Shadow Caster Culling 程序文档

简介：

在*实时渲染*(Real-Time rendering)中，*阴影映射*(Shadow mapping)过程是限制渲染速度的一个非常重要的环节，因为在每一帧渲染之前，我们还需要对整个场景进行一次*深度缓冲*(depth buffer)的渲染(shadow pass)，用来检测物体是否在阴影中。

本工具的作用是，针对该过程，对场景中的物体进行*离线*（off-line)的预处理，将那些在渲染中投射的阴影没有意义的物体标记出来，这样，在渲染过程中，就可以通过读取标记，在shadow pass过程中忽略这些物体，进而对渲染过程进行加速。

原理：

之前提到有些物体在阴影映射中，它所投射的阴影是“*无意义*”的，这种“*无意义*”主要分以下两种情况：

1. 该物体完全被其他物体所遮挡：

在这种情况下，根据shadow mapping的原理，任何处于该物体阴影之下的物体同时也会完全处于其他物体的阴影之下，因此在shadow pass这种物体可以完全忽略。

2. 该物体所投射的阴影中并不存在其他物体：

在这种情况下，在渲染中，并不会有任何物体的颜色计算需要该物体所投射的阴影，因此该阴影也是没有意义的。（这种情况最多发生在地形(Terrain)上）

程序原理：

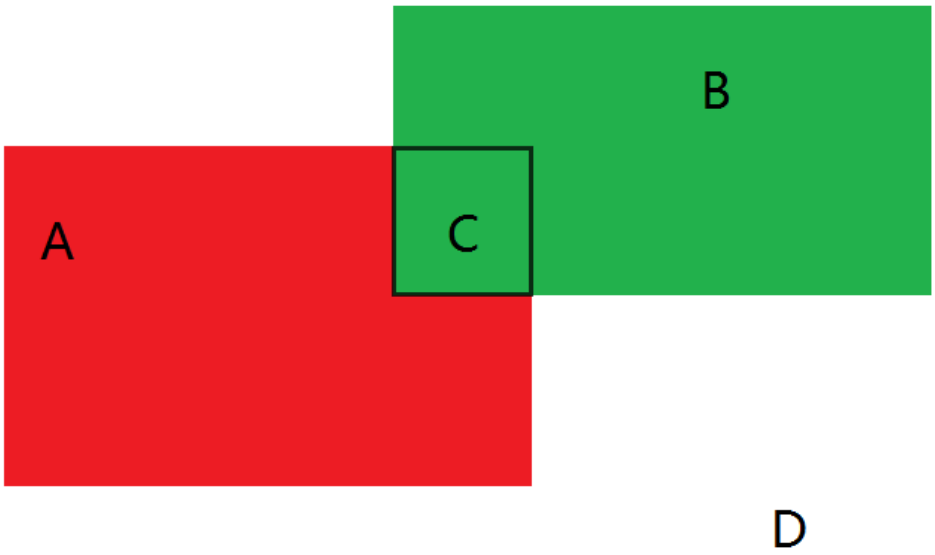
针对上述两种情况，在程序中分别通过以下方法来实现：

1. 该物体完全被其他物体所遮挡：

这种情况可以通过OpenGL的Occlusion Query实现检测。首先，将整个场景渲染一遍，记录每个物体的*深度*(Depth)信息；然后在第二次渲染中，对每一个物体的所有fragment根据之前渲染得到的Depth Buffer中的Depth信息进行Occlusion Query，若返回的结果为0，则表示这个物体的全部fragments均被遮挡，则这个物体是被遮挡的，需要被剔除，否则，这个物体没有被遮挡。

2. 该物体所投射的阴影中并不存在其他物体：

这种情况可以通过渲染中的*模板检测*(Stencil Test)实现检测。首先，依然是将整个场景渲染一遍，对任一物体，均将其对应的fragments的*模板缓冲*(Stencil Buffer)加1。这样，在第一遍渲染结束时，对于任一fragment，若其Stencil Buffer的值大于1，则说明在这个fragment上有多个物体重叠(如下图中C区域所示)；若等于1，则说明在这个fragment上仅有一个物体存在(如下图中A，B区域所示)；若等于0(如下图中D区域所示)，则说明该区域没有物体存在。因此，若该fragment背后存在物体，则其Stencil Buffer的值一定要大于1。



综合1,2，则对于任一fragment，必须同时满足Stencil Buffer的值大于1，并同时通过Occlusion Query才表示其通过检测，而对于任一物体，若存在通过检测的fragment，则其投射阴影是有意义的。

注：因为Occlusion Query是通过Depth Test实现的，而Stencil Test存在于Depth Test之前，因此不通过Stencil Test的fragment也不会通过Occlusion Query，因此对于任一fragment，可以在进行Occlusion Query之前进行Stencil Test，则OQ的结果就是两者的与运算结果。

特殊情况：

1. 穿过地形的物体：

   由于场景构建时会出现一些物体会有一部分穿过地形，这样在检测时，地形下的那部分会被当做有物体存在而检测出地形的投影是有意义的，但实际上我们并不希望这种情况发生，因此我们在渲染之前预先将地形的高度图(Height Map)渲染并保存在Texture中，然后在进行Stencil Buffer的写入时，将高度低于地形的忽略掉。

2. 存在**Alpha Test**的物体**(**例如树叶**)**：

   这类物体在检测中，不作为遮挡物，但作为查询物，意思就是它们不遮挡其他物体，但依然可以被算作被遮挡的物体。因此在检测的时候，我们需要做做两遍Stencil/Depth Buffer的渲染和两遍Query渲染过程，一遍忽略有Alpha Test的物体，另一遍则将该物体作为正常物体进行考虑，最后将两次的结果做或运算得出最终结果。

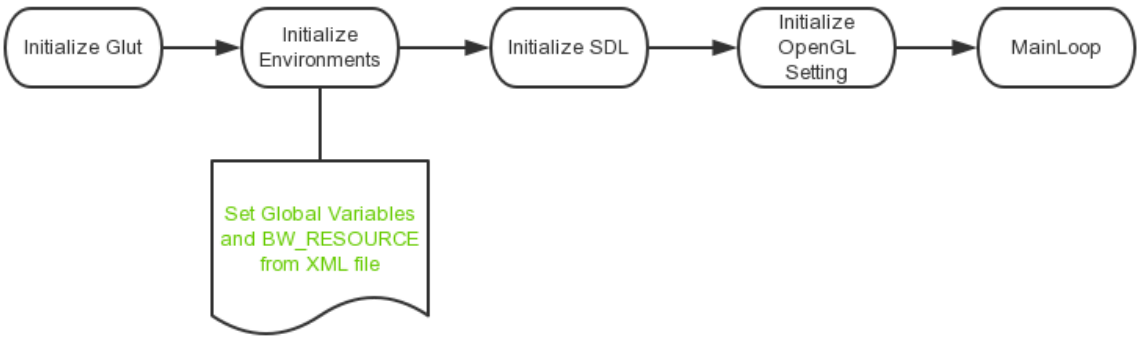
3. 邻近**chunks**的影响：

   因为场景资源是通过区块(Chunk)进行管理的，对一个chunk中物体进行处理时，它的投影不仅仅受到自身chunk中的物体影响，同时也会受到周围邻近的区块中物体的影响。因此我们的做法是，在进行Stencil/Depth buffer的渲染时，我们将中心区块邻近的区块一并进行渲染，但在Occlusion Query时，我们只渲染中心区块以得到最终结果。邻近chunks的数量可以在配置文件中修改配置。

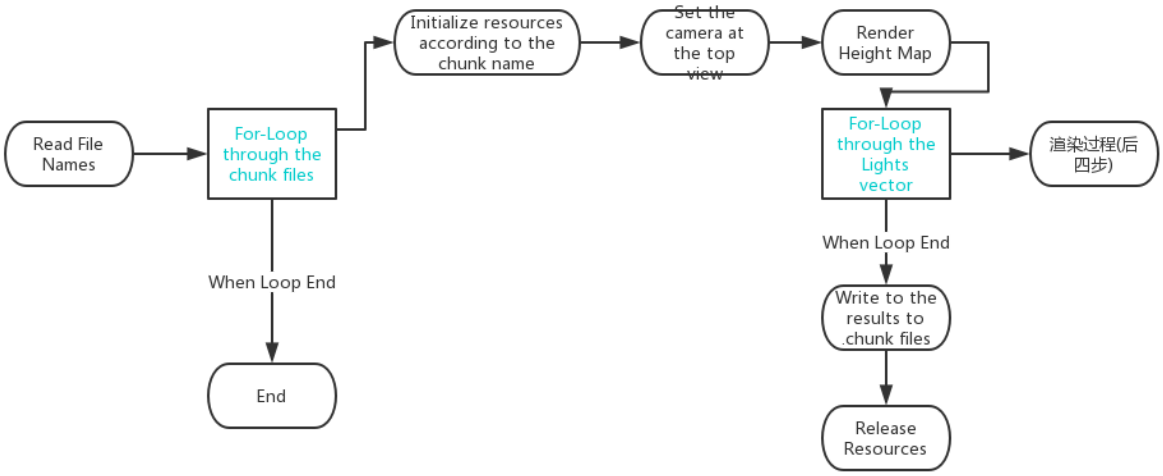
程序设计：

1. 渲染过程：(五遍渲染)

2. **Main** 函数流程：



3. **MainLoop** 流程：



模块介绍：

1. **main** 函数 (main.cpp)：
    - 全局变量定义：

```
//Local Settings:
GLuint program_test,program_Stencil,program_query,program_show,program_heightpass;
//The name of the center chunk(change during the loop)
std::string chunk_Name;
//dynamic VAO and VBO vectors of chunks' models(primitives)
std::vector<GLuint> vao_vector;
std::vector<GLuint> vbo_vector;
std::vector<GLuint> ibo_vector;
std::vector<uint> index_size_vector;
std::vector<GLuint> nbo_vector;
std::vector<glm::mat4> model_matrix_vector;
std::vector<bool> Is_alpha_test_vector;
std::vector<bool> mirror_flag_vector;
//The number of the center Chunk's Prims
int Num_Origin = 0;
//Terrain VAO, IBO, VBO, NBO
std::vector<GLuint> terrain_vao_vector;
std::vector<GLuint> terrain_vbo_vector;
std::vector<GLuint> terrain_ibo_vector;
std::vector<GLuint> terrain_nbo_vector;
std::vector<uint> terrain_index_size_vector;
std::vector<glm::mat4> terrain_model_matrix_vector;
//the max and min value of Y of scene(for fragment culling in the Stencil
Pass(discard))
double maxY = -2000.0, minY = 2000.0;
// Heights Texture(Heights Map)
GLuint HeightsTexture;
// Heights FrameBuffer
GLuint HeightPassFBO = 0;
// Center Chunk Origin
glm::vec3 chunk_Origin;
// Camera
Camera cam;
// Occlusion Query
std::vector<GLuint> queries;
// Results of one light for one chunk
std::vector<int> res;
//For Writting
// Final Results of multiple lights for one chunk
std::vector<uint32> final_results;
// index for res writting
int Idx = 1;
// Tree Struct of the chunk(for writting)
Node *chunk_Root;
//Global Setting:(don't change with the changes of the chunks/ don't free by
releaseResource)
std::string BWRes_Path;
std::string Universe_Path;
std::vector<Light*> Lights;
float Stencil_Bias;
int screen_width, screen_height;
uint32 Render_Delay;

//Light Projection Matrix(read from the Lights Vector)
```

```
glm::mat4 Light_Projection_Matrix;
//Number of the Nearby chunks
int num_left = 1;
int num_right = 1;
int num_top = 1;
int num_bottom = 1;
//Flag of Center Is Having Terrain:
int IsCenterHavTerrain;
```

- **mainLoop** 函数：如之前流程图所示。
- **main** 函数：如之前流程图所示。

2. **Camera** 模块(Camera.h, Camera.cpp)：相机操作相关的函数(Class Camera)。
   - 变量 Vriables:

     `glm::vec3 position`：相机位置

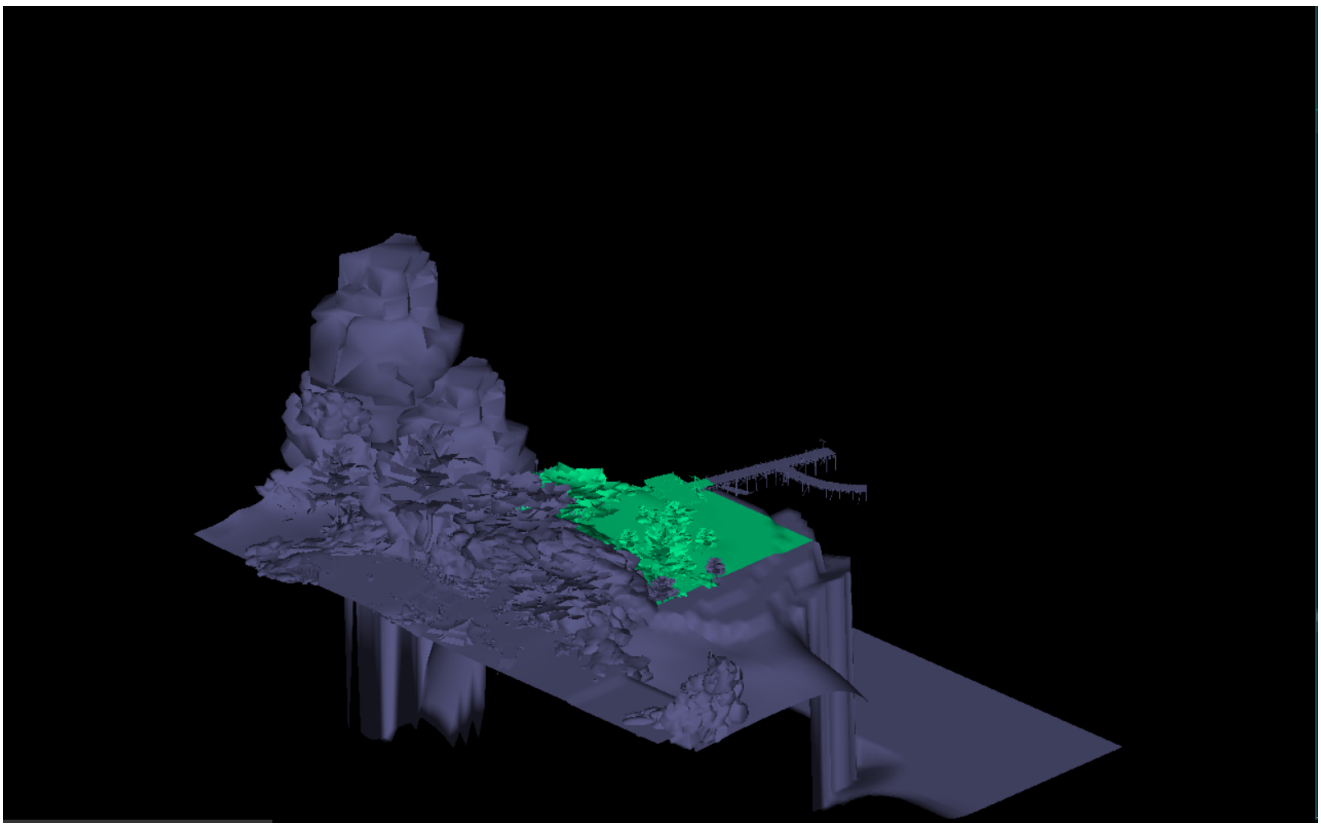
     `float horizontal_Angle, vertical_Angle`：相机转角。`horizontal_Angle = 0`时，面向Z轴正方向；`vertrical_Angle = -PI/2`时，面向y轴负方向。

     `glm::mat4 ViewMatrix, ProjectionMatrix`：View矩阵与Projection矩阵。
   - `void computerMatrices()`：根据当前位置及角度计算View和Projection矩阵。
   - `void computeMatricesFromInputs(SDL_Window *window)`：结合window输入计算View和Projection矩阵。(并不会在本工具中使用到)
   - `void setPosition(glm::vec3 pos){position = pos`：设置position。
   - `void setPosition_Dir(glm::vec3 pos, float horAngle, float verAngle)`：设置position和direction。
   - `void getPosition_Dir(glm::vec3& pos, float &horAngle, float &verAngle)`：获得当前相机的position和direction。
   - `glm::mat4 getViewMatrix()`，`glm::mat4 getProjectionMatrix()`：获得当前相机的View和Projection矩阵。
3. **FBO_Funcs** 模块(FBO_Funcs.h, FBO_Funcs.cpp)：包含FrameBuffer 相关函数。(用于render Height Map)
   - `bool Set_HeightFBO()`：生成并绑定Framebuffer与Texture用于渲染高度图。
4. **File_Funcs** 模块(File_Funcs.h, File_Funcs.cpp)：包含文件名称读取相关函数。
   - `void GetAllFormatFiles(std::string path, std::vector<std::string>& files, std::string format)`：根据目录路径读取该目录下全部的文件名存在files向量中。
5. **Initialization_Funcs** 模块(Initialization_Funcs.h, Initialization_Funcs.cpp)：包含工具的环境初始化相关函数。
   - `bool Initialize_Environments()`：读取配置文件，设置部分全局变量，并初始化资源管理系统。
   - `bool Initialize_OpenGL_Settings()`：初始化OpenGL设置。
6. **Render_Funcs** 模块(Render_Funcs.h, Render_Funcs.cpp)：包含渲染相关函数。
   - `void render_heights(SDL_Window* window)`：从chunk正上方对中心chunk以及邻近chunks的地形进行渲染，将其高度值保存在颜色buffer的G项中，并将渲染结果保存在HeightPassFBO中的HeightTexture中，以供后续渲染使用。
   - `void render(SDL_Window* window, bool WithAlpha)`：根据从配置文件中提取的光源信息(光源位置，光源角度，投影矩阵)对整个场景进行渲染(包括邻近区块)，目的是写Stencil/Depth Buffer以供后续

Occlusion Query Pass使用。其中`bool WithAlpha`标记是否将存在Alpha Test的物体考虑在内。
    - `void occlusion_Query(SDL_Window *window, bool WithAlpha)`：依然根据从配置文件中提取的光源信息，以及Stencil/Depth Buffer中的信息，对中心区块的场景进行渲染并进行Occlusion Query，将结果保存在全局变量res向量中。
    - `void render_scene(SDL_Window* window)`：从相同角度对整个场景进行渲染并展示在窗口中，同时将检测结果用颜色标记出来(灰色：邻近区块；红色：投影无意义；绿色：投影有意义)。
7. **ResMgr_Funcs** 模块(ResMgr.h, ResMgr.cpp)：包含资源管理系统交互的相关函数。
    - `int Read_Chunk(std::string chunk_resource, bool IsCenterChunk = false)`：根据资源管理系统的层次关系，逐层解析，将场景信息读入全局变量中，并将对应的顶点数据和索引信息传输到GPU中，供渲染过程使用。
    - `int Read_Nearby_Chunks(std::string chunk_resource)`：根据从配置文件中提取的邻近区域的发呢为，读取邻近的区块的场景信息。
    - 其他函数：这个模块中包含非常多的资源管理系统交互相关的函数，这些函数均是用于应用在`Read_Chunk`函数中，具体信息可见代码中注释。
8. **Resource_Funcs** 模块(Resource_Funcs.h, Resource_Funcs.cpp)：包含Chunk资源读取相关函数。
    - `bool init_resources()`：根据当前的全局变量`chunk_Name`的值，进行`Read_Chunk`，`Read_Nearby_Chunks`等操作，同时还会设置高度图的FrameBuffer(`Set_HeightFBO`)，以及加载shader文件，初始化Queries缓冲等。
    - `void release_resources()`：释放并删除一切与当前chunk相关的全局变量信息，以供下一个chunk进行初始化。
    - `void free_resources()`：在程序结束时，释放资源。
    - `void TraverseFreeNode(Node* T)`：释放树形结构Node* T。
9. **Shader_Funcs** 模块(Shader_Funcs.h, Shader_Funcs.cpp)：包含Shader相关操作函数。
10. **Struct** 模块(Struct.h)：包含一些结构的定义。
    - `IndexHeader`，`VertexHeader`，`PrimGroup`，`TerrainBlockHeader`，`Node`，`Light`
11. **TransMesh** 模块(TransMesh.h, TransMesh.cpp)：包含顶点信息读取相关函数。
12. **Xml_Funcs** 模块(Xml_Funcs.h, Xml_Funcs.cpp)：包含配置文件(xml格式)的读取相关函数。
    - `bool ReadSettingFile(char* XMLPath)`：读取配置文件，并将信息保存到全局变量中。

结果输出：

运行结果会实时显示在渲染窗口与控制台窗口中。



选择F:\NetEase_Projects\Project1_Shadow_Caster_Culling\6.1_Shadow_Caster_Culling_Beta3.0\Debug\6.1_Shadow_Caster_Cullin...

```
ight No. 1
----------------Finish ffff0000o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0001o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0002o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0003o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0004o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0005o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0006o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish ffff0007o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffff9o.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffffao.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffffbo.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffffco.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffffdo.chunk ----------------
ight No. 0
ight No. 1
----------------Finish fffffffeo.chunk ----------------
```

与此同时，检测结果还会写入到场景文件中的.chunk文件的字段中：

```
<terrain>
  <resource>    ffffffffo.cdata/terrain </resource>
  <shadow>  2   </shadow>
</terrain>
```

同时对于model，还会将结果写入model下的中：

```
<model>
  <resource>    objects/area/st/model/rock.model    </resource>
  <dye>
    <name>  com_tex_shiq01  </name>
    <tint>  huang   </tint>
  </dye>
  <uuid>    6055E290-4CC63BB3-1E14A3AE-0DFBE9A0 </uuid>
  <transform>
    <row0>  -0.991167 0.118751 0.820961 </row0>
    <row1>  -0.662710 0.655975 -0.894993    </row1>
    <row2>  -0.498893 -1.107288 -0.442164   </row2>
    <row3>  46.615074 -96.943344 12.009224  </row3>
  </transform>
  <affectByBlend>   true    </affectByBlend>
  <specPixelPerMeter>   20.000000   </specPixelPerMeter>
  <shadow>  2
    <RenderSet>
      <Geometry>
        <PrimitiveGroup>    2   </PrimitiveGroup>
      </Geometry>
    </RenderSet>
  </shadow>
</model>
```

写入格式为32位的int格式，其中每一位代表从一个光源(Light)出发的检测结果，同时model下字段。在读取检测信息的时候，需要按位读取，其中从右数第一位开始代表光源1到光源N，例如：仅光源01与光源02，若检测结果为光源02为有意义，01无意义，则写入的结果为2，即*0010(前面位数省略)。

而对于写入的结果，如果同一个model存在多个RenderSet或PrimitiveGroup，则结果为其model下所有PrimitiveGroup结果的位或。

```
<model>
  <resource>    objects/area/zw/model/zw_mlqm01_6381.model  </resource>
  <uuid>    BCF2EFBF-46D323BC-204E259B-CB3D47BA </uuid>
  <transform>
    <row0>  -0.327740 -0.027205 0.326609    </row0>
    <row1>  0.000000 0.461894 0.038475  </row1>
    <row2>  -0.327741 0.027205 -0.326609    </row2>
    <row3>  10.023972 -88.602753 57.472660  </row3>
  </transform>
  <affectByBlend>   true    </affectByBlend>
  <dye>
    <name>  Material #35    </name>
    <tint>  bai </tint>
  </dye>
  <shadow>  3
    <RenderSet>
      <Geometry>
        <PrimitiveGroup>    2   </PrimitiveGroup>
        <PrimitiveGroup>    3   </PrimitiveGroup>
      </Geometry>
    </RenderSet>
  </shadow>
</model>
```