Assignment 3 : Realistic Rendering with Ray Tracing

NAME: GE TIANYANG
STUDENT NUMBER: 27869265
EMAIL: USER1@MAILSERVER.XX

1 INTRODUCTION

The project is to trace ray until get the light source. According to the Monte Carlo integration, get the whole energy from a hemisphere for diffuse material, and return the value to each pixel.

2 IMPLEMENTATION DETAILS

2.1 Camera

For camera, there are three fundamental elements: fov, width, height. For simplicity, we take only one ray to trace and large amount of rays from one pixel to simulate the reality. Then consider transfer the pixel on the image coordinate to the world coordinate.

```
double rotiax=(-0.5*_imageW+unit_rand.x()+x)/(double)
    _imageW;
double rotiay=(-0.5*_imageH+unit_rand.y()+y)/(double)
    _imageH;
Vector3d direction=rotiax*_windowRight*_cameraRight +
     rotiay*_cameraUp*_windowTop+_cameraFwd*
     _nearPlaneDistance;
```

2.2 Intersection

Considering the ray intersect objects problems, there are 4 types of objects: Sphere, Quad, Triangles, Mesh. Each type has different normal and intersection caluculate.

2.3 Render

Render part intersect every object in the scene, and choose the one has the shortest distance. Then use a structure storing the distance, material and normal. At last, calculate the reflected ray and continue tracing it, if it reflected 5 times(which could be set by user) stop and return 0.

```
if(isct._hit){
    if(isct._material.getType()==EMIT){
        return isct._material.get_emission();
    }
    if(depth>=4){
//                  return Vector3d();
        return isct._material.get_colour();
    }
    double cos=isct._normal.dot(-ray.direction())/(isct.
        _normal.norm()*ray.direction().norm());
    Vector3d location=ray.origin()+ray.direction()*isct.
        _u;
    Ray re_ray=isct._material.get_reflected_ray(ray,
        location, isct._normal, Xi);
    depth++;
    Vector3d direct={25.0f,25.0f,25.0f};

    return cos*(isct._material.get_colour())*trace_ray(
        re_ray, depth, Xi);
}
```

2.4 Mesh and grid

Mesh consists of a lot of triangles. It could be very large so it obviously could not intersect test every triangle. By grid acceleration, we divide the whole aabbox into eight boxes. Then divide triangles into each box, so if ray intersect the total aabbox, it needs to decide it meets which box then search the whole triangles inside the box.

```
class Box{
    private:
        AABBox *aabbox;
    public:
        AABBox Total;
        //bl min,tr max
        void Init();
        void Dividebox();
        void Grid(int i,Vector3d &p0,Vector3d &p1,
             Vector3d &p2);
        int getid(Vector3d &p);
        AABBox getbox(int i);
        std::vector<std::vector<int>> array;
    };
```

3 RESULTS

1:2 • Name: Ge Tianyang
student number: 27869265
email: user1@mailserver.xx