雪人3D场景

项目简介

- 使用DirectX 11制作一个雪人3D场景，内容要求如下：
  - 场景由2个雪人，地形，天空盒，做移动圆周运动的立方体组成。
  - 场景中有个立方体，立方体在场景里做圆周运动。场景里有两个雪人，一个站在地上，一个站在立方体上跟随立方体一起运动。
  - 场景中的物体都要有贴图与光照效果。
  - 可以用键盘和鼠标在场景中漫游，可以漫游到立方体上。
  - 雪人必须用球体/圆柱/立方体等基本的模型组装，不能使用现成的模型。
  - 场景中的主体是通过球体/圆柱/立方体等基本的几何体组装的雪人， 周围环境可以用现成的模型点缀
  - 天空盒可以用六面体盒子，跟随相机移动形成天空效果。
  - 漫游控制方式参考CodeSample中的 First Person Shooter Controls 教程。
  - 学有余力可以挑战更高的要求，场景可以更为丰富，如添加树木，房子，雪花等物体。显示效果上添加阴影，多种类型灯光，多层纹理等。
- 开发环境：
  - DirectX 11
  - Visual Studio 2017
- 项目概要：

Final View

- 该项目以DirectXTK 作为辅助库进行开发，实现了全部上述的基本功能。

实现细节

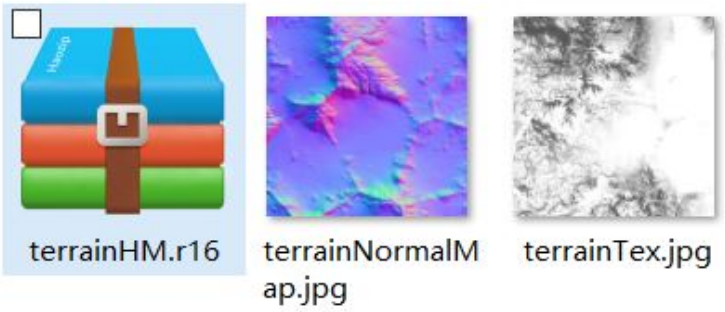
方法细节

- 本项目使用的方法都较为基础，值得讲解的主要有以下几部分：
  - 场景结构组织：

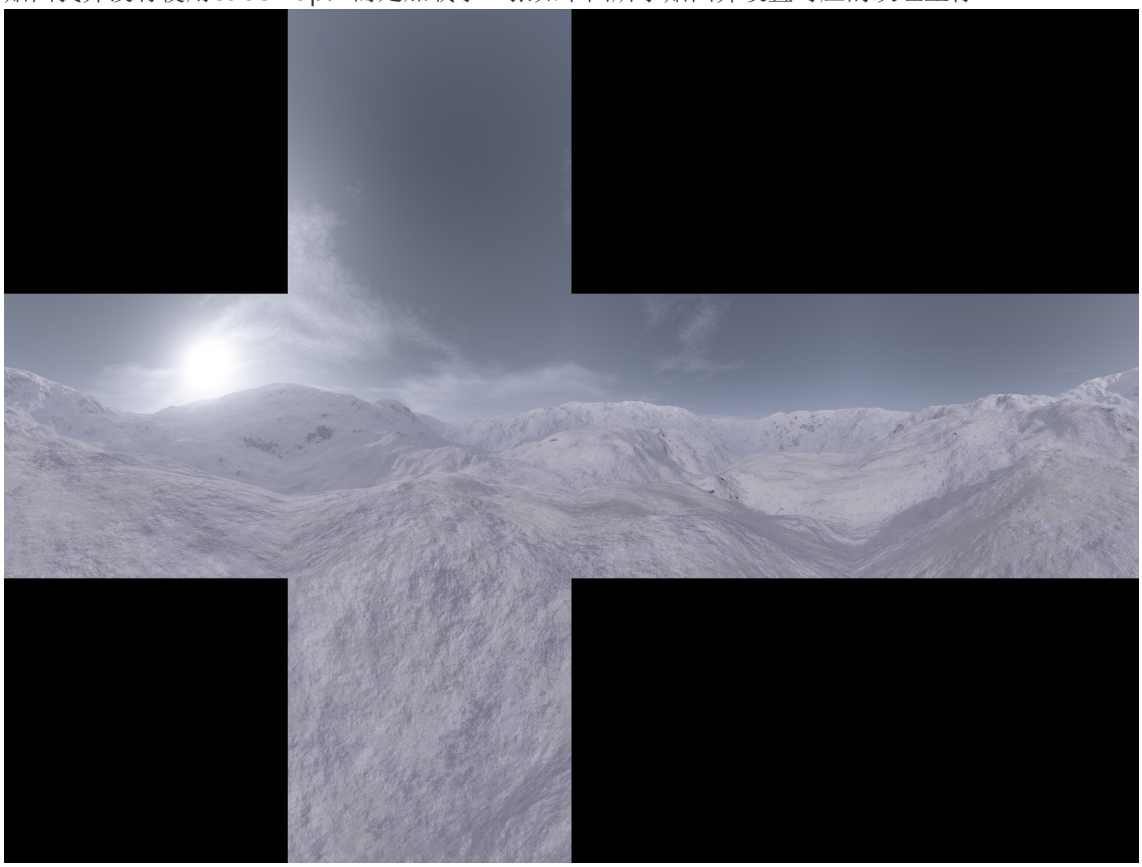
```
// 模型（基本）
class RModel{
    DirectX::XMMATRIX model;
    DirectX::XMFLOAT4 color;
    std::vector<DirectX::VertexPositionNormalTexture> vertices;
    std::vector<uint16_t> indices;
    ID3D11Buffer *vertexBuffer;
    ID3D11Buffer *indexBuffer;
    ID3D11ShaderResourceView* texture = nullptr;
    ID3D11ShaderResourceView* normalMap = nullptr;
}
// 模型（组装）
class drawable{
  std::vector<RModel*> components;
}
// 渲染对象
struct Object {
    drawable* geo;
    DirectX::XMMATRIX WorldM;
    DirectX::XMMATRIX AnimM;
}
// 场景对象
class Scene{
    std::vector<Object*> Objs;
    skybox* SkyBox; //（skybox为drawable的子类）
}
```

  - 雪人组装：雪人一共由11个不同基本模型组装，分别为：
    - 头：1 x 球体
    - 身体：1 x 球体（uniform scaled）
    - 双臂：2 x 圆柱体
    - 双手：2 x 球体（低细分）
    - 眼睛：2 x 球体
    - 鼻子：1 x 圆锥体
    - 帽子：2 x 圆柱体
    - 其中每一个基本模型都有其各自的纹理贴图

- 立方体：
  - 24个点，36个索引
  - 旋转：根据时间修改`Object`的AniM矩阵，进而修改传入shader的World矩阵
- ”上车“：
  - 相机的位置（点）与立方体的碰撞检测：
    - 因为立方体在空间中旋转，因此无法直接使用AABB方法进行判断，因此这里，我将相机的位置向量乘以立方体旋转矩阵的逆矩阵，即将相机逆变换到立方体旋转之前的空间，再在这里使用AABB方法，即可简单的判断是否相交（碰撞）
  - **相机包盒与立方体判断**：根据同样的方法，只需分别检测相机包盒的8个顶点是否与立方体碰撞，有一个点相交即包盒相交（这里假设相机的包盒不随相机的旋转而旋转）
    - 优化：可以在每一帧检测之前，判断相机是否在立方体周围球形范围内（只需几次乘法运算），如果不在，则无需进行碰撞检测。
- “下车”：
  - 当玩家在车上时，按'F'键，即可从侧面下车：具体原理为将相机的位置向外推一定距离以致相机离开立方体范围。
- 贴图与光照：
  - 本项目使用的所有光照模型都为基本的lambert光照模型。
  - 贴图：
    - diffuse贴图：场景中的全部物体均有diffuse贴图
    - 法线贴图：在地形的渲染中，我加入了法线贴图，以实现更逼真的效果
- 地形：
  - 地形我使用World Machine生成高度图，diffuse贴图以及法线贴图，再在程序中对高度图进行加载，生成地形的顶点信息，其中高度图保存为16位的raw data格式。



- 天空盒：
  - 用一个始终跟随相机平移的六面体盒子，并将其z值设为1（即无穷远，这里我使用的是1-epsilon）
  - 贴图我并没有使用cubemap，而是加载了一张如下图所示贴图并设置对应的纹理坐标



- 相机控制：
  - 具体实现借鉴了*Introduction-to-3D-Game-Programming-With-DirectX11*的样例中的相机操控
  - 同时支持键盘和鼠标控制：
    - 键盘：
      - W/A/S/D：前后左右平移
      - Q/E：左右旋转
      - Z/C：上下旋转
    - 鼠标：
      - 按住鼠标左键：可按鼠标移动方向进行旋转
      - 滚轮：控制前后平移

模块说明

- Assets文件夹：包含项目所用的shader：
  - VertexShader/PixelShader：项目中大部分模型的渲染（lambert+diffuse贴图）
  - skyboxVert/skyboxPixel：天空盒渲染
  - terrainVert/terrainPixel：地形渲染（lambert+diffuse贴图+法线贴图）
- Geometry文件夹：
  - 父类 `class drawable`：如之前场景组织结构中所述，为组装后的模型，可包含多个组成部分（`components`）
  - 子类 `class cube`, `class plane`, `class skybox`, `class snowMan`, `class terrain`
  - `class RModel`：基本模型
- Scene.h / Scene.cpp：场景类，程序渲染部分的主体，包含如下四部分：
  - Frame Update：包含控制信息接受及处理，渲染函数的调用
  - Frame Render：每一帧均调用，渲染的主体部分
  - Message Handlers
  - Direct3D Resources：主要为渲染设置以及场景信息的初始化
- Utilities.h / Utilities.cpp：包含其他函数，如索引的反序，碰撞检测，纹理加载等。