Flight Simulation (飞行模拟)

# Introduction - Requirement finish state

* OpenGL+GLSL(着色器)+(open)GLUT
  + 2VS+2FS
* 地形绘制：任选一个地形模型进行绘制，并用给定的纹理图像进行纹理映射。
* 飞行模拟：采用飞行员视图在地形场景中漫游。
* 物体绘制：放置了所有给出的样例模型，材料属性可以通过程序配置。
* 光照：至少两个光源。
  + 计算了面法线与顶点法线，但现仅实现了顶点法线的平滑明暗效果。面法线因为需要重复顶点信息暂未实现。
  + 两个光源，其中一个为平行光；另一个可以设置成飞机上的可交互的探照灯。聚光灯效果目前是全方向的点光源，聚光灯效果已实现，单方向控制上存在BUG，禁用了，实现请参考Fragement Shader 源码。
* 雾：支持全局雾气特效。体积雾气与透明物体渲染相同，较为复杂，未做实现。
* 模型的数据结构、效率和交互方面的考虑。
  + 所有模型数据均使用Buffer，减少了CPU/GPU 之间传输数据的代价。
  + 面向对象式的程序设计，接口设计还算不错^\_^
  + 经典的游戏任务控制方式(W/A/S/D/鼠标按下控制视角方向)

碰撞检测：未有时间完成，除地形外的碰撞模型已生成。

\* GLUT is extremely outdated, there is another version using pure Win32 API, all these source code is hosted on Github.

\* I used some C++ 11 features inside the source, so you need an C++ 11 compiler (VS 11+)

\* There is no specify in the requirement about "Math Library", I use <DirectMath> instead <glm.h> for better performance and a C++ friendly style.

Contents

[Introduction - Requirement finish state 1](#_Toc386485434)

[Model requirement 2](#_Toc386485435)

[Usage : 2](#_Toc386485436)

[Camera operation: 2](#_Toc386485437)

[Technique used or planed: 3](#_Toc386485438)

[Vertex Attributes / Index(Elements) Array Buffer 3](#_Toc386485439)

[Shading: 3](#_Toc386485440)

[Normal computing: 3](#_Toc386485441)

[Culling 3](#_Toc386485442)

[Terrain Rendering 3](#_Toc386485443)

[Sky-Dome 4](#_Toc386485444)

[Collision & Physics: 4](#_Toc386485445)

[User Interaction: 4](#_Toc386485446)

[stuff not changed as previous readme 4](#_Toc386485447)

[SSE2 Acceleration for Vector types 4](#_Toc386485448)

[Redraw only when necessary 4](#_Toc386485449)

[Lib Referenced: 4](#_Toc386485450)

[Referenced: 4](#_Toc386485451)

# Model requirement

put the corresponding obj file into app folder, should look like

mesh\

mesh\camel.obj"

mesh\cow.obj

mesh\enterprise.obj

mesh\klingon.obj

mesh\rabbit.obj

mesh\voyager.obj

mesh\puget\_tex.obj

texture\

texture\puget\_tex.ppm

# Usage :

## Camera operation:

Hold <Left Mouse Button> to rotate the camera in Pitch & Yaw

Press W/A/S/D to move the camera

Hold Z/X to move the camera up/down

# Technique used or planed:

## Vertex Attributes / Index(Elements) Array Buffer

It is well-known that memory access and data transferring between CPU/GPU is extremely time consuming, instead of the flexible glVertex3f like functions, using Array Buffer to send the data into GPU's graphic memory is a much better choice for performance.

We used this technique for every model, all the model's data is copied to GPU when the model is loading, makes the frame-rendering task much more fast.

## Shading:

The customized shader support one Ambient Light, one Directional Light, and one Spot Light, the "flashlight" style light is still in WIP, I have some trouble with it's direction. The Global Fog effect is also support, also the texture effect.

This is done by writing 4 shader to handle the different model type. it's automatic selected by the function for the given model.

Specular is not added yet, because I have a bug with the vertex shader about the light position in view space computing, well, I guess it's there.

This is also a bug affect the Spot Light effect, result in an strange lighting, I guess it's the same one that cause the specular not work.

I do understand having BUGs is so bad, but debugging environment for shaders is just so horrible, and the time is too limited, forgive me. I'll fix it in later release/assignment.

## Normal computing:

The given data doesn’t contains any normal data, but it's computable, I calculate the face surface first, and than average all the adjacent faces’ normal to get the Vertex Normal. Using an weight average is a better idea, like co-tangent weight[ref], but for the time limit, simply average here.

## Culling

This is not implemented yet, but it is a must-add technique.

In this version , we have extract the bounding-box for all the model(object) we are rendering, the next step is to perform an collision check with the camera's view frustum.

## Terrain Rendering

Terrain is an special model to render in the game world, because it's scale.

It's typically structured to answer the query about the height/grad in a specified point.

The typical technique to render Terrain is to divide it into patches, and determinate the LOD and cull for each patches, than render the Corresponding LOD patches into the screen.

But in this assignment, the terrain model is given by an OBJ file, this is defiantly not good. Structuralize the terrain data into the desired form need too many efforts , but the time is not enough...

PS. the texture for this terrain is really horrible...

## Sky-Dome

An big inner-sphere move with the camera to render the sky-effect in distance, to render this, we need a cubic texture, or six connected texture to Express this paranorma.

## Collision & Physics:

Basically the same as culling, but the differences is the is more about the scenario -- we want to simulate the gravity, the collisions, to render the game world as real as we can.

This is typically finished by 3-rd party engine, such Havok or PhysicsX, but in this assignment we can simulate the flight's air dynamic by a simplified model. Apply to other model's movement too.

## User Interaction:

Planned to add function is to create and move object inside the game world, not finished by time.

## stuff not changed as previous readme

### SSE2 Acceleration for Vector types

This is finished by using DirectX Math library, this inline C++ lab provide extreme performance by using SSE2/ARM\_NEON instructions.

### Redraw only when necessary

The redraw is only called when view changed ,not redraw the scene for every frame, this could not improve the frame rate for interactive and improve the user experience, but could save the computational power and battery life.

# Lib Referenced:

openGL/GLU

DirectX Math (Included in Windows Kit, version 8, compile required Visual C++ 11/12(2012/2013)

DirectX Toolkit :: SimpleMath (http://directxtk.codeplex.com/, it's an inline lib, and the header is included inside the package.)

STL (C++11 support required)

\*All libs above should ne installed along with Visual Studio 2012/2013 (Desktop Express)

GLUT/GLEW

# Referenced:

http://www.opengl-tutorial.org/