CS 418: Interactive Computer Graphics

Introduction

Eric Shaffer

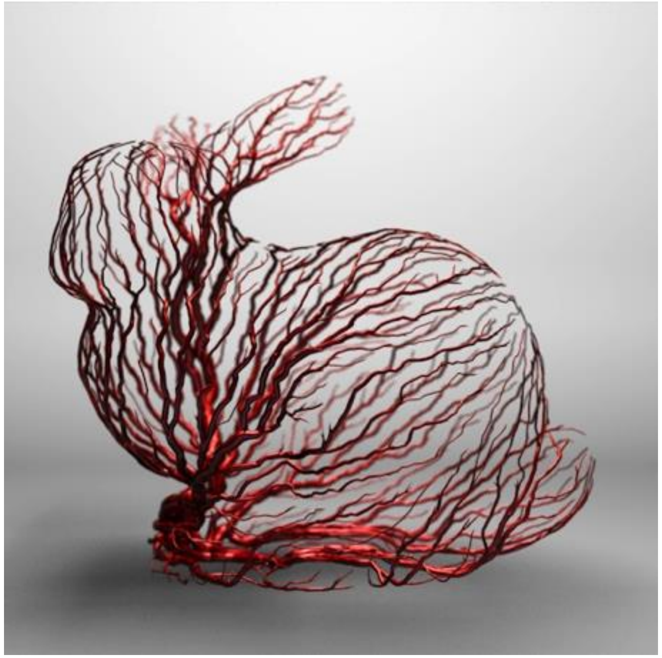
Slides adapted from
Professor John Hart's CS 418 Slides

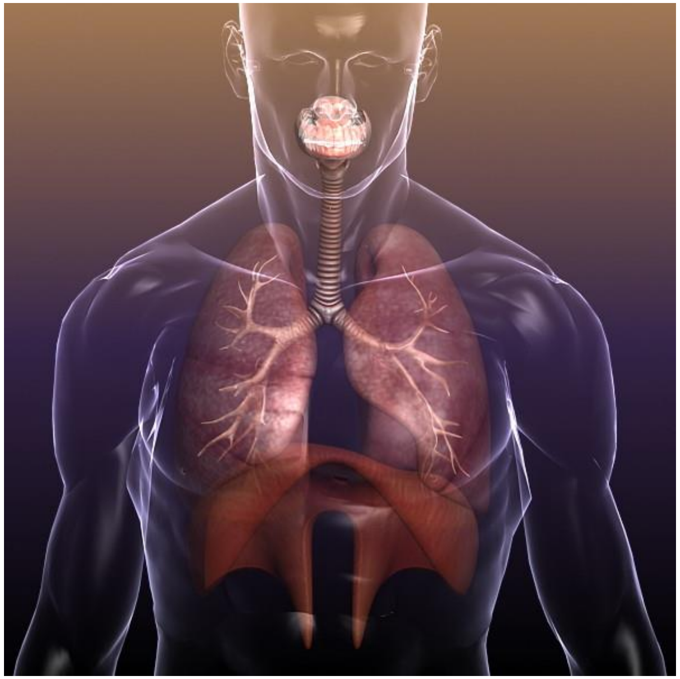
Computer Graphics is Used By…

- Video Game Industry
  - Higher revenue than movies in US ($13B vs $12.9B in 2013)
  - Revenue of $99B globally in 2016

Computer Graphics is Used By…

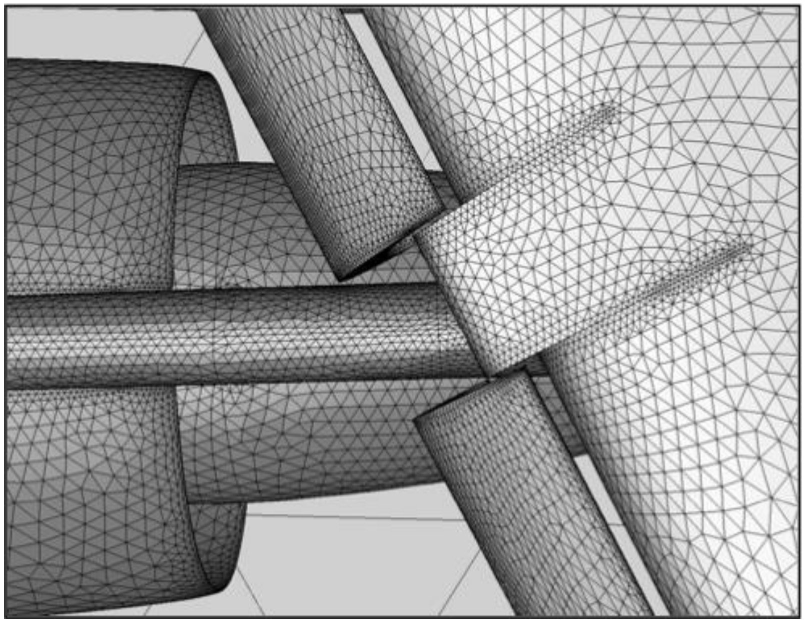
- Medical Imaging and Scientific Visualization
  - Imaging one of the biggest advances in medicine
  - Sci Vis allows people to see previously hidden phenomena

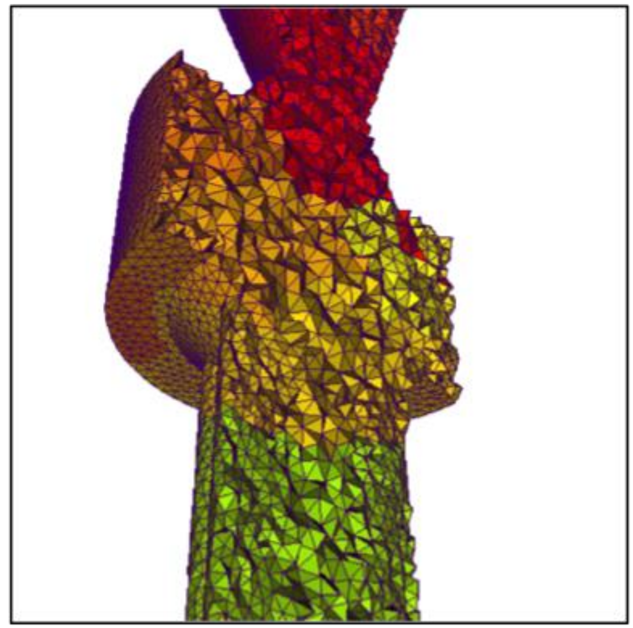




Computer Graphics is Used By…

- Computer Aided Design
  - Engineering, Architecture, the Maker movement





Computer Graphics is Used By…

- Movie Industry
  - Production rendering…non-interactive (CS 419)

CS 418

- **Interactive** Computer Graphics
- Focus on algorithms and techniques used in **rasterization**
  - Rasterization is fast enough for real-time complex 3D rendering
- The course will teach you how to use WebGL
  - Web-based rasterization engine
  - Similar features to many other technologies
    (e.g. OpenGL, Vulkan, D3D)
- We will also cover fundamental graphics algorithms
  - Things like line drawing that reside inside the WebGL library

Things you would not use WebGL for..

- Making a Game
  - Typically would use a game engine like Unity or Unreal

- Making a Movie
  - Renderman

- 3D web app development
  - three.js which is built on WebGL

But to use three.js you need to understand WebGL

And, basic CG concepts need to be understood to use Unity or Renderman as well…

Class Mechanics

- Course Website: https://courses.engr.illinois.edu/cs418/index.html
  - Schedule, lecture materials, assignments
- Piazza: This term we will be using Piazza for class discussions https://piazza.com/illinois/fall2016/cs418/home.
- Grades available on Compass
- **NOTE: No recitation sections this week (8/24)**

Class Mechanics: Grades

Machine Problem 1	10%
Machine Problem 2	15%
Machine Problem 3	15%
Machine Problem 4	15%
Exam 1	15%
Exam 2	15%
Exam 3	15%
No Final Exam	

Grading Scale

- Grades probably on usual scale:
  - 97 to 93: A
  - 93 to 90: A-
  - 90 to 87: B+
  - 87 to 83: B
  - 83 to 80: B-
  - …etc.
- I may adjust the intervals down…but not up

Course Policies

- MPs submitted after the due date lose 10% per day
- Discussing code is fine, copying code is not…
  If we discover plagiarized code, the that code will receive a grade of 0
- In exceptional circumstances where extension may be reasonable (illness, family emergency etc.) arrangement must be made with the instructor (e-mail: shaffer1@illinois.edu)
- Exams are in-class…
- Post technical questions to Piazza

Programming Language

- HTML
- JavaScript
- WebGL
- WebGL version of the GLSL shading language
- Chrome as default browser
- Chrome DevTools to debug code
- **If you have a laptop, bring it to recitation section**
- Some WebGL examples: https://www.chromeexperiments.com/webgl

A word about OpenGL

- Open standard for graphics programming
  - Developed by Silicon Graphics in 1992
  - Available on most platforms…
  - Bindings available for lots of languages…
  - It’s low level
- “Windowing” typically requires another library
  - e.g. GLUT
- Version 3.0 (2008) introduced programmable shaders
  - Deprecated fixed-function pipeline and direct-mode rendering
- **Vulkan** API announced as the successor technology (2015)

WebGL is not exactly OpenGL

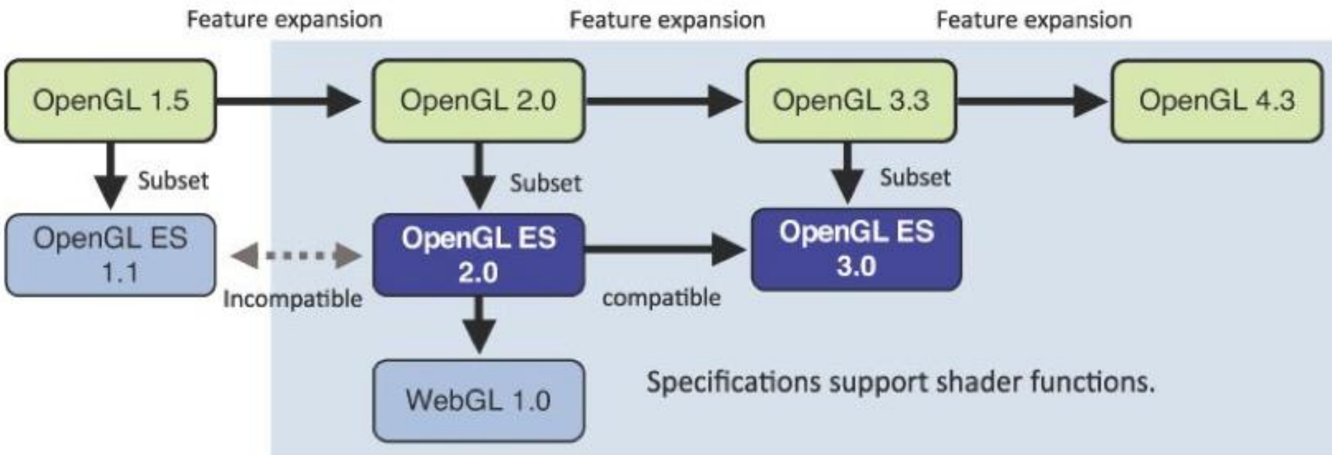


Figure from *WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL* by Matsuda and Lea

WebGL Application Structure

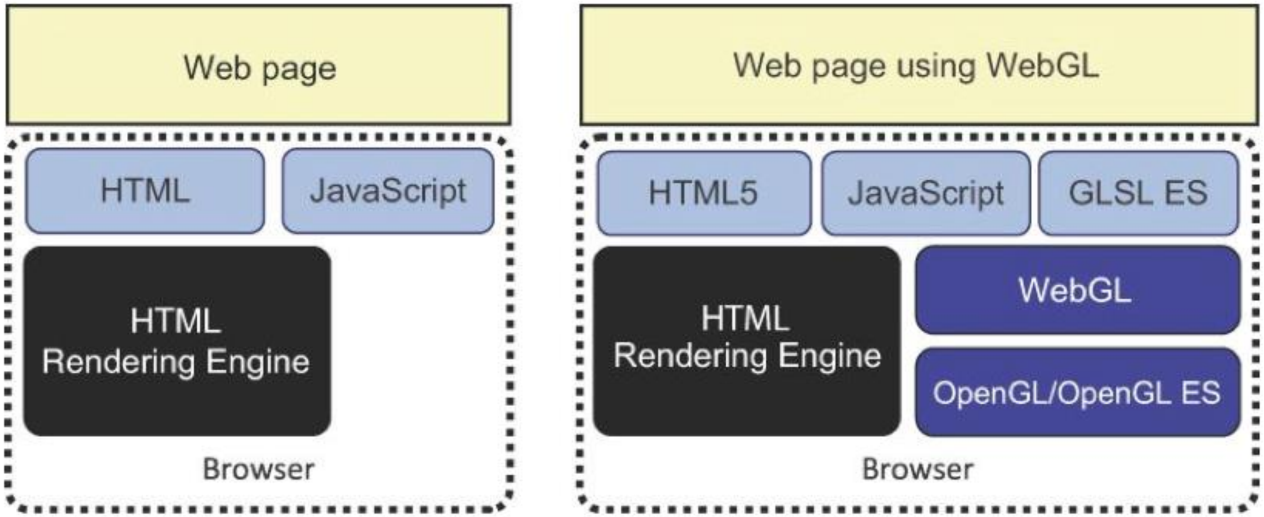


Figure from *WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL* by Matsuda and Lea

Your application will generally just have HTML and JavaScript files

WebGL

- WebGL relatively new (2011) 3D graphics support for web
- WebGL advantages
  - runs in browser
  - naturally cross-platform
  - don't need to obtain/build other libraries
  - gives you "windowing" for free
  - easy to publish/share your stuff
- Disadvantages
  - Depends on how you feel about JavaScript
  - Performance can be tricky

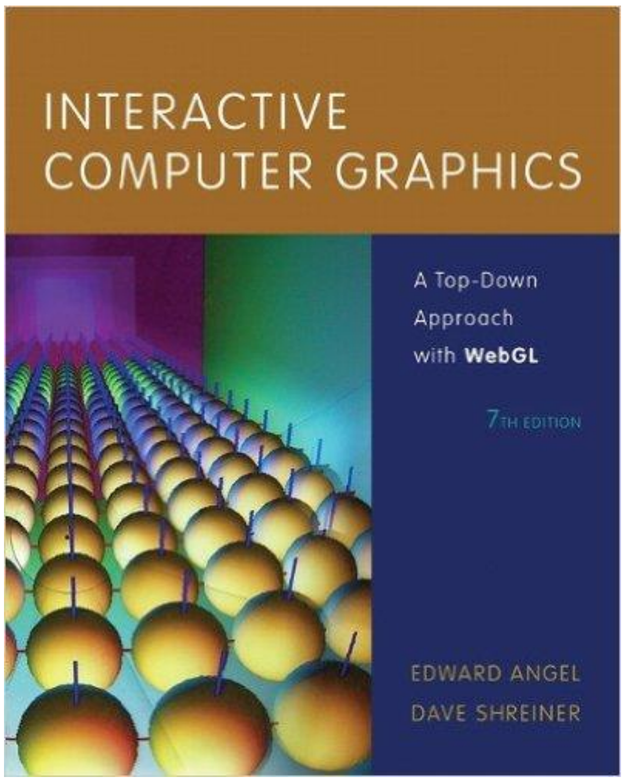
Class Mechanics: No Book

- We'll post notes online
  - It will save you $150

Language References and Resources

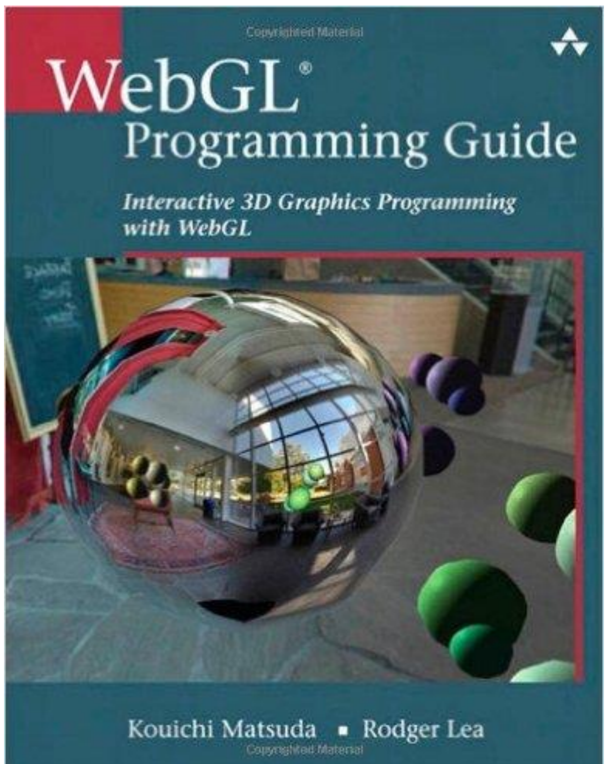
- JavaScript/HTML/CSS: http://www.w3schools.com/
- WebGL Specification: https://www.khronos.org/webgl/
- WebGL Tutorial: http://webglfundamentals.org/
- Suggested Editors: Brackets, LightTable
- Chrome DevTools Overview: https://developer.chrome.com/devtools

Suggested Books



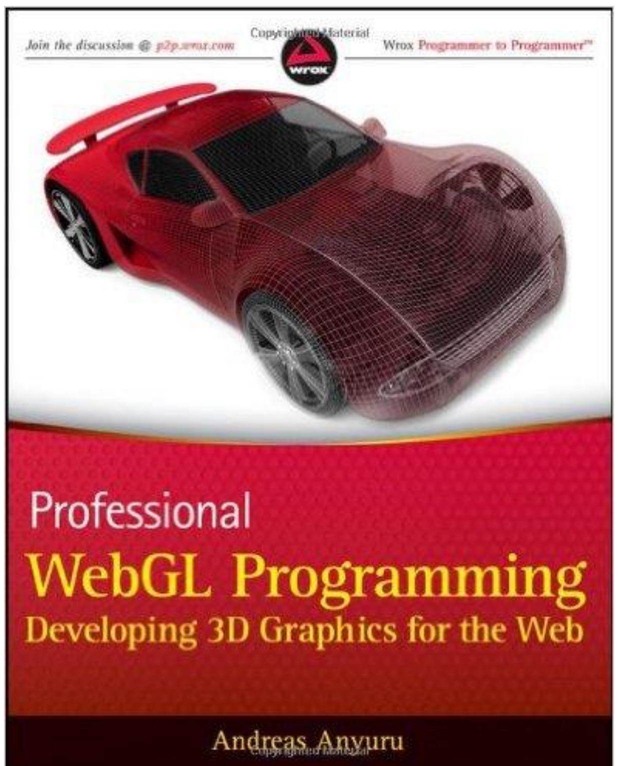
Interactive Computer Graphics: A Top-Down Approach with WebGL (7th Edition)
Mar 10, 2014
by Edward Angel and Dave Shreiner

Suggested Books



WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL (OpenGL)Jul 19, 2013
by Kouichi Matsuda and Rodger Lea

Suggested Books



Professional WebGL Programming: Developing 3D Graphics for the Web
May 8, 2012
by Andreas Anyuru

Course Topics

- Real-time generation of 3D computer graphics through **rasterization**
- Low-level basic algorithms
  - Line-drawing
  - Hidden surface removal
  - Lighting and shading
  - Texturing
  - Scan conversion
- Using these capabilities in WebGL
- Modeling and viewing transformations
- Geometric modeling
- Animation

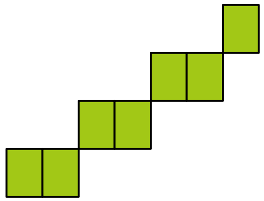
2D Graphics: Vector Graphics and Raster Graphics

Vector Graphics

- Plotters, laser displays
- "Clip art," illustrations
- PostScript, PDF, SVG
- Low memory (display list)
- Easy to draw line

Raster Graphics

- TV's, monitors, phones
- Photographs
- GIF, JPG, etc.
- High memory (frame buffer)
- Hard to draw line



Definitions: Pixel and Raster

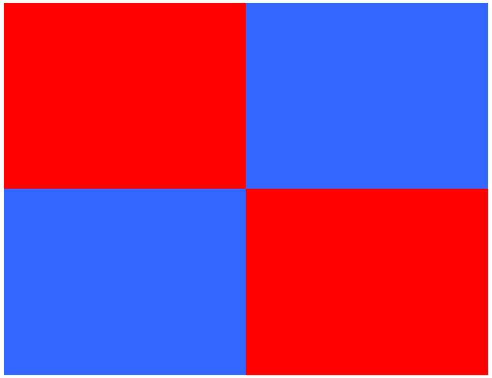
A ***pixel*** is the smallest controllable picture element in an image

A ***raster*** is a grid of pixel values

Typically rectangular grid of color values

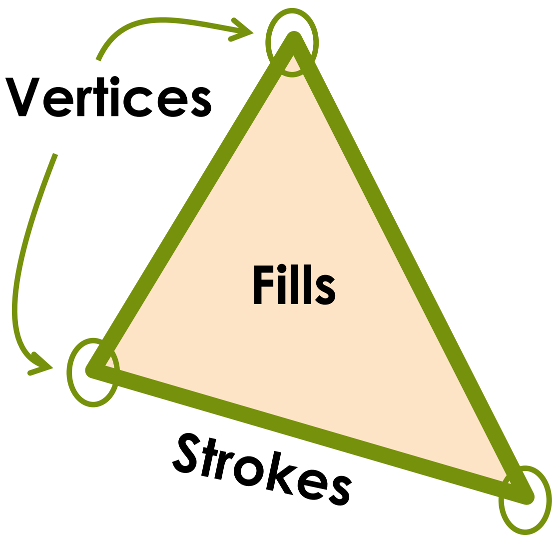
(255,0,0), (0,0,255)

(0,0,255), (255,0,0)

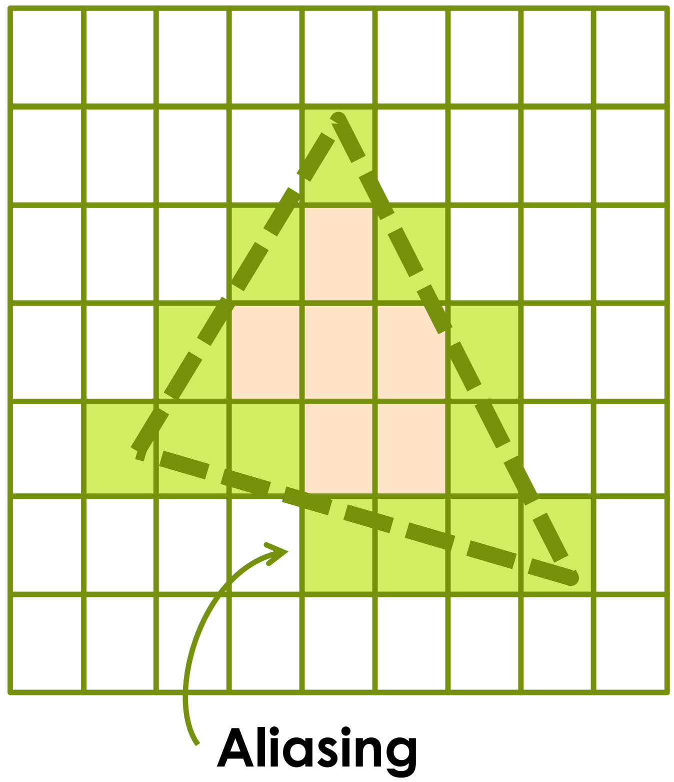


Rasterization

Primitives

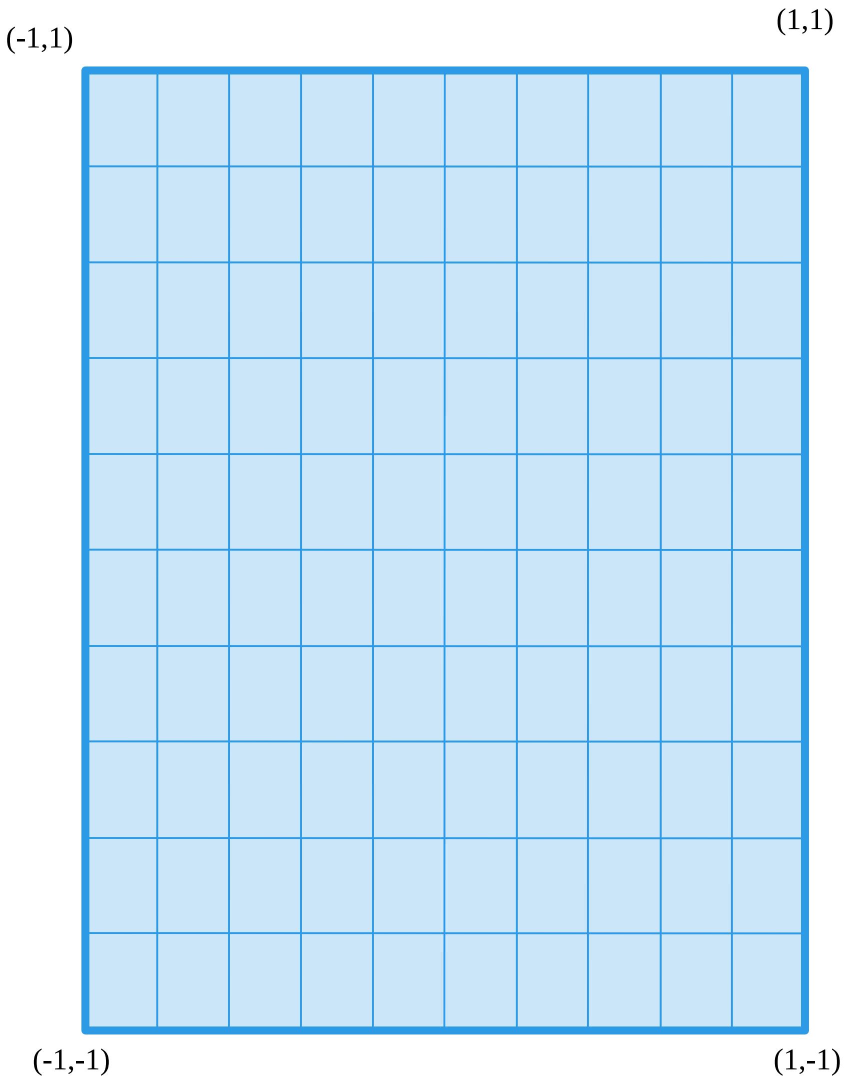


Pixels



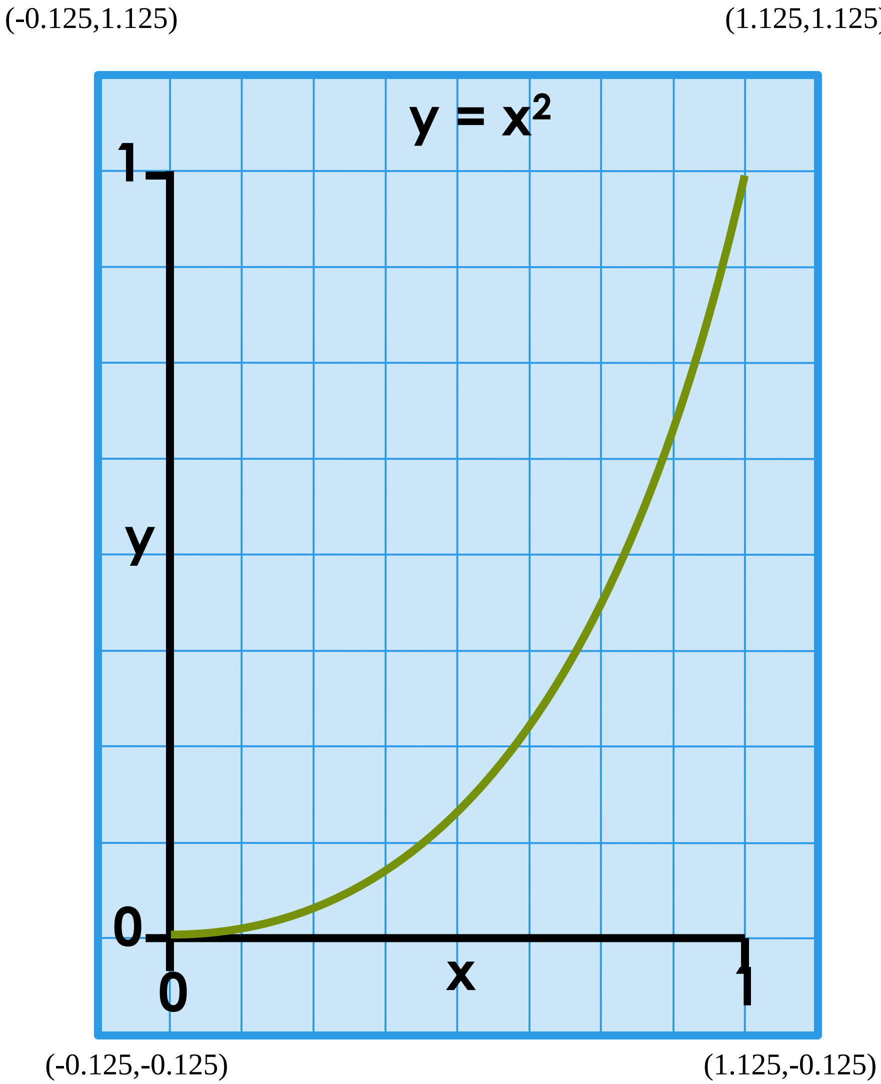
Canvas Coordinates

- Mathematical plotting coordinates
- Used to define positions of vertices for graphics primitives (e.g. triangles)
- **Note: Different technologies may use terminology other than "canvas coordinates" to refer to the same idea**



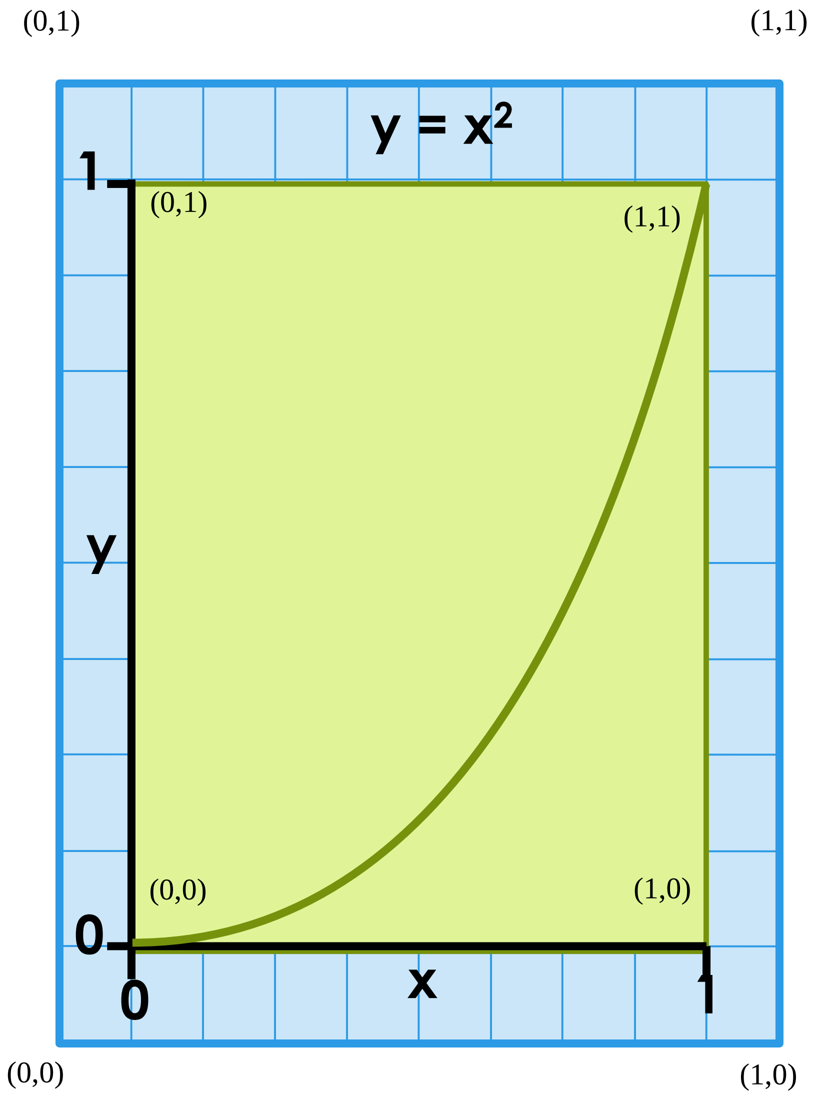
Canvas Coordinates

- Can redefine corners of canvas coordinates to whatever is convenient
- Can use graph's coordinates for domain and range, but leave room for axes and notation



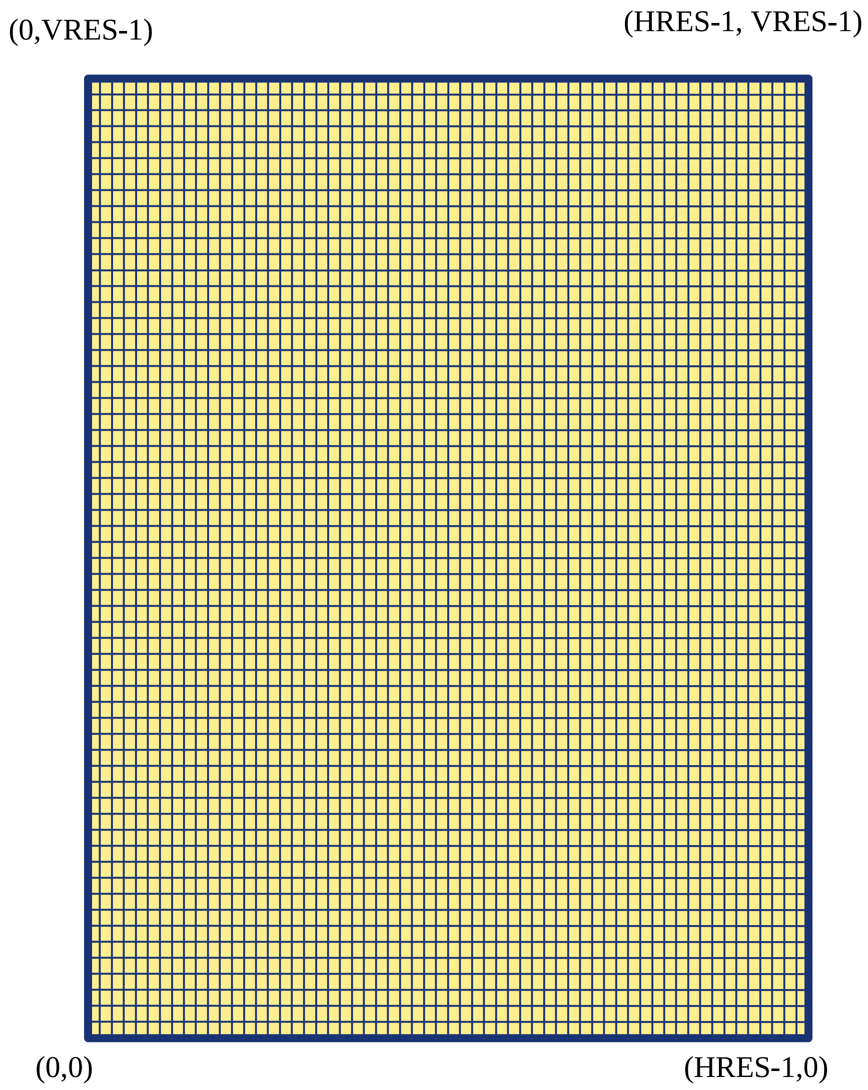
Hierarchical Coordinate Systems

- Create a canvas for entire visualization
  - Extends across area of screen
  - Plots coords from (0,0) to (1,1)
- Create a sub-canvas for plotting data
  - Extends from (1/8,1/8) to (7/8,7/8) of parent canvas
  - Plots coords from (0,0) to (1,1)



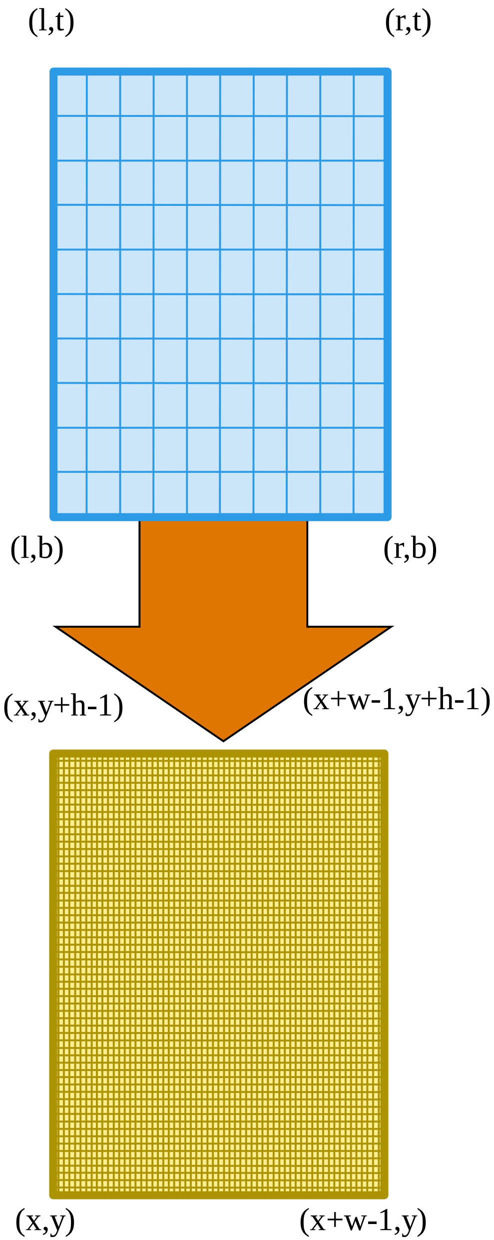
Screen Coordinates

- Physical per-pixel integer coordinates
- Sometimes (0,0) is in the upper left corner (e.g. for mouse input)



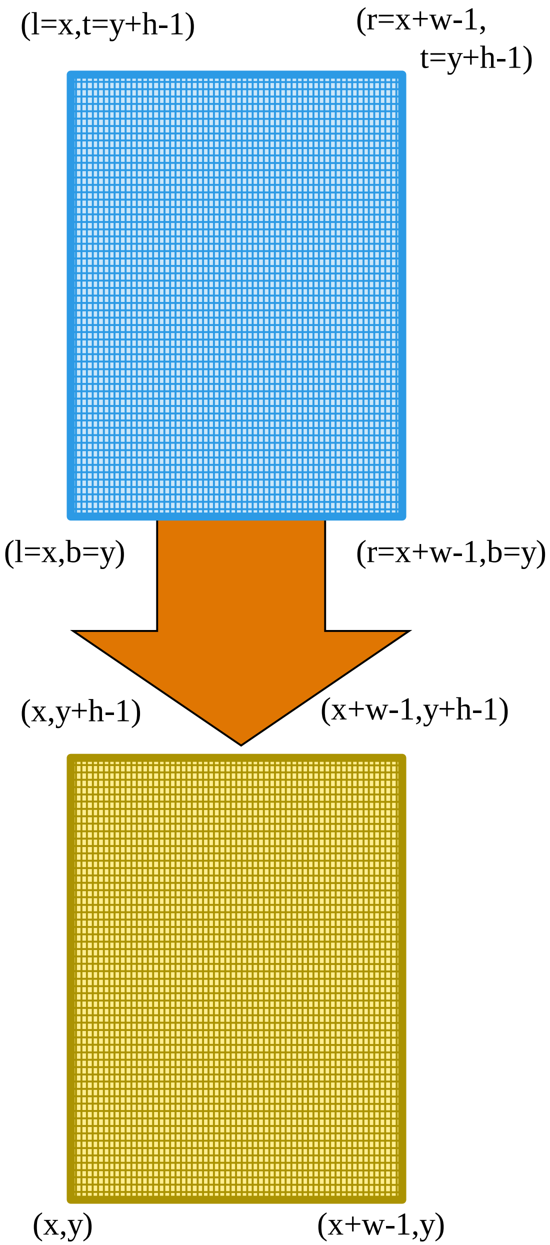
Canvas→Screen Transformation

- Draw primitives in canvas coordinates
  - Extending horizontally from l to r
  - Extending vertically from b to t
- Primitives are transformed to screen's pixel coordinates
- Rasterization fills in transformed outline with pixel
  - Positions
  - Colors



Working in Screen Coordinates

- Can use the same coordinates for both canvas and screen coordinates
- Specify primitives using pixel locations
- Can result in non-scalable resolution dependent output

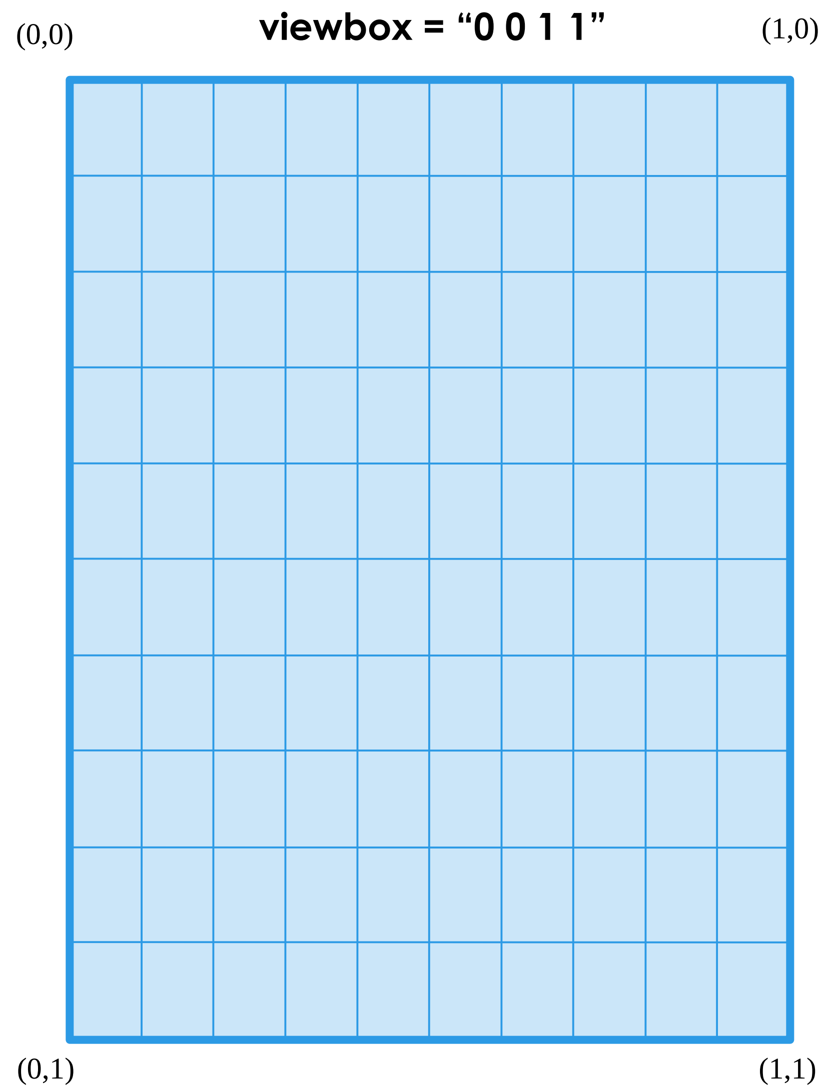


Scalable Vector Graphics

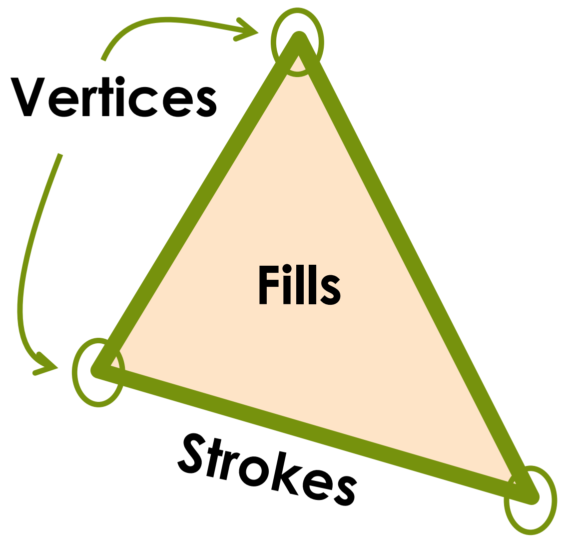
- Format specification for describing 2-D graphics
- Embedded in HTML with <svg> tag

```
<svg width=pw height=ph
     viewbox="x y w h"> … </svg>
```

- Creates a display region of pw x ph pixels in screen coordinates
- Creates a drawing canvas w x h units in canvas coordinates
- Origin always upper-left corner



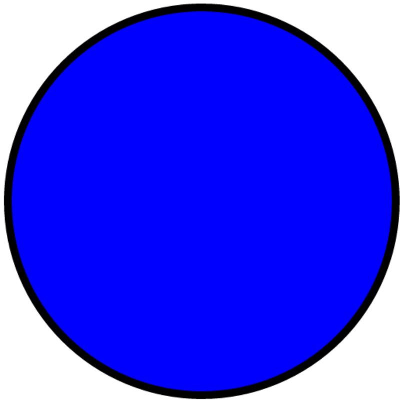
SVG Drawing



Circle

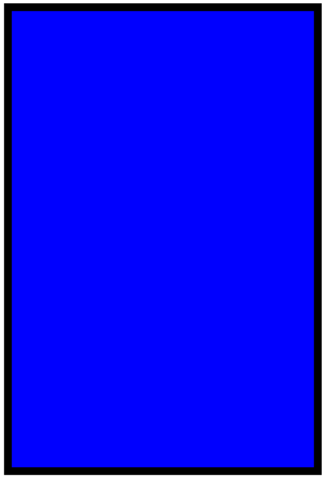
```
<!DOCTYPE html>
<html>
<body>
<svg height= 500 width=500 viewbox = "0 0 1 1">

  <circle cx = "0.5"
          cy = "0.5"
           r = "0.25"
      stroke-width = "0.01"
      stroke = "black"
      fill= "blue"/>
</svg>
</body>
</html>
```



Rectangle

```
<!DOCTYPE html>
<html>
<body>
<svg height= 500 width=500 viewbox = "0 0 1 1">
<rect x = "0.3"
        y = "0.2"
    width = "0.4"
   height = "0.6"
      stroke-width = "0.01"
      stroke = "black"
      fill= "blue"/>
</svg>
</body>
</html>
```



Filled Closed Path

```
<svg height=500 width=500
     viewbox="0 0 1 1">

  <path d = "M 0.2 0.1
             L 0.2 0.3
             L 0.4 0.3
             L 0.4 0.7
             L 0.2 0.7
             L 0.2 0.9
             L 0.8 0.9
             L 0.8 0.7
             L 0.6 0.7
             L 0.6 0.3
             L 0.8 0.3
             L 0.8 0.1
             Z"
     fill = "orange"
  />
</svg>
```

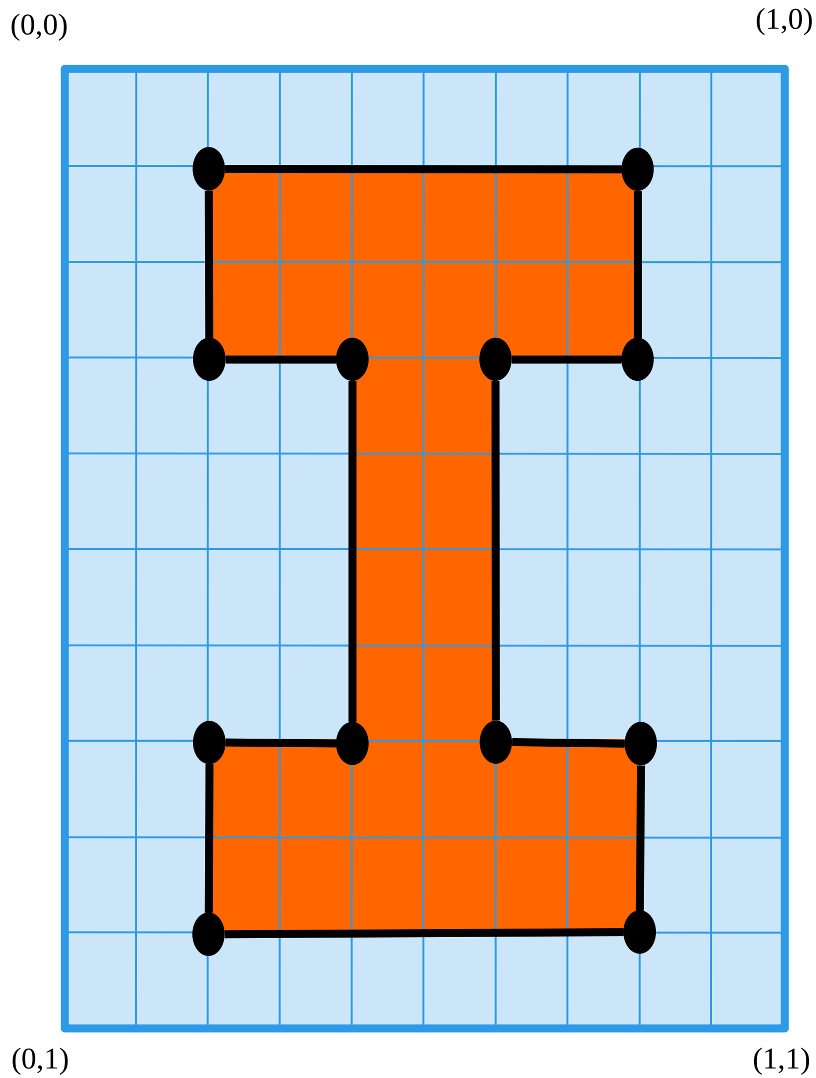
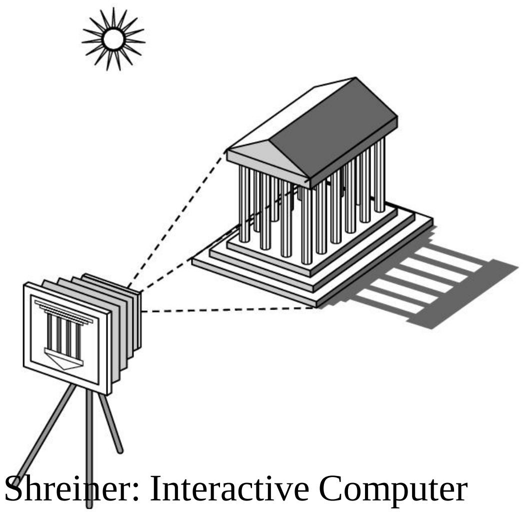


Image Formation

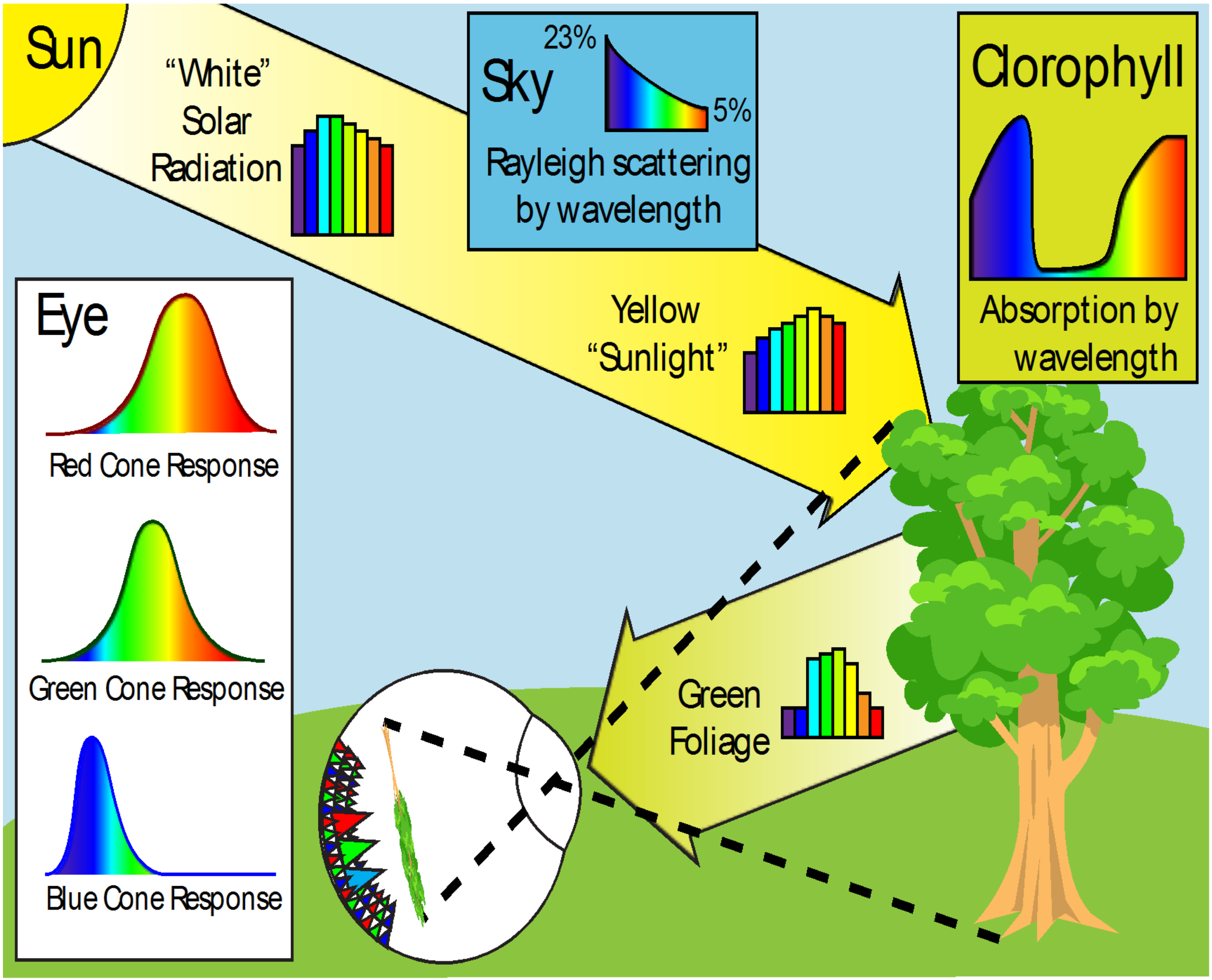
- Typically, goal in CG is to generate a 2D image of a 3D scene…
  - The input data is a scene description
  - Output is an image



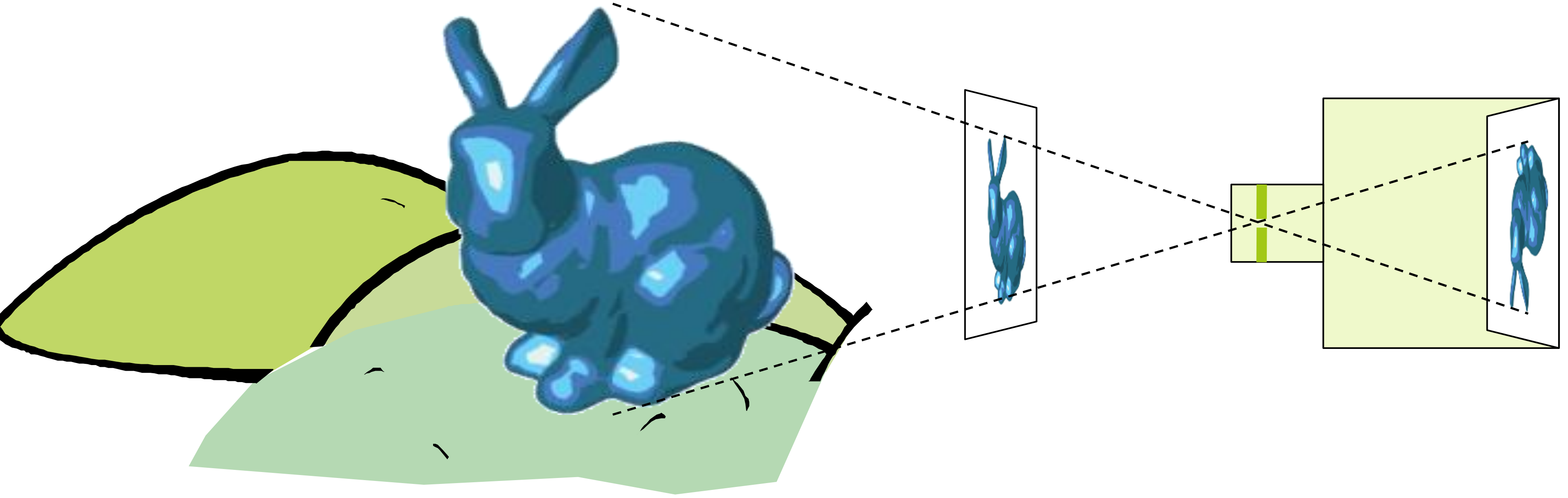
Angel and Shreiner: Interactive Computer Graphics 7E © Addison-Wesley 2015

- One approach is to computationally mimic a camera or human eye

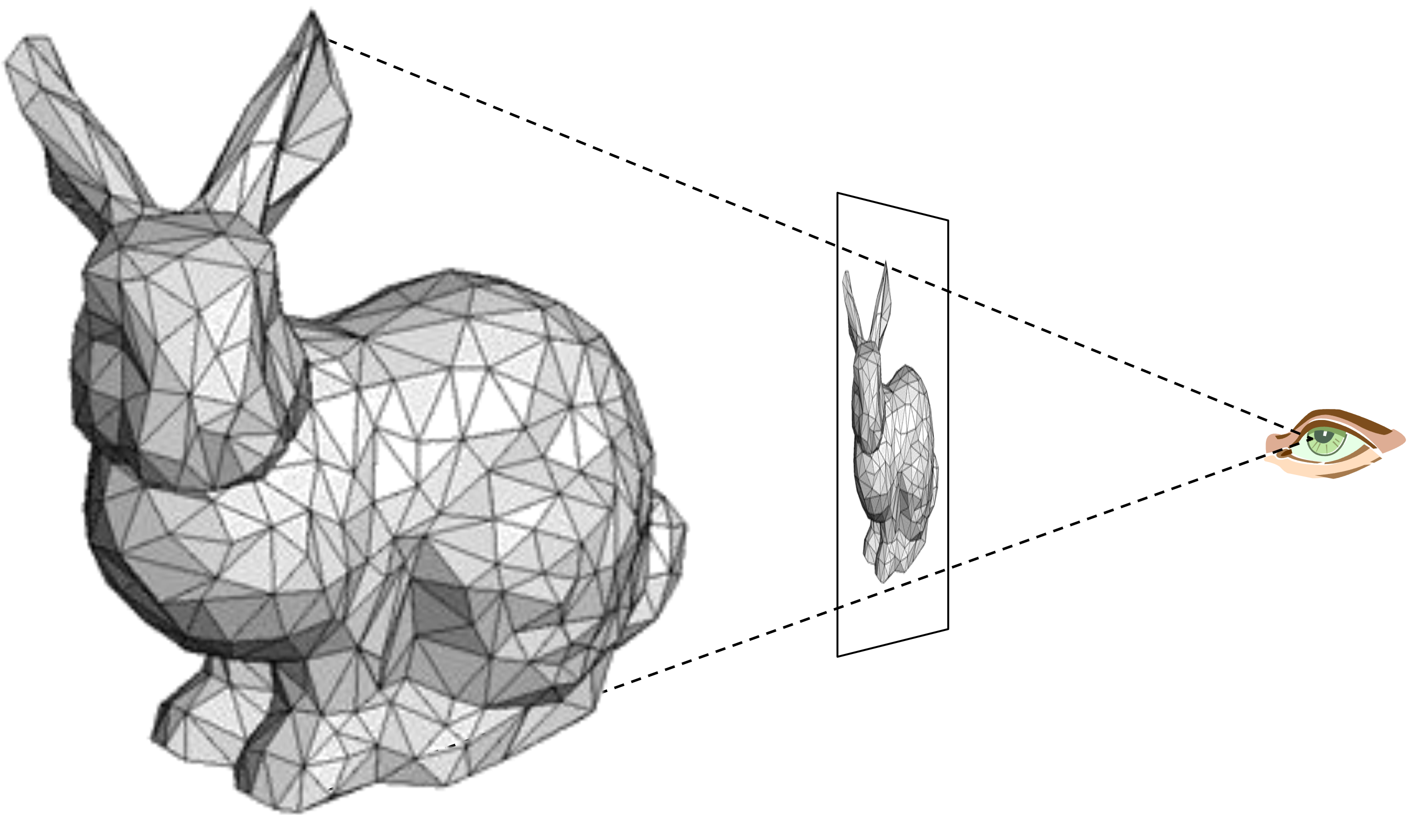
- In the scene…there are objects…lights…and a viewer


Sun
"White" Solar Radiation
Sky
23%
5%
Rayleigh scattering by wavelength
Clorophyll
Absorption by wavelength
Yellow "Sunlight"
Eye
Red Cone Response
Green Cone Response
Blue Cone Response
Green Foliage

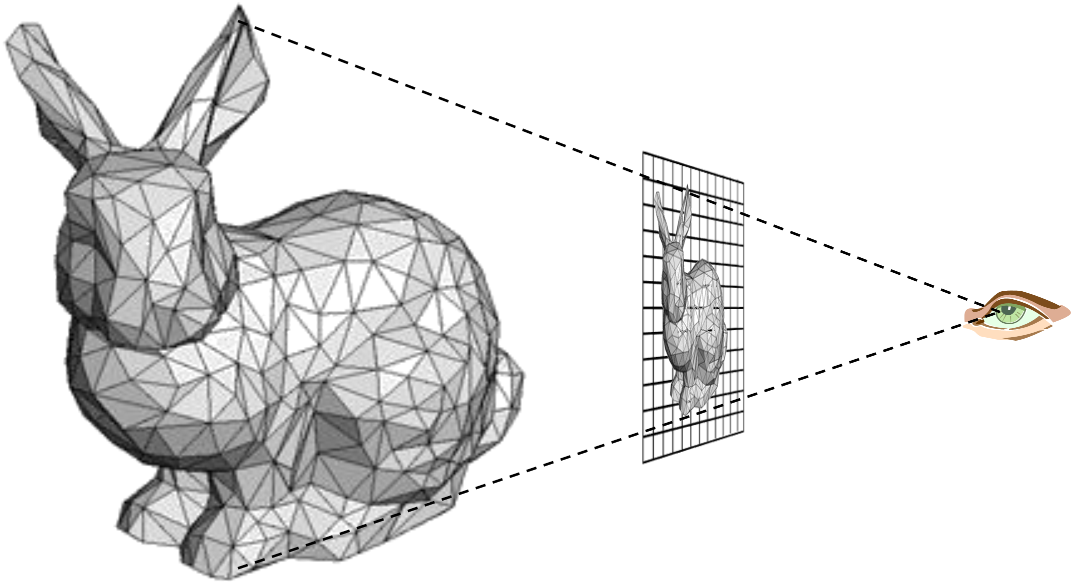
Synthetic Camera Model

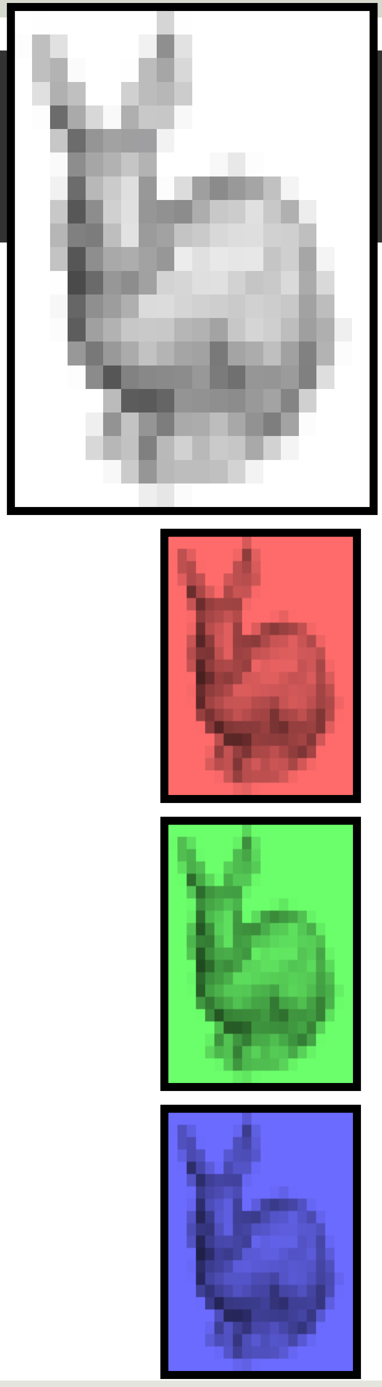


Polygonal Models



Pixel Discretization





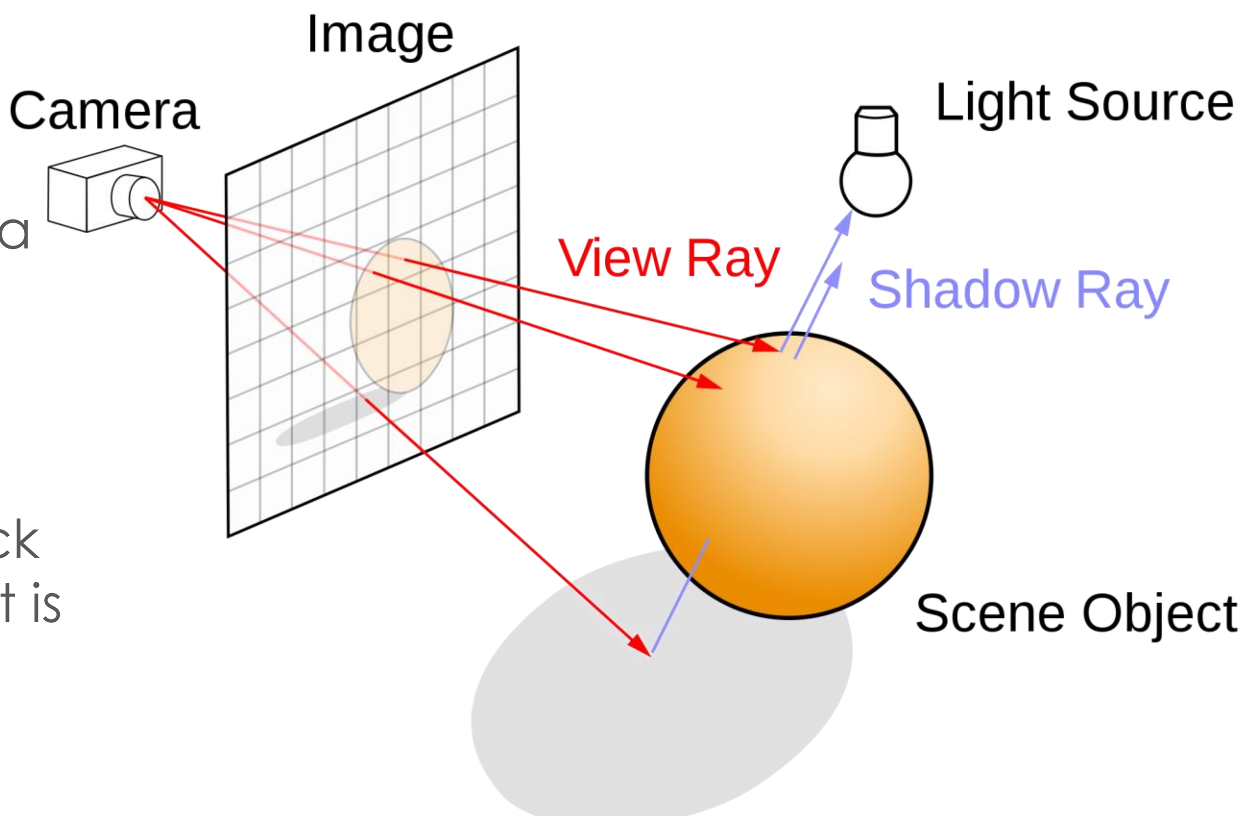
Ray Tracing

Follow ray of light….

Can trace from an eyepoint through a pixel

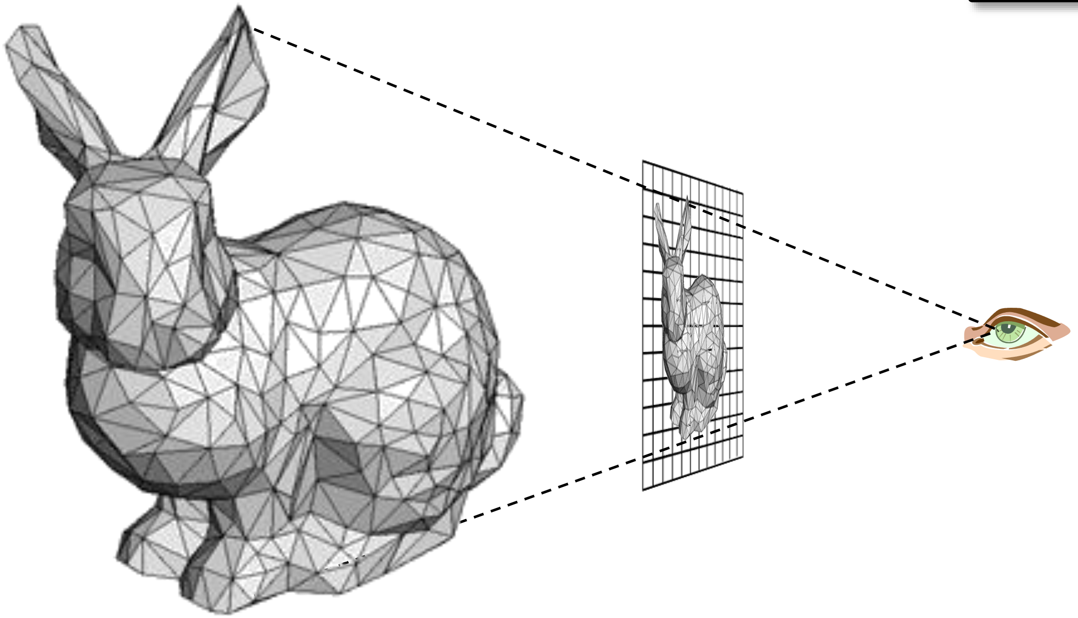
See what object the ray hits…

How would you check to see if the object is lit?



Rasterization

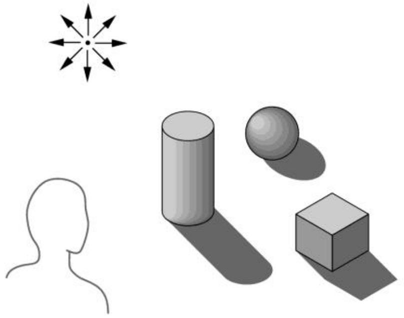
For each primitive:
  Compute illumination
  Project to image plane
  Fill in pixels



Global vs Local Lighting

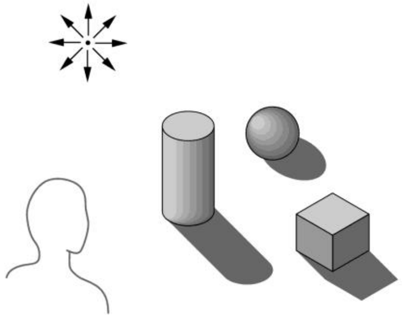
- For true photo-realism:
  Cannot compute color or shade of each object independently

Why?



Global vs Local Lighting

- For true photo-realism:
  Cannot compute color or shade of each object independently
  - Some objects are blocked from light
  - Light can reflect from object to object
  - Some objects might be translucent
- Can rasterization produce global lighting effects?
- Can ray tracing?
- The big advantage of rasterization is…?



What Should You Know?

- Class mechanics
- Difference between Vector Graphics and Raster Graphics
- Definition of Canvas Coordinates
- Definition of Screen Coordinates
- Difference between Ray-Tracing and Rasterization

For Next Class

- If you have a laptop or your own PC
  - Install an editor (e.g. Brackets)
  - Install a browser supporting WebGL (e.g. Chrome)
  - Verify WebGL runs in that browser on your machine
  - https://courses.engr.illinois.edu/cs418/HelloColor.html
- If you don’t have your own computer, try it an EWS lab