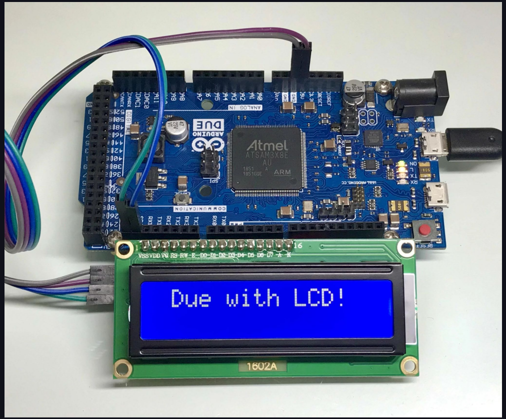
Adafruit GFX - Visual Editing Tool

Implementation Process - Brady Underwood

Background Knowledge

- Microcontrollers can automate tasks and hook up to a display to show data
- The most popular graphics library is AdafruitGFX compatible with 99% of displays
- In order to see the graphics on screen you have to compile and run the code taking **1+ minute** for each iteration



{ App Demo

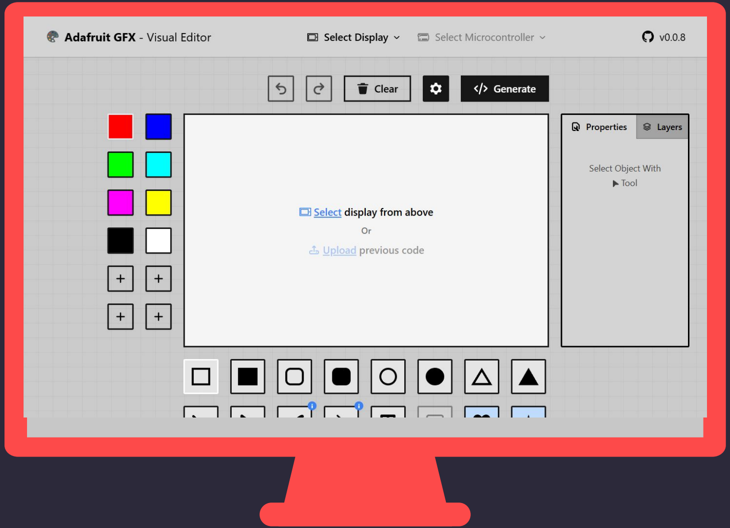
What is it?

A web-app that allows you to create basic designs

What does it do?

Converts designs into code compatible with microcontroller displays

[Link To Demo]



Select display from above
Or
Upload previous code

}

Frontend Implementation

Frontend – SVELTEKIT

*

SSR/SSG vs. CSR

Remix

NEXT.JS

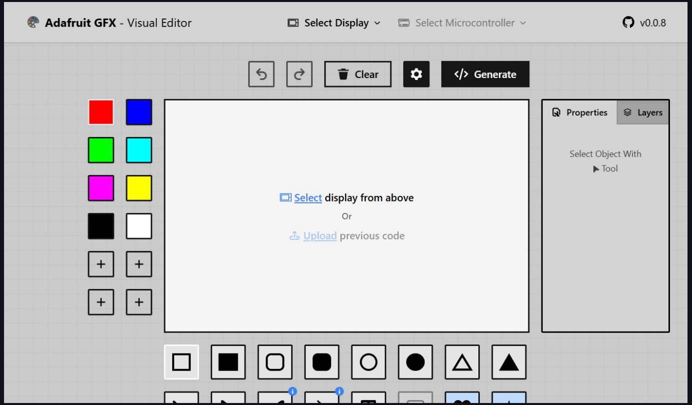
NUXTJS

React

Angular

Vue.js

Component Based Architecture



From Highly Reusable

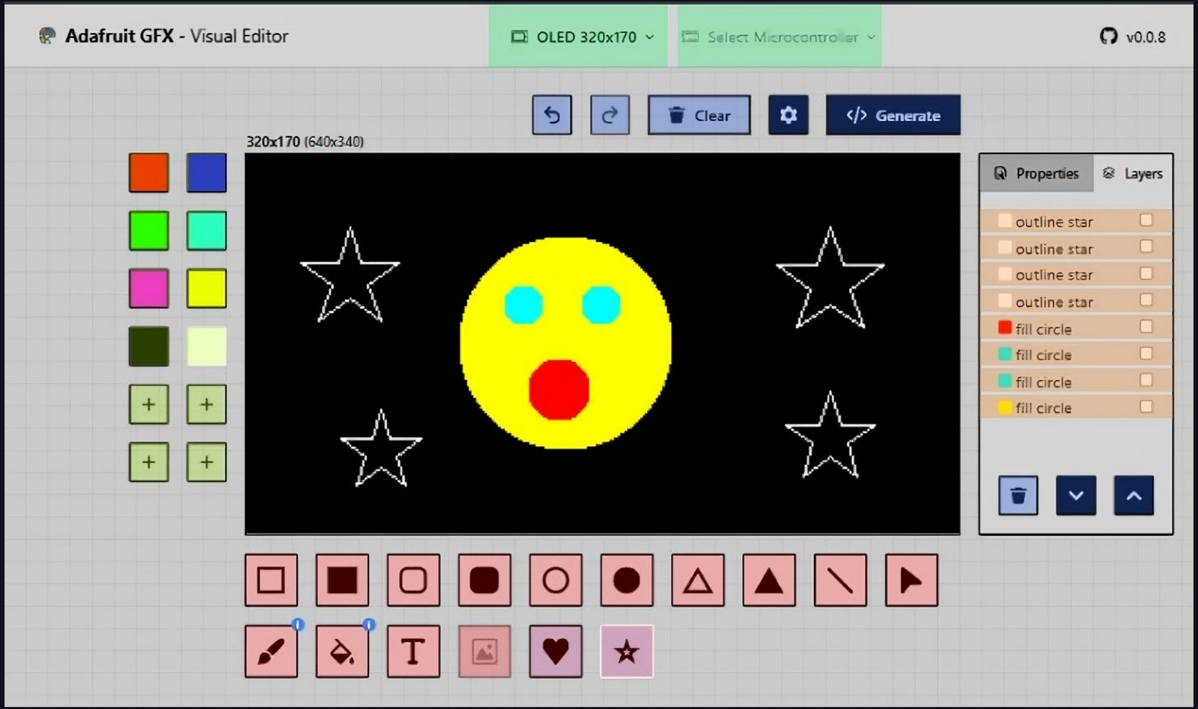
-Buttons

-Links

To Strongly Encapsulated

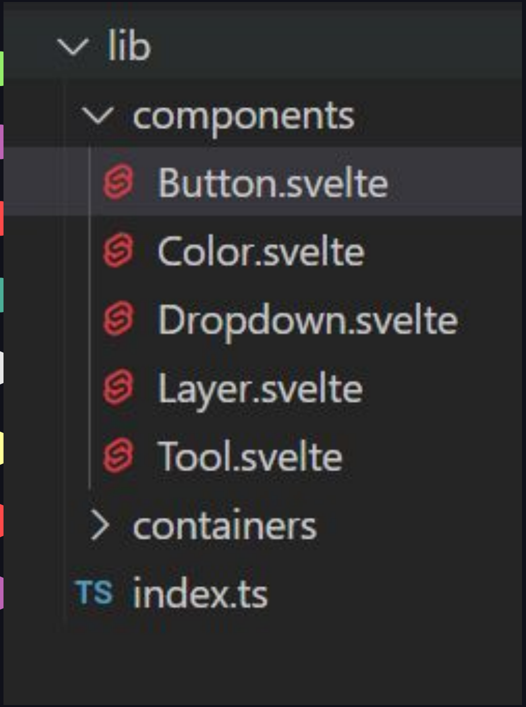
-Canvas

-Properties Panel



Unique Components Highlighted

Code Example #1



```
lib
  components
    Button.svelte
    Color.svelte
    Dropdown.svelte
    Layer.svelte
    Tool.svelte
  containers
  index.ts
```

```
<script lang="ts">
  export let icon: string = "";
  export let text: string;
  export let onClick: () => void;
  export let small: boolean = false;
</script>

<button
  on:click={onClick}
  class={""} //Styling for button
>
  <img src={icon} alt="Button Icon" class={small ?
"h-3" : "h-4"} />
  {text}
</button>
```

Data Structures + Algorithms

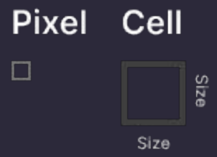
- 2D Array
- Classes (Inheritance, Abstraction, Interfaces)
- Geometry Algorithms
- Linked List/Stack
- Bitmaps

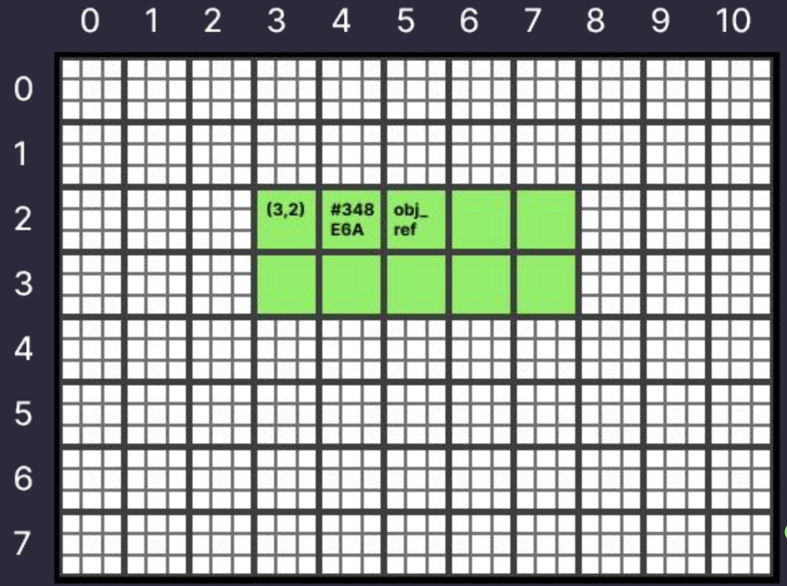
Canvas Implementation

Data Structure:

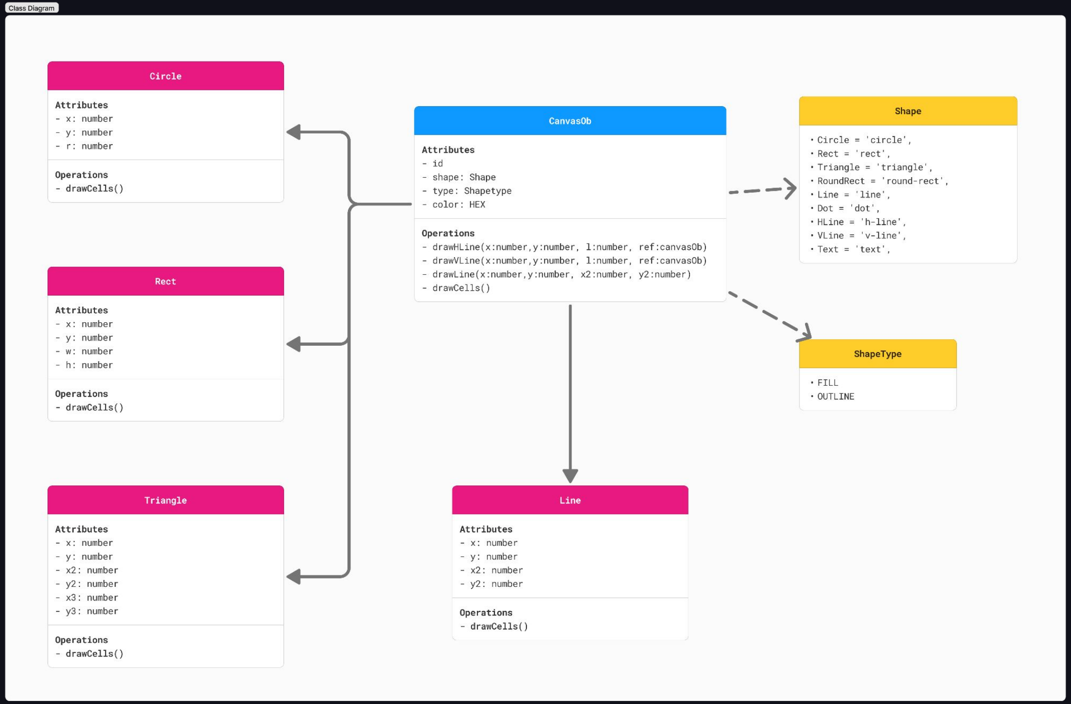
2D Array

```
let cellList: Cell[][] = [];


class Cell{
    readonly x:number;
    readonly y:number;
    readonly size:number;
    color:HEX;
    private _object: CanvasOb | undefined;
}
```





{ .. Object Class Implementation



Data Structure:
Classes, Inheritance

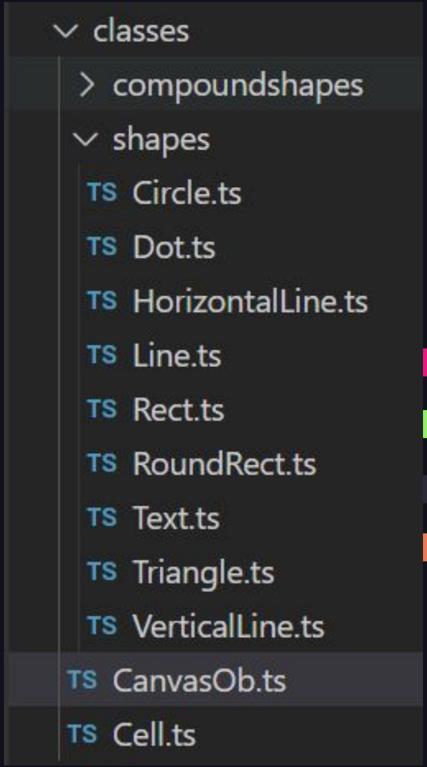
🟥 Class

🟦 Abstract Class

🟨 Enumeration

} ..

Code Example #2

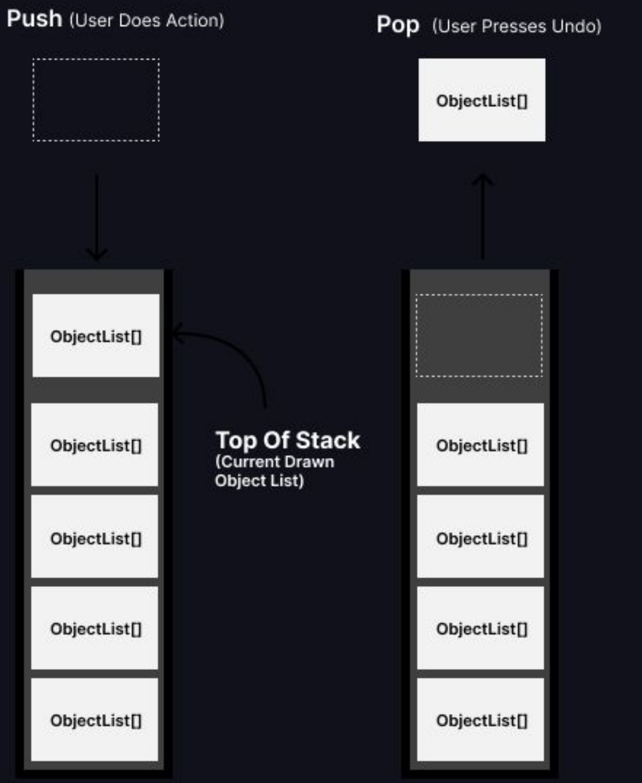


```
classes
  compoundshapes
  shapes
    Circle.ts
    Dot.ts
    HorizontalLine.ts
    Line.ts
    Rect.ts
    RoundRect.ts
    Text.ts
    Triangle.ts
    VerticalLine.ts
  CanvasOb.ts
  Cell.ts
```

```
class CanvasOb{
    readonly id: number;
    private static nextId = 0;
    shape: Shape;
    type: shapeType;
    color;

    constructor(shape:Shape,
type:shapeType, color){
        this.id = CanvasOb.nextId++;
        this.shape = shape;
        this.type = type;
        this.color = color;
    }

    /** Utility Functions */
…
```

```
class Circle extends CanvasOb {
    x: number;
    y: number;
    r: number;

    constructor(type: shapeType, x:
number, y: number, r: number, color:
HEX) {
        super(Shape.Circle, type,
color);
        this.x = x;
        this.y = y;
        this.r = r;
    }

    drawCells(cellList: Cell[][],
altRef?:CanvasOb) {
…
```

```
let objectListStatesWritable: [CanvasOb[][], number];
let objectList:CanvasOb[];
```



Undo/Redo Implementation

Data Structure:

Linked List/Stack

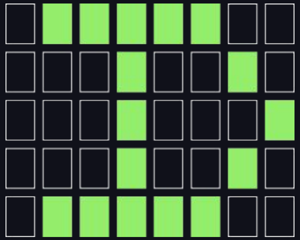
Text Bitmaps

Data Structure:
Bitmaps

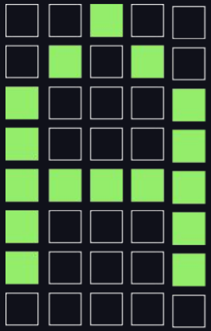
“A” -> Find in Bitmap -> Convert to pixels -> Draw

-Position 85

```
01111100
00010010
00010001
00010010
01111100
```



```
let bitmap = [
   0x3E, 0x51, 0x49, 0x45, 0x3E,
   0x00, 0x42, 0x7F, 0x40, 0x00,
   0x72, 0x49, 0x49, 0x49, 0x46,
   …
]

Let lookUpTable = [
      'A': 85,
      'B': 90,
      …
]
```



Thanks!

< Do you have any questions? >