33 – BIG REVIEW

- This review is NOT everything you need to know
- This is just a list of questions you might want to answer in order to start preparation
- Now is a good time to start preparing!

RENDERING

- What is rendering?
- What is the input for the rendering process? Output?
- What are the stages of rendering?
  - Describe each one
- How do we make rendering real-time?
- How do we make rendering realistic?
THE RENDERING PIPELINE

- Vertices and attributes
  - Vertex Shader
    - Modelview transform
    - Per-vertex attributes
  - Rasterization
    - Scan conversion
    - Interpolation
  - Per-Sample Operations
    - Depth test
    - Blending
  → Framebuffer
- Vertex Post-Processing
  - Viewport transform
  - Clipping
- Fragment Shader
  - Texturing/...
  - Lighting/shading

HOMOGENEOUS COORDINATES

- Why do we use homogeneous coordinates?
- How to convert them from/to Euclidean coordinates?
  - Is such conversion 1-1?
- Where in the pipeline do we operate with HC/EC?
- How to tell a vector from a point in HC?
TRANFORMATION MATRICES

- What's an affine transformation? Linear?
- Can all of them be represented as matrix operations?
- What's a structure of a transformation matrix?

AUGMENTED MATRIX

\[
\begin{bmatrix}
m_{11} & m_{12} & m_{13} & b_x \\
m_{21} & m_{22} & m_{23} & b_y \\
m_{31} & m_{32} & m_{33} & b_z \\
0 & 0 & 0 & w
\end{bmatrix}
\]
TRANSFORMING COORDINATE FRAME

Columns are new basis vectors (and new origin)!

\[
\begin{pmatrix}
\cos \theta & -\sin \theta & p_x \cdot (1-\cos \theta) + p_y \cdot \sin \theta \\
\sin \theta & \cos \theta & p_y \cdot (1-\cos \theta) + p_x \cdot \sin \theta \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\vec{v}_x \\
\vec{v}_y \\
1
\end{pmatrix}
\]

PIPELINE

- What are the transformations involved in the pipeline?
- What are the coordinate systems involved?
- Why do we do perspective divide?
- Why do we do clipping before perspective divide?
- Why do we need viewport transform?
MATH

• What are implicit, explicit, and parametric ways to define geometry?
  • What are their limitations?

• How to intersect two objects if they are
  • Both implicitly defined
  • Both explicitly defined

• How many parameters do we need to represent objects parametrically?
MATH

- How to calculate a normal to an implicit surface/curve?
- How to calculate a tangent plane?
- How to approximate surface area of some 2D shape?
- How to intersect a ray with a planar polygon in 2D? In 3D?

TRIANGLE

- Normal
  \[ n = \frac{(P_1 - P_0) \times (P_2 - P_0)}{\|(P_1 - P_0) \times (P_2 - P_0)\|} \]

- Area
  \[ A = \frac{1}{2} \left\| \overrightarrow{P_0P_1} \times \overrightarrow{P_0P_2} \right\| \]
AFFINE TRANSFORMATIONS

suppose we want

$P_A = \text{Rot}(z, -90) P_h$

$P_W = \text{Trans}(2,3,0) P_A$

$P_W = \text{Trans}(2,3,0) \text{Rot}(z, -90) P_h$
COMPOSING TRANSFORMATIONS

\[ P_w = \text{Trans}(2,3,0) \cdot \text{Rot}(z,-90) \cdot P_h \]

- R-to-L: interpret operations wrt fixed coords
  - moving object
- L-to-R: interpret operations wrt local coords
  - changing coordinate system

\[
\begin{align*}
M_{MV} &= \text{Trans}(2,3,0) \cdot M_{MV} \\
M_{MV} &= \text{Rot}(z,-90) \cdot M_{MV}
\end{align*}
\]

COMPOSING TRANSFORMATIONS

\[ P_w = \text{Rot}(z,-90) \cdot \text{Trans}(-3,2,0) \cdot P_h \]
ROTATION ABOUT A POINT: MOVING OBJECT

1. Rotate about p by \( \theta \):
   \[ \begin{align*}
   \theta \quad p = (x, y) \\
   \end{align*} \]

2. Translate p to origin:
   \[ T(x, y, z)R(z, \theta)T(-x, -y, -z) \]

3. Rotate about origin:
   \[ \begin{align*}
   \end{align*} \]

4. Translate p back:
   \[ \begin{align*}
   \end{align*} \]

SIMPLE COMPOSITIONS

\[ \begin{align*}
Tr(x_1, y_1, z_1) \cdot Tr(x_2, y_2, z_2) &= Tr(x_1 + x_2, y_1 + y_2, z_1 + z_2) \\
Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) &= Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1)
\end{align*} \]

\[ \begin{align*}
Scale(a, b, c) \cdot Scale(d, e, f) &= Scale(ad, be, cf) \\
Scale(a, b, c) \cdot Scale(d, e, f) &= Scale(d, e, f) \cdot Scale(a, b, c)
\end{align*} \]

\[ \begin{align*}
Rot(\alpha, 0, 0, 1) \cdot Rot(\beta, 0, 0, 1) &= Rot(\alpha + \beta, 0, 0, 1) \\
Rot(\alpha, 0, 0, 1) \cdot Rot(\beta, 0, 0, 1) &= Rot(\beta, 0, 0, 1) \cdot Rot(\alpha, 0, 0, 1)
\end{align*} \]
MORE COMPLICATED COMPOSITIONS

\[
Tr(x, y, z) \cdot Scale(a, b, c) \neq Scale(a, b, c) \cdot Tr(x, y, z)
\]
\[
Tr(x, y, z) \cdot Scale(a, b, c) = Scale(a, b, c) \cdot Tr\left(\frac{x}{a'}, \frac{y}{b'}, \frac{z}{c'}\right)
\]

\[
Tr(x, y, z) \cdot Rot(\alpha, 0, 0, 1) \neq Rot(\alpha, 0, 0, 1) \cdot Tr(x, y, z)
\]
\[
Rot(\alpha, 0, 0, 1) \cdot Rot(\beta, 0, 1, 0) \neq Rot(\beta, 0, 1, 0) \cdot Rot(\alpha, 0, 0, 1)
\]
\[
Scale(a, a, a) \cdot Rot(\beta, 0, 0, 1) = Rot(\beta, 0, 0, 1) \cdot Scale(a, a, a)
\]
\[
Scale(a, b, c) \cdot Rot(\beta, 0, 0, 1) \neq Rot(\beta, 0, 0, 1) \cdot Scale(a, b, c)
\]

INVERSE TRANSFORMS

\[
Tr(x, y, z)^{-1} = Tr(-x, -y, -z)
\]
\[
Rot(\alpha, 0, 0, 1)^{-1} = Rot(-\alpha, 0, 0, 1) = Rot(\alpha, 0, 0, 1)^T \quad \text{(orthogonal!)}
\]
\[
Scale(a, b, c)^{-1} = Scale\left(\frac{1}{a'}, \frac{1}{b'}, \frac{1}{c'}\right)
\]
ROTATION AFTER NON-UNIFORM SCALE

• Not what you’d expect!
• $M = \text{Scale}(a, b, c) \cdot \text{Rot}(\beta, 0,0,1) =$

\[
\begin{pmatrix}
a \cdot \cos(\beta) & -a \cdot \sin(\beta) & 0 & 0 \\
b \cdot \sin(\beta) & b \cdot \cos(\beta) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

• Basis vector not orthogonal

TRANSFORMATION HIERARCHIES

• Example

$M_1 = \text{Tr}_{(x,y)} \cdot \text{Rot}\theta_1$

$M_2 = M_1 \cdot \text{Tr}_{(2,5,5)} \cdot \text{Rot} \theta_2$

$M_3 = M_2 \cdot \text{Tr}_{(0,-3,5)} \cdot \text{Rot} \theta_3$
**PROJECTIONS**

- What is the purpose of projections?
- What's the difference between ortho- and perspective projections?
- Who chooses which projection to use?
- Can we get a nearly orthographic projection while using a perspective projection matrix?
- What happens to z in perspective projection?
- What happens to the view volumes?

**CLIPPING**

- What happens to points during clipping? Triangles?
- What are the equations of the frustum planes?
- How can we test if a triangle should be clipped?
RASTERIZATION

• What's rasterization?
• How do we rasterize a polygon?
• Why do we interpolate?
• What are the values we typically interpolate?
• How?
• How is it done in ray/path tracing?

LIGHTING & SHADING

• What's a Gouraud shading?
• What are Lambert/Phong materials?
• If the scene is lit with only ambient light, what will we see?
  • Only diffuse/specular?
• How can we control size of the specular highlight?
• How do we shade in ray tracing? in path tracing?
• In path tracing, how can we simulate more complex materials?
TEXTURING

• How can we tile a wall with bricks?
  • If a texture contains a single brick, what should be texture coordinates for wall's corners?
• Why do we use mipmaps?
• How much storage do we need for them?
• How do we generate mipmaps?
• Where do we get texture coordinates?
• How do we interpolate them?

BUMP AND NORMAL MAPPING

• Why?
• Which mapping would you use to add scales to a fish?
• Bullets on the walls?
• Fur on an animal?
• How do we apply bump mapping?
ENVIRONMENT MAPS

• Why do we need them?
• What are the types?
• How do we generate them?
• How do we apply them?
• When do we re-generate them?

SHADOW MAPS

• Why do we need them?
• How does it fit into pipeline?
• What’s the algorithm?
ILLUMINATION MODELS/ALGORITHMS

Local illumination - Fast
Ignore real physics, approximate the look
Interaction of each object with light
• Compute on surface (light to viewer)

Global illumination – Slow
Physically based
Interactions between objects

BASIC RAY-TRACING ALGORITHM

RayTrace(r,scene)
obj = FirstIntersection(r,scene)

if (no obj) return BackgroundColor;
else {
    if (Reflect(obj))
        reflect_color = RayTrace(ReflectRay(r, obj));
    else
        reflect_color = Black;
    
    if (Transparent(obj))
        refract_color = RayTrace(RefractRay(r, obj));
    else
        refract_color = Black;

    return Shade(reflect_color, refract_color, obj);
}
WHEN TO STOP?

• Algorithm above does not terminate

• Termination Criteria
  • No intersection
  • Contribution of secondary ray attenuated below threshold – each reflection/refraction attenuates ray
  • Maximal depth is reached

SIMULATING SHADOWS

• Trace ray from each ray-object intersection point to light sources
  • If the ray intersects an object in between ⇒ point is shadowed from the light source

```shadow = RayTrace(LightRay(obj,r,light));
return Shade(shadow,reflect_color,refract_color,obj);```
RAY TRACING: IDEA

- Core of ray-tracing ⇒ must be extremely efficient
- Usually involves solving a set of equations
  - Using implicit formulas for primitives

RAY-OBJECT INTERSECTIONS

Example: Ray-Sphere intersection

ray: \( x(t) = p_x + v_x t, \ y(t) = p_y + v_y t, \ z(t) = p_z + v_z t \)

(unit) sphere: \( x^2 + y^2 + z^2 = 1 \)

quadratic equation in \( t \):

\[
0 = (p_x + v_x t)^2 + (p_y + v_y t)^2 + (p_z + v_z t)^2 - 1
= t^2 (v_x^2 + v_y^2 + v_z^2) + 2t(p_x v_x + p_y v_y + p_z v_z) + (p_x^2 + p_y^2 + p_z^2) - 1
\]
RAY-TRACING: DIRECT ILLUMINATION

• Local surface information (normal...)
  • For implicit surfaces \( F(x,y,z)=0 \):
    normal \( \mathbf{n}(x,y,z) \) is gradient of \( F \):
    \[
    \mathbf{n}(x,y,z) = \nabla F(x,y,z) = \begin{pmatrix}
    \frac{\partial F(x,y,z)}{\partial x} \\
    \frac{\partial F(x,y,z)}{\partial y} \\
    \frac{\partial F(x,y,z)}{\partial z}
    \end{pmatrix}
    \]
  • Example:
    \[
    F(x,y,z) = x^2 + y^2 + z^2 - r^2
    \]
    \[
    \mathbf{n}(x,y,z) = \begin{pmatrix}
    2x \\
    2y \\
    2z
    \end{pmatrix}
    \]
    Needs to be normalized!

OPTIMIZED RAY-TRACING

• Basic algorithm is simple but VERY expensive
• Optimize...
  • Reduce number of rays traced
  • Reduce number of ray-object intersection calculations
• Parallelize
  • Cluster
  • GPU
• Methods
  • Bounding Boxes
  • Spatial Subdivision
    • Visibility, Intersection/Collision
  • Tree Pruning
ALIASING & ANTI-ALIASING

MIPMAPPING

use “image pyramid” to precompute averaged versions of the texture
THANK YOU AND GOOD BYE!

FIN