master 1 branch 3 tags Go to file Code

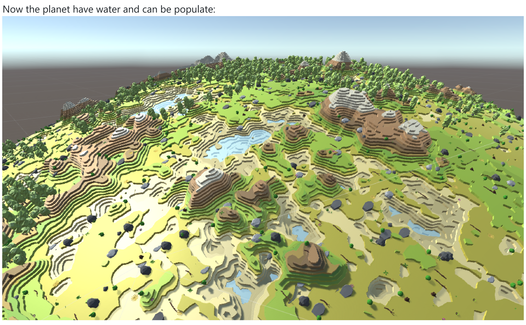
TheHunterJPScripts Update README.md 4d0134b on Nov 19, 2019 153 commits

Scripts	Add files via upload	2 years ago
LICENSE	Initial commit	2 years ago
PopulateData.cs	Add files via upload	2 years ago
README.md	Update README.md	10 months ago

README.md

New feature added:

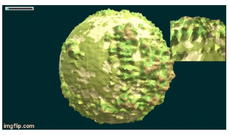
Now the planet have water and can be populate:



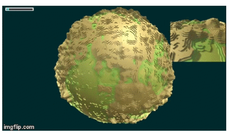
Procedural-planet-generation (Unity Project)

Generate planets proceduraly. (Generate)

Low Poly style:



Terrace style:



How to use?

We only need to import 5 files:

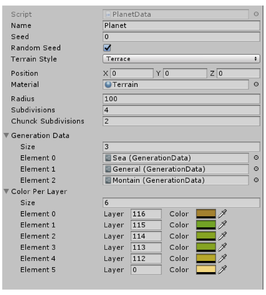
'Planet.cs' , 'PlanetData.cs' , 'Polygon.cs' , 'ColorHeight.cs' , 'GenerationData.cs'.

GenerationData:

This scriptable object stores the information that will then be used to get the height of the map vertices (it uses perlin noise).

PlanetData:

This scriptable object stores the information that will then be used to create a planet.



Generate the planets:

To add a planet to the list of planets that will be created, use the functtion:

> Planet.AddPlanetToQueu();

To start the thread that will calculate the data for the mesh sphere generation and modification of the landscape (will only be executed after StartDataQueu has ended):

> Planet.StartDataQueu();

Instantiate the planet into the scene:

> Planet.InstantiateIntoWorld();

Wait until the planet data have been compute. Then load the chunks in range to the position.

> Planet.HideAndShow(Vector3 position);

Both InstantiateIntoWorld and HideAndShow can be used at the same time. In that case InstantiateIntoWorld will generate the planet until it's fully generated and the will hide all the terrain that is not in range of the position.

The end result should look like this:

```
using UnityEngine;
using Generation;

public class Main : MonoBehaviour
{
    public PlanetData[] planets;
    public GameObject viewer;
    public float viewDistance;
    static public float G_viewDistance;

    public void Start()
    {
        foreach (var item in planets)
            Planet.AddPlanetToQueue(item.newName, item.position, item.seed,
                item.generateRandomSeed, item.terrainStyle, item.seaLevel, item.generationData, item.ColorPerLayer,
                item.population, item.material, item.seaMaterial, item.radius, item.subdivisions, item.chunckSubdivisions);

        Planet.StartDataQueue();
    }
    private void Update()
    {
        G_viewDistance = viewDistance;
        Planet.InstantiateIntoWorld();
        Planet.HideAndShow(viewer.transform.position);
    }
}
```

About

This is an older version of a procedural planet generation. I'm actually working on an other version that is way better optimized and have a lot of added features. Feel free to use this code until I upload the new one. Note: if u plan on using it i recomend to implement an octree for finding the tiles that need to be load, it improve the loop ti...

Readme

Apache-2.0 License

Releases

3 tags

Packages

No packages published

Languages

C# 100.0%