Process & Decision Documentation

Project/Assignment Decisions

For this side quest, I was inspired by the classic internet creepypasta named “The Backrooms”, which is a fictional space that does not exist in reality, and is composed of seemingly endless interiors that tend to be eerie and oddly empty. I wanted to customize the maze code into a backroom styled game, where the player can continue to loop through the levels endlessly, while witnessing some unusual things happen as they go.

I also wanted to incorporate my art into this, and instead of having colored walls or circle characters, I made some pixel art sprites to replace the solid colors, making the game feel more immersive.

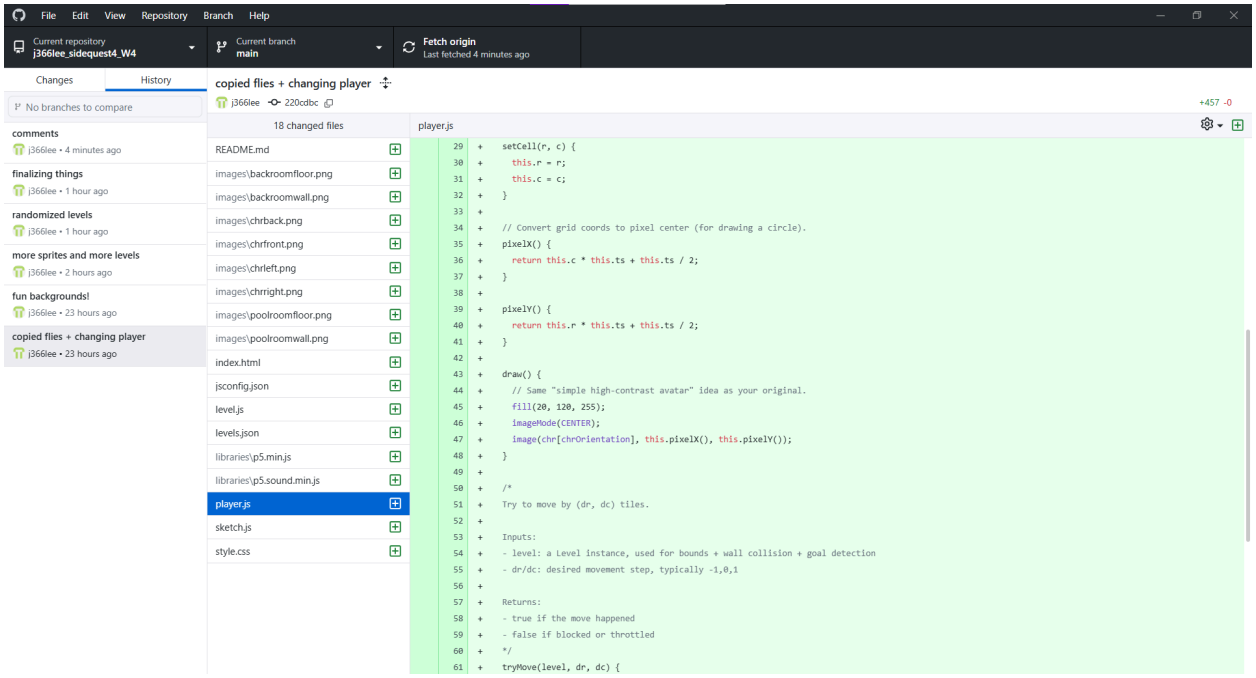
GenAI Documentation

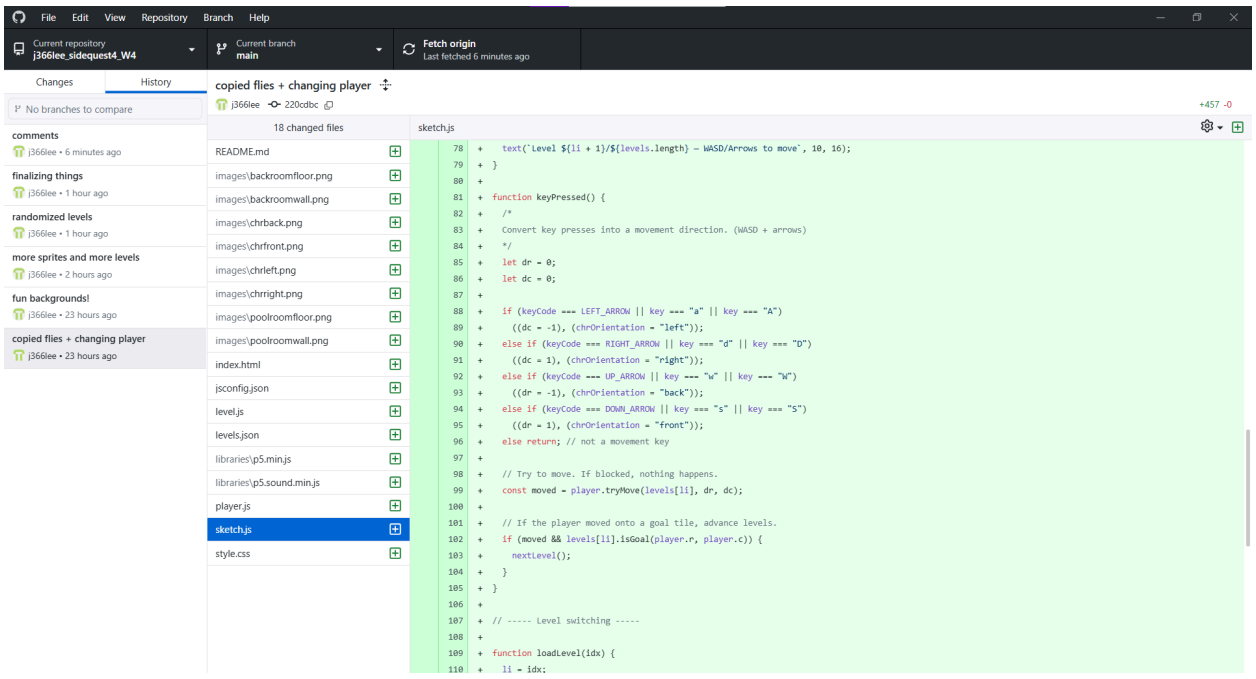
No genAI was used in my addition to the original example code.

Summary of Process

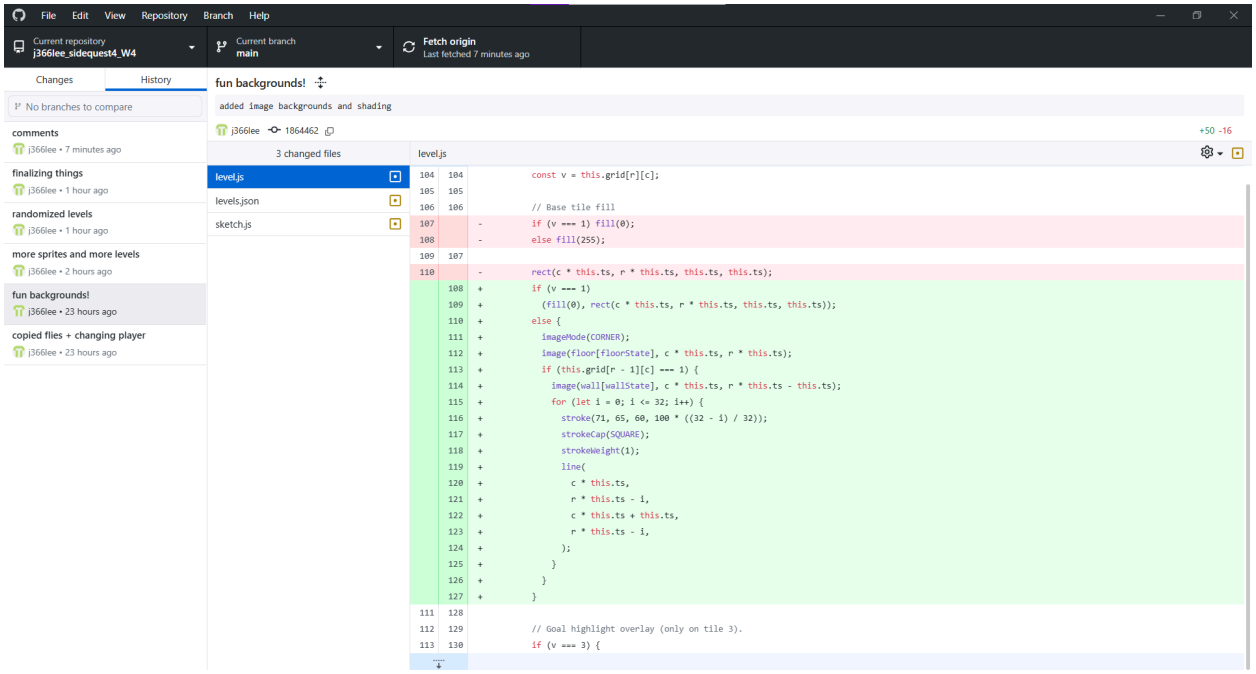
All of the repo changes are available on the github commit page.

The first step I did was copying the files onto my own repo, and changing the player sprite. The player sprite was uploaded to an image folder and then loaded in the preLoad() function. When a key is pressed, the player sprite also changes directions.

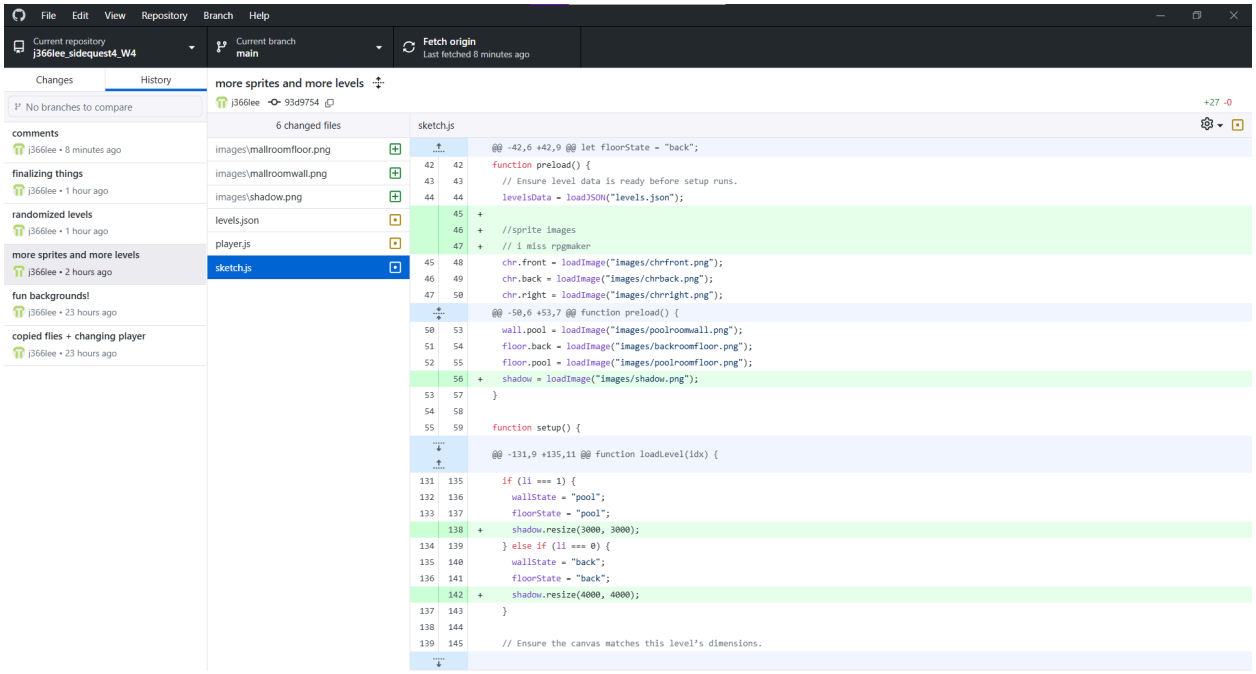




The second commit added the first 2 backgrounds in, as well as changing up some levels to be different from the example file.



The third commit was similar. It added more sprites and more levels, as well as the shadow that prevents the player from seeing the entire map (and figuring out the puzzle quickly). The shadow increases in size as the player progresses.



The fourth commit made it so on the 3rd level, the walls become randomized as the player moves. Something notable that happened in this commit was that my code would break.

In a previous commit, I made this code that will show a wall if the floor is touching a tile that is not a floor on the top.

```
imageMode(CORNER);
image(floor[floorState], c * this.ts, r * this.ts);
if (this.grid[r - 1][c] === 1) {
  image(wall[wallState], c * this.ts, r * this.ts - this.ts);
  for (let i = 0; i <= 32; i++) {
```

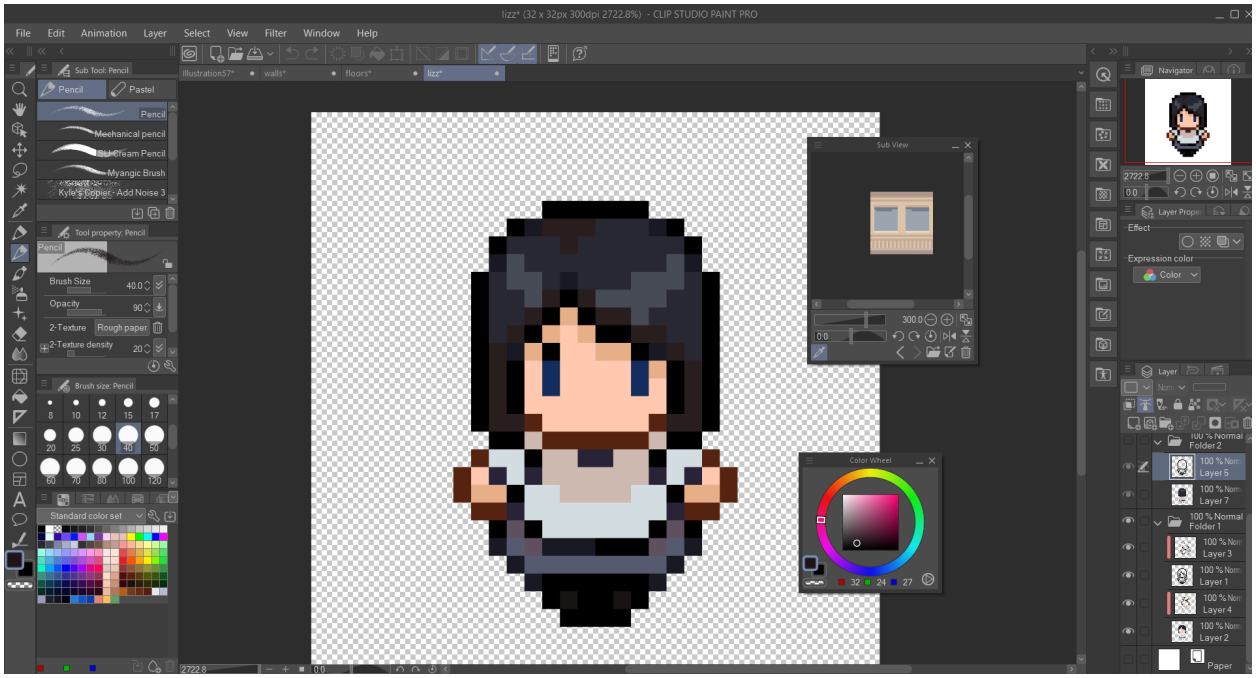
A major issue in this code I didn't notice though, is that if r is 0, the code will not function as grid[-1] is not valid. However, I did not have a problem until this commit as there would be no case where r would be 0, and the grid placement would be on 1, as the top row in my json file levels were 1.

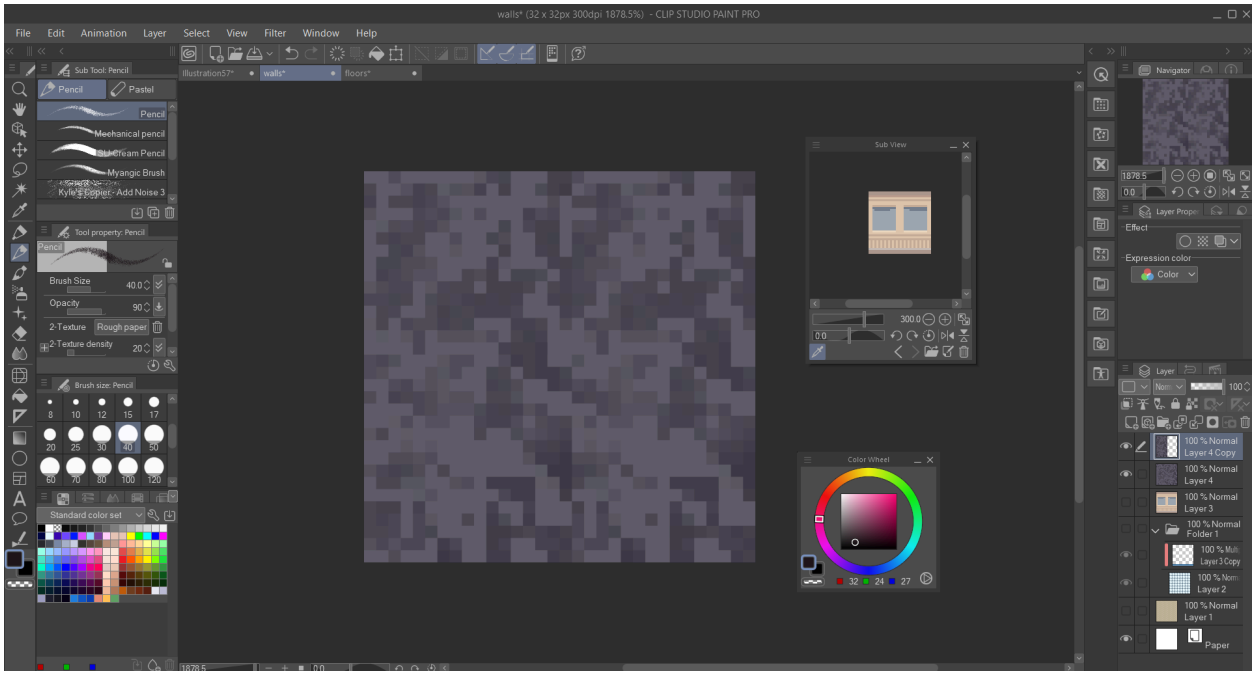
However, I let it so the randomization code could turn a tile in the first row into a floor tile. This was fixed, but the (let r = 1; part was a 0 instead of 1 when the error happened.

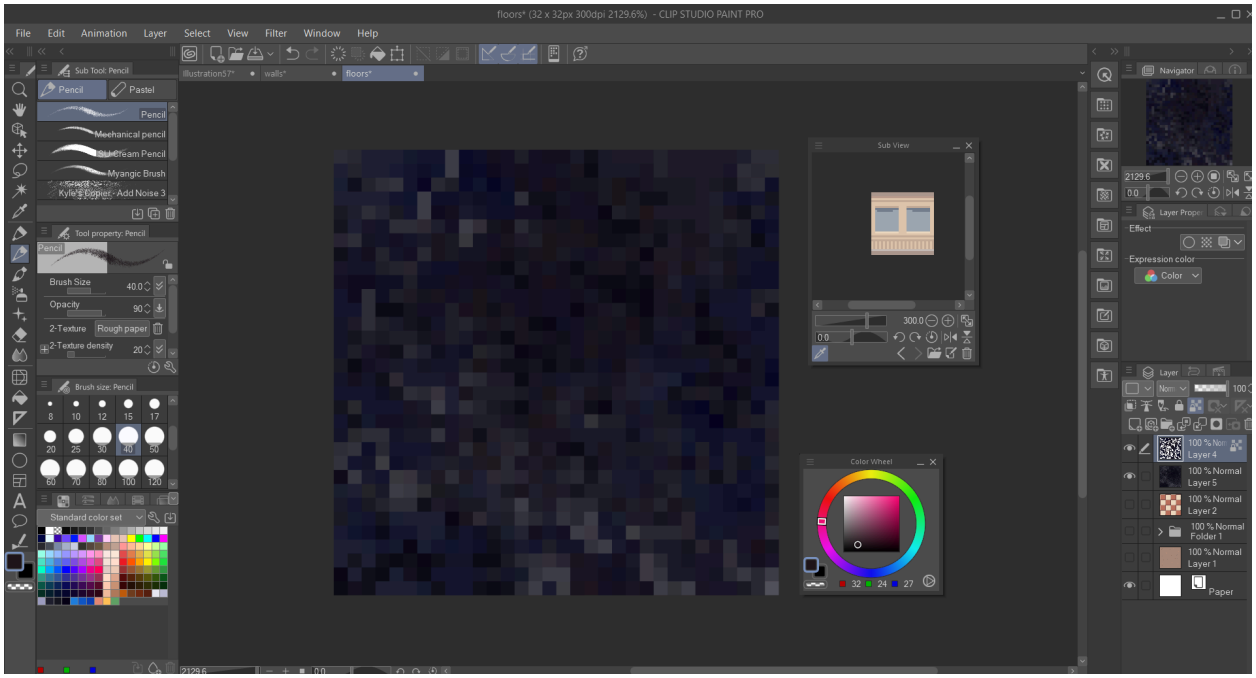
```
let randomStage = levels[2].grid;
for (let r = 1; r < randomStage.length; r++) {
  for (let c = 0; c < randomStage[r].length; c++) {
```

The last two commits were just comments and an extra sprite addition for the final stage, as well as finalizing some details.

In addition, I included screenshots of the sprites I created. Layer information can be seen on the right side.


lizz* (32 x 32px 300dpi 2722.8%) - CLIP STUDIO PAINT PRO
File Edit Animation Layer Select View Filter Window Help
Sub View
Color Wheel
Navigator
Layer
Paper


walls* (32 x 32px 300dpi 1878.5%) - CLIP STUDIO PAINT PRO
File Edit Animation Layer Select View Filter Window Help
Sub View
Color Wheel
Navigator
Layer
Paper


floors* (32 x 32px 300dpi 2129.6%) - CLIP STUDIO PAINT PRO
File
Edit
Animation
Layer
Select
View
Filter
Window
Help
Sub Tool: Pencil
Pencil
Pastel
Mechanical pencil
Cream Pencil
Myangic Brush
Kyle's Dopier - Add Noise 3
Tool property: Pencil
Brush Size
Opacity
2-Texture
Rough paper
2-Texture density
Brush size: Pencil
Standard color set
illustration57*
walls*
floors*
Sub View
Color Wheel
Navigator
Layer Proper
Effect
Expression color
Color
Layer
100 % Normal
Layer 4
100 % Normal
Layer 5
100 % Normal
Layer 2
100 % Normal
Folder 1
100 % Normal
Layer 1
Paper