DevLog

Preliminary Game Design

- **Inspiration**: This game was inspired by a childhood game called Red Ball. The game was a simple 2D platformer where the player controlled a red ball that could navigate obstacles across 10 levels to reach the goal.
- **Game Concept**: A 2D platformer where the player controls a red ball that can jump to reach the goal.
- **Game Mechanics**:
    - The player can move the ball left and right.
    - The player can jump the ball.
    - The camera pans with the movement of the ball within the scene.
    - The player can restart the level by pressing the R key.
- **Level Design**:
    - The initial plan is to build a single level.
    - The level will include moving platforms that the ball can use to reach the objective (a flag post).
    - There will be obstacles such as spikes.
    - Spikes can result in a game over condition where the big red ball breaks into a couple of small red balls, and the game over screen flashes up.

Aesthetic Goals

- **Simplicity**: The game should have a simple and clean design. The red ball has a black line to help indicate the direction of movement. The ground will have a simple grass sprite in places to add aesthetic appeal.
- **Success Indicators**: Players progress the ball towards the clearly marked flag post. When they reach the objective a sucess screen flashes up indicating they completed the level and prompting them to play again.

Core Loop

- **Core Mechanics**: The player navigates the level, avoiding falling and obstacles, searching for the flag post to reach the end of the level.
- **Aesthetic Goals**: With the camera following the player, the player can focus on the core mechanics of moving the ball through the level without seeing the entire level at once, providing an aspect of discovery as they progress through the level.

DevLog Entry - Nov 16, 2024

- **Morning**:
    - Created the red ball prefab using the Unity editor.
    - Created ground prefabs of three different sizes so I can build the level using the Unity editor. Included a grass sprite to add aesthetic appeal.
    - Implemented collisions between the ball and the ground using an EdgeCollider2D component.
    - Implemented the camera follow script to follow the ball. This adds to the aesthetics of the game by keeping the ball in the center of the screen and allowing the player to navigate a

larger level.
    - Implemented the ball movement script including jumping and rolling left and right.
- **Afternoon**:
    - Added a vertically moving platform object.
        - Experienced difficulties with interactions between the ball and the moving platform. When the moving platform moved downwards, the ball would lag behind. This issue was resolved by changing the way that the platform's position was updated - using it's rigidbody2D component instead of transform.
    - Added a horizontally moving platform object - changing the movement script to move the platform between specified positions (transforms).
    - Added a spike object that results in the game resetting when the ball collides with it.
        - I made the decision to reset the game when the ball collides with the spike object rather than breaking the ball into smaller balls. This was a design decision to keep the game simple.
- **Evening**:
    - Added a flag post object that the ball can reach to complete the level.
    - Added a success screen that flashes up when the ball reaches the flag post and prompts the player to restart the game.
    - Implemented the GameManager script to allow the player to complete the level upon reaching the flag (the objective) and restart the game by pressing the R key.
    - The game now resets when the player falls off the level.
    - Added a moveable crate object that the ball can push around to help overcome certain obstacles.
    - Added sound effects to indicate failure (colliding with spikes or falling) and success (reaching the flag post). Initially I did not consider sound effects in the preliminary design, but they definitely add to the game experience by providing feedback to the player upon mistakes or eventual completion of the level.

Postmortem

Summary of Initial Goals

Originally, I set out to create a simple 2D platformer inspired by a childhood Flash game that I loved called Red Ball.

Final Goals

By the end, my goals evolved to include more obstacles and more complex mechanics than I originally envisioned. For example, my moving platforms did not just move up and down, but I actually was able to create them to move between specified positions, making them more dynamic. I also added a crate object that the player could push around to help overcome obstacles.

Accomplishments

I successfully implemented the core mechanic of the game, added moving platforms, obstacles, a flag post, and sound effects. The game includes a complete and challenging level with a success screen and restart functionality once the player reaches the end. I'm proud of the experience my final game offers.

What Went Right

- The core mechanics of jumping worked well from the beginning.
- I was quickly able to create the elements of my scene using the Unity editor directly as well as a few simple free sprites that I downloaded (see CREDITS.txt).
- The camera follow script was easy to implement and added to the aesthetics of the game.
- Implementing moving platforms was relatively straight forward however I did encounter bugs with the mechanics of the red ball which I explain below.

What Went Wrong

- I had significant bugs in the player's movement resulting from the platform interactions:
    - In part this was due to the scales of the objects being different meaning that the red ball would warp when its transfrom was assigned as a child of the moving platform. Resolving this issue required rescaling the sprites.
    - I seperately observed that the ball was lagging behind the platform when it moved downwards. This was resolved by changing the way that the platform's position was updated - using it's rigidbody2D component instead of transform.
- It was difficult to attach the sound effects to the intended events. Resolving this required refactoring the way I handled win and game over conditions before resetting the game.

Lessons Learned

- Understanding the importance of consistent object scaling in Unity is crucial to avoid unexpected behavior.
- Using Rigidbody2D for platform movement ensures smoother interactions with other physics objects.
- Mechanics should be designed with sound effects in mind from the beginning to avoid refactoring later on.