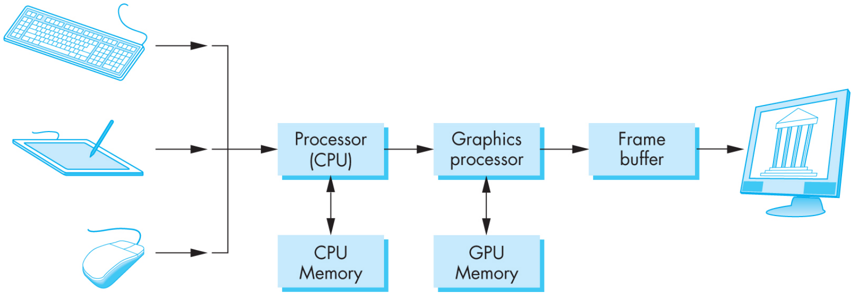
Outline

Applications of Computer Graphics

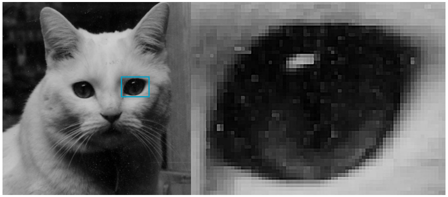
4 Major Areas:

- **Display of information**: Scientific data, maps
- **Design**: Engineering, architecture, art
- **Simulation and animation**: Games
- **User interfaces**

A simple graphics system



The frame buffer



Frame buffer : dedicated memory which stores pixel colors (and possibly more) which is to be read by the video controller of an output device (video monitor, etc.)
Typically arranged as a rectangular (or **raster**) array of values.

Pixels : point samples of a 2D image.

The frame buffer

Characterized by:

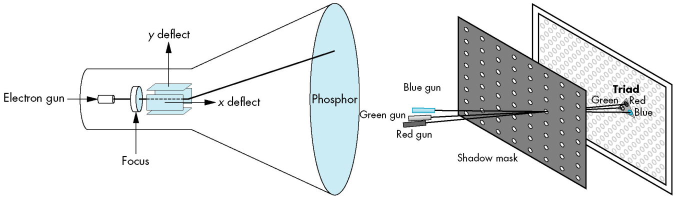
- **Resolution** : number of pixels in the buffer (width $\times$ height).
- **Depth** : number of bits per pixel.
- **Auxiliary buffers** : z-buffer, stencil buffer

RGB Color modes:

- 16 bit : 5-6-5 (the eye is more sensitive to green)
- 16 bit : 5-5-5-1 (1 bit alpha channel)
- 24 bit : 8-8-8
- 32 bit : 8-8-8-8 (8 bit alpha)

Output devices

Cathode Ray Tube (CRT)



Obsolete technology
Many alternatives : LCD, LED, Plasma, DLP.
Display trends : bigger, higher resolution, and brighter...

High Dynamic Range Output

In a real scene the ratio between brightest and darkest points can be $>$ 100,000:1

Most monitors can only display 1000:1

Image Credits : Brightside Technology

The Brightside product can display 60,000:1

Input devices

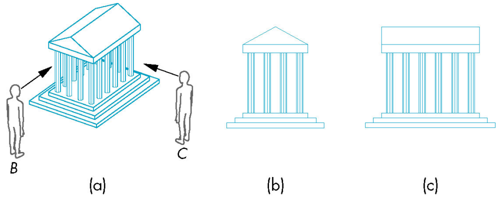
For interactive input or data acquisition.
The obvious examples:

- Keyboard
- Mouse
- Joystick

Some less obvious examples (not handled by glut) ...

Object and viewer

Our approach to 3D graphics will be to model objects and the viewer (camera) independently in 3D space.



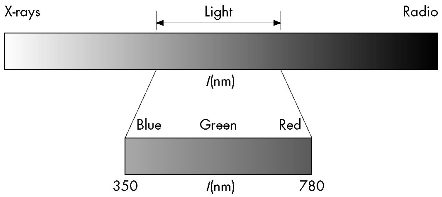
The models will be specified as sets of vertices, and the faces which connect them.

The vertices will be transformed and projected (using matrices) to form an image of the scene from the point-of-view of the camera.

Light

Modeling light is critical for image generation.

Light : a visible form of electromagnetic radiation.



Light may be composed of:

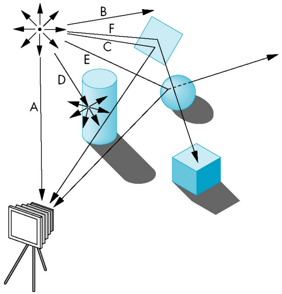
- a single wavelength (monochromatic)
- or a distribution of wavelengths.

Light transport mechanisms:

- Reflection
- Refraction
- Scattering

Ray tracing

A physically-based rendering approach:

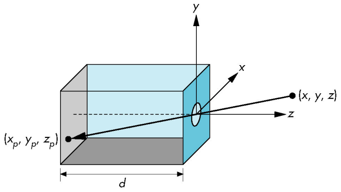


Ray tracing : follow light rays through the scene. The ray is reflected, refracted and scattered as it intersects various objects. Only the rays going through the image plane contribute to the image. This process is slow* - so **OpenGL takes a different approach**.

*Simple scenes can currently be ray-traced in real-time.

Pinhole camera

A simple model for imaging systems:



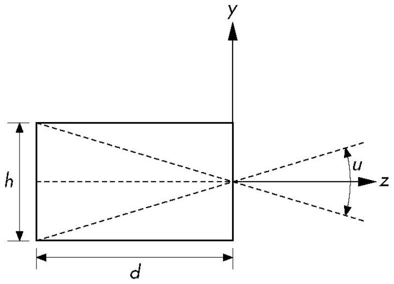
By similar triangles:

$$\frac{x_p}{d} = -\frac{x}{z} \tag{1}$$

$$\frac{y_p}{d} = -\frac{y}{z} \tag{2}$$

Pinhole camera

Field of view is an important characteristic of a camera.

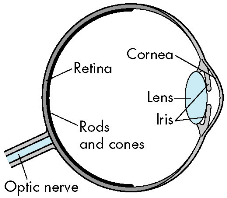


The **field of view**, u, depends on the height and depth of the box.

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

The human visual system

A much more complex imaging system:



- The cornea and lens focus light rays on the retina.
- The iris opens and closes the pupil to control the amount of incoming light.
- Light sensitive cells in the retina form an image.
  - Rod cells : sensitive to light intensity
  - Cone cells : sensitive to color

The human visual system

The tristimulus theory proposes that there are 3 types of cone cells, each sensitive to a different wavelength. Human experiments support this theory.

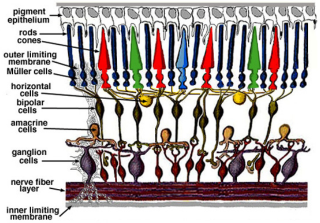
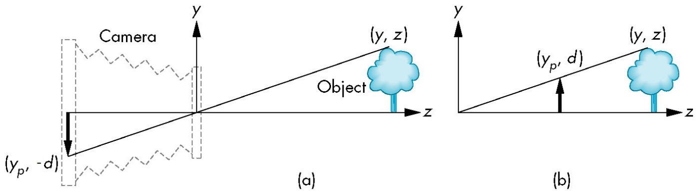


Fig. 2. Simple diagram of the organization of the retina.

The optic nerve contains about 1 million nerve fibers, while the retina has about 100 million photoreceptors. This suggests that some processing is done within the eye itself.

The synthetic camera

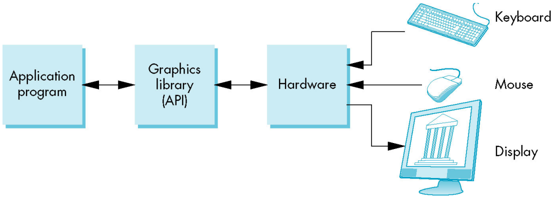
The camera model we will use is similar to the pinhole camera, **except**



The projection plane (image plane) is assumed to be in front of the center of projection. The image is **not** inverted as it is in the pinhole camera.

The application programmer's interface (API)

A high-level interface provides access to the hardware while shielding the programmer from low-level details.



The hardware-dependent details are handled by the driver. The same program may run with different hardware, provided there is an appropriate driver.

3D APIs

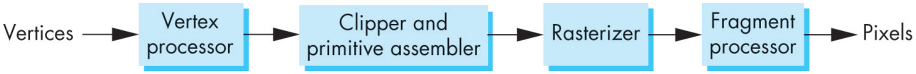
- **DirectX** includes many other facilities : Sound, networking, input. This makes it the API of choice for creating games.
- **OpenGL** is portable across many hardware and OS platforms, so it is more common in research.

Both allow us to specify:

- Geometric objects
- A camera
- Lights
- Material properties
- Rendering state (e.g. shaders)

Pipeline architecture : 4 stages

The purpose of the graphics pipeline is the same as the CPU pipeline : to improve throughput.

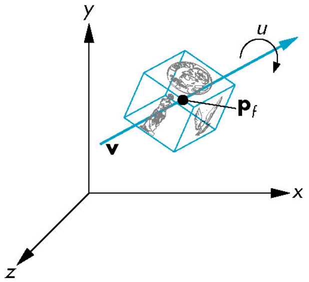


Stage 1 : Vertex Processor

- Vertex transformation (move models relaive to camera)
- Per-vertex lighting computations

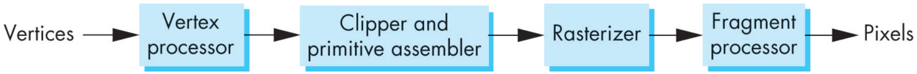
Coordinate systems, Modeling and Viewing Transformations

Object position, orientation and scale are represented by matrices.



The camera is represented by a matrix as well. Vertices are transformed using matrix-vector multiplication.

Pipeline architecture

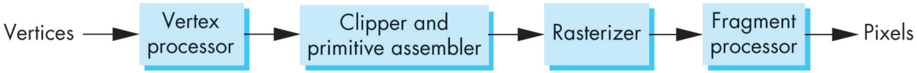


Stage 2 : Clipping and Primitive Assembly

Primitive : a set of vertices representing the basic building block of geometric objects. (E.g. triangles, lines, points)

- Clipping : compute new vertices for primitives restricted to the field of view of the camera.
- Primitive Assembly : group the new set of vertices into clipped primitives.

Pipeline architecture



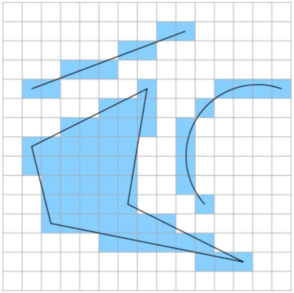
Stage 3 : Rasterization

Converts primitives into **fragments**.
”A fragment can be thought of as a potential pixel that carries with it (additional) information”

- ”potential pixel” : because it may not go into the frame buffer
- ”additional information” : texture coordinates, depth information

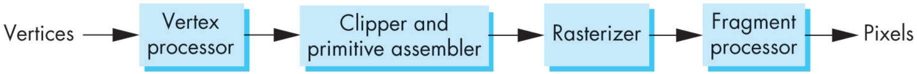
Rasterization

The process of converting a geometric primitive into a raster fragments.



Answers the question: Which pixel locations should light up if this primitive is visible?

Pipeline architecture



Stage 4 : Fragment Processing

- Determine final color.
- Change depth.

The programmable pipeline

In the past, the pipeline functionality was fixed. Today, the vertex processor and fragment processor are programmable.
This enables:

- Simple animation and physics simulation to be applied to vertices.
- Complex per-pixel lighting models.
- GPGPU programming : general purpose graphics processing unit programming for image processing, collision detection

Now (OpenGL 4.5, DirectX 12) other stages of the pipeline are programmable.

Shaders

The term **shaders** refers to small programs that run within a programmable stage of the pipeline.

- Vertex shader = vertex program
- Fragment shader = fragment program

OpenGL includes a high-level shading language (glsl) that shaders can be written in.

Vertex processing rate vs. pixel fill rate

It is possible to have the application frame rate limited by one or the other or both. Our goal is to keep both stages of the pipeline moving.

Consider 2 different scenes:

- Many very small triangles : many vertices, few pixels
- A few very large triangles : few vertices, many pixels

The bottleneck may also be in the application itself.

The main goal of this class

To learn to write high-performance graphics applications with as few bottlenecks as possible using modern OpenGL features.