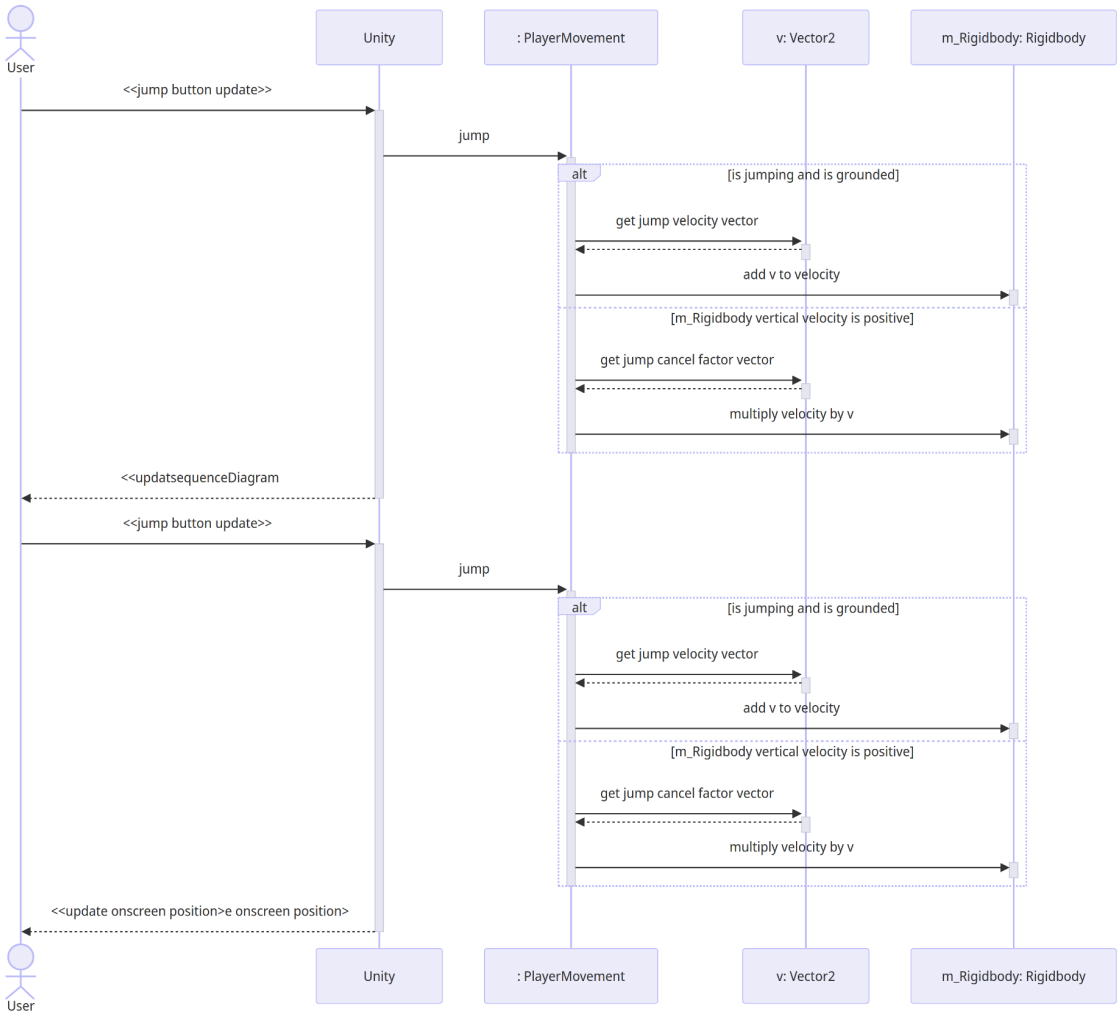
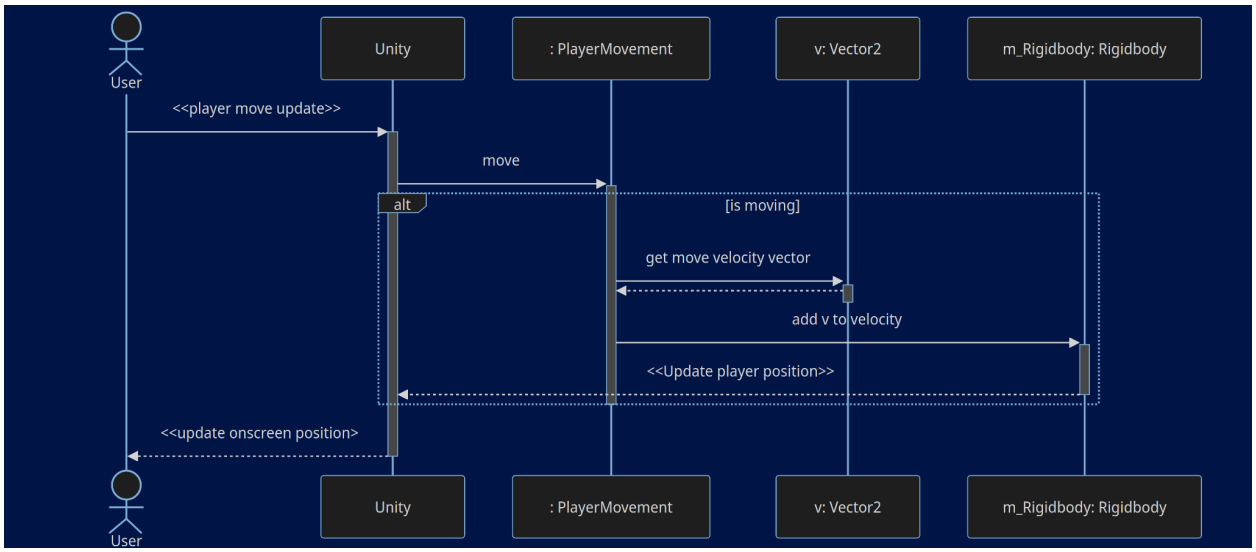
We need 7 Analysis Sequence diagrams and 3 Design Sequence diagrams and their descriptions.
Analysis sequence diagrams: 1:



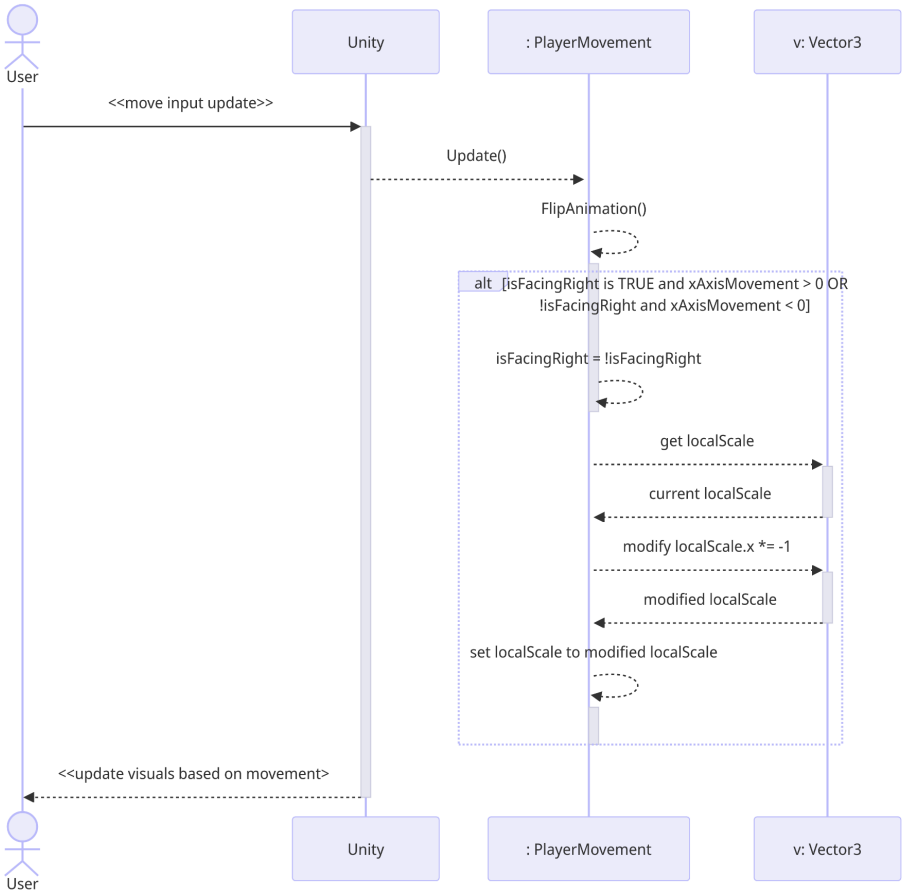
Description:
First the user presses the jump button, and then Unity checks if the jump button has been pressed. Then unity runs the jump function that acts on the Player movement script. Inside the player movement script, after grabbing the jump velocity vector, you add the velocity value returned from the rigidBody component. After that you get the jump cancel factor vector and multiply it by the velocity.

2:



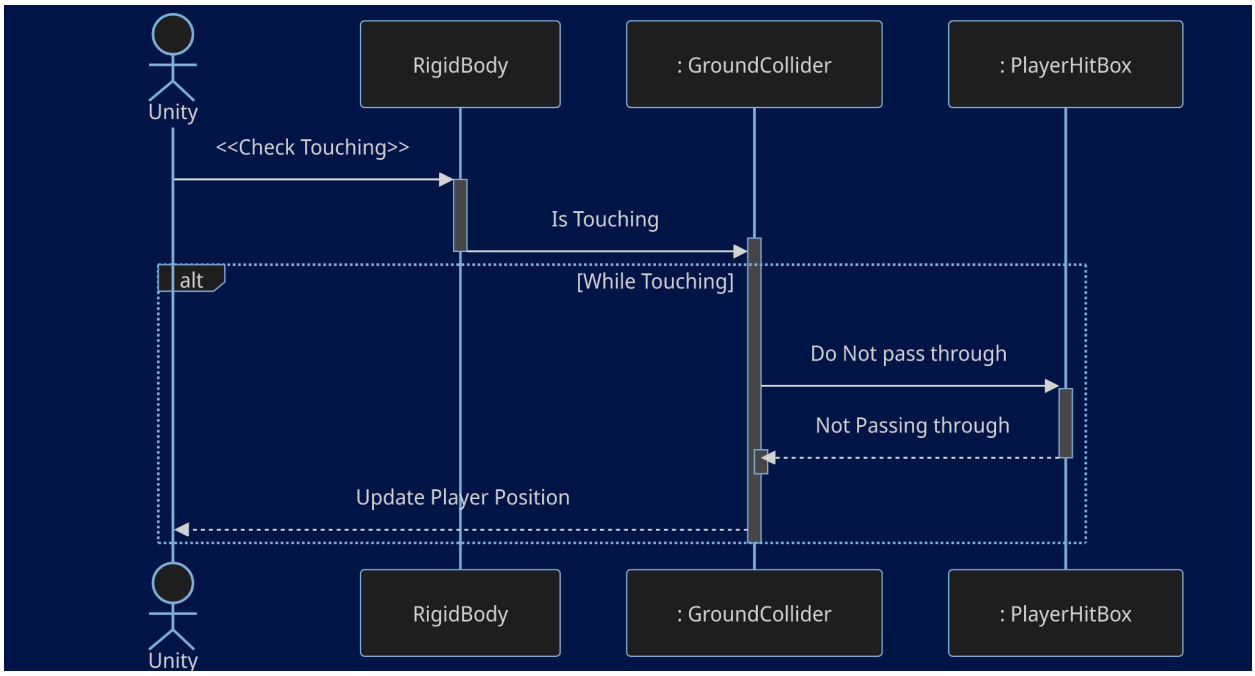
Description: When the player presses the move button the move action is called. While the player is moving, we're getting the vector that stores the velocity. While we're getting the velocity, we're adding it to the velocity of the rigidbody, and updating the player movement each time.

3:



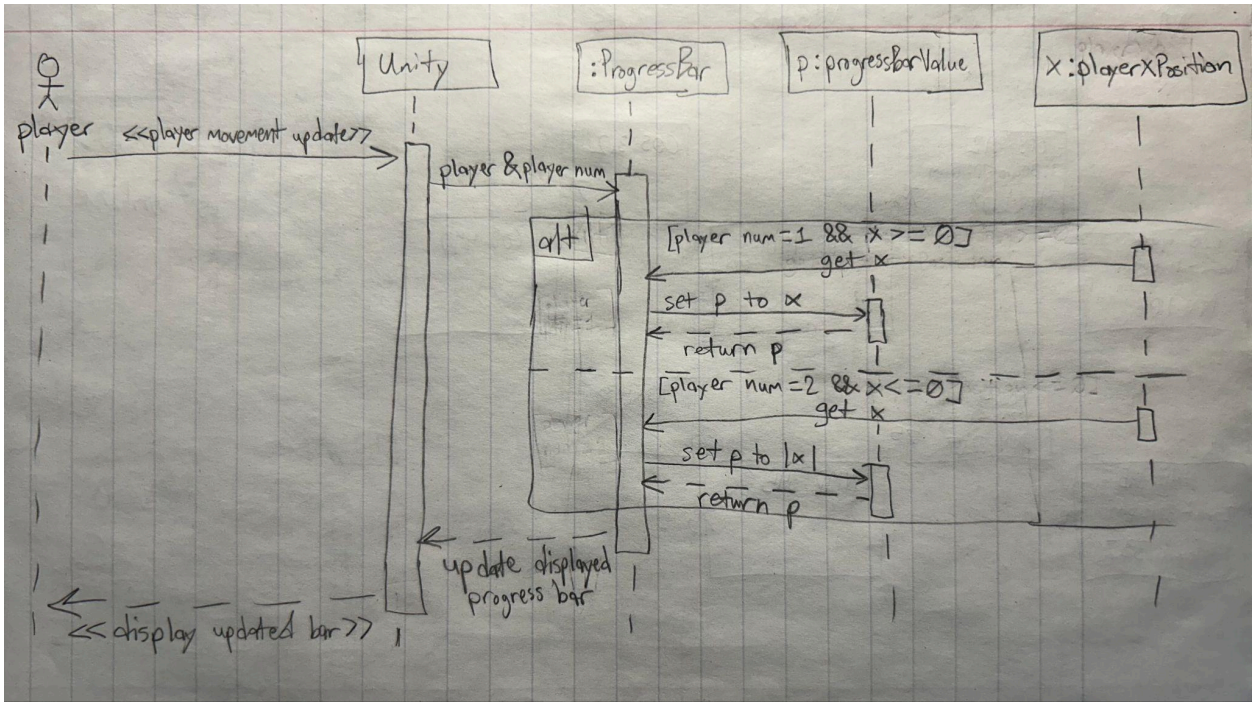
Description: The user first provides movement input (left or right). Unity then calls the Update function within the PlayerMovement script. Inside Update, the script checks if the user's movement direction contradicts the character's current facing direction. If there's a mismatch (e.g., moving right while facing left), the FlipAnimation function is triggered. FlipAnimation flips the character's sprite to match the movement direction. It does this by grabbing the character's current scale (localScale) and mirroring it across the x-axis (multiplying localScale.x by -1).

4:



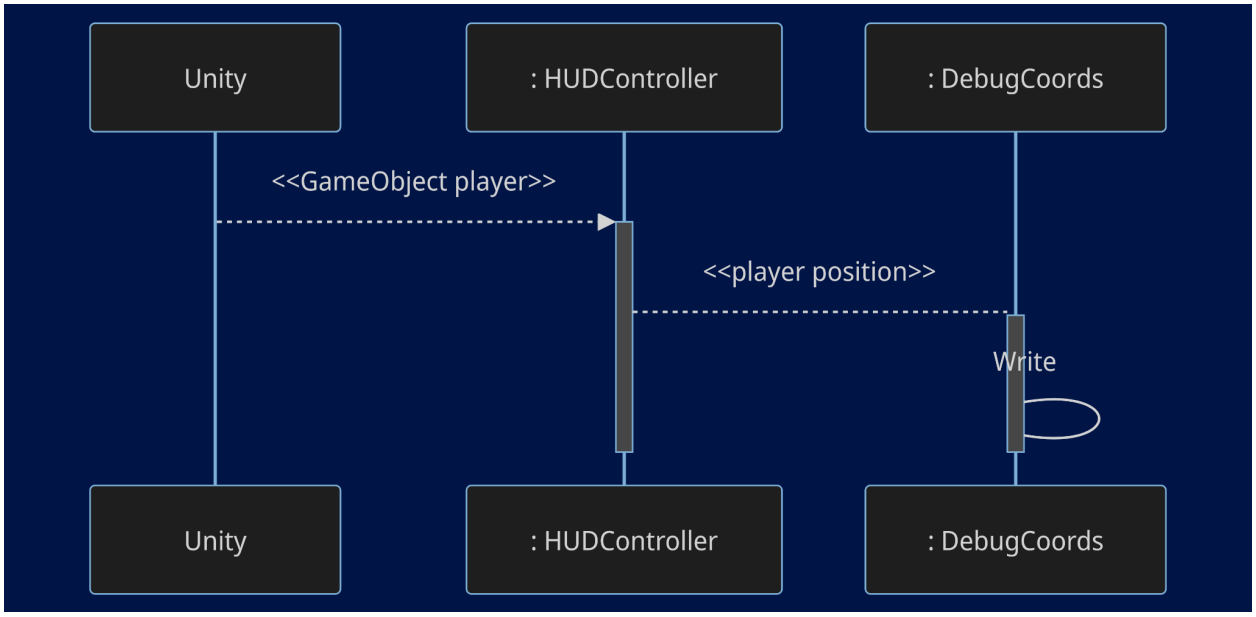
Description:
First Unity checks to see if the RigidBody is touching the ground collider, and if the ground collider is touching the rigidbody the player hitbox does not allow the rigidyBody to pass through the ground collider while they are still touching, and updating the player position as currently being on top of the ground collider.

5:



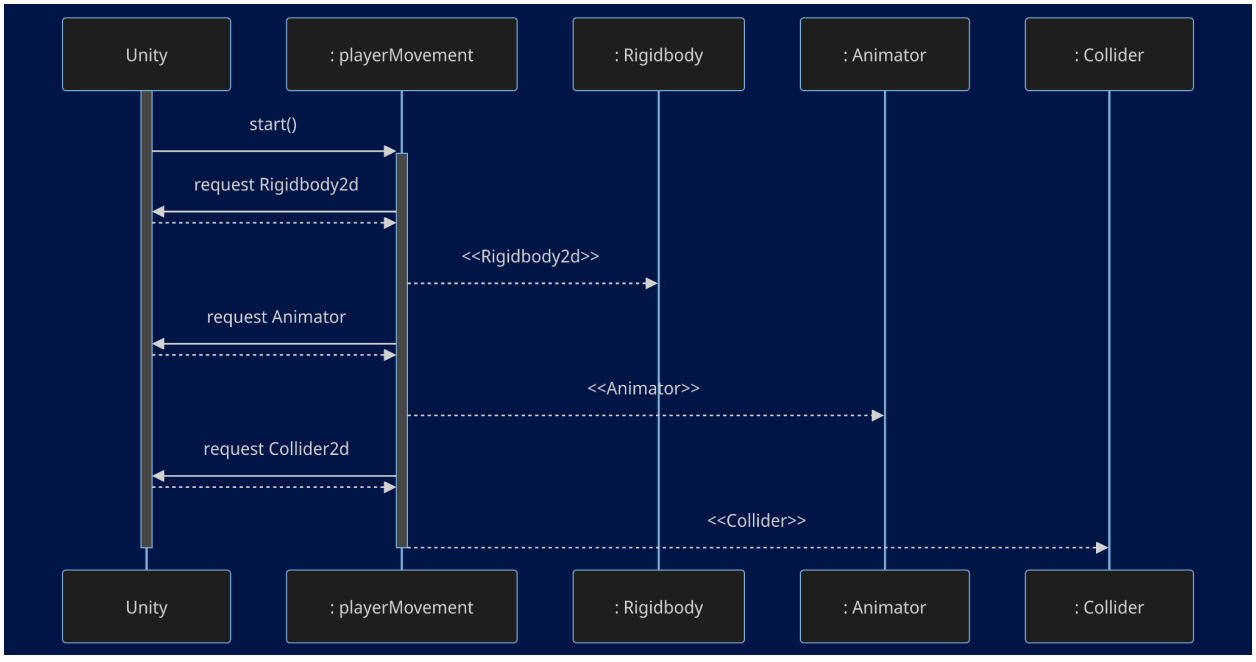
Description:
After a player moves, Unity sends the corresponding player game object and player number are sent to the ProgressBar Object. If the player num is 1 and the player object's x position is greater than or equal to 0, then the player's x position is given to the ProgressBar, the given value is set equal to the progressBarValue, and the value is returned to the ProgressBar. Else, if the player num is 2 and the player object's x position is less than or equal to 0, then the same process will execute, except it will set the absolute value of the player's x position to the progressBarValue. Finally, the ProgressBar sends back to Unity its updated data, and Unity displays the new progress bar.

6:



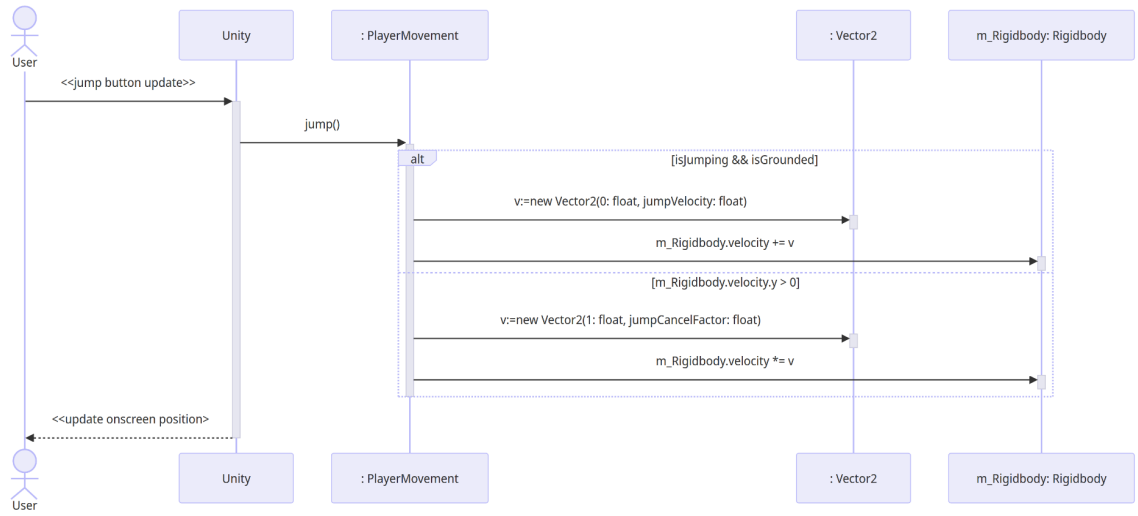
Description:
Unity sends the GameObject player to the HudController script. The HudController script then sends the position of the player to the text object debugCoords, which then saves them.

7:



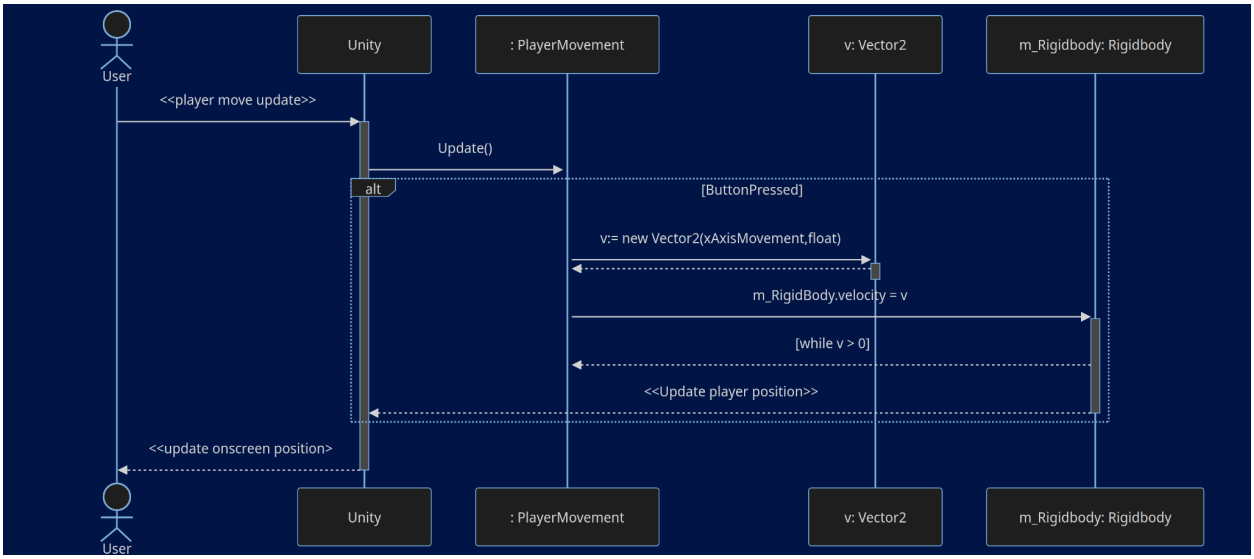
Description:
Unity runs the start method as the first thing it does. In the playerMovement script, it requests Rigidbody2D, Animator, and Collider2d from unity and stores them in local variables Rigidbody, Animator, and Collider, to initialize them.

Design Sequence Diagrams:
1:



Description:
First the user presses the jump button, and then Unity checks if the jump button has been pressed. Then unity runs the jump function that acts on the Player movement script. Inside the player movement script, after grabbing the jump velocity vector, you add the velocity value returned from the rigidBody component. After that you get the jump cancel factor vector and multiply it by the velocity.

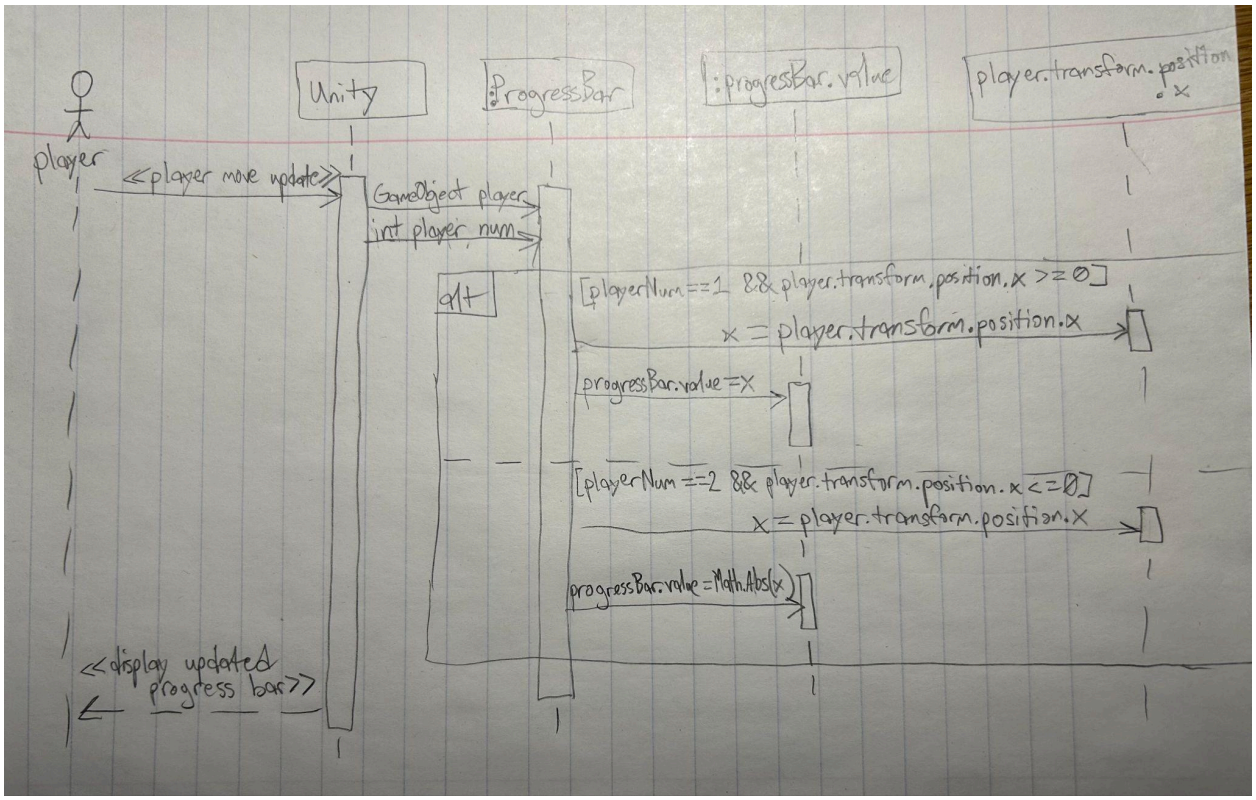
2:



Description:

When the movement button is pressed, we're looking at the Update function in the PlayerMovement script from Unity. While the button is being pressed, each time the update function happens, we're creating a new vector that has the velocity of the player, and then we're adding that velocity to the velocity component of the rigidBody component. This is only happening while the velocity is greater than 0, because if it's 0 that means the player shouldn't be moving because the button isn't being pressed. We're also updating the player position in the update method.

3:



Description:
After a player moves, Unity sends the corresponding player game object and player number are sent to the ProgressBar Object. If the player num is 1 and the player object's x position is greater than or equal to 0, then the player's x position is given to the ProgressBar, the given value is set equal to the progressBarValue, and the value is returned to the ProgressBar. Else, if the player num is 2 and the player object's x position is less than or equal to 0, then the same process will execute, except it will set the absolute value of the player's x position to the progressBarValue. Finally, the ProgressBar sends back to Unity its updated data, and Unity displays the new progress bar.