CECS 528 2020 HW2 (3) Unity game project notes - 2

Game Genre: 2D Side Scrolling with endless level _____ game

2. Generation of endless level (consisting of rooms)

2.1 Room design

`bg`	480x640
`bg_window`	480x640

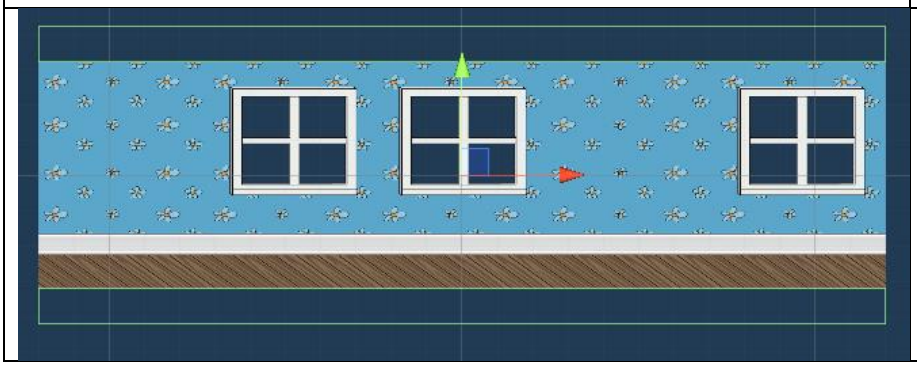
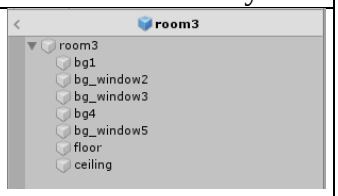
Samples of rooms:

a) room1 (`bf + bf_window + bf`)

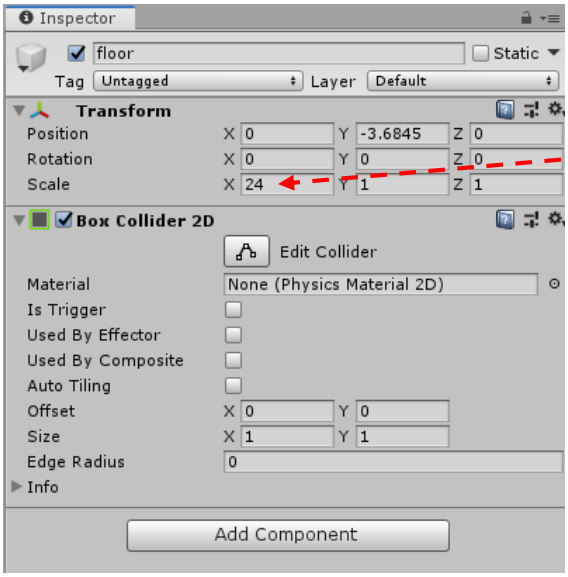
b) room2 (`bf_window + bf + bf_window`)

2.2 Create room prefabs

a) Create room3 in the scene as shown in the following picture:

room3 in the scene	room3 in Hierarchy
	

The `floor` game object components are shown in the following Inspector:



X scale = 24 = 4.8 x 5
(4.8 unit per bf)

b) Create a `Prefabs` folder in the Project.

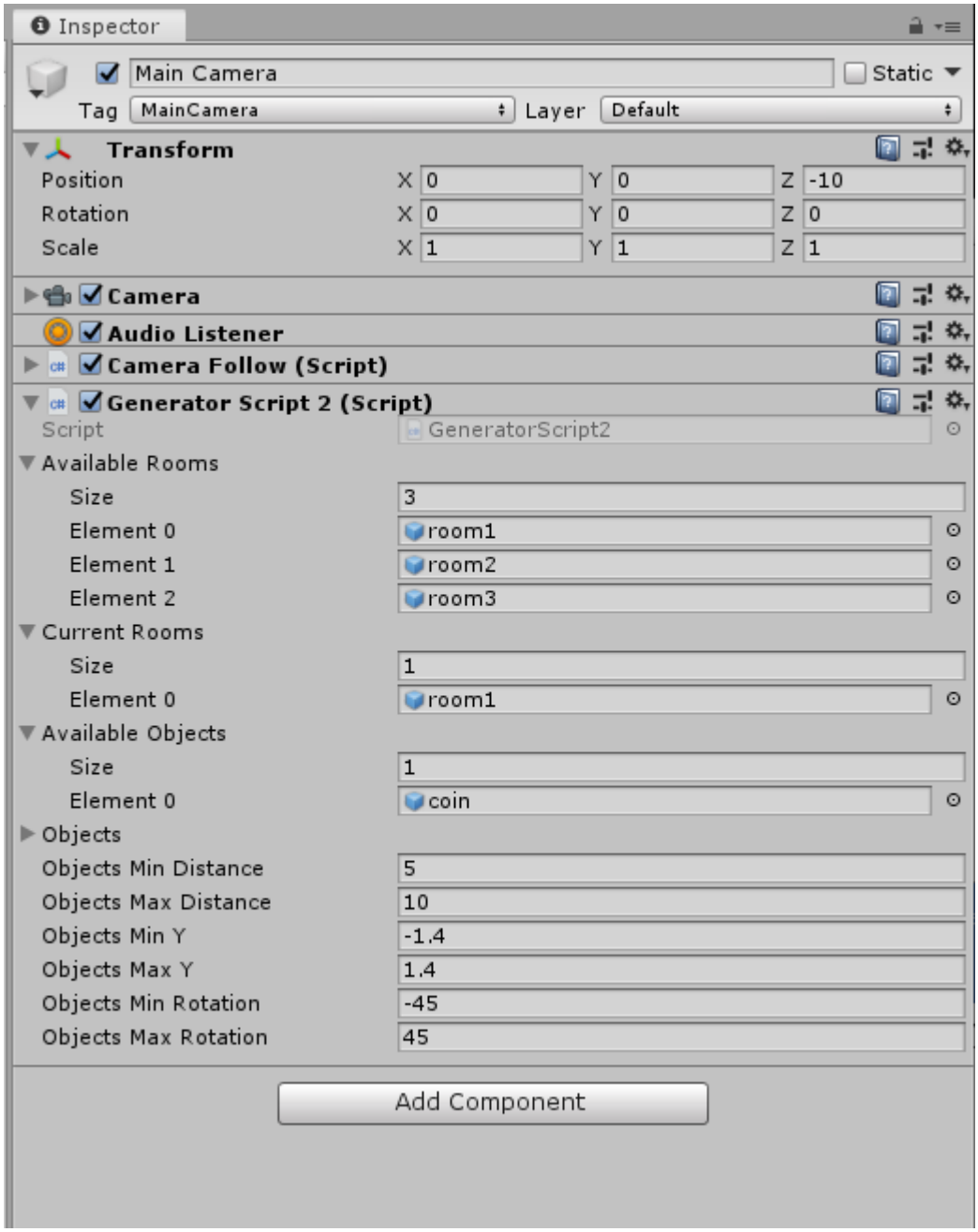
c) Drag the `room3` game object from the scene and drop it in the `Prefabs` folder.

d) Delete the `room3` game object from the scene.

e) Double click the `room3` prefab to edit it.

2.3 Generate endless level using `room` prefabs.

Add the script, `GeneratorScript.cs`, to the Main Camera game object and set the properties of the script as follows:



2.4 The idea behind the endless room generation (GeneratorScript.cs)

```
public GameObject[] availableRooms;
public List<GameObject> currentRooms;
private float screenWidthInPoints;

// These are for dynamic objects (for pick up, etc.)
public GameObject[] availableObjects;
public List<GameObject> objects;

public float objectsMinDistance = 5.0f;
public float objectsMaxDistance = 10.0f;

public float objectsMinY = -1.4f;
public float objectsMaxY = 1.4f;

public float objectsMinRotation = -45.0f;
public float objectsMaxRotation = 45.0f;


private void Start()
{
    float height = 2.0f * Camera.main.orthographicSize;
    screenWidthInPoints = height * Camera.main.aspect;
    StartCoroutine(GeneratorCheck());
}
.....................................................

private IEnumerator GeneratorCheck()
{
    while (true)
    {
        GenerateRoomIfRequired();
        GenerateObjectsIfRequired();
        yield return new WaitForSeconds(0.25f);
    }
}
```

```
private void GenerateRoomIfRequired()
{
    List<GameObject> roomsToRemove = new List<GameObject>();
    bool addRooms = true;

    float playerX = transform.position.x;

    // This is the point after which the room should be removed
    float removeRoomX = playerX - screenWidthInPoints;

    // If there is no room after the addRoomX point, then you need to add a room,
    // since the end of the level is closer than the screen width.
    float addRoomX = playerX + screenWidthInPoints;

    float farthestRoomEndX = 0;

    foreach (var room in currentRooms)
    {
        // You use the floor to get the room width and calculate the roomStartX
        // (the point where the room starts, i.e. the leftmost point of the room)
        // and roomEndX (the point where the room ends, i.e. the rightmost point
        // of the room).
        float roomWidth = room.transform.Find("floor").localScale.x;
        float roomStartX = room.transform.position.x - (roomWidth * 0.5f);
        float roomEndX = roomStartX + roomWidth;

        // If there is a room that starts after addRoomX then you don’t need to
        // add rooms right now.
        if (roomStartX > addRoomX)
        {
            addRooms = false;
        }

        // If the room ends to the left of the removeRoomX point, then it is
        // already off the screen and needs to be removed.
        if (roomEndX < removeRoomX)
        {
            roomsToRemove.Add(room);
        }

        // Here you simply find the rightmost point of the level. This is the
        // point where the level currently ends. It is used only if you need to
        // add a room.
        farthestRoomEndX = Mathf.Max(farthestRoomEndX, roomEndX);
    }

    foreach (var room in roomsToRemove)
    {
        currentRooms.Remove(room);
        Destroy(room);
    }
```

```
    if (addRooms)
        AddRoom(farthestRoomEndX);

}

private void AddRoom(float farhtestRoomEndX)
{
    int randomRoomIndex = Random.Range(0, availableRooms.Length);
    GameObject room = (GameObject)Instantiate(availableRooms[randomRoomIndex]);

    // Since the room is just an Empty GameObject containing all the room parts,
    // you cannot simply take its size. Instead, you get the size of the floor
    // inside the room, which is equal to the room's width.
    float roomWidth = room.transform.Find("floor").localScale.x;

    // This sets the position of the room.
    float roomCenter = farhtestRoomEndX + roomWidth * 0.5f;
    room.transform.position = new Vector3(roomCenter, 0, 0);

    currentRooms.Add(room);
}
```