Final Project of Computer Graphic

招蕴豪 09388327
吴　垚 09388302
陈跃群 09399325

In this project, we create an ocean view in which we made every effort to sculpture a real and believable scene. In the very next paragraphs, I will explain how we meet each of the requirements.

⇝ Modeling Requirements:

1. The scene is consisted of four main parts, namely the terrain, the ocean, the sky and the far away castle. Though we can absolutely add some more interesting things in it such as the sea gulls or sail boats, we think all we had now are already necessarily expressed the quietness and softness of the ocean with kind and warm breeze. So we think the scene we created is complete.

2. Nealy all the objects in the scene is created on our own except the castle which is loaded with a given .obj file. I will simply describe the algorithm we used to create the fantastic terrain and ocean.

    1) Terrain:
    The terrain is generated with a fractal algorithm called `Diamond-Square`. The algorithm begins with a square grid with length of the power of 2. Then it iterates the grids in a special way to generate random heights of the grid and at the same time conserves the smoothness of the heights. This is why the terrain appears so vivid and real.
    2) Ocean:
    The model of the ocean is just a square grid. The waving of the ocean is simulated based on the famous fluid equation Navier-Stokes, but of course, in its simpler form. The initial function is just generated with the `Diamond-Square` algorithm. Its such a waste no to using it :-P. In the waving part, I just use a Laplace-like functor to iterate. For god's sake, it comes out pretty well.

3. The far far away castle is loaded with a .obj file which is provided in the previous time of the course. Actually, this .obj is just added to meet the requirement of the project. We don't see any loss in performance of the scene if it doesn't exist.

4. The objects in the scene is completely modeled and scaled on our own.

5. In fact, all the objects rendered on the screen are textured. We once had a problem dealing with the texture loading in our project that those textures affects each other. We don't know how to separately operate their effect so we map each of the object a texture. The terrain is mapped with a vegetation picture grid-wise. The ocean is simply mapped with a white square picture. Its fantastic effect is achieved by

specifying the vertex normal of each vertex in the ocean grid. The sky is undoubtedly mapped with a sky picture.

6. The scene has one diffuse light source and a specular light source. Without their aides, the ocean have no chance interplaying with the light.

7. Since the terrain and the ocean are both implemented based on the grid model, it is so easy to switch them between filled and wired mode.

⇝ Control and Animation Requirements

1. There are lots of keys defined to tour the viewer around the scene. All of them are listed below:

   - `a : move left`
   - `d : move right`
   - `r : move up`
   - `f : move down`
   - `w : move forward`
   - `s : move backward`
   - `h : turn left`
   - `l : turn right`
   - `j : look up`
   - `k : look down`
   - `q : rotate through z axis`
   - `e : rotate through z axis`

⇝ Responsibilities of members

招蕴豪(33.3%)

1. Responsible for the main structure of the whole program.
2. Terrain and ocean algorithm design and implementation.
3. Giving rendering advises and implement a version of the theme.

吴垚(33.3%)

1. Responsible for the main test task.
2. Ocean rendering implementation and debug.
3. Giving rendering advises and implement a version of the theme.

陈跃群(33.3%)

1. Responsible for the .bmp files reading and .obj files loading.
2. Terrain rendering implementation and debug.
3. Giving rendering advises.

⇝ File specification:

At first glance, the file structure maybe a little tricky, but I will explain it below.

```
├──────┬──── camera
│      │       ├──────── camera.h
│      │       └──────── camera.cpp      # The implementation of class Camera.
│      ├──── lib
│      │       ├──────── bmpReader.h
│      │       ├──────── bmpReader.cpp   # Useful functions to read .bmp files
│      │       ├──────── vector.h
│      │       ├──────── vector.cpp      # Aide vector class, both in 2D and 3D
│      │       ├──────── mathaux.h
│      │       └──────── mathaux.cpp     # Auxiliary math functions.
│      │
│      ├──── sky.h
│      ├──── sky.cpp                     # Sky implementation.
│      ├──── water.h
│      ├──── water.cpp                   # Water implementation.
│      ├──── terrain.h
│      ├──── terrain.cpp                 # Terrain implementation.
│      │
│      ├──── const.h                     # Defines constants in the scene.
│      ├──── setup.h                     # Setups the scene and loads the textures.
│      ├──── main.h                      # The basic logic of the whole project.
│      │
│      ├──── textures                    # Textures needed for the program.
│      └──── obj                         # .obj files needed for the program.
```

⇝ Running guidance:

The program is developed in Ubuntu 11.04 with vim and version controlled with git. So the header file of GL library maybe a little different with those in windows. If you want to compile it in windows, you are definitely got to modify the headers path. But you are compiling it in linux systems, you can simply run the script `make.sh`.

The executable file for linux is generated which is named `terrain`. There is also a window version call `terrain_windows`. Fill free to enjoy the program.