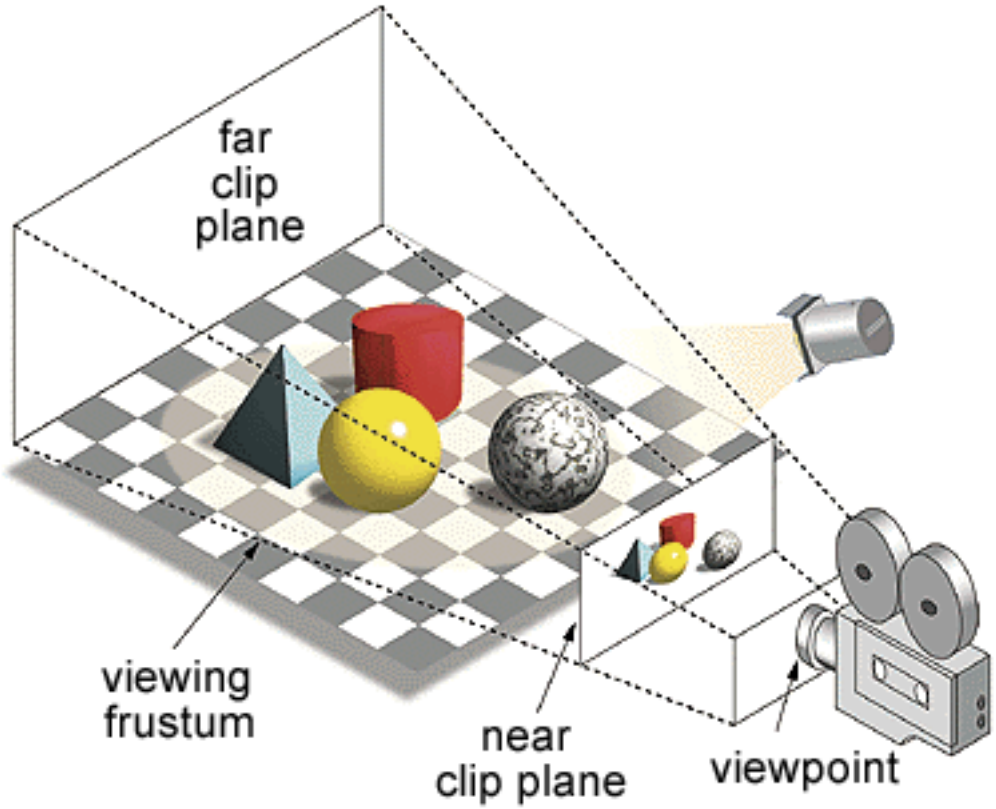
CS 464 Review

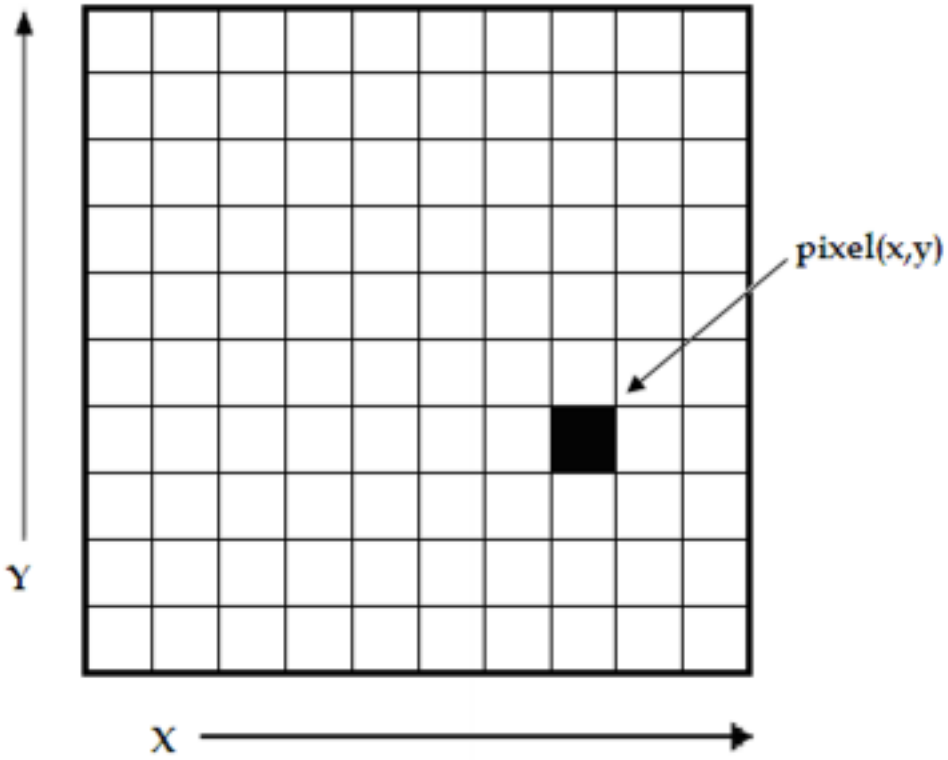
Review of Computer Graphics for Final Exam

Goal: Draw 3D Scenes on Display Device

3D Scene – Abstract Model



Framebuffer – Matrix of Screen Pixels



In Computer Graphics: If it looks right then it is right

Components of 3D Scene Model

Components

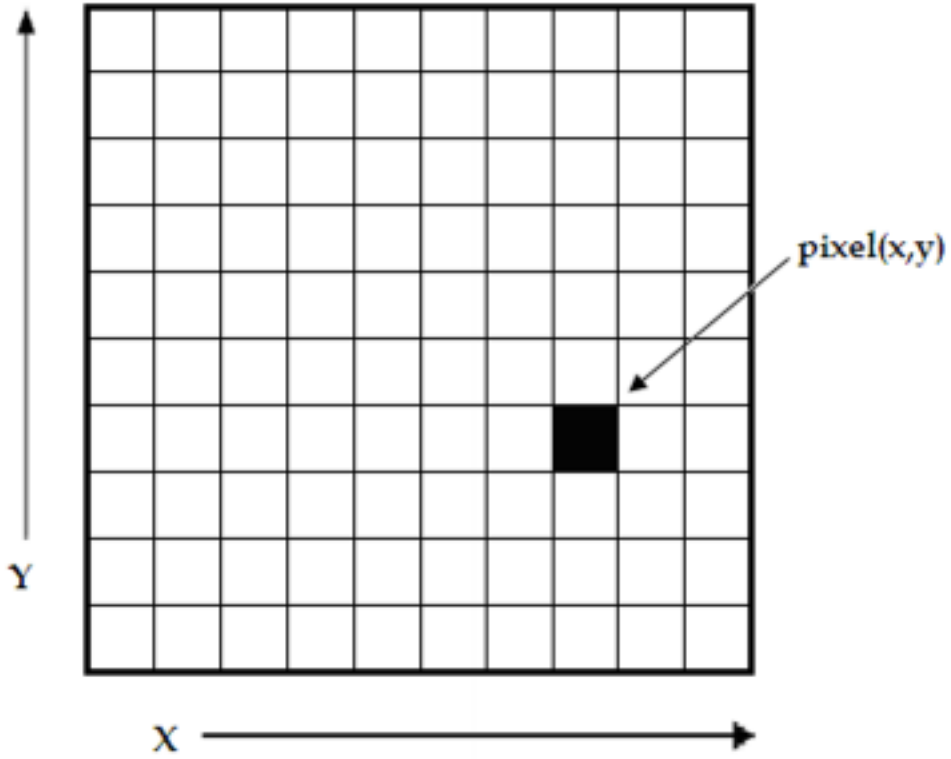
- Coordinate System
- Geometry = Models of Objects
- Lights
- Camera
- Textures
- Materials
- *transforms*

Common Attributes

- Coordinates = x,y,z – Cartesian
- Colors = r, g,b = Red, Green, Blue

Framebuffer – device for showing pictures.

Framebuffer – Matrix of Screen Pixels



Core Features of Framebuffer

- Matrix of Pixels
- Pixels: 8 bits of Red,Green,Blue, Alpha for 32 bits per pixel.
- Fixed width
- Fixed height
- Coordinates go left right, top to bottom.
- clearBuffer(acolor);
- setPixel(x,y, acolor);

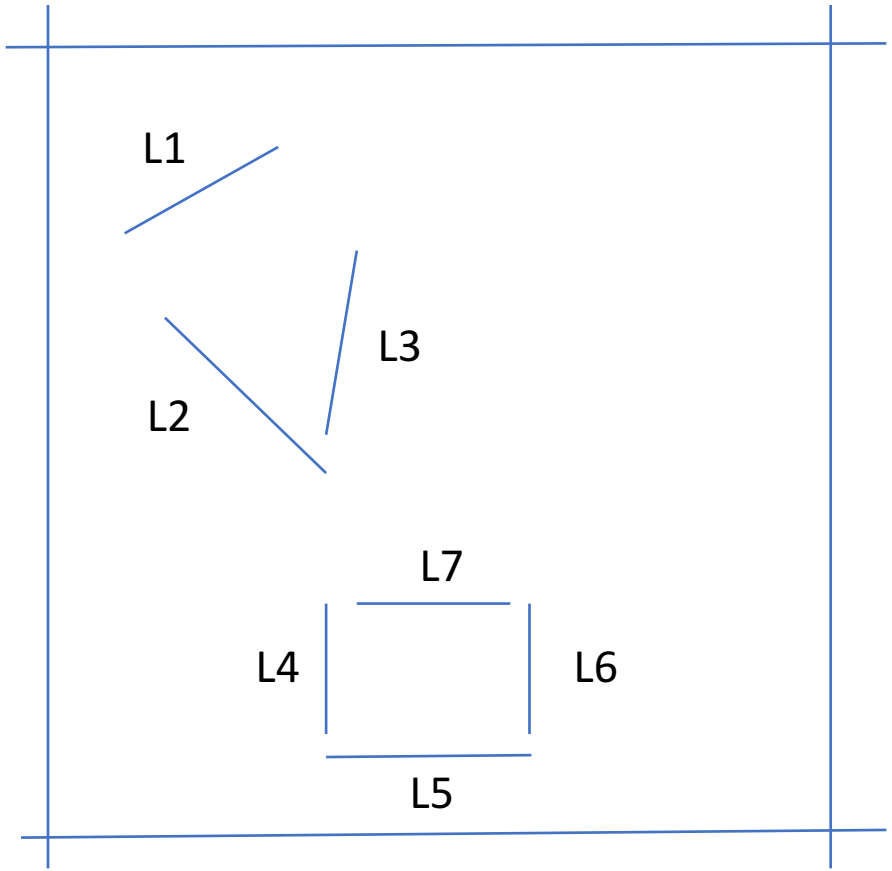
What are our challenges?

- How do we draw simple 2D primitives?
  - Lines (x,y)
  - Triangles (x,y)
- How do we draw simple 3D primitives?
  - Lines (x,y,z)
  - Triangles (x,y,z)

What are our challenges?

- How do we draw simple 2D primitives?
  - Lines (x,y) – rasterize between two points.
  - Triangles (x,y) – clip the triangle, then rasterize.
- How do we draw simple 3D primitives?
  - Lines (x,y,z) – Project from 3D to 2D then use 2D methods.
  - Triangles (x,y,z) – Project from 3D to 2D then use 2D methods.
  - Projection: - Perspective Projection and Orthographic Projection

Simple 2D Scene: Lines



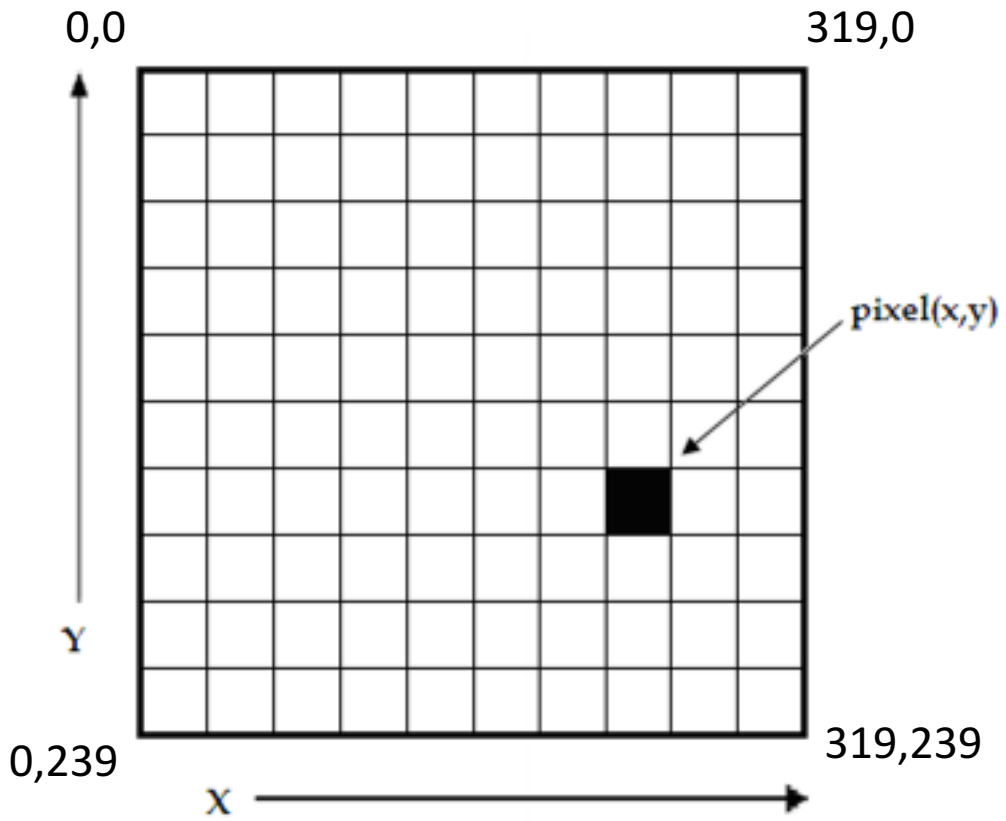
Scene Definition
A simple list of vertices

```
L={
        xb1,yb1,xe1,ye1,
        xb2,yb2,xe2,ye2,
        ....
        xb6,yb6,xe6,ye6
}
```

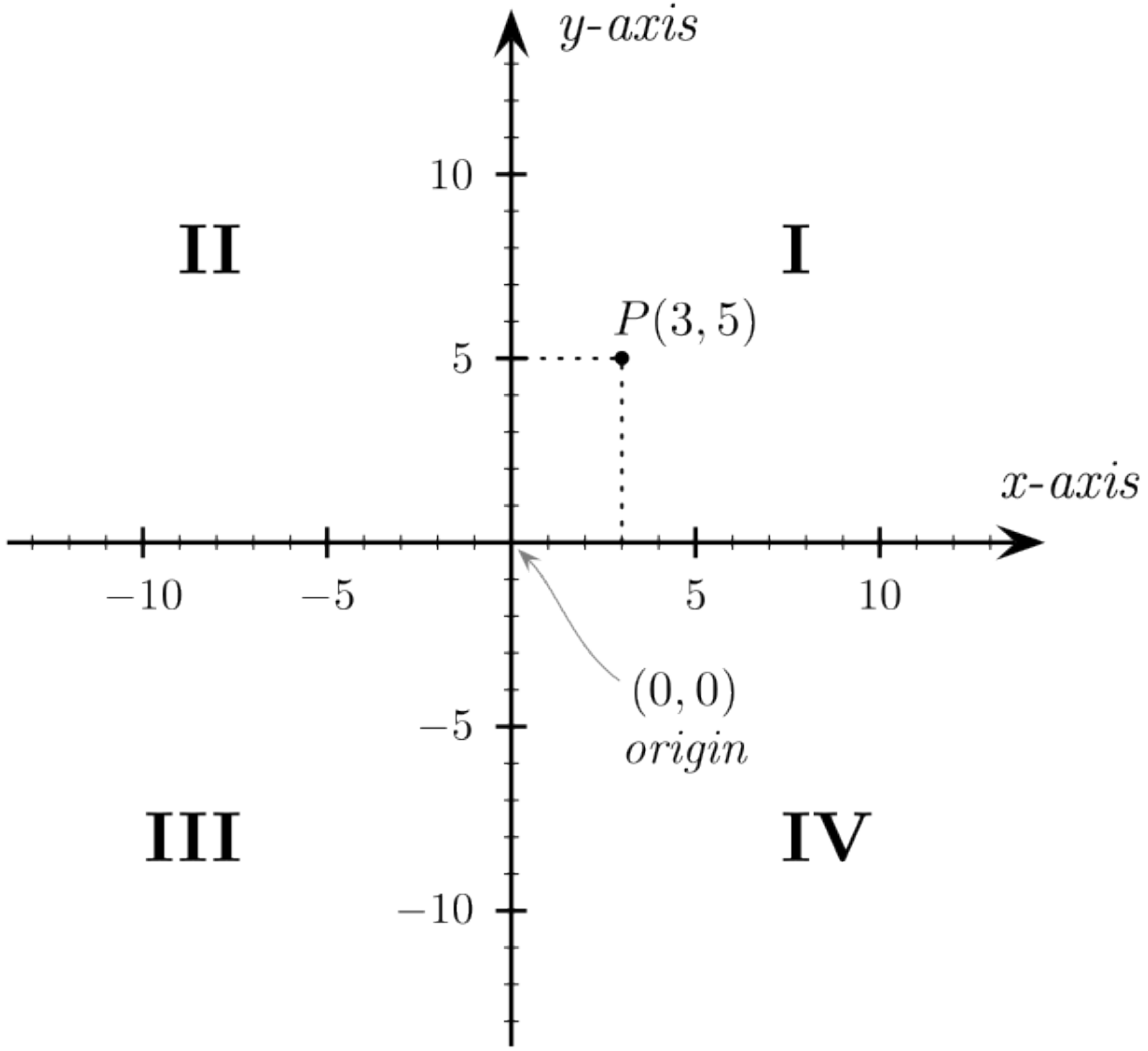
Coordinate Space= Framebuffer

Coordinate Spaces

Framebuffer – Matrix of Screen Pixels



Normalized Device Coordinates



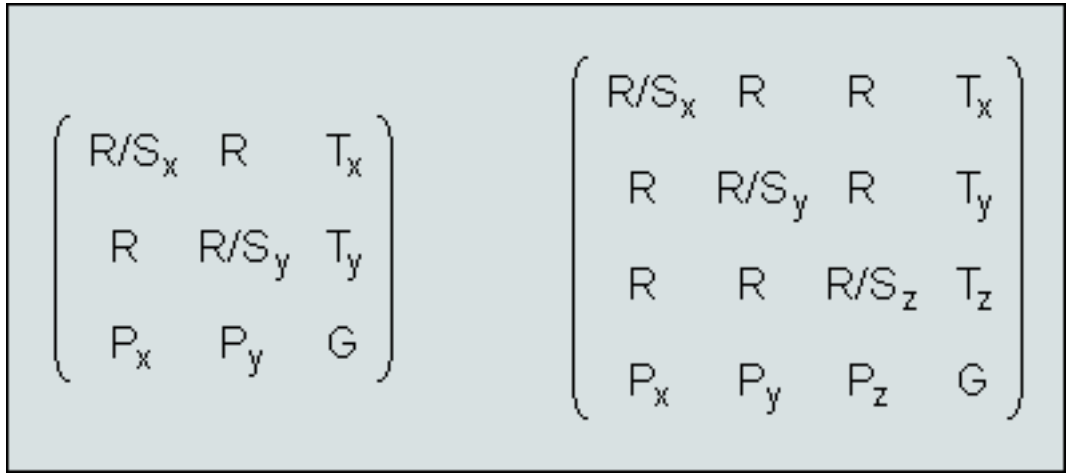
Coordinate Transform: $F_x=W_x$, $F_y=W_y-F_{ymax}$

What are our Challenges?

- Normalized Device Coordinates are very limited
- How do I move objects?
- How do I map objects to different coordinates?
- How can I look at a scene from any view position I would like?

What are our Challenges?

- Normalized Device Coordinates are very limited
- How do I move objects? – Matrix Transform
- How do I map objects to different coordinates? Matrix Transform
- How can I look at a scene from any view position I would like?
  - View Transformation



3D to 2d: Perspective Projection

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Provides foreshortening.

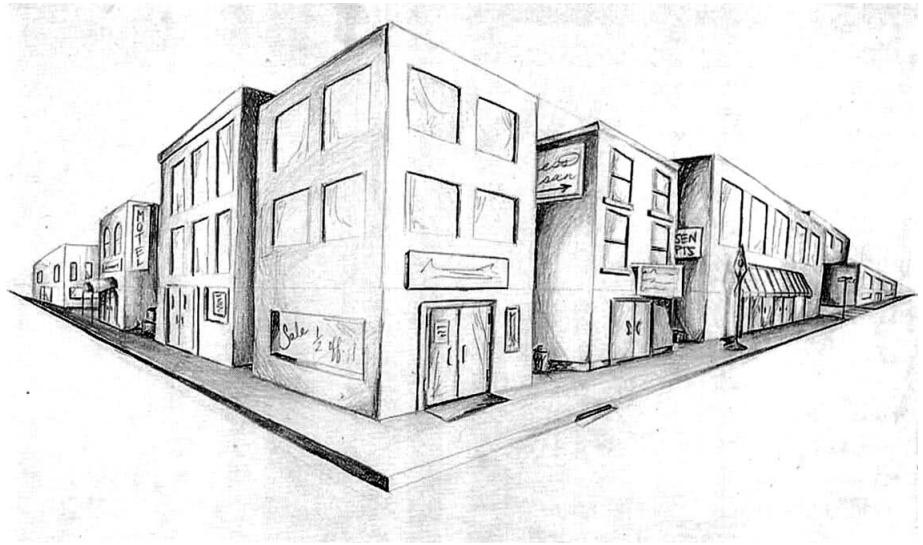
Has much higher visual realism.

What users expect.

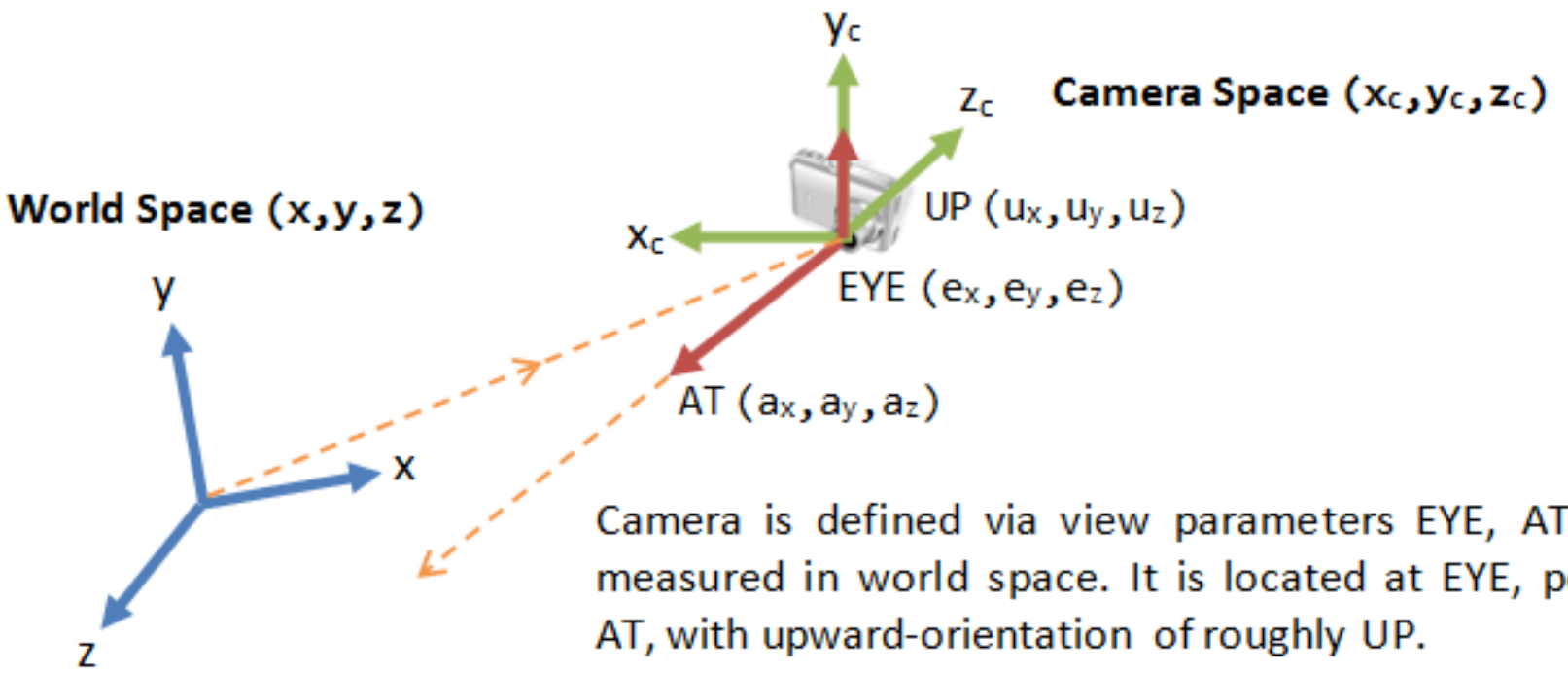
Very simple perspective projection.

X’ = x/z

Y’ = y/z



View Tranformation



Camera is defined via view parameters EYE, AT and UP, measured in world space. It is located at EYE, pointing at AT, with upward-orientation of roughly UP.

In the Camera space, camera is located at origin, pointing at $-z_c$, with upward-orientation of y_c.
z_c is opposite of AT, y_c is roughly UP.

Standard Sequence of Transforms

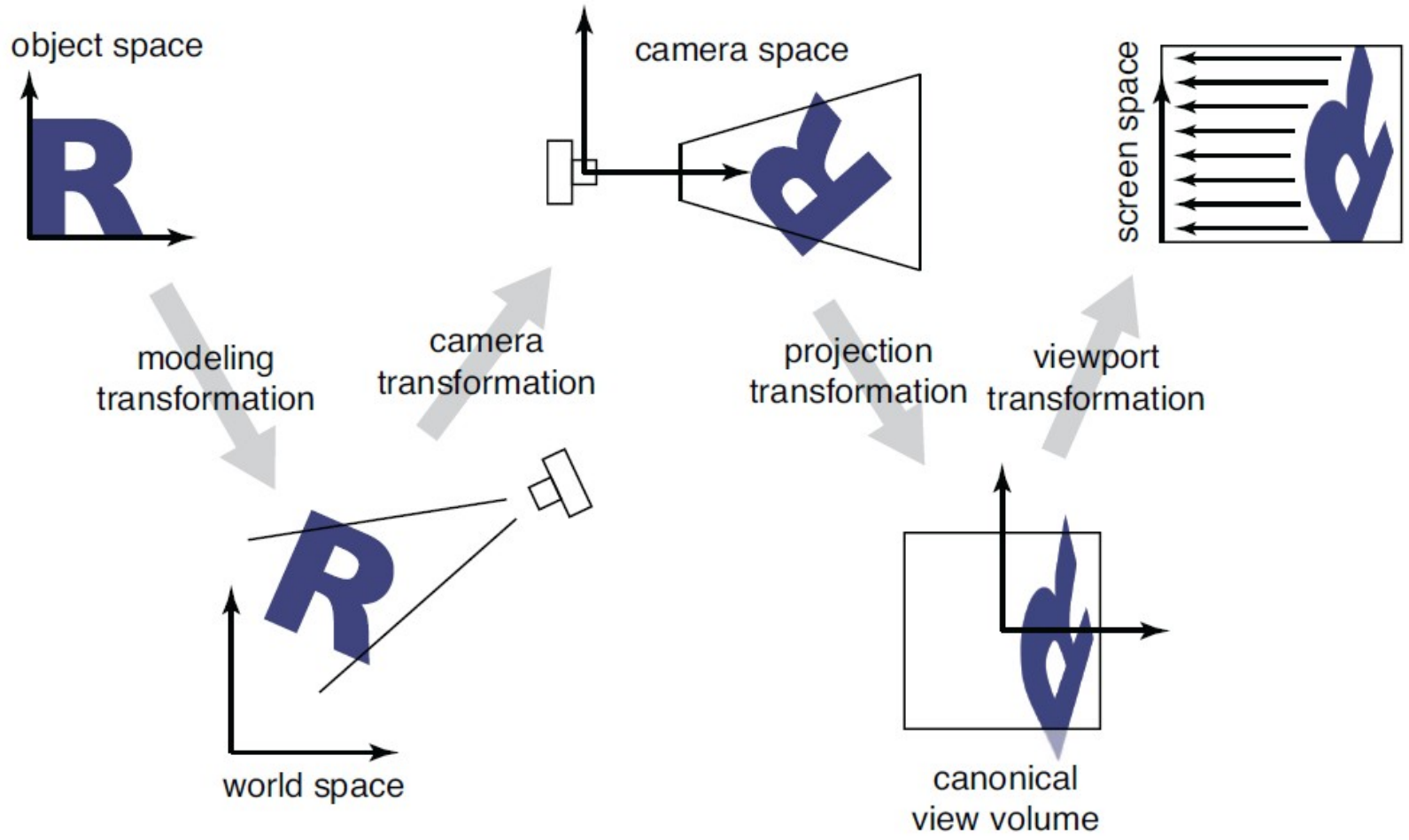
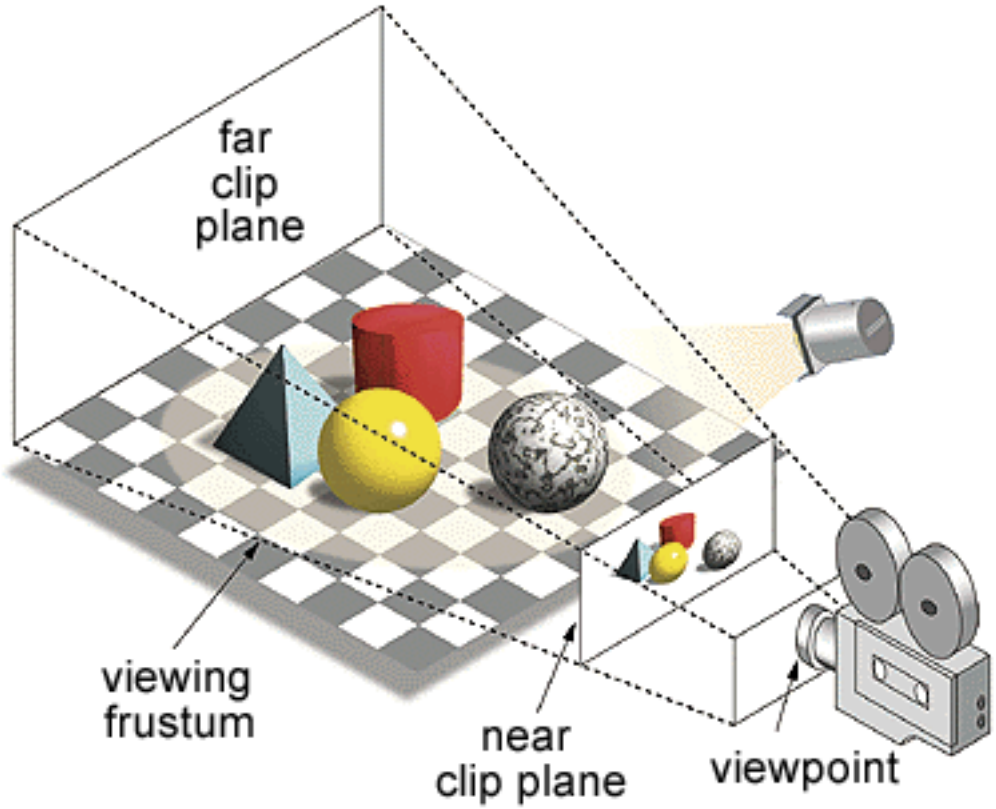


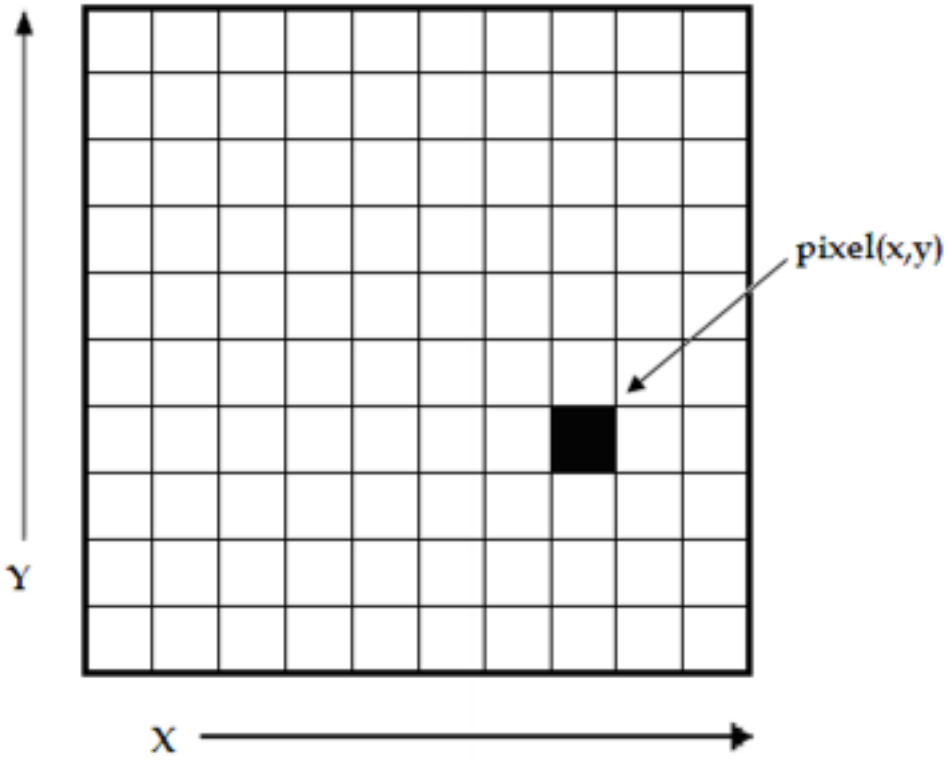
Image credits: http://www.cs.cornell.edu/courses/cs4620/2009fa/lectures/10viewing.pdf

Now Where are We?

3D Scene – Abstract Model



Framebuffer – Matrix of Screen Pixels



In Computer Graphics: If it looks right then it is right

What are our Challenges?

- How do we represent complex Geometry?
- How do we integrate Lighting?
- How do we select colors for the Geometry we want to draw?
- How do we handle overlapping Triangles?

What are our Challenges?

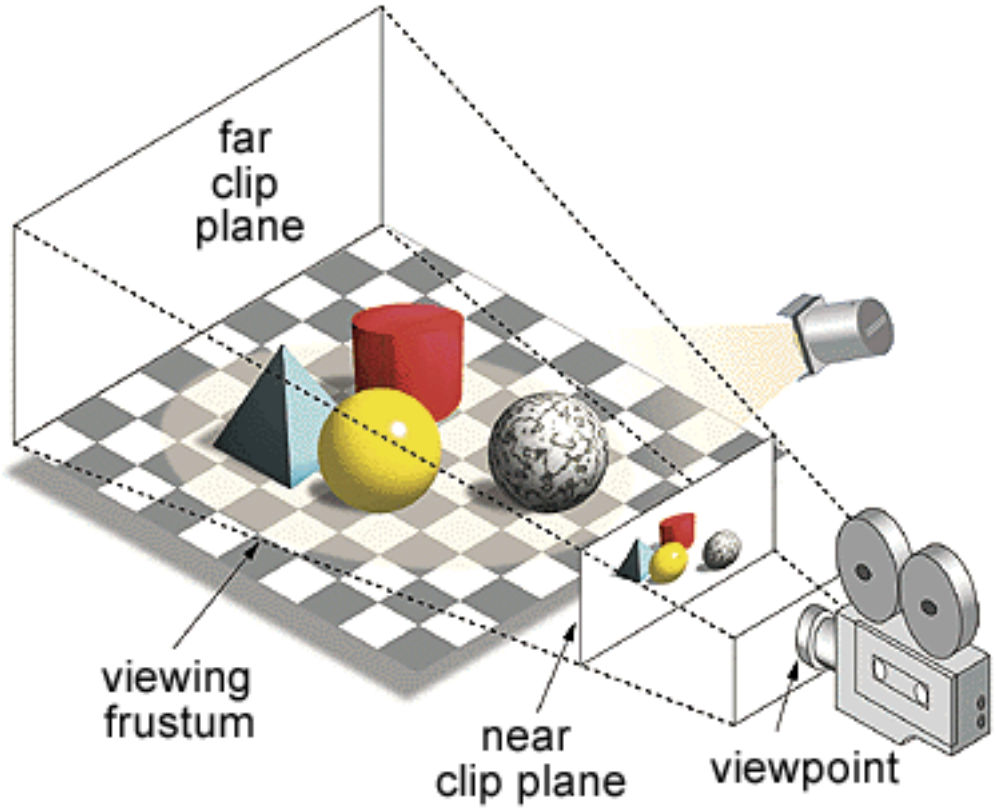
- How do we represent complex Geometry?
  - Modeling = easiest method is to use Triangles.
- How do we integrate Lighting?
  - Lighting Models = ambient, diffuse, specular(phong)
- How do we select colors for the Geometry we want to draw?
  - Shading Models = flat, gouraud, phong
  - Texture Mapping = select a color from an image.
  - Materials = define colors of the triangles and assign directly.
- How do we handle overlapping Triangles?
  - Painters algorithm
  - Z-Buffer → the standard.

How do we implement solutions?

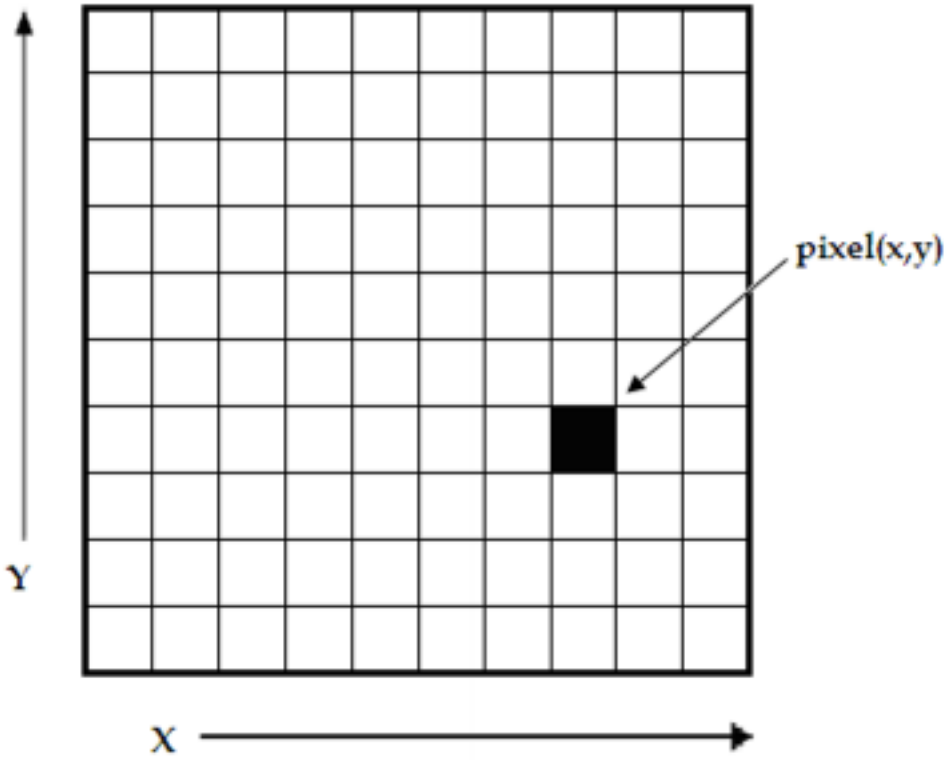
- How do we represent complex Geometry?
  - Triangles = vertices, normals, faces indices, use Program to create large models.
- How do we integrate Lighting?
  - Typically = we generate a light color for each vertex and save it for rasterization.
- How do we select colors for the Geometry we want to draw?
  - Rasterization = when rasterizing, we pick the color from:
    - Light value – interpolate.
    - Texture Map = interpolate texture coordinates.
    - Materials = use a triangle specific color and interpolate its value with the lighting.
- How do we handle overlapping Triangles?
  - Painters algorithm – sort all triangles before rendering and draw back to front.
  - Z-Buffer → For each screen pixel keep a Z value and overwrite the closer pixels.

So what can we do now?

3D Scene – Abstract Model



Framebuffer – Matrix of Screen Pixels



In Computer Graphics: If it looks right then it is right

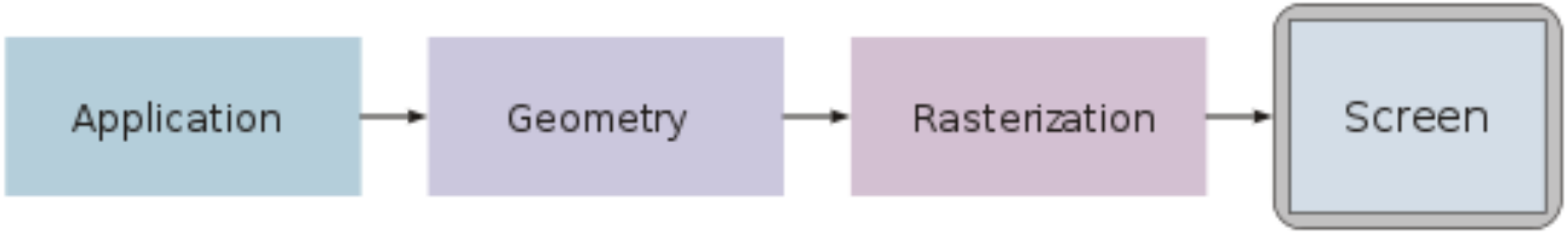
So how do we implement this so it is FAST!

So how do we implement this so it is FAST!

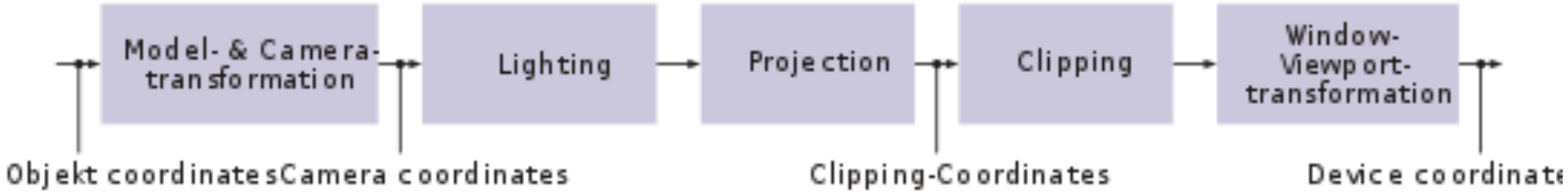
- We Introduce the Graphics Pipeline
  - gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);

Graphics Pipeline: process to draw scene.

General Pipeline



Geometry Pipeline



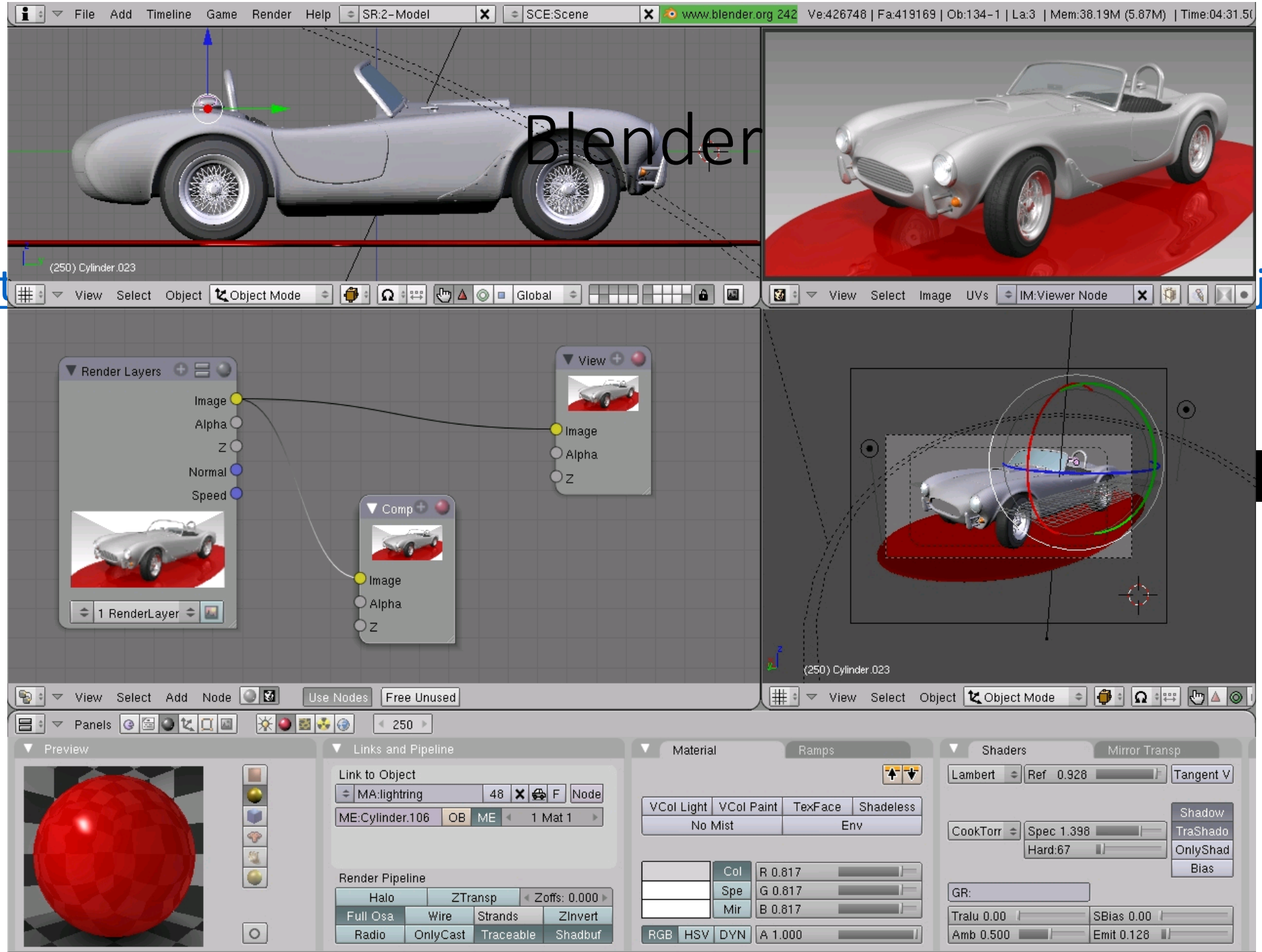
What are our Challenges?

- How do we represent complex Geometry?
  - Modeling = easiest method is to use Triangles.
- How do we integrate Lighting?
  - Lighting Models = ambient, diffuse, specular(phong)
- How do we select colors for the Geometry we want to draw?
  - Shading Models = flat, gouraud, phong
  - Texture Mapping = select a color from an image.
  - Materials = define colors of the triangles and assign directly.
- How do we handle overlapping Triangles?
  - Painters algorithm
  - Z-Buffer → the standard.

How do we represent Geometry?

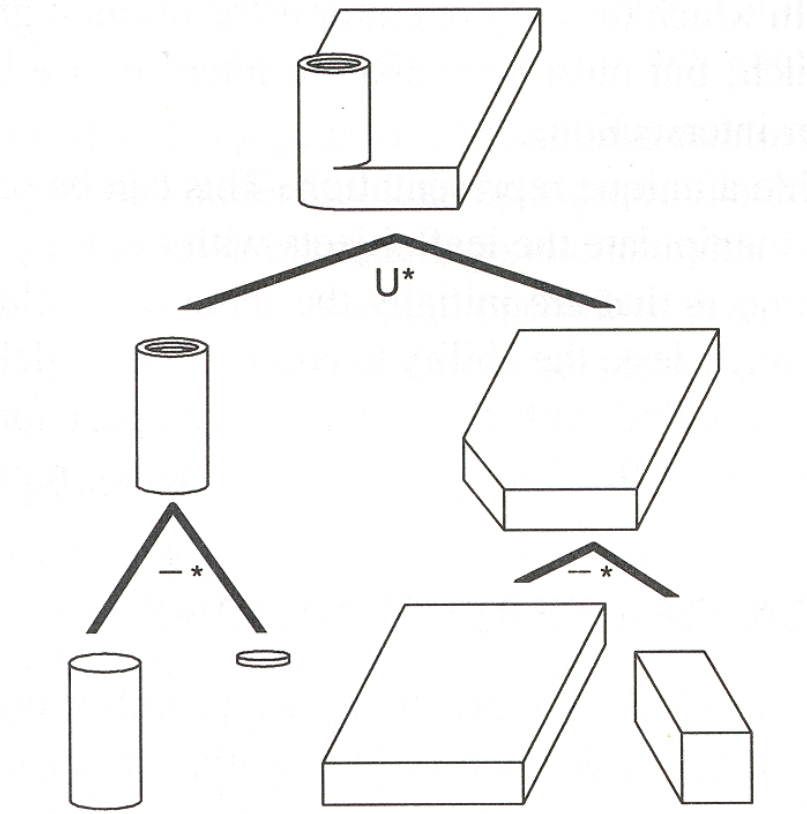
- Modeling
  - Collections of triangles defined by: Vertices, Edges, Faces, Normals
    - A set of vertices, a set of triangle indices = bare minimum geometry.
  - Normals are present for what purposes?
- How to create complex models?
  - Create by hand
  - Use a program = Maya or Blender
  - Scan the real world = Lidar Scanner and post-process.
  - Collections of photographs = photogrammetry.
- Complex Geometry → eventually all translated to triangles.
  - NURBS
  - Parametric Surfaces
  - Subdivision Surfaces
  - Implicit Surfaces
  - Constructive Solid Geometry

- htt jpg


Blender
File Add Timeline Game Render Help SR:2-Model SCE:Scene www.blender.org 242 Ve:426748 | Fa:419169 | Ob:134-1 | La:3 | Mem:38.19M (5.87M) | Time:04:31.5
(250) Cylinder.023
View Select Object Object Mode Global
View Select Image UVs IM:Viewer Node
Render Layers
Image
Alpha
Z
Normal
Speed
1 RenderLayer
View
Image
Alpha
Z
Comp
Image
Alpha
Z
View Select Add Node Use Nodes Free Unused
(250) Cylinder.023
View Select Object Object Mode
Panels 250
Preview
Links and Pipeline
Link to Object
MA:lightring 48 F Node
ME:Cylinder.106 OB ME 1 Mat 1
Render Pipeline
Halo ZTransp Zoffs: 0.000
Full Osa Wire Strands ZInvert
Radio OnlyCast Traceable Shadbuf
Material Ramps
VCol Light VCol Paint TexFace Shadeless
No Mist Env
Col R 0.817
Spe G 0.817
Mir B 0.817
RGB HSV DYN A 1.000
Shaders Mirror Transp
Lambert Ref 0.928 Tangent V
CookTorr Spec 1.398 Hard:67
Shadow
TraShado
OnlyShad
Bias
GR:
Tralu 0.00 SBias 0.00
Amb 0.500 Emit 0.128

Constructive Solid Geometry (CSG)

- Represent solid object as hierarchy of boolean operations
  - Union
  - Intersection
  - Difference



How Do We Select Colors for Pixels?

- Assign Simple Materials = simply store an r,g,b color with each object or face.
- Shading Models → controls how colors are 'modified'
  - Flat Shading
  - Gouraud Shading
  - Phong Shading
- Texture/Image Maps
  - Assign s,t texture coordinates to Vertices and interpolate.
  - What are types of things we can use Image maps for?
- Procedural Textures
  - Pixel = f(s,t) where f is a function → I = cos(theta)*sin(phi)
- Programmable Fragment Shaders
  - Use programmable code to calculate pixel color = most powerful.

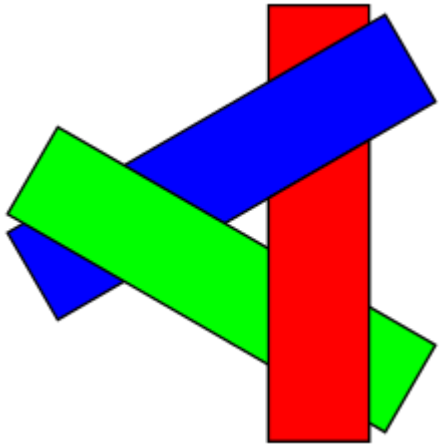
How do we introduce Lighting?

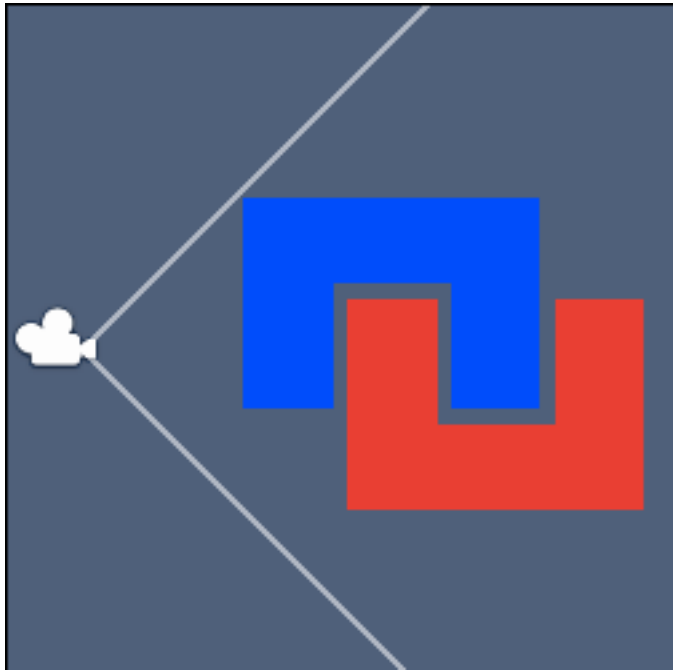
- Lighting Components
  - Ambient = a general representation for the common light in the room.
  - Diffuse = the light that reflects in all direction from a surface.
  - Specular = the light that reflects differently based on the viewing angle.

- Lighting Models
  - Ambient Lighting Model → Ia = Ka*Ia
  - Diffuse Lighting Model → Id = Kd * Id*Dot(Idir, Normal)
  - Phong Lighting Model → Is = Ks * Dot(2*(Dot(Idir,N)N –Idir),Idir)^alpha*Is

How Deal With Overlapping 3D Triangles

- Painters Algorithm
  - What is an example of this failing?
- Z Buffer Algorithm
  - What is an example of this failing?





Color Spaces

- RGB = Red, Green, Blue
- CMYK = Cyan, Magenta, Yellow, Black
- HSV = Hue, Saturation, Value
- HSI = Hue Saturation, Intensity
- LAB = Luminance, A=green-red, B = blue-yellow

Simple Linear Interplation

- Linear interpolation → c = a + t*b : t is a parameter from 0 to 1.0
- Bilinear Interpolation
- Trilinear Interpolation

Simple Vector Arithmetic

- Vector addition = simply addition of all vector terms.
- Vector multiplication = Cross product
- Vector Normalization
- Angle between two products = Dot Product

What are Affine Transformations?

- What do Affine Transforms Preserve?

What are Affine Transformations?

- What do Affine Transforms Preserve?
  - Points, straight lines, parallel lines, planes.
  - Is the Perspective Projection an Affine transform?

- Examples of Affine transforms:
  - Translation
  - Scaling
  - Rotations
  - Shears
  - Our typical Matrix transformations.

So What Can We Do Now?

- ????????
- You Tell Me

What Can’t We Do Yet?

- Shadows
- Reflections
- Anti-aliasing
- Scene Graphs
- Binary Space Partitioning
- Advanced Lighting → Diffuse/Diffuse interaction
- Stereo Systems
- Ray tracing and volume tracing.

What Can’t We Do Yet?

- Shadows --> Draw the Scene Twice.
- Reflections → Draw the Scene once for every reflective surface.
- Anti-aliasing → Over-render the scene and scale down.
- Scene Graphs → Organize complex Scenes with Trees.
- Binary Space Partitioning → Rapid advanced modeling.
- Advanced Lighting → Diffuse/Diffuse interaction
- Stereo Systems → Use Two Cameras.
- Ray tracing and volume tracing.

The End

- Work on your final projects
- Study for Exam
- Practice Exam Available on Thursday