Programming Lab 06

I will provide you with the header file Camera.h, which declares both the public and private portions of the Camera class:

```
class Camera {
    public:
        Camera(void);
        Camera(const Point& E, const Vector& look, const Vector& up,
        float fov, float aspect, float near, float far);
        Point Eye(void) const;
        Vector Right(void) const;
        Vector Up(void) const;
        Vector Back(void) const;
        Vector ViewportGeometry(void) const;
        float NearDistance(void) const;
        float FarDistance(void) const;
        Camera& Zoom(float factor);
        Camera& Forward(float distance);
        Camera& Yaw(float angle);
        Camera& Pitch(float angle);
        Camera& Roll(float angle);
    private:
        Point eye;
        Vector right, up, back;
        float width, height, distance, near, far;
};
```

You are to implement the declared functionality, the details of which are described below.

You may not alter this header file in any way.

Camera() – (default constructor) creates a camera with center of projection at the origin, looking in the direction of the negative z axis, and having up vector in the direction of the y axis. The viewport should have a eld of view of 90 , and aspect ratio of 1, near clipping plane distance of 01, and a far clipping plane distance of 10 in world coordinates.

Camera(E,look,up,fov,aspect,near,far) – (nondefault constructor) creates a camera with center of projection E, looking in the direction specified by the vector look, and oriented by the vector up. Note that up is not necessarily parallel to the actual up vector v of the camera that is created. Rather, up is orthogonal the right vector u of the created camera, and the angle between up and v should be less than 180 degree. The viewport of the created camera have a field of view fov, given in radians, and have the specified aspect ratio (ratio of the width of the viewport to its height), near

clipping plane distance, and far clipping plane distance. All values are given in world coordinates.

Eye() – returns the center of projection E of the camera, in world coordinates.

Right() – returns the right vector u of the camera, in world coordinates. This should be a unit vector.

Up() – returns the up vector v of the camera, in world coordinates. This should be a unit vector.

Back() – returns the back vector n of the camera, in world coordinates. This should be a unit vector. The vectors u, v, n are mutually orthonormal, and in this order, form a right handed coordinate system.

ViewportGeometry() – returns the vector <W,H,D>, where W is the width of the viewport, H is the height of the viewport, and D is the distance from the center of projection to the plane of projection.

NearDistance() – returns the distance (in world coordinates) from the center of projection to the near clipping plane of the viewing frustum.

FarDistance() – returns the distance (in world coordinates) from the center of projection to the far clipping plane of the viewing frustum.

Zoom(factor) – changes the dimensions of the viewport by specified amount. That is, the width and height of the viewport are multiplied by factor. The instance of the camera class is returned.

Forward(distance) moves the camera distance units (in world coordinates) in the direction n (where n is the back vector of the camera). The instance of the camera class is returned.

Yaw(float angle) – rotates the camera by the specified angle (in radians) about an axis parallel to the camera up vector v and passing through the center of projection. The instance of the camera class is returned.

Pitch(angle) – rotates the camera by the specified angle (in radians) about an axis parallel to the camera right vector u and passing through the center of projection. The instance of the camera class is returned.

Roll(angle) – rotates the camera by the specified angle (in radians) about an axis parallel to the camera back vector n and passing through the center of projection. The instance of the camera class is returned.

Although you must meet the above requirements for the public member functions, you are free to use the private member data as you see fit.

You submission for the part of the assignment should consist of a single file, named Camera.cpp. The only header files that you may include are Affine.h, Camera.h, and the standard header file cmath.