Class Topics and Objectives
- Photo-realistic image generation
- Ray Tracing!
- Learn and implement the algorithms needed to create ray traced images
- Serious numerical computing and programming class
- Assumes you know CG I material

Class Structure
- Weekly lectures & reading assignments
- 6 regular programming assignments
- 1 extra credit assignment
- Post images on the web
  - E-mail the URL to david@cs.drexel.edu
- Upload code to Bb Vista
- Grad students give presentations
- Final exam on material not covered by assignments

Grading
- Graduate Section
  - Programming Assignments - 75%
  - In-class Presentation - 10%
  - Final Exam – 15%
- Undergraduate Section
  - Programming Assignments - 85%
  - Final Exam – 15%
- Late policy
  - 1 point/day
  - Maximum 5 points off

Go To Web Sites
- Class web site
- Previous pictures web site

Slide Credits
- Kevin Suffern - University of Technology, Sydney, Australia
- G. Drew Kessler & Larry Hodges - Georgia Institute of Technology
- Fredo Durand & Barb Cutler - MIT
- Computer Graphics I
Ray Casting

- Determines visible surfaces by tracing "light" rays from the viewer’s eye to the objects
- View plane is divided by a pixel grid
- The eye ray is fired from the center of projection through each pixel

Ray Tracing

- Extension of ray casting
- Idea: Continue to bounce the ray in the scene
- Shoot rays to light sources
- Simple and powerful
- Reflections, shadows, transparency and multiple light sources

Ray Tracing Diagrams

Issues

- Ray-object intersections
- Complex, hierarchical models (CSG?)
- Transformations
- Camera models
- Recursive algorithms
- Surface physics (shading models)
- Color representations
- Light representations
- Sampling, anti-aliasing and filtering
- Geometric optics
- Acceleration techniques
- Texture mapping

The primary ray is defined in world coordinates by:
- The camera location is (the ray origin)
- a unit direction vector $d$

The expression for the primary ray is:
$$p = o + i \cdot d$$
where $i$ is the ray parameter.
Calculating Primary Rays

- Given (in world coordinates)
  - Camera (eye point) location $\mathbf{O}$
  - Camera view out direction $\mathbf{Z}$
  - Camera view up vector $\mathbf{V}_{up}$
  - Distance to image plane $\mathbf{d}$
  - Horizontal camera view angle $\theta$
  - Pixel resolution of image plane $(h_{res}, v_{res})$
- Calculate set of rays $(\mathbf{d})$ that equally samples the image plane.

Notice the following difference between parallel and perspective viewing in ray tracing:

In parallel viewing, each primary ray has a different origin, but the same direction.

In perspective viewing, each primary ray has the same origin, but a different direction.

We index the pixels horizontally (from left to right) with

$$ j : 0 \leq j \leq v_{res} - 1,$$

and vertically (from top to bottom) with

$$ k : 0 \leq k \leq h_{res} - 1.$$
Calculate Preliminary Values

- Camera view side direction ($X_v$)
  - $X_v = Z_v \times V_{up}$ (left-handed system)
- Make sure that $Y_v$ is orthogonal to $X_v$ & $Z_v$
  - $Y_v = X_v \times Z_v$
- Be sure to normalize $X_v$, $Y_v$ & $Z_v$
- Horizontal length of image plane ($s_j$)
- Next slide

Calculate Those Rays!

- $P_{0,0} + \alpha X_v - \beta Y_v$ sweeps out image plane
- $0 \leq \alpha \leq S_x$, $0 \leq \beta \leq S_y$

for (j=0; j++; j < h_res)
for (k=0; k++; k < v_res)

\[
\begin{align*}
    d_{j,k} &= (P_{0,0} + S_j(j/(h_res-1)) \times X_v - S_k(k/(v_res-1)) \times Y_v) \cdot O_v; \\
    d'_{j,k} &= d_{j,k} / \|d_{j,k}\|; \\
    \text{Image}[j,k] &= \text{ray_trace}(O_v, d'_{j,k}, \text{Scene}); 
\end{align*}
\]

Vertical length of image plane ($s_j$)

- $s_j = h \times \tan(\theta/2)$
- $s_j = 2h \times \tan(\theta/2)$

Assume square pixels
Size of the pixels

What effect does changing the physical size of the pixels have on the image?

For a specified image resolution \( (h, w) \), the size of the window is proportional to the size of the pixels.

For a fixed view plane distance \( d \), the field of view is proportional to \( \theta \).

For fixed \( \theta \), the size of the window is proportional to \( h_{win} \) and \( v_{win} \).

Increasing \( h_{win} \) and \( v_{win} \) increases the field of view of the camera, provided \( \theta \) and \( d \) are kept the same.

Some of these effects are illustrated in the following figures.
Parameters
- X and Y resolution of image
- Camera location & direction
- Distance between camera & image plane
- Camera view angle
- Distance between pixels
- These are not independent!
- Goal → Choose your independent variables and calculate your $d'$s

I recommend setting …
- X and Y resolution of image
  - $(h_{res}, v_{res})$
- Camera location & orientation
  - $O$ & $V_{up}$
- Distance between camera & image plane
  - $d$ (a positive scalar, e.g. 10)
- Camera view angle
  - $\theta$

Ray/Sphere Intersection (Algebraic Solution)

Ray is defined by $R(t) = R_o + R_d * t$ where $t > 0$.
- $R_o$ = Origin of ray at $(x_o, y_o, z_o)$
- $R_d$ = Direction of ray $(x_d, y_d, z_d)$ (unit vector)

Sphere's surface is defined by the set of points $(x_s, y_s, z_s)$ satisfying the equation:

$$ (x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2 - r_s^2 = 0 $$

Center of sphere: $(x_c, y_c, z_c)$
Radius of sphere: $r_s$
### Possible Cases of Ray/Sphere Intersection
1. Ray intersects sphere twice with \( t > 0 \)
2. Ray tangent to sphere
3. Ray intersects sphere with \( t < 0 \)
4. Ray originates inside sphere
5. Ray does not intersect sphere

### Solving For \( t \)
Substitute the basic ray equation:
\[
\begin{align*}
x &= x_0 + x_d t \\
y &= y_0 + y_d t \\
z &= z_0 + z_d t
\end{align*}
\]
into the equation of the sphere:
\[
(x_0 + x_d t - x_c)^2 + (y_0 + y_d t - y_c)^2 + (z_0 + z_d t - z_c)^2 = r^2
\]
This is a quadratic equation in \( t \): \( At^2 + Bt + C = 0 \), where
\[
\begin{align*}
A &= x_d^2 + y_d^2 + z_d^2 \\
B &= 2[x_d(x_0 - x_c) + y_d(y_0 - y_c) + z_d(z_0 - z_c)] \\
C &= (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2
\end{align*}
\]
Note: \( A = 1 \)

### Relation of \( t \) to Intersection
We want the smallest positive \( t \) - call it \( t_i \)
\[
\begin{align*}
t_0 & \text{ Discriminant } = 0 \\
t_1 & \text{ Discriminant } < 0
\end{align*}
\]
Actual Intersection
Interception point,
\[
(x_i, y_i, z_i) = (x_0 + x_d t_i, y_0 + y_d t_i, z_0 + z_d t_i)
\]
Unit vector normal to the surface at this point is \( \mathbf{N} = (x_i - x_c)/r_s, (y_i - y_c)/r_s, (z_i - z_c)/r_s \)
If the ray originates inside the sphere, \( \mathbf{N} \) should be negated so that it points back toward the center.

### Summary (Algebraic Solution)
1. Calculate \( A, B \) and \( C \) of the quadratic intersection equation
2. Calculate discriminant (If < 0, then no intersection)
3. Calculate \( t_0 \)
4. If \( t_0 \) is negative, then calculate \( t_1 \) (If \( t_1 \) is also negative, no intersection point on ray)
5. Calculate intersection point
6. Calculate normal vector at point

Helpful pointers:
- Precompute \( r_s^2 \)
- Precompute \( 1/r_s \)
- If computed \( t \) is very small then, due to rounding error, you may not have a valid intersection

### Ray-Triangle Intersection
Fredo Durand
Barb Cutler
MIT
Barycentric definition of a plane

- \( P(\alpha, \beta, \gamma) = a\alpha + b\beta + c\gamma \)
- \( \alpha + \beta + \gamma = 1 \)

Barycentric definition of a triangle

- \( P(\alpha, \beta, \gamma) = a\alpha + b\beta + c\gamma \)
- \( \alpha + \beta + \gamma = 1 \)
- \( 0 < \alpha < 1 \)
- \( 0 < \beta < 1 \)
- \( 0 < \gamma < 1 \)

Given \( P \), how can we compute \( \alpha, \beta, \gamma \)?

- Compute the areas of the opposite subtriangle
  - Ratio with complete area
  - \( \alpha = A_1/A \)
  - \( \beta = A_2/A \)
  - \( \gamma = A_3/A \)
- Use signed areas for points outside the triangle

Simplify

- Since \( \alpha + \beta + \gamma = 1 \), we can write \( \alpha = 1 - \beta - \gamma \)
- \( P(\beta, \gamma) = (1-\beta-\gamma) a + b + c \gamma \)

Simplify

- \( P(\beta, \gamma) = (1-\beta-\gamma) a + b + c \gamma \)
- \( P(\beta, \gamma) = a(1-\beta-\gamma) + b(1-\beta-\gamma) + c(1-\beta-\gamma) \gamma \)
- Non-orthogonal coordinate system of the plane

How do we use it for intersection?

- Insert ray equation into barycentric expression of triangle
- \( P(t) = a\beta + (1-t) a + b + c \gamma \)
- Intersection if \( \beta \geq 1 \), \( 0 \leq \beta \) and \( 0 \leq \gamma \)
Intersection

- $R_1 + D = a_1 + \beta (b_1 - a_1) + \gamma (c_1 - a_1)$
- $R_2 + D = a_2 + \beta (b_2 - a_2) + \gamma (c_2 - a_2)$
- $R_3 + D = a_3 + \beta (b_3 - a_3) + \gamma (c_3 - a_3)$

Calculate Intersection Point

- Are $\beta$ and $\gamma$ both non-negative?
- Is $\beta + \gamma \leq 1$?
- Is $t$ non-negative?
- If so, you’ve got an intersection!
- $P = R + tD$

Design Your Ray Tracer!

- “Novice programmers often neglect the design phase, instead diving into coding without giving thought to the evolution of a piece of software over time. The result is a haphazard, poorly modularized code which is difficult to maintain and modify. A few minutes of planning short-term and long-term goals at the beginning is time well spent.”
Modular Functionality
- Read and write image files
- Create hierarchical geometric models with transformations
- Support several geometric primitives
- Geometric calculations & parameters
  - Ray-object intersections
  - Normals
  - Bounding boxes
  - Color & surface properties
  - Texture maps
- Intersect arbitrary ray with scene
  - Stop at first intersection (shadow rays)
- Recursive generation and summation of rays
- Adaptive sampling of image plane
- Light properties

Possible Software Structure

Progression of Assignments
- Basic ray tracer with spheres and triangles
- Triangle/sphere intersection. No shading.
- Simple shading and point light sources
- Acceleration techniques
- Adaptive super-sampling/anti-aliasing
- Shadows and reflections
- Transparency and refraction
- 2D/3D texture mapping

Wrap Up
- First programming assignment
  - Due 4/17/12
  - Go to web page
- Grad students will present papers in class. Several during Week 5.
- Need to pick paper.