Image Formation

- Light Source
- Camera
- Image
- World
Image Formation

Light Source

Camera

Image

World

Incident Ray

Reflected Ray

Transmitted Ray
Image Formation

- Point Light Source
- Incident Ray
- Reflected Ray
- Transmitted Ray
- Camera
- Image
- World
Image Formation

In this model can you reason about:
- Shadows?
- Area light sources?
- Diffraction?
- Energy transfer?
Image Formation

Point Light Source

Incident Ray

Reflected Ray

Camera

Image Model

World

Transmitted Ray
Image Model

How is this image drawn on the computer screen?
An image is an array of raster elements called pixels.
Image Model

An image is an array of *raster* elements called *pixels*. Every pixel has at least a colour value.
Image Model

Let's take a closer look.
Image Model

How is the sphere drawn using the pixels?
To draw a geometrical figure...
...we assign the *correct* pixels with the correct colour. This process is called *rasterization*. 
Continue the pixel colouring to get regions filled with colour.
Why this image model?

The framebuffer is a memory buffer storing the colour value for each pixel displayed.

Display

Framebuffer

Processor

Input
Why this image model?

So the image model mimics the memory model from hardware.
Why this image model?

So the image model mimics the memory model from hardware.

An alternate image model is a vector image model.
How to colour the correct pixels?

```
function line(int x0, int x1, int y0, int y1)
    int deltax = x1 - x0
    int deltay = y1 - y0
    float error = 0
    float deltaerr = deltay / deltax
    // Assume deltax != 0 (line is not vertical),
    // note that this division needs to be done in a way
    // that preserves the fractional part
    int y = y0
    for x = x0 to x1
        plot(x,y)
        error = error + deltaerr
        if error ≥ 0.5 then
            y = y + 1
            error = error - 1.0
```
How to colour the correct pixels?

- Extension for all line directions.
- Optimize.
- Demo
- Curves - read!

CS475/CS675 - Lecture 1

Bresenham's Line Drawing Algorithm

```c
function line(int x0, int x1, int y0, int y1)
    int deltax = x1 - x0
    int deltay = y1 - y0
    float error = 0
    float deltaerr = deltay / deltax
    // Assume deltax != 0 (line is not vertical),
    // note that this division needs to be done in a way
    // that preserves the fractional part
    int y = y0
    for x = x0 to x1
        plot(x,y)
        error = error + deltaerr
        if error ≥ 0.5 then
            y = y + 1
            error = error - 1.0
```
How to fill pixels?
How to fill pixels?
How to fill pixels?
How to fill pixels?

Scanfill Algorithm
Scanfill Algorithm

The Edge List

<table>
<thead>
<tr>
<th>Edge</th>
<th>$Y_{\text{min}}$</th>
<th>$Y_{\text{max}}$</th>
<th>$X$ for $Y=Y_{\text{min}}$</th>
<th>$1/m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_2$</td>
<td>$V_3$</td>
<td>$V_2$</td>
<td>$X_3$</td>
<td>$1/m_2$</td>
</tr>
<tr>
<td>$e_3$</td>
<td>$V_3$</td>
<td>$V_4$</td>
<td>$X_3$</td>
<td>$1/m_3$</td>
</tr>
<tr>
<td>$e_1$</td>
<td>$V_1$</td>
<td>$V_2$</td>
<td>$X_1$</td>
<td>$1/m_1$</td>
</tr>
<tr>
<td>$e_5$</td>
<td>$V_1$</td>
<td>$V_5$</td>
<td>$X_1$</td>
<td>$1/m_5$</td>
</tr>
<tr>
<td>$e_4$</td>
<td>$V_4$</td>
<td>$V_5$</td>
<td>$X_4$</td>
<td>$1/m_4$</td>
</tr>
</tbody>
</table>

- Edges in the edge list become *active* when the $y$-coordinate of the current scan line matches their $Y_{\text{min}}$ value.
- First intersection point between an active edge and a scan line is always the endpoint corresponding to $Y_{\text{min}}$. 
Scanfill Algorithm

- For monotonically increasing/decreasing edges across a shared vertex count *one* intersection.
- Else count *two*.
- Ignore horizontal edges.