Render [render]:
To generate an image or animation.
Two Ways to Make an Image

1. Drawing

2. Photography

http://www.publicdomainpictures.net/view-image.php?picture=female-photographer&image=4360&large=1
Two Ways to Render an Image

Rasterization

http://upload.wikimedia.org/wikipedia/commons/a/ab/Bresenham.svg
Two Ways to Render an Image

Rasterization

http://upload.wikimedia.org/wikipedia/commons/a/ab/Bresenham.svg
Two Ways to Render an Image

Ray Tracing

Diagram: A camera is depicted viewing a scene object. A light source illuminates the scene, casting shadows. Rays are traced from the camera through the scene to determine the final image.
Two Ways to Render an Image

Camera → View Ray → Image → Scene Object → Shadow Ray → Light Source

Ray Tracing

http://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg
For Today

Rasterization

http://upload.wikimedia.org/wikipedia/commons/a/ab/Bresenham.svg
Rasterize [rastərɪz]:
To convert vector data to raster format.
“A triangle is here, a circle is there, ...”

“This pixel is yellow...”
“A triangle is here, a circle is there, …”

“This pixel is yellow…”

Rasterization Process
Figuring out which pixels to shade.
The Basics

Framebuffer
# The Basics

<table>
<thead>
<tr>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
<th>Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
</tr>
<tr>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
</tr>
<tr>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
</tr>
<tr>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
<td>Pixel</td>
</tr>
</tbody>
</table>
The Basics

Scan Conversion
The Basics

Scan Conversion
Framebuffer coordinates

(0,0) (0,1) (1,0) (1,1)
The Basics

Framebuffer coordinates (0,0) +x +y

Pixel center?

Framebuffer coordinates
Problem

Where do lines go?
Must intersect diamond
Must intersect diamond
Desirable Properties

- Fast
- Simple
- Integer arithmetic
Bresenham’s Algorithm

- Introduced in 1967
- Best-fit approximation under some conditions
Bresenham’s Algorithm

- Introduced in 1967
  
  Variation by Pitteway (1967): “Midpoint Algorithm”

- Best-fit approximation under some conditions
0 \cdot m < 1
Easy fix via rotation/reflection

\[ 0.1 < m \leq 1 \]
Which pixel is next?
Bresenham’s Algorithm

Which pixel is next? Me!
Bresenham’s Algorithm

\[ 0 \cdot m < 1 \]
Bresenham’s Algorithm

$d < 0$
Bresenham’s Algorithm

\[ d_2 = d_1 + m \]
Bresenham’s Algorithm

\[ d_2 = d_1 + \frac{\Delta y}{\Delta x} \]

\[ \Delta_x d_2 = \Delta_x d_1 + \Delta y \]
Bresenham’s Algorithm

\[ v_2 = v_1 + \Delta y \]

“Decision variable”

\[ v = \Delta_x d \]
Bresenham’s Algorithm

\[ \tilde{v}_2 = v_2 - \Delta x \]

\[ v_2 = v_1 + \Delta y \]
Given $v_1$:

1. $v_2 = v_1 + \Delta y$

2. If $v_2 \cdot 0$, move east.

3. If $v_2 > 0$, replace with $\tilde{v}_2 = v_2 - \Delta x$ and move northeast.
Given $v_1$:

1. $v_2 = v_1 + \Delta y$

2. If $v_2 \cdot 0$, move east.

3. If $v_2 > 0$, replace with $\tilde{v}_2 = v_2 - \Delta x$ and move northeast.

- Fast
- Simple
- Integer arithmetic
Works (with modification)
Interesting Fact

Works (with modification)

Homework 1
Important Issues We’ve Ignored
Important Issues We’ve Ignored

$$l \approx 7.07$$

$$p = 5$$

$$l = 5$$

$$p = 5$$

Line intensity
Important Issues We’ve Ignored

Line thickness
Important Issues We’ve Ignored

Antialiasing
Important Issues We’ve Ignored

Antialiasing
Important Issues We’ve Ignored

Antialiasing
Filling Triangles

- Optimize a single primitive
- Nice properties:
  - Planar
  - “Inside” well-defined
  - Straightforward shading
Have We Lost Anything?
Have We Lost Anything?

Non-convex!
Have We Lost Anything?

<CS 268 />
Convention

Must contain center
Two Methods for Filling

Check if each pixel in bounding box is inside the triangle.

Rasterize border; sweep from left to right.
Two Methods for Filling

Check if each pixel in bounding box is inside the triangle.  
Parallelizable

Rasterize border; sweep from left to right.  
Less math
More Issues

Adjacency & singularities
More Issues

Small triangles & slivers