CZU: 004.92

GENERATING 3D MESH FROM DATA FOR A GAME USING C#

Igor RUSSOV, **Andrei BORODAI**,
Maxim GONCHAROV, **Gleb POPSOI**, **Iurie STARENCO**
students, Center of Excellence in Informatics and Information Technologies
Olga CERBU, Ph.D., associate professor
Moldova State University

Abstract. In this article is shown 3D mesh generation from data for a game using C# in case if you want to have diverse maps. However, creating each individual 3d model and texturing would be inefficient. To address this problem, we have created a system that can generate a 3D mesh and apply a texture based on simple data. The map consists of hexagonal tiles, each having a terrain type and an elevation. The map data is stored in an SQL database and is created using a graphical map editor.

Rezumat. În acest articol este prezentată generarea mapelor 3D din date pentru un joc folosind C# în cazul în care doriți să aveți hărți diverse. Cu toate acestea, crearea fiecărui model 3D individual și texturarea ar fi ineficiente. Pentru a rezolva această problemă, am creat un sistem care poate genera un mesh 3D și poate aplica o textură bazată pe date simple. Harta este formată din elemente hexagonale, fiecare având un tip de teren și o înălțime. Datele hărții sunt stocate într-o bază de date SQL și sunt create folosind un editor grafic de hărți.

Keywords: mesh, tile, shader.
Cuvinte cheie: plasă, tile, shader.

Introduction

We have created a strategy game on C#. The game is about battles in the medieval times. Battles take place on maps. I wanted to have diverse maps for interesting battles. However, creating each individual 3d model and texturing would be inefficient. To address this problem, I have created a system that can generate a 3D mesh and apply a texture based on simple data.

The map data

The map consists of hexagonal tiles, each having a terrain type and an elevation. The map data is stored in an SQL database and is created using a graphical map editor [2, 4].

Figure 1. Map editor– colors represent terrain types; numbers represent elevation

Mesh generation

In order to generate a mesh, we have to create polygons that represent the surface. Each hexagon (map tile) consists of six triangles. The height of the central point is equal to the elevation of the tile. Heights of other points are equal to the average elevation of all the tiles that contain this point.

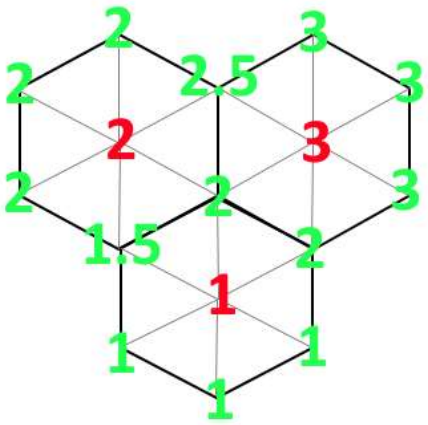

Figure 2. Example of height calculation of points

X and Z coordinates of points are calculated based on the geometrical properties of the hexagon: for example, the top right point X coordinate is equal to the X coordinate of the center + 0.5 of the hexagon total width.

Mesh texturing

The generated mesh is textured according to the terrain types of the map it represents. This is achieved using a surface shader (GLSL) [1]. Texture data for each terrain type is loaded into the shader. For points near the center of the tile, the according texture is simply projected on the mesh using triplanar projection. For the points near the border of the hexes, textures of different terrain types are blended. Each texture's opacity depends on the distance to the center of the hex with the texture's terrain type [3].

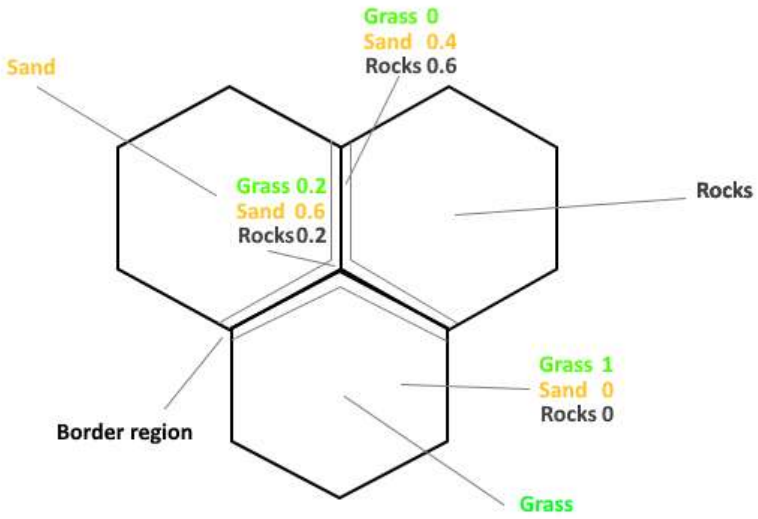

Figure 3. Texture opacity

The diagram shows an example of how texture opacity is calculated in different points of the mesh.

Placing additional objects

To better represent terrain types, trees and bushes are procedurally placed on “Forest” and “Bush” terrain types. Object positions are determined using the Poisson disc sampling.

Displaying the result

The maps are displayed using the Unity game engine.

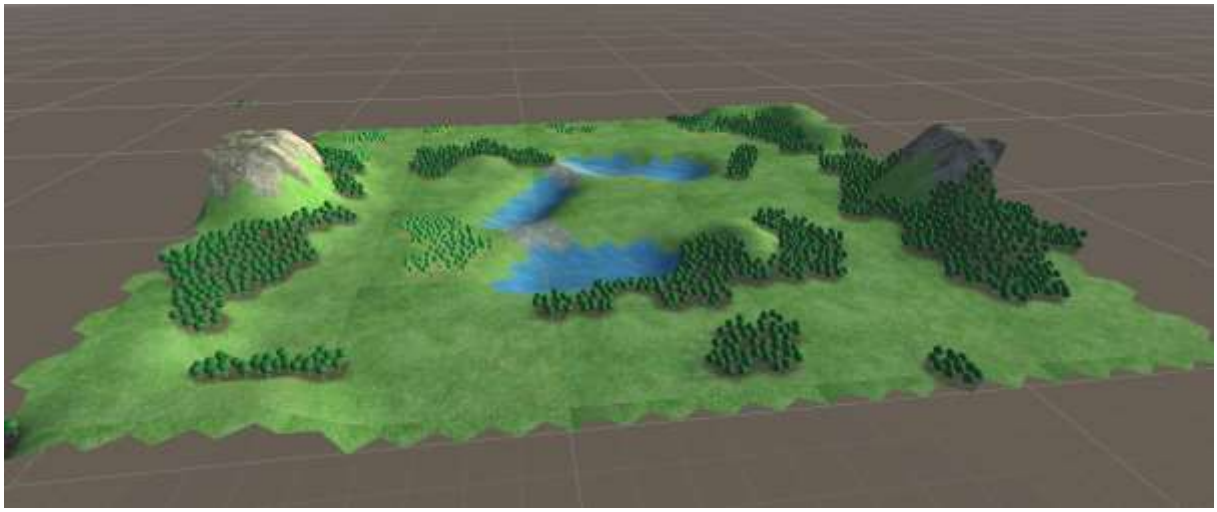

Figure 4. View of the “Center lake” map

Figure 5. Area of the map where many terrain types meet

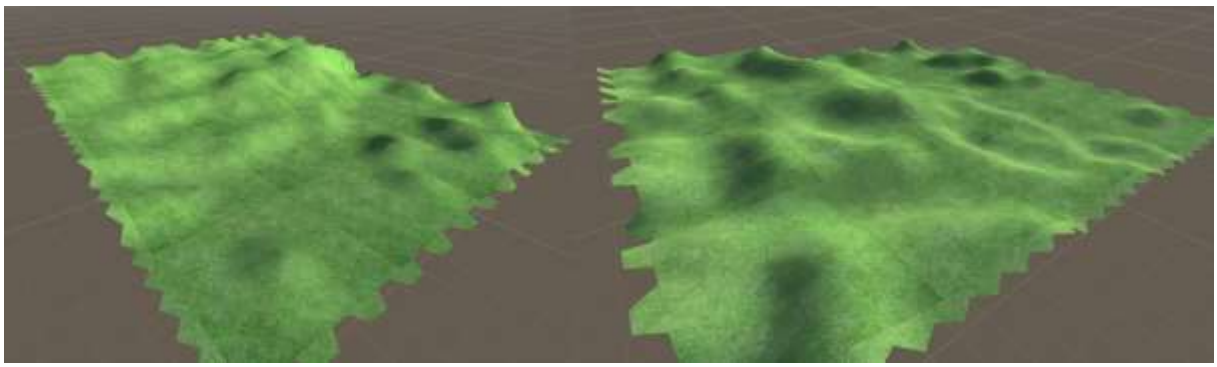

Figure 6. The “Hillside” map from different view angles

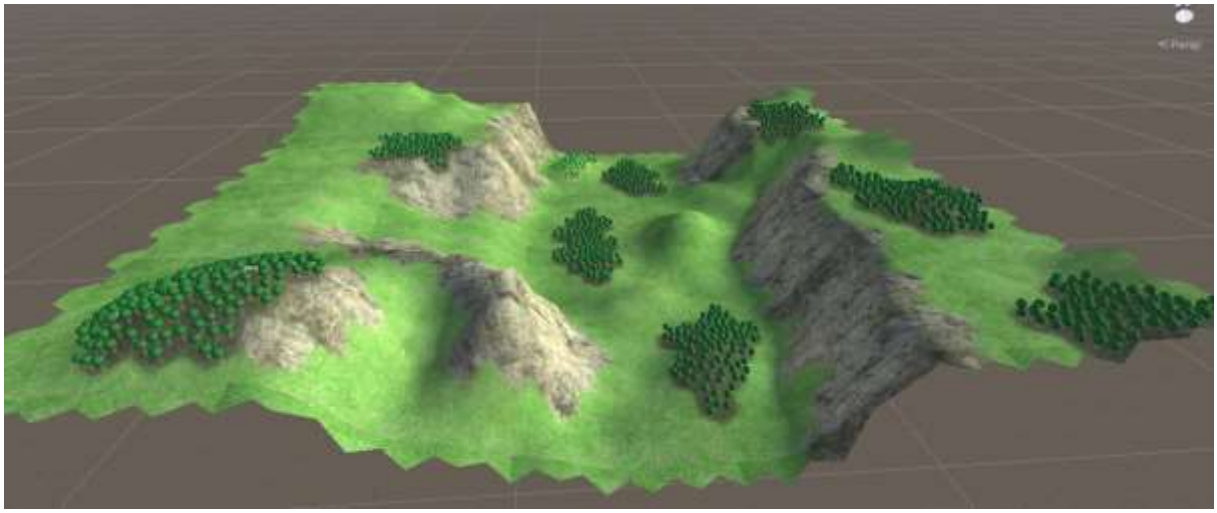

Figure 7. The “Canyon” map

Conclusions

This method will help those who need to generate random maps without modeling and texturing each tile, by using this method it will be faster to implement different types of biomes.

Bibliography

1. GLSL shader language documentation https://docs.gl/
2. Unity engine documentation https://docs.unity3d.com/Manual/index.html
3. Poisson disk sampling https://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph07-poissondisk.pdf
4. Procedural mesh generation guide https://www.udemy.com/course/coding-in-unity-procedural-mesh-generation/