HGD Progress Report

Name: Braeden Moore
Team #: 4
Game name: Golf Quest
Current semester: Fall 2022
Expected game completion: Spring 2023

Sprint 2

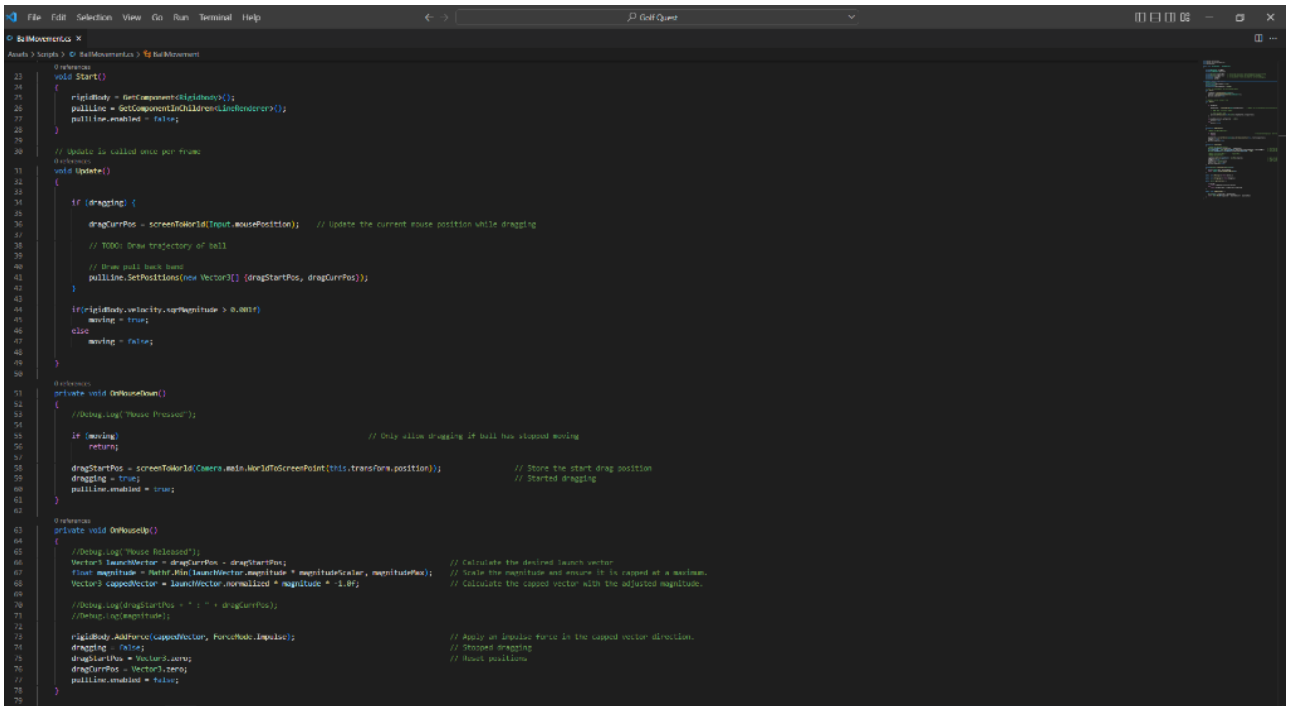
Completed & In-progress tasks

(Finished: 2 hours) Task 1: Ball Movement & Pull Back Line

Commit link:
https://github.com/HuskyGameDev/2022f-team4/commit/5f13eb9c9427e993667843126c29c3b777b97d48

Story: I created the ability to click and drag your mouse on the player ball to charge back and release a shot to move the ball. Direction and magnitude of the launch vector was determined by the pull back of the mouse. I also created a small white indicator that draws a line from the ball to the mouse when you are pulling back to give feedback when it is being done.

Screenshots/videos/GIFs:



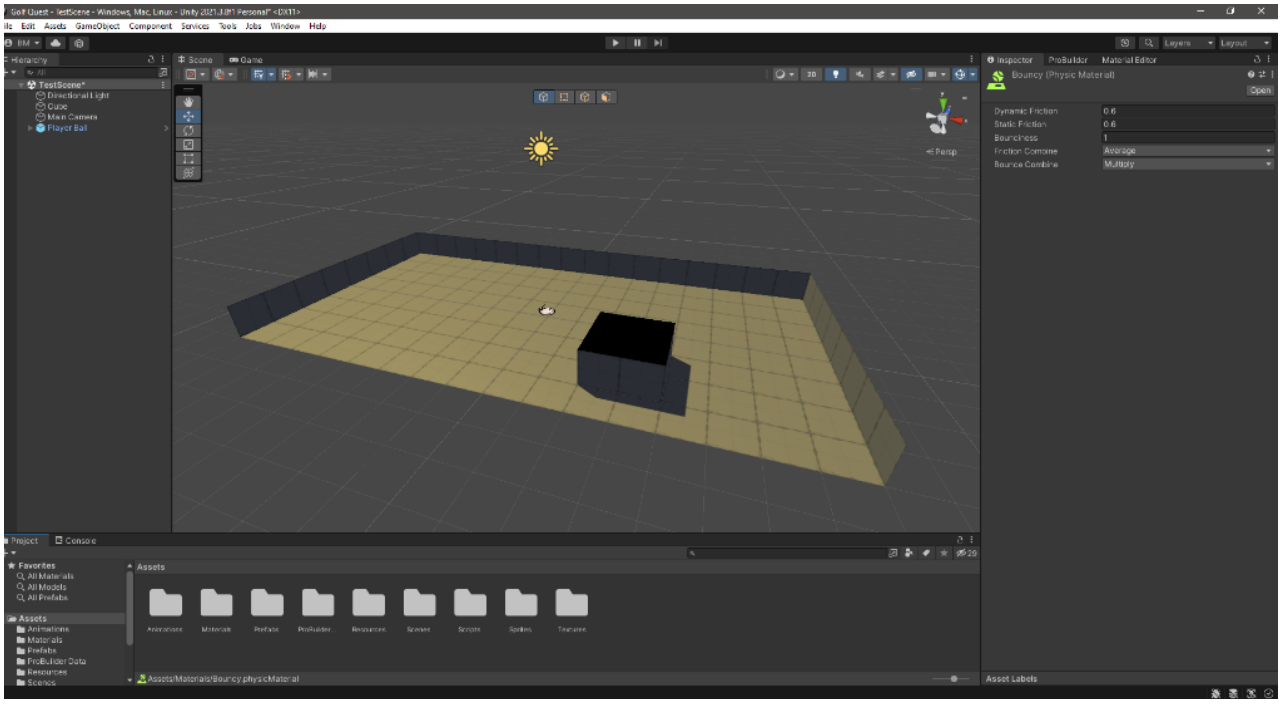
```
void Start()
{
    rigidBody = GetComponent<Rigidbody>();
    pullLine = GetComponentInChildren<LineRenderer>();
    pullLine.enabled = false;
}

// Update is called once per frame
void Update()
{

    if (dragging) {

        dragCurrPos = screenToWorld(Input.mousePosition);    // Update the current mouse position while dragging

        // TODO: Draw trajectory of ball

        // Draw pull back band
        pullLine.SetPositions(new Vector3[] {dragStartPos, dragCurrPos});
    }

    if(rigidBody.velocity.sqrMagnitude > 0.001f)
        moving = true;
    else
        moving = false;

}

private void OnMouseDown()
{
    //Debug.Log("Mouse Pressed");

    if (moving)                                         // Only allow dragging if ball has stopped moving
        return;

    dragStartPos = screenToWorld(Camera.main.WorldToScreenPoint(this.transform.position));              // Store the start drag position
    dragging = true;                                                                                    // Started dragging
    pullLine.enabled = true;
}

private void OnMouseUp()
{
    //Debug.Log("Mouse Released");
    Vector3 launchVector = dragCurrPos - dragStartPos;                  // Calculate the desired launch vector
    float magnitude = Mathf.Min(launchVector.magnitude * magnitudeScaler, magnitudeMax);    // Scale the magnitude and ensure it is capped at a maximum.
    Vector3 cappedVector = launchVector.normalized * magnitude * -1.0f;                     // Calculate the capped vector with the adjusted magnitude.

    //Debug.Log(dragStartPos + " : " + dragCurrPos);
    //Debug.Log(magnitude);

    rigidBody.AddForce(cappedVector, ForceMode.Impulse);                // Apply an impulse force in the capped vector direction.
    dragging = false;                                                   // Stopped dragging
    dragStartPos = Vector3.zero;                                        // Reset positions
    dragCurrPos = Vector3.zero;
    pullLine.enabled = false;
}
```

(Finished: 20 minutes) Task 2: Bouncing Off Walls & Directional Lighting

Commit link:
https://github.com/HuskyGameDev/2022f-team4/commit/6903a58a1f78494727b7eb9ccd92d0a978952aaa

Story: Once launching the ball was added, the ball simply hit the wall and didn't go anywhere. Thanks to Benjamin Collicott's suggestion, I used the Physics Material and made sure that both the walls and the player had a bouncy material. I also decided to add directional lighting to the scene so that the walls could cast shadows and depth could be seen easier in the level too. I made the lighting a warm yellow color so that we could emphasize the dungeon style feel and keep it a more dingy feeling.

Screenshots/videos/GIFs:

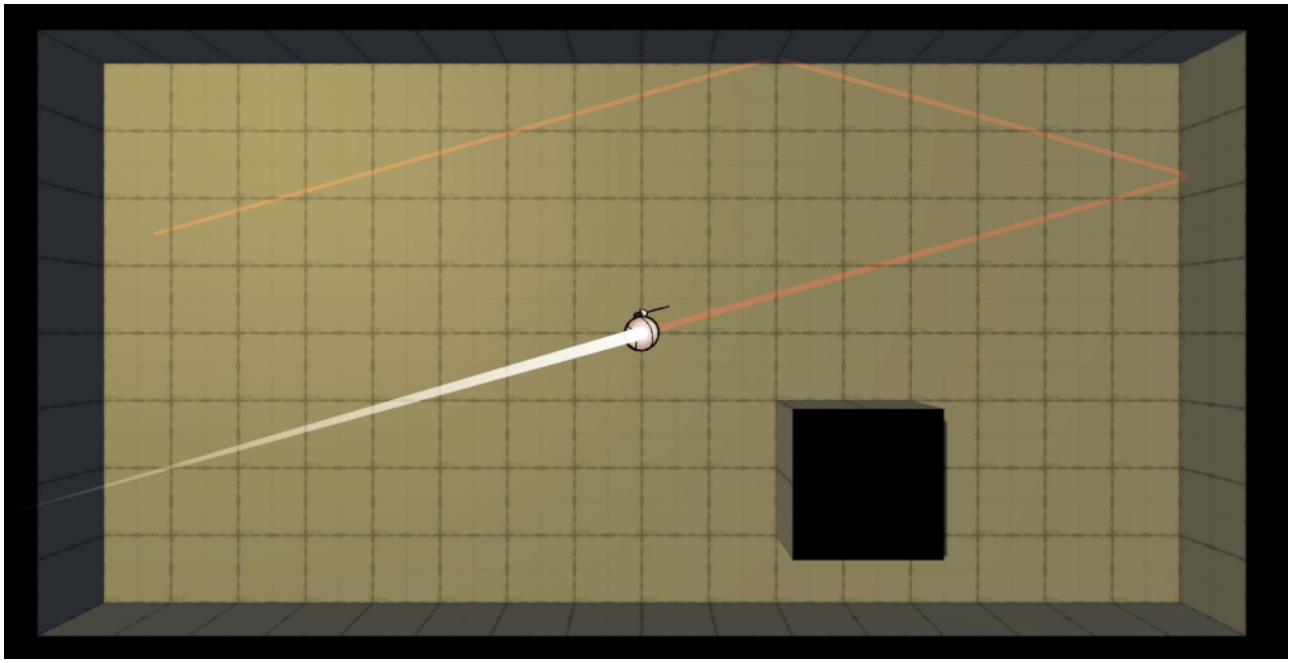


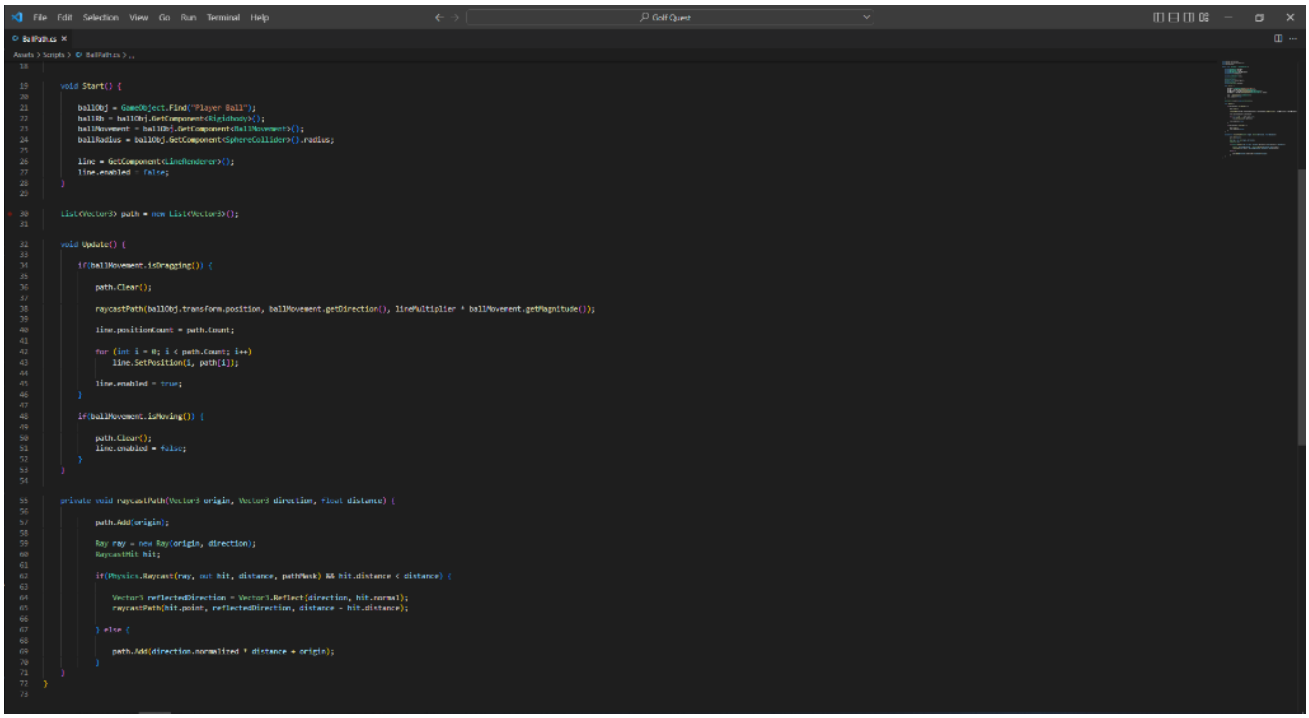
(Finished, Needs Improvement Later: 3 hours) Task 3: Ball Trajectory Line

Commit link:
https://github.com/HuskyGameDev/2022f-team4/commit/2d540388c46d035c96f76d92190e9d221cb85da0

Story: While the ball is being dragged and aimed, a trajectory line is drawn to screen that uses raycasting and reflections over normals of surfaces to predict where the ball will end up after the mouse lets go. In the future, I want to change it to use a SphereCast so that it can account for the width of the ball as it collides with walls instead of a one-dimensional ray.

Screenshots/videos/GIFs:





```
void Start() {

    ballObj = GameObject.Find("Player Ball");
    ballRb = ballObj.GetComponent<Rigidbody>();
    ballMovement = ballObj.GetComponent<BallMovement>();
    ballRadius = ballObj.GetComponent<SphereCollider>().radius;

    line = GetComponent<LineRenderer>();
    line.enabled = false;
}

List<Vector3> path = new List<Vector3>();

void Update() {

    if(ballMovement.isDragging()) {

        path.Clear();

        raycastPath(ballObj.transform.position, ballMovement.getDirection(), lineMultiplier * ballMovement.getMagnitude());

        line.positionCount = path.Count;

        for (int i = 0; i < path.Count; i++)
            line.SetPosition(i, path[i]);

        line.enabled = true;
    }

    if(ballMovement.isMoving()) {

        path.Clear();
        line.enabled = false;
    }
}

private void raycastPath(Vector3 origin, Vector3 direction, float distance) {

        path.Add(origin);

        Ray ray = new Ray(origin, direction);
        RaycastHit hit;

        if(Physics.Raycast(ray, out hit, distance, pathMask) && hit.distance < distance) {

            Vector3 reflectedDirection = Vector3.Reflect(direction, hit.normal);
            raycastPath(hit.point, reflectedDirection, distance - hit.distance);

        } else {

            path.Add(direction.normalized * distance + origin);
        }
    }
}
```

Roadblocks

The only roadblocks I ran into during the development process was finding time to work on a task for an extended period of time because I am also a Resident Assistant in DHH and that often takes a lot of time.

Goal Setting

The goals I have set for myself are to always deliver the best project I possibly can.

I want to improve the trajectory line's accuracy by using SphereCast instead of Raycast so that it can take into account the width of the ball.

Another goal of mine is to create a function in the player that enemies can use to "predict" where the player will be so that some types of enemies can use leading shots.

I also want to start working on making the hole so that once all enemies of a level are defeated the hole opens up and you can progress onward to the next level.