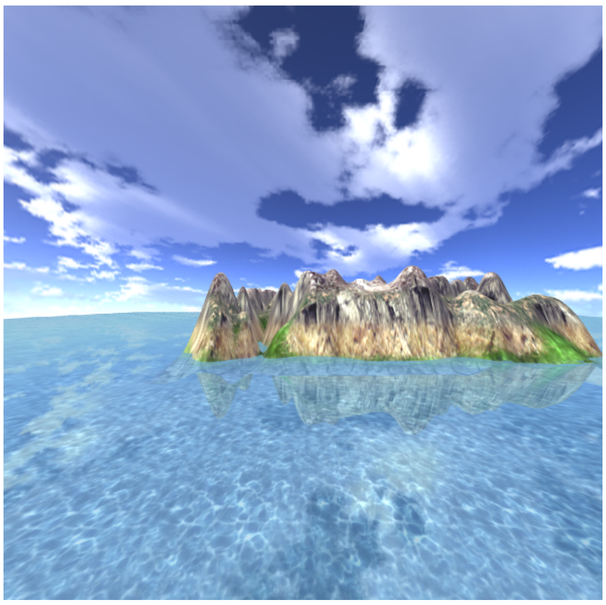
Terrain Engine

Author: PanChuanyu From Department of Physics 2018/12/19



Introduction

> OpenGL provides convenient ways to draw 3D graphics using computer. In this work, I present a method using merely 8 pictures to build a virtual world including sky, dynamic sea water and a detailed island with a lake in it. To make it true-to-life, I make some progress on advanced rebuilding techniques.

Mountain Rebuild

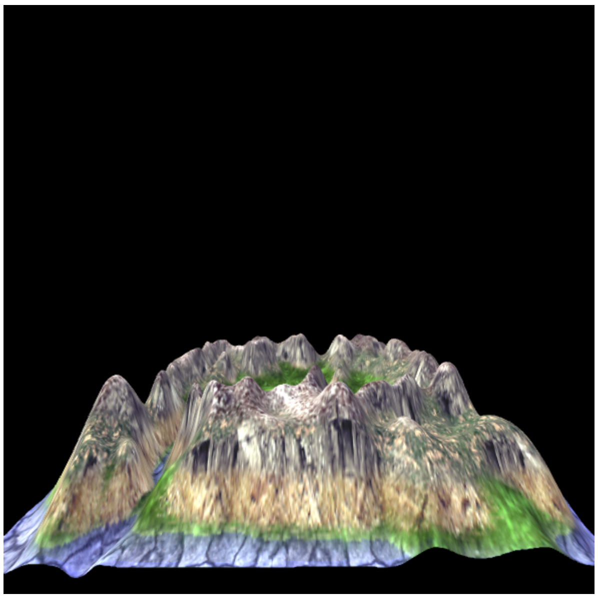
Starting from the depth picture, cv::Mat is used to load the height map and cv::mat.at<u,v> is used to detect height information. Then I put the texture on my model. All information are stored in mountain class.

It needs to be emphasized that the cv::imread function needs to be set as:

```
cv::Mat img2 = cv::imread(detailfilename,CV_BGR2GRAY);
```

In purpose of accurately loading the "gray scale picture", the loading process should be "per pixel".

The result is shown in the following picture:



Rock Textures on mountain

To add some details, I mix rock textures with mountain texture at the rate of :

```
vec4 Colorm = texture(mountTexture, TexCoord);
vec4 Colorr = texture(rockTexture, TexCoord * 8);
Color = Colorm * 1.2 + Colorr * 0.2;
```

Actually this mixing work doesn't turn out well, so I adjust the rock textures' contrast ratio using opencv to make it clearer.

Water wave

The water textures are simply attached on the sea level surface. To make it dynamic, I add a time term on both x-y direction and z direction. The x-y time term controls the translation while the z time term controls the undulation.

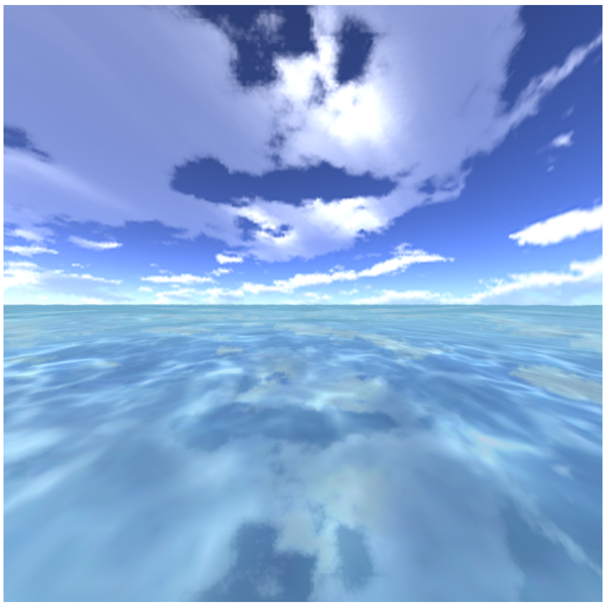
The wave function is:

```
Position.z = Position.z - 3*sin(15*Time+20*Position.x + 20*Position.y)-3*sin(15*Time-
20*Position.x + 20*Position.y) - 3
```

which stands the wave function:

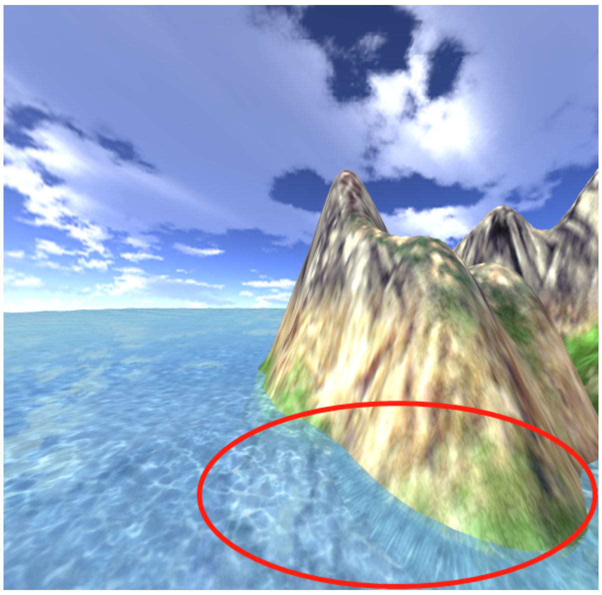
$$z = A_0 \sin(\omega t + kx) + z_0$$

The wave is shown in the following picture, the shadings on distant waves are obvious.



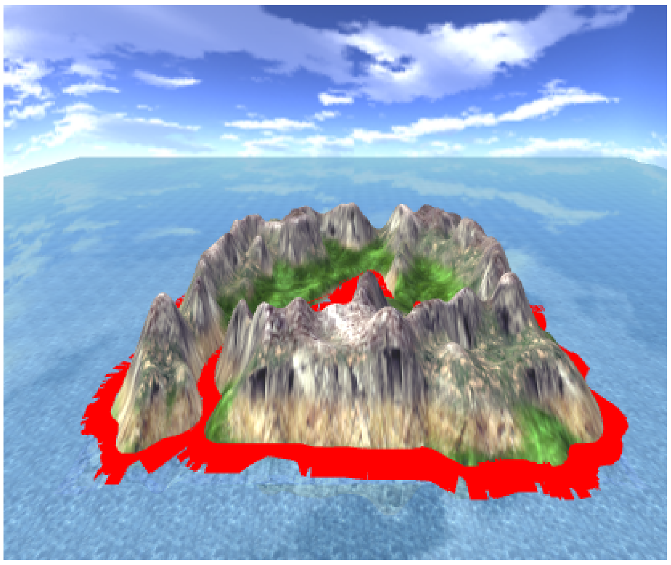
Wave details on the edge

This is a way to realize the interaction between water and mountain edge. To our experience, water will attached on the ground and flow slower than the wave. So I added this kind of "attaching water" to realize the interaction.



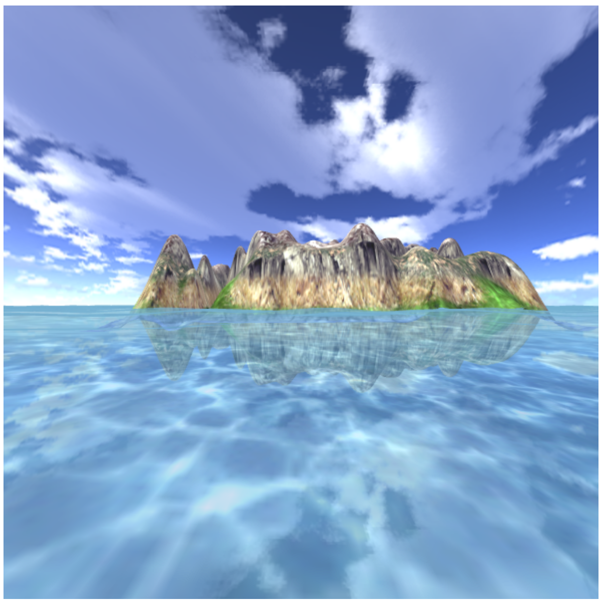
This is realized by finding the silhouette of the mountain and reattaching the texture. A sort() function is used to find related points extracted from the silhouette.

The most important part is to find the points around sea level on the mountain. Dense point selection can help find them. The silhouette is shown as the red part of the following picture:



Reflection

No function can be found in openGL to create the reflection shadings. So I use a little trick to create it, reversing it. By mirroring the model to z<0 plane, the reflection can be found "through" the water.



limitation

We cannot retain both the edge and the "reverse model". The edge will be shown strangely in the water.

Since we can't simply just reverse the mountain, which will lead to the water interaction error, the limit preserved. Adjusting the transparency of the edge may solve this problem.