Process & Decision Documentation

Side Quests and A4 (Individual Work)

I changed the game so that there are two doors, and a player has to pick either the left or the right door to see if they win or lose. I had to change the positioning of the clickable area after not modifying the original code, as it was too low.

GenAI Documentation

Date Used: February 1st 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: Brainstorming, Debugging code and commenting code

Summary of Interaction: I used ChatGPT to generate some ideas of how to modify the class examples to meet the sidequest requirements. I also used it explain the errors I made while coding and whether I should add comments and what they should be.

Appendix

Prompt: can you explain this to me? right now I am trying to modify code that is a simple win lose game: Build an interactive story that unfolds through multiple game states and files, branching like a small decision tree

Yeah, totally — this line is *way scarier than it actually is* 😅
Let's break it down in plain language and tie it directly to your **simple win/lose game**.

What that sentence really means

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

1. Interactive story

This just means:

- The player **makes choices**
- Those choices **change what happens next**

So instead of:

“You win” or “You lose”

You now have:

“You chose A, so this happens…
You chose B, so something else happens…”

---

2. Multiple game states

A **game state** is just “where the player is right now.”

Examples of states:

- Start screen
- In the forest
- Inside the house
- Talking to a character
- Win ending
- Lose ending

Think of it like pages in a storybook.

📌 If you already have **win** and **lose**, congrats — you already have *multiple game states*.

You’re just adding **more in-between states**.

---

3. Through multiple files

This does **NOT** mean anything fancy.

It usually means:

- Each part of the story lives in its **own file**
- Instead of one giant file with everything

Example:

main.rkt → starts the game

forest.rkt → forest scene

house.rkt → house scene

ending-win.rkt → win ending

ending-lose.rkt → lose ending

Each file represents **one state or scene**.

---

4. Branching like a small decision tree

This is the key idea 🌳

A **decision tree** just means:

- One choice → leads to different paths

Example:

Start

├— Go left → Forest

│ ├— Run → Lose

│ └— Hide → Win

└— Go right → House

├— Open door → Win

└— Leave → Lose

That's it. That's the "tree."

How this builds on your win/lose game

Before (what you probably have now)

Start → Play → Win OR Lose

After (what they want)

Start

→ Choice 1

→ Choice 2

→ Win OR Lose

Still ends in **win/lose** — just more story on the way.

Very simple example (conceptual)

Let's say your game starts like this:

State: Start

You wake up in a strange place.
Do you:

1. Explore
2. Go back to sleep

- If **Explore** → load forest.rkt
- If **Sleep** → load ending-lose.rkt

That's already:

- interactive
- multiple states
- branching
- win/lose endings

Big reassurance 😌

You are **not** being asked to:

- make a complex RPG
- use advanced algorithms
- build anything huge

They want:
✔ a few choices
✔ a few states

✔️ more than one file
✔️ at least one win and one lose path

If you want, next we can:

- map **your existing win/lose game** into a decision tree
- or I can help you decide **how many states/files** is "enough" for the assignment
- or translate this directly into **Racket/DrRacket-friendly structure** (since I know you're using BSL)

Just tell me what part feels most confusing right now 💛

Prompt: can you give me some ideas?

Got you — **ultra-simple, no overthinking, "I can finish this" ideas** coming up 😌
These are all **small**, **clear**, and perfect for p5.js with just a few gameStates.

---

1. "Pick a Door" 🚪 (classic + very easy)

Story
You see two doors.

States

start

├─ Left door → win

└─ Right door → lose

Why it's good

- 3 states total
- Literally rectangles + text
- Zero plot complexity

---

2. "Alarm Clock" ⏰

Story
Your alarm goes off.

States

start (alarm ringing)

├─ Snooze → lose

└─ Get up → win

That's it.
Very relatable, very fast to build.

---

3. "Cross the Street" 🚦

Story
You need to cross the road.

States

start (crosswalk)

├─ Cross now → lose

└─ Wait for light → win

Visuals

- Red/green circle
- Stick figure
- Super minimal

---

4. "Choose a Snack" 🍪

Story
You're hungry.

States

start

├─ Healthy snack → win

└ Mystery snack → lose

(You can make this funny instead of moral if you want.)

---

5. “Save the File” 💾 (VERY student-coded)

Story
You finished an assignment.

States

start

├ Save file → win

└ Close laptop → lose

Relatable = easy to explain to a TA.

---

6. “Open the Message” 📩

Story
You get a message notification.

States

start

├ Open it → next state

│ ├ Reply calmly → win

│ └ Panic reply → lose

└ Ignore it → lose

Still tiny, but technically “branching.”

---

7. “Elevator Button” 🛗

Story
You’re in an elevator.

States

start

├─ Press correct floor → win

└─ Press wrong floor → lose

EASIEST structure in p5.js (mentally)

Think:

- **One screen = one state**
- **Two choices max per screen**
- **Text + rectangles**

If it feels "too simple" — it's probably **exactly right** for this assignment.

My top 3 if you want the least stress

1️⃣ **Alarm Clock**
2️⃣ **Save the File**
3️⃣ **Pick a Door**

All are:
✔️ original
✔️ fast
✔️ impossible to mess up

If you want, next I can:

- turn *one* of these into a tiny state diagram
- write **starter p5.js code** you can customize
- help you make it look nicer without adding logic

Just tell me which idea you like 👀

Prompt: why is it that I have to press beneath the doors to see if I win or lose? Why can i press the text of the left or right door in the rectangles to trigger it. How do I fix that?

Ahhh! Now I understand perfectly 😄 — the issue isn't about padding or text alignment; it's that in your current code, **the clickable area is actually somewhere *below* the rectangles**. That usually happens if the y coordinates of the doors don't match the rectangles' **visual position**.

Let me explain:

If your text or visual seems offset, it's usually because the rectangle isn't drawn where you think, or the click code is still using a different instead of the doors.

Likely cause in your code

- That would **only trigger clicks on the old button**, not the doors.
- If you click on the rectangle, it visually looks correct, but the logic is still tied to the **old button position**, which is somewhere below the doors.

✅ That explains why you have to click *underneath the rectangles*.

If you want, I can rewrite your **game.js fully** with this fix applied so it's plug-and-play — it will match the rectangles perfectly and remove the old button.

Do you want me to do that?