Midterm 2

<50%: 4
<60%: 6
<70%: 13
<80%: 13
<90%: 19
<100%: 6
MIDTERM 2 ANALYSIS

• (Whiteboard)
PROGRAMMING A4 OUT TODAY

• Ray Tracing
  • C++
    • In Javascript it would be VERY slow and hard to debug
  • Don’t get stuck, ask for help
  • If you are unfamiliar with C++, post it on Piazza, we & others will help

• Due in two weeks

• OR Path Tracing
  • For those who are brave to explore
  • To boldly go where no man has gone before ©
  • Harder to get the core working, but easier to get creative part + bonus
CPSC 314

25 - GLOBAL ILLUMINATION II
PATH TRACING.

Textbook: 20

UGRAD.CS.UBC.CA/~CS314

Mikhail Bessmeltsev
RAY TRACING LIMITATION

• Only specular reflections consider other objects
  • Well okay, refractions too
• Diffuse and glossy surfaces will only reflect light source!
• We’re using Lambert and Phong models of *direct illumination*
• How can we model diffuse/glossy models in a similar way?
IDEA

• Every object is a light source!
WHAT REALLY HAPPENS AT DIFFUSE SURFACE?

• Imagine a single photon hitting a diffuse surface
• It can’t “scatter”, it has to go somewhere
• Goes into a random direction
• Uniform distribution
WHAT REALLY HAPPENS AT DIFFUSE SURFACE?

- Imagine a single photon hitting a diffuse surface
- It can’t “scatter”, it has to go somewhere
- Goes into a random direction
- Uniform distribution
- **Energy ~ cosine**
WHAT HAPPENS AT GLOSSY SURFACE?

• Goes most probably along the perfect reflection
• With a smaller probability deviates
• Gaussian distribution
PATH TRACING FRAMEWORK

Eye

Image Plane

Light Source

Diffuse object

Reflective object

Glossy object
PATH TRACING FRAMEWORK

Eye → Image Plane → Diffuse object → Light Source → Glossy object

Reflective object
PATH TRACING FRAMEWORK

Image Plane

Eye

Light Source

Diffuse object

Reflective object

Glossy object
PATH TRACING FRAMEWORK

Eye → Image Plane → Diffuse object 

Light Source 

Reflective object 

Glossy object
PATH TRACING ALGORITHM

1. Shoot a ray though pixel (i,j). Set $\text{attenuation}$ to 1.0.
2. Find the closest intersection of the ray with an object
3. Randomly choose between “emission” and “reflection”
   a. If “emission”, return $\text{emissionColor}$;
   b. If “reflection”,
      Reflect a ray in a random direction
      $\text{rayWeight} \,*= \text{reflectance}$;
      Go to 2.
SIMPLEST PATH TRACER

• For all pixels (i,j):
  • Ray \( r = \text{generateRay}(i,j); \)
  • For \( k=1,...,N; \)
    • \( \text{PixelColor}(i,j) += \text{pathTrace}(r)/N; \)
SIMPLEST PATH TRACER

PathTrace(Ray r) {
    P = closestIntersection(r);
    if (random(emit, reflect) == emit)
        return EmissionColor;
    else {
        Ray v = {intersectionPt,
                 randomDirectionInHemisphere(r.normalWhereObjWasHit)};
        double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
        return PathTrace(v)*cos_theta*reflectance;
    }
}
WHAT YOU GET
WHAT YOU GET

Area light

Reflections from diffuse objects 'color bleeding'

Soft shadows
MONTE CARLO METHODS

• General idea: compute something using random sampling
• Used for computing integrals of complex functions
• E.g. areas or volumes
  • If it’s hard to compute analytically
  • But easy to test if a point is inside
• If we throw enough random samples, by the law of large numbers, mean ~ empirical mean
GENERATING RANDOM POINTS

• We usually have a single function random();
• That generates a random number from 0 to 1
• Uniform distribution
GENERATING RANDOM POINTS

• How to generate random points in a unit square?
GENERATING RANDOM POINTS

• How to generate random points in a unit square?

```c
for (i=0; i<N; i++)
{
    x = rand();
    y = rand();
}
```
GENERATING RANDOM POINTS

• How to generate random points in a rectangle?
GENERATING RANDOM POINTS

• How to generate random points in a rectangle?

```plaintext
for (i=0...N)
{
    x = w*rand();
    y = h*rand();
}
```
GENERATING RANDOM POINTS

• In a right triangle?
GENERATING RANDOM POINTS

• In a right triangle?
GENERATING RANDOM POINTS

• In a right triangle?
GENERATING RANDOM POINTS

• In a right triangle?

```
for (i=0...N)
{
    a1 = rand();
    a2 = rand();
    if (a1+a2<1)
    {
        x = a1*w;
        y = a2*h;
    }
}
```
GENERATING RANDOM POINTS

• In an arbitrary triangle?
GENERATING RANDOM POINTS

• In an arbitrary triangle?
GENERATING RANDOM POINTS

- In a circle?
REJECTION SAMPLING

• Say you have a complicated shape, and you want to generate points uniformly in it

• One elegant way is rejection sampling:
  • Generate a point evenly in the bounding box
  • If it’s not inside our shape, discard

• How many points will be rejected (ratio)?
MONTE CARLO METHODS

• Example: approximating $\pi$:

"Pi 30K" by CaitlinJo - Own work. This mathematical image was created with Mathematica. Licensed under CC BY 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Pi_30K.gif#/media/File:Pi_30K.gif
MONTE CARLO METHODS

- Example: computing a weird integral
MonTE-CARLO: RAY TRACING

• Now in RAY tracing, we can generate rays randomly from an area light source
• Instead of shooting a single shadow ray,
  • Generate many randomly towards a light source
    • Generate a point on the light source
    • Shoot a ray towards that point
  • Average their contribution

• Soft shadows in RAY tracing!
SIMPLEST PATH TRACER

PathTrace(Ray r) {
    P = closestIntersection(r);
    if (random(emit, reflect) == emit)
        return EmissionColor;
    else {
        Ray v = {intersectionPt, randomDirectionInHemisphere(r.normalWhereObjWasHit)};
        double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
        return PathTrace(v)*cos_theta*reflectance;
    }
}
HOW TO CHOOSE BETWEEN 3 ACTIONS?

1. “Go to 314 lecture”, $P = 0.4$
2. “Go have lunch instead”, $P = 0.5$
3. “Sleep in”, $P = 0.1$
HOW TO CHOOSE BETWEEN 3 ACTIONS?

1. “Go to 314 lecture”, P = 0.4
2. “Go have lunch instead”, P = 0.5
3. “Sleep in”, P = 0.1

```python
x = random();
if (x < 0.4)
    return 1;
else if (x < 0.9)
    return 2;
else
    return 3;
```
HOW TO GENERATE A DIRECTION IN HEMISPHERE?

• Evenly?
HOW TO GENERATE A DIRECTION IN HEMISPHERE?

- Evenly?
- Option 1:
  - $\phi = random(1.0)$;
  - $\theta = random(1.0)$;
  - $\text{dir} = \begin{pmatrix} \cos(\theta) \sin(\phi) \\ \sin(\theta) \sin(\phi) \\ \cos(\theta) \end{pmatrix}$
HOW TO GENERATE A DIRECTION IN HEMISPHERE?

• Evenly?

• Option 1:
  
  - $\phi = \text{random}(1.0)$;
  - $\theta = \text{random}(1.0)$;
  - $dir = \begin{pmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}$

• It’s not an even distribution!
HOW TO GENERATE A DIRECTION IN HEMISPHERE?

• Evenly?
• Option 1:
  • $\phi = \text{random}(1.0)$;
  • $\theta = \text{random}(1.0)$;
  • $dir = \begin{pmatrix} \cos(\theta) \cdot \sin(\phi) \\ \sin(\theta) \cdot \sin(\phi) \\ \cos(\phi) \end{pmatrix}$

• Option 2:
  • Generate $z$ evenly
  • Generate $\theta$ evenly
  • $dir = \begin{pmatrix} \sqrt{1 - z^2} \cdot \cos(\theta) \\ \sqrt{1 - z^2} \cdot \sin(\theta) \\ z \end{pmatrix}$

Incorrectly distributed points
Correctly distributed points
BACK TO PATH TRACING

• Now we know how to reflect a ray randomly when surface is diffuse
BACK TO PATH TRACING

• And we know how to choose between “emit” and “scatter” events
• We can similarly add “absorb”, “reflect”, “refract”
HOW TO GENERATE REFLECTION FOR A GLOSSY SURFACE

• Random direction in hemisphere
• Multiply energy by phong term!
HOW TO GENERATE REFLECTION FOR A GLOSSY SURFACE

Listing 25.4. Code fragment from the build function for Figure 25.8.

Figure 25.8. Glossy sphere surrounded by the \( \Phi \) image and rendered with the following values of \( \varepsilon \): (a) 1.0; (b) 10.0; (c) 100.0; (d) 1000.0; (e) 10000.0; (f) 100000.0.
• We know how to reflect a random ray
• And we know how to choose between “emit” and “scatter” events
• We can similarly add “absorb”, “reflect”, “refract”
RAY TRACING VS PATH TRACING

• Global illumination algorithms
• Rays emitted FROM camera

• Ray Tracing
  • Single ray per pixel
  • Supports indirect lighting only from specular surfaces
    • No color bleeding
  • Shoots shadow rays to compute direct illumination
    • Soft shadows are harder to get

• Path Tracing (*may produce renders indistinguishable from photos*)
  • Many rays per pixel, their color averaged
  • At each interaction, ray direction changes randomly with some distribution
  • No difference between light sources and objects
    • Soft shadows, complex materials, etc.
    • Supports all sorts of indirect lighting
BONUS 3

Half of a sphere of radius R (‘object’) is lit by a long light source (depicted as a blue line). The light source and the object are centered horizontally. At the edge of the light source there is a small mirror (depicted as a green line). What is the total illumination of the very tip of the sphere (red dot)?

Assume the object and the light have color (intensity) of (1,1,1) and the object is perfectly diffuse.