#pragma once

#include <GL/glew.h>

#include <GL/freeglut.h>

#include <glm/glm.hpp>

using namespace glm;

class Pipeline

{

public:

Pipeline()

{

m\_scale = vec3(1.0f, 1.0f, 1.0f);

m\_worldPos = vec3(0.0f, 0.0f, 0.0f);

m\_rotateInfo = vec3(0.0f, 0.0f, 0.0f);

}

void Scale(float ScaleX, float ScaleY, float ScaleZ)

{

m\_scale.x = ScaleX;

m\_scale.y = ScaleY;

m\_scale.z = ScaleZ;

}

void WorldPos(float x, float y, float z)

{

m\_worldPos.x = x;

m\_worldPos.y = y;

m\_worldPos.z = z;

}

void Rotate(float RotateX, float RotateY, float RotateZ)

{

m\_rotateInfo.x = RotateX;

m\_rotateInfo.y = RotateY;

m\_rotateInfo.z = RotateZ;

}

void SetPerspectiveProj(float FOV, float Width, float Height, float zNear, float zFar)

{

m\_persProj.FOV = FOV;

m\_persProj.Width = Width;

m\_persProj.Height = Height;

m\_persProj.zNear = zNear;

m\_persProj.zFar = zFar;

}

void SetCamera(const vec3& Pos, const vec3& Target, const vec3& Up)

{

m\_camera.Pos = Pos;

m\_camera.Target = Target;

m\_camera.Up = Up;

}

void InitCameraTransform(mat4& m, const vec3& Target, const vec3& Up)

{

vec3 N = Target;

//N.Normalize();

/\*vec3 N = normalize(Target);\*/

normalize(N);

vec3 U = Up;

//U.Normalize();

normalize(U);

//U = U.Cross(Target);

U = cross(U, Target);

//vec3 V = N.Cross(U);

vec3 V = cross(N,U);

m[0][0] = U.x; m[0][1] = U.y; m[0][2] = U.z; m[0][3] = 0.0f;

m[1][0] = V.x; m[1][1] = V.y; m[1][2] = V.z; m[1][3] = 0.0f;

m[2][0] = N.x; m[2][1] = N.y; m[2][2] = N.z; m[2][3] = 0.0f;

m[3][0] = 0.0f; m[3][1] = 0.0f; m[3][2] = 0.0f; m[3][3] = 1.0f;

}

const mat4\* GetTrans();

private:

void InitScaleTransform(mat4& m) const;

void InitRotateTransform(mat4& m) const;

void InitTranslationTransform(mat4& m) const;

void InitPerspectiveProj(mat4& m) const;

//void InitCameraTransform(mat4& m, const vec3& Target, const vec3& Up);

vec3 m\_scale;

vec3 m\_worldPos;

vec3 m\_rotateInfo;

struct {

float FOV;

float Width;

float Height;

float zNear;

float zFar;

} m\_persProj;

struct {

vec3 Pos;

vec3 Target;

vec3 Up;

} m\_camera;

mat4 m\_transformation;

};