Terrain generation

m00ns7ruck

January 22, 2021

Contents

1 Why?

Dynamically generated terrain brings more organic feeling to a game. This is valid even more, when the player is spawned in the same area over and over to fulfill the same goal. In other words it makes the game world more organic, immersive and allows more replayability. I wanted to create a 2D top-down terrain, composed of ground and lava tiles. The terrain also had to be dynamic - the ground should be shrinking within a given period. The lava tiles had to deal damage to the player and every object, that has some kind of health metric (e.g. hit points, durability, etc.). The terrain generation had to spawn obstacles as well. For example boxes and rocks. The box is the movable obstacle and the rock is immovable. The terrain is being used in a prototype of a game I am developing.

I thought of two ways to achieve this. Using prefabs and instantiating them and a dynamically generated tilemap.

2 Stuck point

In the both methods, described below, I stuck on the use of the message 'OnTriggerStay2D'.

As the Unity documentation description states (`https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerStay2D.html`):

> Sent each frame where another object is within a trigger collider attached to this object (2D physics only). Further information about the other collider is reported in the Collider2D parameter passed during the call. Note: Trigger events will be sent to disabled MonoBehaviours, to allow enabling Behaviours in response to collisions.

I understand this as if it should be sent every frame, when two colliders are in contact. I planned to use this message to start a coroutine that does damage to a given target within the desired period. In other words do damage to the targets in the lava tile as long as they stay in there. In reality, the message is sent only when there was a change in the colliding objects. In other words the object took damage only when it was moving.

3 Prefabs

For this method I created 4 prefabs. Two obstacles - movable and immovable, and two terrain tiles - lava and ground.

3.1 Lava tile

The most interesting of these is the lava prefab. It is a game object with sprite and a script.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Lava : MonoBehaviour {
    [SerializeField] private int damage = 1;
    [SerializeField] private float timeBetweenDamage = 2f;

    private bool waiting = false;
    private HashSet<ObjectStats> targets;

    void Start() {
        targets = new HashSet<ObjectStats>();
    }

    void OnTriggerEnter2D(Collider2D target) {
        ObjectStats targetStats = target.GetComponent<ObjectStats>();
        if(targetStats) {
            targets.Add(targetStats);
        }
    }

    void Update() {
        if(! waiting) {
            waiting = true;
            StartCoroutine(DamageTargets());
        }
    }

    void OnTriggerExit2D(Collider2D target) {
        ObjectStats targetStats = target.GetComponent<ObjectStats>();
        if(targetStats) {
            targets.Remove(targetStats);
        }
    }

    private IEnumerator DamageTargets() {
        foreach(ObjectStats targetStats in targets) {
            targetStats.TakeDamage(damage);
        }
        yield return new WaitForSeconds(timeBetweenDamage);
        waiting = false;
    }
}
```

The script works around the problem with the OnTriggerStay2D message by maintaining a collection of all objects that are damageable. The entries are added when OnTriggerEnter2D is sent and are removed on OnTriggerExit2D. Then in the Update message, all of the targets are damaged.

3.2 Generating

The script that creates the terrain has the reference to the four prefabs. It has also different parameters to control the randomness. It also handles the shrinking of the ground.

The terrain is generated around the (0, 0). With given width and height.

```
for (int x = (-width / 2); x < (width / 2); ++x) {
    for (int y = (-height / 2); y < (height / 2); ++y) {
        ...
    }
}
```

Whether the (x, y) tile should be lava or ground is also controlled by the input parameters. There is also random factor (also an input parameter).

```
float groundHeightUpperLimit = Random.Range(
    (groundHeight / 2),
    (groundHeight / 2 - randomFactor)
);
float groundHeightLowerLimit = Random.Range(
    (-groundHeight / 2),
    (-groundHeight / 2 + randomFactor)
);
float groundWidthLeftLimit = Random.Range(
    (-groundWidth / 2),
    (-groundWidth / 2 + randomFactor)
);
float groundWidthRightLimit = Random.Range(
    (groundWidth / 2 - randomFactor),
    (groundWidth / 2)
);
bool isTileGround = x >= groundWidthLeftLimit
    && x <= groundWidthRightLimit
    && y >= groundHeightLowerLimit
    && y <= groundHeightUpperLimit;
```

```
tileToInstantiate = lavaTile;
if(isTileGround) {
    tileToInstantiate = groundTile;
}
```

The tile is then instantiated and added to the children of the game object. This way multiple instances of the prefabs will not pollute the inspector. Based on the value another randomness input parameter, if the tile is ground, an obstacles is instantiated. It is as well added as a child to the game object.

During the generation of the terrain, a list of game objects is loaded. This way, all the ground tiles will be already collected and ready for deletion - this way the ground part will get smaller.

3.3 Shrinking

The shrink period is an input parameter. So in the Update message, the function that replaces part of the ground tiles with lava, is called within that period.

The function simply uses the list of all ground tiles, calculates the distance between the center of the terrain - (0, 0), and the given object. If it is bigger than a given radius, the tile is swapped with lava. The shrink radius is determined by the lower value of height and width of the terrain. Then with each shrink is decreased by 1.

4 Dynamic tilemap

In this case only 2 prefabs were needed. The box and the rock from the prefab method. The lava and the ground are just tiles.

4.1 Generating

In this approach, the tilemap generates the terrain like in the prefab method.

```
for (int x = (- width / 2); x < (width / 2); ++x) {
    for (int y = (- height / 2); y < height / 2; ++y) {
        float groundHeightUpperLimit = Random.Range(
            (groundHeight / 2),
            (groundHeight / 2 - randomFactor)
        );
        float groundHeightLowerLimit = Random.Range(
            (-groundHeight / 2),
            (-groundHeight / 2 + randomFactor)
        );
        float groundWidthLeftLimit = Random.Range(
            (-groundWidth / 2),
            (-groundWidth / 2 + randomFactor)
        );
        float groundWidthRightLimit = Random.Range(
            (groundWidth / 2 - randomFactor),
            (groundWidth / 2)
        );
        bool isTileGround = x >= groundWidthLeftLimit
            && x <= groundWidthRightLimit
            && y >= groundHeightLowerLimit
            && y <= groundHeightUpperLimit;

        Vector3Int newTilePosition = new Vector3Int(x, y, 0);

        tileToInstantiate = lavaTile;
        if(isTileGround) {
            tileToInstantiate = groundTile;
            groundTiles.Add(newTilePosition);
            if(Random.value >= obstacleSpawnChance) {
                int obstacleIndex = (int)Random.Range(0, obstacles.Count);
                GameObject obstacle = Instantiate(
                    obstacles[obstacleIndex],
                    newTilePosition,
                    Quaternion.identity
                );
                obstacle.transform.parent = this.transform;
            }
        }

        tilemap.SetTile(newTilePosition, tileToInstantiate);
    }
}
```

Instead of instantiating objects, however, the tiles were added to the tilemap via <TILEMAP>.SetTile(<POSITION>, <TILE>). The positions of the ground tiles are again stored in a list. The obstacles are the only objects that are instantiated.

4.2 Lava damage

The tiles cannot have colliders. Therefore the collider messages are not sent. Hence they cannot be used. To work around this I used the collider of the tilemap and again maintained a collection. Adding targets in OnTrigger-Enter2D() and removing - OnTriggerExit2D(). If the position of a unit is not in the list of ground tile positions, then it is considered to be on a lava tile. In the Update() function, a loop iterates through all of the targets and calls a function that deals damage to them. The script maintains a queue with the objects that have been damaged. Each damaged unit stays in the collection for the given period of lava damage. After that it is removed. This way it is secured that each object will be hurt a single time in the given period.

4.3 Shrinking

The method is the same as in the prefab approach.