# CubeEngine 文档

## 概述：

CubeEngine 是一款使用C++开发，基于OpenGL图形API的轻量级渲染引擎。

CubeEngine 可以简单的分为若干模块，这些模块的子文件在项目目录下，以不同的文件夹的形式存储。各模块如下：Backend, Base, Entity, External, Geometry, GUI, Light, Material, Renderer, Scene, Shader, Texture。

## 渲染流程

CubeEngine提供了非常方便的方式来渲染物体，以下将叙述一下整个流程：

### 几何体的组装

组装3D几何体需要以下的方式中的一种来进行：

1. 通过Geometry下的Mesh类输入顶点，索引等空间信息，并将其设置好material，然后将其绑定至一个Entity，并设置好相机以及shader，并将其放置进scene里。
2. 通过Entity的构造函数，直接传入模型文件，设置好相机以及Shader，并将其放置进Scene里。

组装2D物体需要通过GUI下的Sprite来进行，2D物体仅通过屏幕空间显示，所以请将其绑定Scene下的GUICamera来设置相机。

### 几何体的仿射变换

上述的几何体都是继承与Base下的Node类，Node类代表了一个空间上的对象，可以使用相应的方法对几何体进行平移，旋转，缩放等仿射变换的操作。

Node类是层次性的，也就是说，可以将一个Node A作为另一个Node B的子节点，A将会附加上B上所有的变换，比方说A加入至B上，那么如果B在旋转，A也会绕着B进行旋转。

注：如果是模型的话，可能会遵循模型文件的骨骼信息，计算其指定帧的骨骼动画。

### Geometry Pass

当所有的实体都通过仿射变换摆正了正确的空间位置后，将会进行Geometry Pass，这个阶段，会将实体的所有相关信息通过MRT技术，写入到若干张纹理内，方便后续的步骤能够在屏幕空间内进行计算，减少计算亮。

### Shadow Pass(可选)

在进行完Geometry Pass之后，还要根据当前场景的光照情况，可选的进行ShadowPass，shadowPass将会以灯光的位置重新渲染一遍整个场景，但是仅仅获得一张深度信息，这个深度信息可以帮助后续步骤绘制阴影。

### Light Pass

对Geometry Pass获得有关信息，在屏幕空间内进行光照计算，这样能够避免很多不必要像素的光照计算，同时能够将光照计算的shader与Geometry Pass的shader解耦和，使得用户若是想修改实体的shader的时候，不用去修改光照部分的计算。

### Post Process

后处理阶段，这个阶段将前面得到的信息进行进一步的处理，比方说对高亮区域进行模糊再与原图叠加等等。

## 模块概览

### Backend

该模块主要是处理和平台相关的操作，用于指定OpenGL的上下文，窗口管理，关键输入事件回调(触摸、点击、按键)。

### Base

该模块包含了引擎基本的、通用的类，包括Node，Camera等等。RenderDelegate 类主要负责用于处理业务逻辑，每一个使用CubeEngine的程序的入口都必须为该类的子类。

### Entity

Entity 是指三维空间中可见的实体，比方说一个模型，或者一个地形，Entity可以通过两种方法创建，第一是通过自己手动输入模型的信息，来创建mesh，并将其绑定。第二种是直接读取外部的模型文件。

Entity还有一种特殊情况是可以通过高度图来创建，高度图被用来创建3D空间内的地形。

SkyBox是一个特殊的类，通过六张图片创建，用它可以创造出所谓“天空盒”的效果，即使用一个极大的方块，将整个场景未渲染的部分给填充完，可以增强渲染效果的真实度。

### External

External包含了引擎使用的外部的框架，当然，这些框架为了能够正确的提供给引擎使用，做了一定的修改。当前使用的有Assimp 以及 SOIL，分别用于读入模型文件以及读入图片资源。

### Geometry

Geometry包含了所有的几何相关的类，最常用的是Mesh，Mesh包含了所有三维物体显示的基本信息。

AABB和Ray用于给模型做简单的碰撞和拾取，同时用于视锥体裁剪以及八叉树分割。

### GUI

GUI包含的Sprite用于显示在屏幕空间下的二维物体，通常来说用以辅助程序运行或起到界面的作用，比方说用其制作按钮、准星等。

### Light

用于控制场景中的光照，分为环境光，方向光，点光源，探照灯。

Light下的类不能直接构造，需要通过Scene的相关构造Light的方法，构造出来。

光源可以设置是否产生阴影，如果过多的光源产生了阴影，可能会造成非常严重的性能问题，一般来说，推荐使用方向光产生阴影。

### Material

和Light一样也是用于控制场景中的光照效果，但是和Light出发点不同的是，Material是从实体的角度去讨论光照，当然，如果场景里完全没有光照(一些简单的场合，或者卡通风格的游戏)，我们仍然能够看见场景里的物体，是因为在无光情况下使用material里的diffuse纹理直接去渲染。

### Renderer

用以包装OpenGL的drawCall的直接调用。

### Scene

渲染的场景

### Shader

一个封装类，用以创建opengl的shader

### Texture

纹理类