CS 413 Introduction to Ray and Vector Graphics

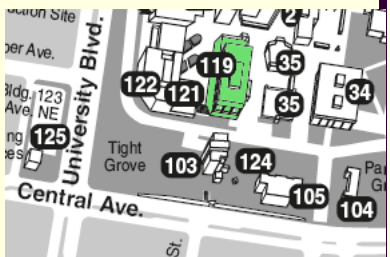
Bare-Bones Ray Tracer

Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

Web: http://cs.unm.edu/~joel/

Farris Engineering Center: 2110



2/6/2019

1

Quiz

Let a and b be vectors were a = (2,0) and b = (0,1)

Draw a cartesian representation of a

Draw a cartesian representation of b

Draw a cartesian representation of a+b

Draw a cartesian representation of -1.5(a+b)

2

2

Assignment 2: Antialiasing

- Due Wednesday, Feb 6: Read Chapter 4: Antialising
- Due Monday, Feb 11: Ray trace the function 4.1: $f(x,y)=\frac{1}{2}(1+\sin(x^2y^2))$
  a) Render the range $(x, y) \in [0,10.83]^2$ at 512×512 pixels in a parallel projection straight down the z-axis with each ray passing through the center of each pixel. Render f=0 black, f=1 white and grays between.
  b) Experiment with different image resolutions.
  c) Experiment with different antialiasing techniques. Examples might be:
    - Cast multiple rays through each pixel and average the results.
    - Cast multiple rays through each pixel and define the pixel's color by applying a weighted average based on the ray's nearness to the pixel center.
    - Cast one ray through a uniformly distributed, random location in each pixel.
    - Cast one ray through a normally or triangularly distributed random location of each pixel with probability decreasing for locations farther from the pixel center.

3

3

Coding Style

- You will be held to the coding standards specified 1.10 of the textbook.
- Use { } placement, etc consistent with code examples in the textbook,

  Except.... All code blocks that extend beyond one line must have { }:

```
for (int y = 0; y < w->vp.vres; y += 16)
     dc.DrawBitmap(tile, x, y, FALSE);
```

4

4

Questions from Chapter 3

3.1: Why is it unlikely that a ray will hit a curved implicit surface tangentially?

What is an implicit surface?

How is an *implicit surface* different from other surfaces?

What does it mean for "*a ray to hit a curved implicit surface tangentially*"?

5

3.2: Why don't we need to test for $d = 0$?

```
bool Sphere::hit(const Ray& ray, double& tmin, ShadeRec& sr) const {

    Vector3D  temp   = ray.o - center;
    double    a      = ray.d * ray.d;
    double    b      = 2.0 * temp * ray.d;
    double    c      = temp * temp - radius * radius;
    double    disc   = b * b - 4.0 * a * c;

    if (disc < 0.0) return(false);
    else {
        double e = sqrt(disc);
        double denom = 2.0 * a;
        double t = (-b - e) / denom;    // smaller root

        if (t > kEpsilon) {
                tmin = t;
                sr.normal = (temp + t * ray.d) / radius;
                sr.local_hit_point = ray.o + t * ray.d;
                return (true);
        }
//continued.....
```

```
Class Ray {
  Point3D o;   //origin
  Vector3D d;  //direction
```

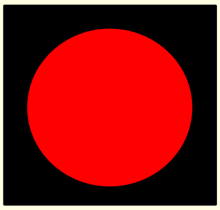
3.2: Why don't we need to test for $d = 0$?

```
bool Sphere::hit(const Ray& ray, double& tmin, ShadeRec& sr) const
    {

⋮

        t = (-b + e) / denom;    // larger root

        if (t > kEpsilon) {
                tmin = t;
                sr.normal    = (temp + t * ray.d) / radius;
                sr.local_hit_point = ray.o + t * ray.d;
                return (true);
        }
    }

    return (false);
}
```

7

7

3.3: What is the maximum radius of a sphere that will just fit into the figure?



What data do you need to answer this question?

8

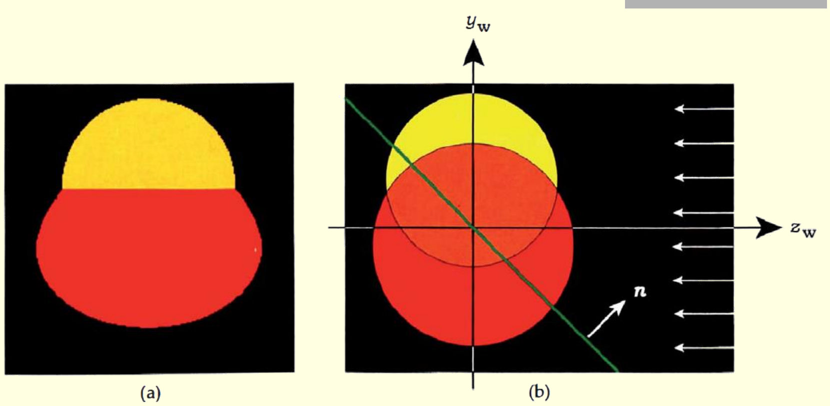
8

3.4: Build Functions

The build functions in Listings 3.11 and 3.18 illustrate how the world pointer is set in the tracer object by calling a tracer constructor with pointer **this** as the argument. The world is incomplete at this stage because the geometric objects haven't been constructed or added to it, but this does not matter. Why?

```
void World::build (void) {
  vp.set_hres(200); // view plane
  vp.set_vres(200);
  vp.set_pixel_size(1.0);

  background_color = black;
  tracer_ptr = new SingleSphere(this);

  sphere.set_center(0.0);
  sphere.set_radius(85.0);
}
```


3.5: Why, in (a), does the intersection of the two spheres appear as a straight line?

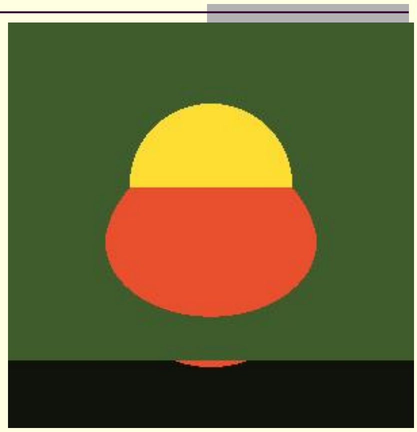


(a) Ray-traced image of two spheres and a plane;

(b) side view of the spheres and plane, looking toward the origin, along the negative x_w axis, with some rays.

3.6: What happened?

(a) rendered at 200x200



(c) rendered at 300x300

Computational Efficiency is Important

What is wrong with these inner loop statements?

```
double distSqr =
   pow((x1-x2),2) + pow((y1-y2),2);

double dx=(x1-x2)
double dy= (y1-y2)
double distSqr = dx*dx + dy*dy


double t = (point - ray.o) * a / M_PI;
```

Consider Using References verses Pointers

```
int x;
void Ptr(const int* p) { x += *p; }
void Ref(const int& p) { x += p; }
```

No need to check that the reference is not NULL. (Creating a NULL reference is possible, but difficult).

References don't require the * dereference operator. Less typing. Cleaner code (both make the same machine code).

Pointers make it extremely difficult for a compiler to know when different variables refer to the same location. Often, this prevents the compiler from generating the fastest possible code. Since a variable reference points to the same location during its entire life, a C++ compiler can do a better job of optimization than it can with pointer-based code.

13

13

Take Advantage of STL Containers

(C++ Standard Template Library)

- Not only is performance good today, it's only going to get better as STL vendors focus their efforts on optimization and compiler vendors improve template compilation.
- It's a standard.
- It's already written. And debugged, and tested. No guarantees, but better than starting from scratch.

However: The STL is not the be-all end-all library of containers and algorithms. You can get better performance by writing your own specialized containers. For instance, the STL list object must be a doubly-linked list. In cases where a singly-linked list would be fine, you pay a penalty for using the STL list object.

14

14

STL Container Usage

- When using an STL container, if several equivalent expressions have the same result, consider using the more general expression. For example:

```
a.empty()                  a.size() == 0
iter != a.end()            iter < a.end()
distance(iter1, iter2)     iter2 - iter1
```

- The former expressions are valid for every container type, while the latter are valid only for some.
- The former are also no less efficient than the latter and may even be more efficient. For example, to get the size of a linked list the list must be traversed, whereas to see that it is empty is a constant time operation.

Consider Inlined functions

- If your compiler allows whole program optimization and automatic inline-expansion of functions, use such options and do not declare any functions inline.
- If such compiler features are not available, declare suitable functions as inline in a header; suitable functions contain no more than three lines of code and have no loops.
- Inline function-expansion avoids the function call overhead. The overhead grows as the number of function arguments increases. In addition, since inline code is near to the caller code, it has better locality of reference. And because the intermediate code generated by the compiler for inlined functions is merged with the caller code, it can be optimized more easily by the compiler.

Limitations of Inlined Functions

- Every time a function containing substantial code is inlined the machine code is duplicated and the total size of the program increases. These larger programs suffer from increased cache misses.
- Inlined code is more difficult to profile. If a non-inlined function is a bottleneck, it can be found by the profiler. But if the same function is inlined wherever it is called, its run-time is scattered among many functions and the bottleneck cannot be detected by the profiler.
- For functions containing substantial amounts of code, only performance critical ones should be declared inline during optimization.

Accessing Memory in Nested Loops

When nested loops access memory, successive iterations often reuse the same word or use adjacent words that occupy the same cache block.

```
int array[1024][1024], x, y;
for(x=0; x<1024; x++)
{
  for(y=0; y<1024; y++)
  {
    total += array[x][y];
  }
}
```

In C++, will interchanging the loops improve spatial locality?

Quiz

1) After applying a linear transformation to a set of points that form two parallel lines, what types of shapes must the resulting points form?

2) After applying a linear transformation to the three basis vectors, $\hat{\imath}$ (1,0,0), $\hat{\jmath}$ (0,1,0) and $\hat{k}$(0,0,1), their new coordinates become: (0,0,-2), (0,1,0) and (1, 0, 0). Where will this transformation move the point (1,1,1)?

Implicit Plane

- We define a plane by specifying a point $\vec{\boldsymbol{a}}$ that lies on the plane and a normal $\vec{\boldsymbol{n}}$ to the plane.
- This defines the plane uniquely because there is only one plane that passes through a given point and has the orientation specified by the normal.
- The dot product two vectors is 0, iff they are $\perp$.
- Let $\vec{\boldsymbol{p}}$ be an arbitrary point on the plane.
- Implicit equation (Vector): $(\vec{\boldsymbol{p}} - \vec{\boldsymbol{a}}) \cdot \vec{\boldsymbol{n}} = 0$
- Implicit equation (Component): $Ax + By + Cz + D = 0$

Plane: Steps from Vector to Component form

- $(\vec{\boldsymbol{p}} - \vec{\boldsymbol{a}}) \cdot \vec{\boldsymbol{n}} = 0$
- $Ax + By + Cz + D = 0$
- $(x, y, z) – (a_x, a_y, a_z) * (n_x, n_y, n_z) = 0$
- $(a_x\text{-}x, \quad a_y\text{-}y, \quad a_z\text{-}z) \quad * \quad (n_x, n_y, n_z) = 0$
- $n_x * (a_x\text{-}x) \quad + \quad n_y * (a_y\text{-}y) \quad + \quad n_z * (a_z\text{-}z) = 0$
- $n_x a_x\text{-}\ n_x x + n_y a_y\text{-}\ n_y y + n_z a_z\text{-}\ n_z z = 0$
- $\text{-}n_x x - n_y y - n_z z \quad + \quad n_x a_x + n_y a_y + n_z a_z = 0$
- Ax + By + Cz + D = 0

21

21

Implicit Surface of a Sphere (spherical shell)

- Let $\vec{\boldsymbol{c}} = \left(c_x, c_y, c_z\right)$ be a point called the center.
- Let $\vec{\boldsymbol{p}} = (x, y, z)$ be any point on the spherical shell.
- Let r be a non-negative number called the radius.

- Vector: $||\vec{\boldsymbol{p}} - \vec{\boldsymbol{c}}|| = r$
- Component:

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 - r^2 = 0$$

22

22

Equation of a Ray

- In common language, people often call any arbitrary, curved path that is fairly thin a “line”.
- In Vector Graphics, a line is straight, infinitely long and infinitely thin.
- Let $\vec{\boldsymbol{c}} = (c_x, c_y, c_z)$ be a point in space (called the *endpoint* or *origin* or *start* or *tail*.
- Let $\vec{\boldsymbol{d}} = (d_x, d_y, d_z)$ be a unit vector (called the *direction*)
- Let *t* (called *time*) be a number where $t \in (-\infty, +\infty)$

$$\vec{\boldsymbol{p}} = \vec{\boldsymbol{d}}t + \vec{\boldsymbol{c}}$$

23