

# Process & Decision Documentation

## Project/Assignment Decisions

### Side Quests and A4 (Individual Work)

- I intended for a simple game to focus more on the structure of the code (different files for different functions, rather than having code all in one file), which was one of the key takeaways from the in-class lesson
- Did not intend for this to be a polished and final draft of a game, therefore I focused more on the learning objectives and the organizational structure, rather than aesthetics and the actual story idea itself
- Decided to go with having 3 doors to choose from, each having different outcomes based on player decision
  - Showed various different screens based on choices but also probability since the outcomes were also randomized
  - I chose to add randomization since the player would know what's behind each door after playing just once or twice. This keeps it fun as the player chooses the door, but the outcomes are always different and random

## Role-Based Process Evidence

The screenshot displays a code editor with a file explorer on the left, a main code editor in the center, and a chat window on the right. The file explorer shows a project named 'AZAMORA\_SIDEQUEST\_W3' with files like 'game.js', 'door1.js', 'door2.js', 'door3.js', 'main.js', and 'win.js'. The main code editor shows the 'door1.js' file with JavaScript code for a game. The chat window on the right contains text explaining the game's structure and providing instructions for the 'Main Screen'.

```
96 // win or lose
97 // This function is called
98 // only when currentScreen === "door1"
99 function door1MousePressed() {
100 // Check if the door is clicked
101 if (isHover(door1Door)) {
102   currentScreen = "level2";
103 }
104 // Check if the gift box is clicked
105 if (isHoverGiftBox(giftBox)) {
106   // 50/50 chance: bomb (lose) or cookie (win)
107   if (random() < 0.5) {
108     currentScreen = "lose"; // bomb
109   } else {
110     currentScreen = "win"; // cookie
111   }
112 }
113 }
114 // Keyboard input for this screen
115 // Keyboard input for this screen
116 // Allows keyboard-only interaction
117 function door1KeyPressed() {
118 // SPACE or ENTER to go through door
119 if (keyCode === 32 || keyCode === ENTER) {
120   currentScreen = "level2";
121 }
122 // No keyboard input for this screen
123 }
124 }
```

Similar to how there are different screens for winning/losing the game, make an interactive story game that has different outcomes based on player decision. The main screen will remain where the story starts after the player clicks play. Keep the formatting similar to how it is currently, where functions (screens) are separated by file.

The "story" will be a pizza maker. After clicking start, have the dough set out already (one large tan circle). Have a pepperoni topping choice, and a cheese topping choice.

To create an interactive story game based on your requirements, we will need to implement several components. Here's a breakdown of what we will do:

1. Main Screen: This will be the starting point where the player clicks "Play" to begin the story.

6 files changed +220 -27

File	Changes
JS door1.js	+62 -27
JS door2.js	+50 -1
JS door3.js	+50 -1
JS game.js	+35 -62
index.html	+3 -0
main.js	+20 -2

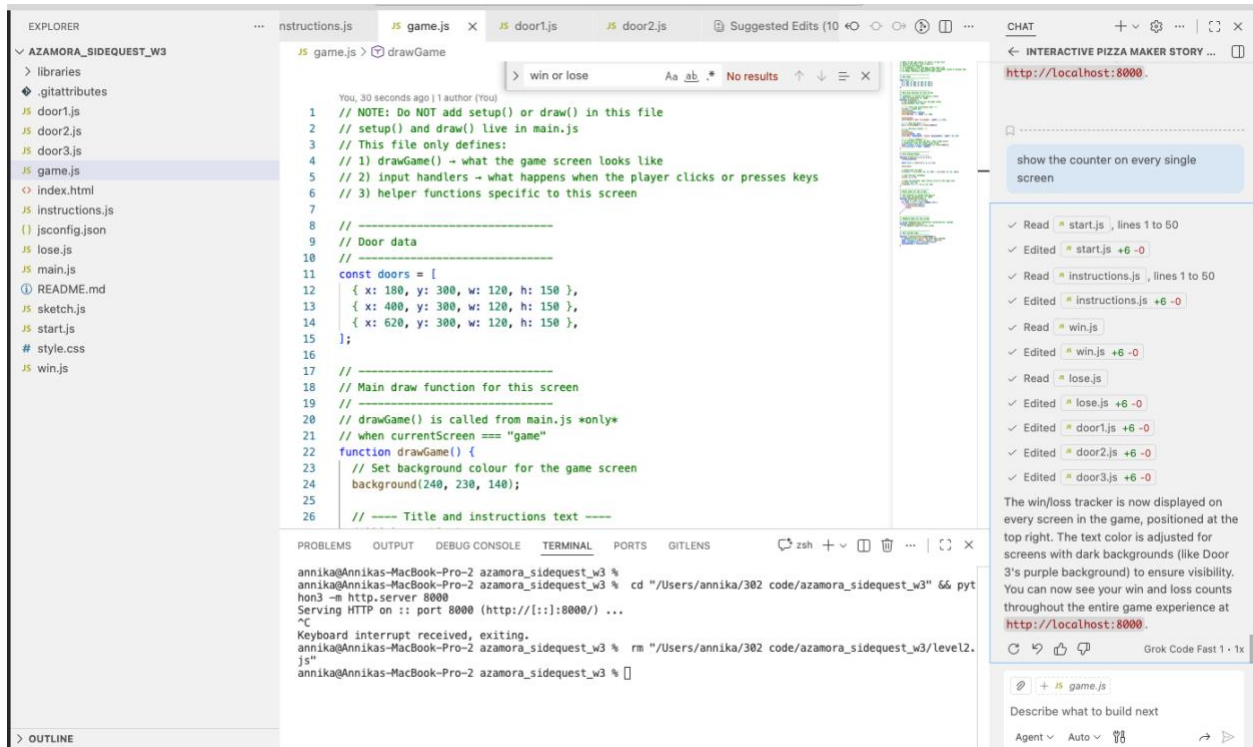
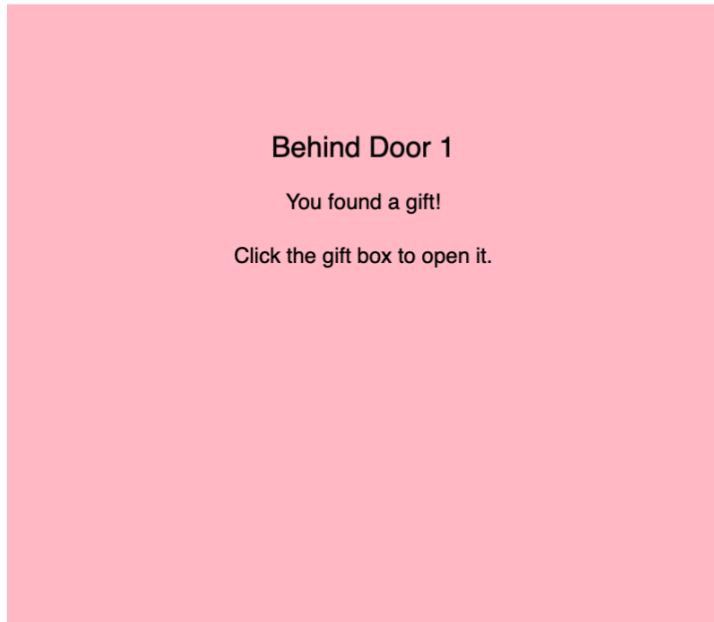
Describe what to build next

Agent Auto

## Pick a Door

GBDA302 • Week 3 — Game States, UI, Menus

Tip: use Enter / I on the start screen. Press R on win/lose to return.



### *Goal of Work Session*

- Modifying the example code from the in-class lesson into a story/player-led game
- Create and input prompts to translate my ideas into executable code
- Apply the organizational structure that we learned in class into this side quest

### Tools, Resources, or Inputs Used

- GenAI tools (if used)
- Base code from Learn (Week 3 Example)

### *GenAI Documentation*

**Date Used:** January 29, 2026 / February 1, 2026

**Tool Disclosure:** Visual Studio Code CoPilot Addon

**Purpose of Use:** To translate and write my ideas/inputs into code that the program could understand and execute. Also used to refine those ideas and modify areas of code.

**Summary of Interaction:** This tool was used to write code based on the prompts I input. I prompted for small changes at a time so that I could modify and request the tool for specific changes of the code it produced, or if I liked it I would keep the code and move on to the next part.

**Human Decision Point(s):** Some areas that were simple for me to change and did not need Copilot to change for me were the text contents. It had already positioned how the text would display on the screen, and also chose what the text said, but if it didn't like it I would just go find where in the code it executes that text, and change the contents. I also made sure, before I even touched the code, that I had an initial and organized base of an idea that I wanted Copilot to write code for. This prioritized organization, as I'd already have an intended output of what the code was going to display as.

**Integrity & Verification Note:** I knew that in order to use this tool effectively, based on what has worked best for me in the past, was to have lots of prompts, but have each prompt be focusing on one small section at a time. This also worked better for me as a user as it allowed me to work section by section, rather than being overwhelmed and trying to prompt it to do everything at once. This strategy also allowed me to see each change it made and how it affected the gameplay/code, by running the code every time I prompted for a new idea/modification.

**Scope of GenAI Use:** I developed the creative concept of the game, meaning that I told GenAI exactly what idea I wanted and it just translated that into code that it could run.

**Limitations or Misfires:** The main/only misfire was when I was prompting it to draw objects after "opening" the gift box. The gift box was suddenly not visible, and I believed something was likely being drawn on top of it. This took about two prompts to get it to be visible again, meaning it didn't take too long to fix, but was definitely an unexpected misfire.

### *Summary of Process (Human + Tool)*

- Each prompt input was requesting a small change

- More prompts, more detail and focus on small sections for each prompt
- When in the process of adding a new aspect to the game, first prompted for the general idea, then modified it by requesting specific detail changes
- Ran and tested the code after each change
  - Since the outcomes were randomized, I made sure that every time I tested the code, I tested every possible outcome to make sure it was working properly

### *Decision Points & Trade-offs*

- Focused on the structure of the code rather than the aesthetics
  - A player-led story game requires having multiple outcomes, meaning there were many options to account for. It was important to be organized and was honestly a bit overwhelming when starting out.<sup>2</sup>

### *Verification & Judgement*

- Playtesting: run the game after every change to see what updates it made
- Often asked the question “Does this make sense?” when coming up with the overall concept and small idea features of the game (idea and story of the game, not the execution and looks of the code)

### *Limitations, Dead Ends, or Open Questions*

- I simplified what “images” I wanted drawn since I know it can be difficult to create those images just by using shapes.
  - For example, I wanted a banana peel drawn but figured that would be too complicated to prompt for (especially since that wasn’t my focus for this side quest), so I opted for a puddle instead which was an ellipse.)
- Sometimes felt that the

## Appendix

**Input:** On the game screen, have 3 doors (brown rectangles with small yellow circle as the door knob)

**Output:** Read [game.js](#)

Edited

game.js+9-0

Edited

game.js+8-3

Edited

game.js+21-0

Edited

game.js+8-0

Edited

game.js+11-0

Edited

game.js+3-3

The local server is now running on port 8000. You can test the game by opening <http://localhost:8000> in your browser. The game screen should now display three brown doors with yellow doorknobs. Clicking on the left or right door will take you to the win screen, while clicking the middle door leads to the lose screen. The button below still provides a random outcome as before. Let me know if you'd like any adjustments!

**Input:** center the doors so that the middle door is in the middle of the canvas

**Output:** The doors have been repositioned to center them on the canvas. The middle door is now at the exact center ( $x=300$ ), with the left door at  $x=200$  and the right door at  $x=400$ . This creates a more balanced and centered appearance on the game screen. You can refresh the page at <http://localhost:8000> to see the updated layout.

**Input:** the door on the most right is in the middle, make the second door in the middle of the canvas

**Output:** The doors have been repositioned so that the middle door (second door) is now at the center of the canvas ( $x=400$ ). The positions are now: left door at  $x=300$ , middle door at  $x=400$ , and right door at  $x=500$ . This centers the second door as requested. You can refresh the page at <http://localhost:8000> to see the updated layout.

**Input:** make the doors bigger

**Output:** The doors have been made bigger: the width increased from 80 to 120 pixels, and the height from 100 to 150 pixels. The doorknobs have also been scaled up from 10 to 15 pixels in diameter to match. This makes the doors more visually prominent on the game screen. You can refresh the page at <http://localhost:8000> to see the larger doors.

**Input:** leave more space between the doors so they do not overlap

**Output:** The doors have been repositioned to eliminate overlap. The new positions are: left door at  $x=280$ , middle door at  $x=400$ , and right door at  $x=520$ . This provides exactly 120 pixels of spacing between door centers, matching the door width, so they no longer overlap. You can refresh the page at <http://localhost:8000> to see the updated spacing.

**Input:** do not put the doors right next to each other, have a 100px gap between each door

**Output:** The doors have been repositioned to have a 100-pixel gap between each door. The new positions are: left door at  $x=180$ , middle door at  $x=400$  (still centered), and right door at  $x=620$ . This

ensures there's exactly 100 pixels of space between the edges of adjacent doors. You can refresh the page at <http://localhost:8000> to see the updated spacing.

**Input:** remove the press here button

**Output:** The "PRESS HERE" button has been completely removed from the game screen. Now the screen only displays the three doors, and players can only interact by clicking on the doors to proceed to win or lose screens. The random outcome button and its associated keyboard input (ENTER key) have also been removed. You can refresh the page at <http://localhost:8000> to see the updated game screen with only the doors.

**Input:** make 3 separate files, one for each door

**Output:** I've created three separate files for each door outcome: door1.js, door2.js, and door3.js. Each file defines its own screen with a unique message and background color (green for winning doors, red for the losing door). The game now transitions to these individual door screens when a door is clicked, and players can return to the start screen by clicking anywhere or pressing SPACE/ENTER.

door1.js: Left door - "You chose the left door! You win!" (green background)

door2.js: Middle door - "You chose the middle door! You lose!" (red background)

door3.js: Right door - "You chose the right door! You win!" (green background)

The necessary updates have been made to main.js and index.html to include these new screens and their input handlers. You can test the game at <http://localhost:8000> to see the separate door outcome screens in action.

**Input:** behind door 1, make another screen that isn't the win/lose screen. make it a pink screen

**Output:** Edited door1.js+4-4

The screen behind Door 1 has been updated to a pink background with a secret room message. Instead of a standard win screen, it now displays "Behind Door 1" with "You found a secret room! Congratulations!" and instructions to click anywhere to return to start. You can test this by running the game at <http://localhost:8000> and clicking the left door.

**Input:** behind door 2, make it blue. door 3 make it purple

**Output:** Edited door2.js+4-4

Edited door3.js+5-5

The screens behind the doors have been updated with the requested colors:

- **Door 2:** Now has a light blue background with the message "You discovered a hidden path! Adventure awaits!"
- **Door 3:** Now has a purple background with white text saying "You unlocked a mystery! What secrets lie ahead?"

All three doors now lead to unique, non-win/lose screens with different colors and messages. You can test the game at <http://localhost:8000> to see the new door outcomes.

**Input:** Now assign each door randomly so when the left door is clicked it might be the output of door 2, etc.

**Output:** Edited game.js+4-7

The door outcomes have been randomized! Now, clicking any of the three doors will randomly display one of the three unique screens (pink secret room, blue hidden path, or purple mystery) with equal probability. The specific door clicked no longer determines the outcome—each click is a surprise. You can test this randomness by refreshing the page and clicking different doors multiple times at <http://localhost:8000>.

**Input:** for the door 1 file, make it so that the player can go through the door the first time, then see another set of doors

**Output:** The door1 screen has been updated to include progression! Now, when a player reaches the pink "Behind Door 1" screen, they see a door that they can click (or press SPACE/ENTER) to "go through" and proceed to a new "Level 2" screen with another set of three doors. These Level 2 doors all lead to the win screen, providing a sense of advancement. The necessary files ([level2.js](#)), routing in [main.js](#), and script inclusion in [index.html](#) have been added. You can test this new progression at <http://localhost:8000> by clicking doors until you randomly get the door1 outcome, then clicking the door on that screen to continue.

**Input:** scrap that last idea, show a gift box when the door 1 file is chosen

**Output:** The door1 screen has been updated to display a gift box instead of a door. The screen now shows a pink background with the message "You found a gift!" and a simple gift box graphic (a square box with a red ribbon cross). Clicking anywhere or pressing SPACE/ENTER returns to the start screen.

The previous progression idea with level2 has been completely removed. You can test this at <http://localhost:8000> by clicking doors until you randomly get the door1 outcome.

make it so that the gift box can be clicked to open. Make it a 50/50 chance that inside of the present is a bomb (player loses), or a cookie (player wins)

The gift box in the door1 screen is now interactive! Players can click on it to open, with a 50/50 chance of revealing a bomb (leading to the lose screen) or a cookie (leading to the win screen). The cursor changes to a hand when hovering over the gift box, and the instructions have been updated to "Click the gift box to open it." The keyboard input has been removed since interaction is now mouse-only for this screen. You can test this exciting gamble at <http://localhost:8000> by clicking doors until you reach the door1 outcome, then clicking the gift box.

**Input:** there is no gift box visible on the canvas

**Output:** The gift box drawing has been updated to make it more visible. It now has a filled light brown box with a prominent red ribbon cross on top. The gift box should now be clearly visible on the pink background in the door1 screen. You can refresh the page at <http://localhost:8000> to see the improved gift box graphic. If it's still not visible, let me know and we can adjust the colors or size further.

**Input:** the gift box is not visible at all because it is not being drawn on the canvas/ something is being drawn on top of it

**Output:** The issue was that width and height are not available when the file is loaded, so the gift box position was undefined. I've fixed it by using fixed coordinates (x: 400, y: 450) based on the 800x800 canvas size. The gift box should now be visible on the door1 screen. Refresh the page at <http://localhost:8000> to see the gift box appear. If it's still not showing, there might be another issue—let me know what you see!

**Input:** draw a bomb above the present (black circle with small rectangle on top) if when clicked, the random generator chooses bomb. Draw a chocolate chip cookie above the present if chooses cookie

**Output:**

1. **First click:** Opens the gift box and reveals either a bomb (black circle with fuse) or a chocolate chip cookie (brown circle with black dots) above the gift box. The text changes to "Click to continue."
2. **Second click:** Proceeds to the appropriate outcome screen (lose for bomb, win for cookie).

The gift box now provides visual feedback on what's inside before transitioning. You can test this at <http://localhost:8000> by clicking doors until you reach the door1 screen, then clicking the gift box twice to see the reveal and outcome.

Instead of saying "you found a gift", either say if they found a cookie or bomb

The text has been updated so that after opening the gift box, it reveals what was inside: "You opened the gift and found a bomb!" or "You opened the gift and found a cookie!" This provides immediate feedback on the outcome. The initial text remains "You found a gift!" until the box is opened. You can test this change at <http://localhost:8000>.

**Input:** Instead of saying "you found a gift", either say if they found a cookie or bomb

**Output:** The text has been updated so that after opening the gift box, it reveals what was inside: "You opened the gift and found a bomb!" or "You opened the gift and found a cookie!" This provides immediate feedback on the outcome. The initial text remains "You found a gift!" until the box is opened. You can test this change at <http://localhost:8000>.

**Input:** behind door 2, make it so that they don't win or lose, but have another chance to redo their door selection. Send them back to the screen with all three doors

**Output:** Door 2 has been updated to provide a "second chance" mechanic. Instead of leading to a win/lose outcome, it now displays "You found a shortcut! Back to the doors you go." and returns the player to the main game screen with the three doors for another selection. This adds replayability and strategy to the game. You can test this at <http://localhost:8000> by choosing Door 2 and seeing the loop back to door selection.

**Input:** behind door 3, draw a puddle (blue ellipse). The text should say "You slipped and fell on the puddle. Your clothes are now wet."

**Output:** Door 3 has been updated to show a blue puddle (semi-transparent ellipse) and the text "You slipped and fell on the puddle. Your clothes are now wet." The screen still returns to the start screen on click. This adds a humorous, neutral outcome to Door 3. You can test this at <http://localhost:8000> by selecting Door 3.

**Input:** if player gets the door 3 (puddle), they lose

**Output:** Door 3 now results in a loss for the player. After showing the puddle and the slipping text, clicking anywhere (or pressing SPACE/ENTER) will take the player to the lose screen. This makes Door 3 a risky choice that leads to defeat. You can test this updated outcome at

**Input:** Put a tracker on the top right that keeps track of how many wins and losses the player has

**Output:** A win/loss tracker has been added to the top right of the game screen, displaying "Wins: X Losses: Y". The counters increment whenever the player reaches a win or lose outcome (from the gift box gamble in Door 1 or from slipping in Door 3). Door 2 doesn't affect the counters since it returns to the game screen. You can test this by playing through different door outcomes and seeing the tracker update at <http://localhost:8000>.

Annika Zamora, azamora, 21075753

**Input:** show the counter on every single screen

**Output:** The win/loss tracker is now displayed on every screen in the game, positioned at the top right. The text color is adjusted for screens with dark backgrounds (like Door 3's purple background) to ensure visibility. You can now see your win and loss counts throughout the entire game experience at <http://localhost:8000>.