A4: FACE TO FACE

- Starting today
- Sign up if you haven’t
- No-show: -20%, marked after
- You have to understand each line of your code
LITTLE FAVOR

• Please answer a couple of questions here:
  • https://survey.ubc.ca/s/cpsc-314/

• It’s not a course evaluation, it’s just for us to know how to make the course more popular.

• Thank you!
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

Vertex Post-Processing
- Viewport transform
- Clipping

Rasterization
- Scan conversion
- Interpolation

Fragment Shader
- Texturing/...
- Lighting/shading

Per-Sample Operations
- Depth test
- Blending

Framebuffer
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?
TRANSFORMATION 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

Linear Transformation

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

Translation
TRANSFORMING COORDINATE FRAME

Columns are new basis vectors (and new origin)!

$$\begin{pmatrix}
\cos \theta & -\sin \theta & p_x \cdot (1-\cos \theta) + p_y \cdot \sin \theta \\
\sin \theta & \cos \theta & p_y \cdot (1-\cos \theta) + p_x \cdot \sin \theta \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
v_x \\
v_y \\
1
\end{pmatrix}$$
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?
Vertex Shader

Modelview transform

Object Coordinate System

Vertices

modelling transformation → view transformation → projection

World Coordinate System

Camera Coordinate System

Clip Coordinate System

Per-vertex attributes

Vertex Post-Processing

clipping → perspective divide → viewport transform

Normalized Device Coordinates

Window Coordinates
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
COMPOSING 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

\[ P_W = Trans(2,3,0)Rot(z,-90)P_h \]

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

\[ M_{MV} = Trans(2,3,0) \cdot M_{MV} \]
\[ M_{MV} = Rot(z,-90)M_{MV} \]
COMPOSING TRANSFORMATIONS

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

![Diagram showing the composition of transformations]

- **Rotate**($z, -90$)
- **Translate**(-3,2,0) in local coords
ROTATION ABOUT A POINT: MOVING OBJECT

\[ T(x, y, z)R(z, \theta)T(-x, -y, -z) \]
SIMPLE COMPOSITIONS

\[ Tr(x_1, y_1, z_1) \cdot Tr(x_2, y_2, z_2) = Tr(x_1 + x_2, y_1 + y_2, z_1 + z_2) \]
\[ Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) = Tr(x_2, y_2, z_2) \cdot Tr(x_1, y_1, z_1) \]

\[ Scale(a, b, c) \cdot Scale(d, e, f) = Scale(ad, be, cf) \]
\[ Scale(a, b, c) \cdot Scale(d, e, f) = Scale(d, e, f) \cdot Scale(a, b, c) \]

\[ Rot(\alpha, 0, 0, 1) \cdot Rot(\beta, 0, 0, 1) = Rot(\alpha + \beta, 0, 0, 1) \]
\[ Rot(\alpha, 0, 0, 1) \cdot Rot(\beta, 0, 0, 1) = Rot(\beta, 0, 0, 1) \cdot Rot(\alpha, 0, 0, 1) \]
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

\[\begin{align*}
Tr(x, y, z) & = Tr(-x, -y, -z) \\
Rot(\alpha, 0, 0, 1) & = Rot(-\alpha, 0, 0, 1) = Rot(\alpha, 0, 0, 1)^T \quad \text{(orthogonal!)} \\
Scale(a, b, c) & = Scale \left( \frac{1}{a}, \frac{1}{b}, \frac{1}{c} \right)
\end{align*}\]
**ROTATION AFTER NON-UNIFORM SCALE**

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

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

- Basis vector not orthogonal!
- (demo)
TRANSFORMATION HIERARCHIES

• Example

\[ M_1 = \text{Tr}_{(x,y)} \cdot \text{Rot} \theta_1 \]
\[ M_2 = M_1 \cdot \text{Tr}_{(2.5,5.5)} \cdot \text{Rot} \theta_2 \]
\[ M_3 = M_2 \cdot \text{Tr}_{(0,-3.5)} \cdot \text{Rot} \theta_3 \]
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?