof(x) !== 'undefined' || typeof(y) !== '
    undefined' ||typeof(z) !== 'undefined'||retract!=0)
            model[layer][model[layer].length] = {
                x: Number(x),
                y: Number(y),
                z: Number(z),
                extrude: extrude,
```

```
503                 retract: Number(retract),
504                 noMove: false,
505                 extrusion: (extrude || retract) ? Number(
                        prev_extrude["abs"]) : 0,
506                 extruder: extruder,
507                 prevX: Number(prevX),
508                 prevY: Number(prevY),
509                 prevZ: Number(prevZ),
510                 speed: Number(lastF),
511                 gcodeLine: Number(i)
512             };
513 //                  if(typeof(x) !== 'undefined' || typeof(y) !== '
    undefined' ||typeof(z) !== 'undefined') model[layer][model[
    layer].length] = {x: x, y: y, z: z, extrude: extrude, retract:
     retract, noMove:false, extrusion: (extrude||retract)?
    prev_extrude["abs"]:0, prevX: prevX, prevY: prevY, prevZ:
    prevZ, speed: lastF, gcodeLine: parseFloat(i)};
514         }
515         if (typeof(sendLayer) !== "undefined") {
516 //                  sendLayerToParent(sendLayer, sendLayerZ, i/gcode
    .length*100);
517 //                  sendLayer = undefined;
518
519             if (i - lastSend > gcode.length * 0.02 &&
                    sendMultiLayer.length != 0) {
520                 lastSend = i;
521                 sendMultiLayerToParent(sendMultiLayer,
                        sendMultiLayerZ, i / gcode.length * 100);
522                 sendMultiLayer = [];
523                 sendMultiLayerZ = [];
524             }
525             sendMultiLayer[sendMultiLayer.length] = sendLayer;
526             sendMultiLayerZ[sendMultiLayerZ.length] = sendLayerZ;
527             sendLayer = undefined;
528             sendLayerZ = undefined;
529         }
530     }
531
532     sendMultiLayerToParent(sendMultiLayer, sendMultiLayerZ, i /
            gcode.length * 100);
533 };
534
535 var parseGCode = function (message) {
536     gcode = message.gcode;
537     firstReport = message.options.firstReport;
538
539
540     doParse();
541     gcode = [];
542     self.postMessage({
543         "cmd": "returnModel",
544         "msg": {
545             model: model
546         }
547     });
548
549 };
550
551 var runAnalyze = function (message) {
552     analyzeModel();
```

```
    model = [];
    z_heights = [];
    gcode = undefined;
    firstReport = undefined;
    z_heights = {};
    model = [];
    max = {x: undefined, y: undefined, z: undefined};
    min = {x: undefined, y: undefined, z: undefined};
    modelSize = {x: undefined, y: undefined, z: undefined};
    filamentByLayer = {};
    filamentByExtruder = {};
    totalFilament = 0;
    printTime = 0;
    printTimeByLayer = {};
    layerHeight = 0;
    layerCnt = 0;
    speeds = {extrude: [], retract: [], move: []};
    speedsByLayer = {extrude: {}, retract: {}, move: {}};
};
var setOption = function (options) {
    for (var opt in options) {
        gCodeOptions[opt] = options[opt];
    }
};

onmessage = function (e) {
    var data = e.data;
    // for some reason firefox doesn't garbage collect when
       something inside closures is deleted, so we delete and
       recreate whole object each time
    switch (data.cmd) {
        case 'parseGCode':
            parseGCode(data.msg);
            break;
        case 'setOption':
            setOption(data.msg);
            break;
        case 'analyzeModel':
            runAnalyze(data.msg);
            break;

        default:
            self.postMessage('Unknown command: ' + data.msg);
    }

};
```

C.4 visualizer/gcodeFactory.js

Listing C.4: gcodeFactory.js is the main gcode header parsing code.

```
/**
 * Originally written by: hudbrog (hudbrog@gmail.com)
 * Angularized by: Mike Meding (mike@mikemeding.com)
 */

var testApp = angular.module('WebSlicer');

```

```
8 testApp.factory('gcodeFactory', function () {
9     // ***** PRIVATE ******
10    var gcode, lines;
11    var z_heights = {};
12    var model = [];
13    var max = {x: undefined, y: undefined, z: undefined};
14    var min = {x: undefined, y: undefined, z: undefined};
15    var modelSize = {x: undefined, y: undefined, z: undefined};
16    var filamentByLayer = {};
17    var filamentByExtruder = {};
18    var printTimeByLayer;
19    var totalFilament = 0;
20    var printTime = 0;
21    var totalWeight = 0;
22    var layerHeight = 0;
23    var layerCnt = 0;
24    var layerTotal = 0;
25    var speeds = {};
26    var slicer = 'unknown';
27    var speedsByLayer = {};
28    var volSpeeds = {};
29    var volSpeedsByLayer = {};
30    var extrusionSpeeds = {};
31    var extrusionSpeedsByLayer = {};
32    var gCodeOptions = {
33        sortLayers: false,
34        purgeEmptyLayers: true,
35        analyzeModel: false,
36        filamentType: "ABS",
37        filamentDia: 1.75,
38        nozzleDia: 0.4
39    };
40
41    var prepareGCode = function () {
42        if (!lines)return;
43        gcode = [];
44        var i;
45        for (i = 0; i < lines.length; i++) {
46            if (lines[i].match(/^(G0|G1|G90|G91|G92|M82|M83|G28)/i
              ))gcode.push(lines[i]);
47        }
48        lines = [];
49 //         console.log("GCode prepared");
50    };
51
52    var sortLayers = function () {
53        var sortedZ = [];
54        var tmpModel = [];
55 //         var cnt = 0;
56        for (var layer in z_heights) {
57            sortedZ[z_heights[layer]] = layer;
58 //             cnt++;
59        }
60        sortedZ.sort(function (a, b) {
61            return a - b;
62        });
63        for (var i = 0; i < sortedZ.length; i++) {
64 //             console.log("i is " + i +" and sortedZ[i] is " +
    sortedZ[i] + "and z_heights[] is " + z_heights[sortedZ[i]] );
```

```
            if (typeof(z_heights[sortedZ[i]]) === 'undefined') continue;
            tmpModel[i] = model[z_heights[sortedZ[i]]];
        }
        model = tmpModel;
        delete tmpModel;
    };

    var purgeLayers = function () {
        var purge = true;
        if (!model) {
            console.log("Something terribly wrong just happened.");
            return;
        }
        for (var i = 0; i < model.length; i++) {
            purge = true;
            if (typeof(model[i]) === 'undefined')purge = true;
            else {
                for (var j = 0; j < model[i].length; j++) {
                    if (model[i][j].extrude)purge = false;
                }
            }
            if (purge) {
                model.splice(i, 1);
                i--;
            }
        }
    };

    var getParamsFromKISSlicer = function (gcode) {
        var nozzle = gcode.match(/extrusion_width_mm\s*=\s*(\d*\.\d+)/m);
        if (nozzle) {
            gCodeOptions['nozzleDia'] = nozzle[1];
        }
        var filament = gcode.match(/fiber_dia_mm\s*=\s*(\d*\.\d+)/m);
        if (filament) {
            gCodeOptions['filamentDia'] = filament[1];
        }
    }

    var getParamsFromSlic3r = function (gcode) {
        var nozzle = gcode.match(/nozzle_diameter\s*=\s*(\d*\.\d+)/m);
        if (nozzle) {
            gCodeOptions['nozzleDia'] = nozzle[1];
        }
        var filament = gcode.match(/filament_diameter\s*=\s*(\d*\.\d+)/m);
        if (filament) {
            gCodeOptions['filamentDia'] = filament[1];
        }
    }

    var getParamsFromSkeinforge = function (gcode) {

        var nozzle = gcode.match(/nozzle_diameter\s*=\s*(\d*\.\d+)/m);
```

```
118         if (nozzle) {
119             gCodeOptions['nozzleDia'] = nozzle[1];
120         }
121         var filament = gcode.match(/Filament_Diameter_(mm)\s*:\s
                *(\d*\.\d+)/m);
122         if (filament) {
123             gCodeOptions['filamentDia'] = filament[1];
124         }
125     }
126
127     var getParamsFromMiracleGrue = function (gcode) {
128
129     }
130
131     var getParamsFromCura = function (gcode) {
132 //          console.log("cura");
133         var profileString = gcode.match(/CURA_PROFILE_STRING:((?:[
                A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0
                -9+/]{3}=|[A-Za-z0-9+/]{4}))/m);
134         if (profileString) {
135             var raw = window.atob(profileString[1]);
136             var array = new Uint8Array(new ArrayBuffer(raw.length)
                    );
137
138             for (i = 0; i < raw.length; i++) {
139                 array[i] = raw.charCodeAt(i);
140             }
141             var data = new Zlib.inflate(array.subarray(2, array.
                    byteLength - 4));
142             var msg;
143             for (i = 0; i < data.length; i += 1) {
144                 msg += String.fromCharCode(data[i]);
145             }
146             var nozzle = msg.match(/nozzle_size\s*=\s*(\d*\.\d+)/m
                    );
147             if (nozzle) {
148                 gCodeOptions['nozzleDia'] = nozzle[1];
149             }
150             var filament = msg.match(/filament_diameter\s*=\s*(\d
                    *\.\d+)/m);
151             if (filament) {
152                 gCodeOptions['filamentDia'] = filament[1];
153             }
154
155         }
156     }
157
158     var detectSlicer = function (gcode) {
159
160         if (gcode.match(/Slic3r/)) {
161             slicer = 'Slic3r';
162             getParamsFromSlic3r(gcode);
163         } else if (gcode.match(/KISSlicer/)) {
164             slicer = 'KISSlicer';
165             getParamsFromKISSlicer(gcode);
166         } else if (gcode.match(/skeinforge/)) {
167             slicer = 'skeinforge';
168             getParamsFromSkeinforge(gcode);
169         } else if (gcode.match(/CURA_PROFILE_STRING/)) {
170             slicer = 'cura';
```

```
171             getParamsFromCura(gcode);
172         } else if (gcode.match(/Miracle/)) {
173             slicer = 'makerbot';
174             getParamsFromMiracleGrue(gcode);
175         }
176
177     }
178
179
180 // ***** PUBLIC *******
181     return {
182
183         loadString: function (gcode) {
184             model = [];
185             z_heights = [];
186             detectSlicer(gcode);
187         },
188         setModel: function (newModel) {
189             model = newModel;
190         },
191         getModel: function () {
192             return model;
193         },
194         setOption: function (options) {
195             for (var opt in options) {
196                 gCodeOptions[opt] = options[opt];
197             }
198         },
199         processLayerFromWorker: function (msg) {
200             model[msg.layerNum] = msg.cmds;
201             z_heights[msg.zHeightObject.zValue] = msg.
                    zHeightObject.layer;
202         },
203         processMultiLayerFromWorker: function (msg) {
204             for (var i = 0; i < msg.layerNum.length; i++) {
205                 model[msg.layerNum[i]] = msg.model[msg.layerNum[i
                        ]];
206                 z_heights[msg.zHeightObject.zValue[i]] = msg.
                        layerNum[i];
207             }
208 //              console.log(model);
209         },
210         processAnalyzeModelDone: function (msg) {
211             min = msg.min;
212             max = msg.max;
213             modelSize = msg.modelSize;
214             totalFilament = msg.totalFilament;
215             filamentByLayer = msg.filamentByLayer;
216             filamentByExtruder = msg.filamentByExtruder;
217             speeds = msg.speeds;
218             speedsByLayer = msg.speedsByLayer;
219             printTime = msg.printTime;
220             printTimeByLayer = msg.printTimeByLayer;
221             layerHeight = msg.layerHeight;
222             layerCnt = msg.layerCnt;
223             layerTotal = msg.layerTotal;
224             volSpeeds = msg.volSpeeds;
225             volSpeedsByLayer = msg.volSpeedsByLayer;
226             extrusionSpeeds = msg.extrusionSpeeds;
227             extrusionSpeedsByLayer = msg.extrusionSpeedsByLayer;
```

```
228
229             var density = 1;
230             if (gCodeOptions['filamentType'] === 'ABS') {
231                 density = 1.04;
232             } else if (gCodeOptions['filamentType'] === 'PLA') {
233                 density = 1.24;
234             }
235             totalWeight = density * 3.141 * gCodeOptions['
                    filamentDia'] / 10 * gCodeOptions['filamentDia'] /
                     10 / 4 * totalFilament / 10;
236
237             gCodeOptions['wh'] = parseFloat(gCodeOptions['
                    nozzleDia']) / parseFloat(layerHeight);
238             if (slicer === 'Slic3r' || slicer === 'cura') {
239                 // slic3r stores actual nozzle diameter, but
                        extrusion is usually slightly thicker, here we
                         compensate for that
240                 // kisslicer stores actual extrusion width - so
                        no need for that.
241                 gCodeOptions['wh'] = gCodeOptions['wh'] * 1.1;
242             }
243         },
244         getLayerFilament: function (z) {
245             return filamentByLayer[z];
246         },
247         getLayerSpeeds: function (z) {
248             return speedsByLayer[z] ? speedsByLayer[z] : {};
249         },
250         getModelInfo: function () {
251             return {
252                 min: min,
253                 max: max,
254                 modelSize: modelSize,
255                 totalFilament: totalFilament,
256                 filamentByExtruder: filamentByExtruder,
257                 speeds: speeds,
258                 speedsByLayer: speedsByLayer,
259                 printTime: printTime,
260                 printTimeByLayer: printTimeByLayer,
261                 totalWeight: totalWeight,
262                 layerHeight: layerHeight,
263                 layerCnt: layerCnt,
264                 layerTotal: layerTotal,
265                 volSpeeds: volSpeeds,
266                 volSpeedsByLayer: volSpeedsByLayer,
267                 extrusionSpeeds: extrusionSpeeds,
268                 extrusionSpeedsByLayer: extrusionSpeedsByLayer
269             };
270         },
271         getGCodeLines: function (layer, fromSegments, toSegments)
                {
272             var i = 0;
273             var result = {first: model[layer][fromSegments].
                    gcodeLine, last: model[layer][toSegments].
                    gcodeLine};
274             return result;
275         },
276         getOptions: function () {
277             return gCodeOptions;
278         }
```

```
    }
});
```

C.5 visualizer/renderFactory.js

Listing C.5: When render frames are requested this file compiles the lines of gcode from the worker and sets up the rest of the canvas frames.

```
/**
 * Originally written by: hudbrog (hudbrog@gmail.com)
 * Angularized by: Mike Meding (mike@mikemeding.com)
 */

var testApp = angular.module('WebSlicer');

testApp.factory('renderFactory', function (gcodeFactory) {

    // ***** PRIVATE ******
    var canvas;
    var ctx;
    var zoomFactor = 3, zoomFactorDelta = 0.4;
    var gridSizeX = 200, gridSizeY = 200, gridStep = 10;
    var ctxHeight, ctxWidth;
    var prevX = 0, prevY = 0;

//    var colorGrid="#bbbbbb", colorLine="#000000";
    var sliderHor, sliderVer;
    var layerNumStore, progressStore = {from: 0, to: -1};
    var lastX, lastY;
    var dragStart, dragged;
    var scaleFactor = 1.1;
    var model;
    var initialized = false;
    var displayType = {speed: 1, expermm: 2, volpersec: 3};
    var renderOptions = {
        showMoves: true,
        showRetracts: true,
        colorGrid: "#bbbbbb",
        extrusionWidth: 1,
//        colorLine: ["#000000", "#aabb88", "#ffe7a0", "#6e7700", "#331a00", "#44ba97", "#08262f", "#db0e00", "#ff9977"],
        colorLine: ["#000000", "#45c7ba", "#a9533a", "#ff44cc", "#dd1177", "#eeee22", "#ffbb55", "#ff5511", "#777788", "#ff0000", "#ffff00"],
        colorLineLen: 9,
        colorMove: "#00ff00",
        colorRetract: "#ff0000",
        colorRestart: "#0000ff",
        sizeRetractSpot: 2,
        modelCenter: {x: 0, y: 0},
        moveModel: true,
        differentiateColors: true,
        showNextLayer: false,
        alpha: false,
        actualWidth: false,
        renderErrors: false,
```

```
46          renderAnalysis: false,
47          speedDisplayType: displayType.speed
48      };
49
50      var offsetModelX = 0, offsetModelY = 0;
51      var speeds = [];
52      var speedsByLayer = {};
53      var volSpeeds = [];
54      var volSpeedsByLayer = {};
55      var extrusionSpeeds = [];
56      var extrusionSpeedsByLayer = {};
57
58
59      var reRender = function () {
60          var gCodeOpts = gcodeFactory.getOptions();
61          var p1 = ctx.transformedPoint(0, 0);
62          var p2 = ctx.transformedPoint(canvas.width, canvas.height)
                ;
63          ctx.clearRect(p1.x, p1.y, p2.x - p1.x, p2.y - p1.y);
64          drawGrid();
65          if (renderOptions['alpha']) {
66              ctx.globalAlpha = 0.6;
67          }
68          else {
69              ctx.globalAlpha = 1;
70          }
71          if (renderOptions['actualWidth']) {
72              renderOptions['extrusionWidth'] = gCodeOpts['
                    filamentDia'] * gCodeOpts['wh'] / zoomFactor;
73          }
74          else {
75              renderOptions['extrusionWidth'] = gCodeOpts['
                    filamentDia'] * gCodeOpts['wh'] / zoomFactor / 2;
76          }
77          if (renderOptions['showNextLayer'] && layerNumStore <
                model.length - 1) {
78              drawLayer(layerNumStore + 1, 0, this.
                    getLayerNumSegments(layerNumStore + 1), true);
79          }
80          drawLayer(layerNumStore, progressStore.from, progressStore
                .to);
81      };
82
83      function trackTransforms(ctx) {
84          var svg = document.createElementNS("http://www.w3.org
                /2000/svg", 'svg');
85          var xform = svg.createSVGMatrix();
86          ctx.getTransform = function () {
87              return xform;
88          };
89
90          var savedTransforms = [];
91          var save = ctx.save;
92          ctx.save = function () {
93              savedTransforms.push(xform.translate(0, 0));
94              return save.call(ctx);
95          };
96          var restore = ctx.restore;
97          ctx.restore = function () {
98              xform = savedTransforms.pop();
```

```
99              return restore.call(ctx);
100         };
101
102         var scale = ctx.scale;
103         ctx.scale = function (sx, sy) {
104             xform = xform.scaleNonUniform(sx, sy);
105             return scale.call(ctx, sx, sy);
106         };
107         var rotate = ctx.rotate;
108         ctx.rotate = function (radians) {
109             xform = xform.rotate(radians * 180 / Math.PI);
110             return rotate.call(ctx, radians);
111         };
112         var translate = ctx.translate;
113         ctx.translate = function (dx, dy) {
114             xform = xform.translate(dx, dy);
115             return translate.call(ctx, dx, dy);
116         };
117         var transform = ctx.transform;
118         ctx.transform = function (a, b, c, d, e, f) {
119             var m2 = svg.createSVGMatrix();
120             m2.a = a;
121             m2.b = b;
122             m2.c = c;
123             m2.d = d;
124             m2.e = e;
125             m2.f = f;
126             xform = xform.multiply(m2);
127             return transform.call(ctx, a, b, c, d, e, f);
128         };
129         var setTransform = ctx.setTransform;
130         ctx.setTransform = function (a, b, c, d, e, f) {
131             xform.a = a;
132             xform.b = b;
133             xform.c = c;
134             xform.d = d;
135             xform.e = e;
136             xform.f = f;
137             return setTransform.call(ctx, a, b, c, d, e, f);
138         };
139         var pt = svg.createSVGPoint();
140         ctx.transformedPoint = function (x, y) {
141             pt.x = x;
142             pt.y = y;
143             return pt.matrixTransform(xform.inverse());
144         }
145     }
146
147
148     var startCanvas = function () {
149         canvas = document.getElementById('canvas');
150         if (!canvas.getContext) {
151             throw "exception";
152         }
153
154         ctx = canvas.getContext('2d');
155         ctxHeight = canvas.height;
156         ctxWidth = canvas.width;
157         lastX = ctxWidth / 2;
158         lastY = ctxHeight / 2;
```

```
159         ctx.lineWidth = 2;
160         ctx.lineCap = 'round';
161         trackTransforms(ctx);
162
163         canvas.addEventListener('mousedown', function (evt) {
164             document.body.style.mozUserSelect = document.body.
                    style.webkitUserSelect = document.body.style.
                    userSelect = 'none';
165             lastX = evt.offsetX || (evt.pageX - canvas.offsetLeft)
                    ;
166             lastY = evt.offsetY || (evt.pageY - canvas.offsetTop);
167             dragStart = ctx.transformedPoint(lastX, lastY);
168             dragged = false;
169         }, false);
170         canvas.addEventListener('mousemove', function (evt) {
171             lastX = evt.offsetX || (evt.pageX - canvas.offsetLeft)
                    ;
172             lastY = evt.offsetY || (evt.pageY - canvas.offsetTop);
173             dragged = true;
174             if (dragStart) {
175                 var pt = ctx.transformedPoint(lastX, lastY);
176                 ctx.translate(pt.x - dragStart.x, pt.y - dragStart
                        .y);
177                 reRender();
178             }
179         }, false);
180         canvas.addEventListener('mouseup', function (evt) {
181             dragStart = null;
182             if (!dragged) zoom(evt.shiftKey ? -1 : 1);
183         }, false);
184         var zoom = function (clicks) {
185             var pt = ctx.transformedPoint(lastX, lastY);
186             ctx.translate(pt.x, pt.y);
187             var factor = Math.pow(scaleFactor, clicks);
188             ctx.scale(factor, factor);
189             ctx.translate(-pt.x, -pt.y);
190             reRender();
191         };
192         var handleScroll = function (evt) {
193             var delta;
194             if (evt.detail < 0 || evt.wheelDelta > 0)delta =
                    zoomFactorDelta;
195             else delta = -1 * zoomFactorDelta;
196             if (delta) zoom(delta);
197             return evt.preventDefault() && false;
198         };
199         canvas.addEventListener('DOMMouseScroll', handleScroll,
                false);
200         canvas.addEventListener('mousewheel', handleScroll, false)
                ;
201
202     };
203
204     var drawGrid = function () {
205         var i;
206         ctx.strokeStyle = renderOptions["colorGrid"];
207         ctx.lineWidth = 1;
208         var offsetX = 0, offsetY = 0;
209         if (renderOptions["moveModel"]) {
210             offsetX = offsetModelX;
```

```
211             offsetY = offsetModelY;
212         }
213
214         ctx.beginPath();
215         for (i = 0; i <= gridSizeX; i += gridStep) {
216             ctx.moveTo(i * zoomFactor - offsetX, 0 - offsetY);
217             ctx.lineTo(i * zoomFactor - offsetX, -gridSizeY *
                    zoomFactor - offsetY);
218         }
219         ctx.stroke();
220
221         ctx.beginPath();
222         for (i = 0; i <= gridSizeY; i += gridStep) {
223             ctx.moveTo(0 - offsetX, -i * zoomFactor - offsetY);
224             ctx.lineTo(gridSizeX * zoomFactor - offsetX, -i *
                    zoomFactor - offsetY);
225         }
226         ctx.stroke();
227
228     };
229
230     var drawLayer = function (layerNum, fromProgress, toProgress,
            isNextLayer) {
231         var i, speedIndex = 0, prevZ = 0;
232         isNextLayer = typeof isNextLayer !== 'undefined' ?
                isNextLayer : false;
233         if (!isNextLayer) {
234             layerNumStore = layerNum;
235             progressStore = {from: fromProgress, to: toProgress};
236         }
237         if (!model || !model[layerNum])return;
238
239         var cmds = model[layerNum];
240         var x, y;
241
242         if (fromProgress > 0) {
243             prevX = cmds[fromProgress - 1].x * zoomFactor;
244             prevY = -cmds[fromProgress - 1].y * zoomFactor;
245         } else if (fromProgress === 0 && layerNum == 0) {
246             if (model[0] && typeof(model[0].x) !== 'undefined' &&
                    typeof(model[0].y) !== 'undefined') {
247                 prevX = model[0].x * zoomFactor;
248                 prevY = -model[0].y * zoomFactor;
249             } else {
250                 prevX = 0;
251                 prevY = 0;
252             }
253         } else if (typeof(cmds[0].prevX) !== 'undefined' && typeof
                (cmds[0].prevY) !== 'undefined') {
254             prevX = cmds[0].prevX * zoomFactor;
255             prevY = -cmds[0].prevY * zoomFactor;
256         } else {
257             if (model[layerNum - 1]) {
258                 prevX = undefined;
259                 prevY = undefined;
260                 for (i = model[layerNum - 1].length - 1; i >= 0; i
                        --) {
261                     if (typeof(prevX) === 'undefined' && model[
                            layerNum - 1][i].x !== undefined)prevX =
                            model[layerNum - 1][i].x * zoomFactor;
```

```
                    if (typeof(prevY) === 'undefined' && model[layerNum - 1][i].y !== undefined)prevY = -model[layerNum - 1][i].y * zoomFactor;
                }
                if (typeof(prevX) === 'undefined')prevX = 0;
                if (typeof(prevY) === 'undefined')prevY = 0;
            } else {
                prevX = 0;
                prevY = 0;
            }
        }

        prevZ = publicFunctions.getZ(layerNum);

        for (i = fromProgress; i <= toProgress; i++) {
            ctx.lineWidth = 1;

            if (typeof(cmds[i]) === 'undefined')continue;

            if (typeof(cmds[i].prevX) !== 'undefined' && typeof(cmds[i].prevY) !== 'undefined') {
                prevX = cmds[i].prevX * zoomFactor;
                prevY = -cmds[i].prevY * zoomFactor;
            }

            if (typeof(cmds[i].x) === 'undefined' || isNaN(cmds[i].x))x = prevX / zoomFactor;
            else x = cmds[i].x;
            if (typeof(cmds[i].y) === 'undefined' || isNaN(cmds[i].y))y = prevY / zoomFactor;
            else y = -cmds[i].y;
            if (renderOptions["differentiateColors"] && !renderOptions['showNextLayer'] && !renderOptions['renderAnalysis']) {

                if (renderOptions['speedDisplayType'] === displayType.speed) {
                    speedIndex = speeds['extrude'].indexOf(cmds[i].speed);
                } else if (renderOptions['speedDisplayType'] === displayType.expermm) {
                    speedIndex = volSpeeds.indexOf(cmds[i].volPerMM);
                } else if (renderOptions['speedDisplayType'] === displayType.volpersec) {
                    speedIndex = extrusionSpeeds.indexOf((cmds[i].volPerMM * cmds[i].speed / 60).toFixed(3));
                } else {
                    speedIndex = 0;
                }

                if (speedIndex === -1) {
                    speedIndex = 0;
                } else if (speedIndex > renderOptions["colorLineLen"] - 1) {
                    speedIndex = speedIndex % (renderOptions["colorLineLen"] - 1);
                }
```

```
305             } else if (renderOptions['showNextLayer'] &&
                    isNextLayer) {
306                 speedIndex = 3;
307             } else if (renderOptions['renderErrors']) {
308                 if (cmds[i].errType === 2) {
309                     speedIndex = 9;
310 //                    console.log("l: " + layerNum + " c: " + i);
311                 } else if (cmds[i].errType === 1) {
312                     speedIndex = 10;
313                 } else {
314                     speedIndex = 0;
315                 }
316             } else if (renderOptions['renderAnalysis']) {
317                 if (layerNum !== 0)speedIndex = -1;
318                 else speedIndex = 0;
319             } else {
320                 speedIndex = 0;
321             }
322
323
324             if (!cmds[i].extrude && !cmds[i].noMove) {
325 //                ctx.stroke();
326                 if (cmds[i].retract == -1) {
327                     if (renderOptions["showRetracts"]) {
328
329                         ctx.strokeStyle = renderOptions["
                                colorRetract"];
330                         ctx.fillStyle = renderOptions["
                                colorRetract"];
331                         ctx.beginPath();
332                         ctx.arc(prevX, prevY, renderOptions["
                                sizeRetractSpot"], 0, Math.PI * 2,
                                true);
333                         ctx.stroke();
334                         ctx.fill();
335                     }
336                 }
337                 if (renderOptions["showMoves"]) {
338                     ctx.strokeStyle = renderOptions["colorMove"];
339                     ctx.beginPath();
340                     ctx.moveTo(prevX, prevY);
341                     ctx.lineTo(x * zoomFactor, y * zoomFactor);
342                     ctx.stroke();
343                 }
344             }
345             else if (cmds[i].extrude) {
346                 if (cmds[i].retract == 0) {
347                     if (speedIndex >= 0) {
348                         ctx.strokeStyle = renderOptions["colorLine
                                "][speedIndex];
349                     } else if (speedIndex === -1) {
350                         var val = parseInt(cmds[i].errLevelB).
                                toString(16);
351                         var crB = "#" + "00".substr(0, 2 - val.
                                length) + val + '0000';
352                         val = parseInt(cmds[i].errLevelE).toString
                                (16);
353                         var crE = "#" + "00".substr(0, 2 - val.
                                length) + val + '0000';
```

```
354                         var gradient = ctx.createLinearGradient(
                                prevX, prevY, x * zoomFactor, y *
                                zoomFactor);
355                         if (cmds[i].errType === 1) {
356                             var limit = (1 - cmds[i].errDelimiter)
                                    ;
357                             if (limit >= 0.99) limit = 0.99;
358                             gradient.addColorStop(0, "#000000");
359                             gradient.addColorStop(limit, "#000000"
                                    );
360                             gradient.addColorStop(limit + 0.01,
                                    crE);
361                             gradient.addColorStop(1, crE);
362                         } else if (cmds[i].errType === 2) {
363                             gradient.addColorStop(0, crB);
364                             var limit = cmds[i].errDelimiter;
365                             if (limit >= 0.99) limit = 0.99;
366                             gradient.addColorStop(limit, crB);
367                             gradient.addColorStop(limit + 0.01, "
                                    #000000");
368                             gradient.addColorStop(1, "#000000");
369                         } else {
370                             gradient.addColorStop(0, crB);
371                             gradient.addColorStop(1, crE);
372                         }
373                         ctx.strokeStyle = gradient;
374                     }
375                     ctx.lineWidth = renderOptions['extrusionWidth'
                            ];
376                     ctx.beginPath();
377                     ctx.moveTo(prevX, prevY);
378                     ctx.lineTo(x * zoomFactor, y * zoomFactor);
379                     ctx.stroke();
380                 } else {
381                     if (renderOptions["showRetracts"]) {
382                         ctx.strokeStyle = renderOptions["
                                colorRestart"];
383                         ctx.fillStyle = renderOptions["
                                colorRestart"];
384                         ctx.beginPath();
385                         ctx.arc(prevX, prevY, renderOptions["
                                sizeRetractSpot"], 0, Math.PI * 2,
                                true);
386                         ctx.stroke();
387                         ctx.fill();
388                     }
389                 }
390             }
391             prevX = x * zoomFactor;
392             prevY = y * zoomFactor;
393         }
394         ctx.stroke();
395     };
396
397
398 // ***** PUBLIC *******
399     var publicFunctions = {
400         init: function () {
401             startCanvas();
402             initialized = true;
```

```
403             ctx.translate((canvas.width - gridSizeX * zoomFactor)
                    / 2, gridSizeY * zoomFactor + (canvas.height -
                    gridSizeY * zoomFactor) / 2);
404         },
405         setOption: function (options) {
406             for (var opt in options) {
407                 if (options.hasOwnProperty(opt)) {
408                     renderOptions[opt] = options[opt];
409 //                      console.log("Got a set option call: " + opt
        + " == " + options[opt]);
410                 }
411             }
412             ;
413
414             if (initialized)reRender();
415         },
416         getOptions: function () {
417             return renderOptions;
418         },
419         debugGetModel: function () {
420             return model;
421         },
422         render: function (layerNum, fromProgress, toProgress) {
423             var gCodeOpts = gcodeFactory.getOptions();
424             if (!initialized)this.init();
425             if (!model) {
426                 drawGrid();
427             } else {
428                 if (layerNum < model.length) {
429                     var p1 = ctx.transformedPoint(0, 0);
430                     var p2 = ctx.transformedPoint(canvas.width,
                        canvas.height);
431                     ctx.clearRect(p1.x, p1.y, p2.x - p1.x, p2.y -
                        p1.y);
432                     drawGrid();
433                     if (renderOptions['alpha']) {
434                         ctx.globalAlpha = 0.6;
435                     }
436                     else {
437                         ctx.globalAlpha = 1;
438                     }
439                     if (renderOptions['actualWidth']) {
440                         renderOptions['extrusionWidth'] =
                            gCodeOpts['filamentDia'] * gCodeOpts['
                            wh'] / zoomFactor;
441                     }
442                     else {
443                         renderOptions['extrusionWidth'] =
                            gCodeOpts['filamentDia'] * gCodeOpts['
                            wh'] / zoomFactor / 2;
444                     }
445                     if (renderOptions['showNextLayer'] && layerNum
                            < model.length - 1) {
446                         drawLayer(layerNum + 1, 0, this.
                            getLayerNumSegments(layerNum + 1),
                            true);
447                     }
448                     drawLayer(layerNum, fromProgress, toProgress);
449                 } else {
```

```
450                     console.log("Got request to render non-
                           existent layer!!");
451                 }
452             }
453         },
454         getModelNumLayers: function () {
455             return model ? model.length : 1;
456         },
457         getLayerNumSegments: function (layer) {
458             if (model) {
459                 return model[layer] ? model[layer].length : 1;
460             } else {
461                 return 1;
462             }
463         },
464         doRender: function (mdl, layerNum) {
465             var mdlInfo;
466             model = mdl;
467             prevX = 0;
468             prevY = 0;
469             if (!initialized)this.init();
470 
471             mdlInfo = gcodeFactory.getModelInfo();
472             speeds = mdlInfo.speeds;
473             speedsByLayer = mdlInfo.speedsByLayer;
474             volSpeeds = mdlInfo.volSpeeds;
475             volSpeedsByLayer = mdlInfo.volSpeedsByLayer;
476             extrusionSpeeds = mdlInfo.extrusionSpeeds;
477             extrusionSpeedsByLayer = mdlInfo.
                   extrusionSpeedsByLayer;
478             offsetModelX = (gridSizeX / 2 - (mdlInfo.min.x +
                   mdlInfo.modelSize.x / 2)) * zoomFactor;
479             offsetModelY = (mdlInfo.min.y + mdlInfo.modelSize.y /
                   2) * zoomFactor - gridSizeY / 2 * zoomFactor;
480             if (ctx)ctx.translate(offsetModelX, offsetModelY);
481             var scaleF = mdlInfo.modelSize.x > mdlInfo.modelSize.y
                    ? (canvas.width) / mdlInfo.modelSize.x /
                   zoomFactor : (canvas.height) / mdlInfo.modelSize.y
                    / zoomFactor;
482             var pt = ctx.transformedPoint(canvas.width / 2, canvas
                   .height / 2);
483             var transform = ctx.getTransform();
484             var sX = scaleF / transform.a, sY = scaleF / transform
                   .d;
485             ctx.translate(pt.x, pt.y);
486             ctx.scale(0.98 * sX, 0.98 * sY);
487             ctx.translate(-pt.x, -pt.y);
488             this.render(layerNum, 0, model[layerNum].length);
489         },
490         getZ: function (layerNum) {
491             if (!model && !model[layerNum]) {
492                 return '-1';
493             }
494             var cmds = model[layerNum];
495             for (var i = 0; i < cmds.length; i++) {
496                 if (cmds[i].prevZ !== undefined)return cmds[i].
                       prevZ;
497             }
498             return '-1';
499         }
```

```
    }
    return publicFunctions;
});
```

C.6 visualizer/viewer.js

Listing C.6: This file takes in all requests for parsing gcode and for canvas frames to the main page.

```
/**
 * Created by mike on 3/10/16.
 */
var app = angular.module('WebSlicer');

app.controller('gcodeController', ['$rootScope','$scope', '$http', 'renderFactory', 'gcodeFactory', function ($rootScope ,$scope, $http, renderFactory, gcodeFactory) {
    $scope.test = "working";

    // define web worker
    var worker = new Worker('viewer/Worker.js'); // no error checking for this

    // define return listener for the worker
    var processMessage = function (e) {
        var data = e.data;
        switch (data.cmd) {
            case 'returnModel':
                console.log('returnModel');
                gcodeFactory.setModel(data.msg.model);

                worker.postMessage({
                        "cmd": "analyzeModel",
                        "msg": {}
                    }
                );
                break;
            case 'analyzeDone':
                console.log('analyzeDone');

                gcodeFactory.processAnalyzeModelDone(data.msg);
                renderFactory.doRender(gcodeFactory.getModel(), 0);
                renderFactory.render(2, 0, 631);

                break;
            case 'returnLayer':
                console.log('returnLayer');
                gcodeFactory.processLayerFromWorker(data.msg);
                break;
            case 'returnMultiLayer':
                console.log('returnMultiLayer');
                gcodeFactory.processMultiLayerFromWorker(data.msg);
                break;
```

```
42              case "analyzeProgress":
43                  console.log('progress: ' + data.msg.progress);
44                  break;
45              default:
46                  console.log("default msg received" + data.cmd);
47          }
48      };
49      worker.addEventListener('message', processMessage, false);
50
51      $rootScope.render = function(gcodeString){
52          // split all lines
53          var lines = gcodeString.split(/\n/);
54
55          // post lines to worker for parsing
56          worker.postMessage({
57              "cmd": "parseGCode",
58              "msg": {
59                  gcode: lines,
60                  options: {
61                      firstReport: 5
62                  }
63              }
64          });
65
66          gcodeFactory.loadString(gcodeString); // load basic string
                 params (parse header)
67
68          delete lines; // be clean with our memory as these files
               are huge
69
70      };
71
72      // layer number and current progress
73      $scope.layerNumber = 2;
74      $scope.progress = 50;
75
76      $scope.$watch('layerNumber', function (newValue, oldValue) {
77          //console.log("layerNumber");
78          //console.log("newValue:" + newValue + ", oldValue:" +
               oldValue);
79          if (newValue) {
80              renderFactory.render($scope.layerNumber, 0, $scope.
                   progress);
81          }
82      });
83      $scope.$watch('progress', function (newValue, oldValue) {
84          //console.log("progress");
85          //console.log("newValue:" + newValue + ", oldValue:" +
               oldValue);
86          if (newValue) {
87              renderFactory.render($scope.layerNumber, 0, $scope.
                   progress);
88          }
89      });
90
91  }]);
```

C.7 model/files/filesController.js

Listing C.7: filesController.js handles all requests for uploading files to the server.

```
/**
 * Created by mike on 2/10/16.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.controller("FilesController", ["$http", "$scope", "$rootScope", "Upload", "OctoprintService", function ($http, $scope, $rootScope, Upload, OctoprintService) {

        $scope.modelFileId = '';

        $scope.logMe = function (item) {
            console.log(item);
        };

        // when the clientID changes we execute getting the file list
        $scope.$watch("$root.clientId", function (newValue, oldValue) {
            if (newValue !== "") {
                $scope.getFileList();
            }
        });

        /**
         * Get the list of flies associated with a client.
         */
        $scope.getFileList = function () {
            console.log($rootScope.clientId);
            $http({
                method: 'GET',
                url: $rootScope.baseUrl + "/getFiles/" + $rootScope.clientId
            }).then(function successCallback(response) {
                //console.log(response);
                $scope.modelFiles = [];
                for (var key in response.data) {
                    if (response.data.hasOwnProperty(key)) {
                        $scope.modelFiles.push({"key": key, "value": response.data[key]});
                    }
                }

            }, function errorCallback(response) {
                console.error(response);
            });
        };

            // upload on file select or drop
        $scope.uploadFile = function (file, errFiles) {
            var uploadUrl = $rootScope.baseUrl + "/importStl/" + $rootScope.clientId;
            $scope.f = file;
            $scope.errFile = errFiles && errFiles[0];
```

```

            if (file) {
                file.upload = Upload.upload({
                    url: uploadUrl,
                    data: {data: file}
                });

                file.upload.then(function (response) {

                    $scope.getFileList(); // refresh the file list

                }, function (response) {
                    if (response.status > 0)
                        $rootScope.errorMsg = response.status + ':
                            ' + response.data;
                }, function (evt) {
                    file.progress = Math.min(100, parseInt(100.0 *
                        evt.loaded / evt.total));
                });
            }
        };
    }]);
})();
```

C.8 model/files/filesDirective.js

Listing C.8: This directive links together the template and controller for ease of use when integrating into the index.html page.

```
/**
 * Created by mike on 2/10/16.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.directive('filecontrols', function () {
        return {
            restrict: 'E',
            templateUrl: 'model/files/filesTemplate.html'
        };
    });
})();
```

C.9 model/files/filesTemplate.html

Listing C.9: HTML snippet which controls the structure of the file upload dialog.

```
<div ng-controller="FilesController">
    <div class="panel panel-primary">
        <div class="panel-heading">
            <h3>Slicer Controls</h3>
        </div>
        <div class="panel-body">
```

```
7           <ul class="nav nav-pills nav-stacked">
8
9               <!--file chooser-->
10              <li role="presentation">
11
12                  <button class="btn btn-primary" type="file" ngf-
                        select="uploadFile($file,$invalidFiles)"
13                          ngf-max-height="1000" ngf-max-size="2MB">
14                      Select File
15                  </button>
16
17                  <div ng-show="f.progress >= 0">
18                      <h3>{{f.name}} {{errFile.name}} {{errFile.$
                            error}} {{errFile.$errorParam}}</h3>
19                      <div class="progress">
20                          <div class="progress-bar" role="
                                progressbar" aria-valuenow="{{f.
                                progress}}"
21                                  aria-valuemin="0"
22                                  aria-valuemax="100" style="width: {{
                                        f.progress}}%;">
23                                  <!--<div class="progress-bar" role="
                                        progressbar" aria-valuenow="{{f.
                                        progress}}" aria-valuemin="0"-->
24                                  <!--aria-valuemax="100">-->
25                                  {{f.progress}}% Complete
26                          </div>
27                      </div>
28                  </div>
29
30              </li>
31
32              <!--current file list-->
33              <li role="presentation" ng-if="$root.clientId">
34                  <p>{{modelFileId}}</p>
35                  <select id="repeatSelect" ng-model="modelFileId"
36                      ng-options="item.value for item in modelFiles
                            track by item.key">
37                  </select>
38              </li>
39
40              <!--slice button-->
41              <li role="presentation">
42                  <!--<button class="btn btn-primary" ng-click="
                        getFileList()">Refresh file list</button>-->
43
44                  <button ng-show="$root.clientId" ng-if="!$root.
                        busy" class="btn btn-primary"
45                          ng-click="doSlice(modelFileId)">
46                      Slice...
47                  </button>
48                  <i ng-if="$root.busy" class="fa fa-3x fa-cog fa-
                        spin"></i>
49              </li>
50
51          </ul>
52
53      </div>
```

```
    </div>
</div>
```

C.10 model/octoprint/octoprintController.js

Listing C.10: The controller for the octoprint integration which was cut late in development.

```
/**
 * Created by mike on 1/28/16.
 */
(function () {
    var app = angular.module("WebSlicer");
    app.controller("OctoprintController", ["$http", "$scope", "$rootScope", "OctoprintService", function ($http, $scope, $rootScope, OctoprintService) {

        // form collection scope variables
        $scope.url = "";
        $scope.port = "";
        $scope.apiKey = "";
        $scope.otherUrl = false;

        // data objects to be displayed
        $scope.octoprintStatus = {};
        $scope.octoprintFiles = [];

        // internal referance variables
        $scope.displayOctoprintData = false;
        $scope.connectionStatus = false;
        $scope.connectedAddress = "";


        var APIError = function (message, where) {
            console.error(message);
            if (where) {
                console.error("from: " + where);
            }
            $scope.displayOctoprintData = false;
        };

        /**
         * Settings object should look like this,
         * {"url":"","port":"","apiKey":""}
         *
         * call get defaults with an api key for another example
         * @returns {string}
         */
        $scope.connectOctoprint = function () {
            // figure out if url requires port and preform address resolution
            var address = OctoprintService.getDefaultAddress();
            if ($scope.port && $scope.url !== address) {
                address = $scope.url + ":" + $scope.port;
            } else {
                if ($scope.url) {
```

```
46                      address = $scope.url + ":5000";
47                  }
48              }
49
50              console.log(address);
51              console.log("API Key: " + $scope.apiKey);
52              $scope.connectedAddress = address;
53
54
55              // wait and resolve promise from pinging new octoprint
                    server
56              OctoprintService.connection(address, $scope.apiKey)
57                  .then(function successCallback(response) {
58                      console.log(response);
59                      if (response.data.current.state === "
                            Operational") {
60                          $scope.connectionStatus = true;
61
62                          // operational connection. make data
                                requests
63                          $scope.getOctoprintStatus(); // get the
                                current printer status
64                          $scope.getOctoprintFiles(); // get current
                                 list of files
65
66
67                      } else {
68                          APIError("Error: Printer not operational."
                                );
69                          $scope.connectionStatus = false;
70                      }
71
72                  }, function errorCallback(response) {
73                      APIError(response, "connectOctoprint");
74                  });
75          };
76
77          /**
78           * Testing getting the files list from octoprint
79           */
80          $scope.getOctoprintFiles = function () {
81              OctoprintService.getFileList($scope.connectedAddress,
                    $scope.apiKey)
82                  .then(function successCallback(response) {
83                      console.log(response);
84                      if (response.data.files) {
85                          $scope.octoprintFiles = response.data.
                                files;
86                          $scope.octoprintDataHere = true;
87                      }
88
89                  }, function errorCallback(response) {
90                      APIError(response, "getOctoprintFiles");
91                  });
92          };
93
94          /**
95           * Get a basic status message with the current printer
                 temps and connection status
96           */
```

```
        $scope.getOctoprintStatus = function () {
            OctoprintService.getPrinterInfo($scope.connectedAddress, $scope.apiKey)
                .then(function successCallback(response) {
                    console.log(response);
                    if (response.data) {
                        $scope.octoprintStatus = response.data;
                    }

                }, function errorCallback(response) {
                    APIError(response, "getOctoprintStatus");
                });
        };

        /**
         * Send the current gcode file to the octoprint server
         */
        $scope.postOctoprintFile = function () {
            console.log($rootScope.gcode);

            OctoprintService.uploadFile($scope.connectedAddress, $scope.apiKey, $rootScope.gcode)
                .then(function successCallback(response) {
                    console.log(response);
                }, function errorCallback(response) {
                    APIError(response, "postOctoprintFile");
                });
        }
    }]);
})();
```

C.11 model/octoprint/octoprintDirective.js

Listing C.11: This directive bound the controller and the template for easy and multi-use integration into the index page.

```
/**
 * Created by mike on 1/29/16.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.directive('octoprint', function () {
        return {
            restrict: 'E',
            templateUrl: 'model/octoprint/octoprintTemplate.html'
        };
    });
})();
```

C.12 model/octoprint/octoprintService.js

Listing C.12: This file wrapped the octoprint API so that it could be easily integrated with the controller.

```
/**
 * Created by mike on 2/1/16.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.service("OctoprintService", ["$http", function ($http) {

        /**
         * This just returns some basic info about the printer and displays it.
         * @param url
         * @param apiKey
         * @returns {*}
         */
        this.connection = function (url, apiKey) {
            return $http({
                    method: 'GET',
                    url: 'http://' + url + '/api/connection',
                    headers: {
                        'Content-Type': 'application/json',
                        'x-api-key': apiKey
                    }
                }
            )
        };

        /**
         * Gets the list of files currently uploaded to the octoprint local address
         * @param url
         * @param apiKey
         * @returns {*}
         */
        this.getFileList = function (url, apiKey) {
            return $http({
                method: 'GET',
                url: 'http://' + url + '/api/files',
                headers: {
                    'Content-Type': 'application/json',
                    'x-api-key': apiKey
                }
            })
        };

        /**
         * Get the current status of the printer including current temps
         * @param url
         * @param apiKey
         * @returns {*}
         */
        this.getPrinterInfo = function (url, apiKey) {
            return $http({
                method: 'GET',
                url: 'http://' + url + '/api/printer',
```

```
54                  headers: {
55                      'Content-Type': 'application/json',
56                      'x-api-key': apiKey
57                  }
58              })
59          };
60
61          /**
62           * Upload a new file to the octoprint server
63           *
64           * @param url
65           * @param apiKey
66           * @param file
67           */
68          this.uploadFile = function (url, apiKey, file) {
69              console.log('file is ');
70              console.dir(file);
71
72              // append form data (file parts) to request
73              var fd = new FormData();
74              fd.append('file', file);
75
76              // multipart/form-data post request
77              var uploadUrl = 'http://' + url + '/api/files/local';
78              return $http.post(uploadUrl, fd, {
79                  headers: {
80                      'Content-Type': 'multipart/form-data',
81                      'x-api-key': apiKey
82                  }
83              });
84
85          };
86
87          /**
88           * Load a filename from the local files list and start
                 printing it right now
89           * @param url
90           * @param apiKey
91           * @param fileName
92           */
93          this.printFile = function (url, apiKey, fileName) {
94              console.error("functionality does not exist yet.
                    printFile()");
95          };
96
97
98          /**
99           * Get the default connection params for OctoPi
100          *
101          * @param apiKey The OctoPrint api key.
102          * @returns {{url: string, port: string, apiKey: string}}
103          */
104         this.getDefaultAddress = function () {
105             return "octopi.local";
106         };
107     }]);
108 })();
```

C.13 model/octoprint/octoprintTemplate.html

Listing C.13: HTML structure which represents the view of the octoprint integration.

```
<!--OctoPrint STATUS PANEL-->
<div ng-controller="OctoprintController">
    <div class="panel panel-success">
        <div class="panel-heading">
            <h3>Connect OctoPrint</h3>
        </div>
        <div class="panel-body">
            <form>
                <div ng-show="otherUrl">
                    <div class="form-group">
                        <label for="ipAddress">OctoPrint IP Address</label>
                        <input ng-model="url" type="text" class="form-control" id="ipAddress"
                            placeholder="OctoPrint IP Address">
                    </div>
                    <div class="form-group">
                        <label for="port">Port</label>
                        <input ng-model="port" type="number" class="form-control" id="port"
                            placeholder="port (default 5000)">
                    </div>
                </div>
                <div class="form-group">
                    <label for="apiKey">API Key</label>
                    <input ng-model="apiKey" type="text" class="form-control" id="apiKey" placeholder="API Key">
                </div>
                <button ng-click="connectOctoprint()" class="btn btn-default">Connect</button>
                <button ng-click="otherUrl = !otherUrl" class="btn btn-primary">Custom Address?</button>
            </form>

            <!--TEMPERATURE-->
            <div class="panel panel-data" ng-if="octoprintDataHere">
                <h3>Printer Info</h3>
                <h4>Nozzle Temp: <strong>{{octoprintStatus.temperature.tool0.actual}}</strong> &#8451;</h4>
                <h4>Bed Temp: <strong>{{octoprintStatus.temperature.bed.actual}}</strong> &#8451;</h4>
                <h3>Current Files</h3>
                <h4 ng-repeat="file in octoprintFiles">{{file.name}}</h4>
                <!--<button class="btn btn-success" ng-click="postOctoprintFile()" ng-if="gcode.gcode"></button>-->
                <button class="btn btn-success" ng-click="postOctoprintFile()"></button>
            </div>

        </div>
    </div>
```

```
</div>
```

C.14 model/profile/settingsController.js

Listing C.14: This code is the main controller which brings in the static settings files and converts them into settings items.

```
/**
 * Created by mike on 9/25/15.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.controller("SettingsController", ["$http", "$scope", "$rootScope", function ($http, $scope, $rootScope) {

        $scope.showGcode = false;

        /**
         * Upload the settings as a formatted JSONObject to the server for slicing
         */
        $rootScope.importSettings = function () {
            var data = buildSettingsObject();
            return $http({
                method: 'POST',
                url: $rootScope.baseUrl + "/importSettings/" + $scope.clientId,
                headers: {
                    'Content-Type': 'application/json'
                },
                data: data
            })
        };

        function buildSettingsObject() {
            /*
             CONFIRMED WORKING SETTINGS
             "wall_line_count",
             "machine_start_gcode",
             "machine_end_gcode",
             "machine_width",
             "machine_depth",
             "machine_height",
             "material_print_temperature",
             "material_bed_temperature",
             "material_diameter",
             */

            var overrideBlackList = [
                "infill_line_distance",
                "wall_line_count",
                "machine_start_gcode",
                "machine_end_gcode",
                "machine_width",
                "machine_depth",
```

```
47                  "machine_height",
48                  "machine_heated_bed",
49                  "machine_head_with_fans_polygon",
50                  "machine_center_is_zero",
51                  "machine_nozzle_size",
52                  "machine_nozzle_heat_up_speed",
53                  "machine_nozzle_cool_down_speed",
54                  "gantry_height",
55                  "machine_use_extruder_offset_to_offset_coords",
56                  "machine_gcode_flavor",
57                  "machine_disallowed_areas",
58                  "machine_platform_offset",
59                  "machine_nozzle_tip_outer_diameter",
60                  "machine_nozzle_head_distance",
61                  "machine_nozzle_expansion_angle",
62                  "material_print_temperature",
63                  "material_bed_temperature",
64                  "material_diameter",
65                  "material_flow"
66              ];
67
68              var overrides = {};
69              var normal = {};
70
71              // build override object from all settings
72              for (var prop in $rootScope.settingsTracker) {
73                  //console.log("obj." + prop + " = " + $rootScope.
                       settingsTracker[prop]);
74                  var temp = {};
75                  for (var item in $rootScope.settingsTracker[prop])
                       {
76                      temp = {}; // clear object
77                      var name = $rootScope.settingsTracker[prop][
                           item].setting;
78
79                      // construct item. (2D arrays are hateful.
                           lucky this one is small)
80                      temp[name] = {"default": $rootScope.
                           settingsTracker[prop][item].default};
81
82                      // if item is not in blacklist
83                      if (overrideBlackList.indexOf(name) === -1) {
84                          angular.extend(normal, temp);
85                      } else {
86                          angular.extend(overrides, temp);
87                      }
88                  }
89
90                  // Fixed settings that must exist
91                  angular.extend(overrides, {"infill_sparse_density"
                       :{"default":100.0}});
92
93              }
94
95              var header = {
96                  "id": "prusa_i3", "version": 1,
97                  "name": $scope.name,
98                  "manufacturer": "Other",
99                  "author": "Other",
100                 "icon": "icon_ultimaker2.png",
```

```
                "platform": "prusai3_platform.stl",
                "inherits": "fdmprinter.json", // this must come from a symbolic link in the directory.
                "overrides": overrides
            };

            var output = angular.merge(header, normal);
            console.log(output);
            return output;
        }

    }]);
})();
```

C.15 model/profile/settingsTabDirective.js

Listing C.15: Each tab in the UI is represented as one of these directives.

```
/**
 * This is the directive which when given a json file as an argument will render the settings contained within as fields that can be edited.
 * Those fields can then be compliled into a larger file to be sent to the main server
 *
 * Created by mike on 1/29/16.
 */
(function () {
    var app = angular.module("WebSlicer");

    app.directive('curasettings', function ($http, $rootScope) {

        var linker = function (scope, element, attrs) {
            // get the json file as specified by the directive
            $http.get(attrs.src).success(function (data) {
                // register all settings
                if (!$rootScope.settingsTracker) {
                    $rootScope.settingsTracker = {};
                }
                $rootScope.settingsTracker[attrs.src] = data;

                scope.settings = data; // bind data to a local scope that we can reference easily
            });
        };

        return {
            restrict: 'E',
            link: linker,
            templateUrl: "model/profile/settingsDirectiveTemplate.html",
            scope: {
                src: '='
            }
        };
    });
})();
```

C.16 model/profile/settingsDirectiveTemplate.html

Listing C.16: This HTML snippet decides what kind of setting needs to be displayed and shows it.

```
<form>
    <div class="form-group" ng-repeat="item in settings">
        <!--<div ng-if="$index !=0">-->
        <h4 ng-if="settings[$index].category !== settings[$index-1].category"><bold>{{settings[$index].category}}</bold></h4>
        <!--</div>-->
        <!--styling for float type input-->
        <div ng-if="settings[$index].type === 'float' || settings[$index].type === 'int'">
            <label for="{{settings[$index].setting}}">{{settings[$index].label}}</label>
            <input type="number" class="form-control" id="{{settings[$index].setting}}"
                ng-model="settings[$index].default">
        </div>
        <!--styling for boolean inputs-->
        <div ng-if="settings[$index].type === 'bool'">
            <label><input type="checkbox" ng-model="settings[$index].default">
                {{settings[$index].label}}</label>
        </div>
        <!--styling for list type inputs-->
        <div ng-if="settings[$index].type === 'list'">
            <label for="settings[$index].setting">{{settings[$index].label}}</label>
            <select class="form-control" id="settings[$index].setting" ng-model="settings[$index].default">
                <option ng-repeat="option in settings[$index].list">{{option}}</option>
            </select>
        </div>
        <div ng-if="settings[$index].type === 'textarea'">
            <label for="{{settings[$index].setting}}">{{settings[$index].label}}</label>
            <textarea rows="20" class="form-control" id="{{settings[$index].setting}}"
                ng-model="settings[$index].default"></textarea>
        </div>
    </div>
</form>
```

C.17 model/profile/json/basic.json

Listing C.17: A full static settings file. There were five more files like this but one is sufficient to show the structure of this code.

```
[
```

```
2   {
3     "setting": "layer_height",
4     "default": 0.1,
5     "type": "float",
6     "category": "Quality",
7     "label": "Layer Height (mm)",
8     "description": "Layer height in millimeters.\nThis
        is the most important setting to determine the
        quality of your print. Normal quality prints are
        0.1mm, high quality is 0.06mm. You can go up to 0
        .25mm with an Ultimaker for very fast prints at
        low quality."
9   },
10  {
11    "setting": "wall_line_count",
12    "default": 2,
13    "type": "float",
14    "category": "Quality",
15    "label": "Shell Thickness In Lines",
16    "description": "Thickness of the outside shell in
        the horizontal direction.\nThis is used in
        combination with the nozzle size to define the
        number\nof perimeter lines and the thickness of
        those perimeter lines."
17  },
18  {
19    "setting": "retraction_enable",
20    "default": true,
21    "type": "bool",
22    "category": "Quality",
23    "label": "Enable Retraction",
24    "description": "Retract the filament when the nozzle
         is moving over a none-printed area. Details
        about the retraction can be configured in the
        advanced tab."
25  },
26  {
27    "setting": "solid_layer_thickness",
28    "default": 0.6,
29    "type": "float",
30    "category": "Fill",
31    "label": "Bottom/Top thickness (mm)",
32    "description": "This controls the thickness of the
        bottom and top layers, the amount of solid layers
         put down is calculated by the layer thickness
        and this value.\nHaving this value a multiple of
        the layer thickness makes sense. And keep it near
         your wall thickness to make an evenly strong
        part."
```

```
33    },
34    {
35      "setting": "infill_line_distance",
36      "default": 0.20,
37      "type": "float",
38      "category": "Fill",
39      "label": "Fill Density (%) Max = 1.0",
40      "description": "This controls how densely filled the
          insides of your print will be. For a solid part
          use 100%, for an empty part use 0%. A value
          around 20% is usually enough.\nThis won't affect
          the outside of the print and only adjusts how
          strong the part becomes."
41    },
42    {
43      "setting": "print_speed",
44      "default": 50,
45      "type": "float",
46      "category": "Speed and Temperature",
47      "label": "Print Speed (mm/s)",
48      "description": "Speed at which printing happens. A
          well adjusted Ultimaker can reach 150mm/s, but
          for good quality prints you want to print slower.
           Printing speed depends on a lot of factors. So
          you will be experimenting with optimal settings
          for this."
49    },
50    {
51      "setting": "material_print_temperature",
52      "default": 210,
53      "type": "int",
54      "category": "Speed and Temperature",
55      "label": "Printing Temperature (C)",
56      "description": "Temperature used for printing. Set
          at 0 to pre-heat yourself.\nFor PLA a value of 21
          0C is usually used.\nFor ABS a value of 230C or
          higher is required."
57    },
58    {
59      "setting": "material_bed_temperature",
60      "default": 70,
61      "type": "int",
62      "category": "Speed and Temperature",
63      "label": "Bed Temperature (C)",
64      "description": "Temperature used for the heated
          printer bed. Set at 0 to pre-heat yourself."
65    },
66    {
67      "setting": "support",
```

```
68      "default": 0,
69      "type": "list",
70      "list": [
71        "None",
72        "Touching buildplate",
73        "Everywhere"
74      ],
75      "category": "Support",
76      "label": "Support Type",
77      "description": "Type of support structure build.\n\"
          Touching buildplate\" is the most commonly used
          support setting.\n\nNone does not do any support
          .\nTouching buildplate only creates support where
           the support structure will touch the build
          platform.\nEverywhere creates support even on top
           of parts of the model."
78    },
79    {
80      "setting": "platform_adhesion",
81      "default": 0,
82      "type": "list",
83      "list": [
84        "None",
85        "Brim",
86        "Raft"
87      ],
88      "category": "Support",
89      "label": "Platform Ashesion Type",
90      "description": "Different options that help in
          preventing corners from lifting due to warping.\
          nBrim adds a single layer thick flat area around
          your object which is easy to cut off afterwards,
          and it is the recommended option.\nRaft adds a
          thick raster below the object and a thin
          interface between this and your object.\n(Note
          that enabling the brim or raft disables the skirt
          )"
91    },
92    {
93      "setting": "material_diameter",
94      "default": 1.75,
95      "type": "float",
96      "category": "Filament",
97      "label": "Diameter (mm)",
98      "description": "Diameter of your filament, as
          accurately as possible.\nIf you cannot measure
          this value you will have to calibrate it, a
          higher number means less extrusion, a smaller
          number generates more extrusion."
```

```
99    },
100   {
101     "setting": "material_flow",
102     "default": 100,
103     "type": "float",
104     "category": "Filament",
105     "label": "Flow (%)",
106     "description": "Flow compensation, the amount of
          material extruded is multiplied by this value"
107   },
108   {
109     "setting": "machine_nozzle_size",
110     "default": 0.4,
111     "type": "float",
112     "category": "Machine",
113     "label": "Nozzle Size (mm)",
114     "description": "The nozzle size is very important,
          this is used to calculate the line width of the
          infill, and used to calculate the amount of
          outside wall lines and thickness for the wall
          thickness you entered in the print settings."
115   }
116 ]
```