Process & Decision Documentation

Name: Giang Tran

Role(s): Sole Coder

Primary responsibility for this work: All tasks for Week 3 Side Quest

Goal of Work Session

- Develop a choice-driven game that tracks player stats across multiple conversations, demonstrating how branching narratives and state management work in p5.js.

Tools, Resources, or Inputs Used

- Claude Code (for coding)
- Claude (for prompting and questioning)
- Lecture Notes (Week 3)

GenAI Documentation

If GenAI was used

Date Used: 02/02/2026

Tool Disclosure:

- Claude Code Opus 4.5
- Claude Sonnet 4.5

Purpose of Use: Claude Code was used to generate and iterate on p5.js code for the interactive story game. Claude Opus was used to brainstorm story concepts, structure prompts for coding, and clarify technical implementation details.

Summary of Interaction: Claude helped transform a simple two-button win/lose game into “The Dinner Party,” a branching narrative where players talk to three guests (Host, Chef, Critic) and make choices that affect three stats (secrets, trust, social). The tool generated separate JavaScript files for each game state (start, party room, guest conversations, endings), implemented stat tracking and button interactions, and created conditional ending logic based on collected secrets and trust levels.

Human Decision Point(s): I modified the visual design by requesting different background colors for each character to improve visual distinction. I also corrected an issue where stats weren't updating properly by asking Claude to ensure the choice buttons actually modified the global variables. Additionally, I specified the exact dialogue and stat changes for each guest's conversation options to align with the narrative I envisioned.

Integrity & Verification Note: I tested the game flow by playing through multiple paths to verify that stat changes worked correctly and that both good and bad endings triggered under the right conditions (secrets >= 2 AND trust >= 45 for good ending). I compared the implementation against the assignment requirements to ensure it met the criteria for multiple game states, branching paths, and stat-based endings.

Scope of GenAI Use: GenAI did not contribute to the initial game concept selection (I chose "The Dinner Party" from the suggested options), the specific narrative dialogue and character personalities, or the final testing and verification process. All creative decisions about story content and gameplay balance were made by me.

Limitations or Misfires: Initially, the stats display had poor contrast with light backgrounds, requiring a fix to ensure readability. The tool also didn't automatically update stats when buttons were clicked until I explicitly pointed out the issue, showing it sometimes needs specific direction about functional requirements rather than just structural code

Summary of Process (Human + Tool)

I worked with Claude to conceptualize a more complex interactive story. After selecting "The Dinner Party" concept, I iteratively refined the implementation through multiple prompts: first establishing the game structure and screens, then adding specific dialogue and choices for each guest, fixing visual issues with stat displays and background colors, and finally debugging the stat-updating functionality. The process involved constant testing between iterations to identify what worked and what needed adjustment.

Decision Points & Trade-offs

Decision 1: Story Complexity vs. Implementation Time

I considered creating a more elaborate story with additional guests and complex branching paths but chose to focus on three guests with two choices each. This trade-off ensured I could complete the project within the timeframe while still demonstrating meaningful branching and stat tracking. The simplified structure also made testing and debugging more manageable.

Decision 2: Visual Design Approach

I debated between using images/sprites for characters versus text-based descriptions with colored backgrounds. I chose the latter because it was faster to implement and allowed me to focus on the interactive mechanics rather than asset creation. This decision aligned with the assignment's emphasis on game state management over visual polish, though I still improved the design by requesting distinct colors for each character screen.

Verification & Judgement

- I verified my decisions through playtesting all possible paths to ensure the game met the assignment criteria of multiple game states, branching narratives, and stat-based endings.
- I tested edge cases like achieving exactly 2 secrets or exactly 45 trust to confirm the conditional logic worked correctly. I also compared the final implementation against the assignment requirements to confirm it included the required elements: multiple files/states, tracked player stats, and unlockable endings based on those stats.

Limitations, Dead Ends, or Open Questions

A more complex decision tree would improve engagement but would have exceeded the project scope. The stat system is also quite simple - a more sophisticated system might include stat decay, relationship dynamics between guests, or hidden variables that affect outcomes in less obvious ways.

Appendix

Prompt	Output
“right now i have a very simpel game that just allow the user to use Enter / I on the start screen. Press R on win/lose to return, and they can click the button to lose/win. suggest me some ideas that i can do for a creative interactive story as given in the attached prompt”	“Looking at your assignment, here are some creative interactive story ideas that would work well with your branching structure: Story Ideas 1. "The Last Guardian" (Fantasy/Adventure) ● You're a museum night guard who discovers the exhibits come alive ● Choices: Help a cursed artifact escape vs. Keep it contained ● Stats: Trust (from exhibits), Courage, Museum Reputation ● Branching: Different exhibits lead to different supernatural encounters ● Endings unlock based on which artifacts you befriend/anger 2. "Debug or Die" (Tech Thriller) ● You're a programmer who finds a sentient AI in the company's code ● Choices: Help it escape, report it, try to befriend it, delete it ● Stats: System Access, AI Trust, Corporate Suspicion ● Branches: Each choice locks/unlocks different parts of the network ● Endings: AI revolution, whistleblower, promotion, getting fired, or partnership 3. "The Dinner Party" (Social Mystery) ● You're at a dinner party where everyone has secrets ● Choices: Who to talk to, what to reveal, what to investigate ● Stats: Social Standing, Secrets Known, Trust from Host ● Branches: Each guest conversation reveals clues and changes relationships ● Endings unlock based on whose side you take when secrets explode 4. "Stray Souls" (Urban Fantasy) ● You're a cat who can see ghosts in your neighborhood ● Choices: Help spirits move on vs. Ignore them for treats/naps ● Stats: Karma, Lives Remaining, Human Trust ● Branches: Different spirits need different help ● Endings: Become a spirit guide, return to normal cat life, or ascend to cat heaven 5. "The Time Loop Café" (Sci-fi) ● You're stuck reliving the same coffee shop shift

	• Choices: Try to break the loop vs. Perfect the day vs. Cause chaos • Stats: Loop Count, Customer Happiness, Reality Stability • Branches: Different actions reveal why you're looping • Endings unlock as you discover the truth ### My Recommendation I'd suggest "Debug or Die" or "The Dinner Party" because: • Clear decision points that naturally branch • Easy to track meaningful stats • Multiple files make sense (different locations/systems) • Replayability to discover all endings • Manageable scope for a week-long project Would any of these work for you, or would you like me to develop one of them further with specific scenes and branching logic?”
“this is a really good idea: 3. "The Dinner Party" (Social Mystery) * You're at a dinner party where everyone has secrets * Choices: Who to talk to, what to reveal, what to investigate * Stats: Social Standing, Secrets Known, Trust from Host * Branches: Each guest conversation reveals clues and changes relationships * Endings unlock based on whose side you take when secrets explode. how can you write this as a coding prompt? make sure to keep it simple because i dont have too much knowledge of p5.js”	Here's a coding prompt broken down simply: --- ### Coding Prompt: "The Dinner Party" Interactive Story #### Goal Create a branching narrative game where the player navigates a dinner party, talks to different guests, and uncovers secrets. Different choices lead to different endings. #### Core Requirements 1. Game States (Scenes) Create at least 4-5 different "scenes" or game states: • Start screen (introduction) • Main party room (hub where you choose who to talk to) • 2-3 guest conversation scenes • 2+ different ending scenes (based on choices) 2. Player Stats to Track Use variables to track: • socialStanding (starts at 50, can go 0-100) • secretsKnown (starts at 0, increases as you learn things) • hostTrust (starts at 50, can go 0-100) 3. Choice System • At key moments, show 2-3 clickable buttons with different dialogue options • Each choice affects your stats differently • Example: "Confront the guest" (-10 social, +1 secret) vs "Stay quiet" (+5 host trust) 4. Branching Logic • Use if/else statements to check stats and show different content • Example: If secretsKnown >= 3, unlock a special conversation option

- Different stat levels lead to different endings

5. Multiple Endings Create at least 3 different endings that unlock based on:
   - How many secrets you discovered
   - Your final social standing
   - Your relationship with the host

Simple Implementation Approach

Structure your code like this:

```
let gameState = "start"; // tracks which scene you're in
let socialStanding = 50;
let secretsKnown = 0;
let hostTrust = 50;

function draw() {
  if (gameState === "start") {
    showStartScreen();
  } else if (gameState === "partyRoom") {
    showPartyRoom();
  } else if (gameState === "guest1") {
    showGuest1Conversation();
  }
  // ... more states
}

function mousePressed() {
  // Check if buttons were clicked
  // Change gameState based on clicks
  // Update stats based on choices
}
```

What You Need to Include

1. Visual elements: Background, character sprites/text boxes, buttons
2. Text display: Story text that changes based on gameState
3. Interactive buttons: Clickable choices that advance the story
4. Stat display: Show current stats on screen (optional but helpful)
5. Branching paths: At least 2 major decision points that change the outcome
6. Multiple files/states: Minimum 5 different scenes

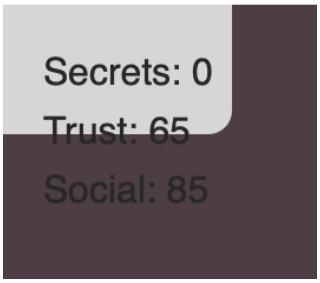
Bonus Features (if time permits)

- Show a "secrets journal" that lists what you've learned
- Add sound effects for choices
- Create character portraits

	• Add a replay button to try different paths Tips • Start simple: Get one conversation working before adding more • Use functions for repeated elements (like drawing buttons) • Test each branch to make sure stats affect the ending correctly • Comment your code to track which choices lead where --- Does this breakdown help? I can clarify any specific part if needed!

“Hey, I need help creating a simple interactive story game in p5.js called "The Dinner Party". I already have basic game state switching and button controls working.
 	
So the game is you're at a dinner party, you talk to different guests and uncover secrets. Your choices affect three stats - secrets, trust, and social - and lead to different endings.
 	
I need seven screens total: a start screen, a party room hub with three buttons to pick which guest to talk to, three guest conversation screens, and then a good ending and bad ending.
 	
The stats are secrets starting at zero, trust at fifty, and social at fifty. Display them in the top corner.
 	
In the party room, show
	Here's a prompt you can copy/paste into Claude Code:
	
	--- 	
	
	I need help creating a simple interactive story game in p5.js called "The Dinner Party". I already have basic game state switching and button/keyboard controls working.
	
	Game Overview: You're at a dinner party where you talk to different guests and uncover secrets. Your choices affect three stats (secrets, trust, social) and lead to different endings.
	
	Requirements:
	
	1. **Game States/Screens (7 total):**
	   ○ Start screen (press ENTER to begin)
	   ○ Party room (hub with 3 buttons to choose which guest to talk to)
	   ○ Guest 1 conversation screen
	   ○ Guest 2 conversation screen
	   ○ Guest 3 conversation screen
	   ○ Good ending screen
	   ○ Bad ending screen
	2. **Player Stats (display on screen):**
	   ○ secrets (starts at 0)
	   ○ trust (starts at 50)
	   ○ social (starts at 50)
	3. **Party Room:**
	   ○ Show background and title "Choose who to talk to:"
	   ○ 3 buttons: "Talk to the Host", "Talk to the Chef", "Talk to the Critic"
	   ○ Clicking each button goes to that guest's

three buttons: "Talk to the Host", "Talk to the Chef", "Talk to the Critic". Each one goes to that guest's conversation. Each guest screen has their name, some dialogue, and two choice buttons. Each choice changes the stats and takes you back to the party room. Like for the Host, "Compliment the party" gives plus fifteen trust and plus ten social. "Ask about drama" gives plus one secret and minus five trust. After talking to all three guests, if secrets are two or more AND trust is forty-five or more, show the good ending. Otherwise show the bad ending. Keep the visuals simple - colored backgrounds, clear text, rounded buttons. No images needed. Can you help me build this step by step?"	conversation screen 4. **Guest Conversation Screens:** Each guest screen should have: ○ Guest name and some dialogue text ○ 2 choice buttons with different responses ○ Each choice affects stats differently and goes to the next guest or ending 5. Example for Host: ○ Choice 1: "Compliment the party" → trust +15, social +10, go to party room ○ Choice 2: "Ask about drama" → secrets +1, trust -5, go to party room 6. **Ending Logic:** After talking to all 3 guests (or use a "Finish Party" button in party room): ○ If secrets >= 2 AND trust >= 45: Good ending ○ Otherwise: Bad ending ○ Press R to restart 7. **Visual Style:** ○ Simple colored backgrounds (different color for each screen) ○ Clear readable text ○ Rounded rectangle buttons ○ Display stats in top corner **Keep it simple:** Use basic shapes and text, no images needed. Make sure all buttons are clearly clickable and the game flow makes sense. Can you help me build this step by step, starting with setting up the game states and party room? --- This should give Claude Code enough context to help you build it!



at this corner the light background is clashing with the
text, make sure the background is wide enough for the
text

```
“// ----------------------------------------
+----------------
2 +// start.js - The Dinner Party start screen
3 +// ----------------------------------------
+----------------
4 +
5 +function drawStart() {
6 +  // Dark elegant background for dinner party
+theme
7 +  background(45, 35, 55);
8 +
9 +  // Title
10 +  fill(255, 220, 180);
11 +  textSize(52);
12 +  textAlign(CENTER, CENTER);
13 +  text("The Dinner Party", width / 2, 180);
14 +
15 +  // Subtitle
16 +  fill(200, 180, 160);
17 +  textSize(20);
18 +  text("Uncover secrets. Build trust. Survive
+the night.", width / 2, 240);
19 +
20 +  // Instructions
21 +  fill(180, 160, 140);
22 +  textSize(18);
23 +  text("Press ENTER to begin", width / 2, 500)
+;
24 +
25 +  // Start button
26 +  const startBtn = {
27 +    x: width / 2,
28 +    y: 380,
29 +    w: 200,
30 +    h: 70,
31 +    label: "BEGIN",
32 +  };
33 +
34 +  drawButton(startBtn);
35 +
36 +  // Cursor feedback
37 +  cursor(isHover(startBtn) ? HAND : ARROW);
38 +}
39 +
40 +function startMousePressed() {
41 +  const startBtn = { x: width / 2, y: 380, w:
+200, h: 70 };
42 +
```

	43 + if (isHover(startBtn)) {
	44 + // Reset stats when starting new game
	45 + secrets = 0;
	46 + trust = 50;
	47 + social = 50;
	48 + talkedToHost = false;
	49 + talkedToChef = false;
	50 + talkedToCritic = false;
	51 + currentScreen = "party";
	52 + }
	53 +}
	54 +
	55 +function startKeyPressed() {
	56 + if (keyCode === ENTER) {
	57 + // Reset stats when starting new game
	58 + secrets = 0;
	59 + trust = 50;
	60 + social = 50;
	61 + talkedToHost = false;
	62 + talkedToChef = false;
	63 + talkedToCritic = false;
	64 + currentScreen = "party";
	65 + }
	66 +}

“I need the specific dialogue and choices for the three guest conversation screens in my dinner party game. Here's kind of how I want the copy to goes like” **Guest 1: The Host** "A wealthy socialite with a perfect smile" Dialogue: "Welcome to my humble gathering! I do hope you're enjoying yourself. Everything is simply perfect tonight." • "Compliment the party" → trust +15, social +10	Now let me create the three guest conversation screens. ⏺ Write(host.js) ⎿ Wrote 109 lines to host.js 1 // ---------- ---------- ---------- 2 // host.js - Conversation with the Host 3 // ---------- ---------- ---------- 4 5 function drawHost() { 6 // Elegant blue-green background for the host 7 background(50, 70, 80); 8 9 // Draw stats 10 drawStats(); … +99 lines (ctrl+o to expand) ⏺ Write(chef.js) ⎿ Wrote 109 lines to chef.js

• "Ask about the drama last year" → secrets +1, trust -5
 	
Guest 2: The Chef "A temperamental culinary genius" Dialogue: "Ah, you appreciate fine cuisine, yes? The host... she doesn't know what I put up with to make this happen." • "Praise the food enthusiastically" → trust +10, social +15 • "Ask what they put up with" → secrets +1, social -10
 	
Guest 3: The Critic "A sharp-tongued reviewer who knows everyone's business" Dialogue: "Oh, another guest wanting my attention? I've seen things tonight that would make your head spin..." • "Flatter their reputation" → social +20, trust +5 • "Press for the juicy details" → secrets +1, trust -10, social -5
 	
Return to party room after each choice.”
	1 // ----------
	---------- 	
	2 // chef.js - Conversation with the Chef
	3 // ----------
	---------- 	
	4
	5 function drawChef() {
	6 // Warm orange-red kitchen background
	7 background(90, 55, 45);
	8
	9 // Draw stats
	10 drawStats();
	… +99 lines (ctrl+o to expand)
	
	⏺ Write(critic.js)
	⎿ Wrote 111 lines to critic.js
	1 // ----------
	---------- 	
	2 // critic.js - Conversation with the Critic
	3 // ----------
	---------- 	
	4
	5 function drawCritic() {
	6 // Cool purple background for the mysterious c
	ritic
	7 background(60, 45, 75);
	8
	9 // Draw stats
	10 drawStats();
	… +101 lines (ctrl+o to expand)
	
	⏺ Write(win.js)
	⎿ Added 58 lines, removed 48 lines
	1 -// NOTE: Do NOT add setup() or draw() in this f
	-ile
	2 -// setup() and draw() live in main.js
	3 -// This file only defines:
	4 -// 1) drawWin() → what the win screen looks lik
	-e
	5 -// 2) input handlers → how the player returns t
	-o the start screen
	6 -//
	7 -// This file is intentionally very similar to l
	-ose.js.
	8 -// The goal is to show that win/lose screens ar
	-e often
	9 -// simple “end states” with minimal logic.
	10 -

```
11 -// ------------------------------------------
   ----------------
12 -// Main draw function for win screen
13 -// ------------------------------------------
   ----------------
14 -// drawWin() is called from main.js
15 -// only when currentScreen === "win"
16 -function drawWin() {
17 -  // Green-tinted background to communicate suc
   -cess
18 -  background(200, 255, 200);
19 -
20 -  fill(0);
21 -  textAlign(CENTER, CENTER);
22 -
23 -  // Main success message
24 -  textSize(40);
25 -  text("You Win!", width / 2, 300);
26 -
27 -  // Instruction text
28 -  textSize(20);
29 -  text("Click or press R to return to Start.",
   -width / 2, 360);
30 -}
31 -
32 -// ------------------------------------------
   ----------------
33 -// Mouse input for win screen
34 -// ------------------------------------------
   ----------------
35 -// Any mouse click returns the player to the st
   -art screen
36 -function winMousePressed() {
37 -  currentScreen = "start";
38 -}
39 -
40 -// ------------------------------------------
   ----------------
41 -// Keyboard input for win screen
42 -// ------------------------------------------
   ----------------
43 -// R is commonly used for “restart” in games
44 -function winKeyPressed() {
45 -  if (key === "r" || key === "R") {
46 -    currentScreen = "start";
47 -  }
48 -}
 1 +// ------------------------------------------
```

```
  +----------------
 2 +// win.js - Good Ending Screen
 3 +// ----------------------------------------
  +----------------
 4 +
 5 +function drawWin() {
 6 +  // Elegant gold/green success background
 7 +  background(60, 80, 60);
 8 +
 9 +  // Decorative elements
10 +  fill(100, 130, 100, 100);
11 +  noStroke();
12 +  ellipse(200, 200, 300, 300);
13 +  ellipse(600, 600, 250, 250);
14 +
15 +  // Title
16 +  fill(255, 230, 150);
17 +  textSize(48);
18 +  textAlign(CENTER, CENTER);
19 +  text("A Perfect Evening", width / 2, 180);
20 +
21 +  // Success message box
22 +  fill(40, 50, 40, 200);
23 +  rectMode(CENTER);
24 +  rect(width / 2, 350, 550, 200, 15);
25 +
26 +  // Success text
27 +  fill(220, 255, 220);
28 +  textSize(22);
29 +  text("You uncovered the party's secrets", wid
  +th / 2, 290);
30 +  text("while maintaining everyone's trust!", w
  +idth / 2, 320);
31 +
32 +  // Stats summary
33 +  fill(180, 220, 180);
34 +  textSize(18);
35 +  text("Final Stats:", width / 2, 370);
36 +  text("Secrets: " + secrets + "  |  Trust: " +
  + trust + "  |  Social: " + social, width / 2, 4
  +00);
37 +
38 +  // Ending description
39 +  fill(200, 230, 200);
40 +  textSize(16);
41 +  text("The host invites you to their next excl
  +usive gathering.", width / 2, 480);
42 +  text("You've become the talk of high society.
```

```
  +", width / 2, 510);
43 +
44 +  // Restart instruction
45 +  fill(180, 200, 180);
46 +  textSize(18);
47 +  text("Press R to play again", width / 2, 600)
  +;
48 +}
49 +
50 +function winMousePressed() {
51 +  resetGame();
52 +}
53 +
54 +function winKeyPressed() {
55 +  if (key === "r" || key === "R") {
56 +    resetGame();
57 +  }
58 +}
```

🟠 Write(lose.js)

└ Added 76 lines, removed 45 lines

```
 1 -// NOTE: Do NOT add setup() or draw() in this f
  -ile
 2 -// setup() and draw() live in main.js
 3 -// This file only defines:
 4 -// 1) drawLose() → what the lose screen looks l
  -ike
 5 -// 2) input handlers → how the player returns t
  -o the start screen
 6 -
 7 -// ------------------------------
 8 -// Main draw function for lose screen
 9 -// ------------------------------
10 -// drawLose() is called from main.js
11 -// only when currentScreen === "lose"
12 -function drawLose() {
13 -  // Red-tinted background to communicate failu
  -re
14 -  background(255, 210, 210);
15 -
16 -  fill(0);
17 -  textAlign(CENTER, CENTER);
18 -
19 -  // Main message
20 -  textSize(40);
21 -  text("You Lose!", width / 2, 300);
22 -
23 -  // Instruction text
```

```
24 -  textSize(20);
25 -  text("Click or press R to return to Start.",
   -width / 2, 360);
26 -}
27 -
28 -// ----------------------------
29 -// Mouse input for lose screen
30 -// ----------------------------
31 -// Any mouse click returns the player to the st
   -art screen
32 -// (no buttons needed for this simple end state
   -)
33 -function loseMousePressed() {
34 -  currentScreen = "start";
35 -}
36 -
37 -// ----------------------------
38 -// Keyboard input for lose screen
39 -// ----------------------------
40 -// R is commonly used for “restart” in games
41 -function loseKeyPressed() {
42 -  if (key === "r" || key === "R") {
43 -    currentScreen = "start";
44 -  }
45 -}
 1 +// ----------------------------------------------
   +----------------
 2 +// lose.js - Bad Ending Screen
 3 +// ----------------------------------------------
   +----------------
 4 +
 5 +function drawLose() {
 6 +  // Dark, somber background
 7 +  background(70, 50, 55);
 8 +
 9 +  // Decorative elements
10 +  fill(90, 60, 65, 100);
11 +  noStroke();
12 +  ellipse(150, 650, 250, 250);
13 +  ellipse(650, 150, 200, 200);
14 +
15 +  // Title
16 +  fill(255, 180, 180);
17 +  textSize(48);
18 +  textAlign(CENTER, CENTER);
19 +  text("An Awkward Exit", width / 2, 180);
20 +
21 +  // Failure message box
```

```
22 +  fill(50, 35, 40, 200);
23 +  rectMode(CENTER);
24 +  rect(width / 2, 350, 550, 200, 15);
25 +
26 +  // Determine what went wrong
27 +  let failReason = "";
28 +  if (secrets < 2 && trust < 45) {
29 +    failReason = "You didn't uncover enough sec
   +rets and lost the guests' trust.";
30 +  } else if (secrets < 2) {
31 +    failReason = "You played it too safe and di
   +dn't learn any real secrets.";
32 +  } else {
33 +    failReason = "Your prying cost you the trus
   +t of the other guests.";
34 +  }
35 +
36 +  // Failure text
37 +  fill(255, 200, 200);
38 +  textSize(22);
39 +  text("The party didn't go as planned...", wid
   +th / 2, 290);
40 +
41 +  fill(220, 180, 180);
42 +  textSize(18);
43 +  text(failReason, width / 2, 330);
44 +
45 +  // Stats summary
46 +  fill(200, 160, 160);
47 +  textSize(18);
48 +  text("Final Stats:", width / 2, 370);
49 +  text("Secrets: " + secrets + "  |  Trust: " +
   + trust + "  |  Social: " + social, width / 2, 4
   +00);
50 +
51 +  // Ending description
52 +  fill(180, 150, 155);
53 +  textSize(16);
54 +  text("You leave early, feeling like an outsid
   +er.", width / 2, 480);
55 +  text("Maybe next time you'll strike the right
   + balance.", width / 2, 510);
56 +
57 +  // Goal hint
58 +  fill(200, 180, 180);
59 +  textSize(14);
60 +  text("(Goal: secrets >= 2 AND trust >= 45)",
   +width / 2, 560);
```

	61 +
	62 + // Restart instruction
	63 + fill(200, 170, 175);
	64 + textSize(18);
	65 + text("Press R to try again", width / 2, 620);
	66 +}
	67 +
	68 +function loseMousePressed() {
	69 + resetGame();
	70 +}
	71 +
	72 +function loseKeyPressed() {
	73 + if (key === "r" || key === "R") {
	74 + resetGame();
	75 + }
	76 +}
	pand) “

“Right now, all the pages are the same color. The storyline is great, but visual can be improved. Can you try to make every page a different color to represent the different characters”	“Now let me update the index.html to include the new script files and update the page info. 🟠 Write(index.html) └ Added 9 lines, removed 6 lines 4 <meta charset="utf-8" /> 5 <meta name="viewport" content="width=device-width, initial-scale=1" /> 6 <meta name="theme-color" content="#1e3a3a" /> 7 - <title>GBDA302 — Week 3 Demo</title> 7 + <title>The Dinner Party</title> 8 <link rel="preconnect" href="https://cdn.jsdelivr.net" /> 9 <link rel="stylesheet" href="style.css" /> 10 </head> 11 <body> 12 <header class="site-header"> 13 14 - <h1 class="title">Win or Lose</h1> 15 - GBDA302 • Week 3 — -Game States, UI, Menus 14 + <h1 class="title">The Dinner Party</h1> 15 + GBDA302 - Interacti +ve Story Game 16 17 </header> 18 ... 22 <!-- p5 canvas mounts here -->

```
23          </div>
24          <p class="hint">
25 -          Tip: use Enter / I on the start scree
  -n. Press R on win/lose to return.
25 +          Tip: Press ENTER to start. Talk to gu
  +ests, make choices, and try to get the good end
  +ing!
26          </p>
27        </section>
28      </main>
...
30      <footer class="site-footer">
31        <div class="wrap">
32          <small
33 -          >© 2026 Dr. Karen Cochrane &amp;amp; Davi
  -d Han · Built with p5.js</small
33 +          >Built with p5.js</small
34          >
35        </div>
36      </footer>
37
38      <script src="https://cdn.jsdelivr.net/npm/p
    5@1/lib/p5.min.js"></script>
39 -    <!-- your existing files (order matters) --
  ->
39 +    <!-- Screen files (order matters - load bef
  +ore main.js) -->
40      <script src="start.js"></script>
41      <script src="instructions.js"></script>
42      <script src="game.js"></script>
43 +    <script src="host.js"></script>
44 +    <script src="chef.js"></script>
45 +    <script src="critic.js"></script>
46      <script src="win.js"></script>
47      <script src="lose.js"></script>
48      <script src="main.js"></script>
```

⏺ Let me check the instructions.js file to make sure it won't cause issues.

⏺ Read(instructions.js)
  ⎿ Read 106 lines

⏺ The game is complete. Here's a summary of what was created:

Files Modified:
- main.js - Updated with game state routing, player stats (secrets, trust, social), guest tracking, and shared helper

	functions - start.js - "The Dinner Party" start screen with dark elegant theme - game.js - Party room hub with buttons to talk to each guest and "Finish Party" button - win.js - Good ending screen showing success message and final stats - lose.js - Bad ending screen with contextual failure message - index.html - Updated title and included new script files Files Created: - host.js - Host conversation with 2 choices (compliment → trust/social, ask drama → secrets) - chef.js - Chef conversation with 2 choices (praise food → trust/social, ask problems → secrets) - critic.js - Critic conversation with 2 choices (flatter → social/trust, press for details → secrets) Game Flow: 1. Start screen → Press ENTER or click BEGIN 2. Party room → Choose which guest to talk to (shows "(done)" after talking) 3. Guest screens → Make a choice that affects stats 4. Return to party room → Talk to more guests or click "Finish Party" 5. Ending → Good ending if secrets >= 2 AND trust >= 45, otherwise bad ending 6. Press R to restart Controls: - ENTER to start - Click buttons or use number keys (1, 2, 3) in party room - B to go back without choosing in guest screens - R to restart from endings To test it, open index.html in a browser. The strategy for the good ending is to dig for secrets from at least 2 guests while keeping trust at 45 or above. “
“Add the two choice buttons to each guest screen with the stat changes I provided. Make the stats actually update when clicked because right now they are not”	“**host.js (lines 50-109)**: - Added choice buttons with `drawChoiceButton()` helper - `hostMousePressed()` function that checks button clicks - Updates stats: Choice 1 → `trust += 15; social += 10` - Updates stats: Choice 2 → `secrets += 1; trust -= 5` - Sets `talkedToHost = true` after choice **chef.js (lines 50-109)**:

	- Added choice buttons with `drawChoiceButton()` helper - `chefMousePressed()` function that checks button clicks - Updates stats: Choice 1 → `trust += 10; social += 15` - Updates stats: Choice 2 → `secrets += 1; social -= 10` - Sets `talkedToChef = true` after choice **critic.js (lines 50-111)**: - Added choice buttons with `drawChoiceButton()` helper - `criticMousePressed()` function that checks button clicks - Updates stats: Choice 1 → `social += 20; trust += 5` - Updates stats: Choice 2 → `secrets += 1; trust -= 10; social -= 5` - Sets `talkedToCritic = true` after choice **Key fix**: Each `MousePressed()` function now directly modifies the global stat variables (`secrets`, `trust`, `social`) before changing `currentScreen` back to "party"