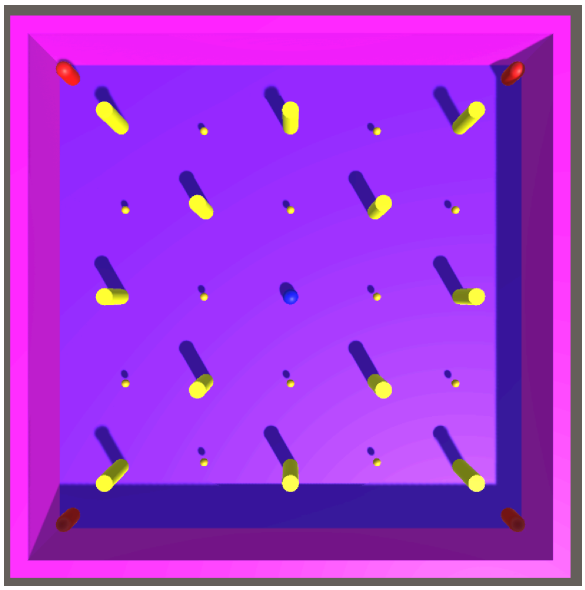
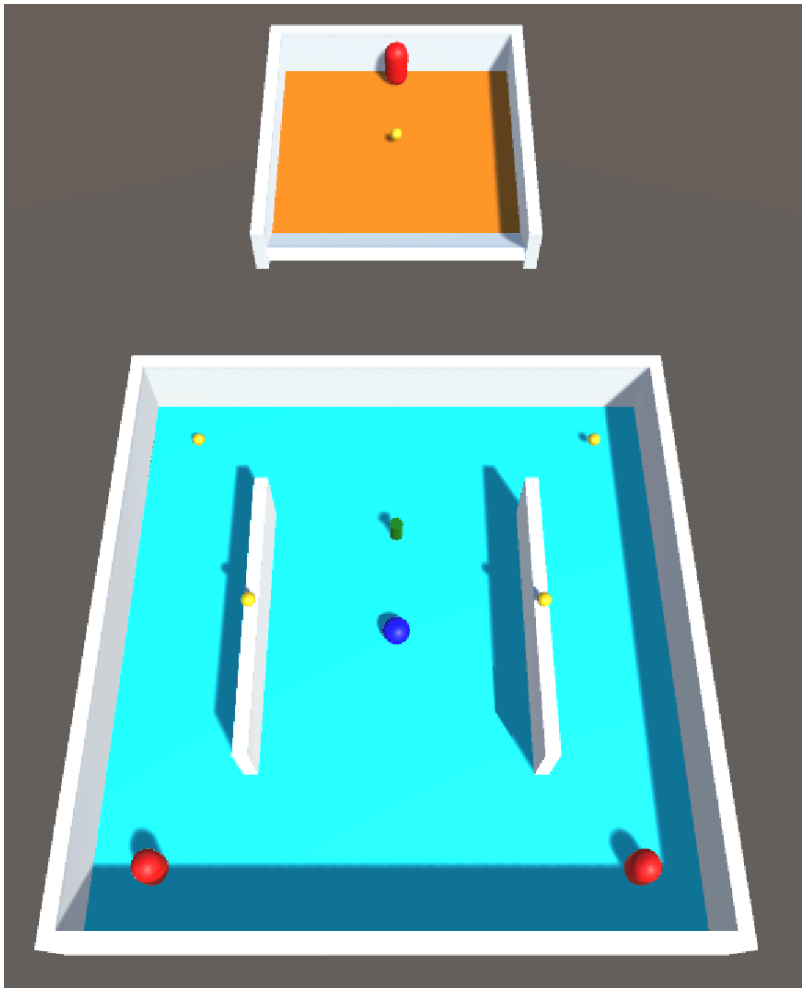
Jacob Yankee
COSC 457
Spantidi

New levels:

- The first new level is relatively simple, just a larger play area with more ghosts and an array of cylinders. This one was made for testing the NavMesh feature and originally had some ramps to further play around with the NavMesh. The color scheme here was chosen for a sort of neon look.

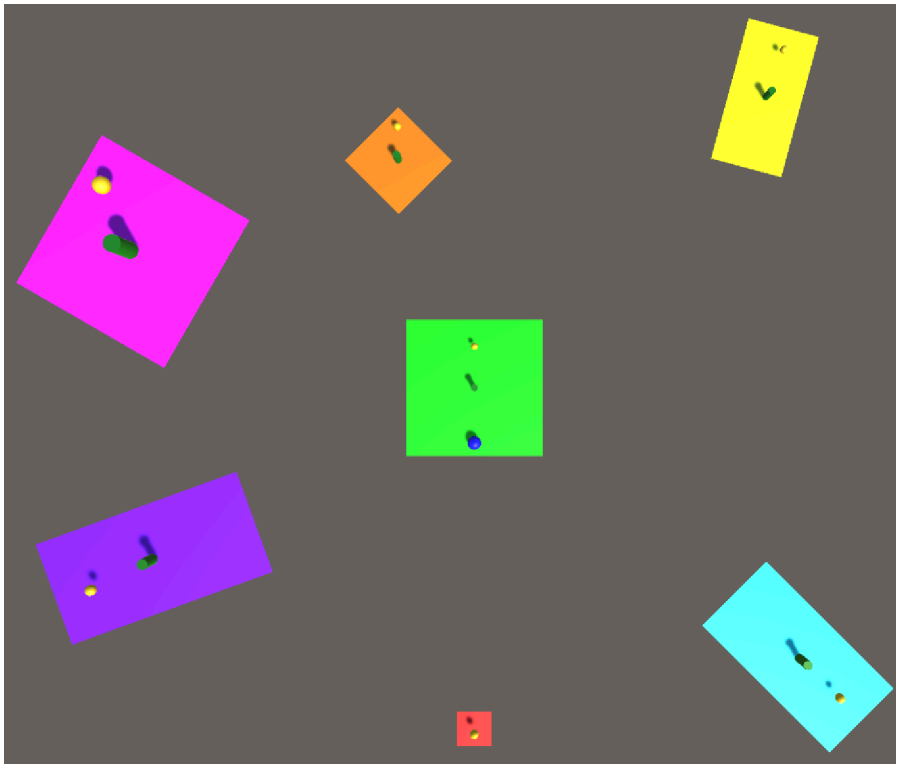


- The second new level is the first implementation of my super jump power-up. It's a relatively simple level with two platforms at different Y levels, with the higher one having a ramp to get back down. Instead of making a large collection of walls to guarantee that the player always makes their jump, I copied part of the code from the ghosts into an invisible cube object underneath everything to make a death plane. The color scheme for this level was chosen to somewhat resemble Portal.



```
public class KillPlane : MonoBehaviour
{
    0 references
    public void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        SceneManager.LoadScene("Menu");
    }
}
```

- The third final level is a more challenging implementation of the super jump power-up. The level focuses on precision over anything else, turning the game from pacman into something like Marble Blast Ultra. I decided to do this simply because it sounded like fun. The color scheme here was chosen just as an excuse to use every material I had made with the exception of the ones being used for objects in the level.



Power-up:

- The moment I saw we needed a power-up, I knew I wanted to add a jump to the game to give an extra level of depth, and because forcing the player to jump for a single collectible sounded fun. Adding the jumping mechanic took more work than expected, possibly due to the rigidbody. I had assumed the FixedUpdate method was causing the jump to not work, but it was actually because of the value used for the jump height as it was simply too weak. Boosting it from 10 to 1000 gave me the super jump I wanted. Furthermore, a canJump boolean was implemented to make sure players could only jump when they collided with the power-up.

```
    void Update()
    {
        if (canJump && Input.GetButtonDown("Jump"))
        {
            Debug.Log("jumped");
            Vector3 jump = new Vector3(0.0f, 1.0f, 0.0f);
            rb.AddForce(jump * jumpHeight);
            canJump = false;
        }
    }

    0 references
    void OnTriggerEnter(Collider other) {
        if (other.CompareTag("Powerup"))
        canJump = true;
    }
```

Collectibles:

- For the collectibles, I referenced code I had written for a previous project that also had collectibles. The code was composed of two scripts, one for the collectible and one to count them. The collectible code simply destroys the object when it's collided with by the player, while the collectible count counts the amount of collectibles destroyed. Once the collected value matches the collectible total, the scene changes. To allow the player to return to the menu after finishing the last level, a second version of the counter script was made.

```
public class Collectible : MonoBehaviour
{
    1 reference
    public static event Action OnCollected;
    1 reference
    public static int total;

    0 references
    void Awake() => total++;

    0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            OnCollected?.Invoke();
            Destroy(gameObject);
        }
    }
}
```

```
public class Counter : MonoBehaviour
{

    1 reference
    public int CollectibleTotal = 0;
    3 references
    int count;

    0 references
    void OnEnable() => Collectible.OnCollected += OnCollectibleCollected;
    0 references
    void onDisable() => Collectible.OnCollected -= OnCollectibleCollected;


    // Update is called once per frame
    0 references
    void Update()
    {
        Debug.Log(count);

        if (count == CollectibleTotal)
        {
            Debug.Log("Level end");
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1);

        }
    }

    2 references
    void OnCollectibleCollected()
    {
        count++;
    }
}
```