

Process & Decision Documentation

Project/Assignment Decisions

In Side Quest Week 5, I developed a meditative, open world game where players control a swan character and aim to carry 7 ducklings to safety. Throughout this Side Quest, I led the decision-making, developed illustrations, and troubleshoot code by ideating a game concept, utilizing Adobe Fresco to create assets, and reviewing lecture slides to refresh my coding knowledge. I then proceeded using GenAI (Claude Haiku 4.5) to support game development in Visual Studio Code. Below is a deeper discussion of my interaction with AI, findings, and recommendations for future GenAI frameworks, with the process detailed in the appendix.

GenAI Documentation

Date Used: February 17-18, 2026

Tool Disclosure: Claude Haiku 4.5 (Copilot on VS Code)

Purpose of Use: I used GenAI for process optimization, assisting with code generation and troubleshooting.

Summary of Interaction: The tool contributed to the JavaScript code development of this game by responding to prompts about specific game mechanics, behavioural outputs, and world elements. I iteratively refined my prompts into more detailed, technically specific, step-by-step instructions (e.g., “replace the player blob in p5.js with my ‘swan.png’ image in the asset folder”) to more accurately guide it in creating a distinct, bug-free game and camera experience for players.

Human Decision Point(s): The process of visual asset development, ideation, and brainstorming was carried out by me independently. Based on the requirements of the Side Quest, the code provided by instructors, and my own interests, I crafted a game concept of a swan who moves around by “flying” and has the goal of the player bringing all ducks to the diving board object for “safety” (a win-state).

Other relevant human decision points include: I created necessary folder and file additions such as “assets and README.md” and added small code up to my skill abilities (additions such as text to the html file and colour additions), then engineered prompts according to my coding knowledge, in-class concepts, and creative ideas, ranging from technically specific such as “to create a realistic experience -- when the swan moves, rotate 90 deg if player input indicates going right, 180 deg if going left, 270 if left and 0 if going up on

keyboard movement” to semantic guidance “whenever a duckie’s position coincides with the diving board, make it stay there in the middle of the diving board rectangle”.

I modified the output as I found that instructions needed to be specific, functional across all files (e.g. update in index.html as well), and iteratively troubleshoot to ensure proper implementation.

Integrity & Verification Note: I checked GenAI for accuracy, bias, appropriateness, and fit by carefully reviewing course policies and testing AI-generated code through manual testing and troubleshooting (utilizing GitHub’s Live Server extension renderings, as well as semantically checking code to ensure brackets were properly structured). Additionally, the inputs I provided were technically specific and heavily human-guided, reducing the risk of bias and plagiarism.

Scope of GenAI Use: In this side quest, AI was only used to edit files in order to add smaller elements such as objects and ducks, update the environment, implement a win state, and make behavioural and mechanic modifications (e.g. “fix the buoy to make the player go 0.4 times faster than its current speed”).

Limitations, Dead-Ends, and Misfires: The tool was powerful in quickly implementing changes across multiple files; however, it frequently produced bugs and required checkpoint restoration due to the number of files and the complexity of objects. I found that it requires prompts grounded in specific ideas rather than vague concepts, forcing me to adjust my mental model and treat it as a semantically based coding framework that responds best to ultra-specific, step-by-step guidance (e.g., frequently using “more specifically...”, “first...”, and “then...”).

Additionally, it often did not retain awareness/knowledge of recently edited files. For example, when I prompted it to add my image of a “swan,” it repeatedly output “swan.jpeg” instead of the correct “swan.png,” leading to frustration and blank Live Server renderings. This led to my discovery of the importance of specifying file extensions, not just file names, in prompts.

Summary of Process (Human + Tool)

- Brainstorming [Human]
- Game Illustrations [Human]
- Core Loop
 - Probing Claude Haiku 4.5 [GenAI]
 - Code Development [GenAI]
 - Troubleshooting [Human]
 - More Specific Prob for GenAI [Human]

- Code Modifications [GenAI]
- Refinement [Human]
- Upload to GitHub [Human]
- Final Test [Human]
- Process Documentation [Human]

Decision Points & Trade-offs

- Options Considered: I considered using GenAI to develop game concept ideas
 - Human Decision: I decided to ideate independently based on game restrictions (a 2D game with limited open-world space).
 - Trade-Off: While GenAI could have provided more novel ideas and greater breath of concept choices, I chose a game concept based on my interests of meditative pool games and the calmness of a bird flying (inspired by my favourite childhood game, “Animal Jam”).
- Options Considered: I considered using an approach of long GenAI prompts
- What Changed: I discovered that my approach needed to shift from long prompts to a step-by-step prompt engineering process. More specific, smaller prompts provided greater control and accuracy, helping to debug effectively and avoid GenAI hallucinations or poor indexing, which could cause it to overlook parts of a prompt.

Future Ethical Framework Ideas

- Avoid brainstorming immediately with GenAI
 - Taking a moment to consider one’s own life experience and individual taste can lead to more meaningful ideation
 - Using GenAI immediately risks limiting one’s imagination to technology produced concepts
 - If an effective idea does not arise, treat it as a back-up method of brainstorming, not the standard
- Avoid brain dumps, aim for specific keywords in prompts.
 - Create specific prompts with minimal information.
 - Consider breaking large prompts into smaller chunks and using human judgment to criticize and combine the output.
 - Using keywords with intention will reduce inaccurate recalls and improve accuracy outputs (like incorrect naming of topics)
 - GenAI may replace memory gaps with biases or stereotypes.
- Break complex ideas into smaller steps
 - Create specific prompts with minimal information.

- Provide specific manner of implementing the change (reduces unexpected or unlearned manner of coding from being employed, e.g. to use random number generation, adding another object file with a specific name, etc...).
- Consider breaking large prompts into smaller chunks and using human judgment to criticize and combine the output.
- Using keywords with intention will reduce inaccurate indexing and recalls and improve accuracy outputs.
- GenAI may replace memory gaps with biases, incorrect technical specifications, or stereotypes (e.g. assuming how large an image of a duck should be).

Appendix

Part 1: Illustration Development

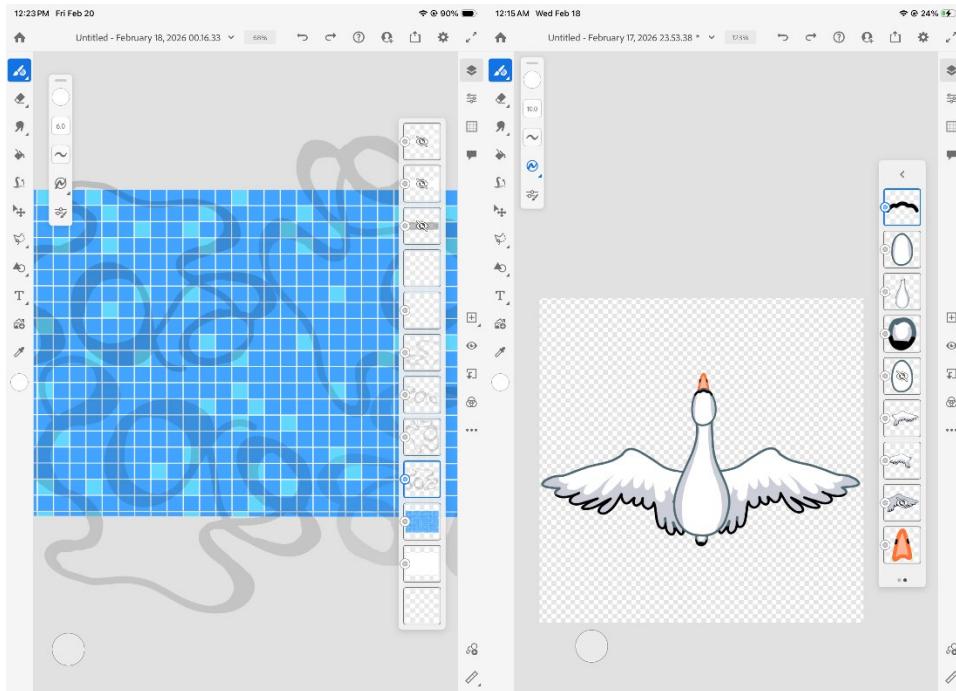


Figure 1-2 : Gaming Illustrations Developed in Adobe Fresco

Part 2: GenAI Conversation

My Prompt

set world.json and actual javascript game so the background is still w 2400 and h 1600 but set to my background image in my assets file "bg"

GenAI Response

I'll help you set up a background image in your game. Let me first check what's in your assets folder and examine the current code.

Now I'll update the files to use your background image. I'll need to:

1. Update [world.json](#) to reference the background image
2. Modify [sketch.js](#) to preload the image
3. Update [WorldLevel.js](#) to draw the background image

Done! I've updated your game to use the background image from your assets folder. Here's what changed:

[world.json](#)

- Changed "bg" from an RGB color array to "assets/bg.jpeg"

[sketch.js](#)

- Added [bgImage](#) variable to store the loaded image
- Added [bgImage = loadImage\("assets/bg.jpeg"\);](#) in [preload\(\)](#) to load it before setup
- Updated the WorldLevel constructor call to pass the image: [new WorldLevel\(worldData, bgImage\)](#)

[WorldLevel.js](#)

- Updated constructor to accept [bgImage](#) parameter
- Modified [drawWorld\(\)](#) to draw the background image at the full world dimensions (2400×1600) instead of filling with an RGB color
- The grid lines and obstacles are drawn on top of the image as before

Your background image will now display as the world background while maintaining the camera system and all game mechanics!

My Prompt

great now remove the grid lines from before

GenAI Response

Grid lines removed!

My Prompt

change the player blob js to my "swan" image in asset file and make so that the direction in which the swan is moving is where the head of it (in the actual image file the head is pointing up/north) is pointing where the cursor is moving. also dont let the character/player to move diagonally, only up down left and right at once

GenAI Response

Done! I've updated the Player to use your swan image with the following changes:

sketch.js

- Added preload for the swan image (assets/character.jpeg)
- Pass the swan image to the Player constructor

Player.js

- Constructor now accepts and stores the swan image
- Added direction tracking (`dirX`, `dirY`) and rotation
- Rewrote `updateInput()` to only allow one direction at a time (no diagonal movement)
 - it checks for right, left, down, up in priority order
- Updated `draw()` to:
 - Display the swan image instead of a blue rectangle
 - Rotate the swan so its head (which points up in the image) faces the direction of movement
 - Use proper p5.js translate/rotate for centering and rotation

The swan will now move strictly in cardinal directions (up/down/left/right only) with its head always pointing toward the direction you're moving!

My Prompt

change the player blob js to my "swan" image in asset file and make so that the direction in which the swan is moving is where the head of it (in the actual image file the head is pointing up/north) is pointing where the cursor is moving. More specifically, rotate 90 deg if going right, 180 deg if going left, 270 if left and 0 if going up on keyboard movement

GenAI Response

Done! I've updated the Player to use your swan image with the following rotations:

Player.js

- Constructor now accepts the swan image
- Added [rotation](#) property to track the swan's facing direction
- Updated [updateInput\(\)](#) to set rotation based on keyboard input:
 - **Up:** 0°
 - **Right:** 90°
 - **Down:** 180°
 - **Left:** 270°
- Updated [draw\(\)](#) to display the swan image rotated to face the direction of movement

sketch.js

- Added swan image preload (assets/character.jpeg)
- Pass the swan image to the Player constructor

The swan will now rotate to face the direction you're moving!

My Prompt

make it so the player can only move in one direction at a time from keyboard input. player should not move diagonally

GenAI Response

Done! The player is now displaying your swan image at 72x72 pixels. The changes include:

- Added swan image preload in [sketch.js:31](#)
- Updated Player constructor to accept and store the image

- Modified the draw method in [Player.js](#) to display the swan image centered at 72x72 pixels

My Prompt

Perfect, now rotate the swan (player) according to the keyboard indicated direction. e.g. rotate 90 deg if going right, 180 deg if going left, 270 if left and 0 if going up on keyboard movement

GenAI Response

Done! The swan now rotates based on keyboard direction:

- **Up:** 0°
- **Right:** 90°
- **Down:** 180°
- **Left:** 270°

The changes include:

- Added [rotation](#) property to track the swan's direction
- Updated [updateInput\(\)](#) to set rotation based on which direction key is pressed
- Modified [draw\(\)](#) to translate and rotate around the swan's position before drawing the image

My Prompt

in the middle left of the background add a white (#7f7f7f7) rectangle that is w 200 h 25 with a shadow (a rectangle below it black with some transparency in the alpha channel and slightly offset down left by 10 pixels). This shall henceforth be the "divingBoard"

GenAI Response

Done! I've added the diving board to the background. It's positioned at the middle left of the world (x: 200, y: vertically centered) with:

- A white rectangle (200×25 px)

- A black shadow beneath it with transparency, offset 10 pixels down and to the right

The diving board is now drawn in [WorldLevel.js:20-37](#) as part of the world rendering.

My Prompt

In random positions across the background each time the page reloads, add 7 ducks that are 64px by 64px of my "ducky" in asset image folder ensure ducky.png not ducky.jpeg

GenAI Response

Done! I've added 7 ducks that appear at random positions across the background each time the page reloads. Here's what I changed:

[sketch.js](#)

- Added `duckyImage` preload loading "assets/ducky.png"
- Passed `duckyImage` to the `WorldLevel` constructor
- Fixed the `keyPressed()` function to pass `swanImage` when resetting the player

[WorldLevel.js](#)

- Updated constructor to accept `duckyImage` and initialize a `ducks` array with 7 ducks at random positions within the world bounds
- Updated `drawWorld()` to draw each duck as a 64×64 pixel image

Each duck is positioned randomly within the world (0-2400 width, 0-1600 height) and will get new random positions every time you reload the page.

My Prompt

Great now have the duckies 112px by 112px and make so they spawn in different rotation orientations randomize

GenAI Response

Done! The ducks are now 112×112 pixels and spawn with random rotations. Each duck has a `rotation` property initialized to a random angle (0 to 2π radians) and is drawn with that rotation applied.

My Prompt

have the player start where y is the middle of the background

GenAI Response

Done! The player now starts at y: 800, which is the vertical middle of the 1600px tall background.

My Prompt

when the player hovers within the space (142px by 142px) of the duckies make the ducky the position of the player but offset by 5px bottom right (semantically the swan player picks up the duck and can move anywhere with it now)

GenAI Response

Done! I've implemented the duck carrying mechanic. When the player (swan) gets within 142px of a duck, the duck will now follow the player with a 5px offset to the bottom-right. Here's what I added:

WorldLevel.js

- Added `carried` property to each duck to track if it's being picked up
- Added `carriedDuck` property to track which duck is currently being carried
- Added `checkDuckCollision(player)` method that detects if the player is within 142px of any duck and marks it as carried
- Updated duck drawing to follow the player when carried, positioned with a 5px bottom-right offset

sketch.js

- Added collision check in the draw loop after player movement
- Updates the carried duck's position to follow the player

Now the swan can "pick up" ducks by moving close to them, and carry them around!

My Prompt

Once a ducky is carried it follows the player forever (indefinitely) - does not go back to its position ever again

GenAI Response

Done! I've fixed the collision logic so that once a duck is picked up, it follows the player forever. The key change is that if a duck is already being carried, the collision check immediately returns without releasing it. The duck will never go back to its original position.

My Prompt

Now, when the player hovers over any ducky it carries it. Not just one, but all duckies on the map this works with on by one

GenAI Response

Done! I've updated the system so the player can now carry all ducks simultaneously. Here's what changed:

WorldLevel.js

- Changed carriedDuck (single index) to [carriedDucks](#) (array of indices)
- Updated [checkDuckCollision\(\)](#) to add ducks to the carried array instead of replacing a single duck
- Ducks that are already carried are skipped, so no duplicates
- Updated the drawing logic to offset each carried duck diagonally based on its position in the array (5px + index * 15px offset), so they stack nicely instead of overlapping

sketch.js

- Updated the loop to iterate through all [carriedDucks](#) and update their positions

Now when the player gets within 142px of any duck on the map, it gets picked up. Multiple ducks follow the player in a staggered arrangement, each adding its own offset for visibility.

My Prompt

Have the duckies be "carried" by the player closer to the player. the duckies should be progressively carried in a farther distance from the player.

GenAI Response

Done! I've increased the spacing of the carried ducks. Now they start closer to the player (20px offset for the first duck) and progressively get further away (each subsequent duck is 45px further), creating a nice trailing effect.

My Prompt

whenever a duckie's position coincides with the diving board, make it stay there in the middle of the diving board rectangle. this behaviour should occur to all duckies when the player brings them there or if it happens to spawn there. Also at top right of canvas add a counter with e.g. "0/7 duckies" duckies where is sets how many duckies are on the divingboard. a duckie on the diving board should not be able to be moved after being stationed there or randomly spawning there

GenAI Response

Perfect! I've implemented the diving board mechanism for the ducks. Here's what I added:

WorldLevel.js

- Added diving board properties (position, dimensions, center coordinates)
- Added [`isDuckOnBoard\(\)`](#) method to check if a duck is within the diving board boundaries
- Added [`onBoard`](#) property to each duck during initialization
- Ducks that randomly spawn on the board are automatically marked as [`onBoard`](#)
- Updated [`checkDuckCollision\(\)`](#) to skip ducks already on the board
- Updated duck drawing to place ducks at the center of the diving board if they're [`onBoard`](#)
- Added a counter in the HUD showing "X/7 duckies" at the top right

sketch.js

- Updated the carried duck logic to check if ducks reach the diving board while being carried

- When a carried duck reaches the board, it's automatically placed there and locked (removed from the `carriedDucks` array)

Now when the player brings a duck to the diving board (or if a duck spawns there), it will:

1. Snap to the center of the diving board
2. Stay there permanently (cannot be moved or picked up again)
3. Update the counter to show how many ducks are on the board

My Prompt

Make the duckies stay where placed on the divingboard (the spot where it first touches the diving board, but slightly offset closer to the center of the board by 10px)

GenAI Response

Done! I've updated the system so ducks stay where they first land on the diving board.

Here's what changed:

WorldLevel.js

- Added `calculateBoardPosition()` method that takes a position and moves it 10px closer toward the center of the diving board
- Updated duck initialization to calculate and store `boardPosition` for ducks that spawn on the board
- Updated duck drawing to use the stored `boardPosition` instead of centering all ducks

sketch.js

- When a carried duck reaches the board, it now calculates and stores its `boardPosition` with the 10px offset toward center
- The duck stays at that exact position permanently

Now each duck will land where the player brings it and stay there (slightly offset toward the center by 10px), creating a nice scattered arrangement on the board!

My Prompt

set the speed of the player to be 1.3x faster but each time it picks up 1 ducky the speed slows by a factor of 0.25 (so the more they carry the increasingly slower the speed of the player gets when keyboard movement)

GenAI Response

Done! I've implemented the speed system where the player gets progressively slower as they carry more ducks. Here's what changed:

[world.json](#)

- Increased base speed from 3 to 3.9 (1.3x faster)

[Player.js](#)

- Added `baseSpeed` property to store the initial speed
- Added `carrieDucksCount` to track how many ducks are being carried
- Added `setCarriedDucks(count)` method to update the duck count
- Added `getActualSpeed()` method that calculates: `baseSpeed * (1 - 0.25 * numDucks)`
- Updated `updateInput()` to use the calculated actual speed

[sketch.js](#)

- Added `player.setCarriedDucks()` call before `updateInput()` to update the player with the current number of carried ducks

Now the speed decreases as follows:

- **0 ducks:** 3.9 (full speed)
- **1 duck:** 2.925 (75% speed)
- **2 ducks:** 1.95 (50% speed)
- **3 ducks:** 0.975 (25% speed)
- **4+ ducks:** 0 (can't move)

My Prompt

add 3 scattered lifebuoy.pngs at size 164px by 164px to scattered randomly each time the page is reloaded to the canvas background. when the playrr goes over its position the buoy should disappear permanently and the players speed should increase by 0.25x for next 15 seconds (regardless if there is a ducky or not being carried).

GenAI Response

Done! I've added the lifebuoy system to your game. Here's what I implemented:

sketch.js

- Added lifebuoy image preload ([lifebuoy.png](#))
- Passed lifebuoy image to WorldLevel
- Added lifebuoy collision detection in the draw loop that activates the speed boost

WorldLevel.js

- Updated constructor to accept lifebuoy image and initialize 3 lifebuoys at random positions
- Added [checkLifebuoyCollision\(player\)](#) method that detects player collision with active buoys
 - Collision radius is 82px (half of 164px size)
 - Marks buoy as inactive when touched
- Added lifebuoy drawing in [drawWorld\(\)](#) - only active buoys are displayed at 164×164px

Player.js

- Added [speedBoostEndTime](#) property to track when boost expires
- Added [activateSpeedBoost\(\)](#) method that activates a 15-second (15000ms) boost
- Added [isSpeedBoosted\(\)](#) method to check if boost is currently active
- Updated [getActualSpeed\(\)](#) to add 0.25x speed when boost is active

- The boost works regardless of how many ducks are being carried

Now when the player touches a lifebuoy:

1. The buoy disappears permanently
2. The player gets a +0.25x speed boost for 15 seconds
3. Multiple boosts can stack if you collect buoys while one is active
4. The boost applies on top of the base speed and duck penalties

Rest of Conversation

Done! I've updated the speed boost system to multiply speed by 1.4x for each active boost. Here's what changed:

Player.js

- Changed from a single speedBoostEndTime to a `speedBoosts` array that tracks multiple boost end times
- `activateSpeedBoost()` now adds a new boost to the array (boosts stack)
- Added `getActiveBoostCount()` that removes expired boosts and returns the count
- Updated `getActualSpeed()` to multiply by $1.4^{(\text{active boost count})}$
 - 1 boost: 1.4x speed multiplier
 - 2 boosts: 1.96x speed multiplier
 - 3 boosts: 2.744x speed multiplier
 - etc.

Now when the player collects multiple lifebuoys, the speed boosts will stack multiplicatively, each one multiplying the current speed by 1.4x!

Done! I've updated the spawn logic to meet your requirements:

WorldLevel.js

For ducks:

- The first duck is guaranteed to spawn in the initial camera view (X: 0-800, Y: 560-1040) but never on the diving board
- The remaining 6 ducks can spawn anywhere in the world, but never on the diving board
- Uses a retry loop (up to 100 attempts) to ensure valid placement

For buoys:

- All 3 buoys are guaranteed to spawn outside the initial camera view
- They will only appear as the player explores beyond the starting camera area
- Uses a retry loop (up to 100 attempts) to find valid positions

Now when the game starts:

- The player will always see at least 1 duck in their starting view
- No buoys will be visible initially - they only appear as you explore
- No buoys or ducks will spawn on the diving board

when all duckies are collected / stationary on the diving board (all 7) aka when the 0/7 becomes a 7/7 make the game stop (player can no longer be moved) and a popup rectangle shows in middle of canvas that says in sans-serif "task complete" and rectangle button below it in sans-serif that reads "play again" that when clicked reloads the page

I see the issue - it looks like the gameComplete logic and popup rendering didn't get properly saved. Let me rebuild the complete [WorldLevel.js](#) with all the necessary functionality:

Now I'll add the missing game completion logic. Let me add the gameComplete flag to the constructor:

Now I'll add the checkGameComplete() method:

Now I'll update [drawHUD\(\)](#) to render the completion popup:

I'll check the current state of both files since your previous edits were undone, then implement this feature.

Now let me check sketch.js:

Now I'll implement the win condition. I need to:

1. Add a `gameComplete` flag and methods to `WorldLevel`
2. Modify the HUD to show a popup when the game is complete
3. Block player movement when complete
4. Add button click handling

Now I'll update [`sketch.js`](#) to check for game completion, block input when complete, and handle button clicks:

Now I'll add the `mousePressed` function to handle button clicks:

Done! I've implemented the win condition with a popup UI. Here's what was added:

[WorldLevel.js changes:](#)

- Added `gameComplete` flag to track when all ducks are on the board
- Added `checkGameComplete()` method to detect when all 7 ducks are on the diving board
- Added `isPlayAgainButtonClicked()` method to detect button clicks
- Enhanced `drawHUD()` to display a popup overlay with:
 - Semi-transparent dark overlay covering the entire canvas
 - White box in the center with "Task Complete" title (sans-serif, larger font)
 - Blue "Play Again" button below the title

[sketch.js changes:](#)

- Added guard to block player input when `gameComplete` is true
- Added `level.checkGameComplete()` call in the draw loop to continuously check if all ducks are collected
- Added `mousePressed()` function that detects button clicks and reloads the page

When all 7 ducks are brought to the diving board, the player will freeze, the popup will appear, and clicking "Play Again" will reload the page to start over.

I'll check the current state of the files and then make the camera follow slower as more ducks are carried.

I'll modify the camera lerp to decrease as more ducks are carried. I'll make it multiplicatively reduce the camera responsiveness:

Done! The camera now lags more as more ducks are carried. Each duck reduces camera responsiveness by 10%, so:

- **0 ducks:** Lerp = 0.12 (normal speed)
- **1 duck:** Lerp = 0.108
- **2 ducks:** Lerp = 0.096
- **3 ducks:** Lerp = 0.084
- ... up to **7 ducks:** Lerp \approx 0.024 (very sluggish)

This creates a nice visual effect where the camera struggles more to keep up as the player gets weighed down with ducks. I've capped the minimum lerp at 0.02 to ensure the camera still follows, just very slowly.

I'll check the current duck spawning logic and add the distance constraints:

I'll add a helper method to check if a position is too close to the diving board or player start, then update both duck spawning sections:

Done! I've added the distance constraints for duck spawning:

Changes made:

1. Added a new [isTooCloseToRestrictedArea\(\)](#) helper method that checks if a position is within 40px of:
 - The diving board center
 - The player's starting position (300, 300)
2. Updated the first duck spawn logic to enforce this constraint while keeping it in the initial camera view (0-800 x, 560-1040 y)
3. Updated the remaining 6 ducks spawn logic to use a retry loop that respects the 40px buffer zones

Now ducks will spawn with proper spacing away from the diving board and player start position, while the first duck still spawns within the initial camera view. This prevents ducks from spawning too close to where the action begins.

Now I'll add a helper method to check if a duck is within the diving board area plus 40px buffer, and apply it to both duck spawning loops:

I'll add the helper method first:

Now I'll update the first duck spawn to use the new helper method:

Now I'll update the remaining ducks spawn to use a retry loop:

Done! I've updated the duck spawning logic to prevent ducks from spawning within the diving board area **plus a 40px buffer on all sides**:

Changes made:

1. Added [`isDuckTooCloseToDivingBoard\(x, y\)`](#) helper method that checks if a position is within the diving board boundaries (x: 0 to 600, y: h/2-12.5 to h/2+107.5) PLUS 40px buffer in all directions
2. Updated the **first duck spawn** to use this check - it retries if it spawns too close to the board while staying in the initial camera view (0-800 x, 560-1040 y)
3. Updated the **remaining 6 ducks spawn** to use a retry loop that skips any spawn position within the 40px buffer zone and re-rolls for a new position

Now all 7 ducks will spawn far enough away from the diving board to give the player space to approach it without ducks overlapping the board area.