University of South Carolina
Scholar Commons

Senior Theses | Honors College

Spring 5-5-2016

Dynamic 3D Zverse Models on the Web

Alexa Ann Breeland
University of South Carolina - Columbia

Ming Wong
University of South Carolina - Columbia

Follow this and additional works at: https://scholarcommons.sc.edu/senior_theses

 Part of the Computer Engineering Commons

Recommended Citation
Breeland, Alexa Ann and Wong, Ming, "Dynamic 3D Zverse Models on the Web" (2016). *Senior Theses*. 66.
https://scholarcommons.sc.edu/senior_theses/66

This Thesis is brought to you by the Honors College at Scholar Commons. It has been accepted for inclusion in Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact dillarda@mailbox.sc.edu.

DYNAMIC 3D ZVERSE MODELS ON THE WEB

By

Alexa Breeland & Ming Wong

Submitted in Partial Fulfillment
Of the Requirements for
Graduation with Honors from
South Carolina Honors College

May, 2016

Approved:

Dr. Jose Vidal
Director of Thesis

Digitally signed by Bridgette Parsons
DN: cn=Bridgette Parsons, o=University of South Carolina, ou=College of Engineering and Computing, email=parsonba@email.sc.edu, c=US
Date: 2016.04.26 19:32:06 -04'00'

Bridgette Parsons
Second Reader

Steve Lynn, Dean
For South Carolina Honors College

Table of Contents

Thesis Summary

ZVerse is a 3D printing company that is responsible for converting 2D objects into 3D objects. This company specializes in generating 3D collegiate products. Whenever a customer orders a 3D model, such as an alumni brick, it is often helpful to allow customers to see what the object looks like before the purchase. Unfortunately, previewing the brick at runtime is quite a challenging task. The purpose of our project is to create a shopping website that models the existing ZVerse company website and incorporates dynamic 3D rendering of the brick model with the user-input text.

The project can be divided into two main sections. The first part is the development of the webpage that is built on top of a SQL database. From this web application, the user can purchase a product or a brick model while the admin user can upload one. We have carefully design the database to ensure that it met the specific requirements of our web application. For example, there are fields such as isStandard and isVisible that determines whether the brick should be displayed on the site. We also split the product into various school categories so that the user can easily search for the product within their chosen school.

Likewise, we make our application as user-friendly as possible. On the homepage, we give a short tutorial on how to navigate the website. For the custom brick object, we allows users to insert their text and move the text on top of the brick model. We also let them select various font styles. When a potential customer adds an item to the cart, he or she can easily update the quantity or view the inscription text on the shopping cart page. On the admin page, there are small previews of the model for the admin to view. The admin user also has the ability to edit both the title and price of the item.

The second part deals with integrating the brick models onto the webpage. We have to consider the directional lights and positions of the brick in x, y coordinates. The brick models comes in different shapes such as rectangular prism, a cube, or a flat plaque. Following our user-friendly principal, we also enable the user to rotate and zoom on the object.

In addition to the brick models itself, we also implement a text layer that aligns perfectly on top of each object. This text layer is made up of three horizontal lines of characters, based upon the value from the three text boxes on the brick display page. A user can input whatever text he or she likes in the textboxes, and the text will be displayed on the brick surface. This is one of the selling features of the webpage because users can now see the preview of the brick in 3D with the text in real time after the user types. Moving the text proves to be even more difficult as we have to make sure that moving the text does not move the underlying brick layer. Eventually, after numerous attempts, we are able to make the brick movable.

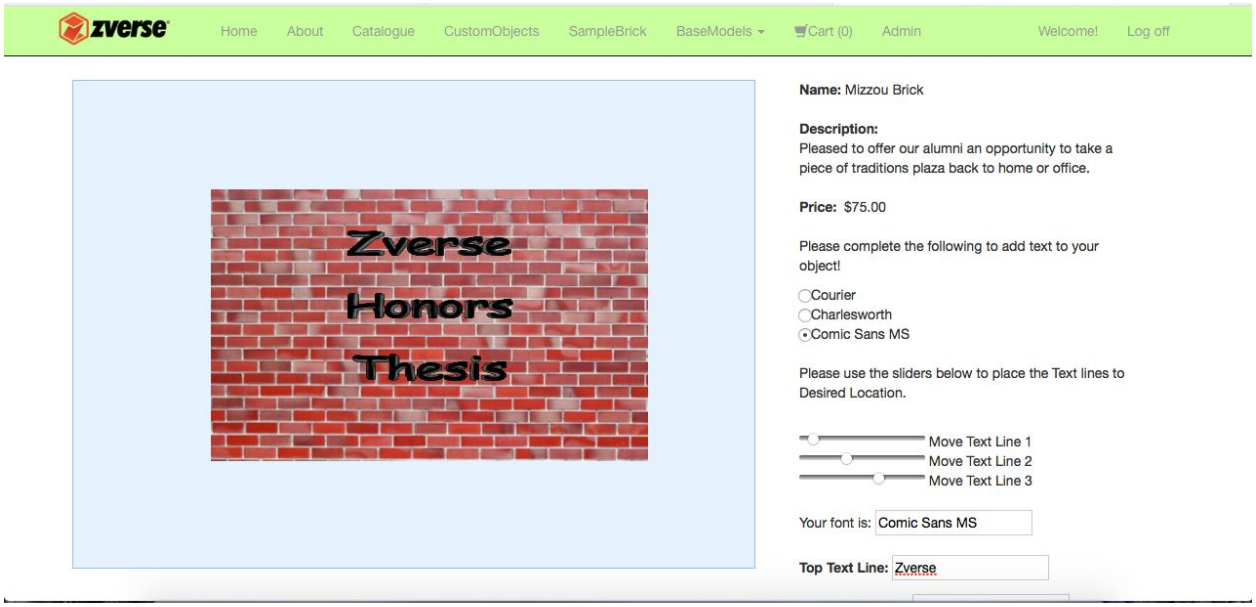
Overall, the project is successful in implementing the core functionality. With just a few simple clicks on our website, the user is able to view a highly customized brick with text before purchase. In the future, we plan to employ more user customization to the individual brick model such as incorporating various font sizes for the text, allowing greater movement of the brick model, and a greater variety of products.

Introduction

The client for this project was ZVerse, Inc. This is a 3D printing company that is a world leader in taking 2D content and making it 3D printable, then printing the given model as a 3D object. One of their major projects was for the University of Missouri, where they created alumni brick replicas for donors. They have a website which allows Mizzou fans to order the 3D replica of their alumni brick, yet the website lacked some functionality. Their main desire was a model of the 3D brick online, which could be dynamically changed at runtime to include any edits the user made to the model. They deemed this a useful feature of their website because often users want to see the text they order on the brick on a sample before they receive the real object. The original purpose of our project was to implement a user interface that would update the 3D model of the brick as the user inputs the configurations.

Overview of the features

Below is a screenshot of a main feature of our website.



As we dove deeper into the project, our thesis director added a multitude of features that would be useful if our project were to become a fully functional stand-alone website. Instead of simply creating the page with the 3D model, we ended up creating an entire site that could be used by a company such as ZVerse that needed full control over the 3D models offered on the site.

Our website is located at http://zverse.azurewebsites.net. The homepage includes a header image of ZVerse's logo, and beneath is a video tutorial of how to use our site. This video autoplays when the user enters the site, but on a mouseover, it will generate a bar with settings that let the user control the video. The right hand side of the homepage lists features of ZVerse that their company website also boasts. These include 3D Content Creation, Product Engineering, and Full Color 3D Printing. There is a description about each of these features, with the option to click to learn more, which directs to ZVerse's website with full information.

The navigation bar across the top is used to navigate across the website. Clicking the ZVerse logo takes the user to the home page, as does simply clicking 'Home'. Clicking on the 'About' tab leads to a page with information about the project itself and contact information for ZVerse. The 'About This Project' section explains that this is a year long Capstone project, lists the contributors names, and includes a link to our github repo to provide the code behind the website. This code is currently private, yet can be made public if we so wish. The ZVerse contact information includes the company's address, phone number, and email address.

The next tab is the 'Catalogue' tab. This was one of the more difficult features to implement. This page displays all of the current items available for purchase from ZVerse. In the top left corner, there is an option to filter the catalogue results by school, leaving only the USC, Clemson, or Mizzou products based on the filter selected. The catalogue displays a picture of

each object, the name of the object, and a price. Clicking on either the picture or the name of the object will lead to a separate page with just the selected object and a description. Once on this detailed item page, there is now the option to ‘Add To Cart’ if this is the desired object for purchase.

The next tab on the site is ‘Custom Objects’. This page is another catalogue, much like the last tab. However, the three objects listed here are the objects that had 3D viewing functionality on the site. The page is labeled ‘Models with Text’ because these objects, when selected, allow the user to input text, manipulate the text, move the text, and save the object with the inscribed text to the user’s cart. Once again, the picture, object name, and price are displayed. Clicking on either the picture or the object name will lead the user to the detailed object page.

Selecting the first object, Mizzou Brick, leads the user to the main function of this project. This was the feature we were enlisted to complete by the client. When selecting Mizzou Brick, the user is sent to a detailed page about the 3D brick object. On the left is a viewing window that contains the brick in 3D form. If the user moves the mouse into this viewing window, the user can scroll in and out, which makes the brick itself zoom in closer and out farther. There have been limits placed so that the user cannot zoom out until the brick disappears or zoom in far enough to enter the 3D brick, as these would confuse the user. With the mouse in the viewing window, the user may click on the brick object and drag the mouse while holding down to spin the object. This allows the user to see all sides of the object, and the top and bottom, but does not allow the user to flip the brick upside down. To the right of the viewing window, there is the name of the object, a brief description, and the price. Beneath this, there are three font options, including Courier, Charlesworth, and Comic Sans. The user can select any of these three fonts,

and begin inputting text. There are three text boxes included on the page. The user simply begins typing in any of these boxes, and the text will appear on the brick in 3D as they type. If the user changes their mind and selects a different font after typing, the text on the brick will automatically change to the new font, without the need to refresh the page. A more recent feature that has been added is the ability to move the inputted text. There are three sliders above the text boxes. Each of these sliders corresponds to its respective text box. These sliders begin with default values of where we decided a user would most likely like his or her text to be aligned. Otherwise, as the user began typing, all of the text would've appeared on top of each other until the user used the sliders to move it. The user simply use the sliders to rearrange the text on the object, moving the text as high or low on the brick as desired, without the option to actually exit the brick. Being able to add 3D text over a 3D object during runtime at real speed as the user types and makes edits was the main feature of the project we were asked to implement. This is where much of the time spent on the project went. Again, this object may be added to the cart.

The next tab is the 'Sample Brick' tab. This is a different type of 3D model than the previous ones because it is generated from files given to us by the client. These obj files are difficult to load on the web, so they asked us to try to load this as an example of the in depth, detailed files that are actually used to print the 3D models. The page displays the actual 3D object that gets printed out of their 3D printer when the user orders a Mizzou alumni replica brick. This is explained on the right side of the object with the option to learn more by following the link.

The next tab is a dropdown menu. Selecting 'Base Models' brings up a drop down that offers the choice between the three 3D objects, Mizzou Brick, USC Cube, and Clemson Plaque.

Selecting ‘Mizzou Brick’ will take the user to the page with the 3D object with text that was discussed previously. Selecting ‘USC Cube’ will take the user to a detailed object page identical to the Mizzou Brick, except the object itself is now a USC Cube, and the text will appear on all sides of the cube. Selecting ‘Clemson Plaque’ will take the user to a page showing a wooden plaque with the chance to type text on top, as with the three other 3D objects offered on the site.

The next tab is the ‘Cart’ tab. This tab displays how many items are in the cart in the tab heading. If ‘Cart’ is selected when the cart is empty, there will be a (0) included in the tab and the page will inform the user that the cart is empty. Every time something is added to the cart, the page redirects to the cart page to show the current summary. On each line, it shows how many of that specific item are in the cart. The cart table columns are set up as follows: ID, Name, Unit Price, Inscription, Quantity, Delete, and Item Total. The ID is the product ID as defined in the product table of the database. The name column has the name of each of the objects, with a hyperlink back to the item details page for that object. For the objects that allow 3D text editing, when the user clicks on the hyperlinked name for the object, it will return to the product page with the text that was on the object when it was added to the cart. This allows the user to see it once more as they had defined it originally. The unit price displays the price for one of the object listed on that line. The inscription displays the text that the user input on the 3D objects that allow text inscriptions, and is blank for the non customizable objects. The quantity displays how many of the item is currently in the cart, but is modifiable while on the cart page. The user can click in the box and change the number, click enter or select ‘Update’ to update the cart. The delete option allows the user to select the check box and click ‘Update’ which removes anything in the cart that was selected under the delete column. The ‘item tota’l displays the unit price times the quantity for that item. This gives the total price for all of that particular item that are being purchased. The bottom row in the cart table gives the subtotal of

the items, simply the totaling of all of the ‘Item Total’ column. Below the cart table, the shipping cost and the sales tax are displayed, which are added to the subtotal and displayed as the order total. Clicking ‘Checkout’ takes the user to a congratulations post that informs them that the order is completed. This could be implemented in the future as an actual checkout page, instead of a dummy order completed page. This was not necessary in the scope of our project, so we simply left it as a ‘Congrats!’ page.

On the right side of the navigation bar, there are two options: Register and Log In. ‘Register’ allows the user to create a new account. The user simply enters a name, an email, and a password, and clicks ‘Register’. The password specifications will appear if the password is not complicated enough. These include: ‘Passwords must be at least 6 characters. Passwords must have at least one non letter or digit character. Passwords must have at least one digit ('0'-'9'). Passwords must have at least one uppercase ('A'-'Z').’ Upon confirming that the password meets these specifications, the page will inform the user: ‘An email has been sent to your account. Please view the email and confirm your account to complete the registration process.’ An email will be sent to the user, with a link asking the user to click on the link to complete registration and activate the account, to ensure that the users are in fact real people with access to the email they are trying to register under. Upon clicking the link, the user is sent to an account confirmation page, which explains that the account has been confirmed and the user may now log in.

Registering a new account leads to the ‘Log In’ tab. Selecting’ Log In’ generates a screen that allows the user to input his or her email and password. The user can select ‘Remember me’ so that if the web page is closed, when the user returns to the site later, he or she will still be logged in. Selecting ‘Log In’ will attempt to log the user in with the inputted

information. At the bottom, they are given the option the ‘Register as a new user’ if they do not already have an account, which will lead the user to the Register page, as discussed in the previous paragraph. Selecting ‘Forgot your password?’ takes the user to a screen which allows the user to input their email and get a password reset link. Clicking on the link from the email allows the user to input their username and a new password.

After logging in, the options on the right side of the navigation bar change to ‘Welcome!’ and ‘Log Off’. ‘Log Off’ simply logs the user out and redirects to the home page as an anonymous user. ‘Welcome’ takes the user to an account management page which allows the user to change his or her password.

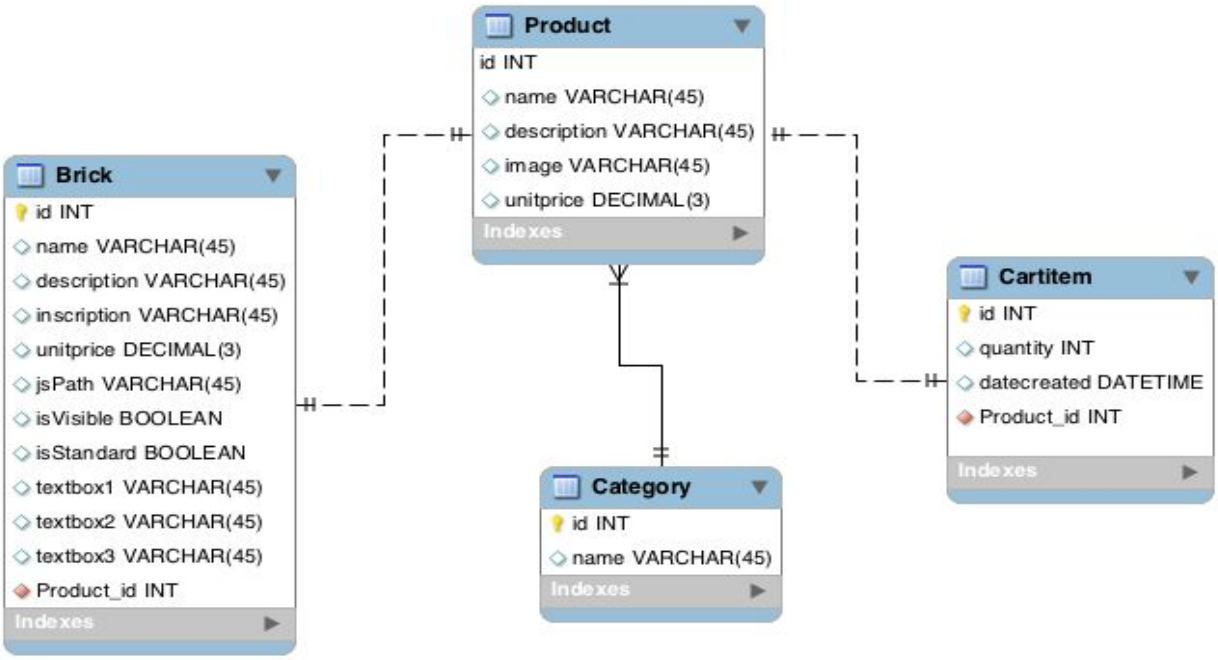
Another aspect of the website allows for an admin to make edits. Upon logging in as an admin user, an additional tab appears on the navigation bar called ‘Admin’,’ with features only available to those who have access to an admin account. Upon logging out of an admin account, the admin tab disappears. Clicking on the admin tab takes the user to the admin page where there is the ability to manage the available models.

Admin users are able to upload the javascript file that renders their own 3D model. Selecting ‘Choose file’ will allow the admin users to browse their computer’s contents and upload a file that renders whatever 3D model they have created. Admin users can only upload Javascript file because this is how our site has been set up to render. Upon choosing the file, the user clicks ‘Upload’ and the new model will appear at the bottom of the page. The picture will say ‘New upload’ and the fields will initially contain default values. The model will initially be hidden from all users because the values are set to defaults and these should be customized before launching the product to avoid any confusion.

There is also a listing of the available models on the admin pages. Each model has a title, a picture, a price and three buttons beneath. The title is changeable by simply typing in the box available. The picture matches the picture in the Custom Objects database so that the admin can distinguish between object. Clicking on the picture will take the admin to the page on the site that displays that particular product for sale. The price is also changeable by typing in the given box labeled ‘Price’ located beneath the model. If anything on the model is changed, the admin has to click ‘Update Model’ beneath the model in order to launch the changes on the site. By clicking ‘Hide,’ the admin will make the product temporarily unavailable to users. This is useful if they are out of a product, yet plan to restock. When an admin clicks hide, the site will update and the product will no longer be listed in the Custom Objects page or as an option in the drop down menu under Base Models. Upon clicking Hide, the button changes to ‘Show.’ Clicking show will make the product again available for all users and it will reappear in all the places it was located before clicking Hide. Lastly, ‘Remove’ deletes the product from the database and makes it no longer available to users or available to admins on the Admin tab. This would be useful to an admin if they plan to never restock an item again and do not want other admins accidentally showing it to users. If it is removed by accident, it can always be added back using the upload feature.

Database Schema

The following illustrates all of the major tables in the database.



Technical Obstacles Overcome

We primarily utilized C# and asp.net for the backend, javascript for the front-end, and WebGL for the 3D rendering. The website is deployed using Microsoft Azure Services. Throughout the project, we encountered numerous obstacles.

One of the major obstacles was the rendering of the 3D model into the asp.net webpage. During our research, we did not find any implementation involving webgl with the asp.net webpages. It had been difficult to simply display the brick model on the page, let alone displaying the text layers on top of the brick model. Eventually, we figured out that by creating a div element that refers to the container in the webgl js file and styling that div element with various margin attributes, the Missouri brick model could be placed into the correct position.

Our next step involved building more models similar to the Missouri brick models such as the Gamecock cube and a Clemson plaque. Aside from just creating the models, we also had to deal with rotation and zooming issues that needed to be customized with each model. To create these models which were different from the detailed model that we rendered for our client, we had to create 3D objects and then map a texture image to the object. For example, we created a rectangular prism for the brick, and then mapped a brick-like texture image to the brick to generate what is now available on the site.

Placing text on top of the brick model was also challenging. We had to make sure that the text layer stays within a reasonable distance above the brick model and does not look awkward when the brick is rotated. We had to attempt many different methods before settling on one because it was difficult to draw the text layer on top of a mapped image. There were certain solutions that allowed for text on a 3D object, but did not allow for text on an object that was mapped to with an image, or a solution that allowed for text on a mapped image, yet did not allow this text to be 3D. After many attempts, we settled on the current solution. Each textbox is its own 3D model that is the same shape as the model itself, yet slightly larger so that it will sit outside/on top of the model. For the brick and plaque, the boxes are only larger on 2 sides, whereas for the cube the 3D text object's box is bigger on all sides to all the text to be visible on all sides. The issue of making the text 3D also involved more text boxes. First, we rendered the object and mapped an image to it. Then we drew a layer for the text background shadowing, which made the text looks 3D. Then finally we rendered the actual text layer on top of this. The background shadowing is simply the same text, yet in a dark gray. The order in which these objects are rendered is what creates the illusion that the text is three dimensional. This all

occurs at runtime, so the user cannot tell these objects are actually being layered over each other. Instead, it looks as if it is just an object moving together as one.

The movement of the object was also a difficult task, as we had many layers of boxes. Getting all of these layers to move together in sync when the mouse moves seemed difficult. There are EventListeners that will call a function when the mouse clicks, and when the mouse drags. Instead of just moving the brick box, the text boxes also have to move since these are technically separate entities. We therefore had to refer to each box in the EventListeners to make sure that they would move in sync.

We also decided that it was important for user to be able to choose different font for the text. At first, the text style would not change even though we had download all the js files associated with each font style. Fortunately, we were able to solve this problem in an easier way by simply calling the name of the text in our webgl code without using the font-specific js file. Later, as we did more testing, we realized one of our chosen fonts French Script was a Windows safe font and did not always work on Mac computers, so we switched over to another font style Courier.

The database design was not as straightforward as we initially thought. Originally, we thought the database would contain only two tables: category and products. The category table contains the name of the school the product belongs to, such as Clemson University and University of Missouri, while the product table has fields such as product name, description, image, and price. As we progressed further in the project, we kept adding fields into the product table and ultimately decided having two simple basic tables was not enough for the project. When we added the custom models into the existing product, we realized that these models have to exist not only in the product table but also under their own table that contains the name

of the corresponding webgl js file and the inscription text. Since this is a prototype for the ZVerse shopping site, we also allowed users to order our products and customized brick model. Therefore, we created a shopping cart that accessed the database to keep track of all the items the user had added to the cart.

One of the biggest issues we have faced was dealing with customized models with text. Adding a regular product to the shopping cart is trivial because the product already exists and we can keep track of the product using the product id. On the other hand, even though the customized model with no text exists in the database, there was no way of tracking the customized model with different user-input text. There was much deliberation over the right approach to solve this problem, but we finally decided to create a new brick model every time the user inputs text and adds the brick model onto the cart. Because we inserted a new brick model into the database, we also had to ensure such models with user-input text would not become visible on the custom model catalogue and the admin page. Therefore, we implemented another field called isStandard to differentiate between the standard brick models that the admin uploads and the custom models that the user creates when he or she adds to the shopping cart.

Even the admin page requires touching the database. As mentioned earlier in the overview section, the admin can decide whether to show or hide the available models on the public domain. Initially, we thought a simple jquery click would enable the admin to show or hide the models. Unfortunately, after the admin logged off, the show and hide features disappeared and the models became available by default. The only way to ensure that the admin changes are saved was to add an isVisible field to the brick model such that the show or hide option is

saved into the database. In this way, after the admin logs off, the model on the public webpage will be displayed only if the value of the isVisible field for that particular model is set true.

Conclusion

Overall, the project is quite successful in building a website that incorporates both the visualization of 3D models and user-friendly shopping functionality. In the future, we hope to allow more customization for the user when creating the brick model, such as adding more font style options, allowing horizontal movement of the text, and changing the colors of the brick. In addition, we can also simplify the admin user's tasks by creating a default simple template for admin to upload models or allowing the batch upload of products. We achieved the functionality required by the project and our client, however, and contributed significantly more features to the site than the client asked for.