

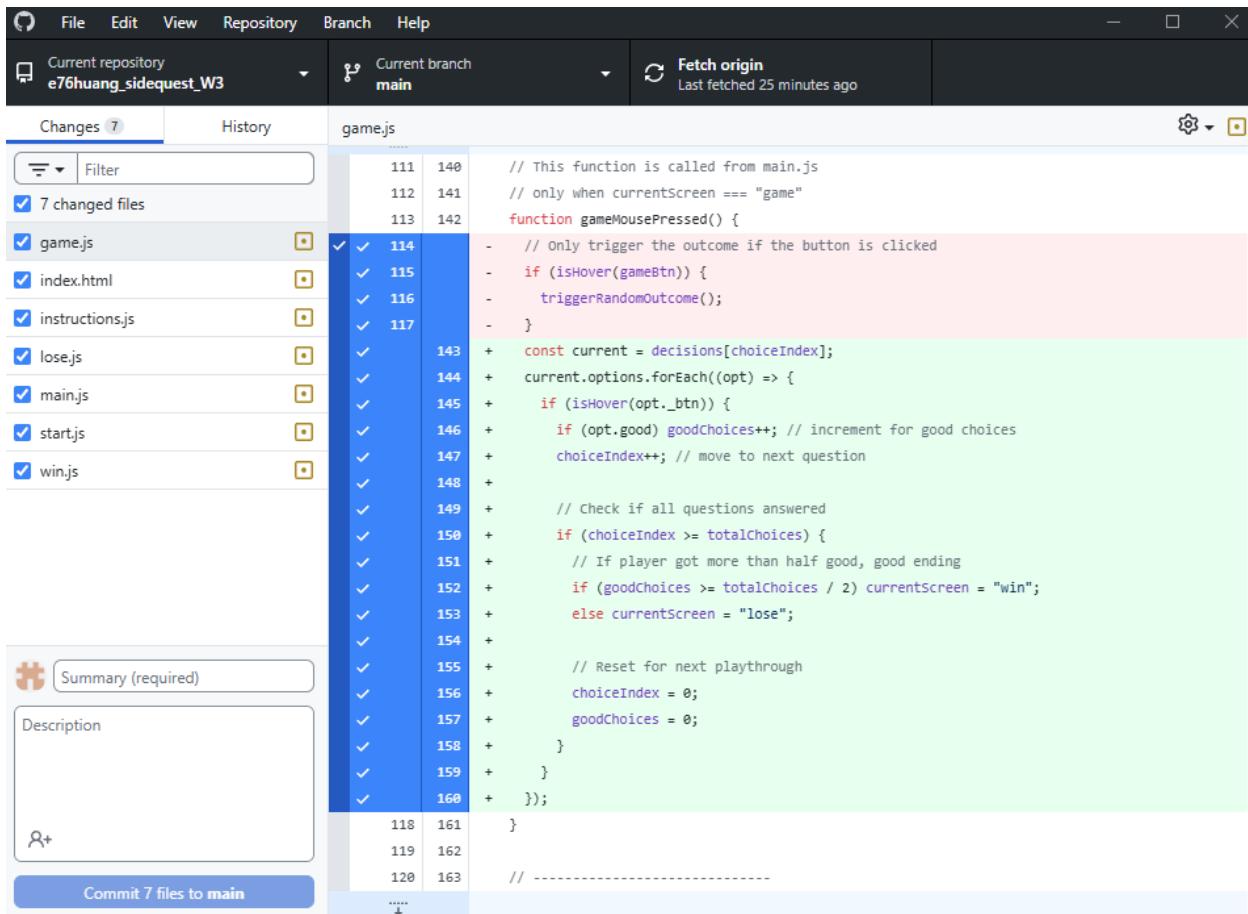
Process & Decision Documentation

Project/Assignment Decisions

The purpose of this assignment is to use the example activity code provided in Week 3's class to turn a probability game with a 50% chance of winning or losing into a "story game" that branches into different decisions. The game uses the same subsections of start, win, lose, end to track progress and produce a certain ending screen.

Using GenAI's assistance I was able to come up with a "Lost Puppy Adventure Story Game" with a good and bad ending depending on the player's accumulated choices and points.

Role-Based Process Evidence



A screenshot of the GitHub interface showing a commit history for a repository named 'e76huang_sidequest_W3'. The current branch is 'main'. The commit details show 7 changed files, with 'game.js' being the primary file. The commit message is 'Summary (required)'. The commit body contains a description and a 'Commit 7 files to main' button. The code diff for 'game.js' shows several lines of JavaScript code, with lines 114 through 160 highlighted in pink, indicating they were added by the user. The code logic involves checking if a button is hovered over and then triggering a random outcome. It then iterates through options, incrementing 'goodChoices' for good ones and moving to the next question. Finally, it checks if all questions are answered and sets the current screen based on the total number of good choices relative to the total number of choices.

```
// This function is called from main.js
// only when currentScreen === "game"
function gameMousePressed() {
    // Only trigger the outcome if the button is clicked
    if (isHover(gameBtn)) {
        triggerRandomOutcome();
    }
}

const current = decisions[choiceIndex];
current.options.forEach((opt) => {
    if (isHover(opt._btn)) {
        if (opt.good) goodChoices++; // increment for good choices
        choiceIndex++; // move to next question
    }
})

// Check if all questions answered
if (choiceIndex >= totalChoices) {
    // If player got more than half good, good ending
    if (goodChoices >= totalChoices / 2) currentScreen = "win";
    else currentScreen = "lose";
}

// Reset for next playthrough
choiceIndex = 0;
goodChoices = 0;
}
```

Screenshot of GitHub commits adjusting the blob's visual appearance and movement. \

```
JS game.js > drawGame
59 | };
60 |
61 // -----
62 // Main draw function for this screen
63 // -----
64 // drawGame() is called from main.js *only*
65 // when currentScreen === "game"
66 |
67 function drawGame() {
68 |
69     background(240, 230, 140);
70     fill(0);
71 |
72     textAlign(CENTER, CENTER);
73 |
74     textSize(28);
75     text("Lost Puppy Adventure", width / 2, 100);
76 |
77     // Show current question
78     const current = decisions[choiceIndex];
79     textSize(22);
80     text(current.question, width / 2, 200);
81 |
82     // Draw buttons for current options
83     current.options.forEach((opt, i) => {
84         const btn = {
85             x: width / 2,
86             y: 300 + i * 100,
87             w: 400,
88             h: 70,
89             label: opt.text,
90         };
91         drawGameButton(btn);
92         opt._btn = btn; // store the button for input checking
93     });
94 |
95     // Show tracker on top-right
96     textSize(18);
97     textAlign(RIGHT, TOP);
98     text(`Good Choices: ${goodChoices}/${totalChoices}`, width - 20, 20);
99 |
100    // Cursor feedback
101    let hover = current.options.some((opt) => isHover(opt._btn));
102    cursor(hover ? HAND : ARROW);
103 }
104 |
105 // -----
106 // Button drawing helper
107 // -----
108 // This function is responsible *only* for drawing the button.
109 // It does NOT handle clicks or game logic.
110 function drawGameButton({ x, y, w, h, label }) {
111     rectMode(CENTER);
112 |
113     // Check if the mouse is hovering over the button
114     // isHover() is defined in main.js so it can be shared
115     const hover = isHover({ x, y, w, h });
116 |
117     noStroke();
118 |
119     // Change button colour when hovered
120     // ...
121 }
```

Screenshots of editing the Starting Screen text to the new game title “Lost Puppy Adventure.”

GenAI Documentation

Date Used: February 3, 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: Use ideas from ideation and brainstorming for different stories and generate starter code to help me get started and understand the contents of the original code.

Summary of Interaction: I used it to help understand which parts of the example code I could modify to achieve my desired outcome for the game, and it helped me choose ideas that were easy to implement.

Human Decision Point(s): I chose a general prompt provided by GenAI and created a storyline that can result in different endings. I came up with the different choices that players can select and create, and manually modified to the instructions, including changing the title of the game and typing the wording for each decision button.

Integrity & Verification Note: I looked over the generated code and tried to learn and understand which each line of code leads to for each function, and I made follow up prompts. I did multiple play tests on Live Server to check if the code if it created my desired output and matched the buttons with the correct decision options, and if the point system was implemented accurately according to how I wanted it to be calculated.

Scope of GenAI Use: GenAI did not come up with the decision points or replace the creativity I had for this assignment. I used a general starter prompt, but it did not write my entire code. Modifications and guidance through follow up prompts, as well as manual input of code by myself, were still required.

Limitations or Misfires: GenAI suggested general ideas, but it did not create a coherent story and included many repetitive decision options that did not follow my vision. I used it to help generate code, but it required guidance in order to create an outcome that aligned with the story I had in mind and fit the requirements of this side quest.

Summary of Process (Human + Tool)

- Modified Original Week 3 Example code into a branching story
- Added point system to track choices and endings
- Used GenAI for ideas and starter code
- Manually edited story, buttons, and game flow
- Tested on Live Server to check points and choices
- Ignored or revised unclear GenAI suggestions

Decision Points & Trade-offs

- Options considered:
 - Immediate win/lose outcomes vs cumulative point system
 - Single-choice decisions vs multi-stage branching
- Human decision points:
 - Used a cumulative point system to reflect the quality of player decisions
 - Manually created diverse, coherent decision options for each stage
- Changes implemented:
 - Reorganized buttons and conditions so each choice affected points
 - Created story progression to trigger good or bad endings based on accumulated points

Verification & Judgement

- Play-tested on Live Server
- Checked decision buttons
- Verified point system calculations
- Confirmed story flow
- Edited and retested code as needed
- Adjusted issues from testing

Limitations, Dead Ends, or Open Questions

- Some GenAI code required manual modifications
- Some decision paths gave repetitive decision options
- GenAI story ideas were repetitive and too general
- Manual changing the original code with new game titles and instructions

Appendix

Prompt #1:

Task: Build an interactive story that unfolds through multiple game states and files, branching like a small decision tree.

Bonus: Track a player stat (e.g., trust, health, karma) across scenes and unlock endings based on it.

ChatGPT Response #1:

Explained your existing screen-based framework (start, game, win, lose) and gave a step-by-step plan to:

- Add a global player stat (e.g., trust) in main.js.
- Replace random win/lose with story states (e.g., game_intro, scene_help, scene_ignore).
- Modify drawGame() to include choice buttons.
- Handle choices and update stat in gameMousePressed().
- Create follow-up scenes (scene_help.js) and route endings based on the stat.
- Modify drawWin() and drawBadEnd() to reflect stat-based endings.
- Update main.js to route new screens and input.

TL;DR: player choices → stat changes → endings unlock based on stat.

Prompt #2:

Task: Come up with some ideas to create an interactive story.

ChatGPT Response #2:

Gave story concept ideas that are easy to implement with your existing screens:

- **The Last Seat on the Bus** – social decisions, karma stat, high/medium/low endings.
- **Lost Phone** – trust stat, moral choices, multiple endings.
- **Group Project Simulator** – reputation/stress stat, multiple endings.
- **Midnight Text** – connection stat, endings based on relationships.
- **The Museum Guard** – order/freedom stat, symbolic outcomes.

- **Yes/No Machine** – honesty stat, minimal screens.

Recommended top 3 for implementation: Lost Phone, Group Project Simulator, Midnight Text.

Prompt #3:

Task: Already have multiple game states (start, win, lose). What are some branching options to include that are easy to implement?

ChatGPT Response #3:

Suggested branching options with minimal code changes:

1. **Choice → Biased Win/Lose:** 2 buttons affect chance of winning.
2. **Win/Lose with Different Meanings:** Hidden stat changes the text of win/lose screens.
3. **Delayed Consequence:** Player choice stored in a stat alters win/lose message.
4. **Two Mini-Rounds Before Ending:** Single extra variable tracks choices over 2 rounds.
5. **Fake Win vs True Win:** Choice affects narrative of win without new screens.

Recommended: Option 2 (Win/Lose with different meanings) for lowest effort and highest clarity.

Prompt #4:

Task: I want an interactive story game with little modification to the code. Come up with story ideas.

ChatGPT Response #4:

Provided story ideas that fit existing screens and require minimal code changes:

- **Lost Pet Rescue** 🐶 – Help or ignore a puppy, affects hidden kindness stat.
- **Cafeteria Lunch Dilemma** 🍔 – Give snack or walk away, affects karma.
- **Midnight Stranger** 🌙 – Help or ignore stranger, affects trust.
- **Magic Door Adventure** 🔑 – Enter or stay, simple win/lose.

- **Test Cheat Dilemma** 📝 – Report, ignore, or copy, affects honesty.

Key idea: keep existing screens, use buttons for choices, track stat to affect ending text.

Prompt #5:

Task: Show me how I can implement it in code.

ChatGPT Response #5:

Step-by-step example using **Lost Pet Rescue** 🐶 :

Add player stat:

```
let currentScreen = "start";
```

```
let kindness = 0; // new stat
```

Modify drawGame() for decision buttons:

```
// Buttons: HELP or IGNORE
```

```
drawGameButton(helpBtn);
```

```
drawGameButton(ignoreBtn);
```

Handle choices in gameMousePressed():

```
if (isHover(helpBtn)) {
```

```
    kindness += 1;
```

```
    currentScreen = "win";
```

```
}
```

```
if (isHover(ignoreBtn)) {
```

```
    kindness -= 1;
```

```
    currentScreen = "lose";
```

```
}
```

Update win/lose screens to reflect stat:

- drawWin() shows different messages if kindness > 0.
- drawLose() shows different messages if kindness < 0.

- ✓ Minimal code changes, no new screens, stat affects ending.
-

Prompt #6:

Task: Provide examples on how to implement the bonus part of this side quest.

ChatGPT Response #6:

Explained how to implement the bonus: track a player stat and unlock endings:

```
let kindness = 0; // track stat
```

```
// Update based on choices
```

```
if (helpedPuppy) { kindness += 1; }

else { kindness -= 1; }
```

```
// Unlock endings based on stat
```

```
if (kindness >= 1) {

    // Good Ending

} else if (kindness === 0) {

    // Neutral Ending

} else {

    // Bad Ending
}
```

- ✓ Fully compatible with existing start → game → win/lose flow.