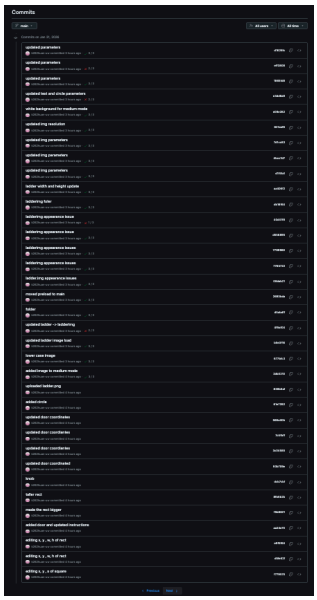
Process & Decision Documentation

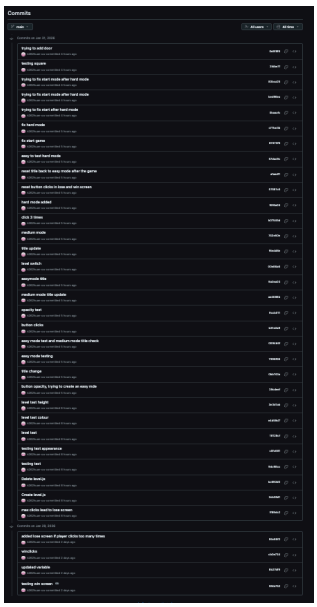
Project/Assignment Decisions

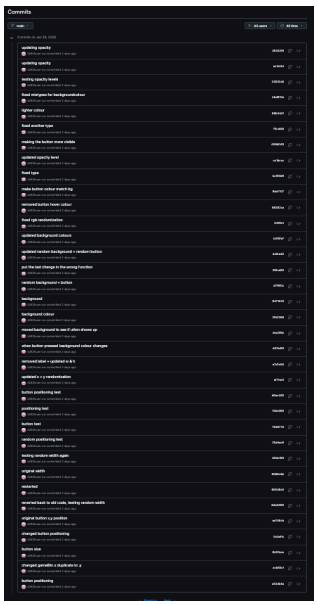
The decisions I made were to split the 3 game mode sections, easy, medium, and hard,d into 3 JavaScript files. I made this decision to make it easier to work on each section of the game; in the end, it made it easier to keep track of what I wanted to display on each game mode section.

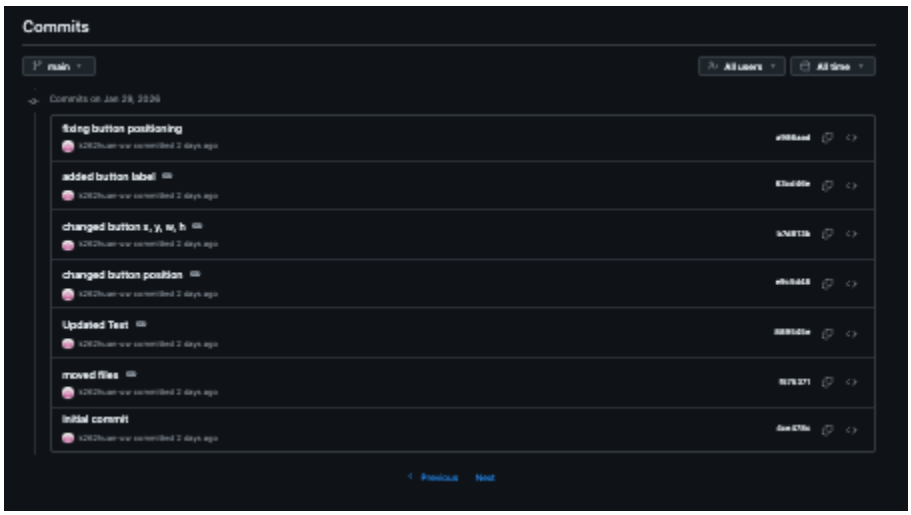
Role-Based Process Evidence

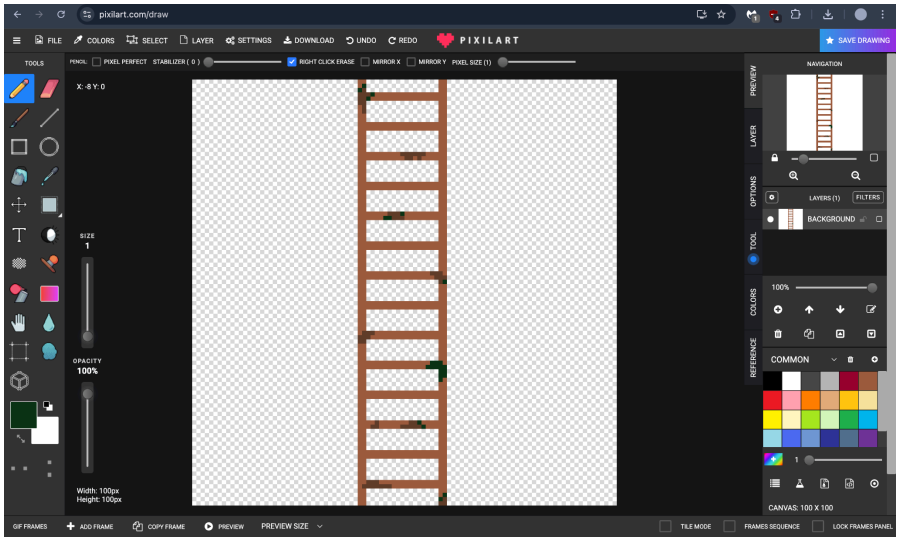
Made commitments 161 times. The commitments include mostly debugging, updating parameters, and testing. Included screenshots of creating graphics.

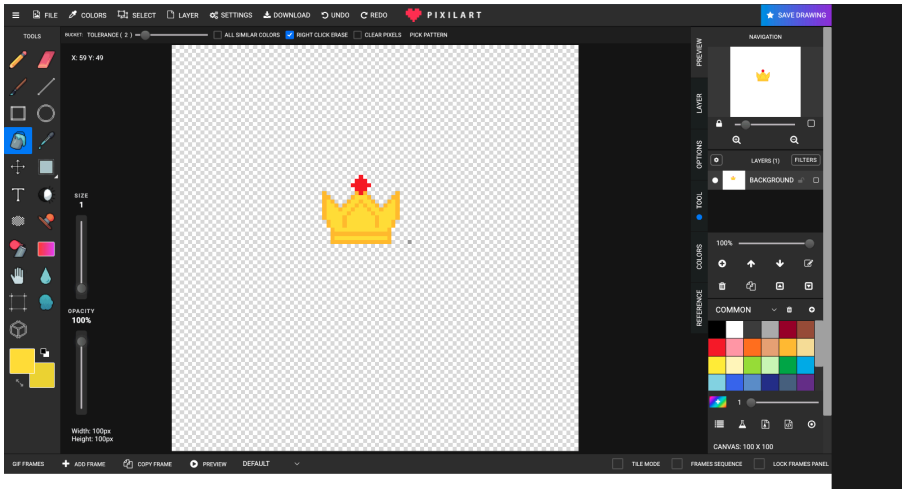


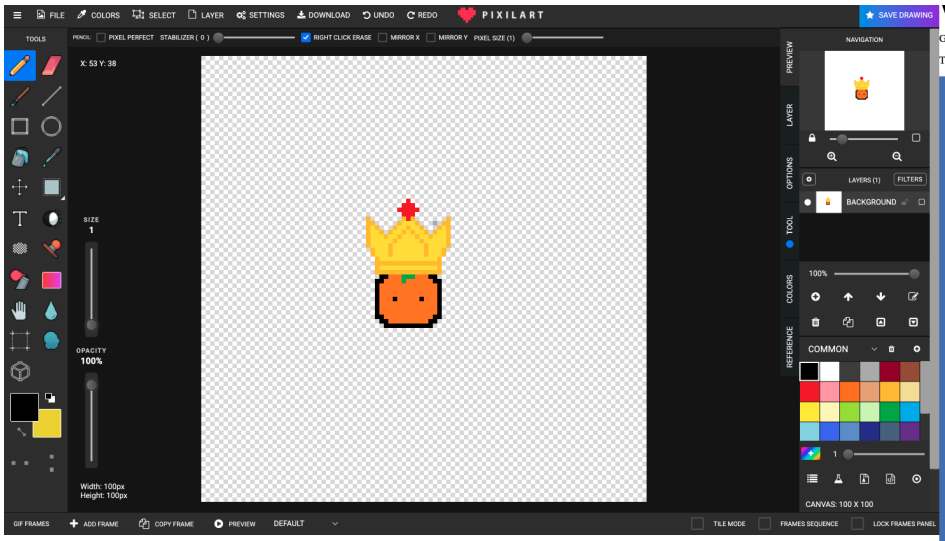












GBDA302

Tip: use En

© 2026 Dr. K

Entry Header

Name: Katrina Huang

Role(s): Designer & Programmer

Primary responsibility for this work: Designed graphics, debugged code, and created code

Goal of Work Session

To create multiple game states through JavaScript files

Tools, Resources, or Inputs Used

ChatGPT 5.2

GenAI Documentation

GenAI was used for Side Quest 3.

If GenAI was used (keep each response as brief as possible):

Date Used: January 31

Tool Disclosure: ChatGPT 5.2

Purpose of Use: I used GenAI to debug my code

Summary of Interaction: Support tool to debug code

Human Decision Point(s): Modified code when I noticed it had the wrong input

Integrity & Verification Note: Used my past knowledge from previous courses and tested if the code worked.

Scope of GenAI Use: Creating graphics, images, shapes

Limitations or Misfires: Debugged the wrong area of code and gave tips that didn't work

Summary of Process (Human + Tool)

- Tested and made updates to the code
- Created graphics
- Debugged code

Decision Points & Trade-offs

- The options I considered were to keep everything on the game page or to split it up into multiple JavaScript files. I decided to split it up into multiple files to make everything easier.
- The options I considered were to use graphics online or create my own. I decided I wanted to create my own for practice to use for future assignments.

Verification & Judgement

I evaluated whether my judgment was appropriate based on the future asks of this course and how it would help me improve my JavaScript skills.

Appendix

Prompt:

Why does the start screen freeze when I press the start button?

ChatGPT 5.2:

Ah! I see exactly what's happening — the start screen "freeze" isn't a p5.js problem; it's a logic/load problem in how your mouse clicks are being routed.

Here's why your START button isn't working:

1️⃣ The `startMousePressed()` coordinates are redefined each draw call

In your `drawStart()` you define:

const startBtn = { x: width / 2, y: 320, w: 240, h: 80 };

But in `startMousePressed()` you redefine them again. This isn't inherently wrong, but if `startMousePressed()` runs before the mouse is over the button (or `isHover()` fails), it will never change `currentScreen`.

✅ Fix: define startBtn and instrBtn globally in `start.js` so they are the same object for drawing and hover checks.

2️⃣ Make sure `isHover()` is called correctly

Right now, in `startMousePressed()`:

```
if (isHover(startBtn)) {
  currentScreen = "easy";
}
```

- `isHover()` uses `mouseX` and `mouseY`.
- Make sure your button coordinates match rectMode(CENTER) (which they do).
- Also, ensure your canvas isn't inside a container that offsets mouse coordinates. If it is, `mouseX`/`mouseY` might not match the visible button.

3️⃣ Use global buttons in start.js

```
// start.js
const startBtn = { x: 400, y: 320, w: 240, h: 80, label: "START" };
const instrBtn = { x: 400, y: 430, w: 240, h: 80, label: "INSTRUCTIONS" };

function drawStart() {
  background(180, 225, 220);
```

```
  fill(30,50,60);
  textSize(46);
  textAlign(CENTER, CENTER);
  text("Win or Lose", width/2, 180);

  drawButton(startBtn);
  drawButton(instrBtn);

  cursor(isHover(startBtn) || isHover(instrBtn) ? HAND : ARROW);
}

function startMousePressed() {
  if (isHover(startBtn)) {
    currentScreen = "easy";  // This switches screens
  } else if (isHover(instrBtn)) {
    currentScreen = "instr";
  }
}
```

Now the same objects are used for drawing and hover detection. This eliminates a common “button doesn’t respond” bug.

4️⃣ Make sure `easy.js` exists

`main.js` routes to `drawEasy()` when `currentScreen === "easy"`. If `drawEasy()` is missing or has an error, it looks like the screen froze.

For testing, make a minimal `easy.js`:

```
function drawEasy() {
  background(200);
  textSize(32);
  textAlign(CENTER, CENTER);
  text("Easy Mode", width/2, height/2);
}

function easyMousePressed() {
  console.log("Clicked in easy");
}
```

If this works, then the start button is functional and the problem was in your `drawEasy()` logic.

If you want, I can rewrite a ready-to-drop start.js + easy.js pair that fixes the freeze and will let you transition smoothly to Easy → Medium.

Do you want me to do that?

Prompt: My code is still stuck on the start screen

ChatGPT 5.2:

You're stuck because **your easy screen is constantly forcing the screen state back**, and your click logic is wrong.

Here are the **only two problems** and the **exact fixes**.

❌ PROBLEM 1: You reset the screen every frame

This line **locks you on easy forever**:

currentScreen = "easy";

`drawEasy()` runs **every frame**, so even if you change screens on click, it immediately snaps back.

✅ FIX

DELETE this line from `drawEasy()`

currentScreen = "easy";

> Screens should change **only in mouse/keyboard handlers**, never inside draw functions.

❌ PROBLEM 2: Your click condition is backwards

You wrote:

```
if (easyClicks < 3) {
  easyClicks = 0;
  currentScreen = "medium";
}
```

That means:

- On **first click**, `easyClicks === 1`

- `1 < 3` → true
- It immediately jumps (or breaks logic)

You want **3 clicks**, not less than 3.

✅ FIX

Change it to:

```
if (easyClicks >= 3) {
  easyClicks = 0;
  currentScreen = "medium";
}
```

✅ FINAL corrected `easyMousePressed()`

```
function easyMousePressed() {
  if (isHover(gameBtn)) {
    easyClicks++;
    randomButton();
    randomBg = color(random(255), random(255), random(255));

    if (easyClicks >= 3) {
      easyClicks = 0;
      currentScreen = "medium";
    }
  }
}
```

✅ FINAL rule (important)

- `drawEasy()` → **draw only**
- `easyMousePressed()` → **change screens**
- **Never** set `currentScreen` inside a draw function

If you want, I can next:

- wire up `medium → hard`
- add clickable ladder / hole
- sanity-check `main.js` screen switching

Prompt: After the win screen when I press start again it goes straight to hardmode

ChatGPT 5.2:

Ah! I see exactly what's happening. The problem is that your **mode flags (`titleEasy`, `titleMedium`, `titleHard`) aren't being reset properly when the game restarts**.

Right now, when you win and start again:

- `buttonClicks` is reset to `0`
- But `titleHard` (or `titleMedium`) may still be `true` from the previous round
- Your `drawGame()` function checks the flags in this order:

if (titleEasy) { … }

if (titleMedium) { … }

if (titleHard) { … }

Because `titleHard` is still `true` from last time, Hard mode is drawn immediately, even though `buttonClicks` is `0`.

How to fix it

You need to **reset all mode flags when the game starts**, like this:

```
function startGame() {
  currentScreen = "game";

  // Reset counters
  buttonClicks = 0;
  totalClicks = 0;
  levelText = false;

  // Reset mode flags to Easy
  titleEasy = true;
  titleMedium = false;
  titleHard = false;

  // Reset button opacity
  opac = 120;
}
```

Then, instead of just setting `currentScreen = "game"` when pressing Start, call:

```
startGame();
```

Additional fixes

1. **Remove this line from `drawGame()`:**

let titleMedium = false;

let titleHard = false;

Right now, you are **declaring local variables inside `drawGame()`**, which are shadowing the global flags. That's why some modes behave weirdly. Use the global variables only.

2. Make your mode logic based on `buttonClicks` **only** inside `gameMousePressed()` — don't touch them in `drawGame()`.

✅ Summary:

- Global flags (`titleEasy`, `titleMedium`, `titleHard`) must be reset when starting a new game.
- Don't declare local variables with the same names inside `drawGame()`.
- Use `buttonClicks` to update the flags.

If you want, I can rewrite your `gameMousePressed()` and `drawGame()` so that **Easy → Medium → Hard works perfectly** and it always starts in Easy mode after restarting.

Do you want me to do that?

Prompt: Is the button opacity fine?

ChatGPT 5.2:

Ah — I see exactly what's happening with your button opacity. Right now you have:

fill(buttonOpacity(10));

The problem is **the opacity is extremely low**. In p5.js, the alpha (opacity) value ranges **0–255**, where 0 is fully transparent and 255 is fully opaque. You're using **10**, which is basically invisible. That's why the button is almost impossible to see.

✅ How to fix it:

If you want the button to **match the background color but still be visible**, use a higher alpha, like **150–200**:

fill(buttonOpacity(180)); // much more visible

And your `buttonOpacity()` function is fine:

```
function buttonOpacity(opacity) {
  return color(
    red(randomBg) * 0.5,   // darker than background
    green(randomBg) * 0.5,
    blue(randomBg) * 0.5,
    opacity                // controls transparency
  );
}
```

So basically:

- Keep the RGB multiplication (`* 0.5`) so the button is slightly darker than the background.
- Increase the alpha to **150–200** so the button is actually visible.

If you want, I can rewrite your `drawGameButton()` so the **opacity changes depending on difficulty** automatically, e.g., easy = 200, medium = 120, hard = 50.

Do you want me to do that?