# CS 331 Project: Ray Tracing

@Xingfan Xia, Alex Schneider

## What is the Project about?

Basically our project is trying to modify our previous software engine to implement the computer graphics rendering technique ray tracing. Ray tracing is a technique to generate an image by tracing the path of light through pixels in an image plane and simulating the effects of the its encounters with objects in space.

The technique is able to produce a very high degree of visual realism in terms of lighting and shadow but with a substantial computational cost. And hence it is most suitable for applications where the image can be rendered slowly ahead rather than rendered in real time. Thus it is used very often if CG cartoon and movie but not in video games which speed is critical.

Ray tracing is capable of simulating a wide range of optical effects, such as reflection, refraction, scattering, dispersion, etc. And in our project, we are trying to make a demo simulating reflection and refraction only.

## Algorithm

The basic algorithm is detailed below in pesudocode.

```
for every pixel in the scene {
    RayInitialize(from view point, to the pixel);
    Intialize NearestDepth = infinity;
    nearestObj = null;

    to everyObj in the scene {
        if (the light intersects the obj) {
            if (intersctiondepth(t) < NearestDepth) {
                NearestDepth = t;
                NearestObj = thisObj;
            }
        }
    }

    if (NearestObj == null) {
        setSphereColor(this.color);
    } else {
        checkIfObjInShadow();
        if (color is reflective) {
            RayInitialize(reflectiveRay);
            RecursiveCall;
        } else if (color is refractive) {
            RayInitialize(refractiveRay);
            RecursiveCall;
        }
        calculateColor();
        setColor();
    }
}
```

## File Organization

```
.
├── 000pixel.h
├── 000pixel.o
├── 800camera.c
├── 800light.c
├── 800mainRay.c
├── 800matrix.c
├── 800ray.c
├── 800sphere.c
├── 800vector.c
├── 810light.c
├── 810mainLighting.c
```

```
├── 810ray.c
├── 820mainDepth.c
├── 820ray.c
├── 830mainSpecular.c
├── 840mainReflection.c
├── 840ray.c
├── 840sphere.c
├── 850mainRecursive.c
├── 850sphere.c
├── 851mainRecursive.c
├── 852mainRecursive.c
├── 853mainReflection.c
├── 860camera.c
├── 860light.c
├── 860mainCamera.c
├── 860ray.c
├── 860sphere.c
├── 861effect.c
├── 861mainCamera.c
├── 862mainDNA.c
├── 870mainShadow.c
├── 870sphere.c
├── 880mainAbstracted.c
├── 881mainRefraction.c
├── 890mainTriangle.c
├── 890pointVary.c
├── 890ray.c
├── 890triangle.c
├── 900mainSpecular.c
├── 910mainShadow.c
├── 920mainRefraction.c
├── 920ray.c
├── 930mainSmooth.c
├── 931mainSmooth.c
├── 940mainFinal.c
├── Makefile
├── README.md
├── README.pdf
├── a.out
├── sprintf.c
└── stb_image.h

0 directories, 52 files
```

And among these files, the following are benchmarks we thought worth presenting.

```
Ray: 800mainRay.c
    clang 800mainRay.c 000pixel.o -lglfw -framework opengl; ./a.out

This is the most primitive version of the ray tracing engine we are implementing. It does not include any light calculation which makes it looks flat and unrealistic.
============================================================

Lighting: 810mainLighting.c
    clang 810mainLighting.c 000pixel.o -lglfw -framework opengl; ./a.out

In this version, we added diffuse lighting to give the spheres a sense of three dimensionality and it looks much more realistic than the pervious versions
============================================================

Depth: 820mainDepth.c
    clang 820mainDepth.c 000pixel.o -lglfw -framework opengl; ./a.out

In this version, we introduce depth into the engine so there nearer object is covering further ones.
============================================================

specular: 830mainSpecular.c
    clang 830mainSpecular.c 000pixel.o -lglfw -framework opengl; ./a.out

In this version, we added specular lighting to make the engine more realistic.
============================================================

recursive: 852mainRecursive.c
    clang 852mainRecursive.c 000pixel.o -lglfw -framework opengl; ./a.out

In this version, we introduce the recursive component so the light keeps bouncing between objects until no intersection happens which makes the reflection more realistic.
============================================================

```

```
spaceworm: 861effect.c
    clang 861effect.c 000pixel.o -lglfw -framework opengl; ./a.out

This is a little funny program we made out of accident. But it looks pretty cool so we include it here.
============================================================

camera: 860mainCamera.c
    clang 860mainCamera.c 000pixel.o -lglfw -framework opengl; ./a.out

In this version we introduced camera controls to the engine.
============================================================

spiral: 861mainCamera.c
    clang 861mainCamera.c 000pixel.o -lglfw -framework opengl; ./a.out

This is one of demo we made to show ray tracing to the class.
============================================================

hexspiral: 862mainDNA.c
    clang 862mainDNA.c 000pixel.o -lglfw -framework opengl; ./a.out

A more polished and interesting demo mimicing the rotation of DNA rotating.
============================================================

reflection: 880mainAbstracted.c
    clang 880mainAbstracted.c 000pixel.o -lglfw -framework opengl; ./a.out

We abstracted the code here.
============================================================

refraction: 920mainRefraction.c
    clang 920mainRefraction.c 000pixel.o -lglfw -framework opengl; ./a.out

We included refraction in the engine. Also the specular lighting involves intersection with a sphere designated as the light as opposed to defining the light as a point.
============================================================

final: 940mainFinal.c
    clang 940mainFinal.c 000pixel.o -lglfw -framework opengl;
```

```
    ./a.out
69
70  A more polished version of 920mainRefraction.
71  ============================================================
```

To run these files, just type in commands defined in `MakeFile` like these:

```
1  $make Ray
2  $make Lighting
3  $make Depth
```

The `MakeFile`:

```
Ray: 800mainRay.c
    clang 800mainRay.c 000pixel.o -lglfw -framework opengl; ./a.out

Lighting: 810mainLighting.c
    clang 810mainLighting.c 000pixel.o -lglfw -framework opengl;
./a.out

Depth: 820mainDepth.c
    clang 820mainDepth.c 000pixel.o -lglfw -framework opengl;
./a.out

specular: 830mainSpecular.c
    clang 830mainSpecular.c 000pixel.o -lglfw -framework opengl;
./a.out

recursive: 852mainRecursive.c
    clang 852mainRecursive.c 000pixel.o -lglfw -framework opengl;
./a.out

spaceworm: 861effect.c
    clang 861effect.c 000pixel.o -lglfw -framework opengl; ./a.out

camera: 860mainCamera.c
    clang 860mainCamera.c 000pixel.o -lglfw -framework opengl;
./a.out

spiral: 861mainCamera.c
    clang 861mainCamera.c 000pixel.o -lglfw -framework opengl;
./a.out

hexspiral: 862mainDNA.c
    clang 862mainDNA.c 000pixel.o -lglfw -framework opengl; ./a.out

refraction: 920mainRefraction.c
    clang 920mainRefraction.c 000pixel.o -lglfw -framework opengl;
./a.out

final: 940mainFinal.c
    clang 940mainFinal.c 000pixel.o -lglfw -framework opengl;
./a.out

run: a.out
    ./a.out
```

## Reference

Raytracing Reflection, Refraction, Fresnel, Total Internal Reflection, and Beer's Law

More Refraction

Triangle Intersection

Ray Tracing

General Code Structure