15-112 Term Project Proposal

Harvey Zheng | harveyz

Project Description

Name: **WebSketch**

The project will allow you to lay out different components to mockup or design a website, and export it to a static page with actual HTML and CSS.

Competitive Analysis

Currently, there are a number of editors that allow you to lay out components for a website, and it will give you the HTML, CSS, and any JavaScript the site may need for things like carousels. These include companies like Mobirise, Bootstrap Studio, Adobe Muse, or SquareSpace.

Mobirise uses pre-existing blocks that the user can edit and maneuver in a static page. Similarly, my project will generate a static page from dragging in already existing blocks. The difference is that Mobirise uses a grid-like system, where things are placed above and below each other, never on top of each other.

Bootstrap Studio also uses pre-existing components built on top of the Bootstrap CSS framework. Since the app uses Bootstrap, they natively use Bootstrap's grid system for the arragement of components. Once again, you cannot overlay things on top of each other, like with layers.

Adobe Muse was software within the Adobe Creative Suite that generated static pages from laying out components onto a canvas, and is not built on any existing web technologies. This is probably most similar to what I aim to build, although my project will be significantly simpler and won't have as many features. Objects can free-float, and lots of custom stuff can be added.

SquareSpace is a popular site builder that uses prebuilt components in a certain order, and its components have limited edits it can make. However, its simplicity is something I'd want to replicate, since it is very easy to design and publish a website with.

Structural Plan

The project has these main parts:

1. Animating GUI for a user to lay out their website with
2. A way to store the objects onto a canvas and have it be retrievable later
3. A way to export the objects into an HTML and CSS files.

And will probably be organized into three different files with these parts.

The GUI

The GUI will include a few different tools that will create or edit objects on the canvas. This will include:

- A color palette, for the current selected color
    - User clicks on a box, inputs a hex code, and the app will store it.
- A create rectangle (div) tool, which will need to create rectangles of different sizes based on where the user drags it. You will also have the ability to enter text into the rectangle.
    - Color is based on the color in the palette. Will have a view aspect (dragging rectangle), a controller aspect (manipulating the rectangle's model from the view), and a model (where its data is stored).
    - There will be two types of divs: absolute and relative. Divs will have children.
- A text tool that inputs text where you want it
    - Font, fontsize, and color are user inputs. Will also be stored in a class. Will appear in the
- A way to put in images and resize/distort them.
    - Input path to image file
- An align tool to align up to two selected objects
    - Probably end up just being a method that is activated when clicked.
- A cursor tool to select objects to be moved, scaled, or deleted.
    - Will need to store the objects that are shift-clicked into a list.

Additional features to implement:

- Warning indicator, such as for an invalid image path/color/font
    - If an exception is thrown, display it and remove it after a certain period of time
- Bounding box around selected object(s) + resizing capabilities.
    - Box just goes around the selected object(s)
    - Method detects corner collision, and object is adjusted based on how the mouse move
- Cleaner text input than the animation framework provides (more like a normal text editor)
    - Keep track of position of what's being edited; splice edited string into two, and move based on where the cursor is
- Inputs for categorizing divs as absolute or relative, and a way to assign a child. Would also need to check if a child is valid.
    - Probably just buttons on the sidebar

Storage

Storing objects should be done in a way that makes it easy to export to HTML and CSS. Each type of component will get a different class (text, div, and images). Classes for each CSS class will also be created, keeping track of properties. A class variable for CSSClass will keep track of every object created.

Exporting to HTML and CSS

This will likely be a function or two that converts the stored objects into HTML and CSS based on the attributes of the objects, and writes into HTML and CSS files based on the position of the elements in space or its layer. Will loop through everything and assign classes accordingly.

Algorithmic Plan

The hardest part of the project will probably be how I convert objects laid out on a canvas into HTML and CSS from a Python animations library. The best way to do this would be to properly store objects and make sure they have the right attributes. The end goal of the project is to have objects be relatively positioned on the web page. All the objects will be stored in a list, with their layer corresponding to their index.

The overall page will be semi-dynamic: the components will all be scaled based on the width of the page, so to properly convert elements from the canvas to html, the widths will be scaled as percentages. Heights will stay.

Divs will be capable of having children, or components that belong to it in the hierarchy, such as in the example below:

```
<div>
    <img src="image.png"/>
</div>
```

Divs will be able to be chained theoretically infinitely. To do so, the Div class will have an attribute (most likely a list) that stores its children, if they are one level below. For handling multiple levels, they will be handled recursively, e.g. the children's children of a div. Assigning children to divs will be handled as follows:

There will be two types of divs. One is absolute, the other is relative. If a div is relatively positioned, it will have the ability to select elements above it in layers and make them children. The div will then check the distance between its boundaries and the object and assign padding based on that. If objects are aligned, then they will be put inside their own invisible div, and flexboxes will be used to make sure they stay properly aligned.

A div's attributes will include any text as a text object, its position, and its color. Text objects will store its position, the content of the text, the kind of alignment, and instantiate a CSS class object that stores its font, font size, and color. If the CSS class object already exists, per a set of CSS class objects stored as a class variable, then it will use the already existing object.

As for converting the Python attributes into CSS classes, each Python class instance will have a CSSClass attached to it to keep track of its properties. CSSClass will have a class set to keep track of all CSSClasses.

When exporting, the overall list of objects is passed into the HTML exporter. It then goes through each element in the list, and writes in all the absolute values first. Then, for the relative components, it sorts the list by y position, and writes it into the HTML file. If there's a child, then it will store than and make sure the child isn't drawn again accidentally when going through all the components.

Timeline Plan

Date	Event	Notes
11/20	Ability to create rectangles and input text; color palette	TP1

Date	Event	Notes
11/22	Add text to rectangles and store input text attributes as CSS classes; resize objects; style text	
11/24	Align objects	
11/26	Export objects to HTML and CSS	TP2
11/30	Add relative divs and export to HTML adn CSS	
12/2	Stylistic stuff (bounding box around selected object, icons, etc.); warnings	
12/5	Finalized project	TP3

Version Control Plan

My plan to back up my code is to regularly push to a public GitHub repository.

Module List

- HTML
- CSS