TINGE

Cameras and Objects

Programming Assignment

Deadline 4 March

§1 Cameras

You need to create a class called `Camera` which will generate `Rays`:

- The Camera Model that we are implementing is a pinhole perspective camera (all of our discussion in the meet can be found in this excellent article here) with an infinitesimally small aperture.
- Recall our discussion about the importance of aperture size, aspect ratio, angular field of view in generating rays corresponding to the field of view plane of a camera.
- Create a prototypical Ray structure (can be found in the code structure below) which stores the ray's origin and direction.
- A camera must take in the following parameters as inputs in the constructor:
  - vertical field of view (in radians)
  - film width (in pixels)
  - film height (in pixels)
  - focal length (in distance units)

```
struct Ray {
    Vec3 origin;
    Vec3 direction;

    Ray(Vec3 origin, Vec3 direction); // Parameterized constructor
    // Direction must be normalized while taking in

    Vec3 at(float t); // Should return origin + t*direction;
}

class Camera {
public:
    float vertical_fov;
    int film_width, film_height;
    float focal_length;
    float z_near = 1e-6f, z_far = 1e6f;

    Camera(float vertical_fov, int film_width, int film_height, float
        focal_length);  // Parameterized constructor
    ~Camera();  // Destructor

    Ray generate_ray(float u, float v);  // Given NDC coordinates (u, v), should
        generate the corresponding ray

    Mat4 worldToCamera; // Matrix containing transformation from world space to
        camera space
};
```

§2 Objects

You need to create 3 structures (they are just classes where everything is public by default): One for a sphere, one for a plane, one for a triangle. Also you need to write code for determining whether a ray intersects the given shape or not.

- Recall the algorithms that you described to determine whether a ray intersects a shape or not from the mini-project mini-task that was circulated during the winter.
- In addition, recall the Möller–Trumbore intersection algorithm that we discussed for ray-triangle intersection which uses the concept of coplanar vectors and barycentric coordinates ($\alpha_1 + \alpha_2 + \alpha_3 = 1, \alpha_i \in [0, 1]$).

```
struct Triangle {
    Vec3 v1, v2, v3; // Position vectors of the three vertices

    Triangle(Vec3 v1, Vec3 v2, Vec3 v3); // Parameterized constructor
}

struct Sphere {
    Vec3 c; // Position vectors of the centre
    float r; // Radius

    Sphere(Vec3 centre, float radius); // Parameterized constructor
}

struct Plane {
    Vec3 n; // Normal of the plane
    Vec3 p; // A point lying on the plane

    Plane(Vec3 normal, Vec3 point); // Parameterized constructor
}

// Methods to determine if a ray intersects an object

bool intersect(Ray& ray, Sphere& sph);

bool intersect(Ray& ray, Plane& plane);

bool intersect(Ray& ray, Triangle& tri);
};
```

Bonus: Prove that if $\vec{v_1}, \vec{v_2}, \vec{v_3}$ are three vertices of a triangle, then $\alpha_1\vec{v_1} + \alpha_2\vec{v_2} + \alpha_3\vec{v_3}$ lies inside/on the boundary of the triangle iff $\alpha_i \in [0, 1]$ when $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

If you have any query about this assignment or anything about these topics in general, feel free to text us. Also **keep in mind that we have divided you into two groups so that you help each other and enhance your learning while completing tasks more efficiently.**

Let us know when you are done implementing this, you may get some coolbiz treats for completing things before the deadline 😎.

Good luck on your code! Here's Master of Torture signing off~