# Softrendercore（渲染核心库dll类型）

### Basicdatastructure.hpp

在这个hpp文件当中会定义各种需要使用的数据结构类型

**别名使用：**

using Color = glm::vec4;

using Vector3D = glm::vec3;

using VectotI3D = glm::ivec3;

using Vector4D = glm::vec4;

using VectorI4D = glm::ivec4;

using Coord2D = glm::vec2;

using Coord4D = glm::vec4;

using Matrix4D = glm::mat4;

struct Vertex

{

Coord4D world\_position\_; //顶点的世界坐标

Coord4D view\_window\_position\_; //视窗坐标

Coord2D texture\_position\_; //纹理坐标

Vector3D normal\_; //法线向量

Color color\_; //颜色向量

float screen\_depth; //顶点深度

};

struct Fragment

{

Coord4D world\_position\_; //fragment的世界坐标

Coord2D screen\_position\_; //fragment的屏幕坐标

Coord2D texture\_position\_; //纹理坐标

Vector3D normal\_; //法线向量

Color color\_; //颜色向量

float screen\_depth; //fragment深度

};

## 成员类：

### Framebuffer

FrameBuffer主要实现对帧对象操作、存储，存储包括color\_buffer和dpeth\_buffer

**私有成员变量：**

int \_width\_; //帧宽度

int \_height\_; //帧高度

Std::vector<float> \_depth\_buffer\_; //图形深度缓存float vector容器

QImage \_color\_buffer\_ //图像颜色缓存

**成员函数：**

构造函数和析构函数

FrameBuffer();

FrameBuffer(const int& w, const int& h);

~FrameBuffer();

bool CompareDepth(const int& x, const int& y, const float& depth); //比较坐标深度大小

void SetPixelColor(const int& x, const int& y, const Color& color); //对pixel设定颜色

void ClearBuffer(const Color& color); //清空FrameBuffer

bool SaveImage(const QString& file\_path); //将当前FrameBuffer以图片格式保存到本地路径

QImage& GetImage(){return \_color\_buffer\_;} //获取当前FrameBuffer的颜色buffer

### SoftRenderCore

SoftRenderCore为渲染核心、接受传入三角形、各种算法处理以及对三角形进行光栅化，

**私有成员变量：**

int \_width\_; //窗口宽度

int \_height\_; //窗口高度

FrameBuffer framebuffer; //声明一个FrameBuffer对象

Color clearcolor; //声明Color对象，用于framebuffer清空帧

**成员函数：**

构造函数和析构函数

SoftRenderCore();

SoftRenderCore(const int& w, const int& h);

~SoftRenderCore();

void ClearBuffer(){framebuffer.ClearBuffer(clearcolor);} //清空framebuffer对象的帧

QImage& GetBuffer(){return framebuffer.GetImage();} //获取famebuffer对象的图像

//定义静态函数供外部调用，创建SoftRenderCore静态对象，直到程序终止才释放。

static SoftRenderCore& GetInstance(const int& w = 0 ,const int& h = 0)

static void Init(const int& w,const int& h) //定义静态函数，外部调用初始化窗口宽高

### Shader

对顶点、fragment进行各种shader处理

**私有成员变量：**

Matrix4D \_Model\_Matrix\_; //Model Transform变换矩阵

Matrix4D \_View\_Matrix\_; //View Transform摄像机变换矩阵

Matrix4D \_Projection\_Mattix\_; //Projection Transform投影变换矩阵

**成员函数**

构造函数和析构函数

Shader() //对MVP矩阵初始化、设为单位矩阵

~Shader();

virtual void VertexShader(Vertex& vertex) //顶点着色器

virtual void FragmentShader(Fragment& fragment) //Fragment着色器

void SetModelMatrix(const Matrix4D& m) //设定Model Transform变换矩阵

void SetViewMatrix(const Matrix4D& v) //设定View Transform摄像机变换矩阵

void SetProjectionMatrix(const Matrix4D& p) //设定Projection Transform投影变换矩阵

### SoftRenderMain.cpp （qt主界面）

### Model（加载model函数）

public:

Model();

void DrawModel(); //对model进行绘制

private:

std::vector<Mesh> meshes\_;

Mesh ProcessMesh();

int vertex\_count = 0; //vertex计数

int triangle\_count = 0; //triangle计数

### Mesh（网格类）

public:

Mesh() = default;

void DrawMesh(); //绘制网格

private:

std::vector<unsigned> EBO\_; //元素缓冲对象，存储vertex的index

std::vector<Vertex> VBO\_; //顶点缓冲对象,

### SoftRenderWidget（图形显示窗口）

**私有成员变量：**

int \_width\_; //窗口宽度

int \_height\_; //窗口高度

**成员函数**

void InitRender(); //初始化SoftrenderCore对象

paintEvent(QPaintEvent\* evet) //virtual protected 绘图事件（将framebuffer绘制到屏幕）