Precept 8

Huiwen Chang

Intro

Some Examples

GLSL

1. Similar grammer as C
2. provide useful functions
   a. min, max, sqrt
   b. normalize, reflect, refract

   For more information: https://www.opengl.org/wiki/Core_Language_%28GLSL%29
3. Difference!

Tips

Using loop in this way:

```
#define MAX_OBJECTS 5

uniform int numObjects;
for (int i=0; i<MAX_OBJECTS; i++) {
            if ( i>= numObjects ) break;
}
```

Tips

1. Array index - using constant or loop variable

```
int u = 5;
for (int i=0; i<MAX_OBJECTS; i++) {
    object[i]   OK!
     object[3]   OK!
    object[u]   NO!
    if( u == 5 )
         object[u]    NO!
```

Tips

1. Array index - using constant or loop variable

```
int u = 5;
for (int i=0; i<MAX_OBJECTS; i++) {
    object[i]   OK!
     object[3]   OK!
    object[u]   NO!
    if( u == 5 )
         object[u]    NO!
```

Tips

function parameter

```
in(copy in), out(copy out)
-void sqr( float x, out float res ) { res = x*x; }
-float sqr (float x) { return x*x;}
```

Tips

Recursive

```
#define MAX_RECURSION 10
function f(float x, int depth) {
    if( depth >= MAX_RECURSION) return 0;
   return 0.3 + 0.8 * f(x+1,depth+1)
}
function g() { return f(0,0) }
```

Replaced by loop

```
#define MAX_RECURSION 10
function g() {
   float x = 0.0, weight = 1.0, res = 0.0;
   for (int i=0;i < MAX_RECURSION;i++ ) {
        res = res + weight * 0.3;
        weight = weight * 0.8;
        x = x + 1.0;                    //Not x = x + 1
   }
   return res;
```

Raytracer

1. Calculate the initial ray through every pixel
2. Intersect ray with scene

    - find closest intersection with object

3. return “material color” + “reflect/recursive color”

a. material color
    i. Check if there is a light at this intersection point (construct the ray from light)
    ii. Calculate normal vector

b. reflect/recursive color(mirror/glass material)
    i. Construct new ray(needs normal vector)
    ii. calculate the color of this ray recursively.

Object

```
struct Object {
    Shape shape;
    Material material;
};
```

intersect with ray

```
void hit( Ray ray, out float hit_length, out Object obj ) {
     hit_length = INFINITY;
     for (int i=0; i<MAX_OBJECTS; i++) {
          if ( i>= numObjects ) break;

          float cur_length = findIntersection(ray, objects[i].shape);
          if ( cur_length > EPS && hit_length > cur_length ) {
               hit_length = cur_length;
               obj = objects[i];
          }
     }
}
```

Tips

EPS

```
if (a!==0)


if( a < -EPS || a > EPS )
```