实 验 报 告 一

（2017-2018 学年第二学期）

3D 动画游戏设计算法

实验题目：Ogre：Billboard 系列

目录：

1. 分析 Ogre Billboard 相关类，
2. 相关类有：

Ogre::Billboard,Ogre::BillboardChain,Ogre::BillboardSet,Ogre::BillboardParticleRenderer(还有些 Factory 就是工厂类，不作详细解析）

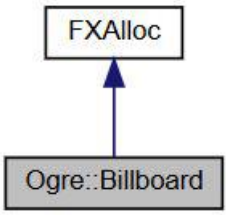
实验报告：

1. Introduction：

OGRE 版本：1.10

前言：Billboard 的中文翻译“公告板”，一般而言就是由一堆相同贴图的四边形组成的，是为了表现大规模的简单物体而生的，例如大平原上的草，就需要很多相同的四边形来表现。而这些四边形也不能死板地定在原地，不然就跟 static mesh 没什么两样了，所以 billboard 的每个 quad 都需要根据 camera 的 Lookat 和 Direction 来相应地调整旋转与朝向。在 Ogre 里面 billboard 一般不会单独存在的，它是需要在一个 BillboardSet 里面进行一批次(batch)的渲染。

2. Ogre::Billboard




```
//A billboard is a primitive which always faces the camera in every frame.
```

所以 Ogre::Billboard 类是 Billboard 公告板系统的一个图元(primitive)类，也应该是可单独操作的最小单位，就是一个空间矩形，即由两个三角形组成。而且每一帧都会面对着 camera（而这个旋转的操作就是由 Ogre 来实现）。

Billboard 拥有的字段：

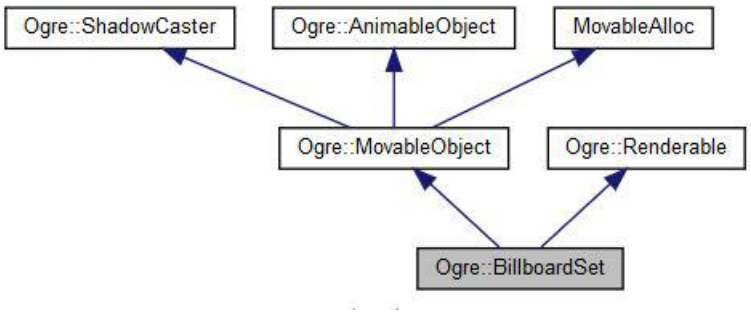
```
    protected:
        bool mOwnDimensions;
        bool mUseTexcoordRect; //是否使用纹理坐标
        uint16 mTexcoordIndex; // BillboardSet 里面纹理坐标数组索引
        FloatRect mTexcoordRect; //单独的纹理坐标
        Real mWidth;//quad 的 width
        Real mHeight;//quad 的 height
    public:
```

```
        // Note the intentional public access to main internal variables used at runtime
        // Forcing access via get/set would be too costly for 000's of billboards
        Vector3 mPosition;
        /// Normalised direction vector
        Vector3 mDirection;
        BillboardSet* mParentSet;
        ColourValue mColour;
        Radian mRotation;
```

再结合它的一些接口分析得，这就是一个比较简易的容器类，装一点基本的 billboard geometric information，给其他类使用，例如 BillboardSet，ParticleRenderer.

然后后每个刚才在 Introduction 说了,billboard 一般是不会单独存在的,但是你硬是要自己定义一个 billboard 也不是不行...所以 billboard 的 constructor 和 destructor 都是 public 的。

3. Ogre::BillboardSet




```
/*A collection of billboards (faces which are always facing the given direction) with the same (default) dimensions,
material and which are fairly close proximity to each other..*/
```

BillboardSet 是管理一批 billboard 的类，所以一般就不需要自己自己私底下创建 billboard 了，可以调用 BillboardSet 的:Billboard* Ogre::createBillboard(...)进行 BB 的创建并加入到这个 BBSet 里面。其中每个 BBSet 里面的 billboard 都有同样的维度(dimension)、材质(Material)、局部性(locality)，这是为了渲染的时候可以一个批次画出来，提高效率。

BillboardSet 的构造函数是 protected 的，可以用 SceneManager::createBillboardSet()来创建一个公告板集。

每个 BBSet 都有自己的坐标系，所以创建的一个 billboard 的坐标是在 BBSet 坐标系下的。绑定到 Scene Graph 的 basis 也是 BBSet 的 basis。

看到 BillboardSet 里面有这样的属性：

```
        /// Current camera
        Camera* mCurrentCamera;
        /// Parametric offsets of origin
        Real mLeftOff, mRightOff, mTopOff, mBottomOff;
        /// Camera axes in billboard space
        Vector3 mCamX, mCamY;
        /// Camera direction in billboard space
        Vector3 mCamDir;
        /// Camera orientation in billboard space
        Quaternion mCamQ;
        /// Camera position in billboard space
        Vector3 mCamPos;
```

这就是说，因为 billboard 实际渲染的 vertices 是 view-dependent 的，所以在处理同一批次的 billboard 时，Ogre 选择了把 Camera 的信息转换到 BillboardSet 的局部空间里。其实也说的过去，因为单个 billboard 的

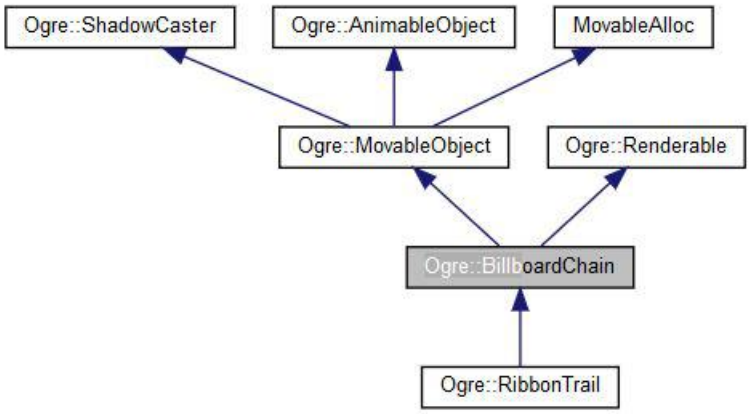
mPosition 也是定义在 local space 里面的，那肯定只变换 camera 比变换一堆 billboard 到 view space 要省力一点。

Billboard 也有多种类型，而类型就是由 BillboardSet 来管理，这决定了 billboard 的几何变换与渲染的方法。

```
enum BillboardType
{
    //默认类型，总是朝着摄像机直立(upright)
    BBT_POINT,
    //一个 set 里所有 billboard 绕着一条公有的方向向量（如 y 轴）旋转以使得 billboard quad 朝向 camera
    BBT_ORIENTED_COMMON,
    //每个 quad 都可以绕着自己的方向向量旋转以朝向 camera
    BBT_ORIENTED_SELF,
    //与 BBT_ORIENTE_COMMON 相对，这个是 billboard 朝向与给定 shared direction vector 垂直
    BBT_PERPENDICULAR_COMMON,
    BBT_PERPENDICULAR_SELF//同理
};
```

BillboardSet::injectBillboard(const Billboard& bb)会按照 Billboard Type 来更新 billboard 的 x,y 轴（由一个 orgin+x,y axis 就可以生成一个 quad，在各种 genXXX()里面具体地根据 camera 更改顶点位置并赋值至相应的 Ogre::Billboard)。Billboard quad 顶点计算的方式就在上面的注释里写了。

4. Ogre::BillboardChain




```
/*A billboard chain operates much like a traditional billboard, i.e. its segments always face the camera; the difference being that instead of a set of disconnected quads, the elements in this class are connected together in a chain which must always stay in a continuous strip. This kind of effect is useful for creating effects such as trails, beams, lightning effects, etc.*/
```

这个跟传统的 billboard 工作方式类似，但是不同的是，BBChain 的 quad 都是连在一起的，而传统的 Billboard 都是不连续的，独立的。这个东西就很好实现诸如发光物体的轨迹之类的东西。

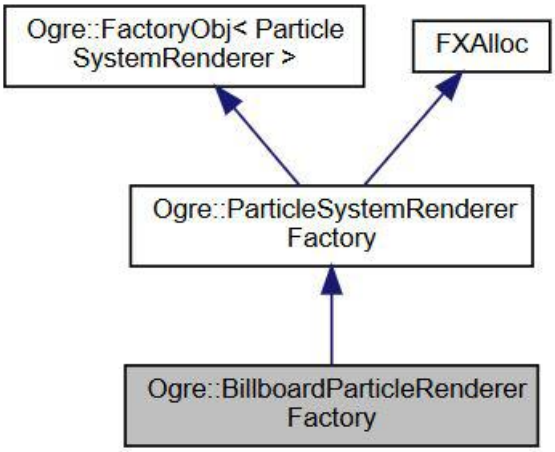
```
class _OgreExport Element
{
public:
    Element();
    Element(const Vector3 &position,
        Real width,
        Real texCoord,
        const ColourValue &colour,
        const Quaternion &orientation);

    Vector3 position;
```

```
        Real width;
        /// U or V texture coord depending on options
        Real texCoord;
        ColourValue colour;
        /// Only used when mFaceCamera == false
        Quaternion orientation;
        };
```

这个 Ogre::BillboardChain::Element 是定义在 BillboardChain 内部的，就是一条 BBChain 内的一个元素，可以通过 Ogre::BillboardChain::AddChainElement()来添加元素。

5. Ogre::BillboardParticleRenderer



/** Specialisation of ParticleSystemRenderer to render particles using a BillboardSet. This renderer has a few more options than the standard particle system, which will be passed to it automatically when the particle system itself does not understand them. */

这个是粒子系统渲染器的特化(specialisation)，就是比普通的 particle system 多一点选项，而且是专门用 billboardSet 来渲染粒子。不过粒子系统就是另外一个故事了，所以在这里就不展开分析。

```
class _OgreExport BillboardParticleRenderer : public ParticleSystemRenderer {...}
```

End.