Rendering

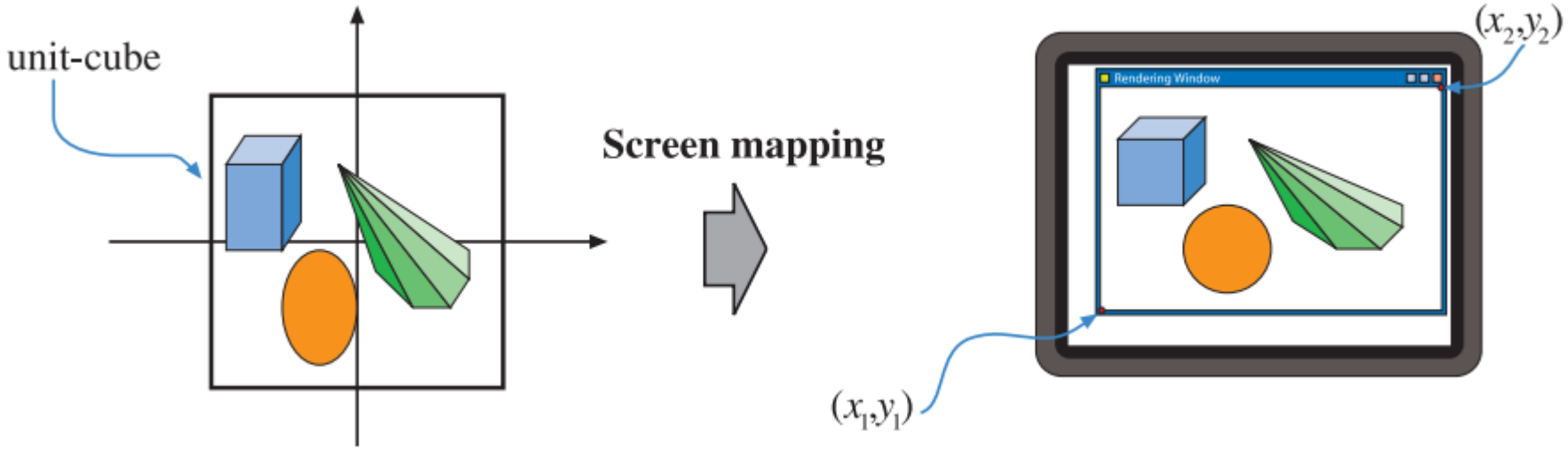
Triangle Rasterization

CS 415: Game Development

Professor Eric Shaffer

ILLINOIS

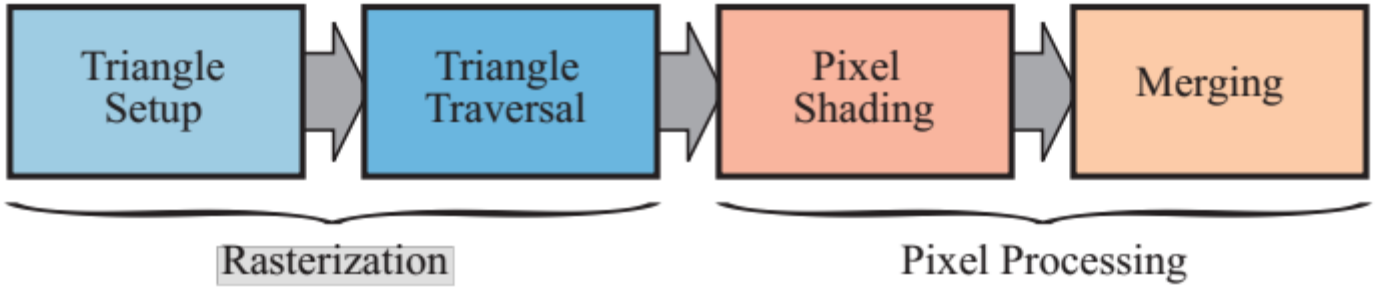
Rasterization and the Pipeline



- Rasterization happens after the viewport transformation
  - "screen mapping" = "viewport transformation"
- The vertex positions are now screen coordinates
- The z values for depth are also included

ILLINOIS

Triangle Rasterization



Triangle Setup

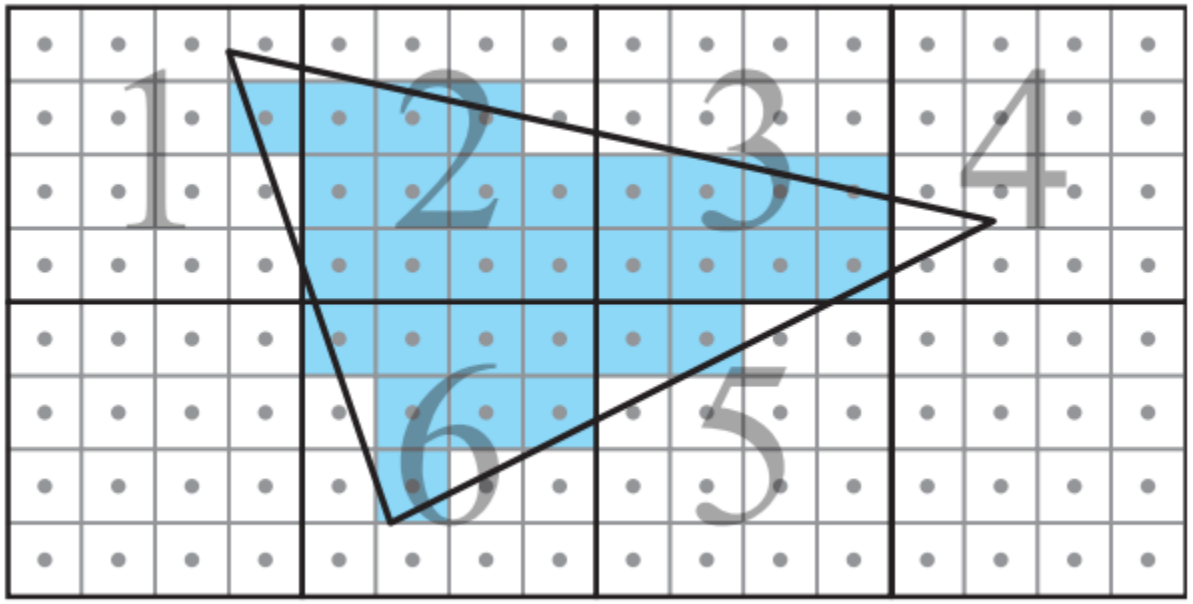
Edge equations and other data are computed.
This data may be used for triangle traversal, as well as for interpolation of other data.

Triangle Traversal

A fragment is generated for the part of the pixel that overlaps the triangle.
Data associated with each fragment is computed using interpolation.

ILLINOIS

Tiled Rasterization



Pixels are grouped into tiles (4x4 in this example)

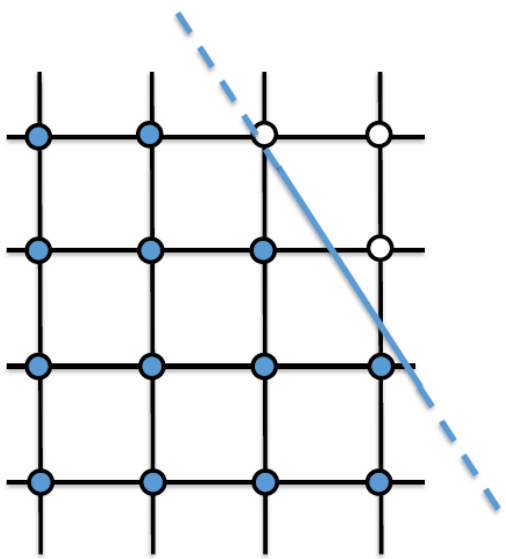
Tile processing has greater memory coherence than scanline

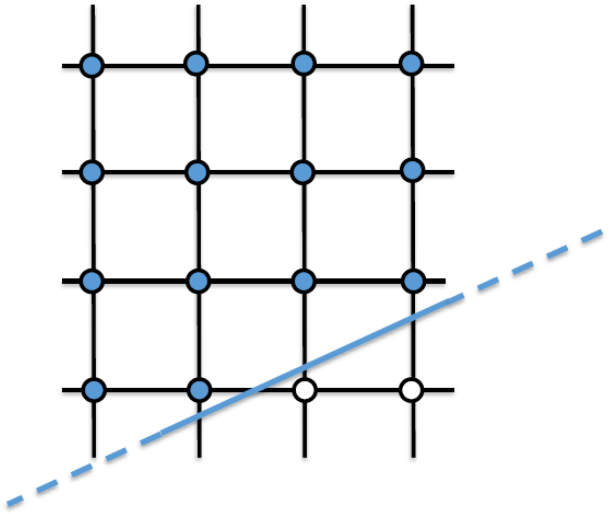
For example, texel data is cached more effectively

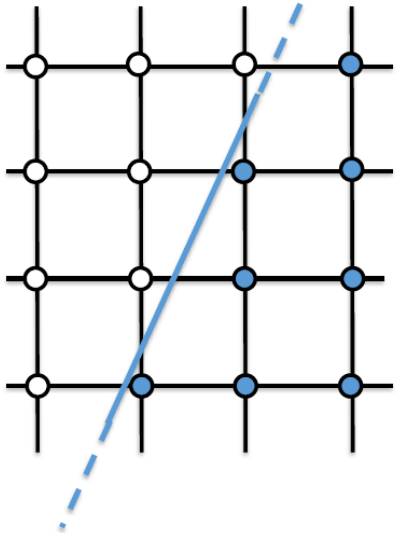
ILLINOIS

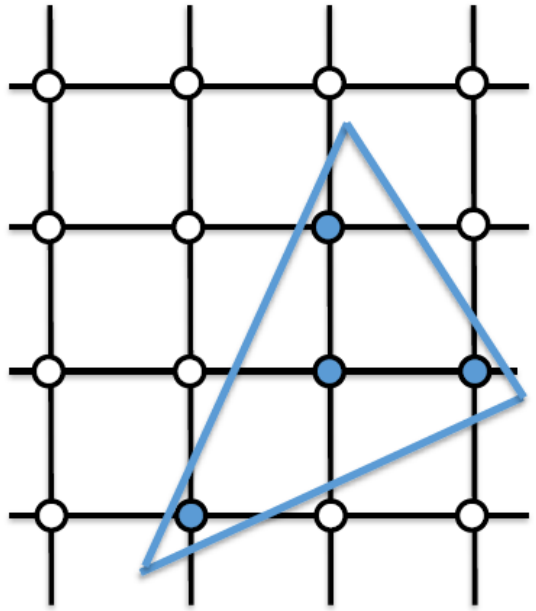
Basic Idea

Find which pixels lie on positive side of the three line equations









How can you test which side of a line a pixel falls on?

ILLINOIS

Edge Functions

Suppose the vertices of the triangle are p_0 p_1 and p_2

For each triangle edge compute a function

$$\mathbf{n} \cdot ((x, y) - \mathbf{p}) = 0$$

n is a normal vector pointing to inside of the triangle

p is a point on the line formed by the edge

Example: Edge Functions

$$\mathbf{n} \cdot ((x, y) - \mathbf{p}) = 0$$

Edge vector is $p_1 - p_0$

Normal is $p_1 - p_0$ rotated 90 degrees counterclockwise:

$$\mathbf{n}_2 = (-(p_{1y} - p_{0y}), p_{1x} - p_{0x})$$

n_2 points to the inside of the triangle

inserting n_2 and p_0 into the edge function we have

$$\begin{aligned} e_2(x, y) &= -(p_{1y} - p_{0y})(x - p_{0x}) + (p_{1x} - p_{0x})(y - p_{0y}) \\ &= -(p_{1y} - p_{0y})x + (p_{1x} - p_{0x})y + (p_{1y} - p_{0y})p_{0x} - (p_{1x} - p_{0x})p_{0y} \\ &= a_2 x + b_2 y + c_2. \end{aligned}$$

ILLINOIS

Optimizations

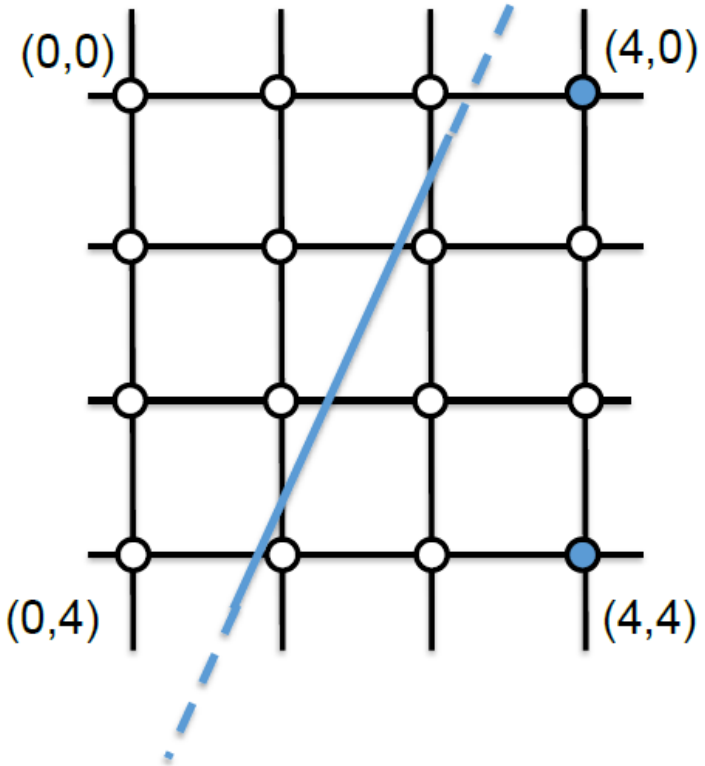
- Can use fixed point screen space coordinates (integer arithmetic)
- Can do incremental updates to edge functions when testing pixels
  - Example: Suppose we test tested a pixel center at (x,y)
    Then, to test pixel center at (x+1,y) we have

$$e(x+1,y) = a(x+1) + by + c = a + ax + by + c = a + e(x,y)$$

Optimizations

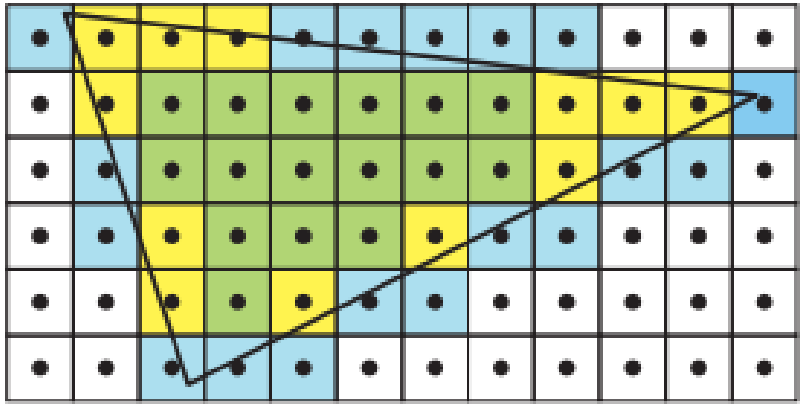
Can quickly check if an entire tile lies outside triangle

- If all corners of tile outside triangle → tile is outside
- If all corners of tile inside → tile is inside
- Otherwise, an edge passes through the tile



OIS

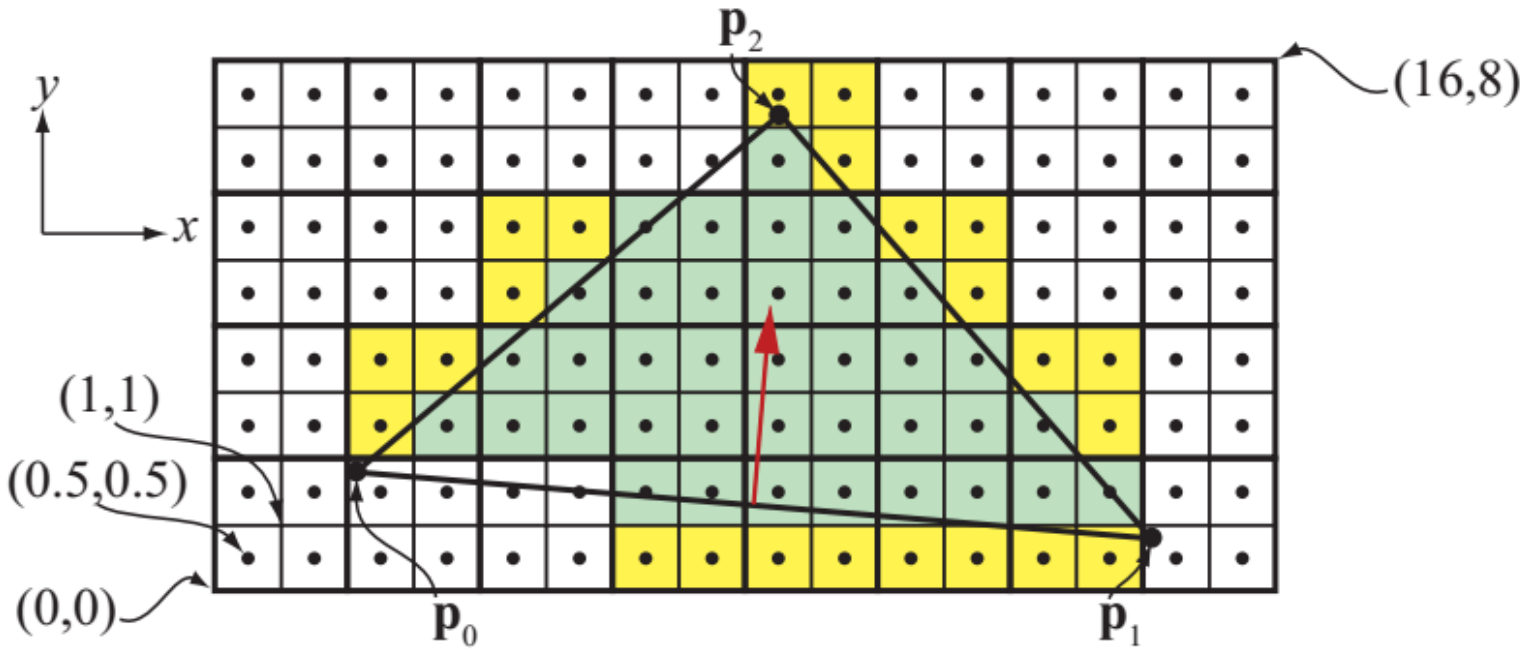
Conservative Rasterization



Outer Conservative Rasterization: all colored pixels (any pixel partially or fully in triangle)
Inner Conservative Rasterization: green pixels (pixels fully inside triangle)

ILLINOIS

Triangle Rasterization



- Fragments grouped into quads for texturing and other operations
- Yellow fragments are “helper fragments”

ILLINOIS