Preproduction

Aesthetic Goals:

1. **Goal:** Create a sense of anticipation and excitement in players as they make shots in 8-ball pool.
    a. **Signs of Success:** Players experience a heightened sense of excitement when aiming and taking shots, feeling engaged and eager to see the results. They should find the gameplay thrilling and unpredictable, which keeps them coming back for more.
    b. **Signs of Failure:** If players become disinterested or bored with the game, or if the gameplay becomes too predictable, it suggests a failure to achieve this aesthetic goal.
2. Goal: Foster a sense of competition and strategy in 8-ball pool players.
    a. **Signs of Success**: Players should feel challenged by the game and constantly engaged in strategizing their shots, considering various angles, potential moves, and defensive plays. They should enjoy the competitive aspect of the game and have a desire to improve their skills to outsmart their opponents.
    b. **Signs of Failure**: If players find the game too easy or not competitive enough, or if they lose interest in the strategic aspects, it indicates a failure to meet this aesthetic goal.

Core Loop:

The core mechanics of 8-ball pool involve players taking turns to aim and shoot their cue ball to pot their assigned balls (stripes or solids) and finally the 8-ball to win the game. The core loop can be described as follows:

1. Player's Turn:

a. Player takes aim and adjusts the angle and power for their shot.

b. The player shoots the cue ball to hit their assigned balls or target a strategic move.

c. The cue ball interacts with other balls, following the laws of physics.

d. The player's turn ends when they fail to pot a ball or commit a foul.

2. Opponent's Turn:

a. If the player successfully pots a ball, they continue their turn.

b. If the player fails to pot a ball or commits a foul, the opponent takes their turn.

3. Winning the Game:

a. The player aims to pot all their assigned balls (stripes or solids) before their opponent.

b. The player sets up a strategy to pot the 8-ball after clearing their assigned balls without fouling.

4. Game Outcome:

a. The player who successfully pots the 8-ball without fouling wins the game.

b. If a player fouls while potting the 8-ball, they lose the game.

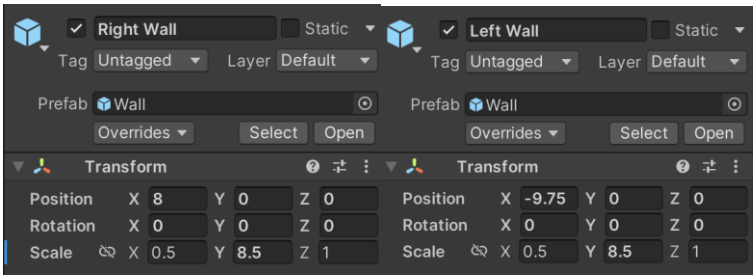
The core loop serves the aesthetic goals as follows:

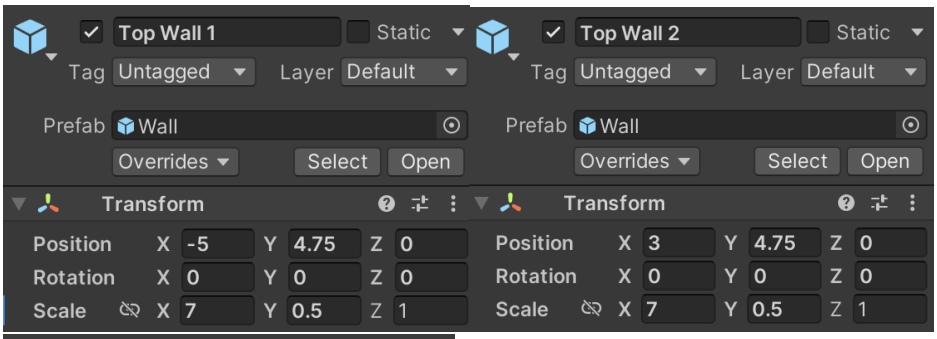
- Aesthetic Goal 1: The core loop incorporates the excitement of making shots, as players carefully aim their cue ball to pot balls and anticipate the outcomes of each shot.
- Aesthetic Goal 2: The core loop emphasizes competition and strategy as players must carefully plan their shots, considering the position of the balls, their assigned balls, and the 8-ball, creating a strategic and competitive experience.

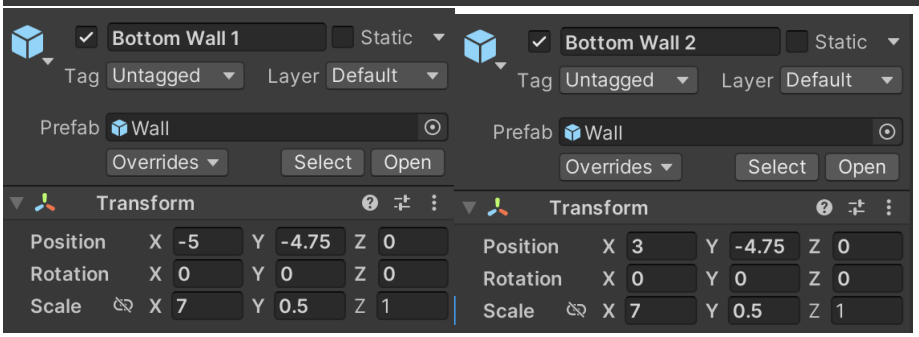
Developer Notes

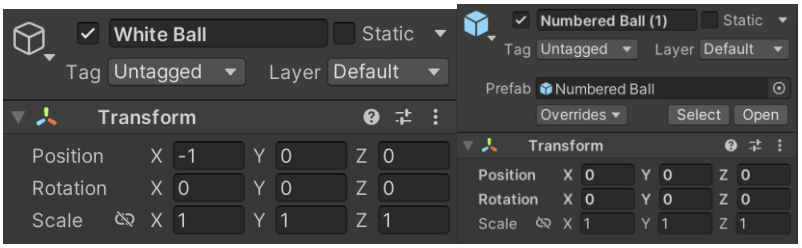
11 / 05: Created Basic Prefabs (Sprites)

- Assigned the sprites their starting positions









11 / 06: Created aiming mechanism for white ball

White Ball

Initial Script Setup

- Created the WhiteBall script to manage the behavior of the white ball in the game.

- Defined variables for Rigidbody, SpriteRenderer, pastVelocity, hitPowerScale, Aim, aimTransform, and more

Aiming Mechanism

- Introduced basic friction to the ball's movement to simulate a natural slowdown.

- Implemented the core gameplay loop in the play() function to handle ball movement and hitting.

Hit Functionality

- Added functionality to initiate a hit when the spacebar is pressed.

- Calculated the hit velocity based on the aiming direction and player-controlled hit power.

Collision Handling

- Implemented OnCollisionEnter2D to handle collisions with pockets and walls.

- Added logic to destroy the white ball and the aiming diamond when the ball is pocketed.

Aim

Initial Script Setup

- Created the Aim script responsible for managing the aiming diamond in the game.

- Defined variables for Transform, whiteBall, whiteBallTransform, whiteBallrb, rotationSpeed, angle, radius, myRenderer, and numberedBalls.

Initialization and Starting Position

- Initialized the aimTransform to the current GameObject's transform.

- Set the starting position of the aiming diamond to (-5, 0, 0).

- Located and assigned the whiteBall GameObject using FindObjectOfType.

- Accessed the transform and rigidbody of the whiteBall.

Aiming Mechanism Implementation

- Implemented the Revolve function to make the aiming diamond revolve around the whiteBall.

- Calculated the circular path position based on user input and rotation speed.

- Adjusted the orientation of the aiming diamond to look at the whiteBall.

Checking if Balls are in Play

- Added a method checkIfInPlay to determine whether the whiteBall or any numbered ball is still in motion.

- Used this information to control the visibility of the aiming diamond.

- The aiming diamond becomes invisible when balls are still in play.

User Input Handling

- Incorporated user input handling to control the rotation of the aiming diamond.

- Utilized Input.GetKey to detect arrow key presses for rotationInput.

- Adjusted the Revolve function to respond to arrow key inputs.

11 / 07: Implemented the physics for bouncing off walls

Pocket Collision

- The method checks if the collided GameObject has the "Pocket" tag.

- If true, it triggers the `DestroyWhiteBall` function, marking the white ball as pocketed (`pocketed = true`).

- The white ball is effectively removed from the game when it enters a pocket.

Wall Collision

- The method checks if the collided GameObject has the "Wall" tag.

- Upon a collision with the wall, it simulates bouncing by reversing the velocity of the white ball.

- Utilizes `Vector3.Reflect` to calculate the reflection vector based on the wall's normal.

- The white ball's velocity is adjusted by multiplying the reflection vector with a factor (0.7f), causing the ball to slow down.

- The pastVelocity variable is updated to store the current velocity of the white ball.

11 / 08: Added more numbered balls and implemented score

Score Display Update

- The `Update` method is called once per frame and ensures that the score display text is updated to reflect the current value of `current_score`.

- The score is displayed in the format "Score: [current_score]".

- The score increases by 10 every time a numbered ball is pocketed and decreases by one every time the white ball is hit.

Added more balls

- Duplicated the numbered ball prefabs onto the scene

11 / 08: Adding Game Over Screen and Power Bar

Game Over UI

- Create a UI canvas for the game over screen with Game Over text

- Implement a script to manage the game over screen, including displaying the final score

Power Bar

- Created a UI power bar slider that determines the velocity of the ball

11 / 09: Added sound and finishing touches

Game Over UI

- Create a UI canvas for the game over screen with Game Over text

- Implement a script to manage the game over screen, including displaying the final score

Added Sound

- Added sound using AudioSource and clips for each sound

- Source is an AudioSource for game over sound effects and clips is a single AudioClip for game over events