Boone Tison

Midterm Exam

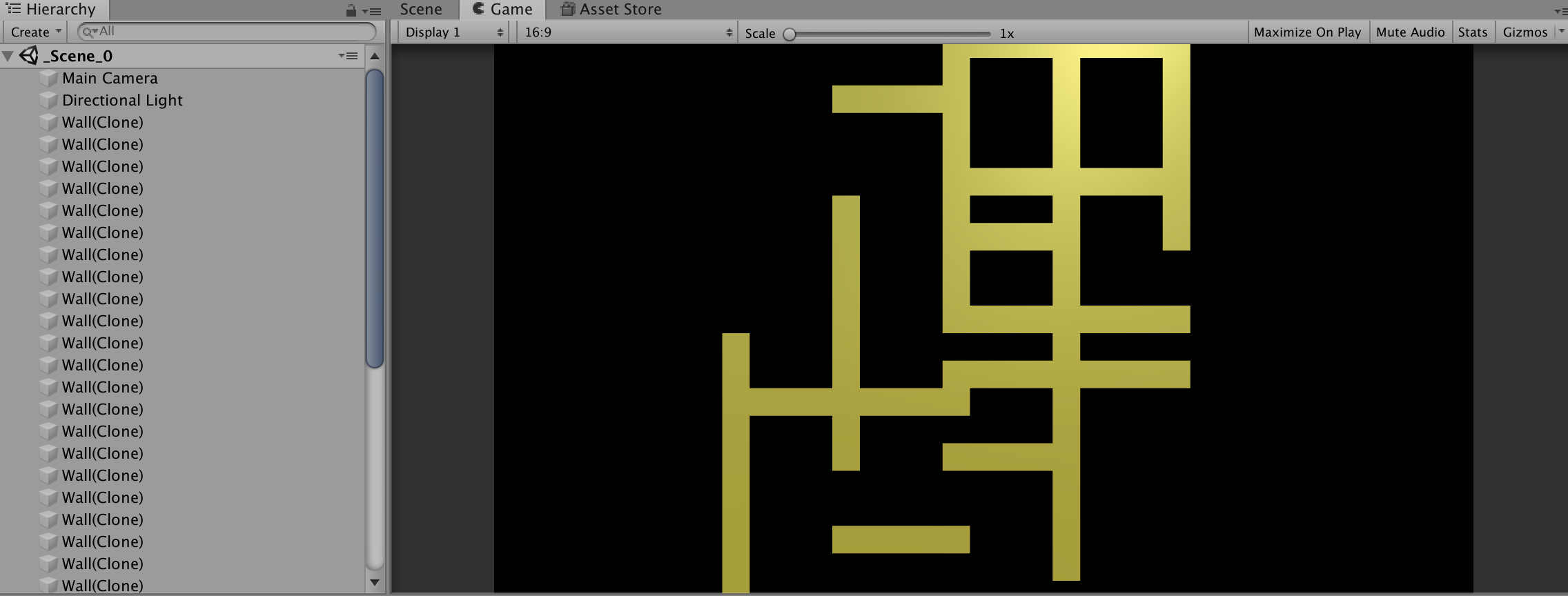
1. Consider a 2D side-scrolling game such as Super Mario Bros or Braid. Our goal is to design a small scene from a level in which the player must pass through the same scene 3 times. However, once a player uses a particular technique to traverse from a starting point to an ending point, the player may not use that same technique again. In particular, you must design three distinct ways to go from a starting point to an ending point in your scene. These methods are not allowed to interact. As in, there is no way to go from start to end using a combination of two methods. Your design must have a theme. Any equipment that a player may use must be reasonable for that theme and the game world. For example, in Super Mario Bros. 3, Mario had a raccoon suit that allowed him to fly for a given amount of time. In your solution, you must describe the scene, its theme, and any equipment required. You must also design clear reasons why the same traversal technique cannot be used a second time. For example, if the character can fly, they would be forced to lose the ability of flight for the other two traversals.

The scene for the level is a cave, with both a ceiling and floor that each change elevation throughout the level. The theme of the level is a cave explorer. At certain points in the level, the ceiling and floor have sharp stalagmites that kill the player when they touch them. There would also be holes with water on the floor, if the player falls into them, they would drown and die. The first traversal of the level has the player traveling on foot on the floor. They have to jump over the stalagmites and avoid them when jumping towards the ceiling. The player also needs to jump over the water holes. The second traversal would result in the player being magnetized to the ceiling. This would mean their head is stuck to the ceiling by magnets, so they cannot simply walk and jump across the level. The player would use a grappling hook that they can use to pull themselves towards the floor. They would release the hook while moving to get forward momentum to traverse around the stalagmites. In addition, the water is highly metallic and is now is creating vertical columns that stretch from the floor towards the ceiling. Therefore, the player needs to quickly move through them using the grappling hook, because if they get stuck in the water, they could drown. During the third traversal, the cave would fill with water and the player would be given scuba gear. The player would now travel through the cave by swimming, still trying to avoid the spikes. However, as an added challenge, the holes which previously were the only sources of water, now are spewing water into the cave at a high velocity, so when the player crosses these holes, they would be pushed quickly towards the ceiling, and potentially spikes.

1. Games that feature many levels often rely on a lock-and-key mechanism to control the player’s progress through each level. In some cases, these lock-and-key mechanisms are depicted as actual locks and keys. For example, in the Legend of Zelda, Link may pick up normal dungeon keys facilitating dungeon traversal. However, in later dungeons Link must also pick up a ‘boss’ key in order to enter the room and fight the boss. For this question, your task is to design at least four (4) ways of using a sword as a key to a ‘lock’. That is, the same sword will be used to unlock four (4) distinct ‘locks’ in the game. Please recall that locks in your game do not have to be actual locks. As an example of a physical lock you may not use in your solution is “A character may unlock a lock by inserting the sword’s blade into an orifice in the wall.” For your solution, (1) clearly describe each lock, (2) how the sword is used to unlock each lock, and (3) how the use of the sword as a key is unique to the other unlocking scenarios. If necessary, provide appropriate details about the game and its associated world.

The first lock would be a puzzle-based one. There are three vertical stones with holes at the top that the player would place their sword in. With the sword in the stones, the player can now rotate their sword and doing so would also rotate the stones. The player needs to get the stones to the correct rotations to unlock the door. To provide a hint for the puzzle, the room would contain hints on how to orient the stones. The second lock requires you to break some chains. The chains are holding the door closed and the player must break all of them with their sword to fully open the door. There are several chains spread throughout the room and as the player breaks them, the door would slowly rise with each chain broken. This would give the player a visual indicator of their progress. The third lock requires the player to throw their sword through a particular hole. There are several holes to pick from and picking the wrong one will lead to deadly consequences. There is also a challenge involved with getting the sword thrown at the right angle in order to fit it in the hole. Once the player throws through the right hole, the lock will open and the player will find their sword on the other side of the door. The fourth lock is a puzzle and moral dilemma. The player is presented with three unarmed and docile enemies. In order to unlock the door, the player must spill the blood of the correct enemy, so kill them. The player has to use the clues in the room to pick the correct enemy. However, killing all the enemies will open the lock, as the correct blood was spilled, just not efficiently.

5. Consider the Unity tutorial Roll-a-ball ‘game’ we implemented as a lab. Write a C# script that will attach to the main camera that will procedurally generate a set of walls to add to the playfield and thus simulate a 2D maze. For simplicity, you may focus on generating copies of one ‘maze wall’ prefab. For the purposes of this question, the maze does not need to be solvable (a path from start to finish does not need to be verified). No assets other than the camera, directional light, and exterior walls should be listed in the Hierarchy when the game is executed. In the inspector, we should be able to set the number of walls, but they should be limited to a reasonable number (around 20). Copy and paste the formatted, commented source code as well as relevant screenshots demonstrating the success of your implementation.



using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Maze : MonoBehaviour

{

private List<GameObject> wallList;

[Header("Set in Inspector")]

public GameObject wallPrefab;

public int numWalls = 20;

public int wallSpacingY = 5;

public int wallSpacingX = 2;

public float minX = -17f;

public float minY = -10f;

public float maxX = 17f;

public float maxY = 10f;

private int rotateZ = 90;

// Start is called before the first frame update

void Start()

{

// Find the initial start of the generation

int startX = (int)Random.Range(minX, maxX);

int startY = (int)Random.Range(minY, maxY);

int rot = (int)Random.Range(1, 3) \* rotateZ;

for (int i = 0; i < numWalls; i++)

{

GameObject tWallGO = Instantiate<GameObject>(wallPrefab);

Vector3 pos = Vector3.zero;

pos.x = startX;

pos.y = startY;

tWallGO.transform.position = pos;

tWallGO.transform.rotation = Quaternion.Euler(0, 0, rot);

// Choose the next random direction

int chance = (int)Random.Range(1, 100);

if (chance <= 25)

{

startY += wallSpacingY; // Go up

}

else if (chance <= 50)

{

startY -= wallSpacingY; // Go down

}

else if (chance <= 75)

{

startX -= wallSpacingX; // Go left

rot -= rotateZ;

}

else

{

startX += wallSpacingX; // Go right

rot += rotateZ;

}

// Bounds check

if (startX > maxX || startX < minX)

{

startX = (int)Random.Range(minX, maxX);

}

if (startY > maxY || startY < minY)

{

startY = (int)Random.Range(minY, maxY);

}

}

}

}