HUYNH Cong Lap – P1419778 – TRAVAIL SEUL

The demo video is recorded by a bad software and a low FPS (15 FPS), so the quality is not good.
Some frames are broken and delay sometimes.
The project is very smooth at real-time.

Part 1: Préparation

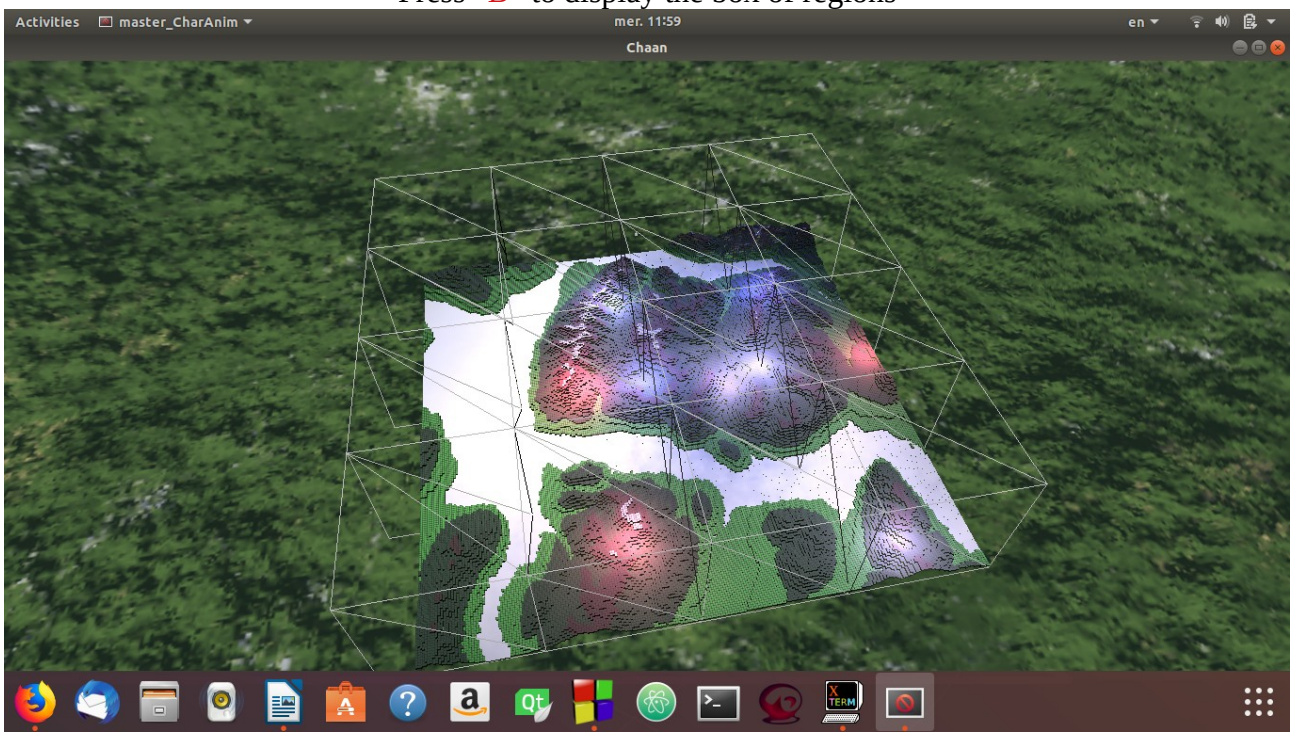
Solution used:
glBindVertexArray()
  pour chaque region de la scene
    si la region est visible
      glDrawInstancedBaseInstance()

Part 2: Visibility

A Terrain divided in cubes (Regions): with glDrawInstancedBaseInstance()

Class Plane: Stimulate a plan by a normal vector and three points.
Class: Frustum → Stimulate the Frustum of camera. Extracting 6 planes of the Camera's frustum
Class: Box → Region contain 64x64

Press "B" to display the box of regions

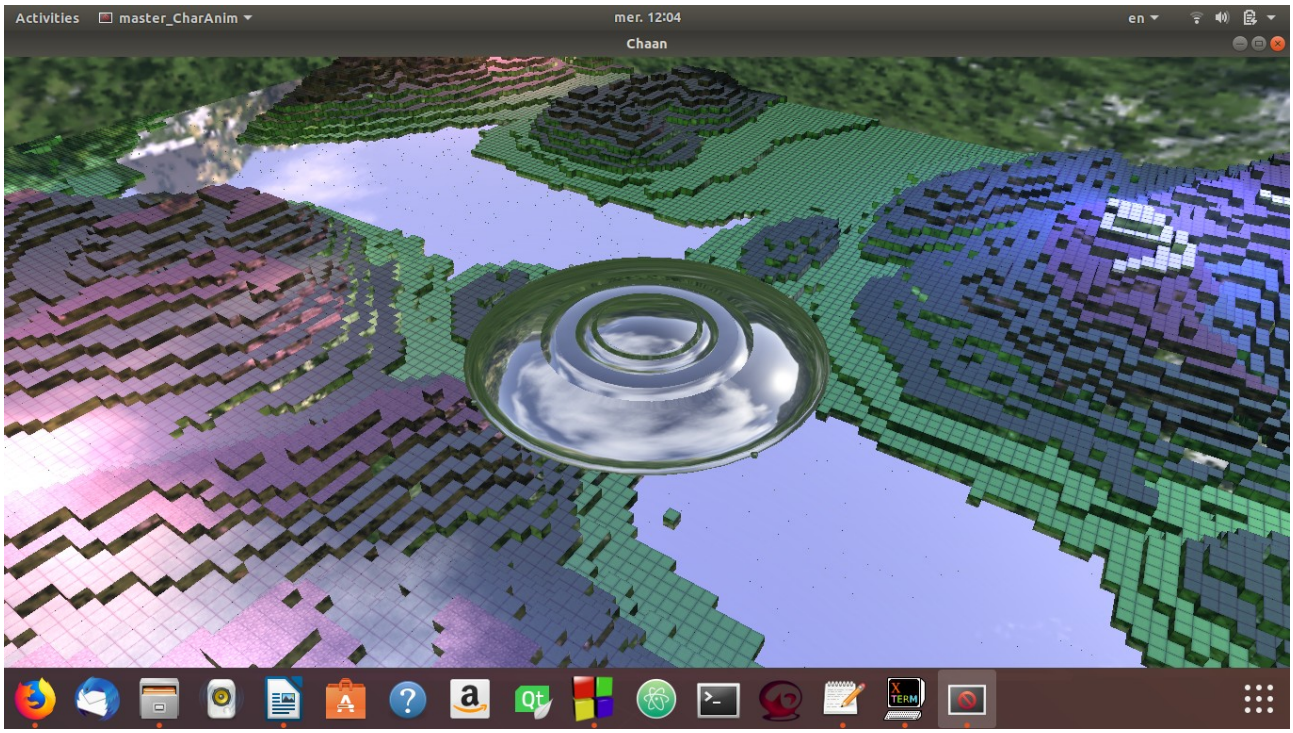


VISIBILITY TEST :
int Frustum::boxInFrustum(Box b)
Return 3 values : **Inside, Outside, Intersect**
We display Intersect and Inside Box with Frustum

Another Test of visibility is in the Shader, Test if the fragment is out/in of the screen

In = -1 < x < 1 && -1 < y < 1 (repere d'image)
Discard out of screen pixels.

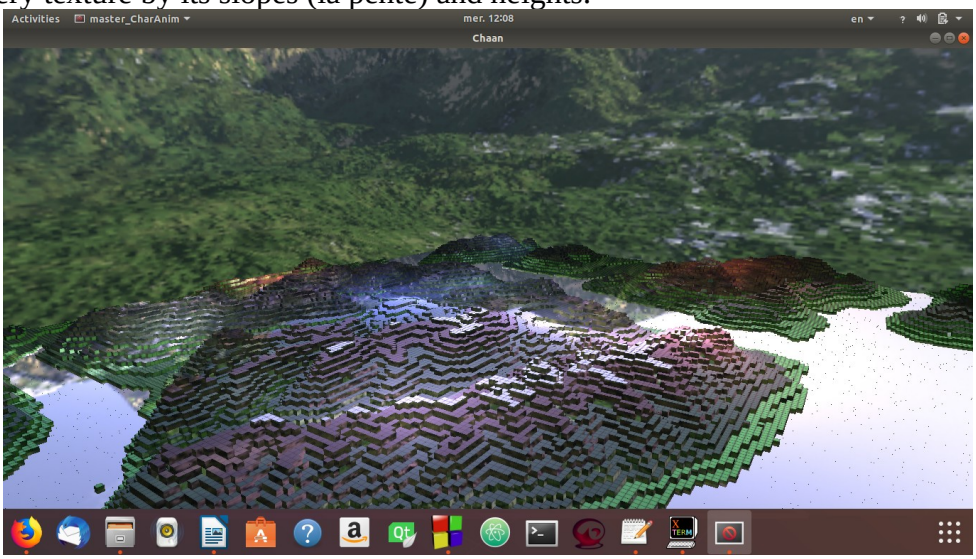
Bonus: I modified the class Orbiter to free the camera control . Add a Class Spaceship as a character to move around the map. This spaceship using a Reflect Shader (Environment map). Press "**M**" move forward the spaceship and right-click and drag the mouse to rotate the direction like in video game.



Class Spaceship

Others optimizations: Enable Depth Testing, Enable Culling Testing

Part 3: plusieurs matières
Using many textures from package Kudocraft
I display every texture by its slopes (la pente) and heights.



Part4: Les ombres

Follow this tutorial: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

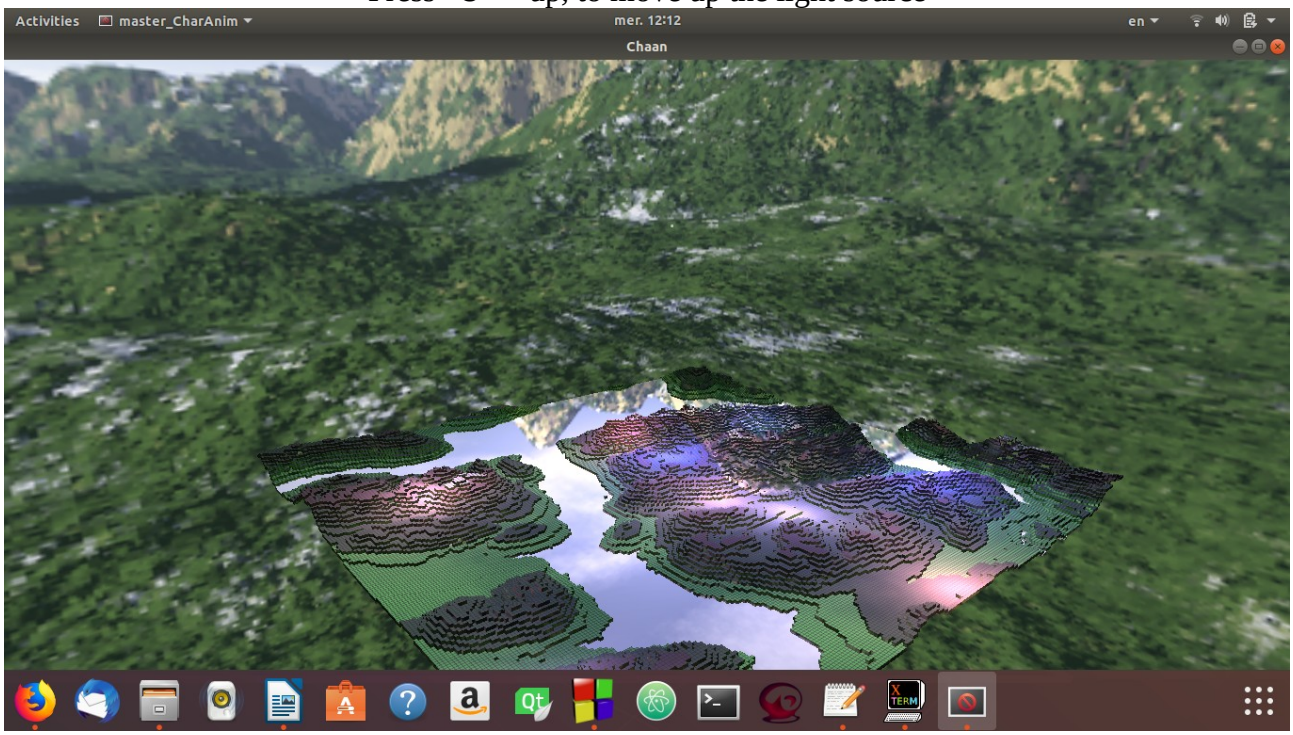
Class: ShadowMap.Cpp

Using a **Frambuffer** to contain the depth map. Write the zbuffer in a texture.

Before rendering the real scene, i render the terrain with a simple Shaders to get depth information from light source posititon. Then render the second time the real screen from the camera, using the depth map to calculate the shadow.

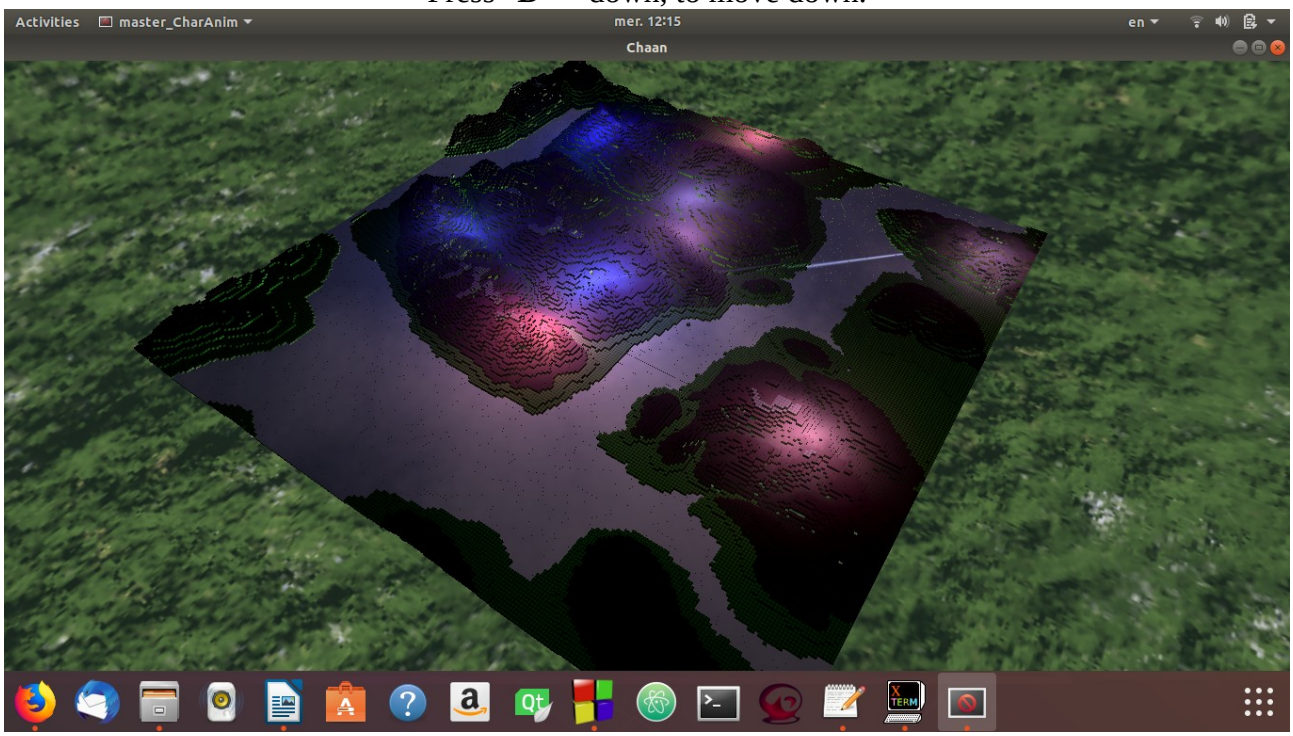
To see the change of shadow in the whole terrain

Press "**U**" = up, to move up the light source



Light source UP

Press "**D**" = down, to move down.



Light source DOWN

Add method :

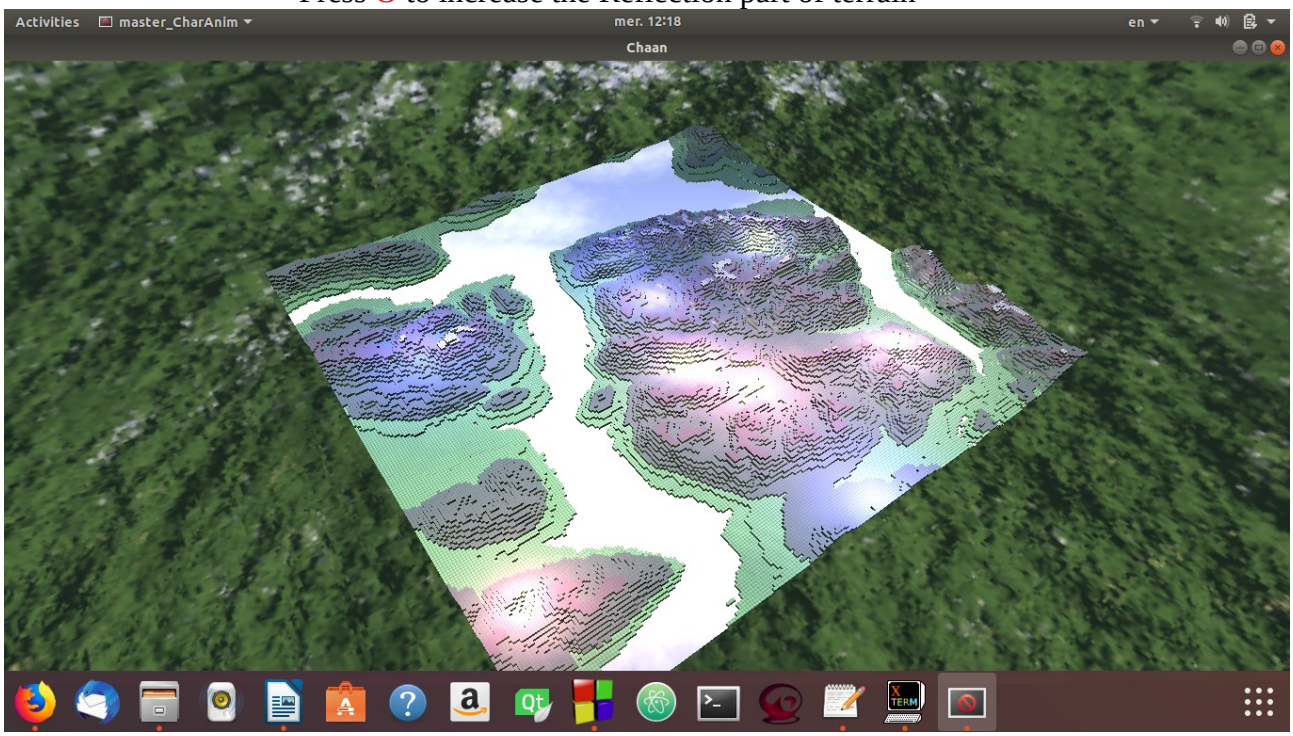
Transform Ortho(const float left, const float right, const float bottom, const float top, const float znear, const float zfar) to calculate the shadow.

Calculate the shadow value in terrain_shader.glsl

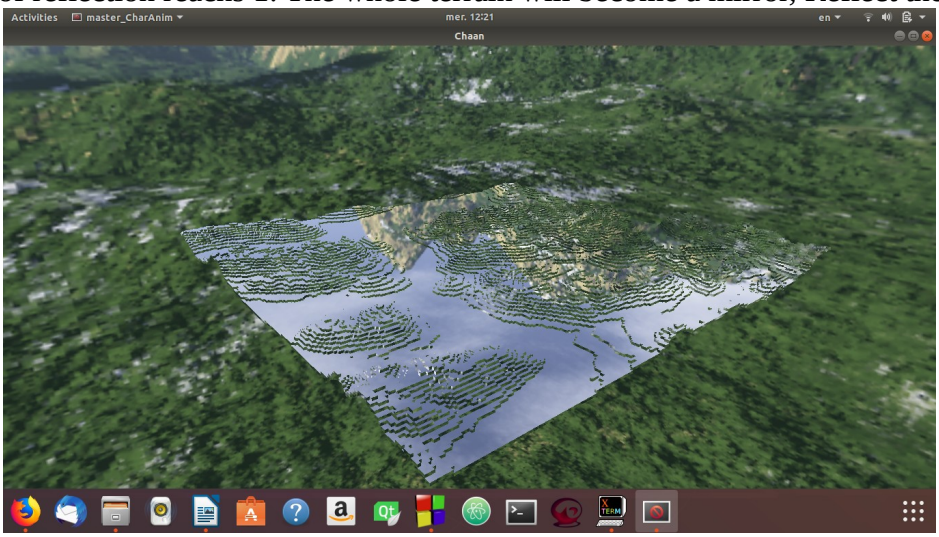
Bonus:
A Cube Map.

Using **Reflection Shader** for the Terrain, Mix the Reflection Color and the Real Color in the shaders
Press “O” and “P” to change the weight of mix.

Press O to increase the Reflection part of terrain

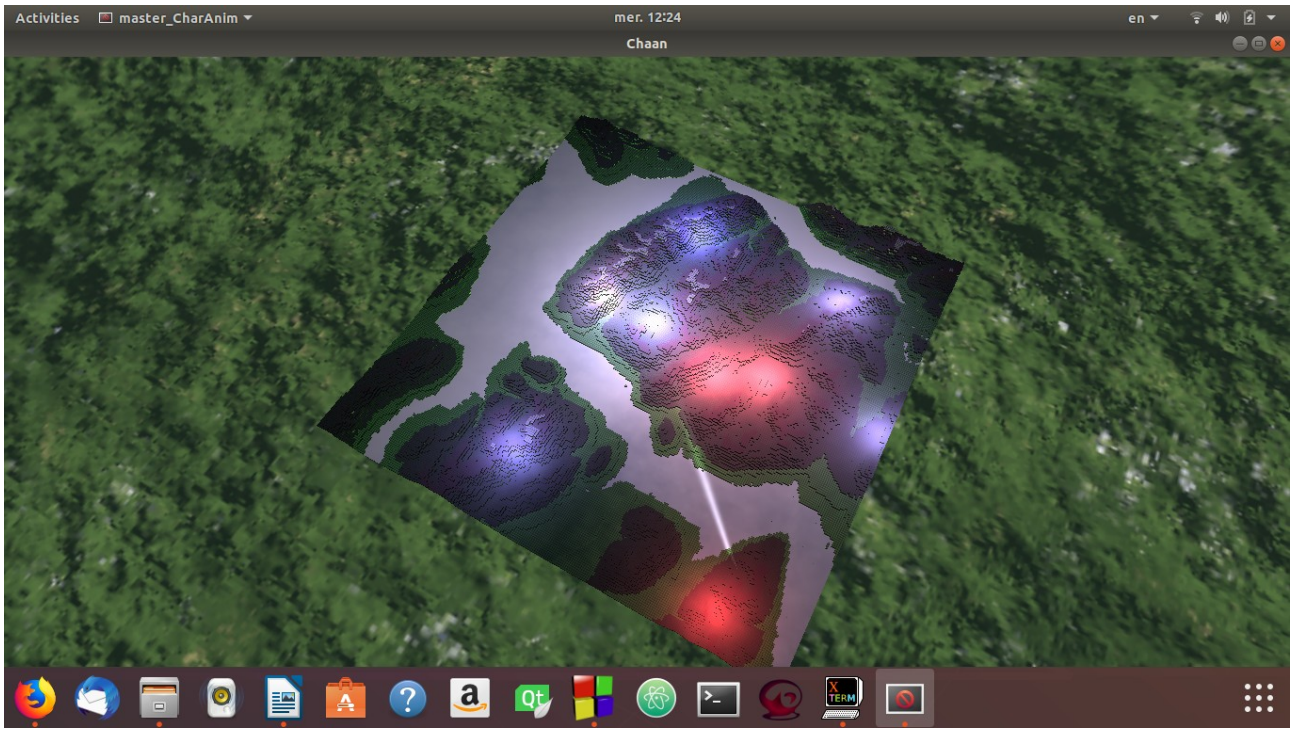


If the weight of reflection reachs 1: The whole terrain will become a mirror, Reflect the cubemap.



Completely Reflect

Add 10 differents light sources, changing colors over time, changing also the material of color (ambient, diffuse, specular), the colors of fragments are affected by these light source.



The Spaceship is also a moving light source.

Shaders used: in folder Tutos

terrain_shader.glsl : main shader for terrains.
simple_shader.glsl: a simple shader to render terrain to get zBuffer for shadow map.
cubemap.glsl: a shader to create a Skybox
spaceship.glsl: reflect shader for space ship

If there is a compile error:
try to change include line : #include "/usr/include/GL/glew.h" by your path of glew.h
In File : Shadow_map.h

Video demo on youtube: /watch?v=AjCQkg8_x8U&feature=youtu.be
Github: https://github.com/HuynhCongLap/3DWorld