Lab Week 4

50.033 Game Design and Development

1004147 --- Riley Riemann Chin
Are you participating in the Weekly Lab competition? ~~Yes~~/No

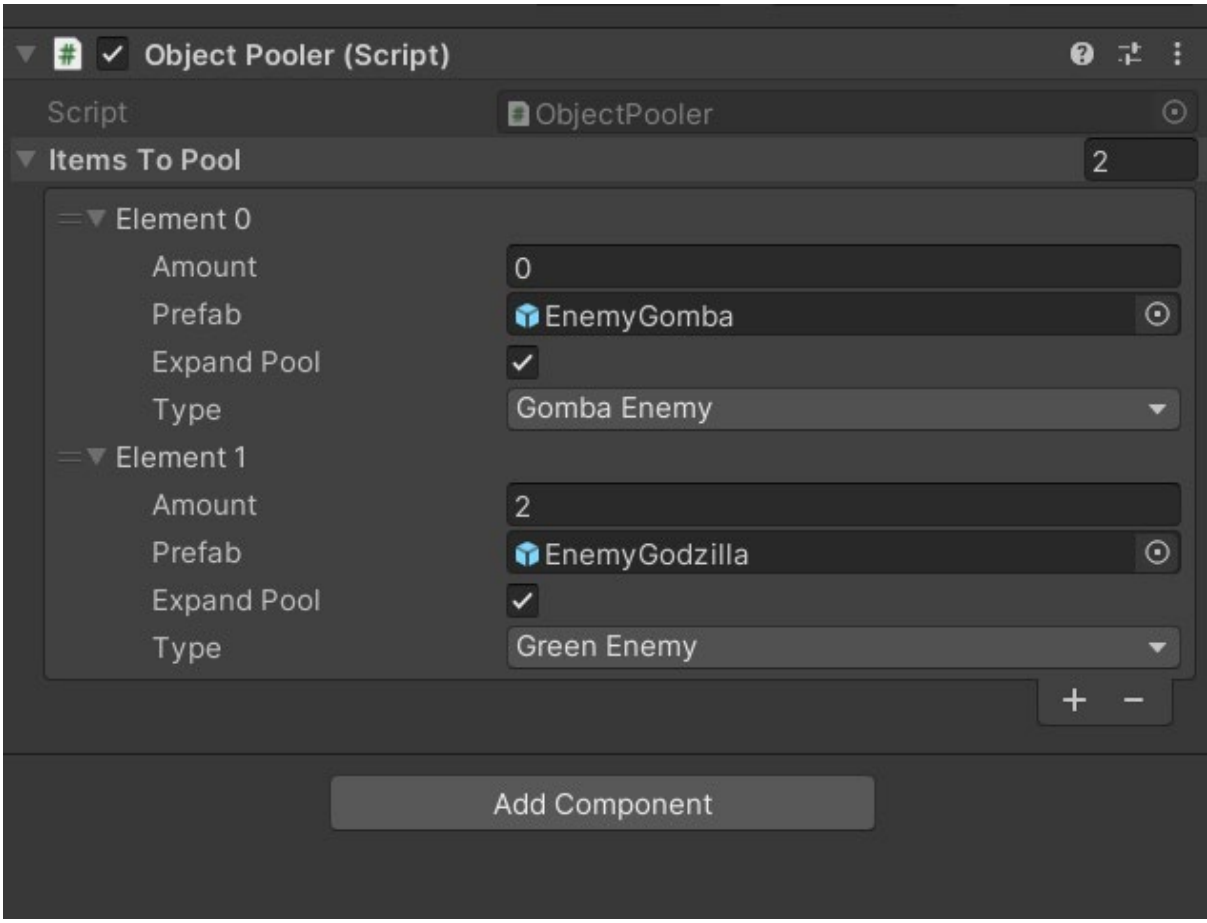
Provide the YouTube/other platform link to your screen recording:
https://youtu.be/Qs2hay9C54g

Provide the link to your lab repository:
https://github.com/rileychin/50.033-Game-Dev-Labs/tree/lab4

Describe what you have done to achieve the desired checkoff requirement for this lab:
[Your **high level** description here]

- You don't need to be too specific. This is just to assist us when we check your repository
- Scripts added, Assets added if any
- General modifications that you have done: eg animating the enemies, implementing FSM for the NPCs, etc.

- Use the ObjectPooler and spawn 2 enemies at a time. The Player can “Kill” the enemy any way you want, such as stepping from the top. This will increase score and spawn one new enemy. Make sure you show that you use an object pooler by recording the Hierarchy as well.
    - To spawn 2 enemies at a time, I just edited SpawnManager to spawn 2 greenEnemy when the script starts. This will call the spawn from pooler object twice for the green enemy and just add it into pooledObjects. The pooledObjects are already instantiated with the objects by setting the settings from the gameObject settings directly.



- I will expand pool for both so that it will add into the existing pool if there is not enough objects in pool

```
// Start is called before the first frame update
void Start()
{
    GameManager.OnCollectCoin += spawnNewEnemy;
    GameManager.OnEnemyDeath += spawnNewEnemy;
    // SceneManager.sceneLoaded += startSpawn; // will spawn a ne

    for (int i = 0; i < 2; i++){
        spawnFromPooler(ObjectType.greenEnemy);
    }
}

void startSpawn(Scene scene, LoadSceneMode mode)
{
    for (int i = 0; i < 1; i++){
        spawnFromPooler(ObjectType.gombaEnemy);
    }
}
```

- startSpawn is disabled so we don't spawn any goomba's at first, but the gomba will be added into the existing pool because of the objectpooler settings for expandPool
- 

- Here's the hierarchy for enemyspawnpool when the game starts

- Create some "interactive" bricks to collect coins or whatever rewards that will increase score but spawn one new enemy. It is up to you to determine how many max enemies can be present at a time. In the gif below, we limit total number of gomba enemies and green enemies to be 5.
  - We first create a coin prefab, attached with a coin.cs script.
  - This script will literally only have the OnTriggerEnter2D() function for when the player (haven't coded out that portion) will collide with the coin, this will cause the cascading function calls for CentralManager.centralManagerInstance.increaseScore()
  - I also added a collectCoin() function in CentralManager which adds the spawnNewEnemy method in SpawnManager, such that it will also spawn a new enemy.
  - ```
    public class SpawnManager : MonoBehaviour
    {

        float groundDistance = -1.0f;

        // Start is called before the first frame update
        void Start()
        {
            GameManager.OnCollectCoin += spawnNewEnemy;
            GameManager.OnEnemyDeath += spawnNewEnemy;
            // SceneManager.sceneLoaded += startSpawn; // will spawn a new gomba everytime the scene loads

            for (int i = 0; i < 2; i++){
                spawnFromPooler(ObjectType.greenEnemy);
            }
        }
    ```
  - To limit the number of enemies I just used an if condition in ObjectPooler.cs which will not spawn and add a new object to pool if the count is already >= 5.
    ```

```
// this will be called when no more active object is present, item to expand pool if required
if (pooledObjects.Count < 5)
{
    foreach (ObjectPoolItem item in itemsToPool)
    {

        Debug.Log(item.type);
        if (item.type == type)
        {
            if (item.expandPool)
            {
                GameObject pickup = (GameObject)Instantiate(item.prefab);
                pickup.SetActive(false);
                pickup.transform.parent  =  this.transform;
                pooledObjects.Add(new  ExistingPoolItem(pickup, item.type));
                Debug.Log("adding items to pool");
                Debug.Log(pooledObjects.Count);
                return  pickup;
            }
        }
    }
}
```

    - It is a simple solution, but the problem is that if it reaches the maximum 5 pooledobjects, then it will return null and not spawn a new object. But it will just spawn until the maximum object count and then stop adding items to pool from there.
- Player will “die” under certain conditions (up to you), and this will cause the enemy to “rejoice”. Both actions of player dying and enemy rejoicing must be clear to the player. In the gif sample below, player will die if it collides with the enemy from anywhere except from the top. The enemies have this cute little dance when the player dies, and player’s death is animated as well.
    - Originally when mario dies in lab 2 I had to disable time.timescale to be 0.0f, which effectively pauses a game and stops all movement. However in this lab I changed it such that I stop mario and the enemies from moving by vector2(0,0) for both. They will then call the marioDeathDance() coroutine and enemydance() coroutine where it is just effectively flipping the X transform of the gameobject periodically.

```
IEnumerator enemyDance()
{
    if (gameObject.activeSelf)
    {
        velocity = new Vector2(0,0);
        while (true)
        {
            //Debug.Log("Flipping");
            enemySprite.flipX = true;
            yield return new WaitForSeconds(0.5f);
            //Debug.Log("Flipping back");
            enemySprite.flipX = false;
            yield return new WaitForSeconds(0.5f);
        }

    }
}
```

- Here is an example of the enemy dancing.

```
// animation when player is dead
void  EnemyRejoice(){
    Debug.Log("Enemy killed Mario");
    // do whatever you want here, animate etc
    // ...

    StartCoroutine(enemyDance());

}
```

- Calling EnemyRejoice.

```
// called when the player dies
void  PlayerDiesSequence(){
    // Mario dies
    Debug.Log("Mario dies");
    // do whatever you want here, animate etc
    // ...
    restartText.gameObject.SetActive(true); // Set active to be true for restart text to tell people how to restart
    restartText.text = "Press 'R' to restart";
    marioAnimator.SetTrigger("isDead");

    marioBody.velocity = new Vector2(0,0);
    deadState = true;
    StartCoroutine(marioDeathDance());
}
```

- Calling PlayerDiesSequence()

- All these scripts will be called because of the event defined in GameManager, this will help by making it easier to compile all the things that will happen when the player dies. For every method added to the event, we have to remove it when the script destroys, if not it will continuously add it when the scene is reloaded again!

```
// Start is called before the first frame update
void  Start()
{
    marioSprite = GetComponent<SpriteRenderer>();
    // Set to be 30 FPS
    Application.targetFrameRate =  60;
    marioBody = GetComponent<Rigidbody2D>();
    restartText.text = ""; //set restart text to be nothing when game started
    marioAnimator  =  GetComponent<Animator>();
    marioAudio = GetComponent<AudioSource>();

    dustCloud = GameObject.Find("DustCloud").GetComponent<ParticleSystem>(); // cannot use GetCompon

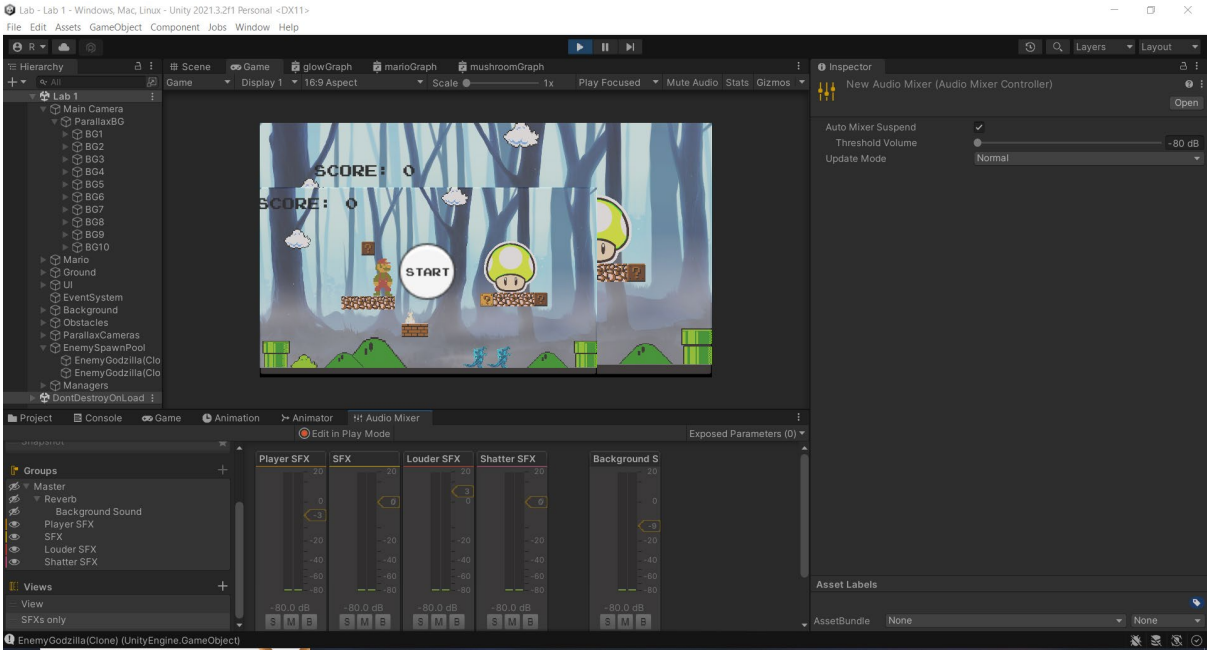
    GameManager.OnPlayerDeath  +=  PlayerDiesSequence; // when the player dies do a death sequence
}
```

- Adding the last line at the start() method

```
void OnDestroy()
{
    GameManager.OnPlayerDeath  -=  PlayerDiesSequence;
}
```

- Removing it when the scene restarts.

- Use AudioMixer in your project, in any way you want.
  - By basically following the tutorial I added the audioMixers for the different sound effects for certain events.



    - The audio mixer is then attached to the output defined by the audio mixer created.
    - Additional audio mixers are added for the background and the brick breaking sound and also the jump sound. (will not be shown)
- Powerups: create two different powerups from anywhere e.g: the question boxes. Player can “collect” it as shown, and “cast” it later. Bind the “casting” to two distinct keys, signifying that the player consumes it. The effect should disappear after a fixed number of seconds. For example, the red mushroom allows the player to jump higher for 5 seconds. The max number of powerups that can be collected can be fixed to 1 for each type for simplicity.
    - Also done by following the tutorial, first started by defining the scripts needed to upgrade the player speed or jump.
    - Then creating a powerupmanager script that will manage the object getting consumed by the player, showing up on the UI, etc.
    - Then the last step is to just cast the to consume the powerups.
- Usage of ScriptableObject to store some unchanged constants like enemy speed, fixed patrol locations, etc. This won’t be visible easily in the game, so you need to record yourself clicking to the scriptable object instance and show that the values are used by other scripts.

- ScriptableObjects acts kinda like a database that stores many constants.

```
// for Scoring system
int currentScore;
int currentPlayerHealth;

// for Reset values
Vector3 gombaSpawnPointStart = new Vector3
// .. other reset values

// for Consume.cs
public  int consumeTimeStep =  10;
public  int consumeLargestScale =  4;

// for Break.cs
public  int breakTimeStep =  30;
public  int breakDebrisTorque =  10;
public  int breakDebrisForce =  10;

// for SpawnDebris.cs
public  int spawnNumberOfDebris =  10;

// for Rotator.cs
public  int rotatorRotateSpeed =  6;

// for testing
public  int testValue;

public float maxOffset = 5.0f;
```

- Here are the list of constants provided.

```
private  Rigidbody2D rigidBody;
private  Vector3 scaler;
public GameConstants gameConstants;

// Start is called before the first frame update
void  Start()
{
    // we want the object to have a scale of 0 (disappear) after 30 frames.
    scaler  =  transform.localScale  / (float) gameConstants.breakTimeStep
    rigidBody  =  GetComponent<Rigidbody2D>();
    StartCoroutine("ScaleOut");
}

IEnumerator  ScaleOut(){

    Vector2 direction =  new  Vector2(Random.Range( 1.0f, 1.0f), 1);
    rigidBody.AddForce(direction.normalized  * gameConstants.breakDebrisFor
    rigidBody.AddTorque(gameConstants.breakDebrisTorque, ForceMode2D.Impuls
    // wait for next frame
    yield  return  null;

    // render for 0.5 second
    for (int step =  0; step  < gameConstants.breakTimeStep; step++)
    {
        this.transform.localScale  =  this.transform.localScale  -  scaler;
        // wait for next frame
        yield  return  null;
```

- Example in debris.cs

```
private AudioSource brickAudio;
public Rigidbody2D prefab;
public  GameObject consummablePrefab;
public GameConstants gameConstants;

// Start is called before the first frame update
void Start()
{
    brickAudio = GetComponent<AudioSource>();
}

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

}

void  OnTriggerEnter2D(Collider2D col)
{
    if (col.gameObject.CompareTag("Player") &&  !broken){
        broken  =  true;
        // assume we have 5 debris per box
        for (int x =  0; x<gameConstants.spawnNumberOfDebris; x++){
            Instantiate(prefab, transform.position, Quaternion.identity);
        }
        gameObject.transform.parent.GetComponent<SpriteRenderer>().enabled  =  false;
        gameObject.transform.parent.GetComponent<BoxCollider2D>().enabled  =  false;
        GetComponent<EdgeCollider2D>().enabled  =  false;
        brickAudio.PlayOneShot(brickAudio.clip); // plays the break brick sound
        Instantiate(consummablePrefab,
        new  Vector3(this.transform.position.x,
        this.transform.position.y  +  1.5f,
        this.transform.position.z),
        Quaternion.identity);
    }
}
```

- Example in breakbrick.cs

```
public GameConstants gameConstants;
private Rigidbody2D enemyBody;


void Start()
{
    enemySprite = GetComponent<SpriteRenderer>();
    enemyBody = GetComponent<Rigidbody2D>();
    // get the starting position
    originalX = transform.position.x;

    moveRight = Random.Range(0,2) == 0 ? -1 : 1;
    ComputeVelocity();
    GameManager.OnPlayerDeath  +=  EnemyRejoice; // reference to static class, enemy will rejo
}


void ComputeVelocity(){
    velocity = new Vector2((moveRight)*gameConstants.maxOffset / gameConstants.enemyPatrolTime
}
```

- Example in enemyController.cs