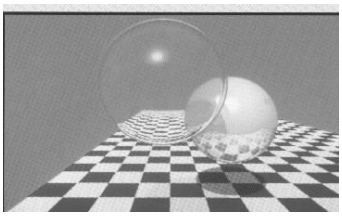
So You Want to Write a Ray Tracer

Assignment #1

- Which is something you may wish to do since it *is* Assignment #1
- In fact…

Assignment #1

- Goal is to be able to reproduce this:



Assignments

- Some advice:
  - Choose either #1 (Ray tracing) or #2 (radioisity)
  - #3 will be real time shading…
    - Challenge will be getting environment set up
  - #4 – Tone Reproduction
    - Modification of #1 or #2
- In fact
  - If you choose to do assignments 1 & 2, you need not do any other.

Assignment #1

- Write a simplified recursive ray tracer for a scene which may contain:
  - Spheres
  - Planar checkerboard rectangle
  - A single point light source
- Use Phong illumination for color calculations
  - Light and color use 0-1 range

Assignment #1

- Program may be:
  - Interactive – the final image is rendered to a window on the screen
  - Batch – the final image is rendered to a file.
- In either case, program should be involved:
  - raytrace infile <outfile>
    - Outfile required only for batch applications.

Assignment #1

- Infile – Describes scene to be rendered.
  - Very simple formatted text file
  - Format:
    - Line 1: Camera Params (position, lookat, up, focal length, frame size)
    - Line 2: Light Parameters (position, rgb, ambient rgb)
    - Line 3: Image Resolution and maxdepth
    - Line 4: Background Color (rgb)
    - Line 5: k_r k_t
    - Lines 6-n: Object descriptions

Assignment #1

- Infile – Object descriptions / 2 lines per object
  - Sphere
    - S center(x,y,z) radius, isTransparent (0,1), index of refraction
    - ambient rgb, diffuse rgb, specular rgb, k_a, k_d, k_s
  - Floor
    - F vertex1 vertex2 vertex3 vertex4, cheksize
    - ambient rgb1, diffuse rgb1, specular rgb1,ambient rgb2, diffuse rgb2, specular rgb2, k_a, k_d, k_s
    - Normal for floor will always be (0, 1, 0)
    - Floors are not transparent

Assignment #1

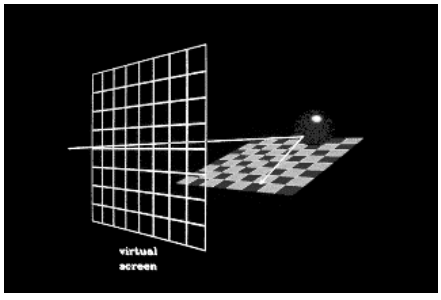
- Example infile

```
0 -10 0 0 0 0 0 1 0 5 10 5
10 10 10 0.7 0.7 0.7 0.2 0.2 0.2
480 240 5
0 0 1
0.33 0.33
S 0 0 0 3.0 1 1.0
0.5 0.5 0.5 0.5 0.5 0.5 1.0 1.0 1.0 0.2 0.4 0.5
S 0 0 10 2.5 0 0.8
0.0 0.8 0.5 0.0 0.8 0.5 1.0 0.0 0.0 0.2 0.4 0.5
F -3 -2 -3 -3 -3 3 3 -3 3 3 -3 -3 0.5
1.0 0.0 0.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0 0.0 1.0 0.0 1.0
  1.0 1.0 0.2 0.4 0.5
```

The algorithm

```
for(each pixel on the viewing area) {
   for(each primitive in the world model) {
          if(ray-pixel intersection) {
             select the frontmost intersection;
             recursively trace the relection and
                   refraction rays;
             calculate color;
          }
   }
}
```

The algorithm



The to-do list

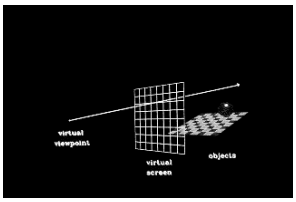
1. Input processing – parse input file
2. Spawn a ray and send into scene
   1. Define ray direction
   2. Check for intersection
   3. Calculate and return color
3. Display or save final image

Tips

- Coordinate spaces
  - Can do in camera space or world space
  - Camera space
    - Must transform all objects/lights to camera space
  - World space
    - Must transform initial rays to world space

Tips – World Space

- Need only transform the location of 1st "pixel" location on image plane and dx, dy, and dz as you move across and down the plane



Tips – Camera Transformations

$$M = \begin{bmatrix} u_x & u_y & u_z & -o_x \\ v_x & v_y & v_z & -o_y \\ n_x & n_y & n_z & -o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (u_x, u_y, u_z) are coordinates of unit u vector w.r.t. world space
- Similar for v, n,
- (o_x, o_y, o_z) is the origin of view space w.r.t world space

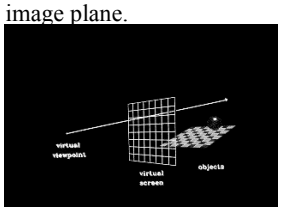
Projection

- Note: Projection not required as this will be done as part of the ray tracing process

$$\begin{bmatrix} P_u \\ P_v \\ P_n \\ w \end{bmatrix} = M \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Tips – Calculating Color

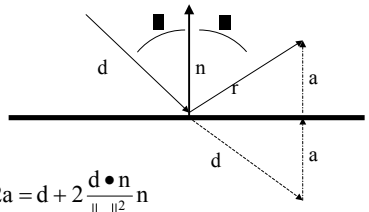
- Find point of intersection
  - Good Safety tip – only consider intersections if they occur past the image plane.
- Spawn rays
  - Shadow
  - Reflective
  - Transmission



Shadow Ray

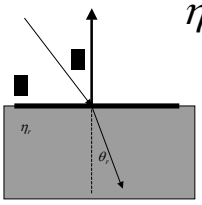
- For shadow rays, ray is spawned toward each light source
  - No need for point of intersection for each object, just need to know if there *is* an intersection.

Calculating Reflection Rays



$$r = d + 2a = d + 2\frac{d \bullet n}{\|n\|^2} n$$

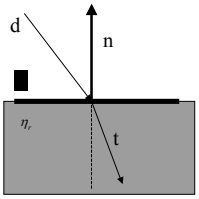
Calculating Transmission Rays



$$\eta_i \sin\theta_i = \eta_r \sin\theta_r$$

Calculating Transmission Rays

$$\eta_i \sin\theta_i = \eta_r \sin\theta_r$$



t Can be solved as:

$$t = \frac{\eta_i(d - n(d \bullet n))}{\eta_r} - n\sqrt{1 - \frac{\eta_i^2(1-(d \bullet n)^2\theta_i)}{\eta_r^2}}$$

Assumes d and n are normalized

Tips – Creating images

- You don't really need the full power of a 3D API to do ray tracing
  - Just need the ability to write color values to pixels
  - Some of the matrix operation routines may be helpful.

Tips – Writing image files

- C library
  - Netppm
    - Netpbm is a freeware toolkit/library for manipulation of graphic images, including conversion of images between a variety of different formats.
    - http://netpbm.sourceforge.net/
- Java
  - Java2D
    - java.awt.image
    - javax.imageio
  - Java Advanced Imaging
    - http://java.sun.com/products/java-media/jai/

Questions?

Assignments

- Grading
  - Each assignment is worth 20 points:
    - 5 points – for something that compiles
    - 10 points – for something that runs incorrectly
    - 15 points – for something that runs correctly
    - 20 points – something that runs + extras
      - Well structured and documented code
      - Additional bells and whistles

Bells and Whistles

- 15 points
  - Doing reflection rays
- 20 points
  - Doing reflection and transmission
- Note
  - Still must parse input correctly if transmission not done
  - Please, please, please submit readable and documented code!

Due dates

- If doing both assn 1 & 2
  - Due Feb 6th
- If doing assn 1 but NOT assn 2
  - Due Jan 23rd