Rendering Overview
Rendering

- Algorithmically generating a 2D image from 3D models

- Raster graphics
CSE OSU rendering courses

• CSE 5542 Real Time Rendering
  – Comprehensive list of topics in real-time rendering using OpenGL and GLSL, including coordinate systems, transformations, viewing, illumination, texture mapping, and shader-based algorithms.

• CSE 5545 Advanced Computer Graphics
  – Advanced topics in computer graphics; image synthesis, lighting and rendering, sampling and material properties, volume rendering.
Topics

• Lighting models and shading

• Viewing transformation

• Raytracing overview
Lighting and Shading
Physics of light

Red lines – single bounce; local illumination
Green lines – multiple bounce; global illumination
Vectors for light modeling
Point light

Emit light in all directions

Vector from point to light
Directional light

Light vector always the same
Spotlight

Cutoff angle
Area light (only in Unity Pro)

Any point within an area is a point light source.
Shadows

Area light source can produce more realistic shadow effects. It is typically approximated by a number of point light sources. So it is more computationally expensive.
Shadows

Point Light Source

Area Light Source (or multiple point light)
Light scattering on a surface

Figure from http://en.wikipedia.org/wiki/Bidirectional_scattering_distribution_function
Illumination model

Ambient + Diffuse + Specular = Total
Ambient illumination

- Approximation for global illumination
- Often set as a constant value, resulting in object having one, flat color
- $L_{ambient} = I_{ambient} \times K_{ambient}$
- [Light = Intensity * Material]
Diffuse illumination

• Diffuse reflection assumes that the light is equally reflected in every direction.
• In other words, if the light source and the object has fixed positions in the world space, the camera motion doesn’t affect the appearance of the object.
• The reflection only depends on the incoming direction. It is independent of the outgoing (viewing) direction.
Diffuse illumination

- \( L_{\text{diffuse}} = I_{\text{diffuse}} \times K_{\text{diffuse}} \times \cos \theta_i \)
Ambient and Diffuse illumination example

$K_{\text{diffuse}}$  

$K_{\text{ambient}}$
Specular illumination

• Some materials, such as plastic and metal, can have shiny highlight spots.
• This is due to a glossy mirror-like reflection.
• Materials have different shininess/glossiness.
Specular reflection

- \( R = 2(L \cdot N)N - L \)

- \( L_{\text{specular}} = I_{\text{specular}} * K_{\text{specular}} * \cos^n \phi \)

- \( n \) is shininess coefficient
- \( \phi \) is the angle between EYE and R
Specular illumination example

An Example

$K_{\text{specular}}$  
Shininess $n$
Normal of polygon(s)
Normal of vertex

Average the normals of the polygons it is used in
Shading model comparison

Flat shading – one normal per polygon

Gouraud shading – one normal per vertex; compute color of a point on the polygon by: computing color at each vertex and linearly interpolate colors inside the polygon

Phong shading – one normal per vertex; compute color of a point on the polygon by: linearly interpolate the normal vector, then perform lighting calculation
Bump Map

• A simple way to enrich surface details is to use bump map.

• The basic idea is to modify the normal using a texture.

• Red and blue colors stand for normal change in two directions.
Viewing and Projection Transformations
Graphics pipeline reminder

1. Local coordinate
2. Local->World
3. World coordinate
4. World->Eye
5. Eye coordinate
6. Projection Matrix
7. Clip coordinate
8. others
9. Screen coordinate
Projection

• Transform a point from a high dimensional space to a low-dimensional space.

• In 3D, the projection means mapping a 3D point onto a 2D projection plane (or called image plane).

• There are two basic projection types:
  – Parallel (orthographic)
  – Perspective
Orthographic projection

\[
\begin{align*}
\begin{bmatrix}
    x \\
    y \\
    k \\
    1
\end{bmatrix}
&= 
\begin{bmatrix}
    1 & 0 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 & 0 \\
    0 & 0 & 0 & k & 0 \\
    0 & 0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\end{align*}
\]
Orthographic projection
Properties of orthographic projection

- Not realistic looking
- Can preserve parallel lines
- Can preserve ratios
- Does not preserve angles between lines
- Mostly used in computer aided design and architectural drawing software
Foreshortening – Pietro Perugino fresco example
Orthographic projection
no foreshortening
Perspective projection has foreshortening.
Perspective projection viewing volume – frustum

Can you find near, far and field of view in Unity?
Properties of perspective projection

- Realistic looking
- Lines are mapped to lines
- Parallel lines may not remain parallel
- Ratios are not preserved
- Distances cannot be directly measured, as in parallel projection
Raytracing
Ray Tracing

• Algorithm:
  Shoot a ray through each pixel
  Find first object intersected by ray

• Computation:
  Compute ray (orthographic or perspective?)
  Compute ray-object intersections (parametric line equation)
  Compute shading (use light and normal vectors)
Shade of Object at Point

- Ambient
- Diffuse
- Specular
- Texture

- Shadows
- Reflections
- Transparency (refraction)
Shadows

• Determine when light ray is blocked from reaching object.
• Ray-object intersection calculation

For each pixel
for each object
for each light source
for each object
Transparency
Transparency & Refraction

- Ray changes direction in transition between materials
- Material properties give ratio of in/out angles
Transparency & Refraction
Recursive Ray Tracing
Sampling and Aliasing

Problem: Representing pixel by a single ray.
Efficiency

- $1280 \times 1024 = 1,310,720 \approx 10^6$ pixels.
- $10^6$ initial rays.
- $10^6$ reflection rays.
- Potentially $10^6$ refraction rays.
- $3 \times 10^6$ shadow rays (3 lights.)

Next level:
- Potentially $4 \times 10^6$ refraction/reflection rays.

1,000,000 polygons.
$10^7 \times 10^6 = 10^{13}$ ray-polygon intersection calculations.
Intersection Data Structures

Coarse test to see if a ray could *possibly* intersect object

OR

Divide space up
  – sort objects into spatial buckets
  – trace ray from bucket to bucket
Bounding Boxes
Spatial Subdivision