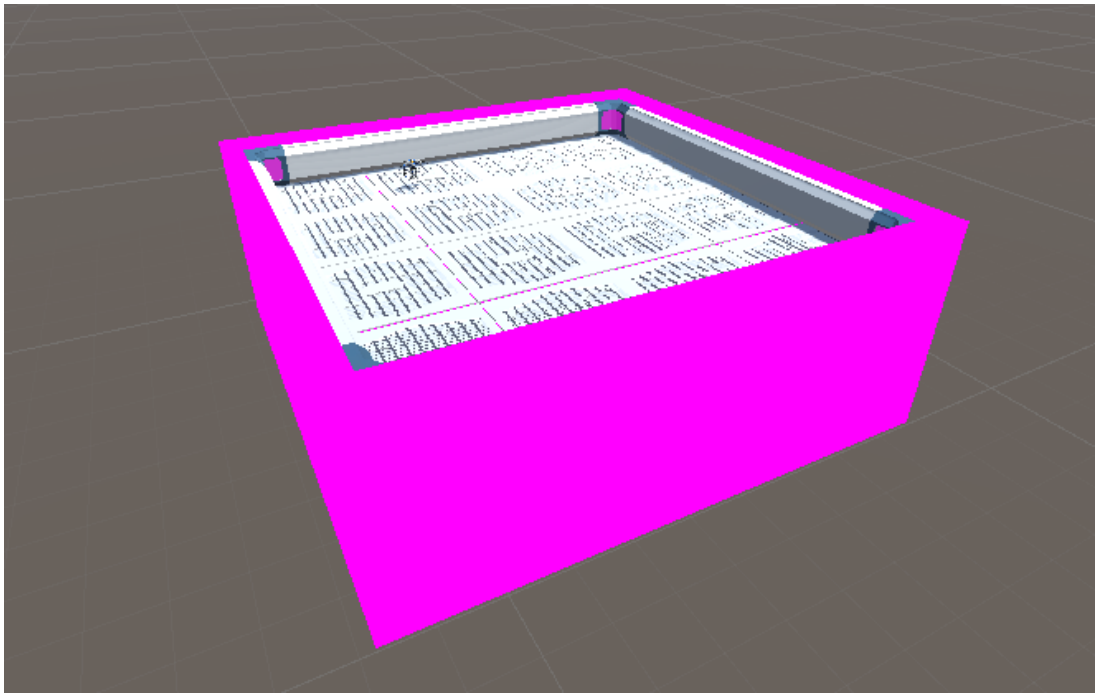
A Singleton manager to rule them all – Exercise 06

Dr. David Israel Flores Granado

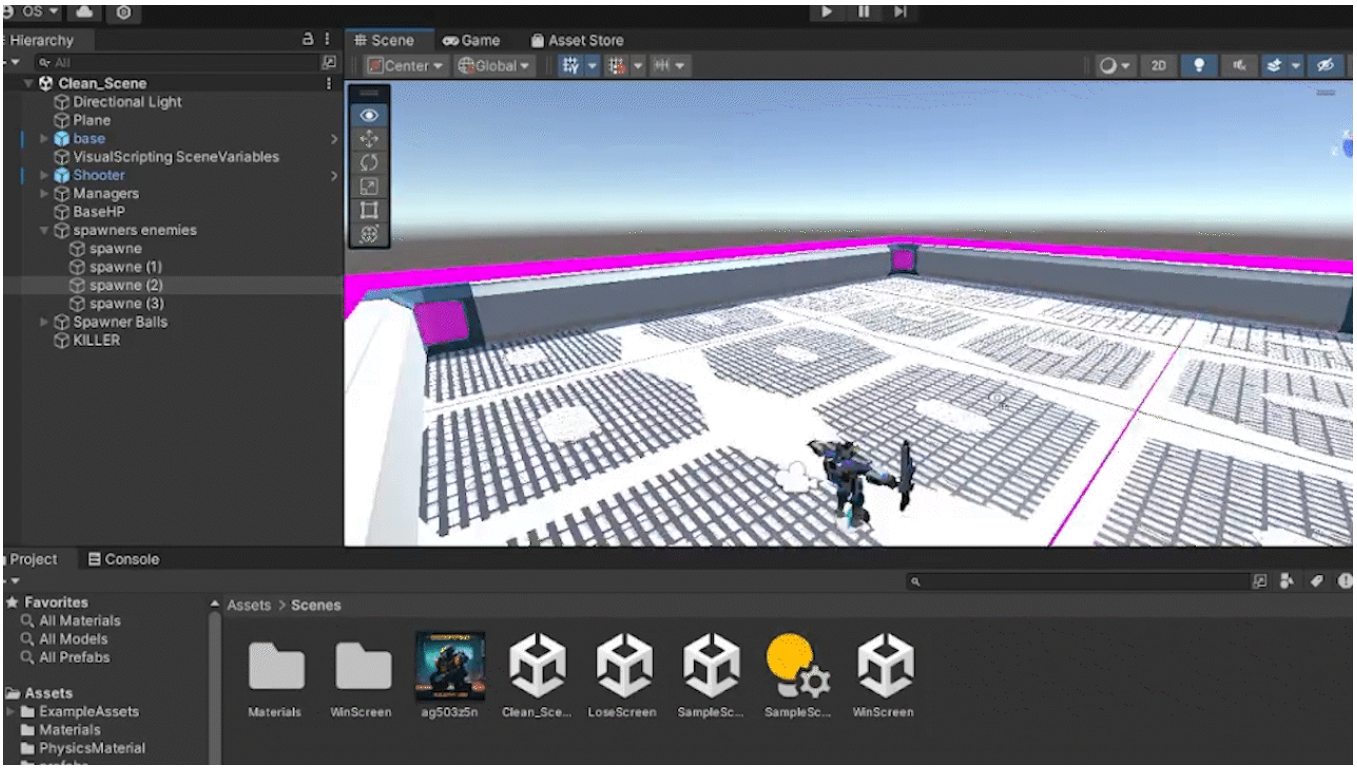
11/8/2023

Team	Responsibility
Ariel	C# Script
Yadiel	C# Scripts
Omar	GIFs/PDF file

The Stage



The Spawners (Click for GIF)



C# Scripts

Enemy Manager

```
6   public class EnemyManager : MonoBehaviour
7   {
        2 references
8       public static EnemyManager instance;
        2 references
9       public List<Enemy> enemies;
        2 references
10      public UnityEvent onChanged;
11
12      // Simpleton design pattern.
        0 references
13      void Awake(){
14          if(instance == null){
15              instance = this;
16          }
17          else{
18              Debug.LogError("Duplicated Enemy Manager",gameObject);
19          }
20      }
21
        0 references
22      public void AddEnemy(Enemy enemy){
23          enemies.Add(enemy);
24          onChanged.Invoke();
25      }
26
        0 references
27      public void RemoveEnemy(Enemy enemy){
28          enemies.Remove(enemy);
29          onChanged.Invoke();
30      }
31  }
32
```

Score Manager

```
public class ScoreManager : MonoBehaviour {

    2 references
    public static ScoreManager instance;

    0 references
    public int amount;

    0 references
    void Awake(){
        if(instance == null){
            instance = this;
        }
        else{
            Debug.LogError("Duplicated Score Manager",gameObject);
        }
    }
}
```

Wave Manager

```
public class WaveManager : MonoBehaviour
{
    2 references
    public static WaveManager instance;
    2 references
    public List<WaveSpawner> waves;
    2 references
    public UnityEvent onChanged;

    // Simpleton design pattern.
    0 references
    void Awake(){
        if(instance == null){
            instance = this;
        }
        else{
            Debug.LogError("Duplicated wave Manager",gameObject);
        }
    }

    0 references
    public void AddWave(WaveSpawner wave){
        waves.Add(wave);
        onChanged.Invoke();
    }

    0 references
    public void RemoveWave(WaveSpawner wave){
        waves.Remove(wave);
        onChanged.Invoke();
    }
}
```

Game Mode

```
public class WavesGameMode : MonoBehaviour
{
    2 references
    [SerializeField] public Life playerLife;
    1 reference
    [SerializeField] public Life playerBaseLife;
    0 references
    void Start(){
        playerLife.onDeath.AddListener(OnPlayerOrBaseDied);
        playerBaseLife.onDeath.AddListener(OnPlayerOrBaseDied);
        EnemyManager.instance.onChanged.AddListener(CheckWinConditions);
        WaveManager.instance.onChanged.AddListener(CheckWinConditions);
    }

    2 references
    private void OnPlayerOrBaseDied()
    {
        SceneManager.LoadScene("LoseScreen");
    }

    2 references
    private void CheckWinConditions()
    {
        if(EnemyManager.instance.enemies.Count <= 0 && WaveManager.instance.waves.Count <= 0){
            //print("You win!");
            SceneManager.LoadScene("WinScreen");
        }
        if(playerLife.amount <= 0){
            //print("You lose :(");
            SceneManager.LoadScene("LoseScreen");
        }

    }

}
```

Floor Manager

```
public class FloorManager : MonoBehaviour
{
    2 references
    public static FloorManager instance;
    2 references
    public List<TypeChild> tiles;
    2 references
    public UnityEvent onChanged;

    // Simpleton design pattern.
    0 references
    void Awake(){
        if(instance == null){
            instance = this;
        }
        else{
            Debug.LogError("Duplicated Floor Manager",gameObject);
        }
    }

    0 references
    public void AddFloor(TypeChild faller){
        tiles.Add(faller);
        onChanged.Invoke();
    }

    0 references
    public void RemoveFloor(TypeChild faller){
        tiles.Remove(faller);
        onChanged.Invoke();
    }
}
```

```
public class AHHFALL : MonoBehaviour
{
    0 references
    public static AHHFALL instance;
    5 references
    public static TypeChild[] arrayTypeChild;//the array where all the floors are being stored
    0 references
    void Start(){
        if (transform.childCount>0){
            arrayTypeChild=transform.GetComponentsInChildren<TypeChild>(true);
        }
    }
    0 references
    public static void SPLEEF(){

            //generate random numbers
            int num = Random.Range(0, 16);
            //print the floor in the array that will get deleted
            print(arrayTypeChild[num]);
            //destroy the chosen array tile
            Destroy(arrayTypeChild[num].gameObject);
        }

    0 references
    public static void KO(){
        for(int i=0; i < arrayTypeChild.Length; i++){
            Destroy(arrayTypeChild[i].gameObject);
        }
    }
}
```

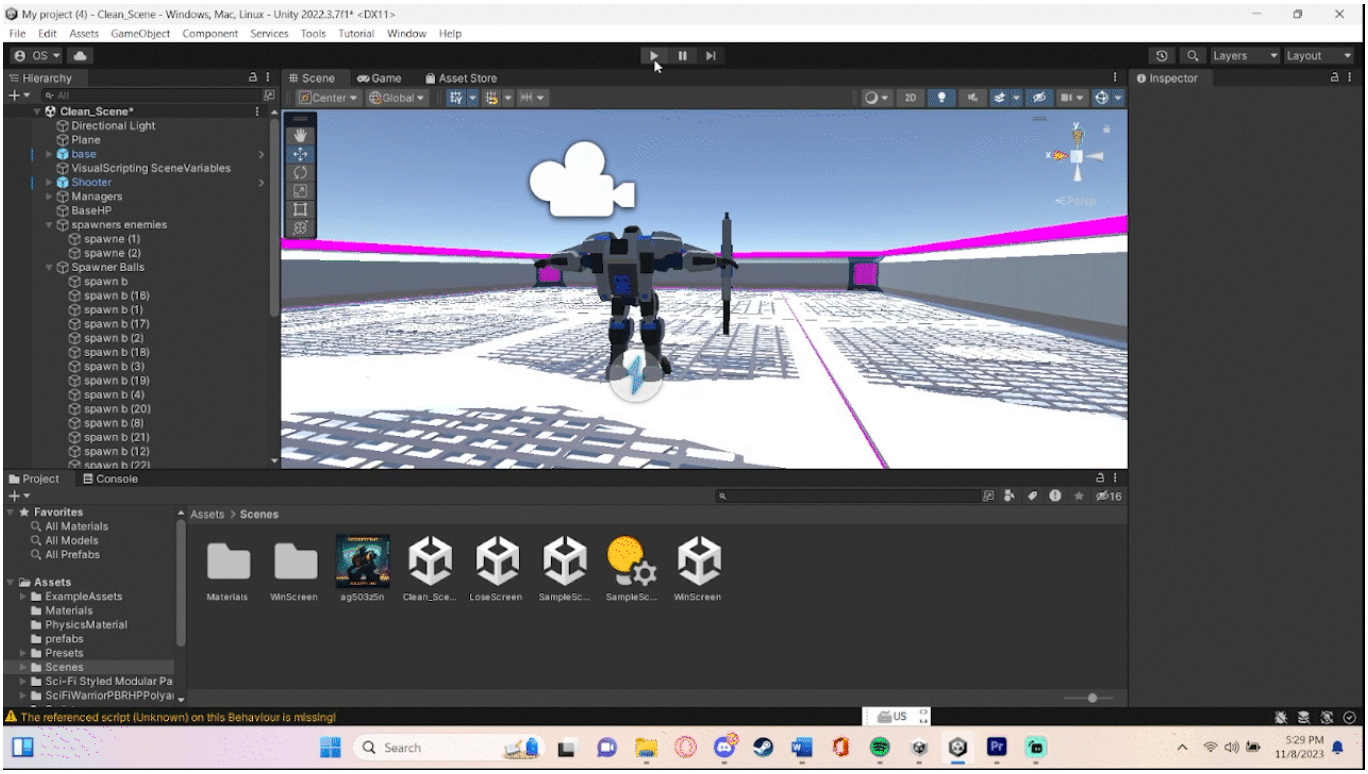
Falling Destroyer

```
public class AHHHIT : MonoBehaviour
{
    0 references
    void OnTriggerEnter(Collider other){
    //public void CallAHHFALL(){
            AHHFALL.SPLEEF();
        //}

    }

}
```

Ball Manager

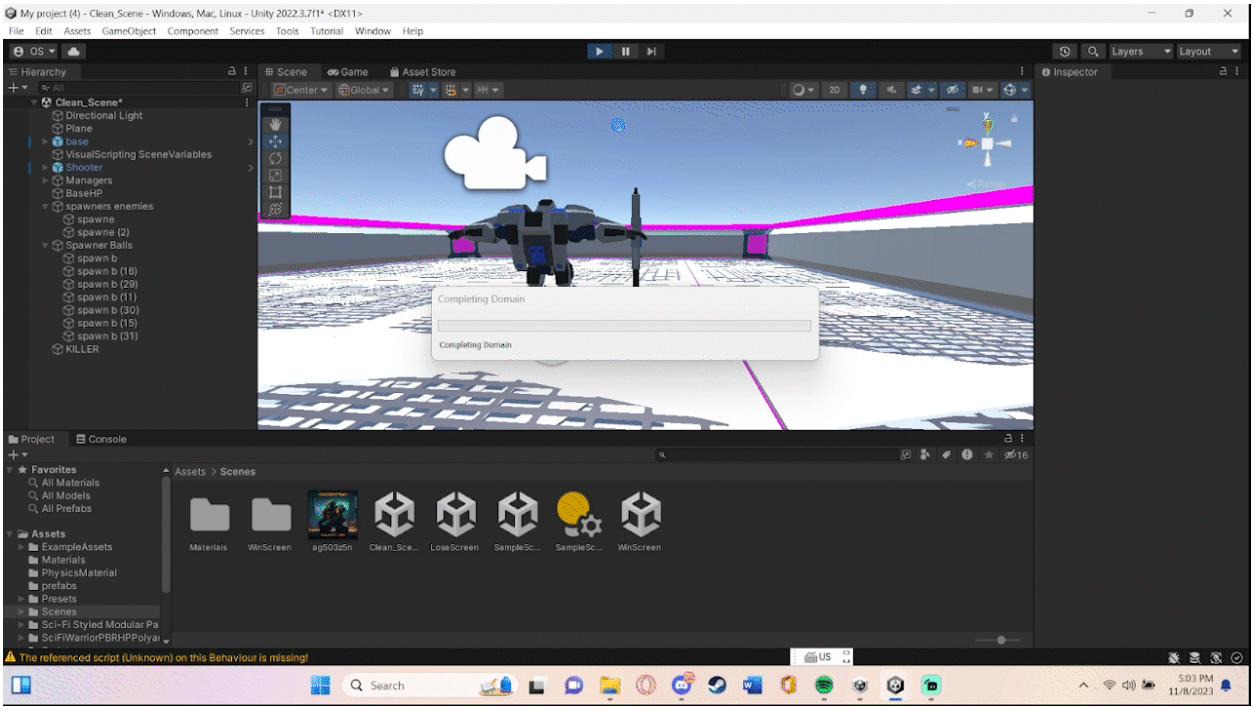
```
public class BallManager : MonoBehaviour
{
    2 references
    public static BallManager instance;
    2 references
    public List<BallSpawner> Balls;
    2 references
    public UnityEvent onChanged;

    // Simpleton design pattern.
    0 references
    void Awake(){
        if(instance == null){
            instance = this;
        }
        else{
            Debug.LogError("Duplicated wave Manager",gameObject);
        }
    }

    0 references
    public void AddBalls(BallSpawner BBalls){
        Balls.Add(BBalls);
        onChanged.Invoke();
    }

    0 references
    public void RemoveBalls(BallSpawner BBalls){
        Balls.Remove(BBalls);
        onChanged.Invoke();
    }
}
```

Rain Showcase (Click for GIF)



Scenes

Win Scene (Click for GIF)



Lose Scene (Click for GIF)