

Process & Decision Documentation

Name: Oma Tahir – o3tahir

Role: Story line creator/Coder

Primary responsibility for this work: Making changes to existing example code

Goal of work session: Taking the example code and adding a story line to it, that allows the player to pick an option that has an attached outcome affecting their player health

Project/Assignment Decisions

For side quest 3, I chose to build a short interactive storyline using the existing code that was provided. I wanted a regular start option, from there the user is given a story scenario with 3 options to pick from with different outcomes. The player has a health status at the top left of the screen, which is affected by the option they have picked which affects the ending of the game. I wanted to try my best to follow the requirements for this side quest, by including the story line, different options to branch off into, and an outcome attached to them affecting player health.

GenAI Documentation

GenAi was used

Date Used: Feb 9, 2026

Tool Disclosure: Copilot (built into VS code) ChatGPT 5.2

Purpose of Use: I went back and forth with ChatGPT 5.2 to help me create clear and concise prompts to give to the built in AI copilot in VS code to get it to do what I wanted. I used the copilot Ai to implement the changes to my example code provided. Having to make revisions to my prompts to better align the changes to the end product I wanted from VS code copilot.

Summary of interaction: I used ChatGPT 5.2 to go back and forth making revisions to my prompts, to help me phrase a clear and specific prompt for the VS code copilot. I then put those prompts into Copilot to make the actual changes to the example code. I tested it to it completed the task needed to and then clicked keep for the changes. It took a few iterations to ensure the changes did not mess up other parts of the code, while still including what I needed in it.

Human Decision Points: I decided on the overall concept for the assignment, this included the structure of the interactive story and key decision points made. I came up with the story scenario, along with the different branches of options to choose from and the outcome that followed. I also picked the health status of the player based on what choice they picked and how that would lead to different endings.

Integrity & Verification Note: I reviewed the code changes made by copilot to make sure the output was actually what I wanted and followed the story line I wanted to create. I ran the code after new changes to make sure nothing went wrong, and if it did make revisions to correct the mistake.

Limitations or Misfires: When ChatGPT 5.2 was giving me prompts to put into copilot it was not understanding what I wanted, and what code was already provided. Due to that it made the game crash and the game screen as a whole was just gone. With further iterations, the prompts provided by ChatGPT were getting rid of certain aspects that were needed such as the start button, or leaving unnecessary old code that did not match the new storyline I created.

Summary of Process (human + tool): I reviewed the starter code that was provided from the example to see how it was set up and where what code could be found. I came up with my idea of what I wanted my story to be, the different options to pick from and what the attached outcome would be. I used ChatGPT to help me create some prompts that I would put into copilot. The initial few prompts given, needed me to go back and change what I was asking for which was not specific enough and making the game crash. I had to give ChatGPT the example code that I was given, this provided a better scope of what code existed already, and what needed to be changed/added specifically to implement the storyline. It told copilot the strict guidelines to follow, what parts of the code it should not touch and what it needed to add to help me create what I wanted. Once I got the ideal prompt, I put it into copilot and ran the code to see if the results were what I wanted. It took a few back and forth prompts to get it to match the exact idea I had in my head of what I wanted or help fix minor issues in the code. When I felt confident with the results I clicked keep and saved the changes to my VS code.

Scope of GenAI: I created the idea of the story and the outcomes, as well as how the health status worked depending on the outcome picked. GenAI was only used to help me generate different prompts that changed what copilot added/changed of the code.

Verification & Judgement: I verified the work by running the game after each change to ensure the story flow, choices, and health tracking behaved as intended. I also checked that the final implementation matched my original idea and the side quest requirements before keeping and saving the changes in VS Code.

Role-Based process Evidence

Current Repository
o3tahir_sidequest3_W3

Current Branch
main

Push origin
Last fetched 1 ho...

Changes

History

No Branches to Compare

updated read me file
Oma Tahir · 15 minutes...

AI updates/ changes to code
Oma Tahir · 1 hour ago

Added files
Oma Tahir · 11 days ago

Initial commit
Oma Tahir · 11 days ago

9 changed

ending.js

I also went in and removed some tags from html at the top of the screen

```
1 // HTML: UN PUT HOW BEHOLDY MY MERRY AT LIES TALE
2 // Final ending screen - displays message based on Health threshold
3
4 + const endingBtn = {
5 +   x: 400,
6 +   y: 680,
7 +   w: 260,
8 +   h: 70,
9 +   label: "Restart Story",
10 + };
11
12 + function drawEnding() {
13 +   let bgColor, titleText, messageText, textColor;
14 +
15 +   // Determine ending based on Health
16 +   if (health >= 1) {
17 +     // Good ending
18 +     bgColor = [150, 220, 150];
19 +     textColor = [30, 80, 30];
20 +     titleText = "GOOD ENDING";
21 +     messageText =
22 +       "You took responsibility for yourself this morning.\n" +
23 +       "By making smart choices, you stayed healthy and productive.\n\n" +
24 +       "Your day went great! Keep up this positive energy.";
25 +   } else if (health === 0 || health === -1) {
26 +     // Neutral ending
27 +     bgColor = [200, 200, 150];
28 +     textColor = [80, 80, 40];
29 +     titleText = "NEUTRAL ENDING";
30 +     messageText =
31 +       "Your day was just... average.\n" +
32 +       "You made some decent choices, some not so great.\n\n" +
33 +       "Nothing terrible happened, but you didn't excel either.\n" +
34 +       "Tomorrow is a new opportunity.";
35 +   } else {
36 +     // Bad ending (Health <= -2)
37 +     bgColor = [220, 120, 120];
38 +     textColor = [100, 30, 30];
```

Current Repository
o3tahir_sidequest3_W3

Current Branch
main

Push origin
Last fetched 1 ho...

Changes

History

No Branches to Compare

updated read me file
Oma Tahir · 15 minutes...

AI updates/ changes to code
Oma Tahir · 1 hour ago

Added files
Oma Tahir · 11 days ago

Initial commit
Oma Tahir · 11 days ago

9 changed

ending.js

I also went in and removed some tags from html at the top of the screen

```
55 + // Ending message
56 + fill(...textColor);
57 + textSize(15);
58 + textAlign(CENTER, TOP);
59 + text(messageText, width / 2, 200, 700);
60 +
61 + // Health display
62 + fill(...textColor);
63 + textSize(14);
64 + textAlign(LEFT, TOP);
65 + text("Final Health: " + health, 30, 30);
66 +
67 + // Return button
68 + drawEndingButton(endingBtn, textColor);
69 + cursor(isHover(endingBtn) ? HAND : ARROW);
70 + }
71 +
72 + function drawEndingButton({ x, y, w, h, label }, textColor) {
73 +   rectMode(CENTER);
74 +   const hover = isHover({ x, y, w, h });
75 +
76 +   noStroke();
77 +   // Button color matches the ending theme
78 +   if (health >= 1) {
79 +     fill(hover ? color(120, 200, 120) : color(150, 200, 150));
80 +   } else if (health === 0 || health === -1) {
81 +     fill(hover ? color(180, 160, 80) : color(200, 180, 100));
82 +   } else {
83 +     fill(hover ? color(180, 60, 60) : color(200, 80, 80));
84 +   }
85 +
86 +   rect(x, y, w, h, 10);
87 +
88 +   fill(255);
89 +   textSize(10);
90 +   textAlign(CENTER, CENTER);
91 +   text(label, x, y);
```


Current Repository
o3tahir_sidequest3_W3

Current Branch
main

Push origin
Last fetched 1 ho...

Changes

History

I've No Branches to Compare

updated read me file
Oma Tahir · 16 minutes...

AI updates/ changes to code
Oma Tahir · 1 hour ago

Added files
Oma Tahir · 11 days ago

Initial commit
Oma Tahir · 11 days ago

AI updates/ changes to code
I also went in and removed some tags from html at the top of the screen

Oma Tahir · eb0157b

+561 -61

9 changed main.js

ending.js

ind...html

main.js

menu.js

miss...y.js

onTime.js

pare...ll.js

start.js

story.js

82

80

81

82

83

84

85

83

84

85

87

88

89

90

91

92

93

94

95

93

96

97

98

99

100

101

95

96

97

98

99

100

101

102

103

104

- else if (currentScreen === "lose") loseMousePressed7.();

+ if (currentScreen === "menu") menuMousePressed7.();

+ else if (currentScreen === "story") storyMousePressed7.();

+ else if (currentScreen === "onTime") onTimeMousePressed7.();

+ else if (currentScreen === "missDay") missDayMousePressed7.();

+ else if (currentScreen === "parentsYell") parentsYellMousePressed7.();

+ else if (currentScreen === "ending") endingMousePressed7.();

}

// -----

@@ -87,18 +90,14 @@ function mousePressed() {

// -----

// This routes keyboard input to the correct screen handler.

function keyPressed() {

- // Each screen «may» define a key handler:

- // start.js - startKeyPressed()

- // instructions.js - instrKeyPressed()

- // game.js - gameKeyPressed()

- // win.js - winKeyPressed()

- // lose.js - loseKeyPressed()

+ // Menu + Story flow. Each screen «may» define a key handler.

- if (currentScreen === "start") startKeyPressed();

- else if (currentScreen === "instr") instrKeyPressed();

- else if (currentScreen === "game") gameKeyPressed7.();

- else if (currentScreen === "win") winKeyPressed7.();

- else if (currentScreen === "lose") loseKeyPressed7.();

+ if (currentScreen === "menu") menuKeyPressed7.();

+ else if (currentScreen === "story") storyKeyPressed7.();

+ else if (currentScreen === "onTime") onTimeKeyPressed7.();

+ else if (currentScreen === "missDay") missDayKeyPressed7.();

+ else if (currentScreen === "parentsYell") parentsYellKeyPressed7.();

+ else if (currentScreen === "ending") endingKeyPressed7.();

}

// -----

Current Repository
o3tahir_sidequest3_W3

Current Branch
main

Push origin
Last fetched 1 ho...

Changes

History

I've No Branches to Compare

updated read me file
Oma Tahir · 16 minutes...

AI updates/ changes to code
Oma Tahir · 1 hour ago

Added files
Oma Tahir · 11 days ago

Initial commit
Oma Tahir · 11 days ago

AI updates/ changes to code
I also went in and removed some tags from html at the top of the screen

Oma Tahir · eb0157b

+561 -61

9 changed menu.js

ending.js

ind...html

main.js

menu.js

miss...y.js

onTime.js

pare...ll.js

start.js

story.js

@@ -0,0 +1,68 @@

1 + // NOTE: Do NOT add setup() or draw() in this file

2 + // Menu screen - entry point to the story

3 +

4 + const menuBtn = {

5 + x: 400,

6 + y: 400,

7 + w: 240,

8 + h: 80,

9 + label: "START",

10 + };

11 +

12 + function drawMenu() {

13 + background(100, 150, 180); // Soft teal background

14 +

15 + // Title

16 + fill(255);

17 + textSize(40);

18 + textAlign(CENTER, CENTER);

19 + text("Wake Up Late!", width / 2, 150);

20 +

21 + // Instructions

22 + fill(255);

23 + textSize(16);

24 + textAlign(CENTER, TOP);

25 + text("Press START or ENTER to Begin", width / 2, 280);

26 +

27 + // Start button

28 + drawMenuButton(menuBtn);

29 + cursor(isHover(menuBtn) ? HAND : ARROW);

30 + }

31 +

32 + function drawMenuButton({ x, y, w, h, label }) {

33 + rectMode(CENTER);

34 + const hover = isHover({ x, y, w, h });

35 +

36 + // Button styling

Current Repository
o3tahir_sidequest3_W3

Current Branch
main

Push origin
Last fetched 1 ho... (1.8)

Changes

History

No Branches to Compare

updated read me file
Oma Tahir • 16 minutes...

AI updates/ changes to code
Oma Tahir • 1 hour ago

Added files
Oma Tahir • 11 days ago

Initial commit
Oma Tahir • 11 days ago

AI updates/ changes to code

I also went in and removed some tags from html at the top of the screen

Oma Tahir eb0157b +561 -51

9 changed

ending.js

ind...html

main.js

menu.js

miss...y.js

onTime.js

pare...ll.js

start.js

story.js

@@ -26,28 +26,38 @@ function drawStart() {
// and also reuse the same information for hover checks.
const startBtn = {
x: width / 2,
- y: 320,
+ y: 300,
w: 240,
h: 80,
label: "START",
};

+ // ADDED: Story button
+ const storyBtn = {
+ x: width / 2,
+ y: 410,
+ w: 240,
+ h: 80,
+ label: "STORY",
+ };

const instrBtn = {
x: width / 2,
- y: 830,
+ y: 520,
w: 240,
h: 80,
label: "INSTRUCTIONS",
};

- // Draw both buttons
+ // Draw all buttons
drawButton(startBtn);
+ drawButton(storyBtn);
drawButton(instrBtn);

// ---- Cursor feedback ----
- // If the mouse is over either button, show a hand cursor

Appendix:

Copilot transcript

My prompt:

You are my coding assistant. Add an interactive branching story game to my existing project.

Goal

- Build a small decision-tree story that plays across multiple game states and multiple files (modular, not all in one file).
- The player starts in a “WakeUpLate” scene and must choose 1 of 3 options.
- Each choice changes a persistent player stat called Health, and leads to a different scene/outcome.
- Use the Health value to unlock different endings (at least 2 endings), based on thresholds.

Story + Choice Rules (must match exactly)

Scene 1: WakeUpLate (intro text: player wakes up late already)

Present 3 choices:

- 1) “Get up quickly and get ready” -> Health +1 -> Outcome: get to school on time
- 2) “Go back to sleep” -> Health -2 -> Outcome: miss the day
- 3) “Rot in bed and check phone” -> Health -1 -> Outcome: parents yell at you

Endings (Health-based)

- After the immediate outcome scene, show an ending screen that depends on current Health.
 - Define clear thresholds, e.g.:
 - Health $\geq 1 \Rightarrow$ “Good Ending”
 - Health $= 0$ or Health $= -1 \Rightarrow$ “Neutral Ending”
 - Health $\leq -2 \Rightarrow$ “Bad Ending”
- (You can adjust exact words)

Prompt Outcome:

I'll help you add an interactive branching story game to your project. Let me start by examining the existing game structure to understand how it's organized.

Now I understand the structure. I'll create a modular branching story system that fits into your existing game framework. Let me build this step by step.

I'll create:

1. **story.js** — manages story state (current scene, Health stat)
2. **wakeupLate.js** — opening scene with 3 choices

3. **storyOutcome1/2/3.js** — immediate outcomes for each choice
4. **goodEnding/neutralEnding/badEnding.js** — final endings based on Health
5. Update `main.js`, `start.js`, and `index.html` to integrate

Perfect! I've successfully added a complete interactive branching story game to your project. Here's what I've created:

Files Created

Story System:

- `story.js` — Manages game state (Health stat, scene transitions, story logic)

Story Scenes:

- `wakeupLate.js` — Opening scene with 3 choices
- `storyOutcome1.js` — "Got to school on time" (Health +1)
- `storyOutcome2.js` — "Missed the whole day" (Health -2)
- `storyOutcome3.js` — "Parents yell at you" (Health -1)

Ending Screens (Health-based):


- `goodEnding.js` — Health ≥ 1 (Success)
- `neutralEnding.js` — Health = 0 or -1 (Average)
- `badEnding.js` — Health ≤ -2 (Poor choices)

Files Updated

- `index.html` — Added script tags for all new story files
- `main.js` — Added routing for all story screens
- `start.js` — Added "STORY" button (press S or click button to start)

How It Works

1. **Click "STORY"** on the start screen (or press **S**)
2. **Choose one of 3 options** in the WakeUpLate scene:
 - Option 1: Get up quickly → Health +1 → Good outcome
 - Option 2: Go back to sleep → Health -2 → Bad outcome
 - Option 3: Check phone → Health -1 → Neutral outcome

3. **View your immediate outcome** with Health displayed
4. **See your ending** (Good/Neutral/Bad) based on final Health value
5. **Play again** by pressing  or clicking "Play Again"

The story system is fully modular—each screen is independent, and the Health stat persists across scenes. Players can replay and try different paths to unlock all three endings.



My prompt #2:

You are my VS Code Copilot. My last change broke the game (blank screen). Do NOT rewrite the whole project or change the engine/framework. Make the smallest safe changes that match the patterns already in my code.

First: Stabilize (no blank screen)

- 1) Read the existing code structure and identify how screens/states are currently implemented (e.g., a state variable, a switch-case, a scenes array, a router, etc.).
- 2) Fix the crash/blank screen by aligning with the current render/update loop and existing event handlers.
- 3) Add temporary safe debugging ONLY if needed (console.log for currentState and errors), then keep it minimal.

Then: Add the interactive branching story USING EXISTING PATTERNS (no new architecture)

- Implement ONE new story state inside the existing state system (don't create new folders/modules unless the project already does that).
- Add a "WakeUpLate" scene with 3 choices (buttons/keys) using the same UI method the project already uses.

Choices and stat changes (must match):

- A) Get up quickly and get ready -> Health +1 -> go to "OnTime" scene
- B) Go back to sleep -> Health -2 -> go to "MissDay" scene
- C) Rot in bed and check phone -> Health -1 -> go to "ParentsYell" scene
- Health must persist across these scenes (store it wherever the project already stores player data; if none exists, add ONE global/object in the same file where other globals are).

Ending logic (simple, no extra complexity)

- After each outcome scene, show an ending message based on Health:
 - Health \geq 1: Good ending
 - Health 0 to -1: Neutral ending
 - Health \leq -2: Bad ending
- Display Health on screen in each scene.

Constraints

- Keep changes minimal: prefer editing existing files over creating new ones.
- Do not rename existing functions or restructure the loop.
- Use the project's existing input approach (mouse clicks / key presses) and existing drawing/text style.
- Make sure every new state is reachable and every transition sets a valid state so the screen never goes blank.

Output

- Show a brief list of what you changed and why (1–2 sentences).
- Provide the exact code edits with file names and clearly marked sections (e.g., “// ADDED”).

Prompt Outcome#2:

I'll diagnose the blank screen issue and stabilize the game using the existing patterns. Let me first check the current state of the files.

I see the issue—the code is back to vanilla state, but the story files I created were deleted, leaving broken references (if any remained). Let me stabilize by adding the story system **directly into the existing file structure** using the same modular pattern.

Plan:

1. Add player state object to `main.js` (where globals live)
2. Create 5 minimal story scene files following the existing pattern
3. Update `main.js` routing to handle story screens
4. Add "STORY" button to `start.js`
5. Update `index.html` to load the new files

Let me start:

Now I'll add the "STORY" button to the start screen:

Now I'll create the 5 story scene files. First, the opening scene:

Now I'll update the `index.html` to load the new story files:

Perfect! Let me verify the files are all created:

Let me verify the main routing is correct:

I see the file got corrupted with my edits. Let me fix `main.js` completely:

My prompt#3:

You are editing an existing p5.js multi-screen project. DO NOT add setup() or draw() anywhere except main.js (they already exist). Follow the existing architecture:

- main.js routes screens using currentScreen string:
"start" | "instr" | "game" | "win" | "lose"
- Each screen file defines:
drawX() + optional XMousePressed() + optional XKeyPressed()
- Buttons use rectMode(CENTER) and hover uses isHover({x,y,w,h}) from main.js

My issue: previous changes caused a BLACK SCREEN. That happens when currentScreen is set to a value that main.js does not route to a draw function, or a handler references a missing function. Fix this by ONLY adding new screen names to main.js routing and ensuring every referenced draw/handler exists.

TASK: Add an interactive branching story with a persistent Health stat (global), using the SAME patterns as game.js/instructions.js. Keep changes minimal and safe.

STORY REQUIREMENTS (must match):

New first scene: "story" (WakeUpLate)

- Display title "Wake Up Late" + short story text
- Display Health on screen at all times
- Show 3 clickable buttons (same button style as game.js):
 - 1) "Get up quickly" -> Health +1 -> currentScreen = "onTime"
 - 2) "Go back to sleep" -> Health -2 -> currentScreen = "missDay"
 - 3) "Check phone in bed" -> Health -1 -> currentScreen = "parentsYell"

Outcome scenes:

- "onTime": text says you got to school on time. Show Health. Show ONE "Continue" button -> goes to "ending"
- "missDay": text says you missed the day. Show Health. "Continue" -> "ending"
- "parentsYell": text says parents yelled at you. Show Health. "Continue" -> "ending"

Ending scene:

- "ending": choose message based on Health thresholds:
 - Health >= 1 => Good ending
 - Health is 0 or -1 => Neutral ending
 - Health <= -2 => Bad ending
- Show Health and a "Return to Start" button that sets currentScreen="start" and resets Health to 0.

INPUT:

- Mouse: clicking buttons should work (use isHover just like other files)

- Keyboard: add simple shortcuts:
On "story": keys 1/2/3 choose options
On outcome/ending screens: ENTER continues/returns

FILES / CHANGES YOU MUST MAKE (minimal):

1) index.html: add script tags for new files in the correct order BEFORE main.js

Add:

```
<script src="story.js"></script>
<script src="onTime.js"></script>
<script src="missDay.js"></script>
<script src="parentsYell.js"></script>
<script src="ending.js"></script>
```

2) main.js:

- Add new screen names to the draw router:
else if (currentScreen === "story") drawStory();
else if (currentScreen === "onTime") drawOnTime();
else if (currentScreen === "missDay") drawMissDay();
else if (currentScreen === "parentsYell") drawParentsYell();
else if (currentScreen === "ending") drawEnding();
- Add routing for mousePressed() and keyPressed() for these screens:
storyMousePressed(), storyKeyPressed(), etc. Use ?.() only if you purposely omit a handler.
- Add ONE new global at top-level:
let health = 0;

3) Create new files (match style of other files, NO setup/draw):

- story.js defines drawStory(), storyMousePressed(), storyKeyPressed(), and a small helper to draw buttons if needed.
- onTime.js defines drawOnTime(), onTimeMousePressed(), onTimeKeyPressed()
- missDay.js defines drawMissDay(), missDayMousePressed(), missDayKeyPressed()
- parentsYell.js defines drawParentsYell(), parentsYellMousePressed(), parentsYellKeyPressed()
- ending.js defines drawEnding(), endingMousePressed(), endingKeyPressed()

SAFETY CHECKS TO PREVENT BLACK SCREEN:

- Never set currentScreen to a string unless main.js routes it.
- Do not reference functions that don't exist.
- Keep button objects local in each file (like instructions.js) OR constants (like gameBtn), but ensure names don't conflict.

OUTPUT FORMAT:

- Provide full code for each new file
- Show exact code edits for index.html and main.js (copy/paste ready)

Prompt Outcome#3:

I'll fix this carefully by adding only what's necessary and following the exact existing patterns. Let me start with the minimal changes to `main.js` and `index.html`, then create the 5 new story files.

Let me do these edits sequentially to ensure accuracy:

Now update `index.html` to load the story files:

Now I need to add a "STORY" button to the start screen:

Now I'll create the 5 story scene files. Starting with `story.js` (the opening scene):

Perfect! Let me verify all files are created:

I need to fix the `mousePressed` function - it got mangled during edits:

My prompt#4:

You are editing an existing p5.js multi-screen project. DO NOT add `setup()` or `draw()` anywhere except `main.js`. Use the existing router pattern (`currentScreen` string + `drawX/handlers` per file).

Goal: REMOVE the old "Win or Lose" flow from the user experience.

The game should start directly in the interactive story (wake up late), and once the story completes it should go to an ending screen, with an option to restart the story.

IMPORTANT: Do NOT show or use the old Start screen, Instructions screen, or the old "game/win/lose" random outcome flow. We can leave old files in the project, but they must not be reachable and must not be used by routing.

What to change (minimal + safe, no black screen):

1) `main.js`

- Set the initial screen to the story:

```
let currentScreen = "story";
```

- Remove/disable routing to old screens so they can't be reached:

Do NOT call `drawStart()`, `drawInstr()`, `drawGame()`, `drawWin()`, `drawLose()` anywhere.

The `draw()` router should ONLY handle:

```
"story" | "onTime" | "missDay" | "parentsYell" | "ending"
```

Same for `mousePressed()` and `keyPressed()`: only route to story/outcome/ending handlers.

- Add ONE global stat:

```
let health = 0;
```

2) `index.html`

- Keep script order correct so new screens load before `main.js`.

- Add:

```
<script src="story.js"></script>
```

```
<script src="onTime.js"></script>
<script src="missDay.js"></script>
<script src="parentsYell.js"></script>
<script src="ending.js"></script>
```

- You may leave the old script tags in place, BUT the app must not use them anymore because main.js won't route to those screens.

(Optional cleanup: you can remove start.js, instructions.js, game.js, win.js, lose.js script tags, but only if that won't break anything. If unsure, leave them.)

3) Implement story screens using existing patterns (NO setup/draw in these files)

Scene "story" (WakeUpLate):

- Show title "Wake Up Late" + short text: player wakes up late already
- Show Health on screen always
- 3 buttons (rectMode(CENTER) + isHover from main.js) and keyboard shortcuts 1/2/3:
 - 1) "Get up quickly" -> health += 1 -> currentScreen="onTime"
 - 2) "Go back to sleep" -> health -= 2 -> currentScreen="missDay"
 - 3) "Check phone in bed" -> health -= 1 -> currentScreen="parentsYell"

Outcome scenes:

- "onTime": text says got to school on time; show Health; one "Continue" button + ENTER -> currentScreen="ending"
- "missDay": text says missed the day; show Health; Continue/ENTER -> "ending"
- "parentsYell": text says parents yelled; show Health; Continue/ENTER -> "ending"

Ending scene:

- Use thresholds:
 - health >= 1 => Good ending
 - health 0 or -1 => Neutral ending
 - health <= -2 => Bad ending
- Show Health
- Button: "Restart Story" (and key R) resets health=0 and sets currentScreen="story"

Safety rules to prevent black screen:

- Never set currentScreen to a value not handled in main.js.
- Ensure every draw/handler referenced in main.js exists in the correct file.
- Use the same font/text styling approach as existing screens.

Output:

- Provide exact copy/paste edits for main.js and index.html
- Provide full code for: story.js, onTime.js, missDay.js, parentsYell.js, ending.js

Prompt Outcome#4:

I'll fix this carefully. The key is to:

1. Set initial currentScreen to "story"
2. Only route story screens in the draw/input handlers
3. Create the 5 story files following existing patterns

Let me start with main.js:

Now I'll create the 5 story scene files. Starting with story.js:

Good! The story files already exist from my previous creation. Let me quickly verify they're all present and have the correct content:

Perfect! All files exist. Let me verify the `ending.js` has the correct threshold logic:

I need to fix the `ending.js` file to return to "story" instead of "start" (since we're no longer using the start screen):

Perfect! Let me verify the complete `main.js` routing is correct:

Let me verify the outcome files are correct. Checking onTime.js:

Perfect! Let me verify the `index.html` has the correct script order:

Perfect! Everything is set up correctly. Here's a summary of what I've done:

Summary of Changes

Goal Achieved: The game now starts directly in the interactive story (wake up late) and removes the old Win/Lose flow from the user experience.

1. `main.js` — Streamlined to story-only flow

- ☒ Initial screen set to "story" instead of "start"
- ☒ Only routes story screens: `story` | `onTime` | `missDay` | `parentsYell` | `ending`
- ☒ Global `health = 0` persists across scenes
- ☒ No references to old screens (start, instr, game, win, lose)

2. `index.html` — Correct script load order

- ☒ Story files loaded before `main.js`
- ☒ Old files left in place but unreachable (won't cause errors)
- ☒ p5.js loaded first

3. Story Flow Files (already created)

- `story.js` → Wake Up Late scene with 3 choices (1/2/3 keys or buttons)

- Choice 1: Health +1 → `onTime`
- Choice 2: Health -2 → `missDay`
- Choice 3: Health -1 → `parentsYell`

- `onTime.js` → "Got to school on time" outcome + Continue button
- `missDay.js` → "Missed the whole day" outcome + Continue button
- `parentsYell.js` → "Parents yelled" outcome + Continue button
- `ending.js` → Final screen with Health-based endings

- Health ≥ 1 : Good Ending
- Health = 0 or -1: Neutral Ending
- Health ≤ -2 : Bad Ending
- Restart Story button (press R) resets and returns to story

4. Safety Verification

- ✓ No black screen risk: `currentScreen = "story"` is routed in `draw()`
- ✓ All transitions set valid screen names
- ✓ All referenced draw/handler functions exist
- ✓ Health displays in every story scene

The game is now ready to play! Players will immediately see the "Wake Up Late" scene.

My prompt#5:

You are editing an existing p5.js multi-screen project. DO NOT add `setup()` or `draw()` anywhere except `main.js`. Follow the existing router pattern: `currentScreen` string + `drawX()` + `XMousePressed()/XKeyPressed()` per screen file. Buttons use `rectMode(CENTER)` and `isHover()` from `main.js`.

Goal:

- The game should open on a simple Start Menu screen with ONE "START" button.
- After clicking START (or pressing ENTER), the branching story begins.
- Do NOT use the old original "start/instr/game/win/lose" screens. They can remain in files, but must not be reachable.

SCREENS TO USE (ONLY these):

"menu" -> "story" -> ("onTime" | "missDay" | "parentsYell") -> "ending"

1) `main.js` changes (minimal and safe):

- Set initial screen:

- ```
let currentScreen = "menu";
```
- Add one global stat:  
let health = 0;
  - Update draw() router to ONLY call:  
drawMenu(), drawStory(), drawOnTime(), drawMissDay(), drawParentsYell(), drawEnding()
  - Update mousePressed() router to ONLY call:  
menuMousePressed(), storyMousePressed(), onTimeMousePressed(), missDayMousePressed(), parentsYellMousePressed(), endingMousePressed()
  - Update keyPressed() router to ONLY call:  
menuKeyPressed(), storyKeyPressed(), onTimeKeyPressed(), missDayKeyPressed(), parentsYellKeyPressed(), endingKeyPressed()
  - Remove/disable any routing to drawStart/drawInstr/drawGame/drawWin/drawLose so there is no path to old screens and no black screen.

## 2) index.html changes:

- Add new script tags BEFORE main.js (order matters):  

```
<script src="menu.js"></script>
<script src="story.js"></script>
<script src="onTime.js"></script>
<script src="missDay.js"></script>
<script src="parentsYell.js"></script>
<script src="ending.js"></script>
```
- You may leave old script tags in place, but the app must not use them anymore because main.js won't route to them.

## 3) New files (NO setup() / draw() in these files):

### A) menu.js

- drawMenu(): simple background, title, short instruction, one START button
- menuMousePressed(): clicking START sets health=0 and currentScreen="story"
- menuKeyPressed(): ENTER also starts (same behavior)
- Use isHover() and rectMode(CENTER), same button style as game.js/instructions.js

### B) story.js (WakeUpLate)

- drawStory(): title "Wake Up Late", short story text, show Health on screen always
- Show 3 choice buttons + keyboard shortcuts 1/2/3:
  - 1) "Get up quickly" -> health += 1 -> currentScreen="onTime"
  - 2) "Go back to sleep" -> health -= 2 -> currentScreen="missDay"
  - 3) "Check phone in bed" -> health -= 1 -> currentScreen="parentsYell"

### C) Outcome screens:

- onTime.js: drawOnTime() shows "You got to school on time." + Health + Continue button
- missDay.js: drawMissDay() shows "You missed the day." + Health + Continue button
- parentsYell.js: drawParentsYell() shows "Your parents yelled at you." + Health + Continue button

Each outcome file must include MousePressed + KeyPressed:

- Clicking Continue OR pressing ENTER -> currentScreen="ending"

D) ending.js

- drawEnding(): show ending based on thresholds:
  - health  $\geq 1 \Rightarrow$  Good ending
  - health 0 or -1  $\Rightarrow$  Neutral ending
  - health  $\leq -2 \Rightarrow$  Bad ending
- Show Health always
- Button "Restart" + key R:
  - sets health=0 and currentScreen="menu"

SAFETY RULES (prevent black screen):

- Never set currentScreen to a value not handled in main.js.
- Ensure every draw/handler referenced in main.js exists.
- Keep changes minimal and match existing style.

OUTPUT:

- Provide copy/paste-ready code edits for main.js and index.html
- Provide full code for: menu.js, story.js, onTime.js, missDay.js, parentsYell.js, ending.js

#### Prompt Outcome#5:

I'll update `main.js` to start on "menu", create the menu.js file, and update index.html. Let me do this carefully to avoid any routing issues.

Summarized conversation history

All Phase 4 updates complete! Here's what I've implemented:

1. **Created** `menu.js` — New entry screen with:
  - o Teal background, "Wake Up Late?" title
  - o One START button (x: 400, y: 400)
  - o Keyboard support (ENTER to start)
  - o Resets health to 0, transitions to story
2. **Updated** `main.js` **routing** — Added menu support:
  - o draw() includes `if (currentScreen === "menu") drawMenu();` first
  - o mousePressed() routes to `menuMousePressed?().()`
  - o keyPressed() routes to `menuKeyPressed?().()`
3. **Updated** `index.html` — Added script loading:
  - o `menu.js` loads before `story.js` and all other scene files
  - o All scripts now load before `main.js`
4. **Updated** `ending.js` — Fixed restart logic:

- Changed both `endingMousePressed()` and `endingKeyPressed()` to transition to "menu" instead of "story"
- health resets to 0 on restart

The flow is now: **menu** → **story** → **outcomes** → **ending** → **menu** ✓



#### My prompt#6:

My p5 game still runs, but after I choose an option in the story, I can't restart or continue properly. I removed some HTML header/hint text in index.html and now I can't "start again" after an ending/outcome.

Do NOT change the overall architecture. Keep `setup()/draw()` only in `main.js` and keep the `currentScreen` router. Make the smallest fixes needed so the flow always works:

Required flow:

menu -> story -> (onTime | missDay | parentsYell) -> ending -> menu

Fix goals:

- 1) Ensure each outcome screen has a visible "Continue" button AND ENTER key works to go to "ending".
- 2) Ensure ending screen has a visible "Restart" (or "Play Again") button AND R key works to return to "menu" and reset health=0.
- 3) Ensure menu screen has a visible "START" button AND ENTER key starts story (health=0, currentScreen="story").
- 4) Ensure `mousePressed()` routing in `main.js` calls the correct handler for the currentScreen for ALL these screens, and `keyPressed()` routing does too. No missing handlers.
- 5) Add a safety fallback in `draw()`: if currentScreen is not one of the valid screens, set currentScreen="menu" so we never get a blank/soft-lock.

Important:

- The HTML header/hint removal should not affect canvas buttons; buttons are drawn in p5. So the issue is likely missing routing, missing handlers, or button coordinates not matching `isHover` bounds.
- Check that each screen's button rectangle (x,y,w,h) in `MousePressed` matches exactly where it is drawn in `draw()`.
- Make sure `rectMode(CENTER)` is used consistently when drawing AND when checking `isHover`.

Output:

- Show the exact edits needed in `main.js` (`draw/mousePressed/keyPressed` routing + fallback).

- Show the exact fixes in any screen files where Continue/Restart/Start is missing or not clickable.

#### Prompt Outcome #6:

I'll diagnose the issue by checking the current state of all files. Let me read them first:

I found the issues. Let me show you the exact fixes needed:

**Issue 1: Duplicate `main.js` in HTML** — `main.js` appears twice at the bottom (lines 47-50), which can cause routing conflicts.

**Issue 2: Missing safety fallback in `draw()`** — If `currentScreen` becomes invalid (typo/corruption), the game goes blank.

Here are the fixes:

#### Summary of fixes made:

1. **Removed duplicate** `<script src="main.js"></script>` from `index.html` (was loading twice, which could cause routing conflicts)
2. **Added safety fallback in `draw()`** — Now if `currentScreen` is ever invalid (by accident or typo), it automatically resets to "menu" and displays it instead of a blank screen

#### Verification of all screen buttons and handlers:

| Screen | Button Coords                   | Draw Match            | MousePressed                 | KeyPressed           |
|--------|---------------------------------|-----------------------|------------------------------|----------------------|
| menu   | x:400<br>y:400<br>w:240<br>h:80 | ✓<br>rectMode(CENTER) | ✓ START →<br>story           | ✓ ENTER<br>→ story   |
| story  | 3<br>buttons,<br>w:380<br>h:60  | ✓<br>rectMode(CENTER) | ✓ Each choice<br>transitions | ✓ Keys<br>1/2/3 work |

| Screen             | Button Coords                   | Draw Match            | MousePressed                      | KeyPressed          |
|--------------------|---------------------------------|-----------------------|-----------------------------------|---------------------|
| <b>onTime</b>      | x:400<br>y:680<br>w:250<br>h:70 | ✓<br>rectMode(CENTER) | ✓ Continue →<br>ending            | ✓ ENTER<br>or SPACE |
| <b>missDay</b>     | x:400<br>y:680<br>w:250<br>h:70 | ✓<br>rectMode(CENTER) | ✓ Continue →<br>ending            | ✓ ENTER<br>or SPACE |
| <b>parentsYell</b> | x:400<br>y:680<br>w:250<br>h:70 | ✓<br>rectMode(CENTER) | ✓ Continue →<br>ending            | ✓ ENTER<br>or SPACE |
| <b>ending</b>      | x:400<br>y:680<br>w:260<br>h:70 | ✓<br>rectMode(CENTER) | ✓ Restart →<br>menu<br>(health=0) | ✓ R or<br>ENTER     |

All buttons use `rectMode(CENTER)` consistently, and all handlers are properly routed in main.js. The flow `menu → story → outcomes → ending → menu` should now work correctly.

## ChatGPT 5.2 transcript

### My prompt:

i need help creating a clean and consise prompot to put in copiolit built into vs code, i want it to I want it to build an interactive story that plays out over multiple game states and files. The story should branch based on player choices, kind of like a small decision tree, where each choice leads to a different scene or outcome. I want the story to include someone waking up late alre4ady, they then are faces with three options, to either get up quickly and get ready, stay in bed and go back to sleep, or just rot in bed to check their phone. each choice leads to a different outcome. Getting up and getting ready means they get to school on time, fgoing to sleep means missing the day, and staying in bed gets you yelled at by your parents.Bonus: I also want to track a player stat (like trust, health, or karma) across scenes, and use that stat to unlock different endings depending on the choices the player makes. based on the differnt outcomes i gave you for each choice it tracks the health accordingly, gaining a point for getting up and ready on time, lossing 2 points for going to sleep, and 1 point lost for staying in bed on yopur phone

## Prompt Outcome:

Here's a clean Copilot prompt you can paste into VS Code (it tells it exactly what to build, how to structure files, and what the branching + stat rules are):

You are my coding assistant. Add an interactive branching story game to my existing project.

### Goal

- Build a small decision-tree story that plays across multiple game states and multiple files (modular, not all in one file).
- The player starts in a "WakeUpLate" scene and must choose 1 of 3 options.
- Each choice changes a persistent player stat called Health, and leads to a different scene/outcome.
- Use the Health value to unlock different endings (at least 2 endings), based on thresholds.

### Story + Choice Rules (must match exactly)

Scene 1: WakeUpLate (intro text: player wakes up late already)

Present 3 choices:

- 1) "Get up quickly and get ready" -> Health +1 -> Outcome: get to school on time
- 2) "Go back to sleep" -> Health -2 -> Outcome: miss the day
- 3) "Rot in bed and check phone" -> Health -1 -> Outcome: parents yell at you

### Endings (Health-based)

- After the immediate outcome scene, show an ending screen that depends on current Health.

- Define clear thresholds, e.g.:

- Health  $\geq 1$  => "Good Ending"
- Health == 0 or Health == -1 => "Neutral Ending"
- Health  $\leq -2$  => "Bad Ending"

(You can adjust exact wording, but keep thresholds consistent and include at least 2 endings.)

### Implementation Requirements

- Create these modules/files (or equivalent):
  - gameState or state manager: handles current scene + transitions
  - playerData: stores Health and persists it across scenes
  - scenes folder with separate files per scene (WakeUpLate, OnTime, MissDay, ParentsYell, Ending)
- Use clean functions like render(), handleInput(choice), and transitionTo(sceneName).
- UI: Use the same UI style my project already uses (text + buttons/keys). If my project has a rendering loop, integrate with it instead of replacing it.
- Show Health on-screen in every scene.
- Add comments so I can understand what each file does.
- Do not break existing code. If something is missing (like a scene registry), create it.

### Deliverable

- Generate the code changes for each file, and clearly label each file path with its contents.

If you want a *shorter* version (still works, just less strict), tell me what your game is built with (p5.js? Phaser? plain HTML canvas? terminal?) and I'll shrink it to the smallest possible prompt that fits your setup.

### My prompt#2:

that did not work, it made the game crash and now a screen does not show up at all, i want to ensure it fits within the range of things it can do and the code already has built in, revise the prompt

### Prompt Outcome#2:

You are my VS Code Copilot. My last change broke the game (blank screen). Do NOT rewrite the whole project or change the engine/framework. Make the smallest safe changes that match the patterns already in my code.

First: Stabilize (no blank screen)

- 1) Read the existing code structure and identify how screens/states are currently implemented (e.g., a state variable, a switch-case, a scenes array, a router, etc.).
- 2) Fix the crash/blank screen by aligning with the current render/update loop and existing event handlers.
- 3) Add temporary safe debugging ONLY if needed (console.log for currentState and errors), then keep it minimal.

Then: Add the interactive branching story USING EXISTING PATTERNS (no new architecture)

- Implement ONE new story state inside the existing state system (don't create new folders/modules unless the project already does that).
- Add a "WakeUpLate" scene with 3 choices (buttons/keys) using the same UI method the project already uses.

Choices and stat changes (must match):

- A) Get up quickly and get ready -> Health +1 -> go to "OnTime" scene
- B) Go back to sleep -> Health -2 -> go to "MissDay" scene
- C) Rot in bed and check phone -> Health -1 -> go to "ParentsYell" scene
- Health must persist across these scenes (store it wherever the project already stores player data; if none exists, add ONE global/object in the same file where other globals are).

Ending logic (simple, no extra complexity)

- After each outcome scene, show an ending message based on Health:
  - Health >= 1: Good ending
  - Health 0 to -1: Neutral ending
  - Health <= -2: Bad ending
- Display Health on screen in each scene.

Constraints

- Keep changes minimal: prefer editing existing files over creating new ones.
- Do not rename existing functions or restructure the loop.

- Use the project's existing input approach (mouse clicks / key presses) and existing drawing/text style.
- Make sure every new state is reachable and every transition sets a valid state so the screen never goes blank.

Output

- Show a brief list of what you changed and why (1-2 sentences).
- Provide the exact code edits with file names and clearly marked sections (e.g., "// ADDED").

**My prompt#3:**

**Prompt Outcome#3:**