# Lecture 1:
# Graphics Systems and Models

**Topics:**

1. Applications of computer graphics
2. Graphics systems
3. Camera models

Chapter 1 of Angel.

# Applications of computer graphics

**Computer Graphics**: all aspects of producing pictures or images using a computer.

### 1. Display

- architectural drawings, e.g. plan of a building
- maps: geographical information
- plotting statistical graphs, e.g. share prices
- medical images: Computed Tomography (CT), Magnetic Resonance Imaging (MRI)
- scientific visualization

**2. Design** (interaction important)

- Computer Aided Design (CAD)
- design of very-large-scale integrated (VSLI) circuits

**3. Simulation**

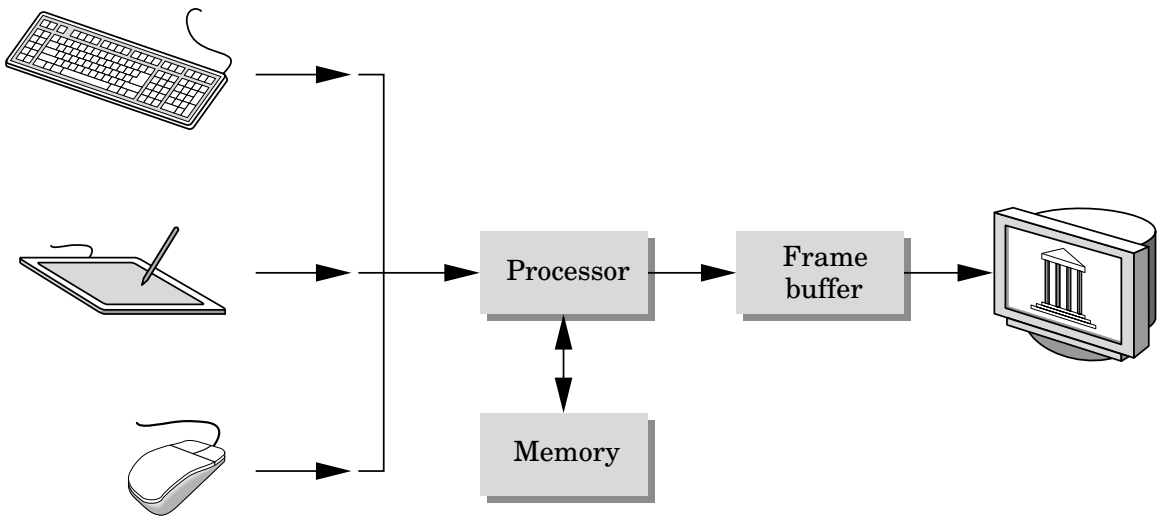
- flight simulation for training pilots
- computer games
- television and computer-animated films: Toy Story (Pixar), Ice Age
- virtual reality

**4. User interfaces**

- window-based operating systems: Microsoft Windows, Macintosh, X Windows
- internet browsers: Netscape, Explorer

# A Graphics System

1. Processor
2. Memory
3. Frame buffer
4. Output devices
5. Input devices



### Pixels and the Frame Buffer

Most graphics systems are raster-based.

The **raster** is an array of picture elements — **pixels**.

The pixels are stored in the **frame buffer**.

The **depth** of the frame buffer = num. bits used per pixel.

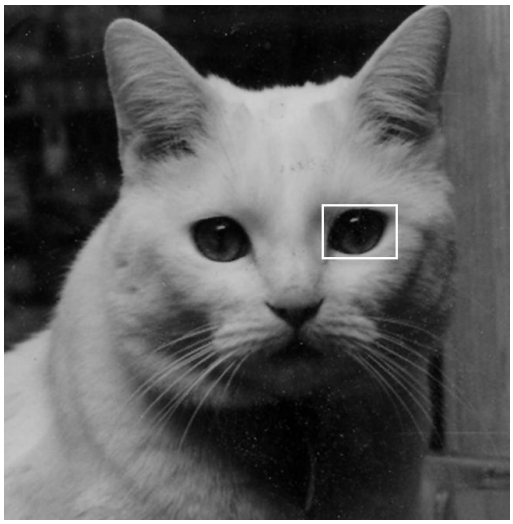
**1-bit-deep** ⇒ black and white only.

**8-bit-deep** ⇒ $2^8 = 256$ colours.

**24-bit-deep** ⇒ the RGB-colour system: red, green, blue, 256 shades for each.

The **resolution** of the frame buffer = number of pixels.

**Rasterization** or **scan conversion** is the assignment of values to pixels in the frame buffer that best represent graphical primitives: lines, circles, polygons, etc. Done by the **processor**. Sophisticated graphics systems use special-purpose processors for this.

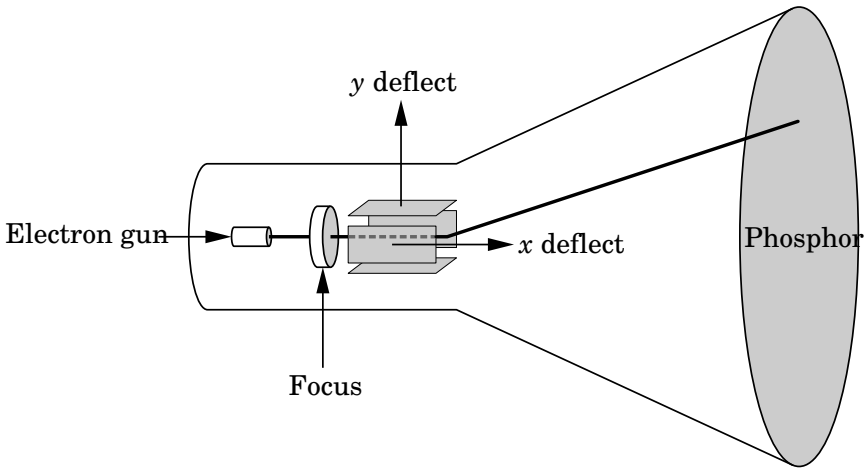


### Output devices

The usual display type is a **cathode-ray-tube** (CRT). Pixels in the frame buffer are displayed as points on the surface of the display. The points last only a few milliseconds, so the content of the frame buffer must be redisplayed or **refreshed**. The rate must be high enough to avoid flicker: the **refresh rate**.

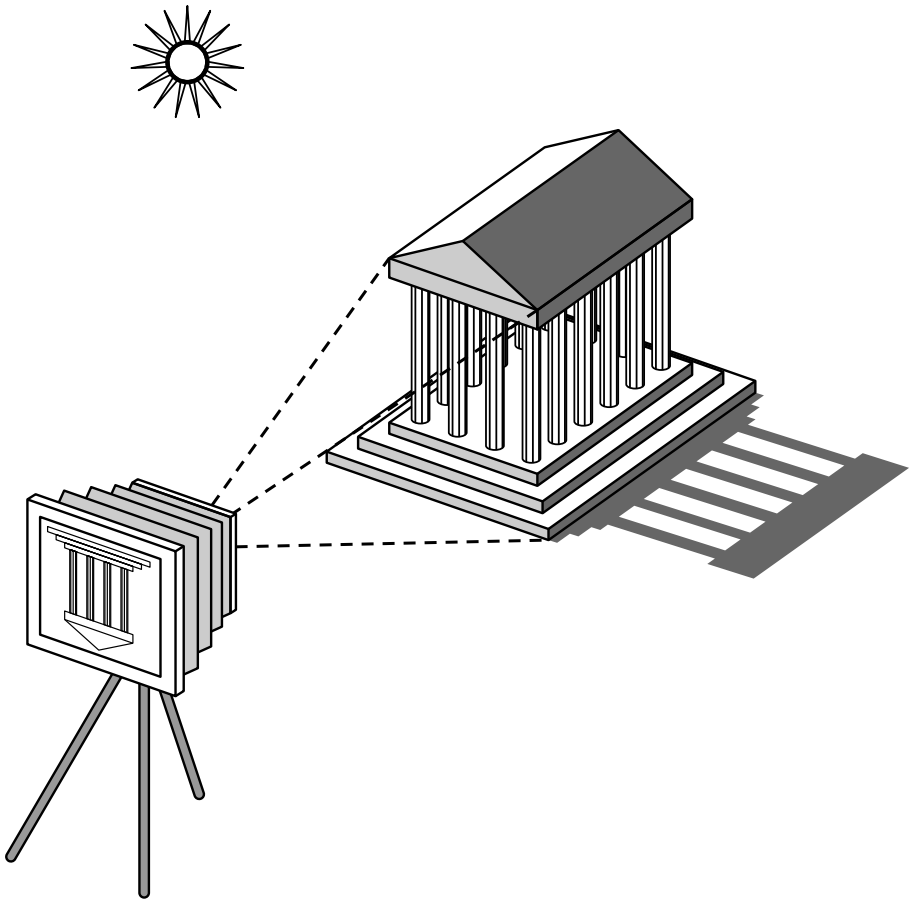
There are two ways of displaying pixels:

- **non-interlaced:** row by row, i.e. scan line by scan line, typically 50 to 80 Hz (times per second).
- **interlaced:** odd and even rows are refreshed alternately, used in television. For example, at 60 Hz, the screen is redrawn only 30 times per sec.



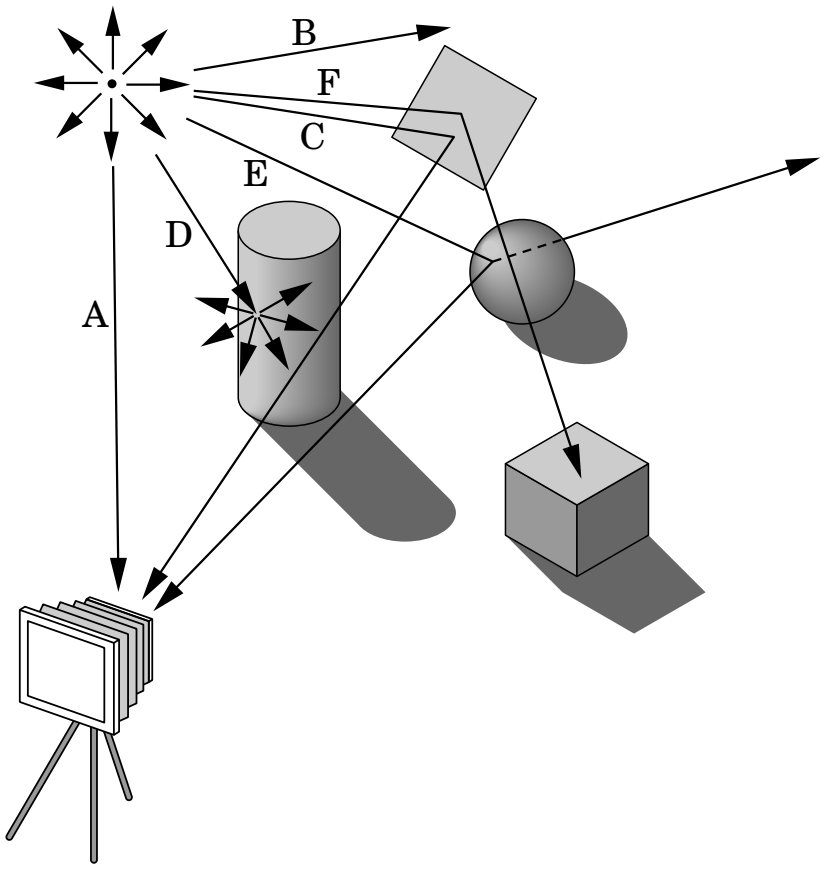
**What is an image?**

A combination of **objects**, **lights**, and a **viewer** (or **camera**).

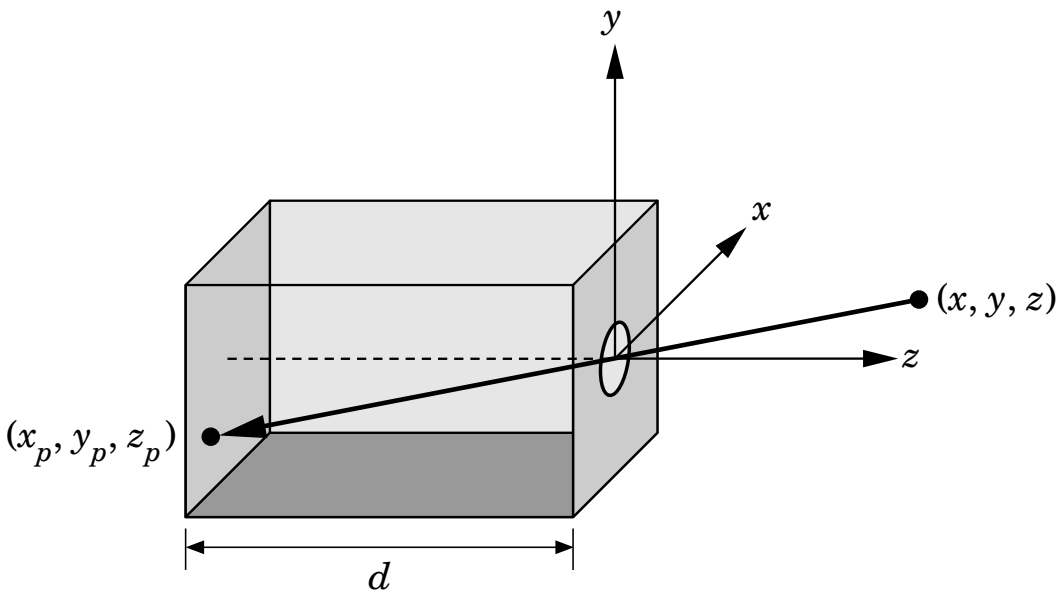


### Ray tracing

Trace rays from each light source and see whether they reach the viewer. Those rays that do form the image. We could do ray tracing in computer graphics but it is computationally extremely expensive. In practice we avoid ray-tracing by simplifying our models, for example by assuming all objects are equally bright (i.e. there are light sources everywhere). Then each object (e.g. triangle) looks like a light emitter. It is possible to add light sources and material properties and still avoid ray-tracing.



**The pinhole camera:**



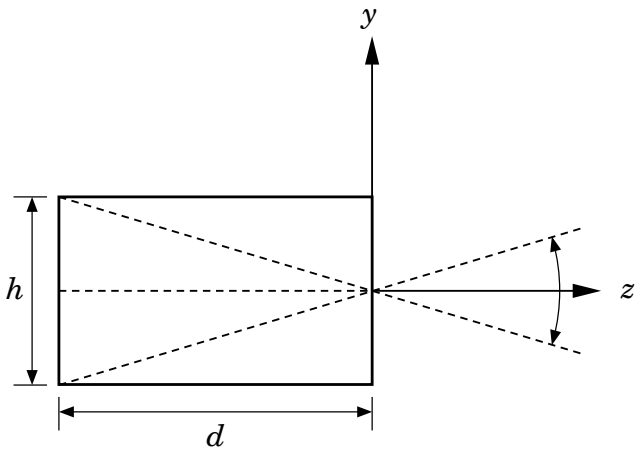

a box with a small hole in the centre of one side with the film placed at the other end. It is easy to model geometrically. Orient the camera along the $z$ axis with the pinhole at the origin. Suppose the camera has length $d$. Assuming the pinhole allows only one ray of light from any point $(x, y, z)$, the ray is clearly projected to the point $(x_p, y_p, -d)$, with coordinates

$$x_p = \frac{-dx}{z}, \qquad y_p = \frac{-dy}{z}.$$

The point $(x_p, y_p, -d)$ is called the **projection point**. In an idealized model, the colour on the film plane at this point is the colour of the point $(x, y, z)$.

The **field** or **angle of view** of the pinhole camera is the angle of the largest object that is fully visible on the film plane. If $h$ is the height of the camera and $\theta$ the angle of view then

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




The pinhole camera has an infinite **depth of field**: every point in the field of view is in focus.

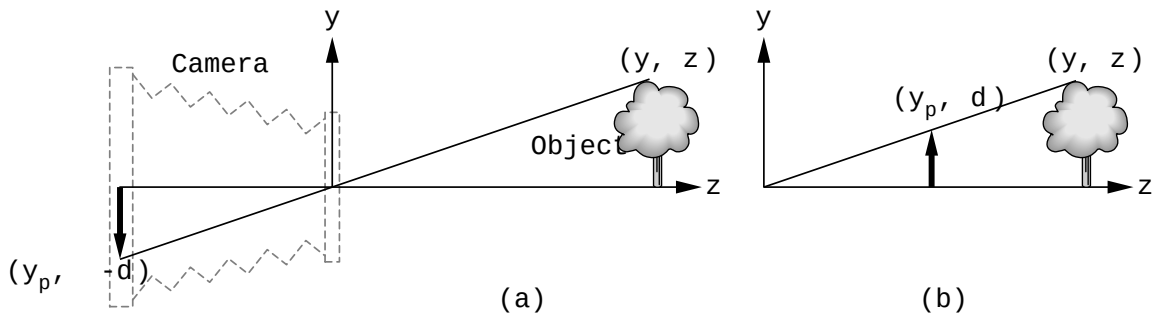
The pinhole camera has two disadvantages: (1) too little light gets in, and (2) the angle of view cannot be adjusted.

More sophisticated cameras replace the pinhole with a lens. The lens allows more light to enter and different lenses give different angles of view (c.f. a 'wide angle lens'). Lenses, however, do not have an infinite depth of field; not all distances are in focus.

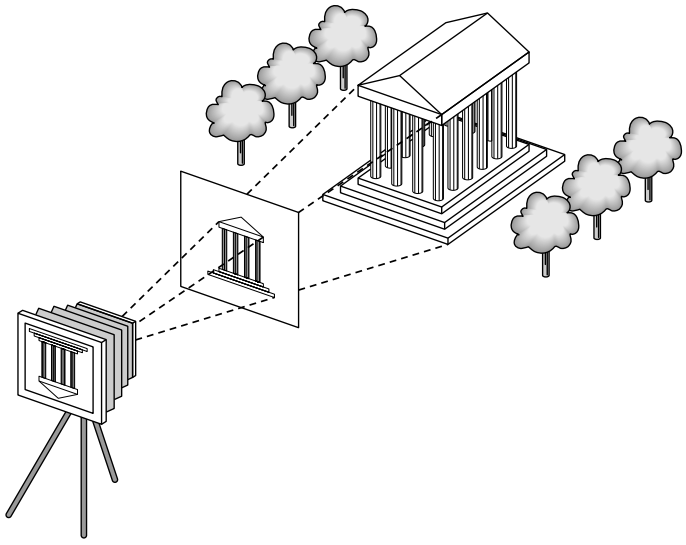
Like the pinhole camera, computer graphics produces images in which **all objects are in focus**.

### The synthetic camera model

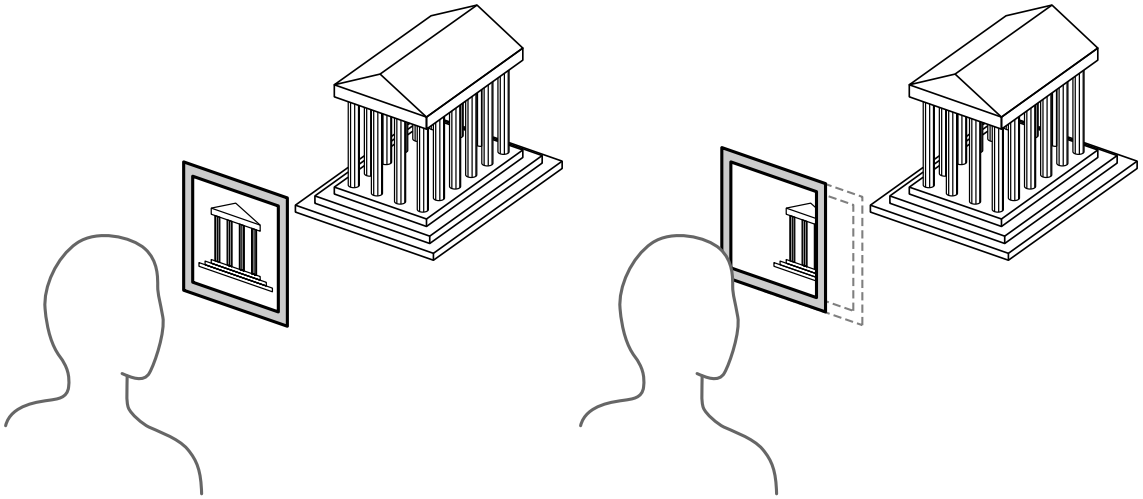
In computer graphics we use a **synthetic camera model** to mimic the behaviour of a real camera. The image in a pinhole camera is inverted. The film plane is behind the lens.




In the synthetic camera model we avoid the inversion by placing the film plane, called the **projection plane**, in front of the lens.

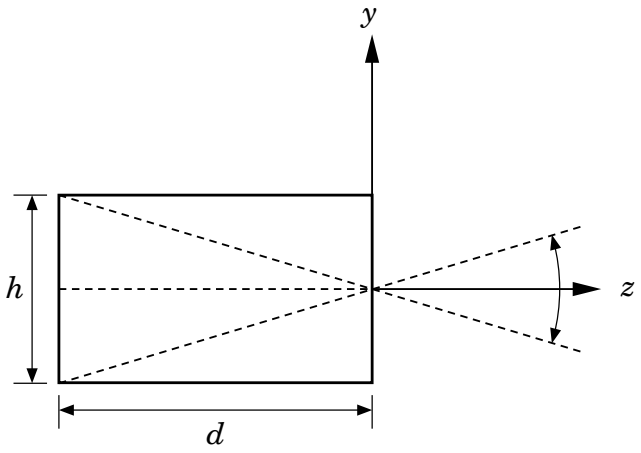


The **clipping rectangle** or **clipping window** determines the size of the image.



(a) (b)

This is similar to the effect of the angle of view in a pinhole camera.



**Application Programmer's Interface (API)**

We will use OpenGL. Other possibilities are: PHIGS, Direct3D, VRML, JAVA-3D.

We need to specify:

1. Objects
2. Viewer
3. Lights
4. Material properties

**Objects** are defined by a set of vertices, e.g. line segments, rectangles, polygons, etc. A circle can be represented by its centre and one point on the circle or by three points on the circle. OpenGL also allows the definition of more complex curve and surface types (by approximating by triangles). It also provides direct access to the frame buffer.

**The viewer** is determined by: position, orientation, focal length, film plane.

**Light sources** have location, strength, colour, directionality, etc.

**Material properties** are attributes of the objects.

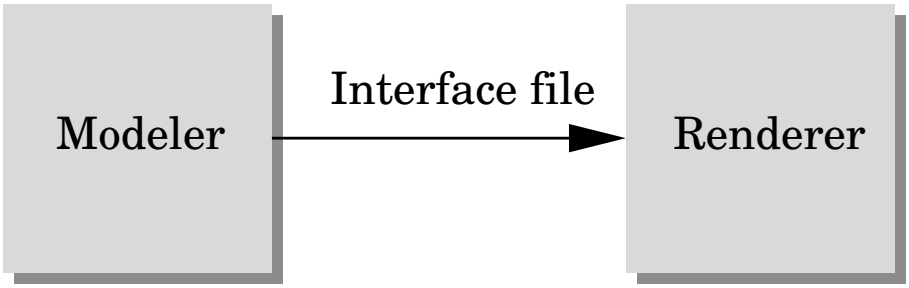
### The Modelling-Rendering Paradigm

It can be an advantage to model and render a scene separately, using tailored software and hardware for each task.

**Modelling** is very interactive but user does not need to see all the detail.

**Rendering** involves mainly number crunching and no user interaction.

Pixar's Renderman provides an interface (i.e. file format) between the two tasks.



### The Geometric Pipeline

The four major steps in the imaging process are:

1. Transformation
2. Clipping
3. Projection
4. Rasterization

These operations can be **pipelined**, i.e. applied in parallel, to speed up the imaging process. Several steps can be represented by $4 \times 4$ matrices. Such matrices can be multiplied together, or **concatenated**, a process which lends itself to pipeline architectures and parallelism.