CS475m - Computer Graphics

Lecture 1 : Rasterization Basics

Image Formation

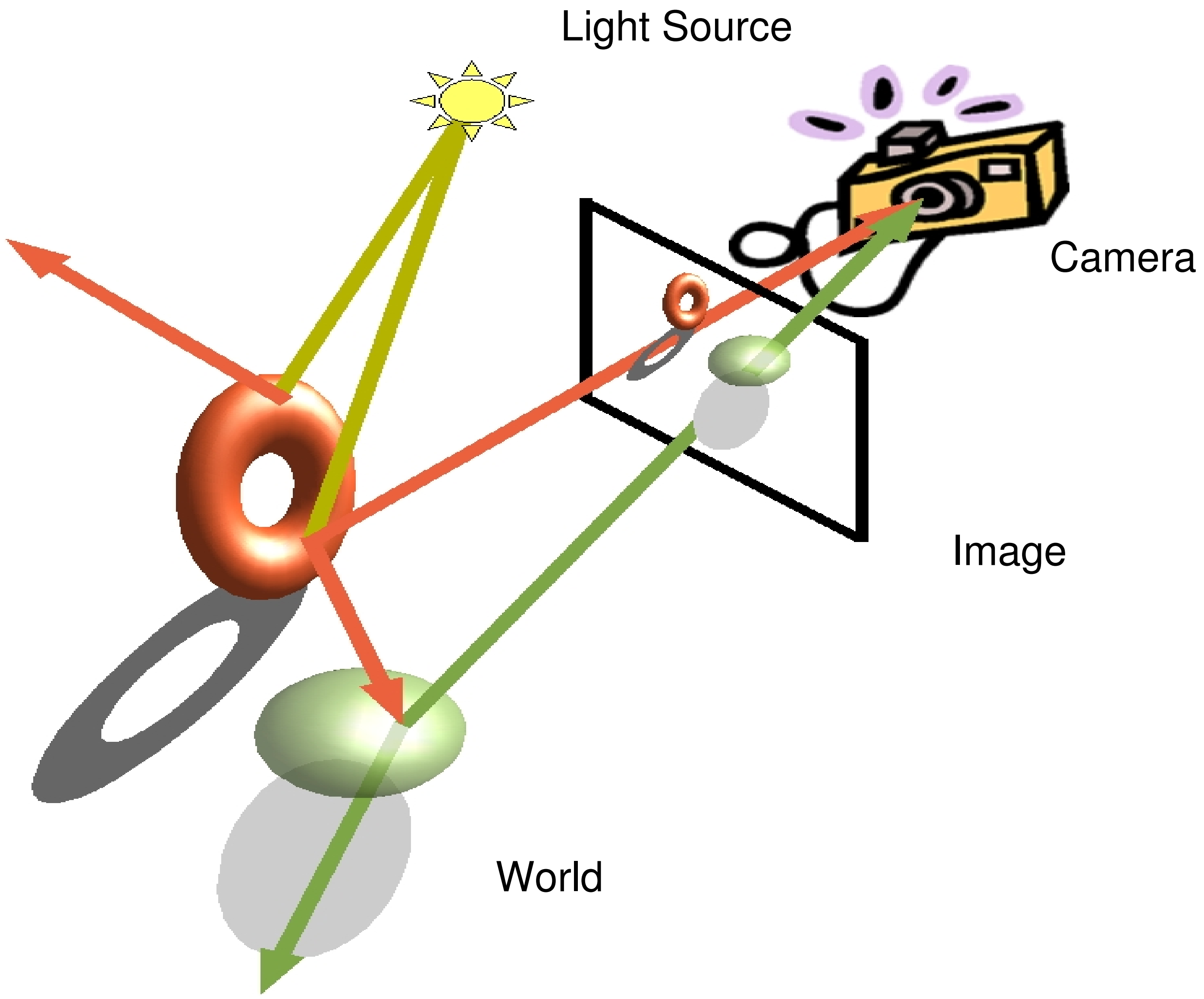


Image Formation

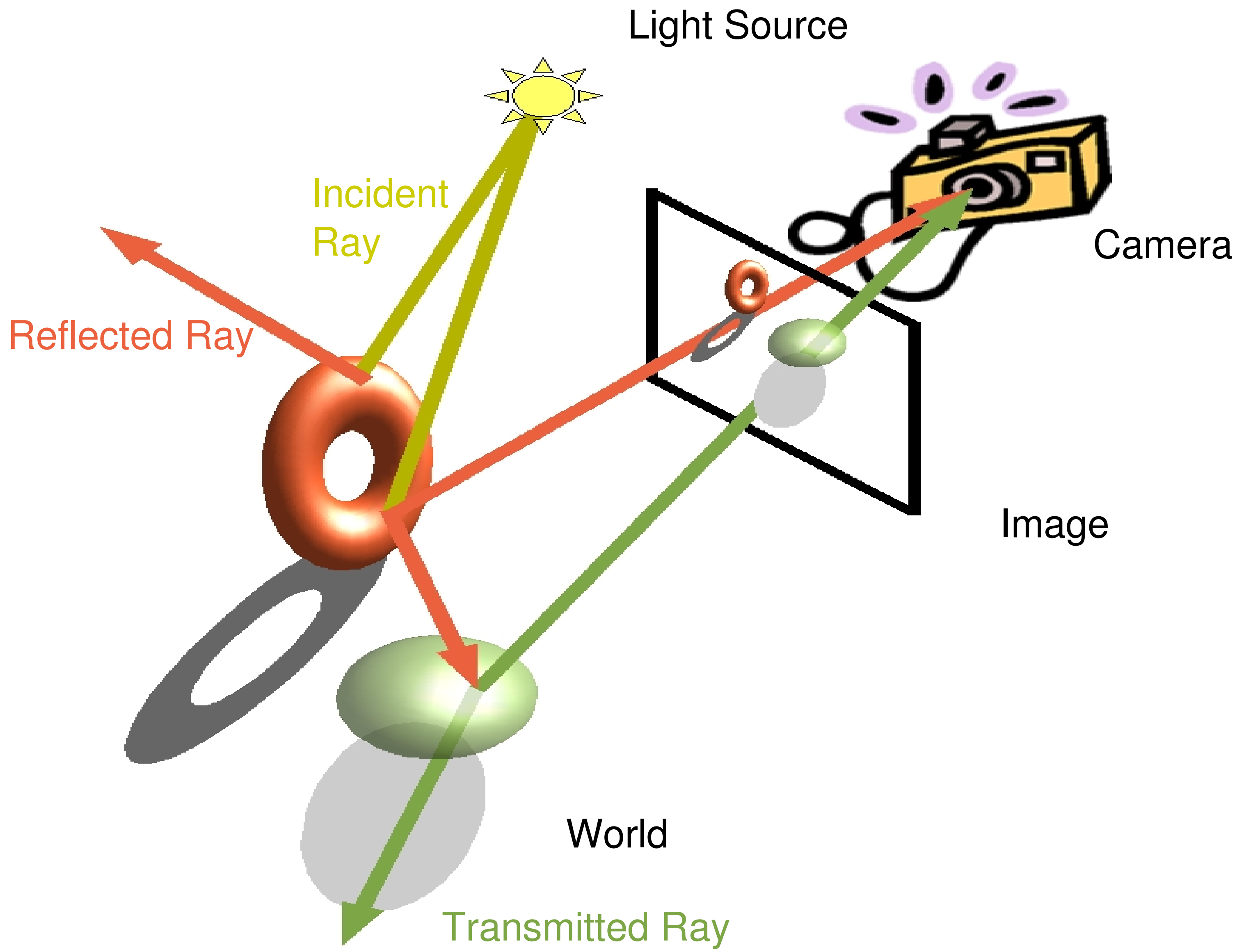


Image Formation

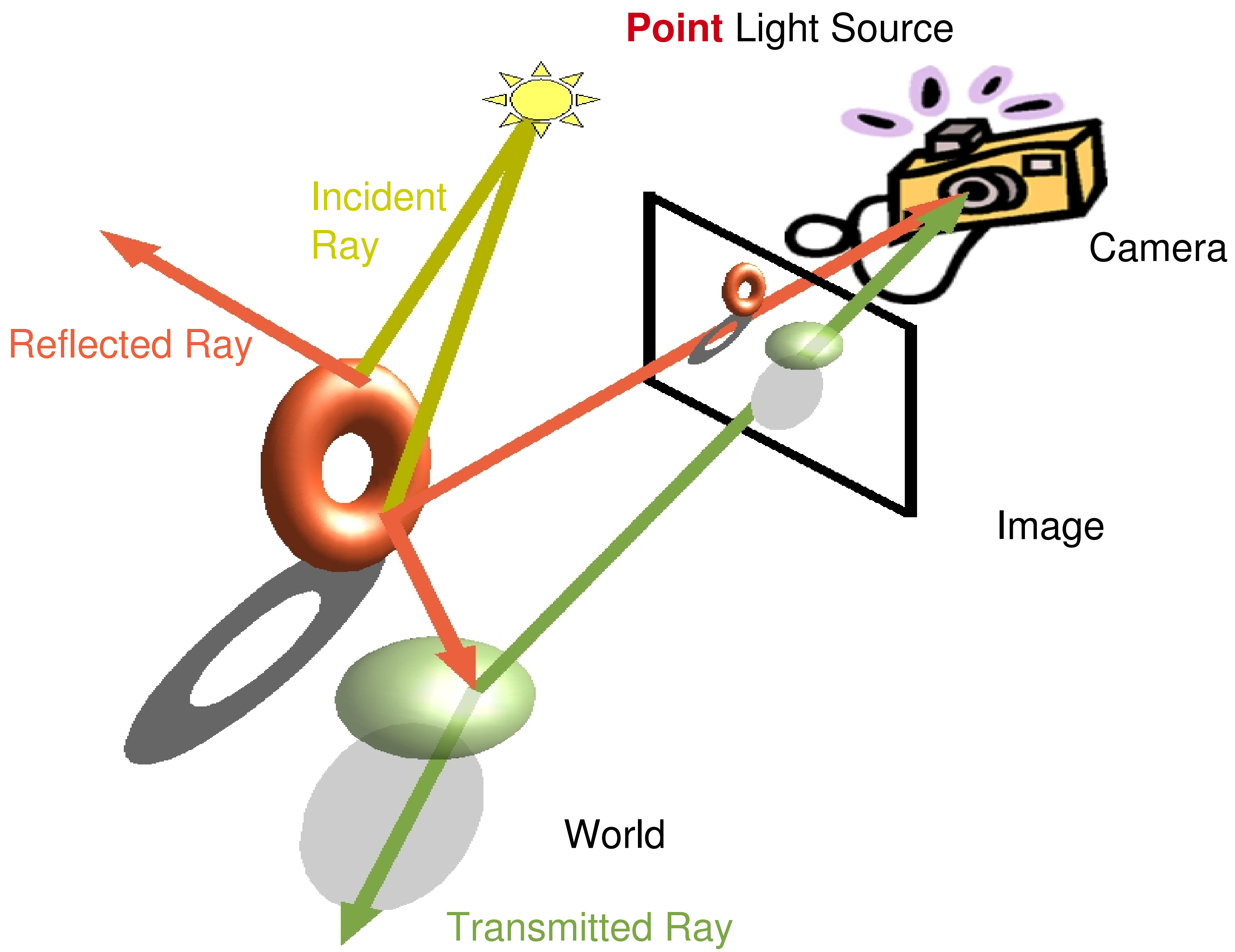
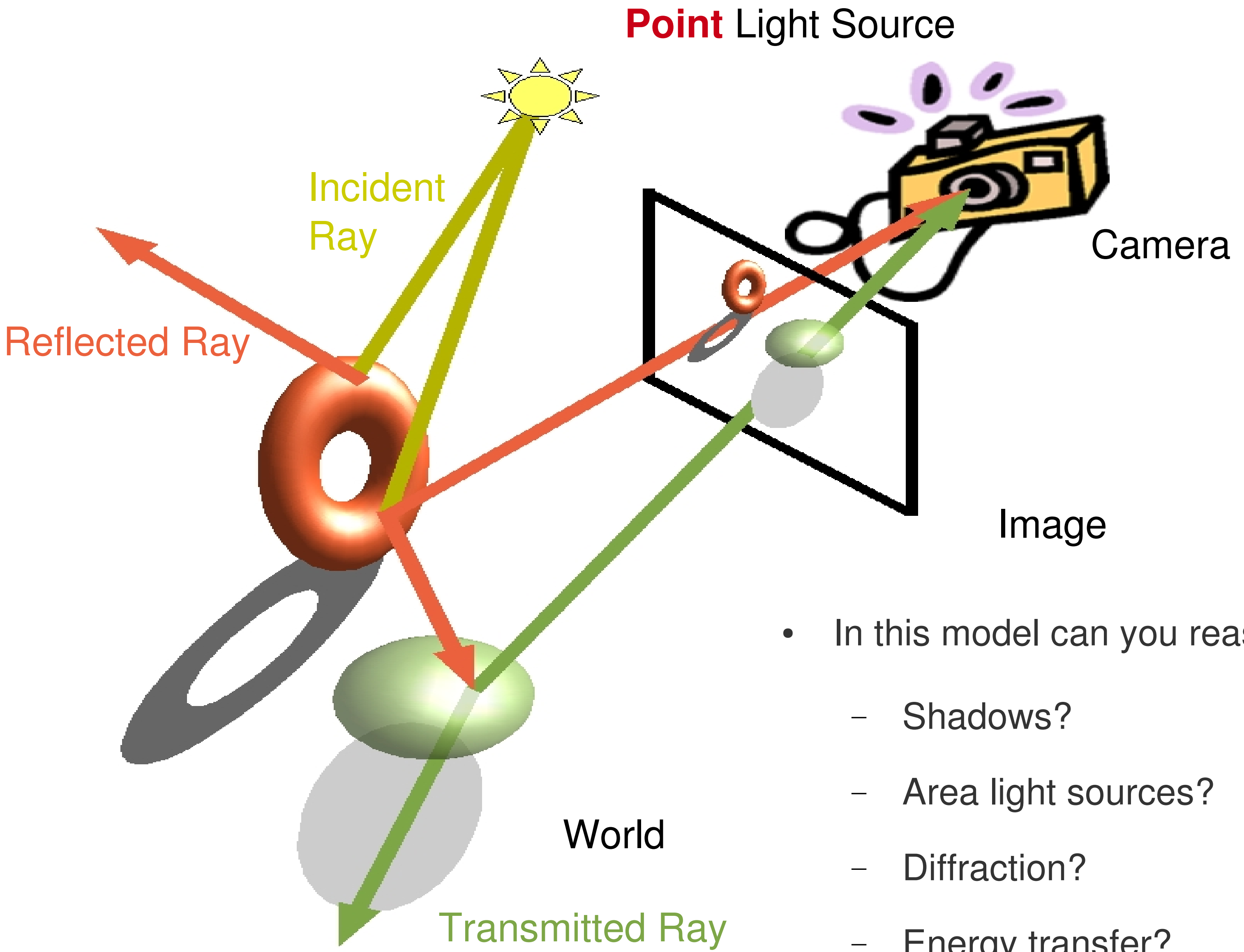


Image Formation



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

Image Formation

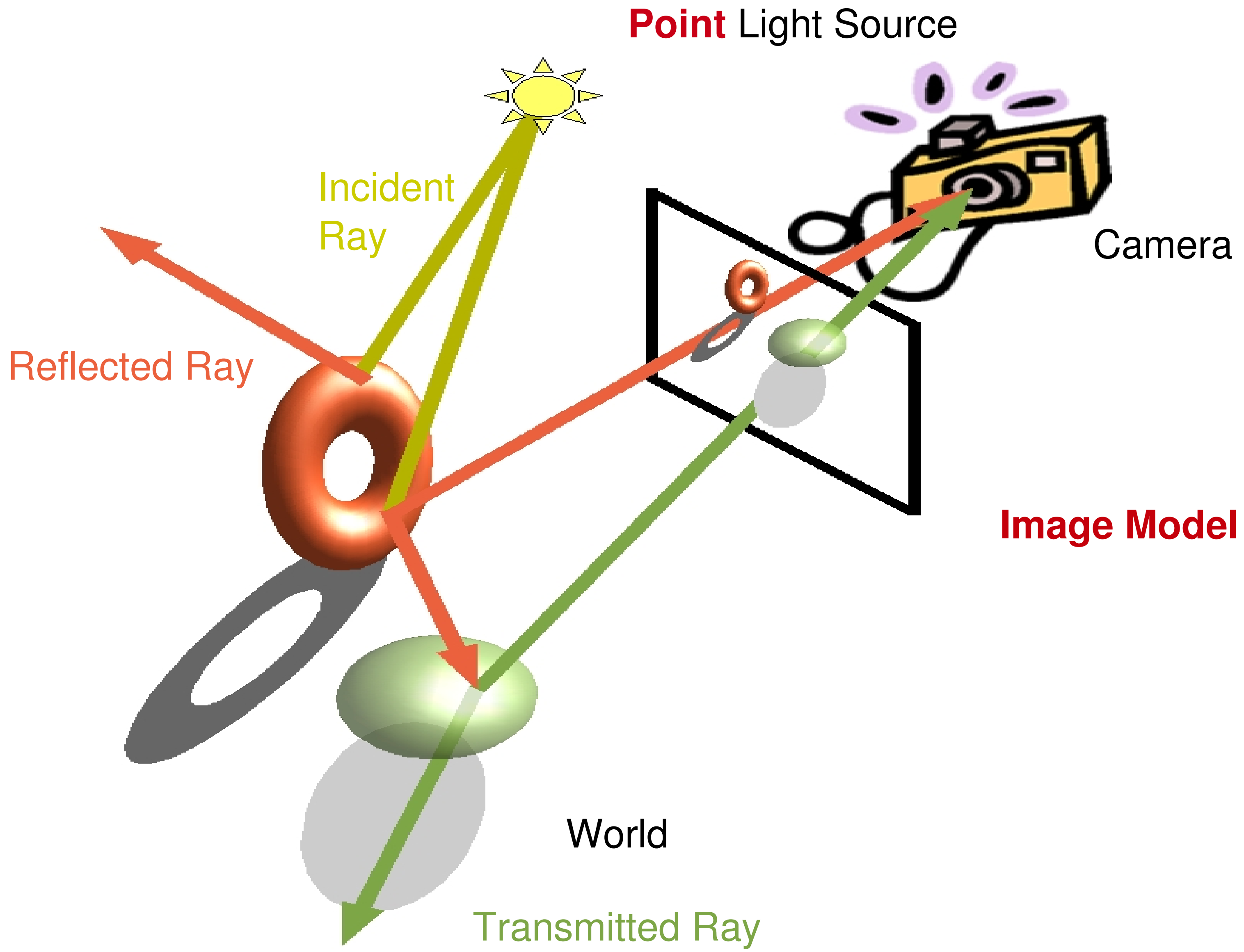


Image Model

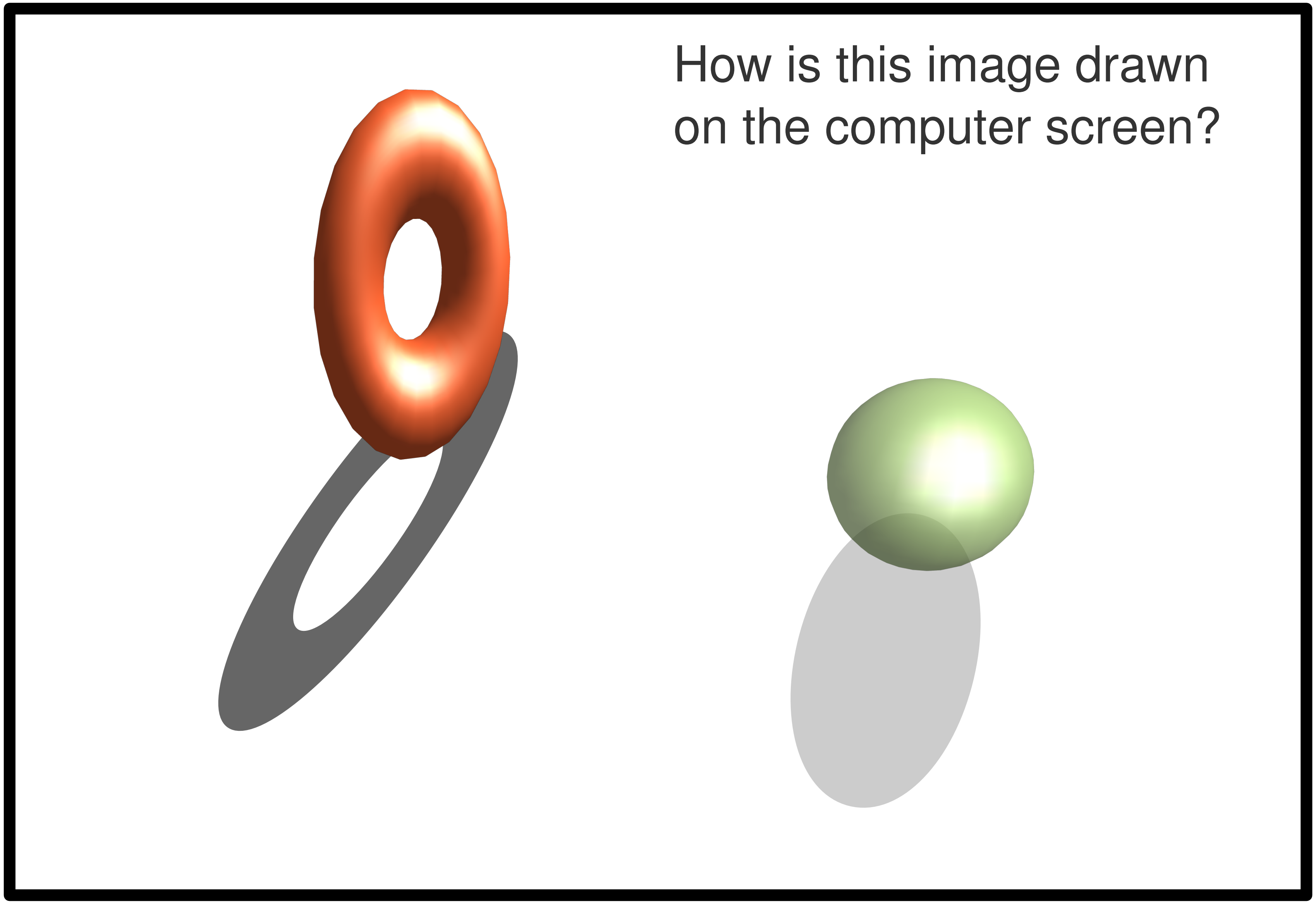
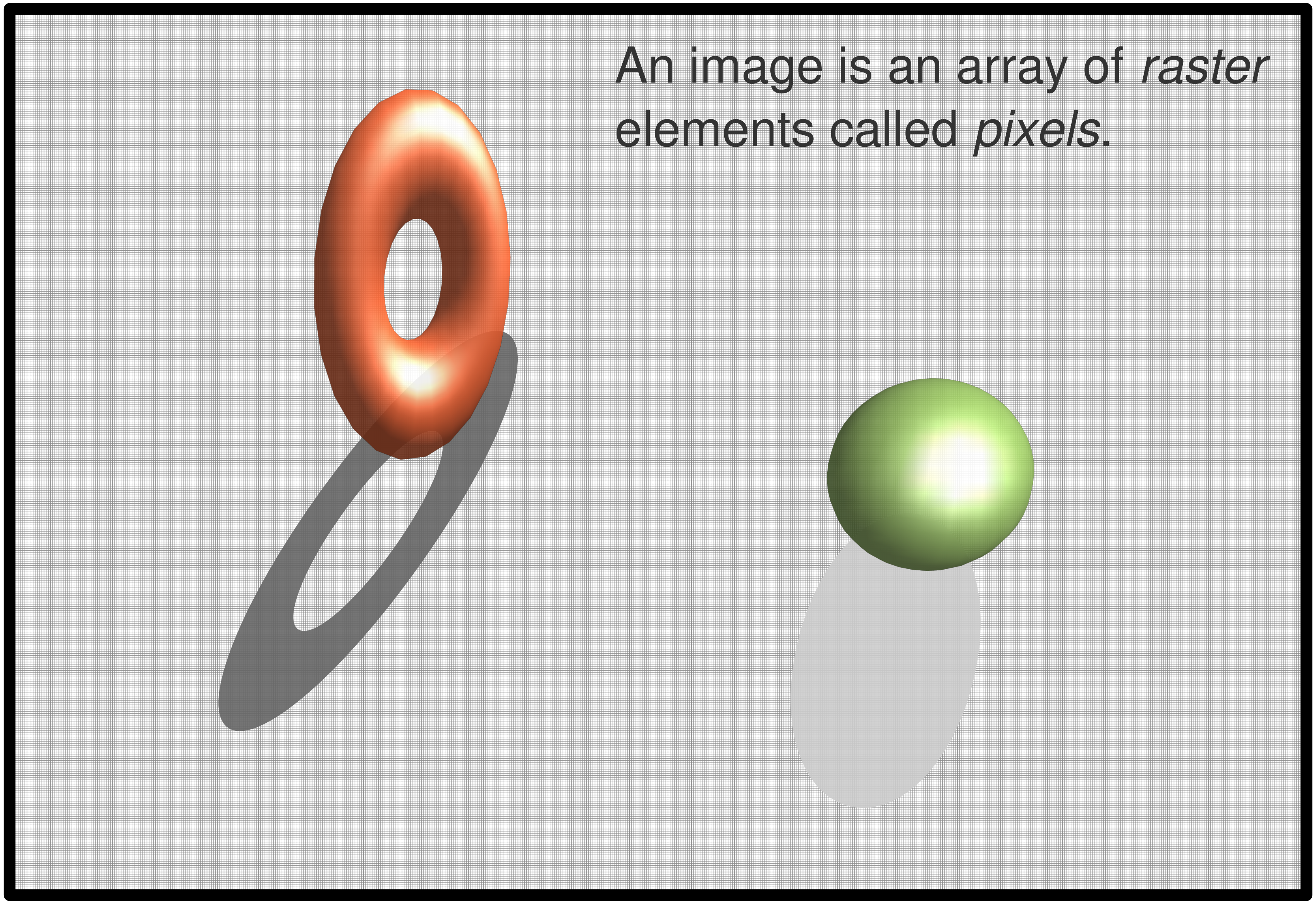
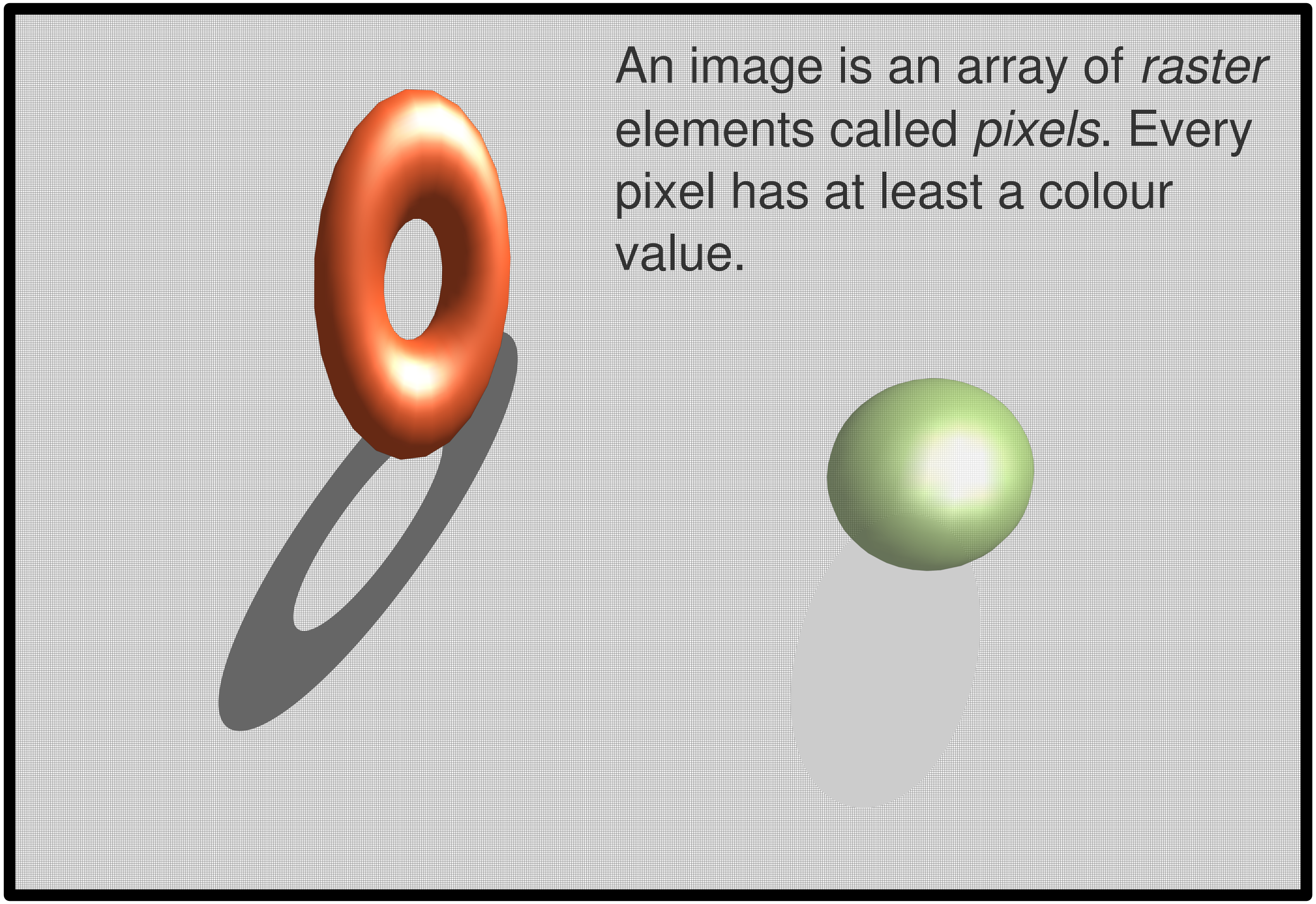

How is this image drawn on the computer screen?

Image Model



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

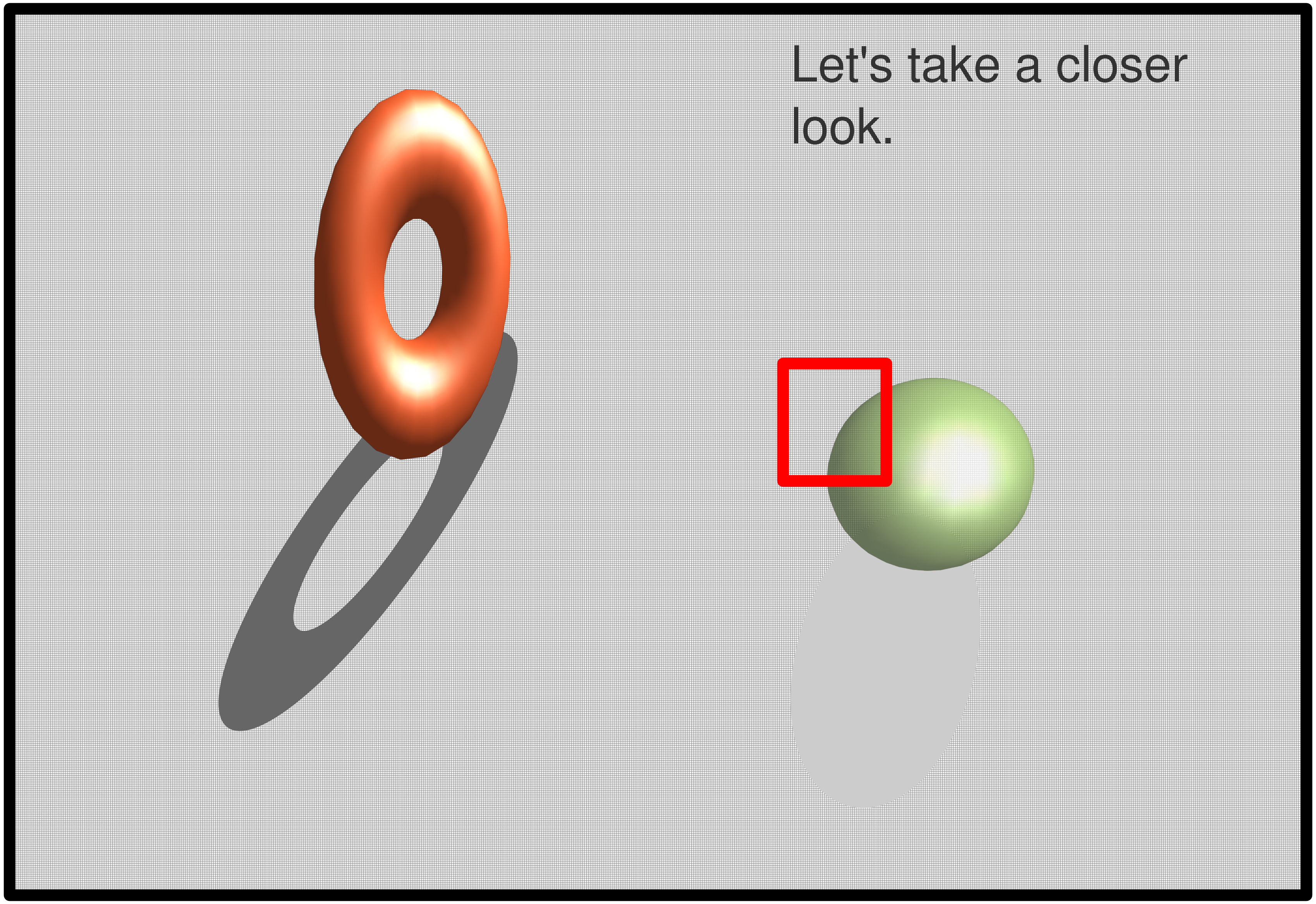
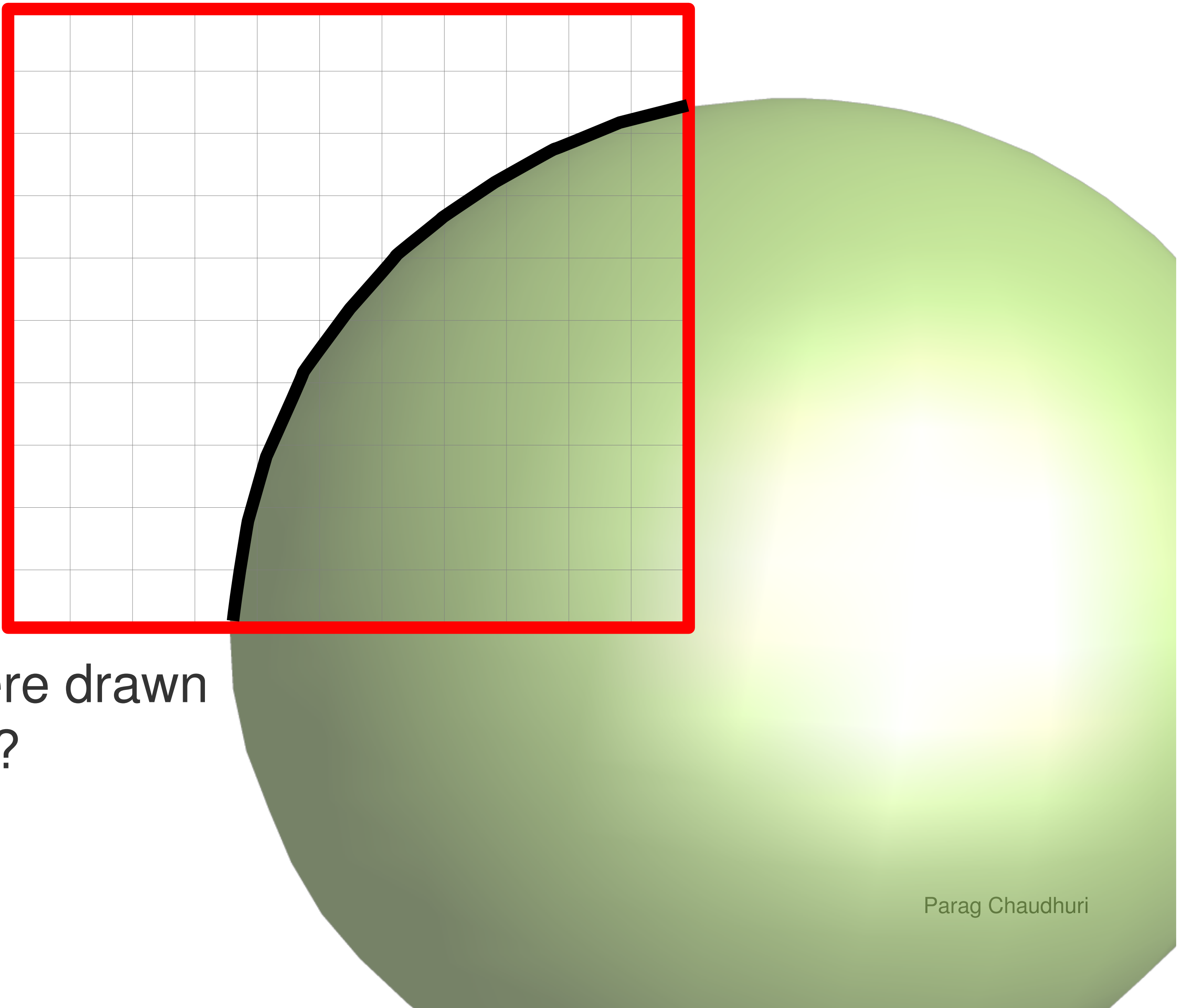
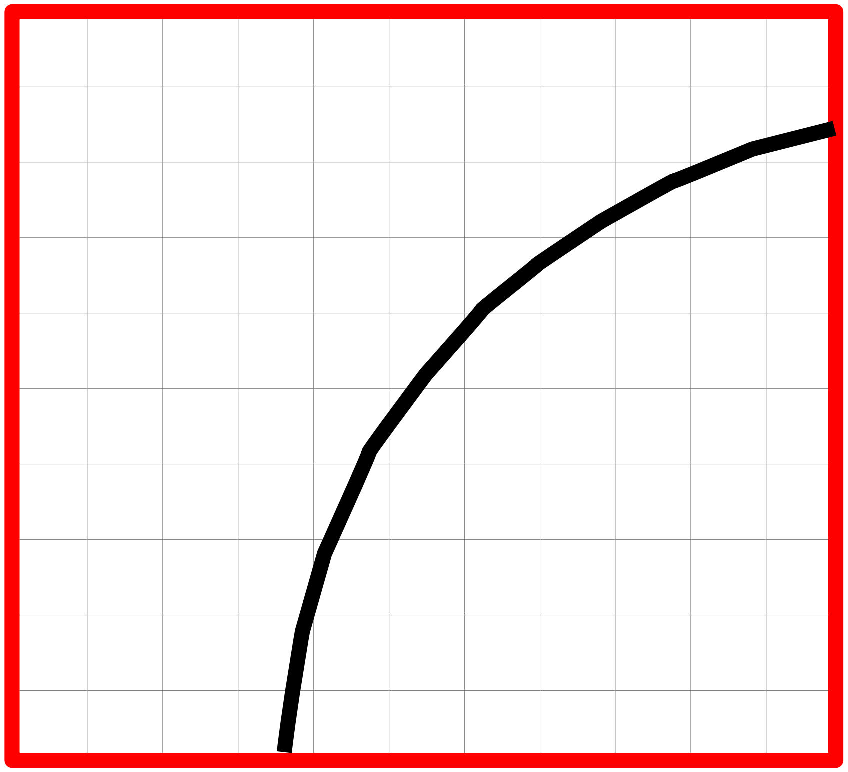


Image Model



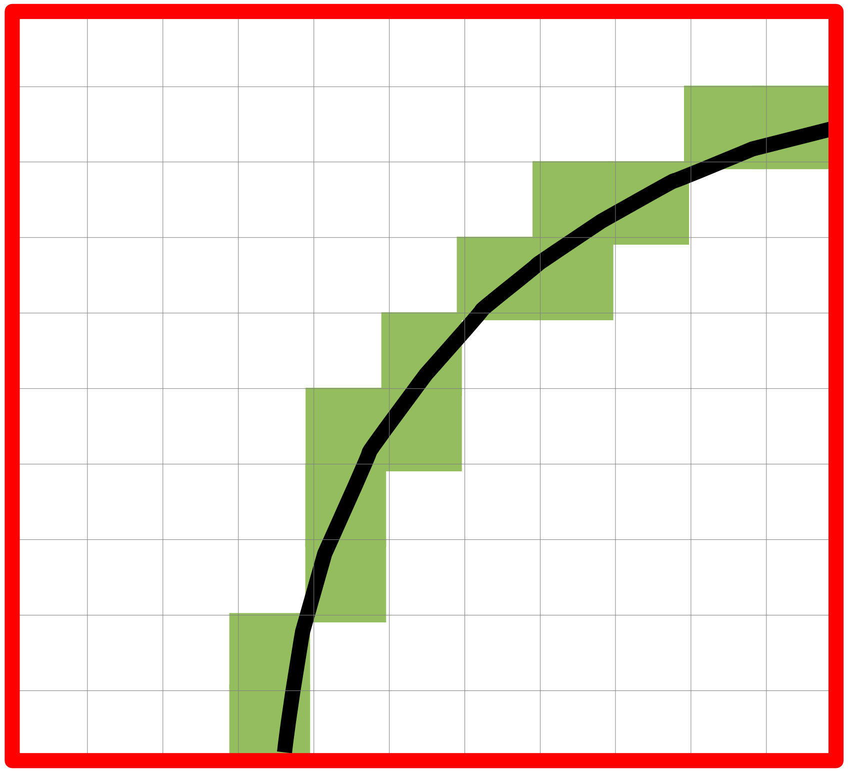
How is the sphere drawn using the pixels?

Image Model



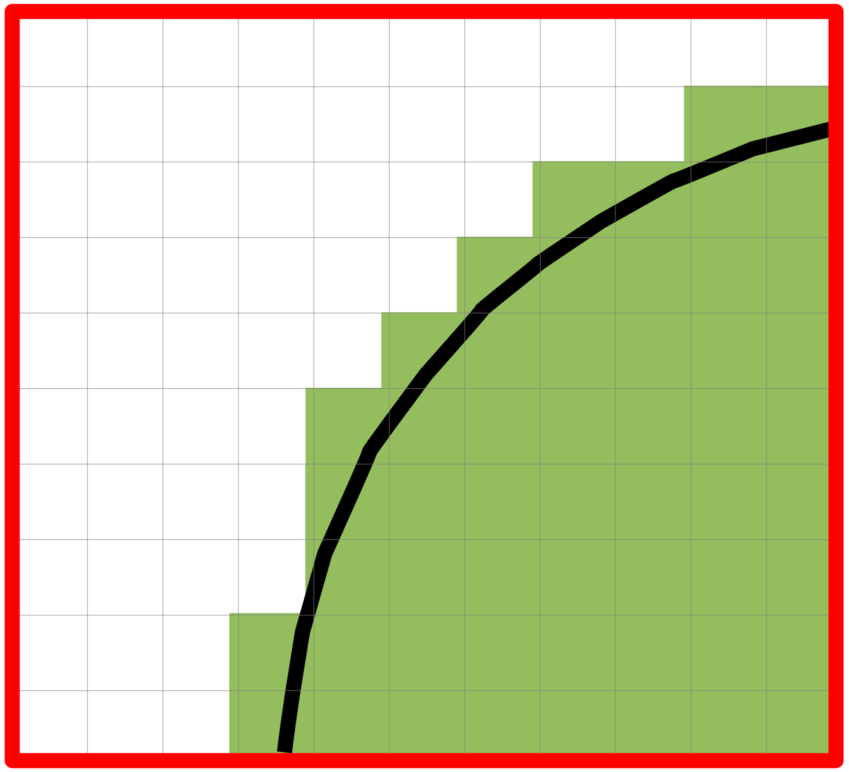
To draw a geometrical figure...

Image Model



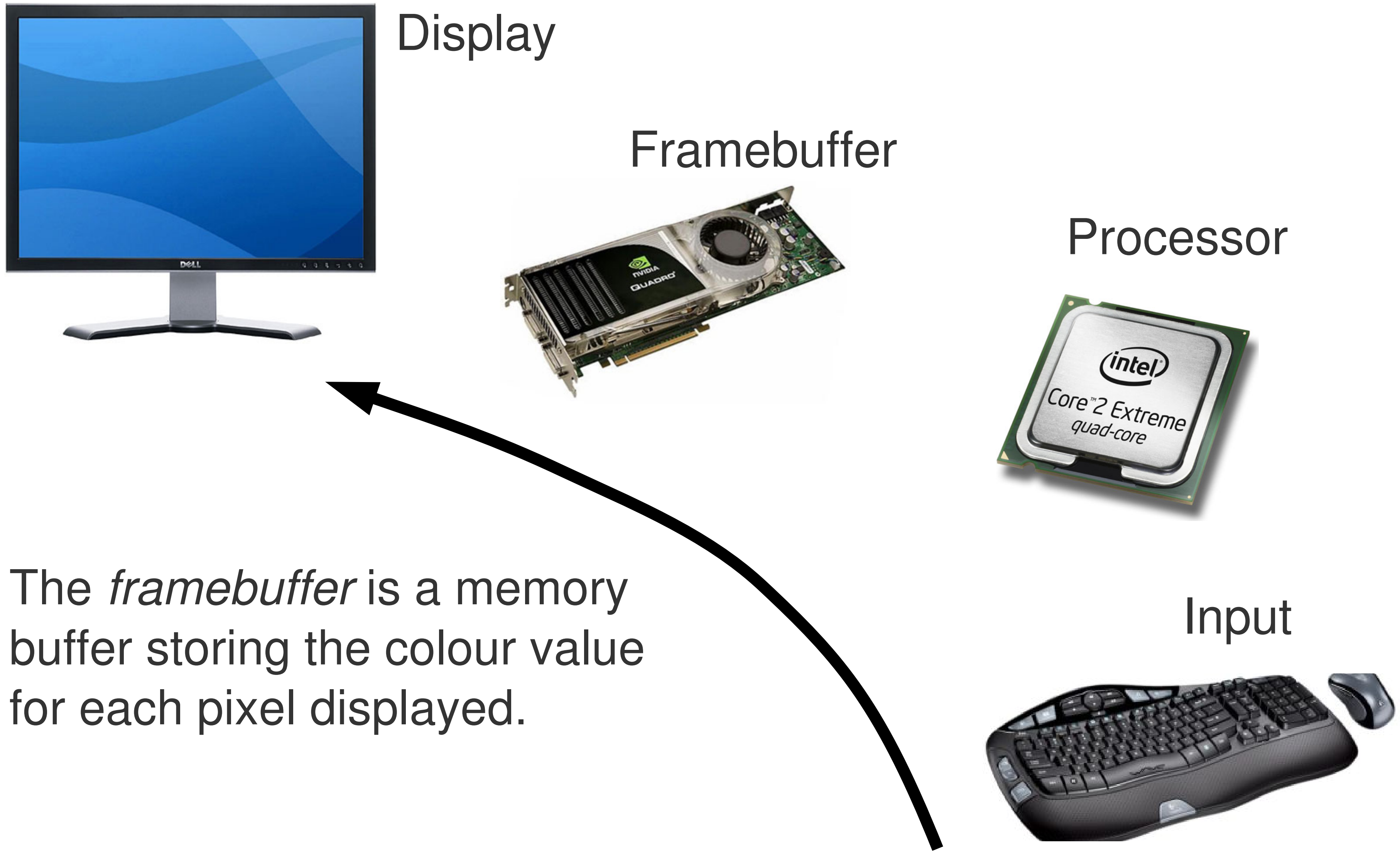
...we assign the *correct* pixels with the correct colour. This process is *rasterization.*

Image Model



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.

Why this image model?

Display

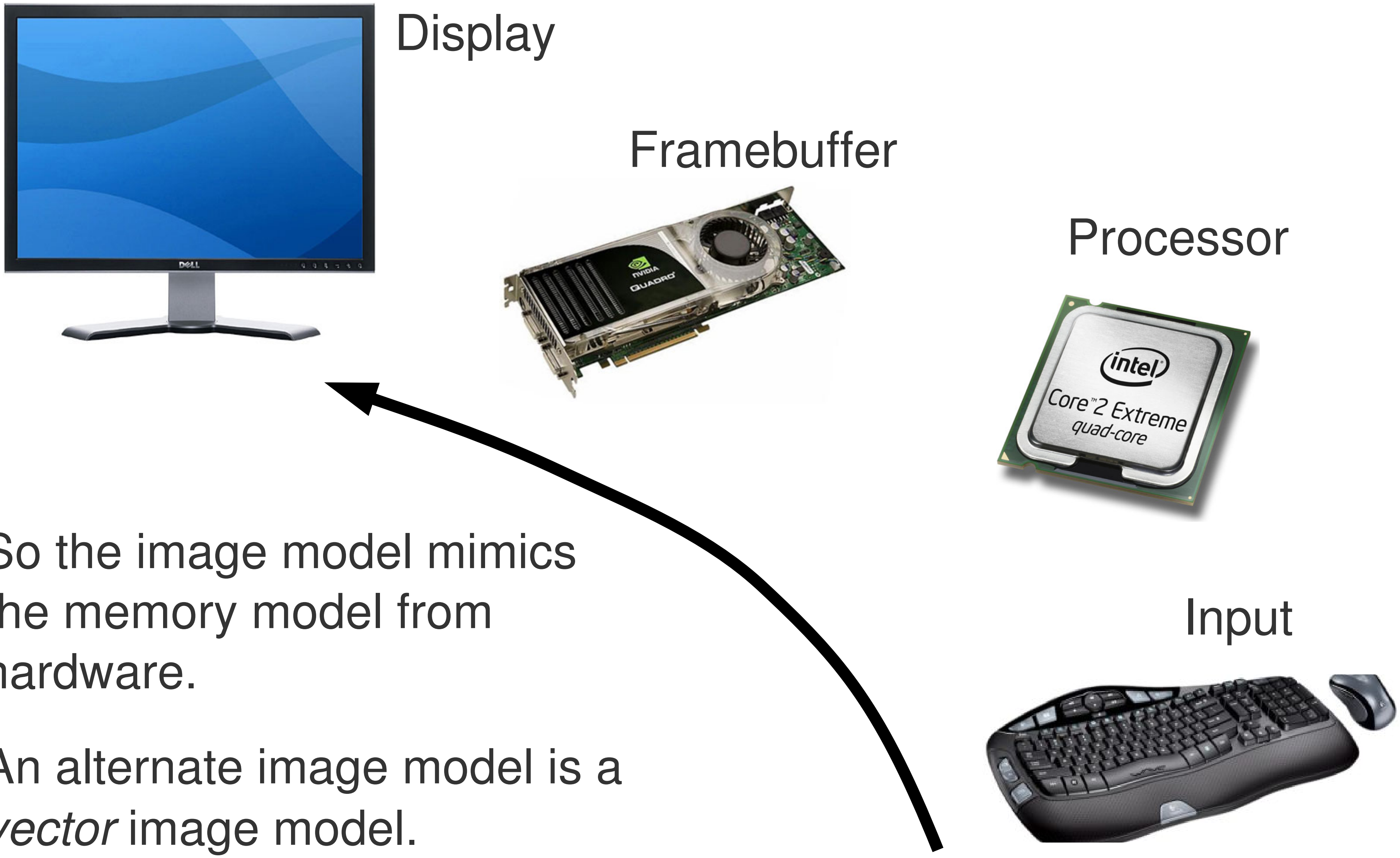
Framebuffer

Processor

Input

So the image model mimics the memory model from hardware.

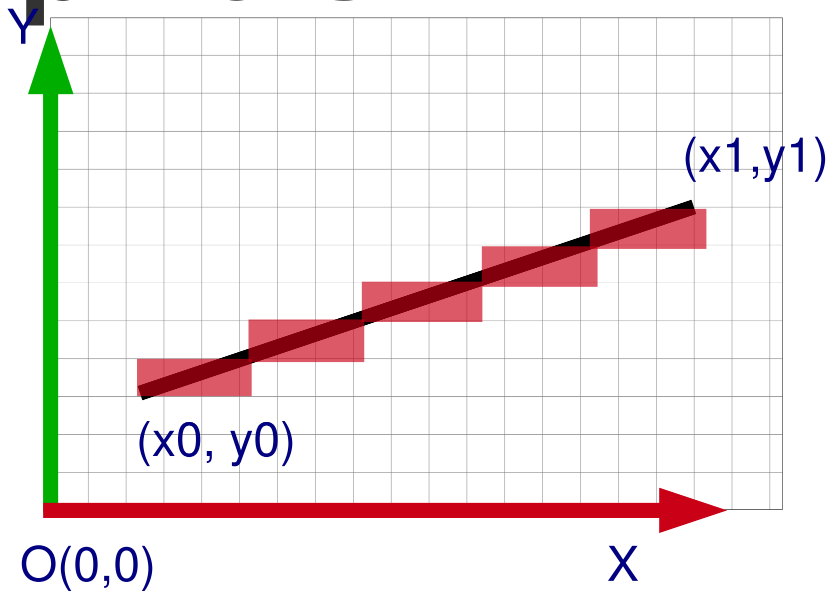
Why this image model?



So the image model mimics the memory model from hardware.

An alternate image model is a *vector* image model.

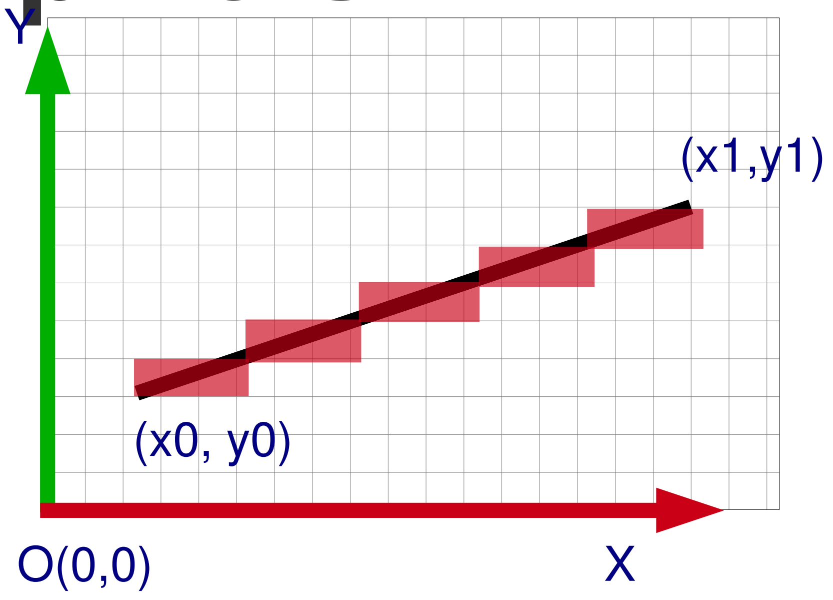
How to colour the correct pixels?



```
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
```

Bresenham's Line Drawing Algorithm

How to colour the correct pixels?

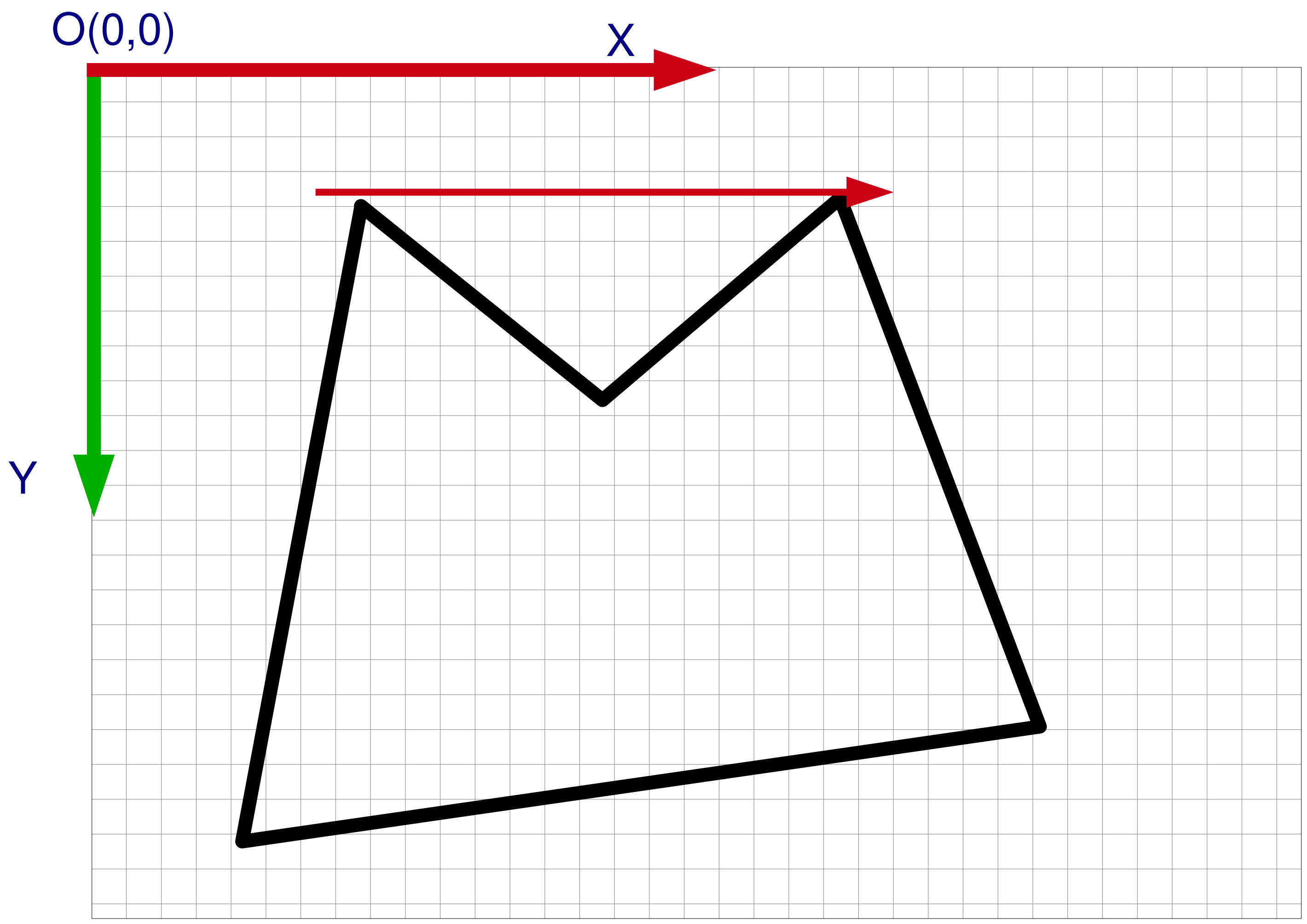


```
function line(int x0, int x1, int y0, int y1)
    int deltax = x1 - x0
    int deltay = y1 - y0
    float error = 0
    float deltaerr = deltay / deltax
      // Assume deltax != 0 (line is not vertical),
     // note that this division needs to be done in a way
        that preserves the fractional 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
```

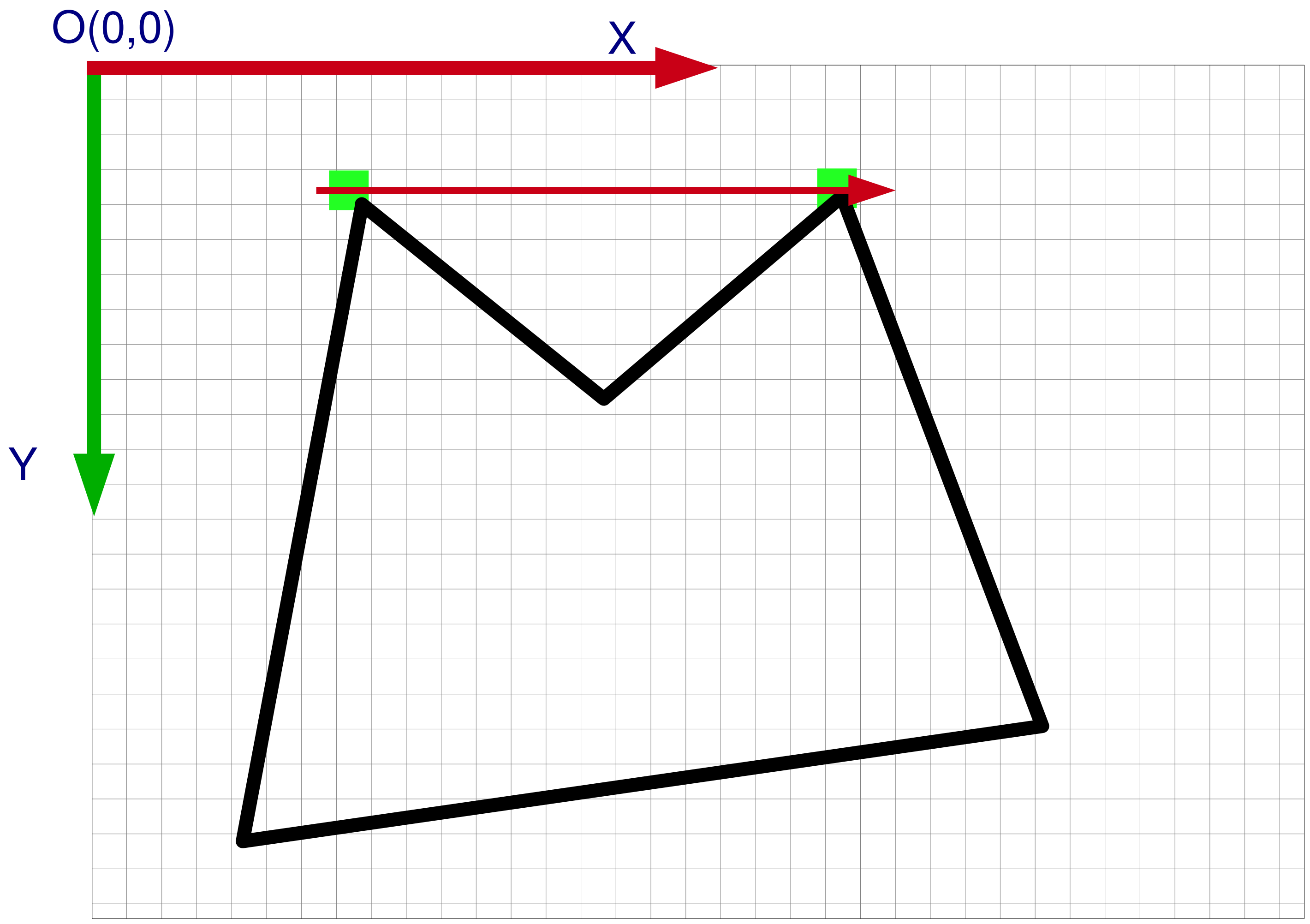
- Extension for all line directions.
- Optimize.
- Demo
- Curves - read!

Bresenham's Line Drawing Algorithm

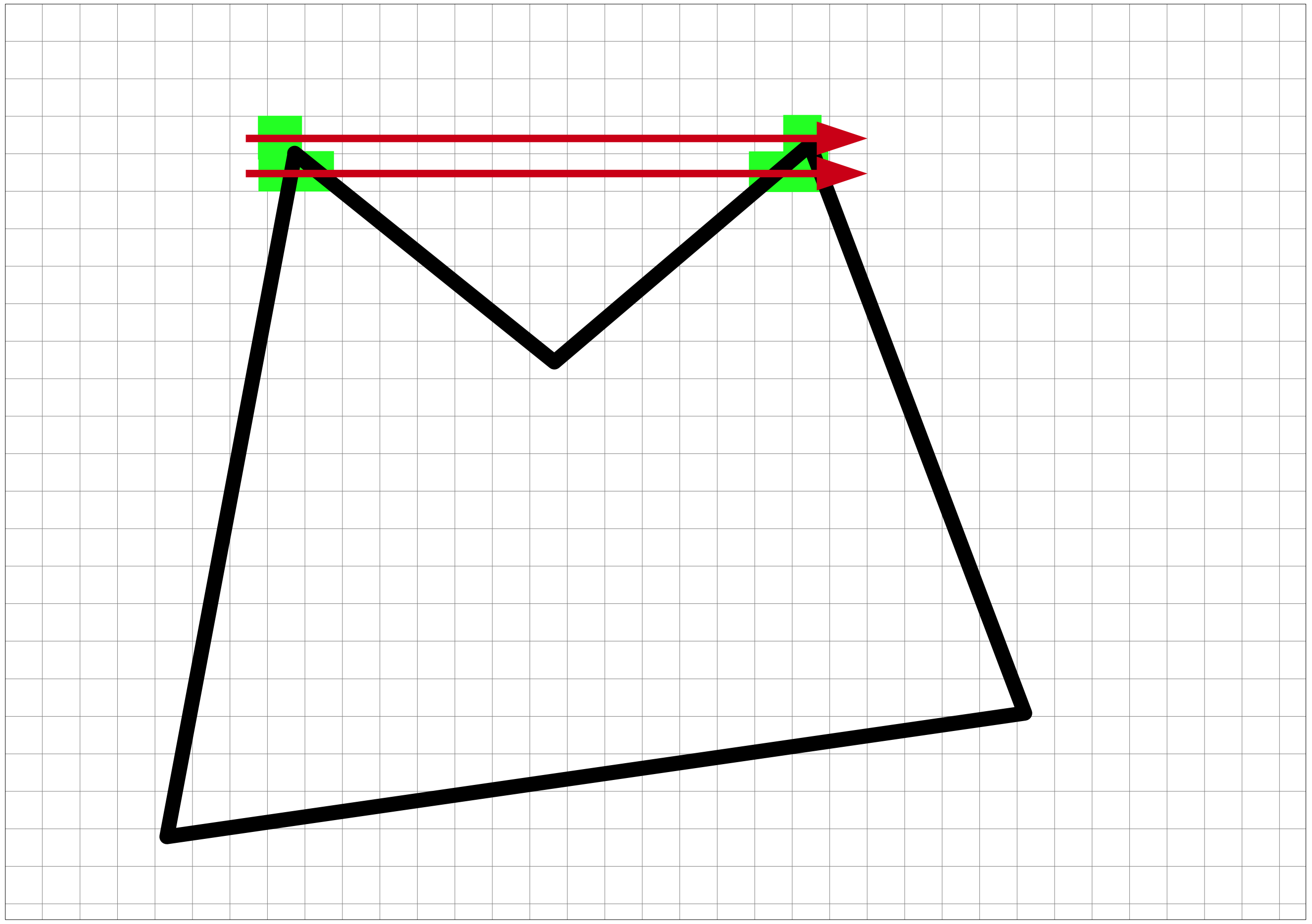
How to fill pixels?



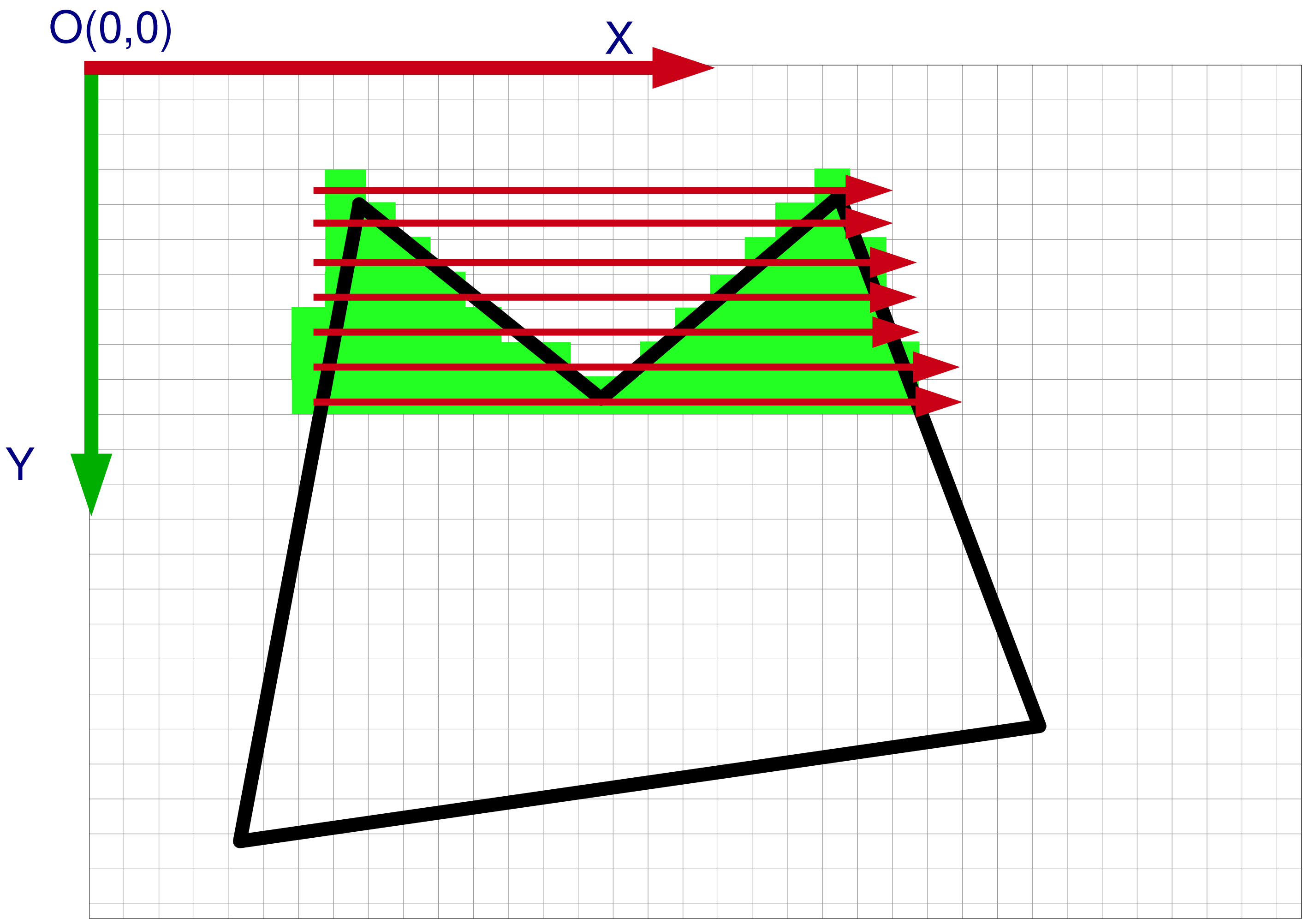
How to fill pixels?



How to fill pixels?

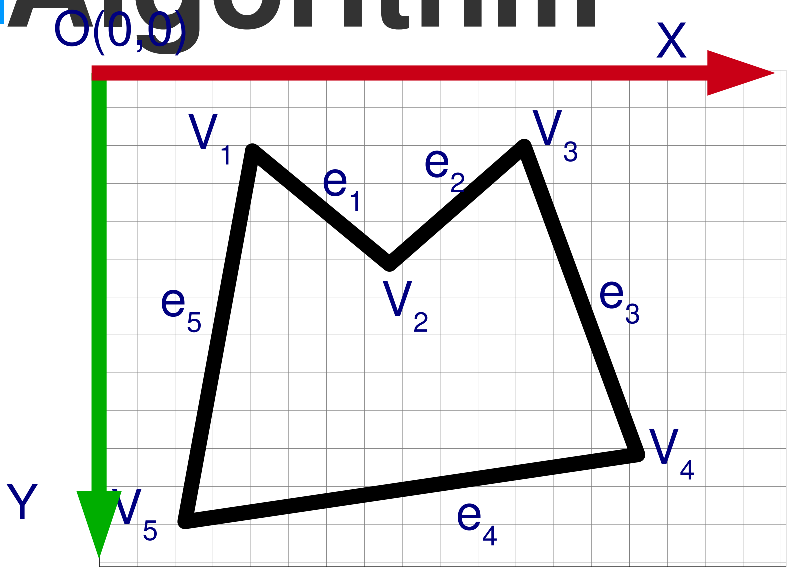


How to fill pixels?



Scanfill Algorithm

How to fill pixels – Scanfill Algorithm

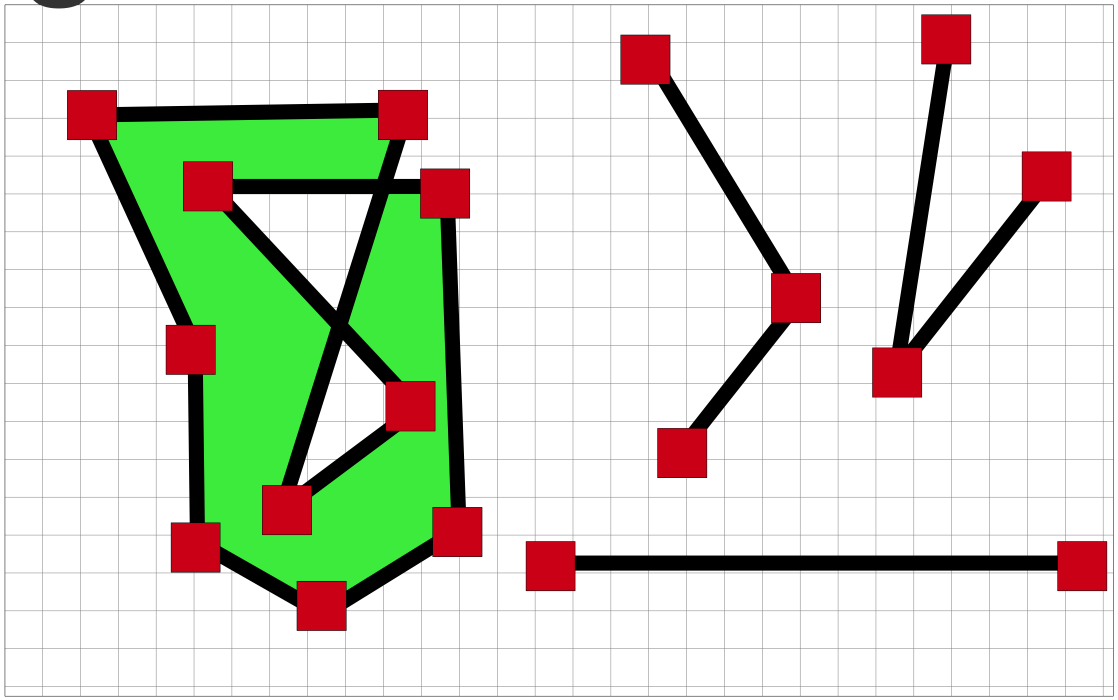


The Edge List

Edge	Y_{min}	Y_{max}	X for $Y=Y_{min}$	1/m
e_2	V_3	V_2	X_3	$1/m_2$
e_3	V_3	V_4	X_3	$1/m_3$
e_1	V_1	V_2	X_1	$1/m_1$
e_5	V_1	V_5	X_1	$1/m_5$
e_4	V_4	V_5	X_4	$1/m_4$

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

How to fill pixels – Scanfill Algorithm



Inside / Outside

Special Cases

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