ProtoCases
A Computer Engineering Senior Project

Christopher M. Polis
Advisor: Dr. Christopher Lupo
California Polytechnic State University, San Luis Obispo

June 11, 2013

1 Introduction

The goal of this senior project is to develop a web application where users can easily design a smart phone case and generate a model file that can be 3D printed. One intent of this project is to utilize the ability to create unique products and the personalization capability of 3D printing for a widely used consumer product. This project will also enable crowdsourcing of smart phone case design by allowing users to share their designs.

This web application features three methods of designing cases that vary in the experience and time necessary to use them. For the least technically savvy users, there is a template system that provides a user with a few fields to fill out. There is also a drag and drop editor that allows the user to use text and shapes to design a case. For users with design experience, there is the ability to upload a black and white image to design their case.

Case designs are based on a black and white image to represent the back of the case; black area on the image will be either negative space or in relief on the back of the case. There may also be potential to customize the sides of each case.

The editor and templating system must be easy enough to use that people with little or no design experience can use them; they must also be able to create designs that are personalized and visually appealing enough for consumers to order them. Another goal of this project is to allow users to

have a 3D preview of their designs in the browser to help them visualize what their case will look like and make it seem tangible than a simple 2D image.

This project is intended to be publicly launched into a commercially viable business. The web application will integrate with a company that provides 3D printing as a service to handle order fulfillment and payment processing. The technology for this project, particularly for model generation, editors and templating system will be designed and implemented in a way where they can be reused to design other products.

The current working name for this application is ProtoCases, chosen because the prefix proto- means *first*, *e.g. "prototype."*

2 Background

3D printing technology has significantly improved and become more widespread in the past few years. Some improvements have been made in the quality and speed of 3D printing as well as a reduction in the cost of 3D printing; this has allowed 3D printing to be used for consumer products such as phone cases and not just for rapid prototyping. This project would not be possible without these advances.

However, the technical focus of this project is on developing a web application. There have also been advances in web technologies in the past few years that have improved the quality of this web application and reduced development time.

The website will be powered by a popular open source web framework called Ruby on Rails and a PostgreSQL database. Ruby on Rails allows for rapid development by using paradigms such as *convention over configuration*, *model-view-controller architecture(MVC)*, *test-driven-development(TDD)* and *dont-repeat-yourself(DRY)*[**?**, rails] The Rails community is very active and there are many plugins, called gems, that allow Rails developers to quickly add in common functionality.

The application is fairly complex and is broken into separate codebases and deployed as different modules; it utilizes a somewhat service oriented architecture that allows for easier maintenance and increased performance. Different parts of the application, such as the model generator or 3rd party API handlers are deployed as discrete applications that communicate through HTTP or message queues. This application is deployed on Amazon Web Services. Seperate EC2*(Elastic Cloud Compute)* instances are used for the

following: the main application, workers to handle API calls and email, and 3D model generation. S3*Simple Storage Service* is used to serve static assets and store design images and model files.

Because of the interactive nature of the website, the front end will be JavaScript heavy and utilize JavaScript libraries such as jQuery, three.js and fabric.js. jQuery is used to handle user interactions and page manipulation without navigating to a new page, while fabric.js adds functionality to the editor such as dragging, resizing and rotating objects.

A key aspect of the site is the ability to see 3D previews of a design that a user is working on or a 3D model of a design that has been created by a different user. This is made possible by WebGL, a technology that is available in modern web browsers. Using WebGL, users will be able to interactively rotate and zoom on a 3D model of their case in a web browser.

One of the most important technical aspects of this project is the ability to generate 3D models from 2D images in a way that is quick and lightweight enough for the web. Initially, a basic version of this functionality was implemented in Ruby, a higher language level, but this was too slow and would not be feasible for a web application. This functionality was ported from Ruby and made more robust with the C programming language.

The standard model file format for 3D printing is .STL*(for STereoLithography)*. These files consist of a set of triangles that makeup a solid object and are in either an ASCII or binary format[1]. STL files are used in this project for both browser based previews and actual 3D printing.

3 Description

The architecture and implementation of this project is fairly complicated, but can broken down into a few major parts; as noticed in the previous section, this web application is built with a somewhat service oriented architecture.

3.1 Model Generation

The core technical challenge of this project was to come up with a system to convert an arbitrary black and white image into a 3D model where the black regions of the image become either negative or relief space. There are existing software tools to do this, however, because this is part of a web

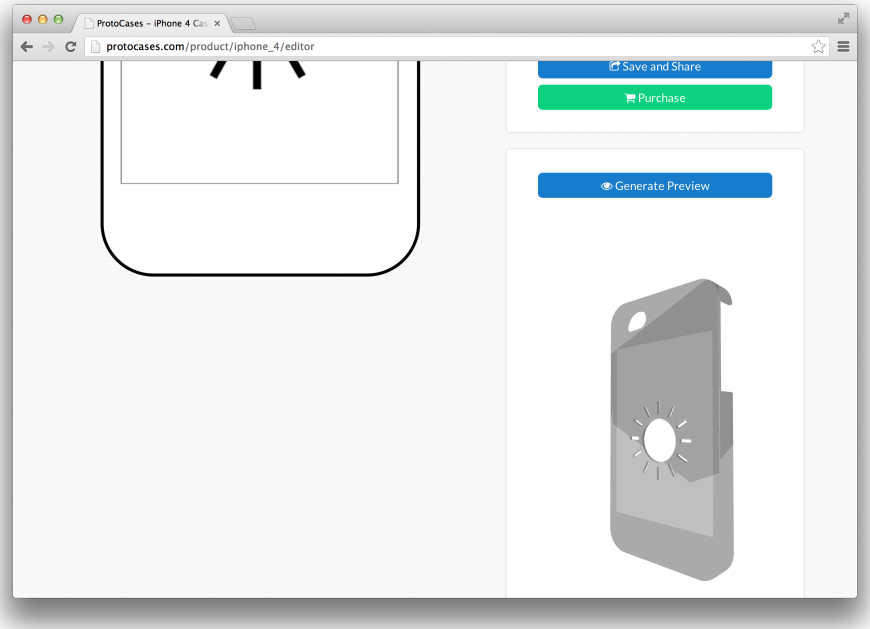
application, it has the following requirements that were not met by existing tools:

Automated this process must be able to run without user interaction and as an independent service, *i.e. from the command line.*

Fast models must be generated in less than a few seconds because it is part of a web application, where users expect minimal wait times

Lightweight minimal computing power should be used to generate models to allow for high traffic with minimal hosting resources

Flexible the system should allow for different products and model customization styles



3.1.1 Overview

The system for generating models uses templates with a rectangular empty space where the design image will fill in. Thus, the model generator needs to just generate the faces for the cutout rectangular area.

3.1.2 Algorithm

For generating the faces for the rectangular area that is represented by the design image, the design image is treated as a grid of cubes where each black or dark pixel is a cube and the rest of the image represents empty space. To generate the STL representation of this, the following faces must be created*(note: this is from the perspective of looking down on the image)*: the faces on the four sides of the dark areas and the top and bottom faces. We could simply add the faces that makeup each cube, however, this would be wildly inefficient and create large STL files because there are faces on the inside of solids. Similarly, outer edges that border each other should be combined to reduce the triangle count of the output file. The algorithm works in the following way:

1. The image data is converted into a 2d array of either 1's or 0's to represent 'solid' or 'not solid' where the RGB value is tested against a threshold to see if it is dark or light. In C, this data is contained in an array*(accessed as a 2D array)* of chars.

2. The solid faces along each row are generated by traversing between each row and detecting when there is a 1 above and a 0 below or a 0 above and 1 below, there should be a face in this location. The first time this is detected, the column position is saved. Once this 1/0 or 0/1 condition is not satisfied, the face that spans the length of the 1/0 or 0/1 area is created.

3. In a similar fashion, solid faces along each column are generated by finding the locations between solid and empty space and creating faces that are as long as possible.

4. The top and bottom faces are generated with the following algorithm:

   (a) Copy the image data grid

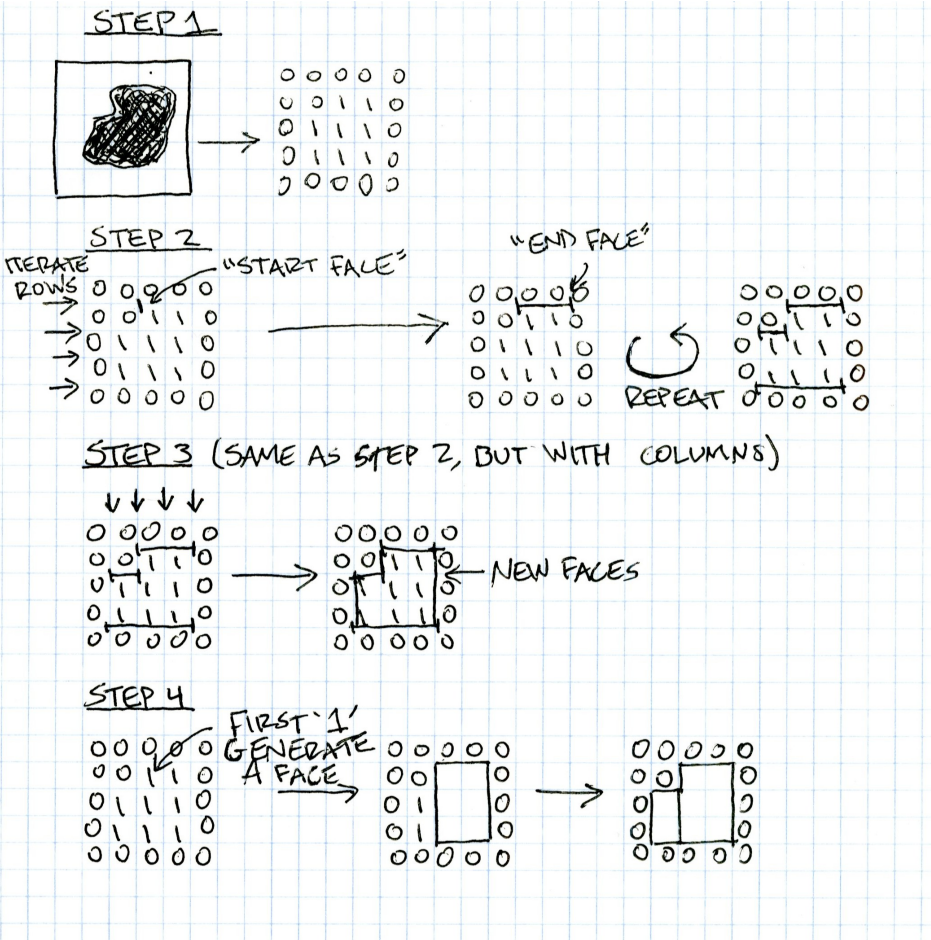
   (b) Traverse the copied grid, going down each row until a 1 is detected.

   (c) Generate a rectangle by stretching in the X generation while there are 1's

   (d) Then stretch this rectangle in the Y direction if the part of the grid after the first stretch are all 1's

   (e) Add this face and clear out the 1's that it spans

(f) Repeat this algorithm until the copied grid is all 0's



3.1.3 Implementation

This system was first implemented in Ruby because of ease of development. The initial implementation proved that the algorithm worked, helped with debugging and figuring out edge cases and showed that model templates could be used to generate complete models of products.

3.1.4 Port to C

However, the initial implementation was too slow and memory heavy for its intended application. The algorithm was ported to C and a more robust

and flexible STL editing library was written alongside it. The C program to generate models takes the following parameters:

- STL template file to add to (optional)
- Binary or ASCII file output
- Extrusion type: 'Extrude', 'Cut', 'Sunken' or 'Relief'
- Extrusion dimensions: depth, height and width

The C library that is used for model generation is significantly faster than its Ruby counterpart and fast enough for a web application. When generating models from an image 1000px by 1000px, the Ruby script took on average*after 100 tests* took 28.4s while its C counterpart averaged only 340ms per run. The C program also had a much smaller memory footprint because it utilizes primitive data types and low level memory manipulation while all data in Ruby are treated as objects and there is less ability to manage memory.

3.2 Website Architecture

As noted previously, the website is built with the open source MVC framework, Ruby on Rails.

3.2.1 Main Models

The primary models and database tables for this application are:

Users User information such as a short bio and location as well as login information and metrics

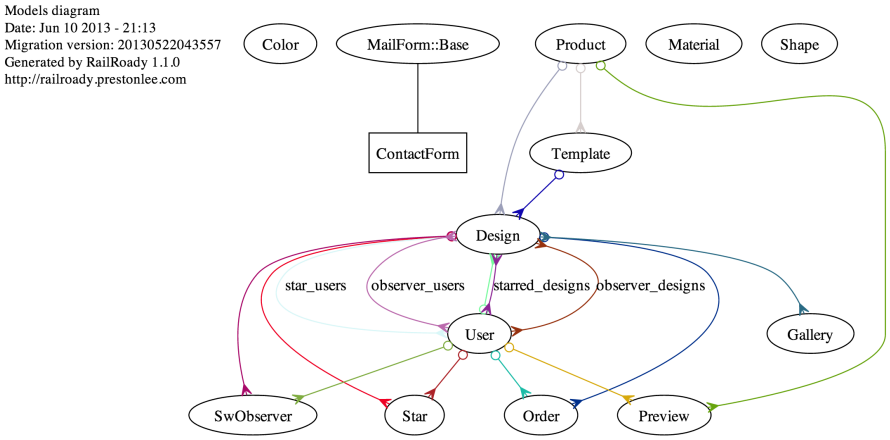
Designs User created designs and metadata

Products Metadata such as name and editor sizes for each product

Stars Users can 'star' or add designs to their favorites for quicker access and to share with their friends

Shapeways Observers Once a user selects a design to purchase, if it is not on Shapeways yet, they will be notified by email when it is ready. This model indicates that a user is waiting for a model to become ready for purchase.

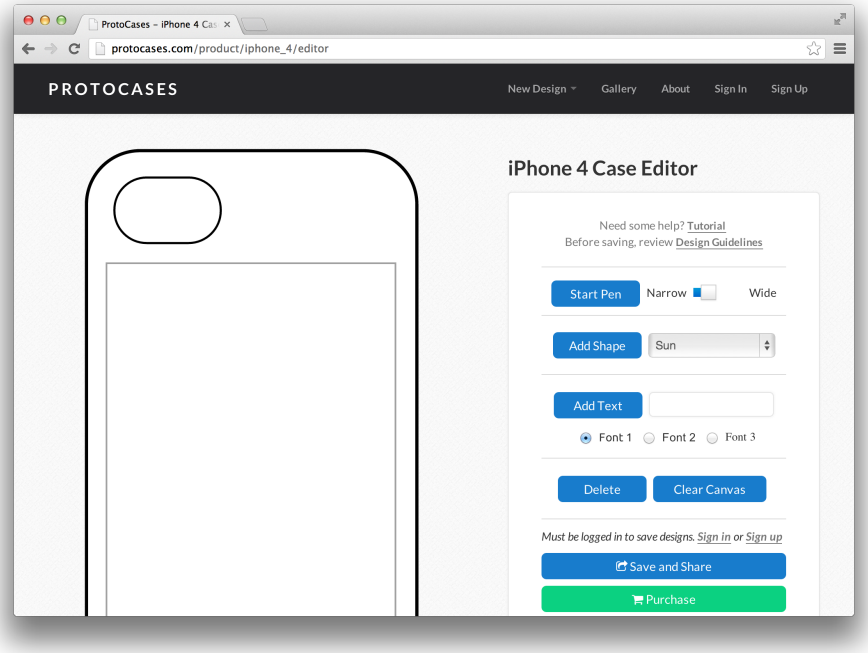
Materials Material information such as color, price and markup; used when working with the Shapeways API



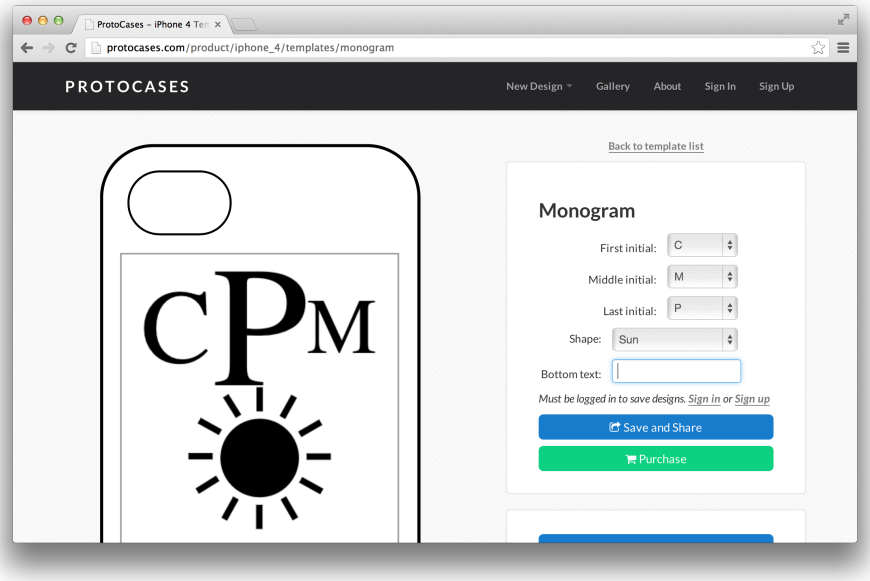
3.2.2 Main Pages(Controller Actions and Views)

This web application features many static and dynamic pages. Many pages reuse code for shared functionality, *e.g. the editor pages for different phones differ slightly.* Here is a list of the more important ones along with their functionality and how they work:

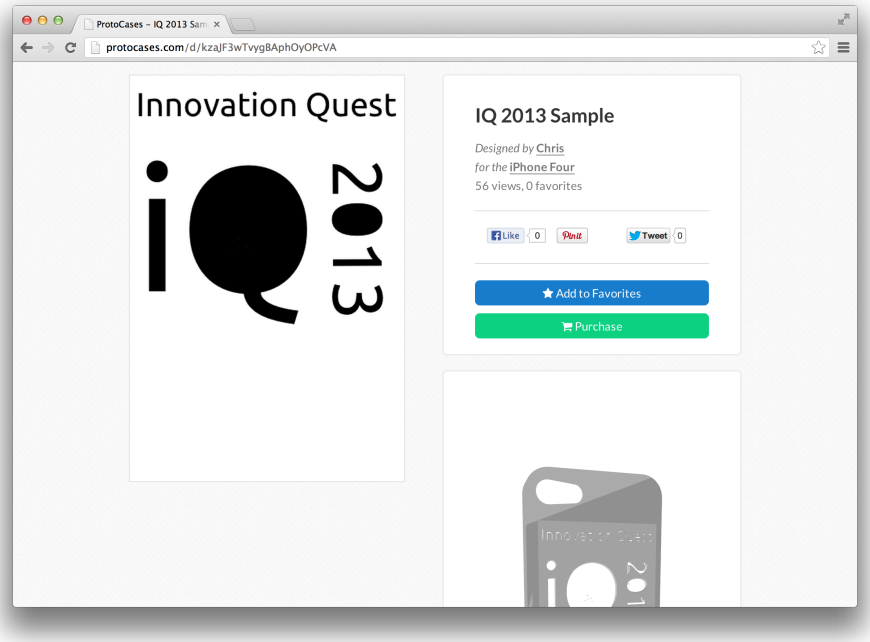
Editor This page features a robust 2D image editor for designing cases. The editor allows users to add, drag, resize and remove text and shapes as well as draw with a pen tool. The black and white 2D design that is created on this page is used for generating a model for either preview or purchase. Design data is stored in both raster(PNG) and vector(JSON/SVG) formats.



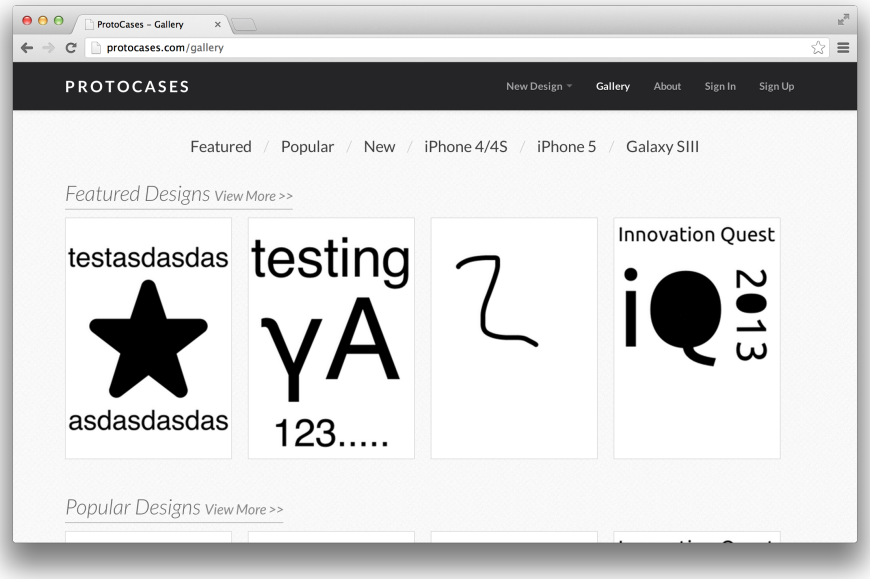
Template Similar to the editor page, this allows users to create designs. However, these pages are designed to be simpler to use and allow the user to customize their design through text fields and select menus.



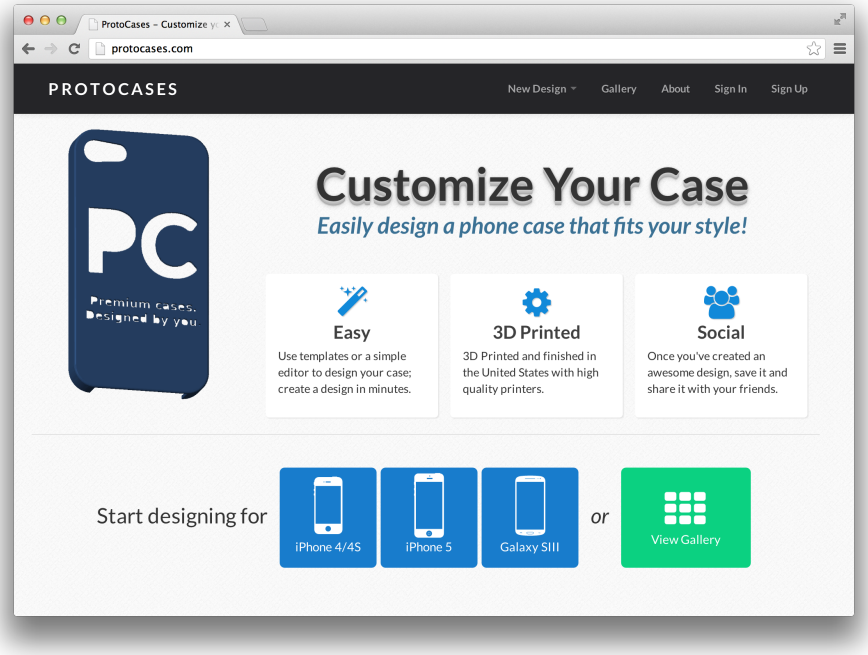
Design Page Once a user has created a design, a page gets generated where they can share, review and purchase their design. This page features buttons to share via social media, add to favorites on ProtoCases or purchase through Shapeways.



Gallery Not all users are going to want to create their own design and will often want to see what others have made. For this case, the site will feature a main gallery page as well as more specific gallery pages. Gallery pages are large grids of design images that show more information when a user mouseovers one of them. Specific gallery pages are created for each phone, popular(in terms of views), features(manually selected designs) and new designs.



Home/Landing Page This is the root page for ProtoCases. It is a static page that briefly describes what the web application is and how it works. It features large call to acton buttons to get new users to design a case or view the gallery.



3.3 Editor and Template System

3.3.1 Overview

The core user interface of this application are the previously mentioned editor and template pages. These are built using HTML5 canvas, JavaScript and open source libraries. The design is built on top of an HTML5 canvas element that contains different vector elements. Building this system to use vector instead of bitmap data allows for more responsive editing, less bandwidth and sharper designs.

3.3.2 Canvas, fabric.js

Fabric.js, an open source JavaScript library is used to handle adding shapes, pen and text elements to the canvas. It also has built in functionality that allows for dragging and resizing elements.

3.3.3 User Interaction

The controls on the right side of the template and editor pages are managed by jQuery. Each input is bound to an event that performs some DOM manipulation and/or calls fabric.js to modify the design canvas.

3.3.4 Saving and Previews

The editor and template system use vector graphics. However, raster image data is needed for the 3D model generator and for saving designs. HTML5 and modern browsers feature a function that allows for converting an HTML canvas element into raw image data: *toDataURL('image/png'))*[2]. Users can view 3D previews without navigating pages by making an AJAX call that goes through the main Rails application which calls the 3D model generator service and on success returns the URL of a 3D model file to the client. The 3D preview is displayed in the browser by using the three.js library which utilizes WebGL, another modern browser technology.

4 Evaluation

This project was a technical success; all major parts of the web application have been developed and are working well, *i.e. the template system, the*

editor, uploads, model generation, previews, user management, Shapeways integration, etc.... A user can now go through the entire process of creating a design with a template, the editor or an upload, saving their design and checking out through Shapeways. The application is deployed and in a production ready state with a soft launch planned for the near future.

Page load times and preview generation times are reasonable. When tested with Apache Bench and settings of 200 requests and concurrency level of 5, the homepage, editor page and design page had 90

There haven't been any formal usability studies for the website, but when shown for the first time to new users, the template system and editor have been well received and easily used.

5 Conclusions

Despite requiring significantly more work than anticipated, this project was very rewarding and technically interesting. Cases have been created on a personal 3D printer from models generated on the site and hopefully many more will be created from users through Shapeways; seeing a completely unique physical good be generated from this software has been very satisfying.

Further development of this project will include making the site more feature rich as well as adding the ability to design for different products.

There has been tremendous growth in 3D printing and accompanying software and web applications in the past few years and this project demonstrates another way in which this emerging technology can be used.

References

[1] Marshall Burns, Ph.D. *Automated Fabrication: Improving Productivity in Manufacturing, Section 6.5* 1993. www.ennex.com

[2] HTML Canvas Element - Mozilla Developer Web API Reference. https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement

[3] Ruby on Rails. http://rubyonrails.org/