Lecture 6:
Rasterization 1 (Triangles)
Announcements

• Bad news
  - Professor is ill

• Good news
  - I (Goksu, your TA) will be covering today’s lecture!
  - And today’s topics are pretty easy

• Tomorrow’s section
  - Walkthrough of provide code framework
  - Perspective projection - Non-uniform z
Last Lectures

• Transformation
  - Modeling + Viewing
  - Model + View + Projection
  - Linear, Affine, Projection transformations
  - Homogeneous coordinates
Today

• Finishing up Viewing
  - Viewport transformation

• Rasterization
  - Different raster displays
  - Rasterizing a triangle
Perspective Projection

• What’s near plane’s l, r, b, t then?
  - If explicitly specified, good
  - Sometimes people prefer defining them with vertical **field-of-view** (fovY) and **aspect ratio**, and assume symmetry i.e. l = -r, b = -t

Vertical Field of View (fovY)

Aspect ratio = width / height
Perspective Projection

- How to convert from \( \text{fovY} \) and aspect to \( l, r, b, t \)?
  - Trivial

\[
\tan \frac{\text{fovY}}{2} = \frac{t}{|n|}
\]

\[
\text{aspect} = \frac{r}{t}
\]
What’s after MVP?

- **Model** transformation (placing objects)
- **View** transformation (placing camera)
- **Projection** transformation
  - Orthographic projection (cuboid to “canonical” cube $[-1, 1]^3$)
  - Perspective projection (frustum to “canonical” cube)
- **Canonical** cube to ?
Canonical Cube to Screen

• What is a screen?
  - A typical kind of raster display

• Raster == discretize (personally)
  - Usually to an array of pixels
  - Size of the array: resolution

• Pixel (FYI, short for “picture element”)
  - For now: A pixel is a little square with uniform color
  - Color is a mixture of (red, green, blue)
Canonical Cube to Screen

- Defining the screen space
  - Slightly different from the “tiger book”

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>(2, 0)</td>
<td>(2, 0)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(0, 1)</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>(2, 1)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>(0, 2)</td>
<td>(0, 2)</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>(2, 2)</td>
<td>(2, 2)</td>
</tr>
</tbody>
</table>

Pixels’ indices are in the form of (x, y), where both x and y are integers.

Pixels’ indices are from (0, 0) to (width - 1, height - 1).

Pixel (x, y) is centered at (x + 0.5, y + 0.5).

The screen covers range (0, 0) to (width, height).
Canonical Cube to Screen

- Irrelevant to z
- Transform in xy plane: $[-1, 1]^2$ to $[0, \text{width}] \times [0, \text{height}]$
Canonical Cube to Screen

- Irrelevant to $z$
- Transform in $xy$ plane: $[-1, 1]^2$ to $[0, \text{width}] \times [0, \text{height}]$
- Viewport transform matrix:

$$M_{\text{viewport}} = \begin{pmatrix}
\frac{\text{width}}{2} & 0 & 0 & \frac{\text{width}}{2} \\
0 & \frac{\text{height}}{2} & 0 & \frac{\text{height}}{2} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}$$
Today: Rasterizing Triangles into Pixels
Drawing Machines
CNC Sharpie Drawing Machine

Aaron Panone with Matt W. Moore
Slide courtesy of Prof. Ren Ng, UC Berkeley
Laser Cutters

Slide courtesy of Prof. Ren Ng, UC Berkeley
Oscilloscope
Oscilloscope Art

Jerobeam Fenderson
Slide courtesy of Prof. Ren Ng, UC Berkeley

https://www.youtube.com/watch?v=rtR63-ecUNo
Cathode Ray Tube

Slide courtesy of Prof. Ren Ng, UC Berkeley
Television - Raster Display CRT

Cathode Ray Tube

Raster Scan
(modulate intensity)

Raster Scan Pattern of Interlaced Display

Slide courtesy of Prof. Ren Ng, UC Berkeley
Frame Buffer: Memory for a Raster Display

Image = 2D array of colors

DAC = Digital to Analog Convertors

Analog

Digital

Slide courtesy of Prof. Ren Ng, UC Berkeley
A Sampling of Different Raster Displays
Flat Panel Displays

Low-Res LCD Display

Color LCD, OLED, ...
LCD (Liquid Crystal Display) Pixel

Principle: block or transmit light by twisting polarization

Illumination from backlight (e.g. fluorescent or LED)

Intermediate intensity levels by partial twist
LED Array Display

Light emitting diode array

Slide courtesy of Prof. Ren Ng, UC Berkeley
DMD Projection Display

DIGITAL MICRO MIRROR DEVICE (DMD)
(SLM - Spatial Light Modulator)

MICRO MIRRORS CLOSE UP
DMD Projection Display

Array of micro-mirror pixels

DMD = Digital Micromirror Device

Slide courtesy of Prof. Ren Ng, UC Berkeley
Electrophoretic (Electronic Ink) Display

Slide courtesy of Prof. Ren Ng, UC Berkeley
Drawing to Raster Displays
Polygon Meshes

Slide courtesy of Prof. Ren Ng, UC Berkeley
Triangle Meshes
Triangle Meshes
Triangles - Fundamental Shape Primitives

Why triangles?

- Most basic polygon
- Break up other polygons
- Optimize one implementation
- Triangles have unique properties
  - Guaranteed to be planar
  - Well-defined interior
  - Well-defined method for interpolating values at vertices over triangle (barycentric interpolation)
Drawing a Triangle To The Framebuffer ("Rasterization")
What Pixel Values Approximate a Triangle?

Input: position of triangle vertices projected on screen

Output: set of pixel values approximating triangle

(2.2, 1.3)
(4.4, 11.0)
(15.3, 8.6)
Today, Let’s Start With A Simple Approach: Sampling
Sampling a Function

Evaluating a function at a point is sampling.
We can **discretize** a function by sampling.

```c
for( int x = 0; x < xmax; x++ )
    output[x] = f(x);
```

Sampling is a core idea in graphics.
We sample time (1D), area (2D), direction (2D), volume (3D) ...

Let’s Try Rasterization As 2D Sampling
Sample If Each Pixel Center Is Inside Triangle
Sample If Each Pixel Center Is Inside Triangle
Define Binary Function: \( \text{inside}(\text{tri}, x, y) \)

\[
\text{inside}(t, x, y) = \begin{cases} 
1 & \text{pixel (x, y) in triangle } t \\
0 & \text{otherwise}
\end{cases}
\]
Rasterization = Sampling A 2D Indicator Function

\[
\begin{array}{l}
\text{for( int } x = 0; x < xmax; x++ )} \\
\quad \text{for( int } y = 0; y < ymax; y++ )} \\
\quad \text{Image}[x][y] = f(x + 0.5, y + 0.5); \\
\end{array}
\]

Rasterize triangle \( \text{tri} \) by sampling the function 
\( f(x,y) = \text{inside}(\text{tri},x,y) \)
Recall: Sample Locations

Sample location for pixel $(x, y)$

Slide courtesy of Prof. Ren Ng, UC Berkeley
Evaluating inside(tri, x, y)
Triangle = Intersection of Three Half Planes
Each Line Defines Two Half-Planes

Implicit line equation
  - \( L(x,y) = Ax + By + C \)

- On line: \( L(x,y) = 0 \)
- Above line: \( L(x,y) > 0 \)
- Below line: \( L(x,y) < 0 \)
Line Equation Derivation

Line Tangent Vector

\[ T = P_1 - P_0 = (x_1 - x_0, y_1 - y_0) \]
Line Equation Derivation

Perp($x, y$) = ($-y, x$)

General Perpendicular Vector in 2D
Line Normal Vector

\[ N = \text{Perp}(T) = (-(y_1 - y_0), x_1 - x_0) \]
Line Equation Derivation

\[ V = P - P_0 = (x - x_0, y - y_0) \]
Line Equation

\[ L(x, y) = V \cdot N = -(x - x_0)(y_1 - y_0) + (y - y_0)(x_1 - x_0) \]
Line Equation Tests

\[ L(x, y) = V \cdot N > 0 \]
Line Equation Tests

\[ L(x, y) = V \cdot N = 0 \]
Line Equation Tests

\[ L(x, y) = V \cdot N < 0 \]
Point-in-Triangle Test: Three Line Tests

\[ P_i = (X_i, Y_i) \]

\[ dX_i = X_{i+1} - X_i \]
\[ dY_i = Y_{i+1} - Y_i \]

\[ L_i(x, y) = -(x - X_i) dY_i + (y - Y_i) dX_i \]

\[ = A_i x + B_i y + C_i \]

\[ L_i(x, y) = 0 : \text{point on edge} \]
\[ < 0 : \text{outside edge} \]
\[ > 0 : \text{inside edge} \]

Compute line equations from pairs of vertices
Point-in-Triangle Test: Three Line Tests

\[ P_i = (X_i, Y_i) \]

\[ dX_i = X_{i+1} - X_i \]
\[ dY_i = Y_{i+1} - Y_i \]

\[ L_i(x, y) = -(x - X_i) dY_i + (y - Y_i) dX_i \]

\[ = A_i x + B_i y + C_i \]

\[ L_i(x, y) = 0 \text{ : point on edge} \]
\[ < 0 \text{ : outside edge} \]
\[ > 0 \text{ : inside edge} \]

\[ L_0(x, y) > 0 \]

Slide courtesy of Prof. Ren Ng, UC Berkeley

CS180, Winter 2020

Lingqi Yan, UC Santa Barbara
Point-in-Triangle Test: Three Line Tests

\[ P_i = (X_i, Y_i) \]
\[ dX_i = X_{i+1} - X_i \]
\[ dY_i = Y_{i+1} - Y_i \]
\[ L_i(x, y) = -(x - X_i) \, dY_i + (y - Y_i) \, dX_i = A_i \, x + B_i \, y + C_i \]

\[ L_i(x, y) = 0 \quad \text{: point on edge} \]
\[ < 0 \quad \text{: outside edge} \]
\[ > 0 \quad \text{: inside edge} \]

\[ L_1(x, y) > 0 \]
Point-in-Triangle Test: Three Line Tests

\[ P_i = (X_i, Y_i) \]

\[ dX_i = X_{i+1} - X_i \]
\[ dY_i = Y_{i+1} - Y_i \]

\[ L_i(x, y) = -(x - X_i) dY_i + (y - Y_i) dX_i \]
\[ = A_i x + B_i y + C_i \]

\[ L_i(x, y) = 0 : \text{point on edge} \]
\[ < 0 : \text{outside edge} \]
\[ > 0 : \text{inside edge} \]

\[ L_2(x, y) > 0 \]
Point-in-Triangle Test: Three Line Tests

Sample point \( s = (sx, sy) \) is inside the triangle if it is inside all three lines.

\[
\text{inside}(sx, sy) = \quad \quad L_0(sx, sy) > 0 \quad \& \& \quad L_1(sx, sy) > 0 \quad \& \& \quad L_2(sx, sy) > 0;
\]

Note: actual implementation of \( \text{inside}(sx, sy) \) involves \( \leq \) checks based on edge rules.
Edge Cases (Literally)

Is this sample point covered by triangle 1, triangle 2, or both?

Slide courtesy of Prof. Ren Ng, UC Berkeley
Incremental Triangle Traversal (Faster?)
Signal Reconstruction on Real Displays
Real LCD Screen Pixels (Closeup)

iPhone 6S

Galaxy S5

Notice R,G,B pixel geometry! But in this class, we will assume a colored square full-color pixel.
Aside: What About Other Display Methods?

Color print: observe half-tone pattern

Slide courtesy of Prof. Ren Ng, UC Berkeley
Assume Display Pixels Emit Square of Light

Each image sample sent to the display is converted into a little square of light of the appropriate color:
(a pixel = picture element)

* LCD pixels do not actually emit light in a square of uniform color, but this approximation suffices for our current discussion
So, If We Send The Display This Sampled Signal
The Display Physically Emits This Signal
Compare: The Continuous Triangle Function
What’s Wrong With This Picture?

Jaggies!

Slide courtesy of Prof. Ren Ng, UC Berkeley
Aliasing (Jaggies)

Is this the best we can do?
Thank you!