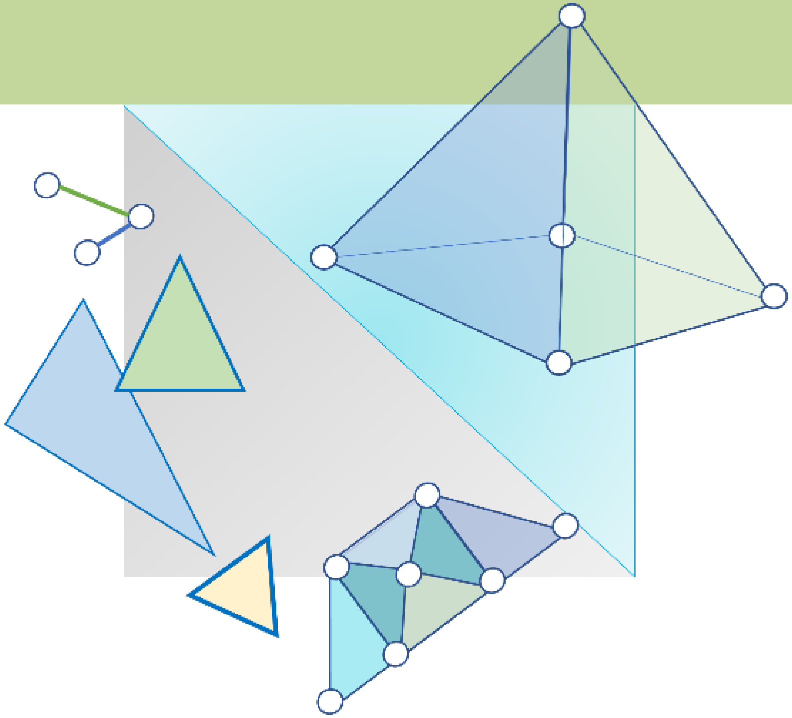
# CITS3003 Graphics & Animation

Introduction
and
Admin Matters



# Content

- Introduction to the unit
- Introduction to image formation
- Introduction to OpenGL

# Teaching Team

**Naeha Sharif**
Unit Coordinator & Lecturer

Room 1.05, First Floor
CSSE building

Consultation Hour
3:00 - 4:00pm Thursdays

Email:
naeha.sharif@uwa.edu.au

**David Charkey**
Lab Facilitator

**Jasper Paterson**
Lab Facilitator

# Timetable

| | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| 8:00 AM | | | | | |
| 9:00 AM | | | | | |
| 10:00 AM | CITS3003_SEM-1_CR - Laboratory Venue: CSSE: [ 203] (Weeks 10-15,17-21) Day: Monday 10:00 - 12:00 CSSE 203 | CITS3003_SEM-1_CR - Lecture Venue: MATH: [ G401 (Weeks; CITS3003_SEM-1_CR_OL - Lecture Venue: (Weeks 9-15,17-21) | | CITS3003_SEM-1_CR_OL - Laboratory Venue: (Weeks 10-15,17-21) Day: Thursday 10:00 - 12:00 Online | |
| 11:00 AM | | Weatherburn Lecture Theatre | | | |
| 12:00 PM | | CITS3003_SEM-1_CR - Laboratory Venue: CSSE: [ 203] (Weeks 10-15,17-21) Day: Tuesday 12:00 - 14:00 CSSE 203 | | | |
| 1:00 PM | CITS3003_SEM-1_CR - Lecture Venue: MATH: [ G401 (Weeks; CITS3003_SEM-1_CR_OL - Lecture Venue: (Weeks 9-15,17-21) | | | | |
| 2:00 PM | Weatherburn Lecture Theatre | CITS3003_SEM-1_CR - Laboratory Venue: CSSE: [ 203] (Weeks 10-15,17-21) Day: Tuesday 14:00 - 16:00 CSSE 203 | | | CITS3003_SEM-1_CR - Laboratory Venue: CSSE: [ 203] (Weeks 10-15,17-21) Day: Friday 14:00 - 16:00 CSSE 203 |
| 3:00 PM | | | CITS3003_SEM-1_CR_OL - Laboratory Venue: (Weeks 10-15,17-21) Day: Wednesday 15:00 - 17:00 Online | Consultation Hour | |
| 4:00 PM | | | | | |
| 5:00 PM | | | | | |

# Other Admin Matters

- Recorded lectures will be on LMS
  - Check regularly for announcements and updates
  - Lectures uploaded every teaching week
- Unit webpage (link)
  - Labs are already available on the unit webpage
  - Lectures updated every teaching week
- David's guide to set up your personal system with OpenGL/Linux (link)

# Assessment

- The assessment will consist of:
  - 10%: Mid-semester test (week 06)
  - 40%: Programming project (due in week 12)
  - 50%: Final exam
- How mid-sem test will be conducted exactly?
  - Will have to wait on that.

# Breakdown of Lectures

1. Introduction & Image Formation
2. Programming with OpenGL
3. OpenGL: Pipeline Architecture
4. OpenGL: An Example Program
5. Vertex and Fragment Shaders 1
6. Vertex and Fragment Shaders 2
7. Representation and Coordinate Systems
8. Coordinate Frame Transformations
9. Transformations and Homogeneous Coordinates
10. Input, Interaction and Callbacks
11. More on Callbacks
12. Mid-semester Test
13. 3D Hidden Surface Removal
14. Computer Viewing

- Study break

15. Programming Project Discussion
16. Shading
17. Shading Models
18. Shading in OpenGL
19. Texture Mapping
20. Texture Mapping in OpenGL
21. Hierarchical Modelling
22. 3D Modelling: Subdivision Surfaces
23. Animation Fundamentals and Quaternions
24. Skinning

# Project and Labs

- There will be a total of 5 labs, starting from week#2.
- Lab sheets will be provided (along with the solutions) link

- Labs are not assessed but it is important to complete them to be able to complete the project.

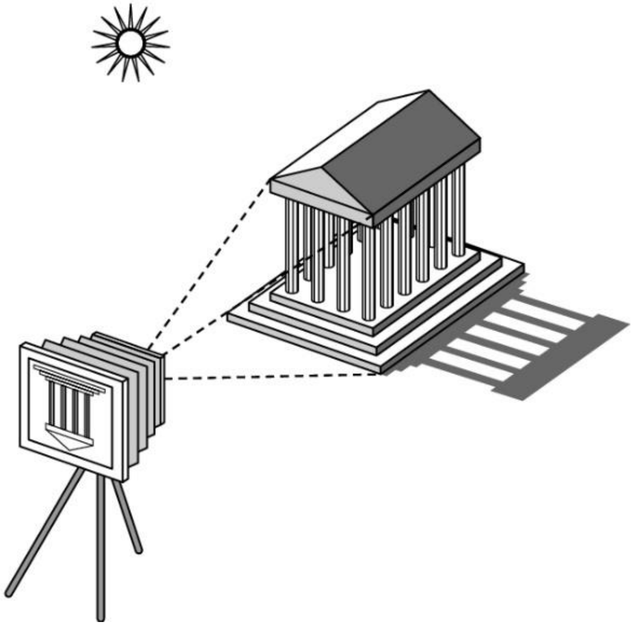
- Project will be released in week 07 but discussed in week 08.

# Introduction to Image Formation

# Image Formation

- In computer graphics, we form images which are generally two dimensional using a process analogous to how images are formed by physical imaging systems
  - Cameras
  - Microscopes
  - Telescopes
  - Human visual system

# Elements of Image Formation



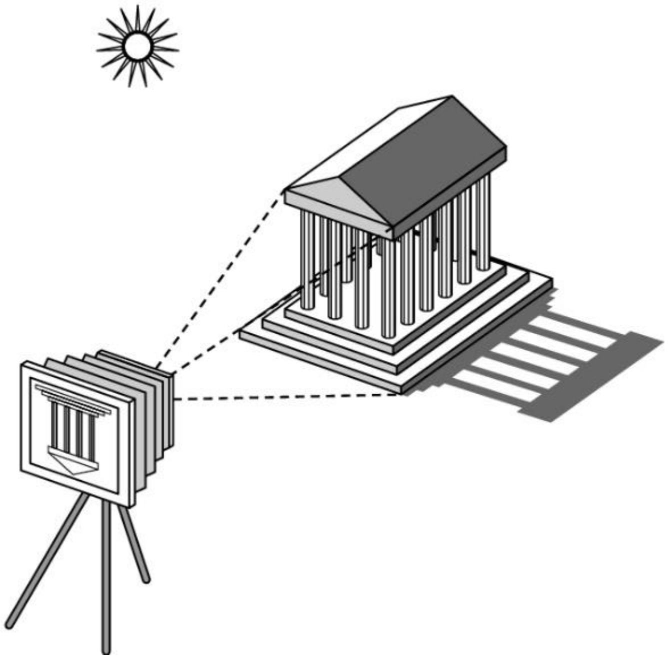
1. Objects
2. Viewer
3. Light source(s)

- Attributes that govern how light interacts with the material in the scene

*Note the* ***independence*** *of the objects, the viewer, and the light source(s)*

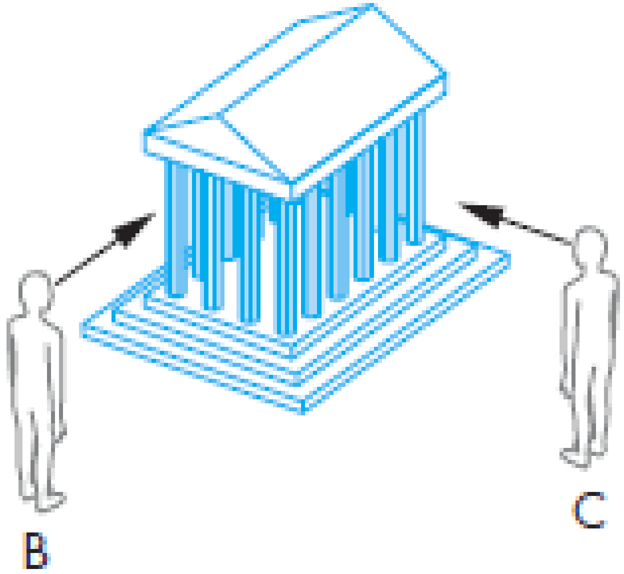
# Objects

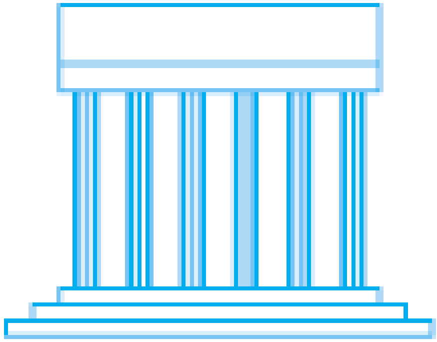
- Objects in space are independent of any image formation process and of any viewer
- A set of locations (vertices) in space is sufficient to define or approximate most objects



# Viewer

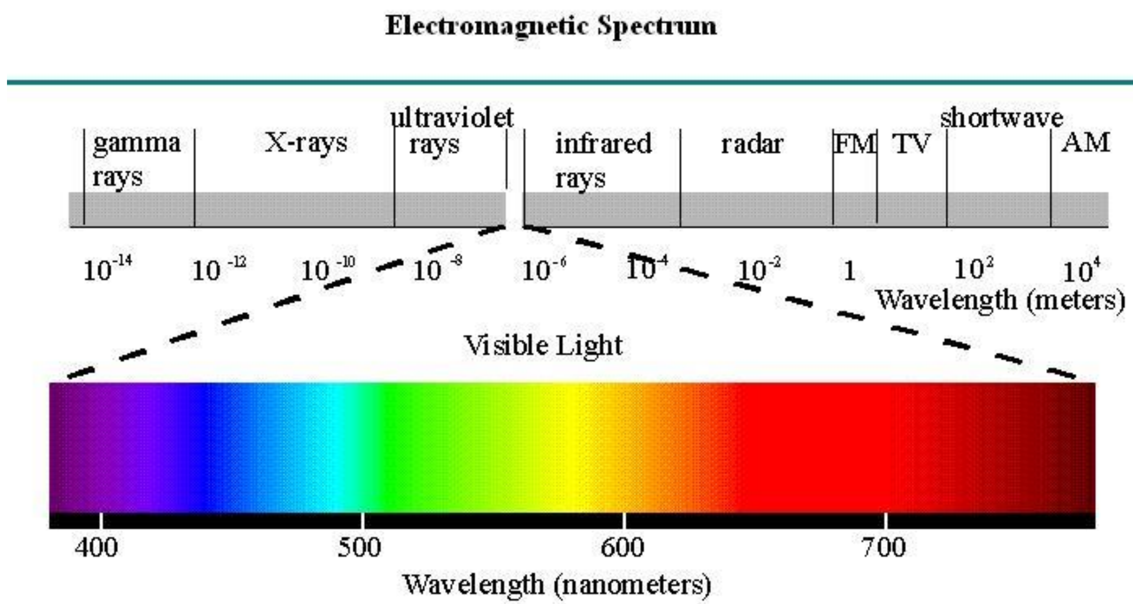
- To form an image, we must have someone or something that is viewing our objects, be it a human, a camera, or a digitizer. It is the **viewer** that forms the image of our objects.



# Light

- If there were no light sources, the objects would appear dark
- Light is the part of the electromagnetic spectrum that causes a reaction in our visual system
- Generally, these are wavelengths in the range of about 350-750 nm (nanometers)
  - Long wavelengths appear as reds and short wavelengths as blues



# Light in Geometric Optics

- Geometric optics models light sources as ‘emitters of light energy’, each of which have a fixed intensity.
- Light travels in straight lines, from the sources to those objects with which it interacts.
- An ideal point source emits energy from a single location at one or more frequencies equally in all directions.
- A particular source is characterized by
  - the intensity of light that it emits at each frequency
  - that light’s directionality

# Elements of Image Formation

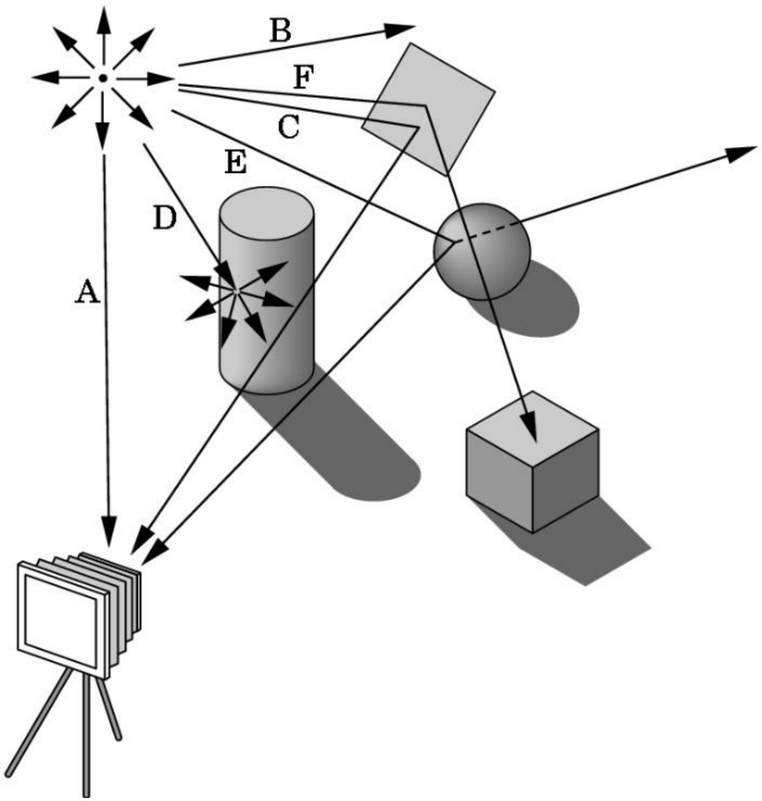
**Advantages (of modeling independent components):**

- Separation of objects, viewer, light sources (can model them separately).
- Leads to simple software API
  - Can specify objects, lights, camera, attributes separately
  - Let implementation determine image by interaction
- Leads to fast hardware implementation
- Two-dimensional graphics becomes a special case of three-dimensional graphics

# Ray Tracing: Physical Approach to Image Formation

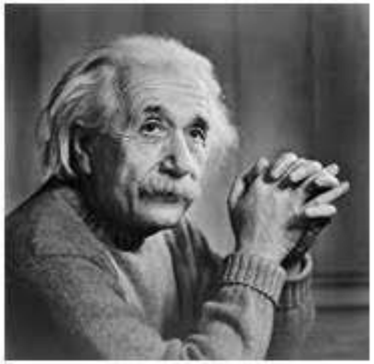
One way to form an image is to follow rays of light from a source, finding which rays enter the camera lens.

However, rays of light may have multiple interactions with objects, get absorbed, or go to infinity.



# Luminance Images

- Luminance Image
  - Monochromatic
  - Values are gray levels
  - Analogous to working with black and white film or television



# Color Images

- Color Image
  - Has perceptional attributes of hue, saturation, and lightness

<u>Hue</u>
another word for color
(wavelength dependent)

<u>Saturation (Chroma)</u>
the intensity or purity of hue
(100% pure = no addition of gray)

<u>Lightness (Value)</u>
relative degree of black/white

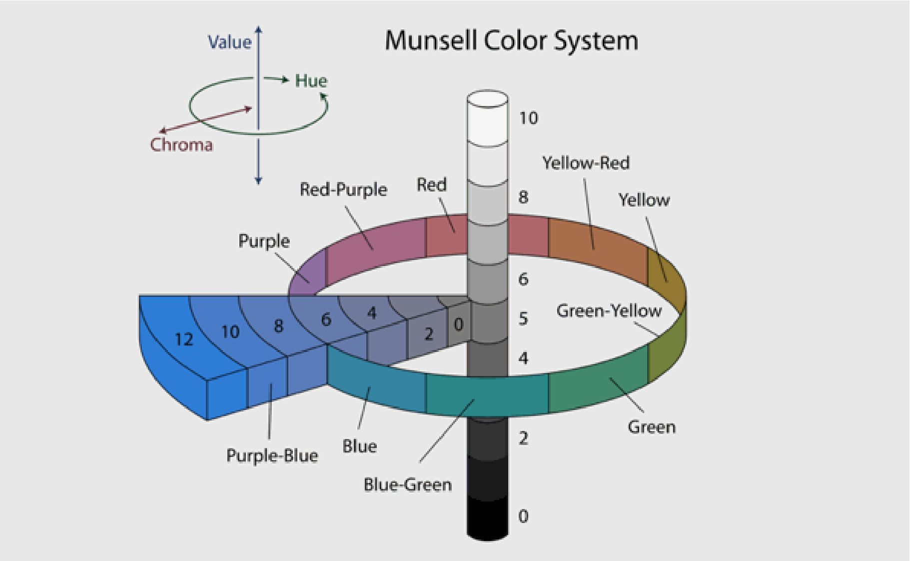
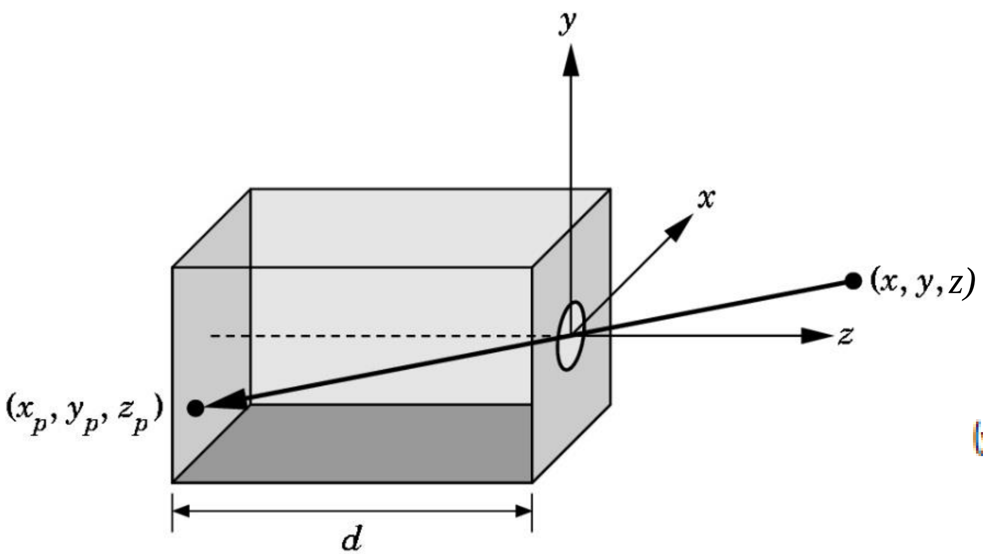
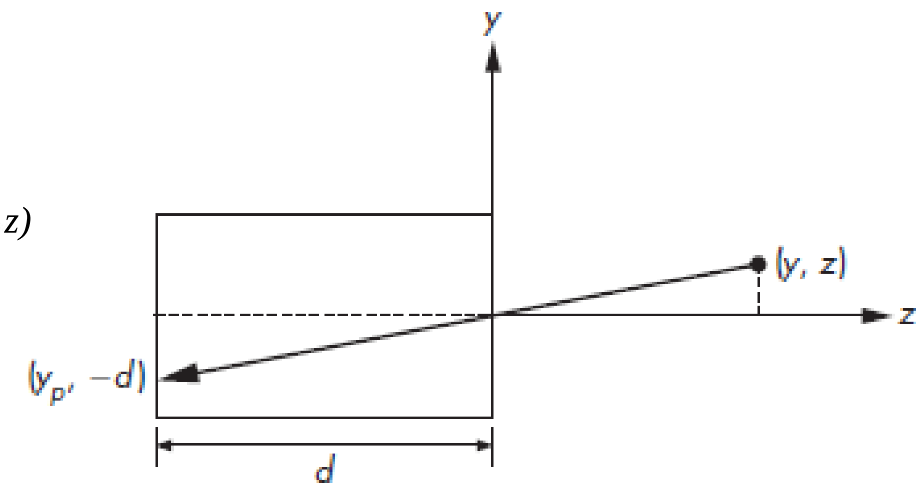



Image from (https://vanseodesign.com/web-design/hue-saturation-and-lightness/)

# Imaging System

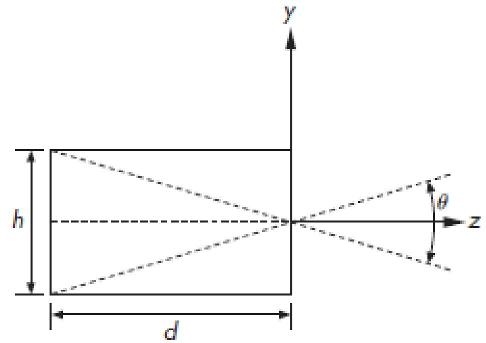
# Pinhole Camera




- Use trigonometry to find projection of point at $(x, y, z)$
- $x_p/z_p = x/z \quad y_p/z_p = y/z \quad z_p = -d$
- These are equations of simple perspective
- The point $(x_p, y_p, -d)$ is called the **projection** of the point $(x, y, z)$.

# Pinhole Camera (cont..)




- The **field, or angle of view** of our camera is the angle made by the largest object that our camera can image on its film plane.

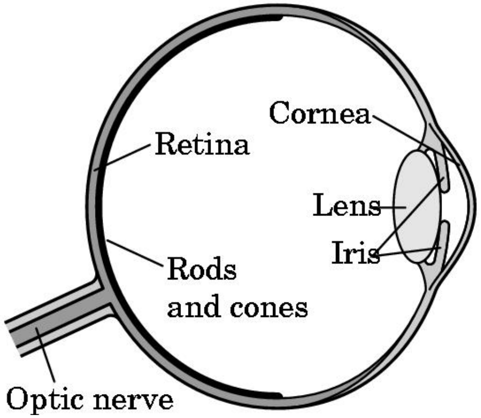
$$\theta = 2\tan^{-1}\frac{h}{2d}.$$

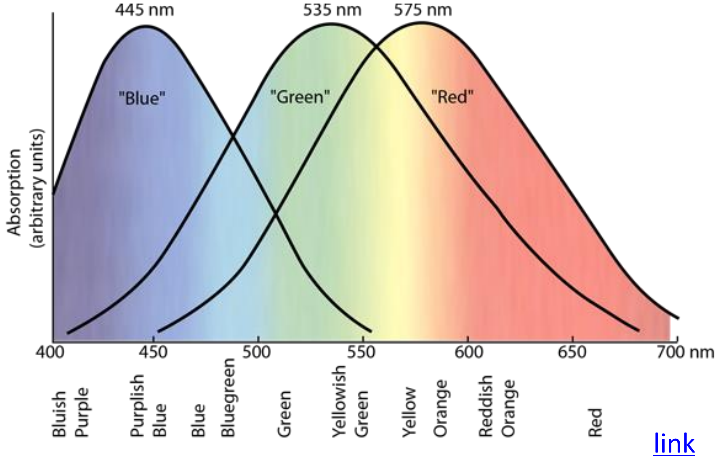
- The ideal pinhole camera has an infinite **Depth Of Field (DOF)**
  – DOF is the distance between the nearest and the farthest objects that are in acceptably sharp focus in an image

- The pinhole camera has two disadvantages:
  – It admits only a single ray from a point source—almost no light enters the camera.
  – The camera cannot be adjusted to have a different angle of view

# Human Visual System

- The human visual system has two types of sensors
  - Rods (up to 125M)
    - Monochromatic, night vision
  - Cones (6M+)
    - Color sensitive
    - Three types of cones
    - Only three values (the ***tristimulus*** values) are sent to the brain
- That is, we need only match these three values
  - →Need only three *primary* colors

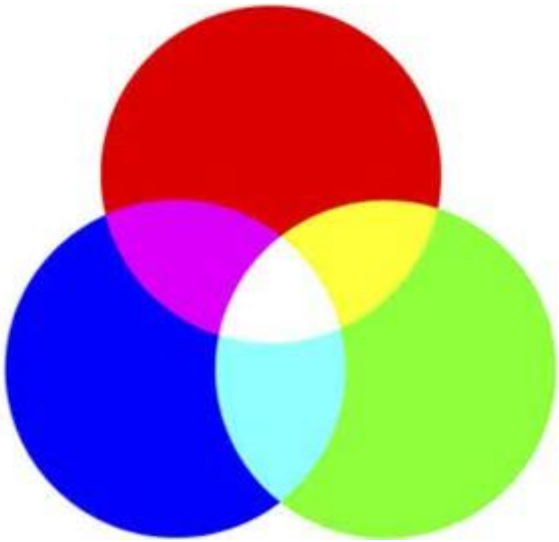


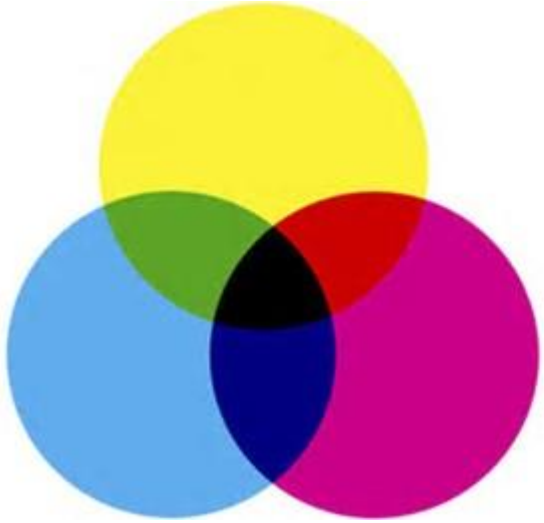

link

# Additive and Subtractive Color

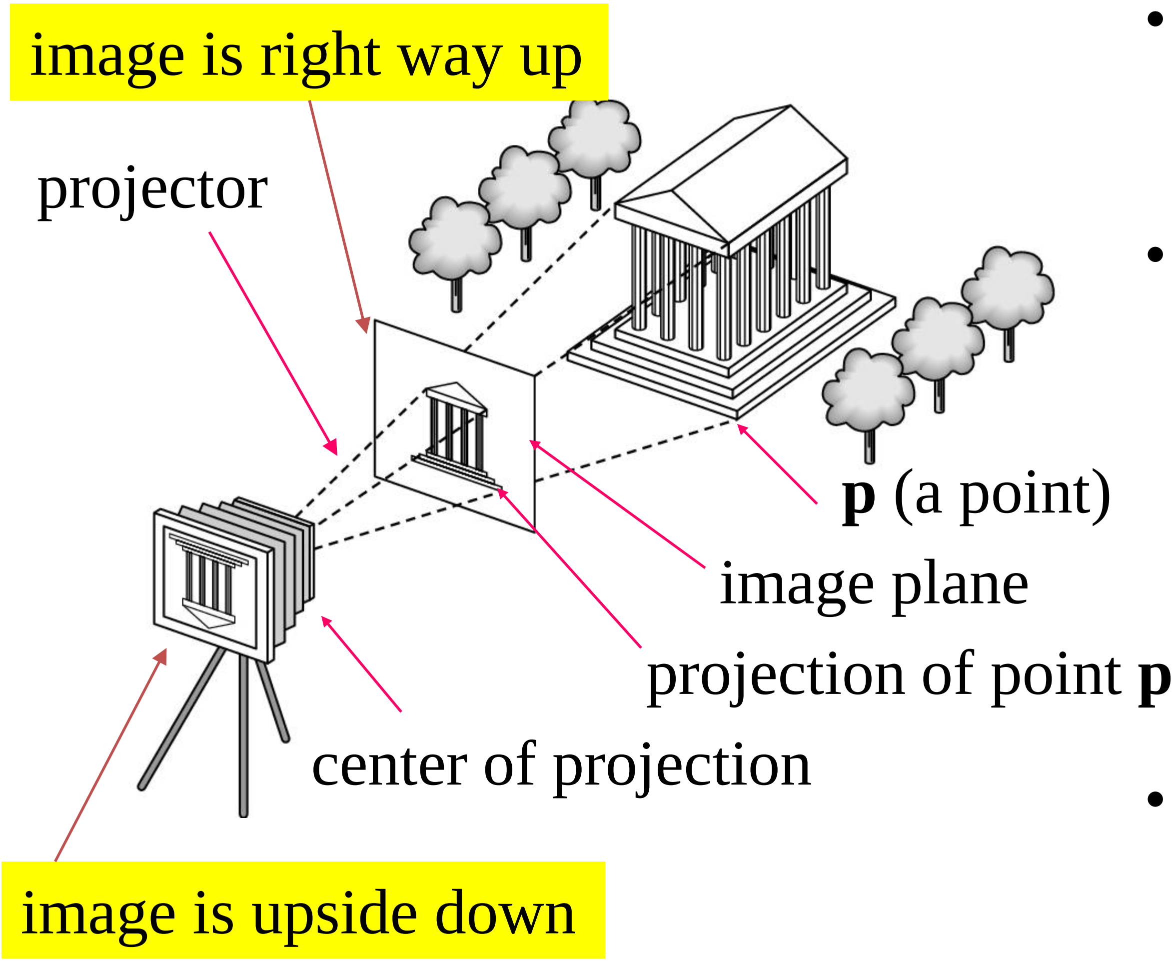
- **Additive color**
  - Form a color by adding amounts of three primaries
    - CRTs, projection systems, positive film
  - Primaries are Red (R), Green (G), Blue (B)



- **Subtractive color**
  - Form a color by filtering white light with cyan (C), Magenta (M), and Yellow (Y) filters
    - Light-material interactions
    - Printing
    - Negative film



# Synthetic Camera Model



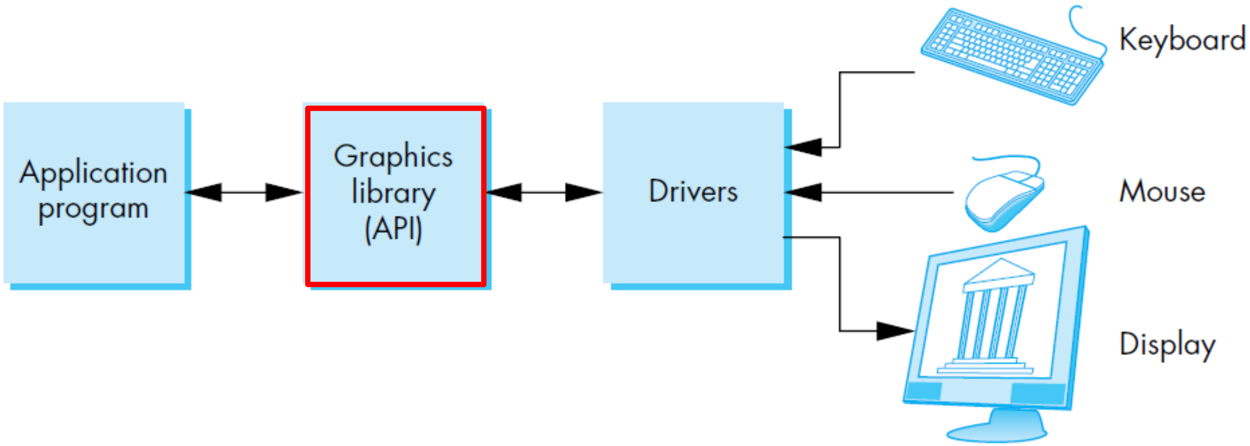

- OpenGL uses the synthetic pin hole camera model
- Since the image of the object is flipped relative to the object on the back of the camera, we draw another plane in front of the lens.
- With this synthetic camera model, the object is the right way up.

# Introduction to OpenGL

# What is OpenGL

OpenGL is a platform-independent Application Programmers' Interface (API) that

- Is close enough to the hardware to get excellent performance
- Provides a link between the low-level graphics hardware and the high-level application program that you write
- Is easy to use




- Most of the concepts related to OpenGL covered in week 01 are for introduction purpose.
- Many of these concepts will be repeated in more detail in the weeks to follow.

# Versions of OpenGL

- Latest versions are completely shader-based:
  - No default shaders
  - Each application must provide both a vertex and a fragment shader i.e., you must additionally write vertex and fragment shader programs
- OpenGL ES
  - Is suitable for embedded systems
  - Version 1.0 is a simplified version of OpenGL 2.1
  - Version 2.0 is a simplified version of OpenGL 3.1
- OpenGL 4.1 and 4.2
  - Add geometry shaders and tessellator
- For labs and project, Version 3+ are all ok!

# Versions of OpenGL (cont.)

- WebGL
  - Is a derivative of OpenGL ES version 2.0
  - Provides JavaScript bindings for OpenGL functions, allowing an HTML page to render images using any GPU resources available on the computer where the web browser is running

- WebGL is not included in the curriculum

# OpenGL Libraries

- OpenGL core library
  - OpenGL32 on Windows
  - GL on most unix/linux systems
- OpenGL Utility Library (GLU)
  - Provides higher level drawing routines for OpenGL (e.g., simple positioning of the camera)
- Links with window/windowing system
  - GLX for X window systems
  - WGL for Windows
  - AGL for Macintosh

# GLEW

- GLEW is an OpenGL Extension Wrangler Library
- GLEW makes it easy to access OpenGL extensions available on a particular system
- Application only needs to include glew.h and run a glewInit()

# GLUT

- OpenGL Utility Toolkit (GLUT)
  - A window system independent toolkit for writing OpenGL programs
  - Implements a simple windowing API for OpenGL
  - Makes it easy to learn and use OpenGL
  - Code is portable but GLUT slightly the functionality of a high-end toolkit for a specific platform
    - No slide bars

# freeGLUT

- GLUT was created long ago and has been unchanged
- freeglut updates GLUT
  - Added capabilities
    - Context checking

# Which Function is in which Library?

- You don’t need to memorize the functionalities of different OpenGL libraries
- Instead, you decide on your objects, lights and camera, then work out which OpenGL functions are required.
- Include libraries that contain your functions.
- For the practical issues you will have the OpenGL documentation to help.

https://docs.gl/

# Further Reading

“Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL” by Edward Angel and Dave Shreiner, 6$^{th}$ Ed, 2012

- Sec. 1.2 A graphics system
- Sec. 1.3 Images: Physical and Synthetic
- Sec. 1.4 Imaging Systems
- Sec. 1.5 The Synthetic Camera Model
- Sec. 1.6 The Programmer’s Interface