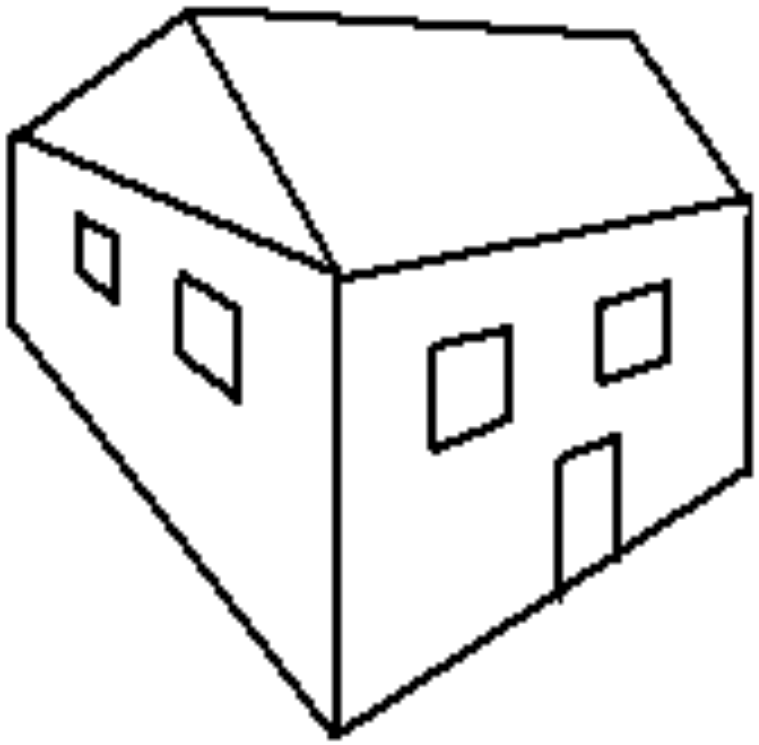
Computer Graphics

COMP H3016

Lecturer: Simon McLoughlin

Lecture 1



Course Overview

- Video signal generation
- Classic computer graphics algorithms
- Creating synthetic video in 2 and 3 dimensions
- Manipulation of graphics objects
- Surface rendering and projective transformations
- Analysis of graphics API (Open GL)

CA Overview

- 2 lab-exams -> 25% each (Dates to be announced)
- 1 Assignment (20%)
- Weekly lab work corrected in class (30%).

Reading

“3D Computer Graphics” – Alan Watt

“Computer Graphics, C version” Hearn and Baker

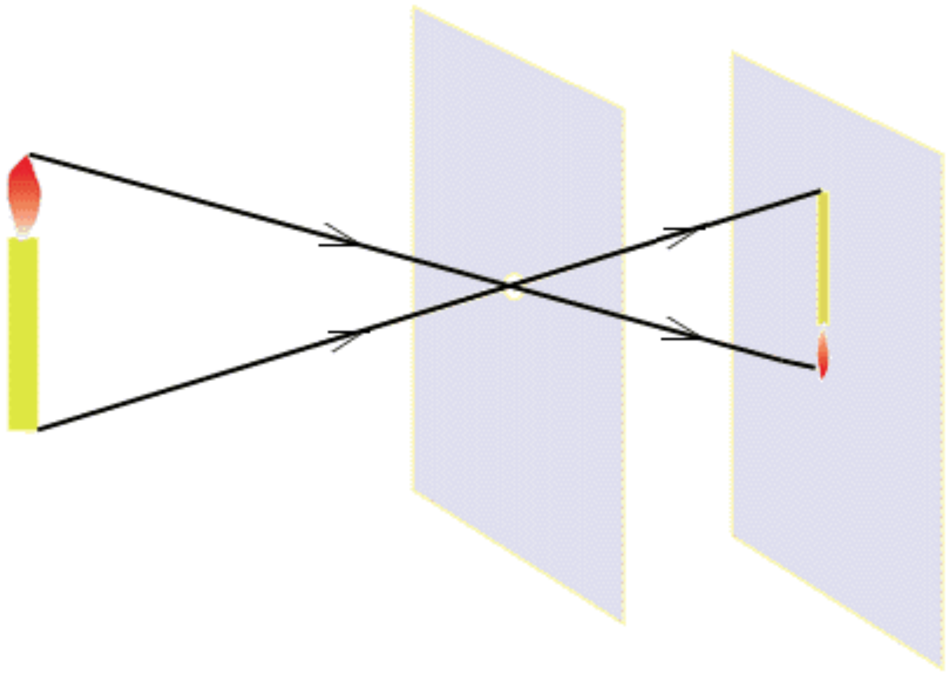
“Computer Graphics in openGL” Hearn and Baker

“Computer Graphics for Java Programmers”, Ammerall

“Computer Graphics – Mathematical first steps”, Egerton and Hall

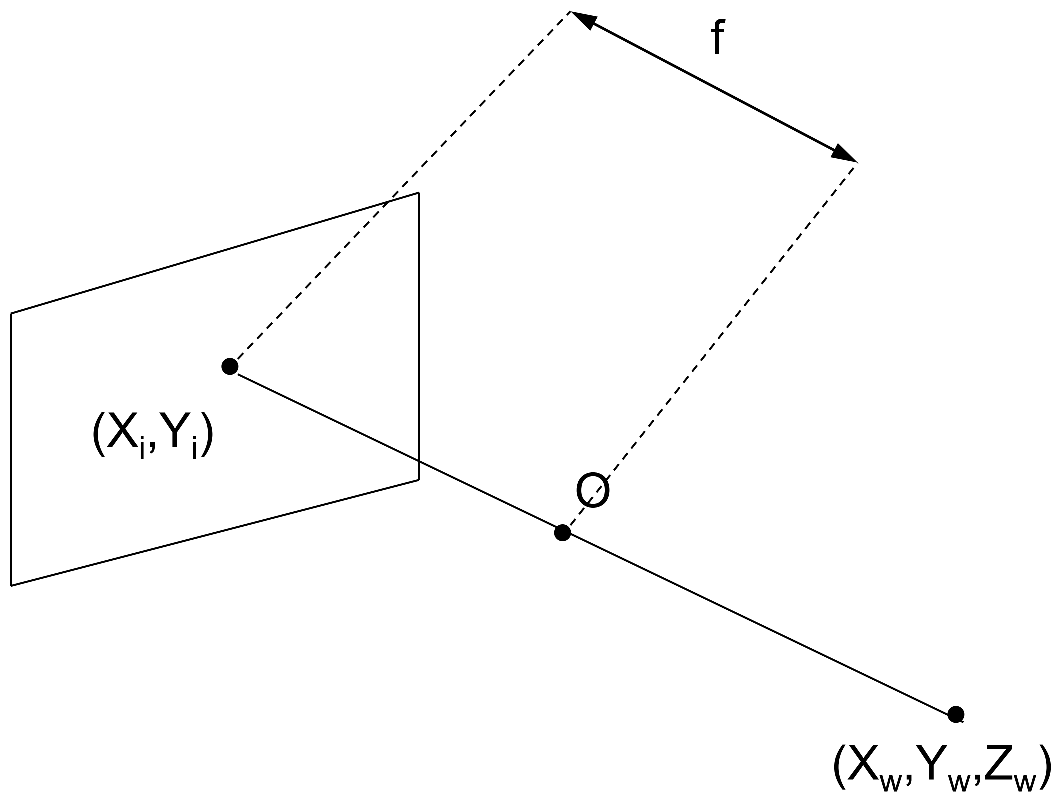
A picture!

The Pinhole Camera



- Simplest imaging model!

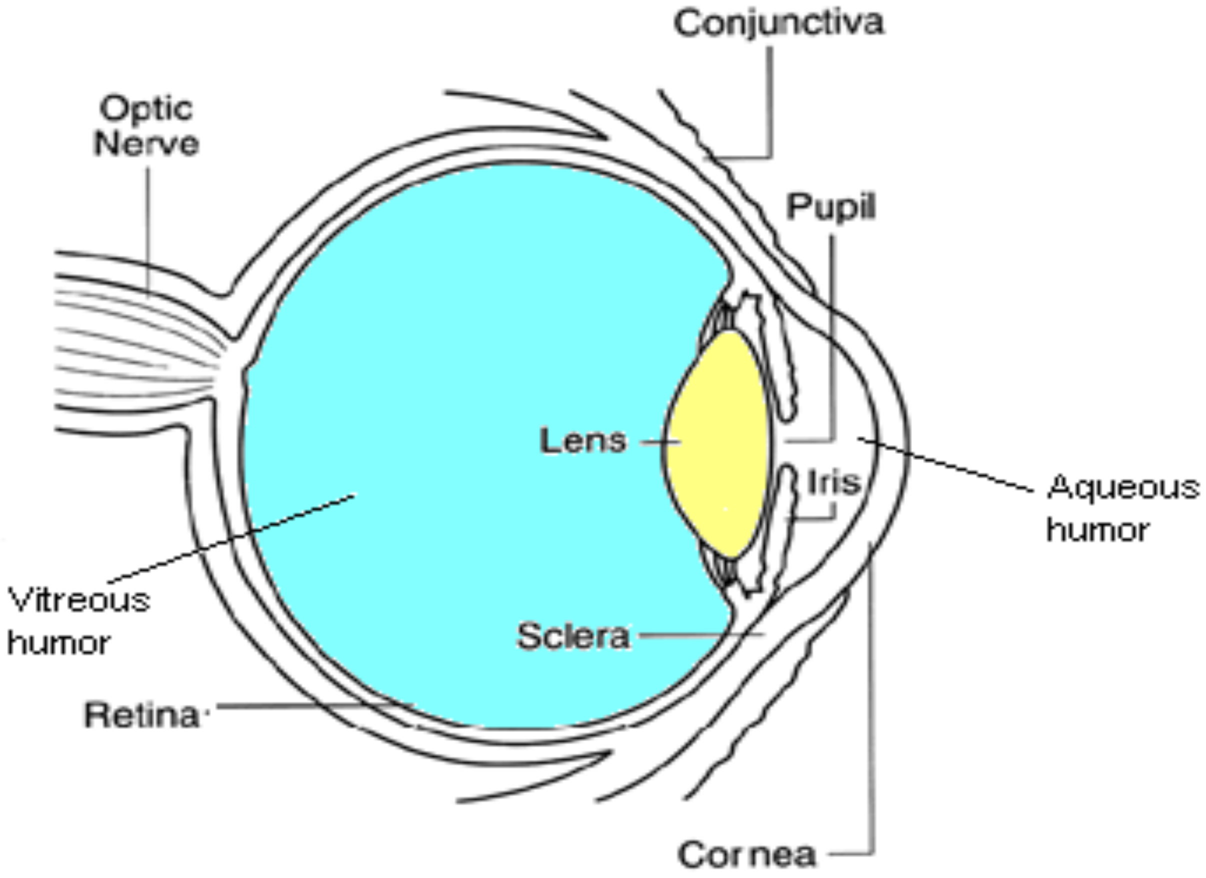
The Pinhole Camera - mathematically



$$X_i = fX_w/Z_w$$

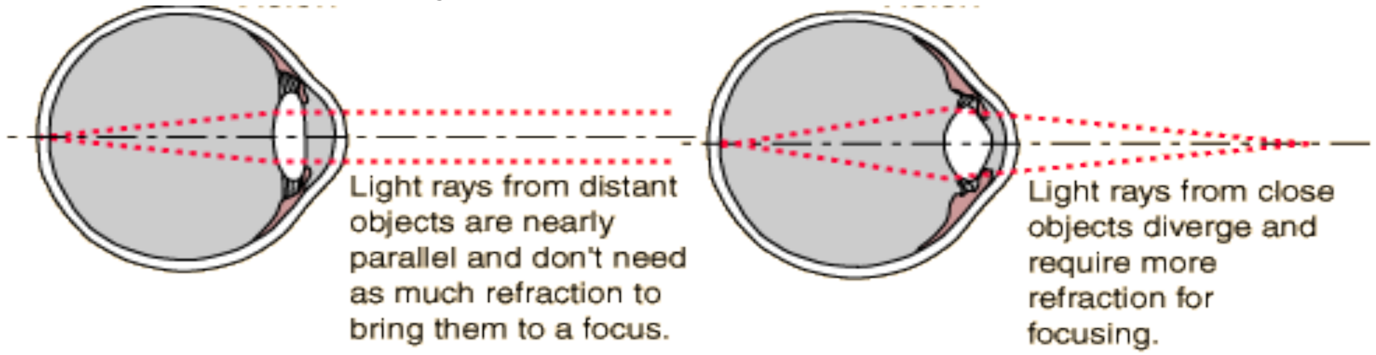
$$Y_i = fY_w/Z_w$$

The Human Visual System



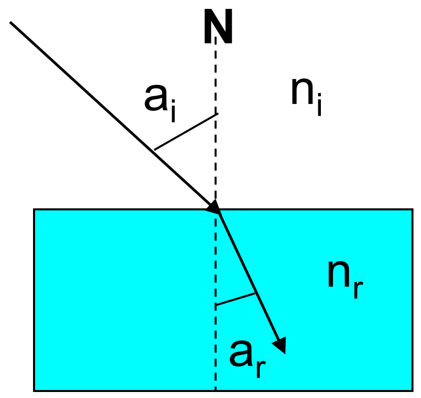
The Human Visual System

- The amount of light entering the eye is controlled by the **Pupil** in the **Iris**
- Light is focused by the cornea and lens onto the photo-sensitive layer -> **The Retina**
- This process is known as **refraction** (bending of light)
- **Refraction is the bending of a wave** when it enters a medium where it's speed is different. The refraction of light when it passes from a fast medium to a slow medium bends the light ray toward the normal to the boundary between the two media
- In the human visual system, light rays for close objects require a **thickened more rounded lens** for more refraction than for distant objects
- Process of refraction adjustment is called **accommodation**


Light rays from distant objects are nearly parallel and don't need as much refraction to bring them to a focus.
Light rays from close objects diverge and require more refraction for focusing.

Snell’s Law

- The refraction of light is governed by Snell’s law
- When light passes from a medium where its speed is faster (e.g. air) to a medium where its speed is slower (e.g. water/glass (lenses)) the **light bends towards the surface normal** at the intersection point
- When light passes from a medium where its speed is slower (e.g. water/glass) to a medium where its speed is faster (e.g. air) the **light bends away from the surface normal** at the intersection point



the refraction angle, a_r can be computed from Snell’s law:

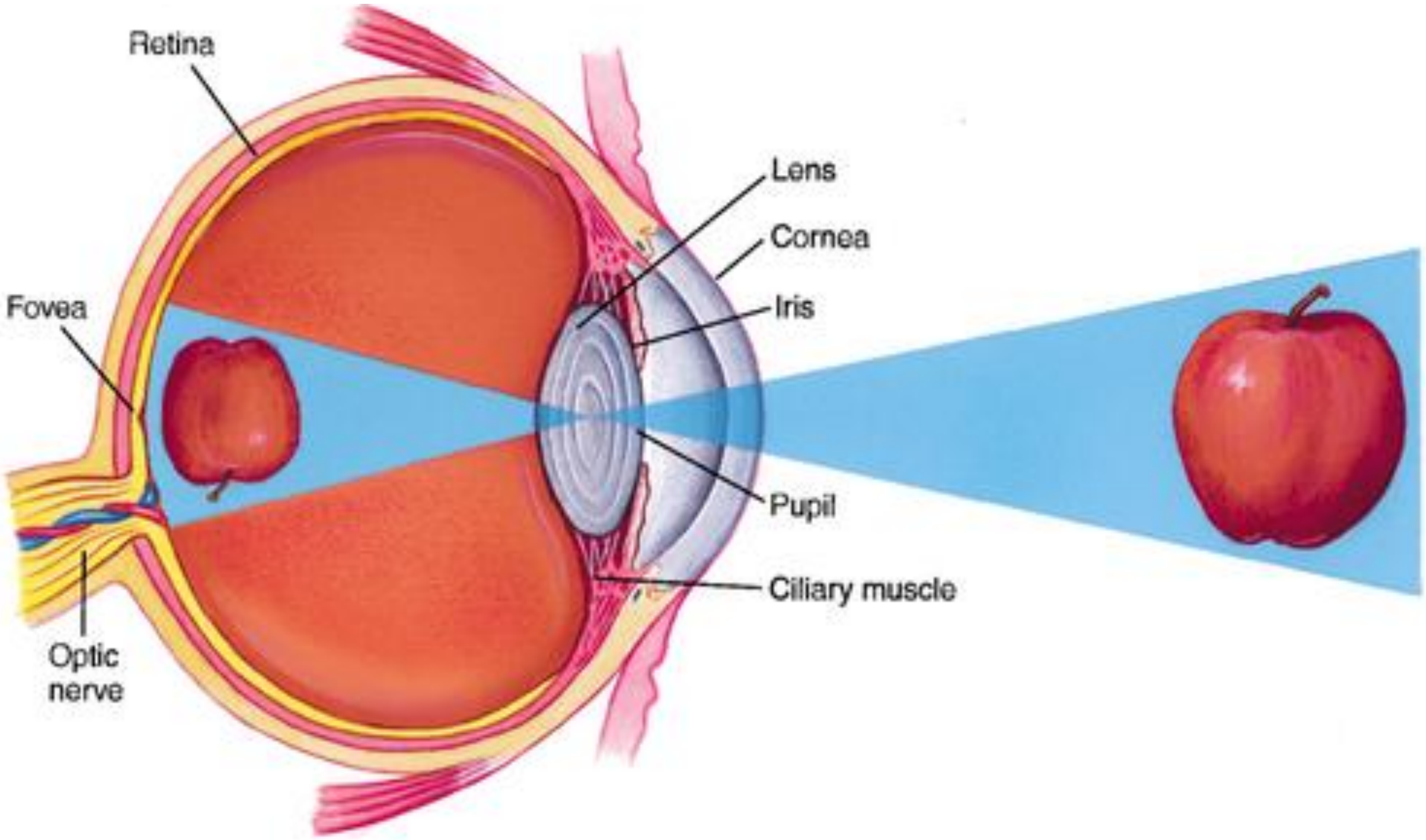
$$n_i \sin(a_i) = n_r \sin(a_r)$$

where a_r and a_i are the angles the incident and refracted rays make with the surface normals and n_i and n_r the **refractive indexes** of the mediums

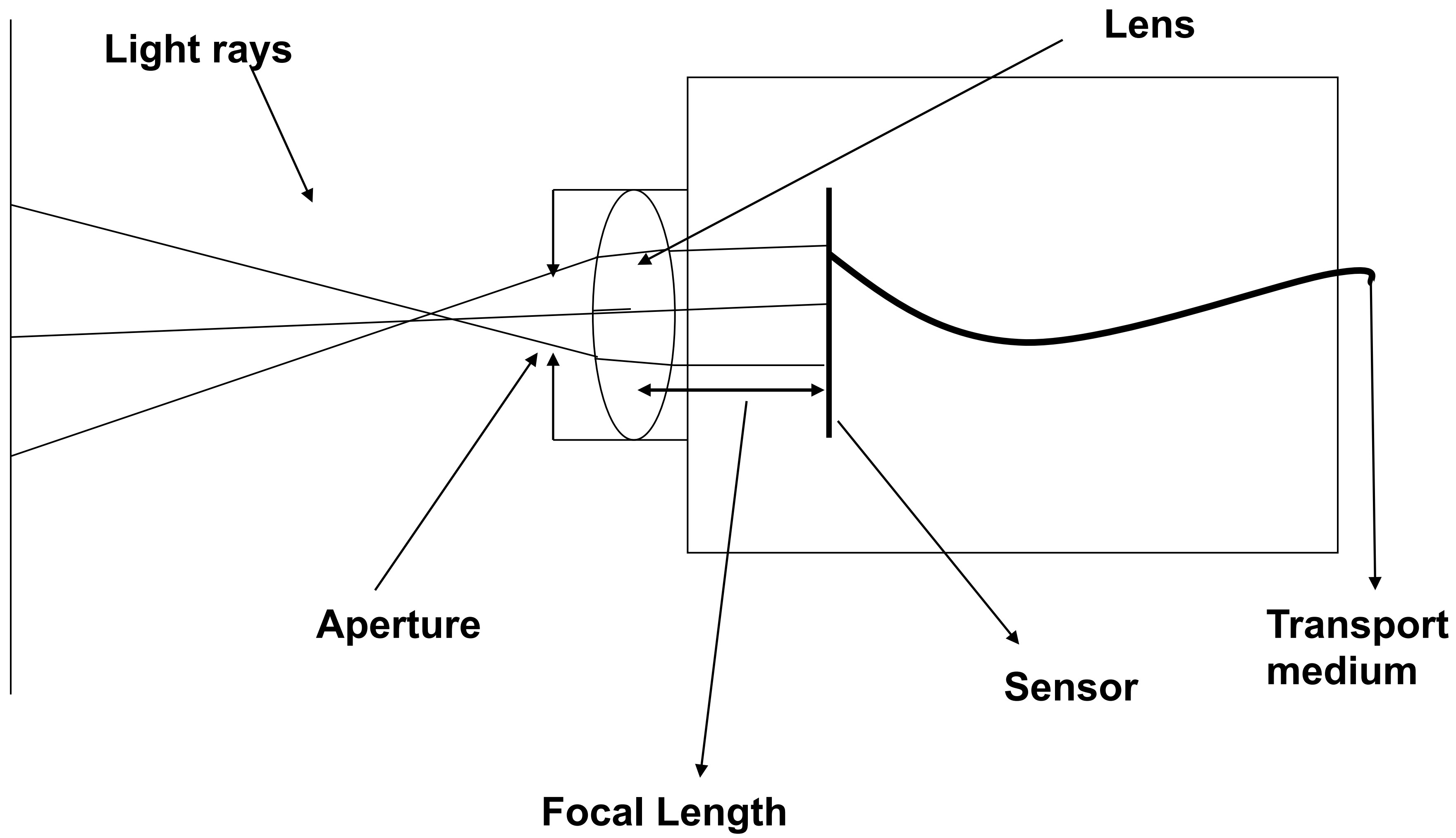
The Human Visual System

- The **retina** is populated with **photo-receptors** called **rods** and **cones**
- **Cones** dominate a central area of the **retina** know as the **fovea**
- **Cones** are densely populated around this region and provide for high resolution and color imaging
- **Rods** dominate the periphery of the **retina** and provide for vision at the peripheral field of view as well as in low lighting levels
- The signal leaves the **retina** to go to the **brain** through the **optic nerve**
- **Image formation** is primarily a physical process that **captures scene illumination through a lens system** and relates the measured light energy to an electrical signal that is perceived by the brain

The Human Visual System



The Video Camera

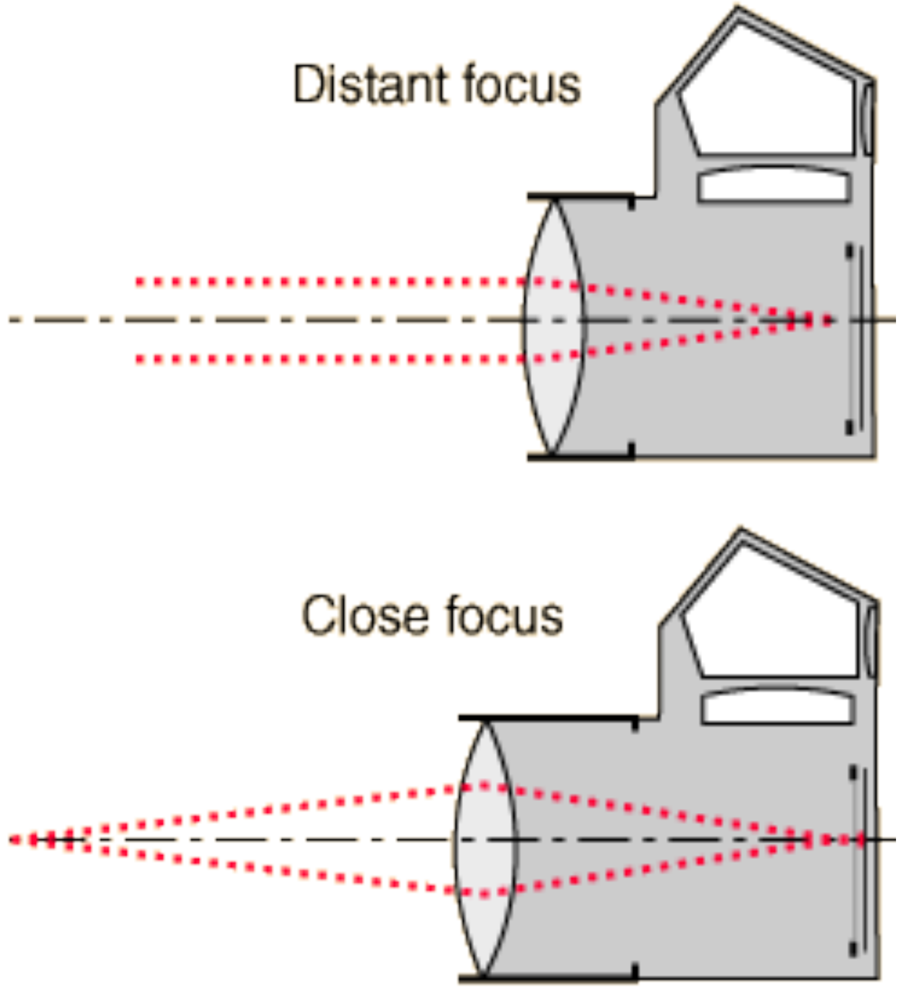


The Video Camera

- The amount of light entering the camera is controlled by the **Aperture**
- Light is focused by the lens onto the photo-sensitive layer -> **The image sensor**
- This process is known as **refraction** (bending of light)
- **Refraction** is the bending of a wave when it enters a medium where it's speed is different. The refraction of light when it passes from a fast medium ……………………………

- Light rays for close objects requires movement of the lens away from the sensor (**increasing the focal length**)
- Process of focal length adjustment is called **accommodation**

The Video Camera

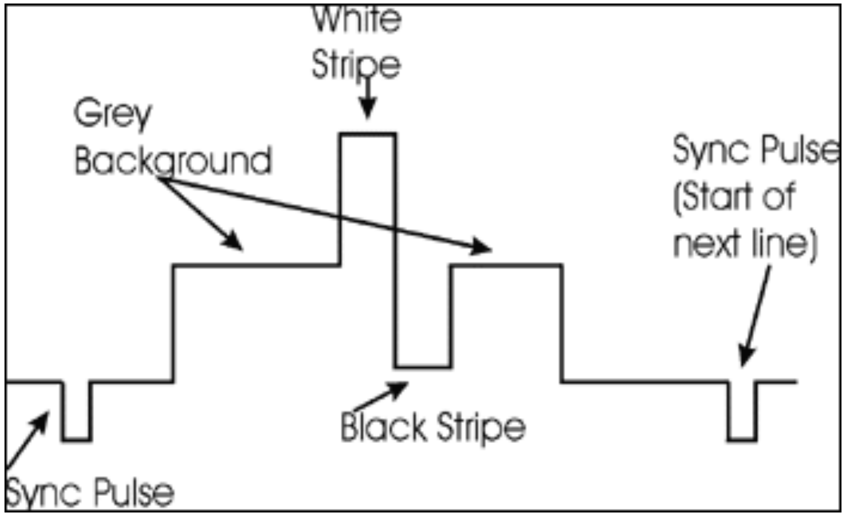


Analog Video Signals

- We know now how images or **video signals are formed** and their relationship to those **perceived by humans**
- But there is **more** to understanding video signals than meets the eye!
- Currently we have only seen how the **scene is projected onto a sensor** and how this **projection is represented as an electrical waveform**
- But there is still a lot of questions that need answering:
  - How can a one-dimensional waveform represent a two-dimensional image?
  - What about color?
  - What about motion?
  - Can we digitize the signal?
- Now we will answer these questions

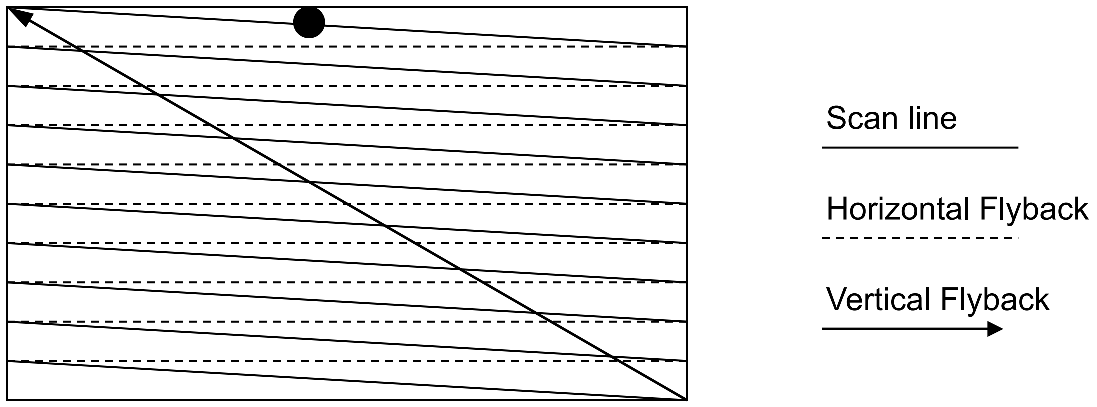
Analog Video Signals

- The key elements to representing a 2-d signal with a 1-d waveform are **synchronizing** and **scanning**
- First we will look at scanning then we will look at synchronizing
- CRT's display the video signal **line by line** (which is also the way it is read from an interline CCD sensor) and **frame by frame** - this is referred to as **"scanning"**
- Therefore we must send additional timing information along with the light intensity signal which tells the display device that a line has finished or a frame has finished - this is referred to as **"synchronizing"**



Scanning

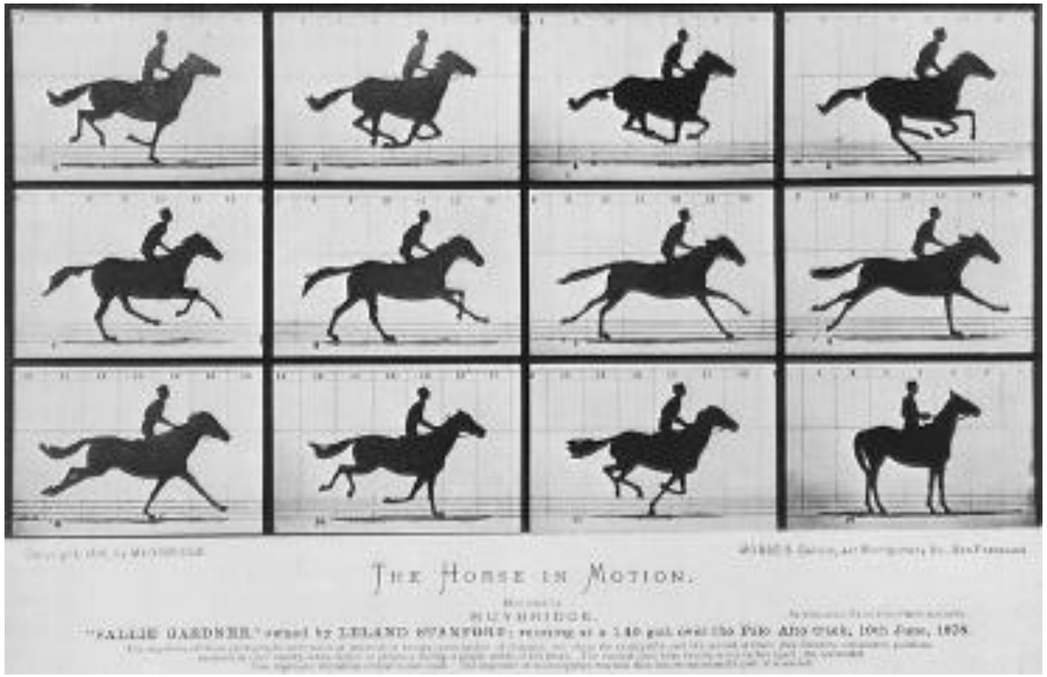
- To transmit 2-d images as a 1-d signal each frame is split up into a sequence of horizontal strips or **lines**
- Each line is transmitted separately to the display device where they are painted or **“scanned”** in sequence onto the viewing screen



- **Progressive Scan -** Signal is scanned line by line onto viewing screen. Timing (synchronization) information tells electron beam when to “flyback” to scan another line or frame

Motion Pictures

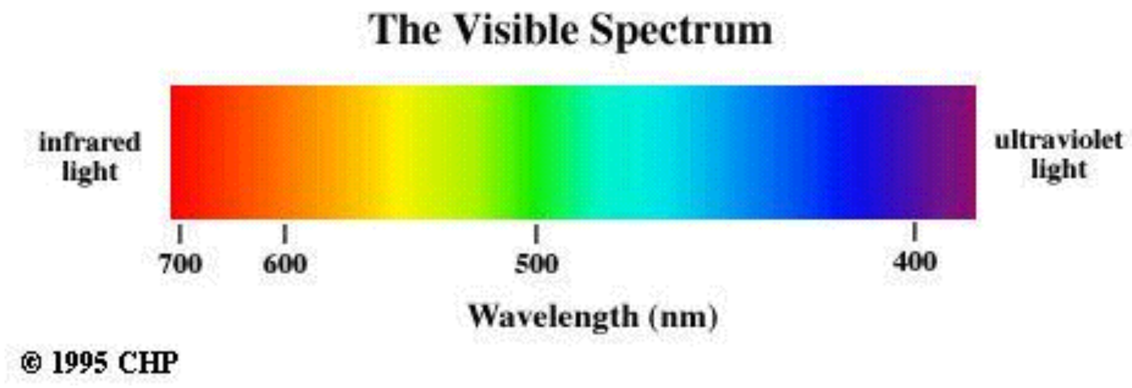
- TV and video pictures are made up of a number of images (**frames**) sequentially displayed on the screen to give the illusion of movement (PAL 25 frames/sec)


THE HORSE IN MOTION.
MUYBRIDGE.
"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

- Eadweard Muybridge - motion picture pioneer (1830-1904)
- Used multiple cameras triggered by tripwires to capture multiple pictures of a horse in motion
- Why? - to settle a wager!

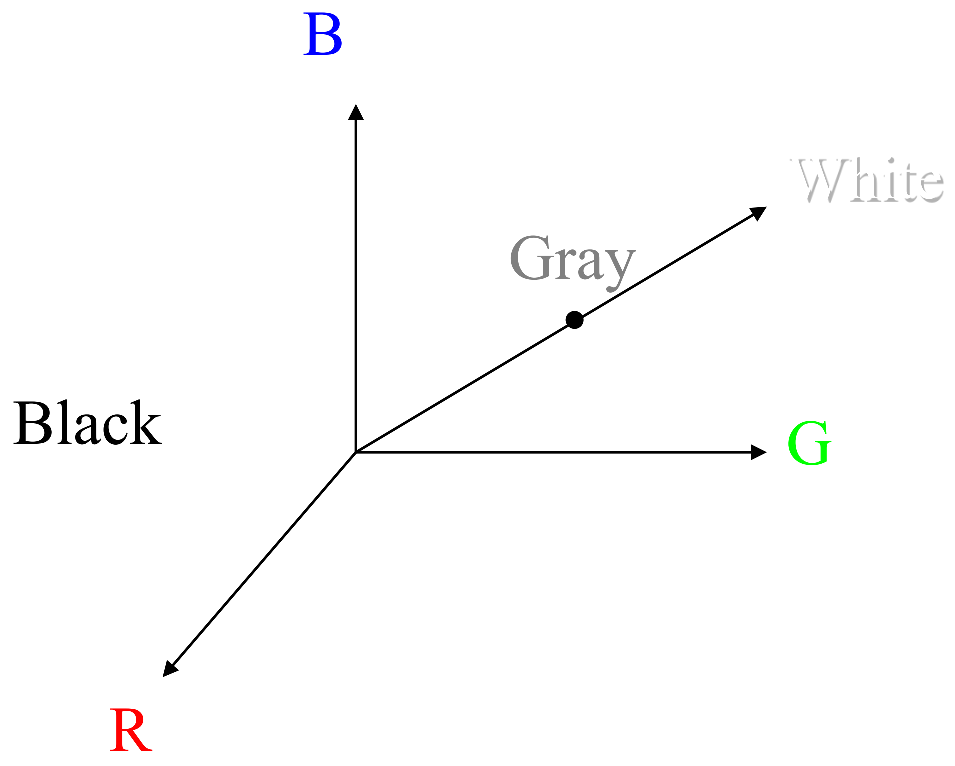
Adding Color

- Colors in a display device are created by the **additive mixing** of the three primary colours – Red(700 nm), Green(546.1 nm) and Blue(435.8 nm)



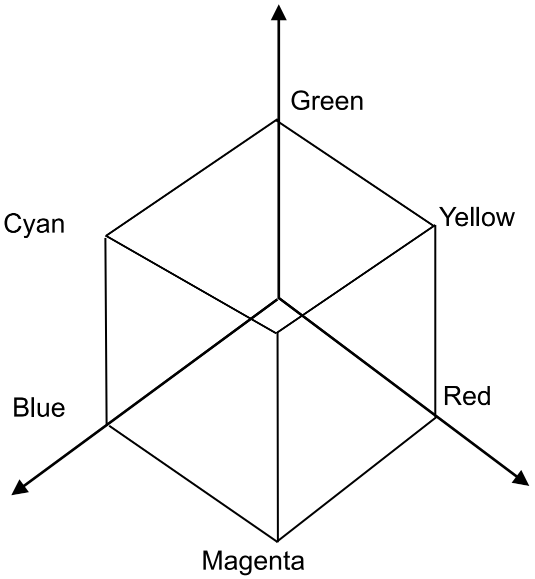
- **Any color** in the visible spectrum can be made from the mixing of these three primaries
- A **colorimeter** can be used to find the required intensity of each primary to create any color in the visible spectrum

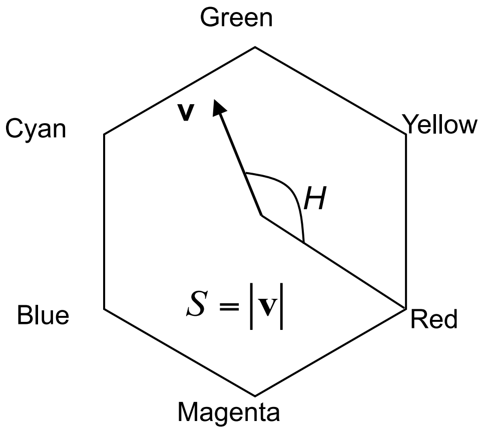
RGB color space

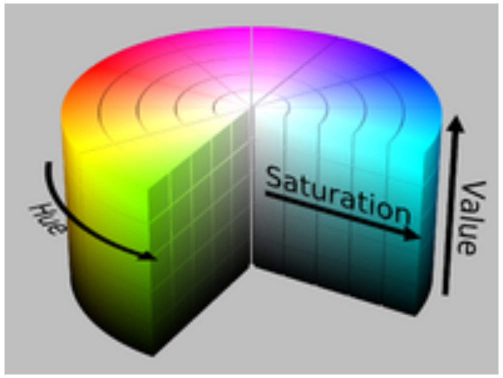


- The monchromatic space or gray-scale space can be created by **varying the rgb intensities equally**
- A color or RGB camera and display device can be directly linked together!

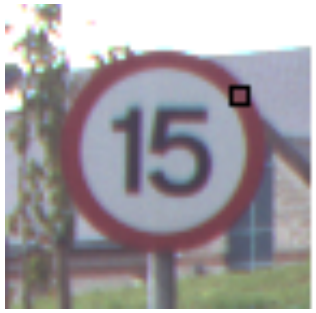
HSI color space







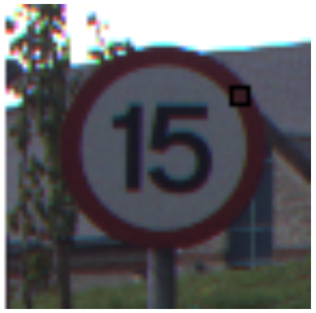
HSI space (Hue, Saturation Intensity) is another colour space that is easier to deal with for certain applications e.g. computer vision. In HSI space, the variations in ambient illumination are reflected in the intensity component



```
R=126.2
G= 73.8
B= 87.9

H=343.3°
S=0.42
I=0.49
```

Bright

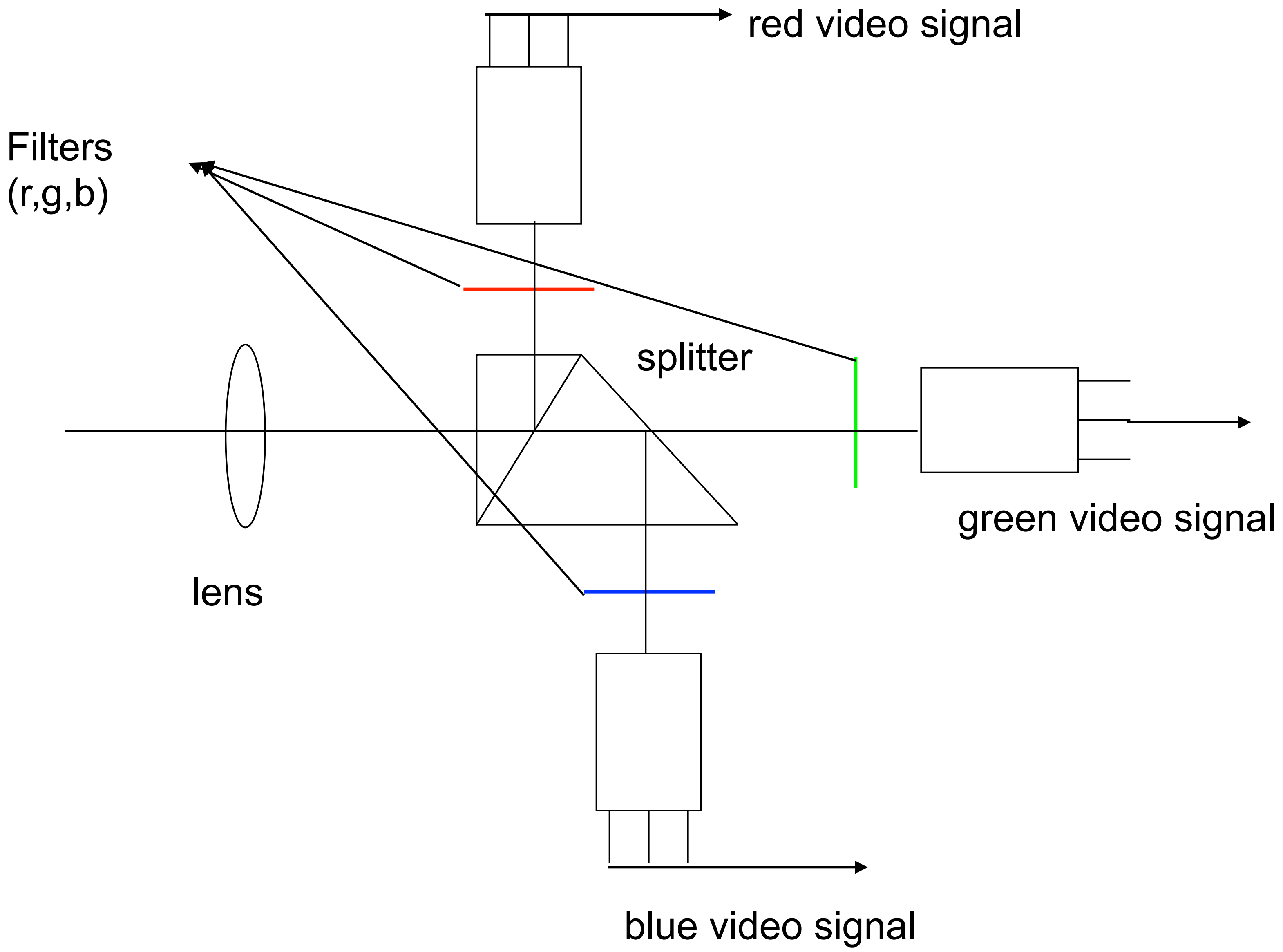


```
R=56.0
G=28.7
B=36.9

H=342.2°
S=0.49
I=0.22
```

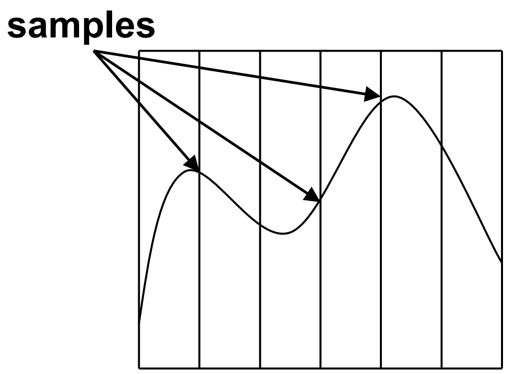
Dark

Simple Color Camera



Digital Video - Sampling

- Digital video is simply an **alternative means of carrying a video waveform**
- The analog signal is measured at regular intervals called **“samples”**

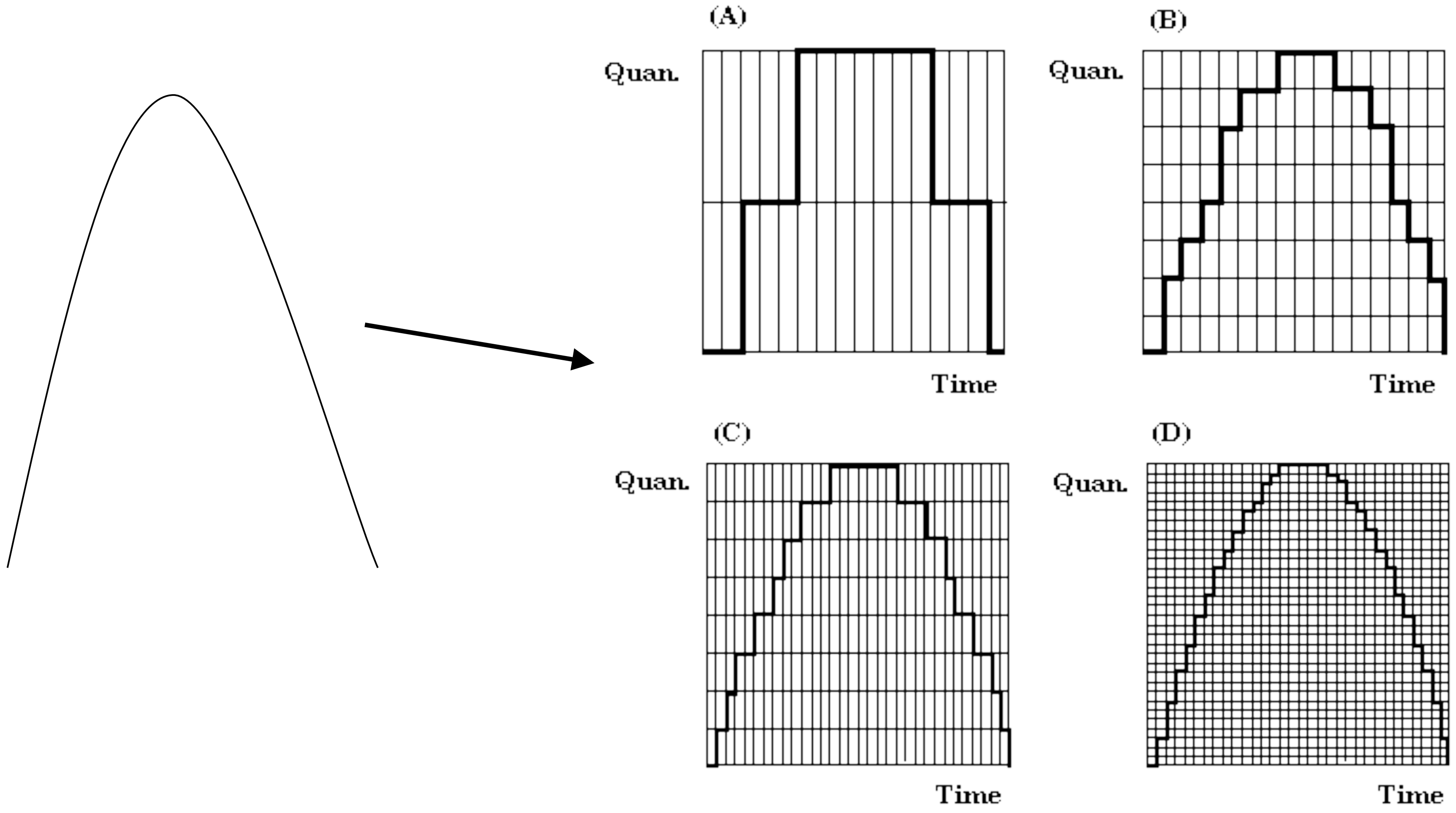


- If enough of these samples are taken the video signal can be **unambiguously reconstructed from the samples**
- The frequency at which samples are taken is called the **sampling rate**
- A lower bound on the sampling rate was shown by Nyquist-Shannon to be **twice the highest frequency in the analog signal** – TV?

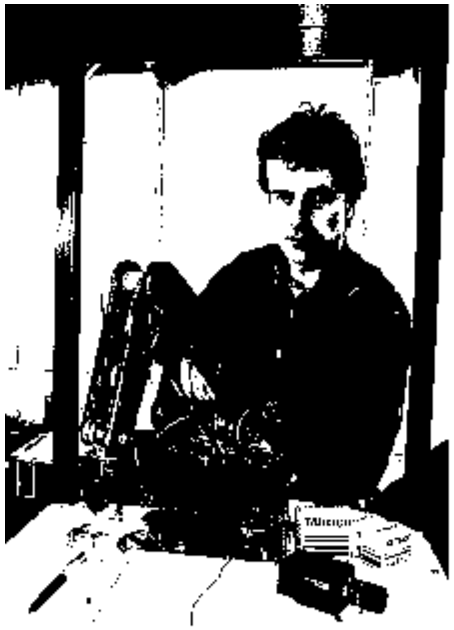
Digital Video - Quantization

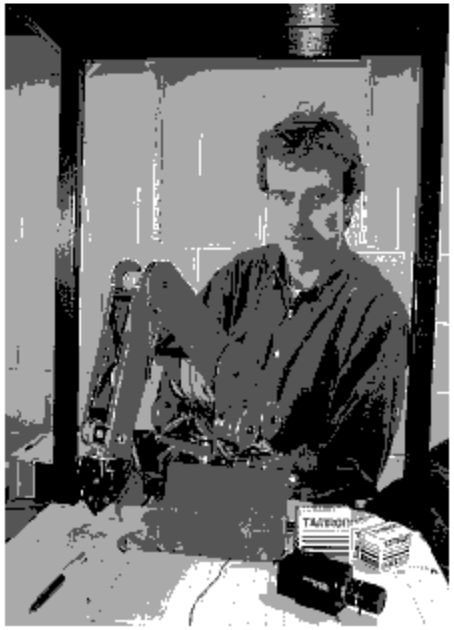
- After sampling the analog signal we have a **set of numbers or digits** (the samples) which constitute the video – hence the name digital video
- Although the signal is now represented by a set of samples, we must define **upper and lower bounds on the samples and only allow a sample to represent a finite number** between these upper and lower bounds
- This process is known as **“quantization”**
- When quantizing an image, we do not change the number of samples but do **limit the range of values a sample can take on** therefore the lower the level of quantization the lower the amount of memory/channel bandwidth we need to store/transmit the signal
- Monochrome digital video frames (commonly referred to as images) usually have **256 different values** (0-255) a sample can take on. Black=0 White=255
- This way we only need **1 byte (8 bits) of storage** to hold a (uncompressed) sample

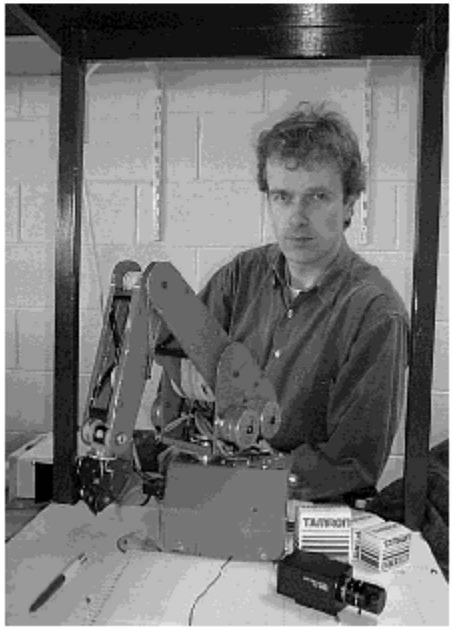
Digital Video - Quantization

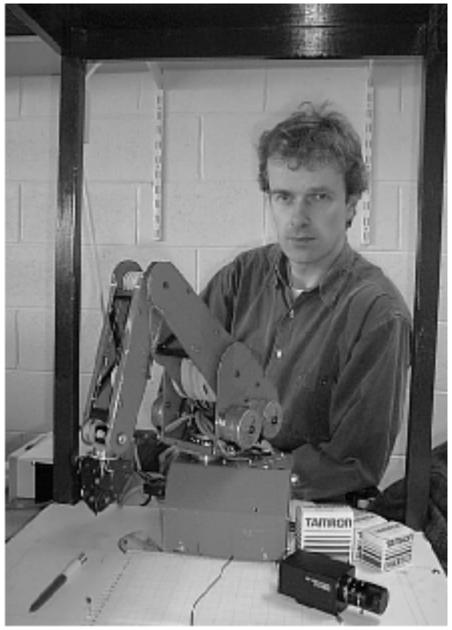


Digital Video - Quantization


1 bit, 2 levels


2 bits, 4 levels

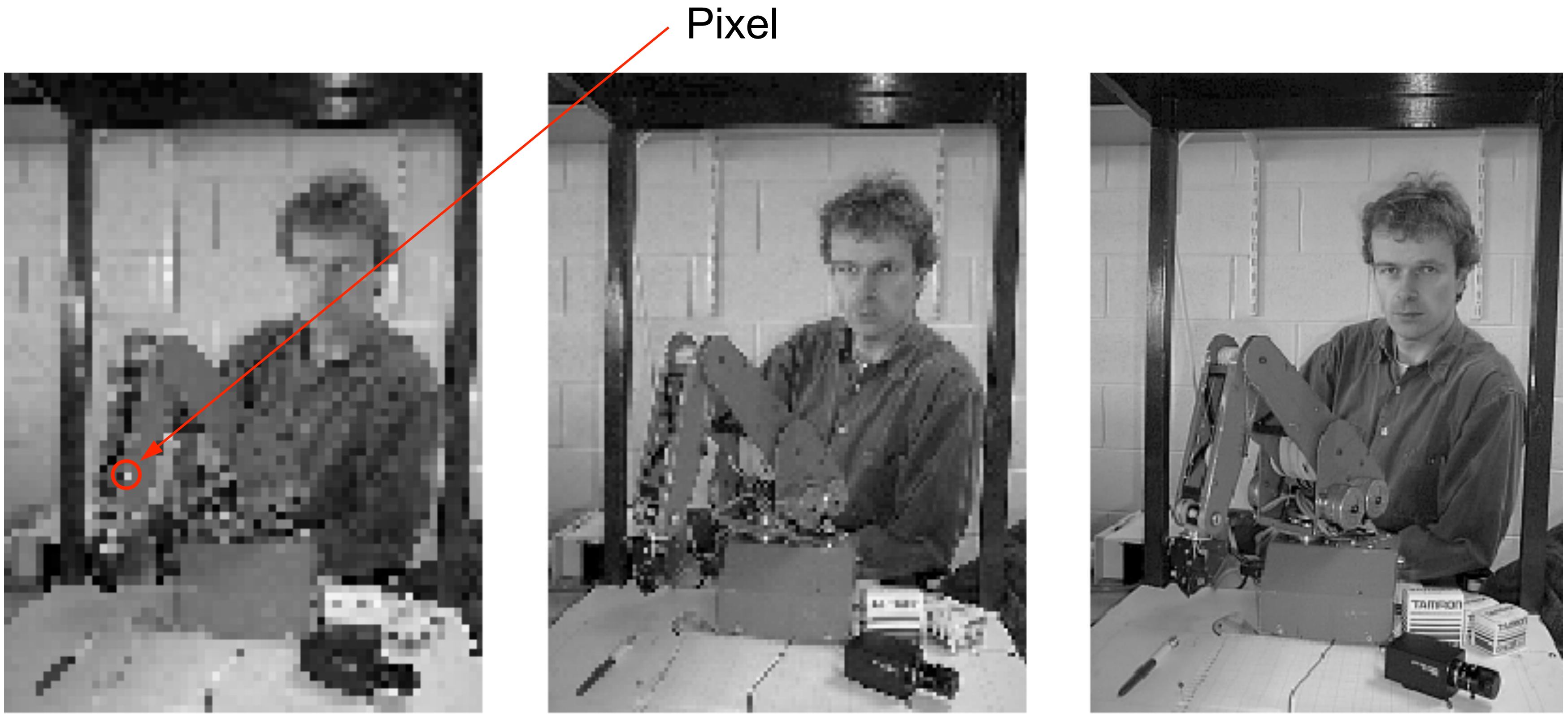

4 bits, 16 levels


8 bits, 256 levels

Digital Video - Resolution

- It should be obvious what the numbers that make up the image actually mean
- They are **samples of the original light intensity waveform** so therefore each one represents a discrete light intensity value
- If the sampling rate is a **whole multiple of the line rate** of the video signal then the samples will form a **rectangular array**
- Each element of the array is called a picture element or **“pixel”** and is a sample of the scene intensity at some position (x,y) at some time t
- When these array elements are placed close enough together the observer **perceives a continuous image**
- The number of pixels in 2-d array is known as the **image resolution**

Digital Video - Resolution



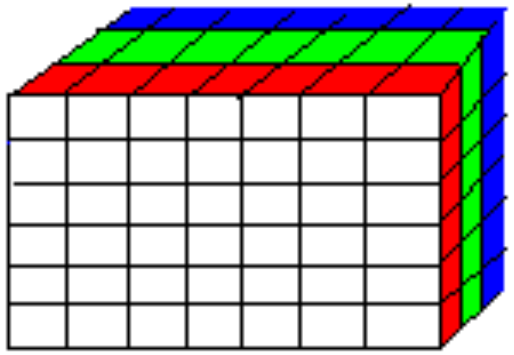
80 x 60

160 x 120

320 x 240

Digital Video - Color

- The notion of color in digital video or images is quite straightforward
- Instead of each pixel in an image being a scalar representing light intensity of some scene point it is a 3D vector containing the amount of red, green and blue light intensities
- We can think of color images as being 3-dimensional arrays



- So a true color RGB image requires 24 bits per pixel
- The color resolution of a 24 bit color image is 16,777,216

Digital Video - Color

R=139, G=85, B=43

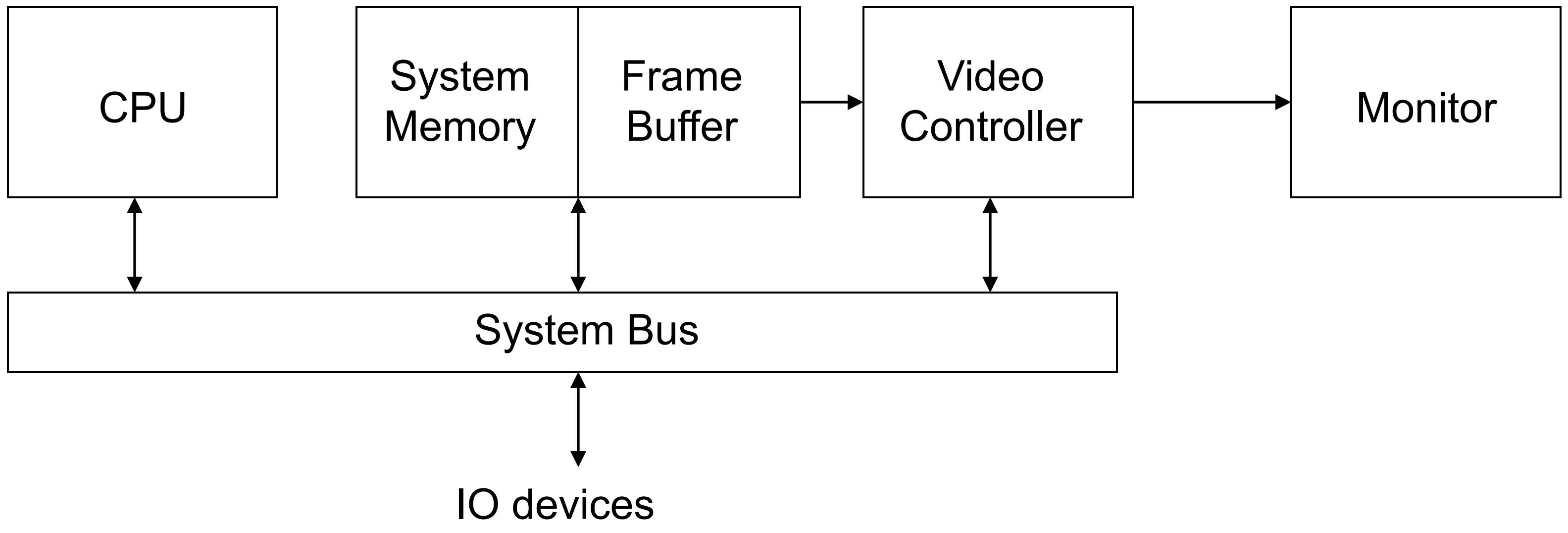
Digital Video – Computer Graphics

- The video signal is represented by a **series of numbers** (0,0,120, 255,10, 255……..)
- Converting from decimal to binary means we can now represent our signal by a **series of bits** which can be processed by a computer (image processing) (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,1,0,1,0,1, 1,1,1,1,1,1,1………..)
- If a real video signal is sampled from meaningful light energy in the real world and **converted into machine code** then can the opposite be done
- That is can a piece of machine code that has a meaningful light energy interpretation be created by a computer
- The answer is yes and the process of doing so is what's termed today as **"computer graphics"**
- Many routines and procedures exist for this purpose and we will look at "some" of the more important ones in this course

Digital Video – Computer Graphics

- **“Scan conversion”** lies at the heart of computer graphics
- This is the procedure of taking a **description of the scene** (mathematical or otherwise) and converting it into a **set of pixel intensity values** that can be read by the display device
- Pixel intensity values are **loaded into a part of memory called the frame buffer or refresh buffer**
- These pixel values can then be retrieved from the frame buffer and painted or scanned onto the screen
- The video controller is a **special purpose processor used for controlling the refresh operation** between frame buffer and monitor
- Consider the following diagram:

Computer Graphics – A Simple Graphics Architecture



Architecture of a raster graphics system with a fixed portion of system memory reserved for the frame buffer

Computer Graphics - Not another processor!

- Images typically **contain a lot of data (640*480*24)**
- Therefore **storage of images requires a lot of memory and processing of images requires a lot of processing power**
- To **relieve the CPU** of some of the chores associated with computer graphics another processor called the **graphics processor (GPU) is integrated into the graphics architecture**
- The GPU **provides the computational power to do the scan conversion mentioned previously**
- It is a piece of **hardware optimsed to perform common graphics functions such as generating lines or curves or performing transformations and manipulations on displayed objects**
- **Application programming interfaces (API's)** provide a framework to allow the programmer access to the hardware of the graphics architecture in a manner that is relatively uncomplicated and abstract
- OpenGL is an example of a computer Graphics API

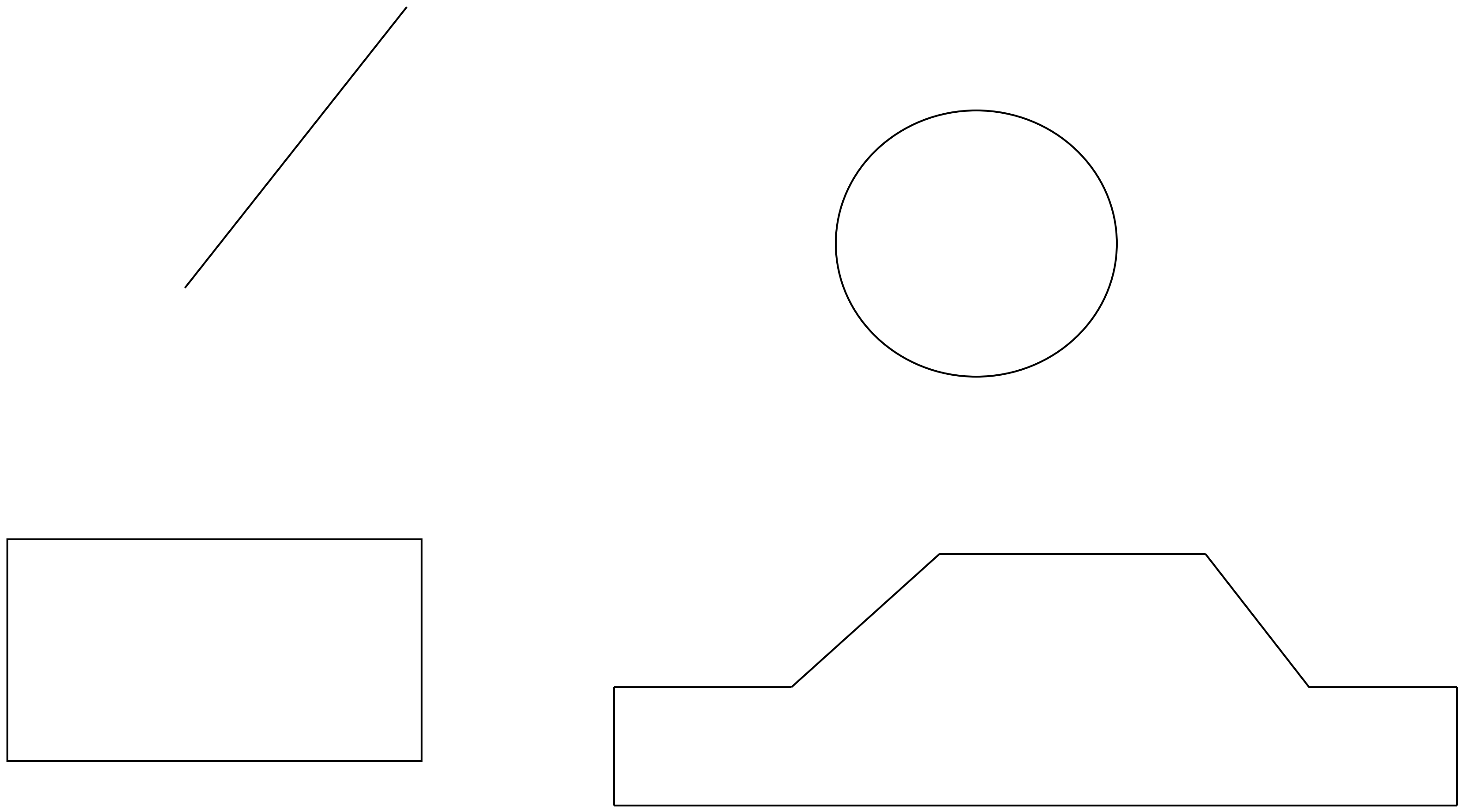
Graphics API functions

- Graphics API's provide a **host of functions** for creating/manipulating images that are made available to the programmer
- These functions can be categorised according to whether they deal with **output, input, attributes, transformations, viewing or general control**
- **Output primitives** are the basic building blocks for pictures. They consist of **character strings and geometric entities such as points, lines, circles, polygons.** Routines for generating these primitives are the basic tools for graphic construction
- **Attributes are the properties of the output primitives**. They include intensity and color specification, line styles, area filling patterns, textures, text styles etc.
- **Geometric transformations** are used to change the **size, position or orientation** of an object within a scene
- **Modeling transformations** are used to construct a scene from model coordinates
- **Viewing transformations** are used to generate a **particular view of an object in the scene**

Graphics API functions

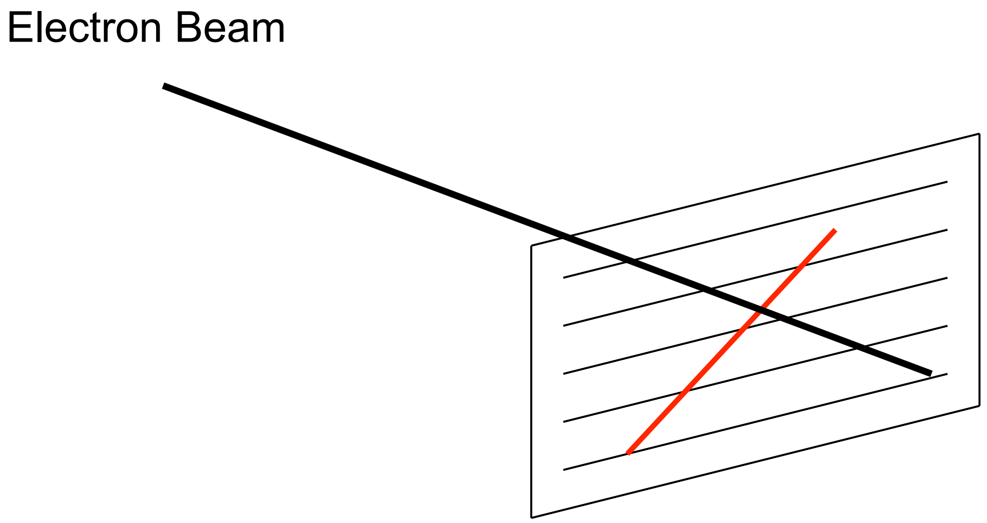
- **Interactive graphics applications use input devices** such as the mouse or joystick. **Input functions exist to control and process the data flow from these devices**
- A graphics package also contains **control operations** to do the general housekeeping tasks like clearing a screen

Output Primitives



Output primitives

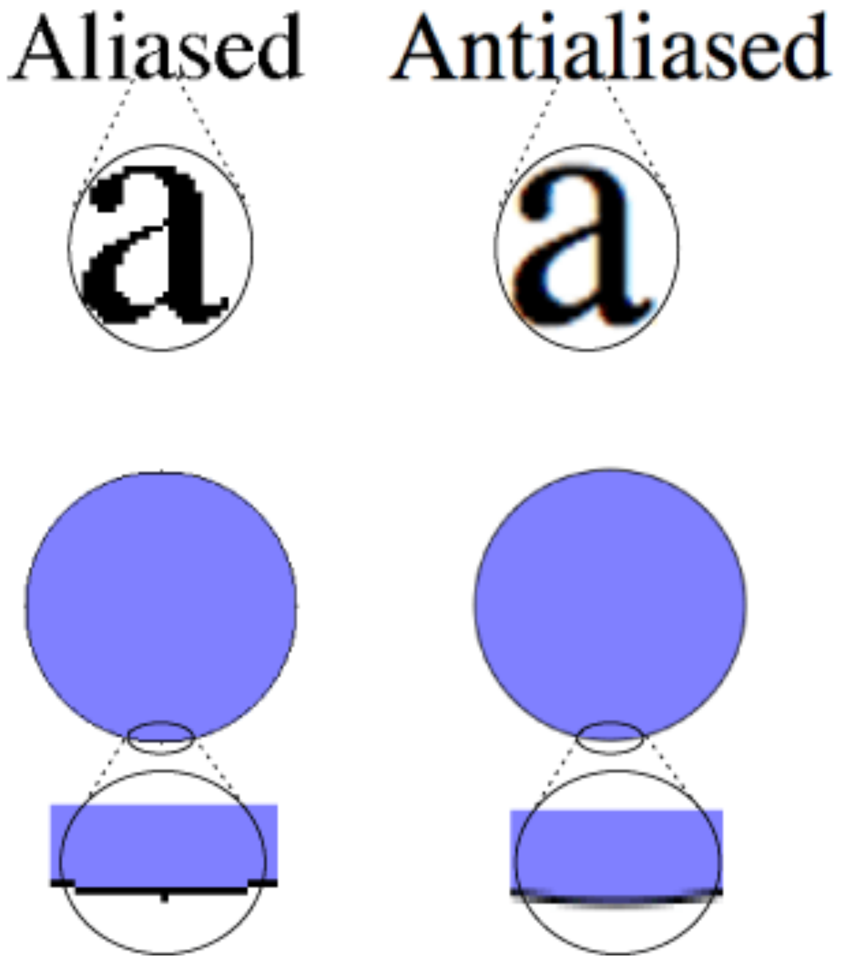
- Output primitives are the basic geometric structures used to form complex structures
- Points, Lines, Circles, Polygons, Conics, Quadrics, Character Strings
- We will look firstly at displaying points and lines on a “raster display device”



Points and Lines

- **Points are the most basic** of the output primitives and plotting them is quite trivial
- For a raster system the position in the frame buffer for a corresponding screen position is set to the required intensity
- Lines on the other hand are a **bit more difficult to scan-convert**
- Lines can be defined in a number of ways e.g. slope-intercept, two points etc.
- To scan convert a line in a raster graphics system where two end points are known we have to **plot discrete coordinate positions along the line path**
- As the screen coordinates are specified discretely raster line drawing results in **lines that are somewhat jagged (aliasing).**
- Adjusting the pixel intensities along the line path can reduce this effect (anti-aliasing)

Aliasing



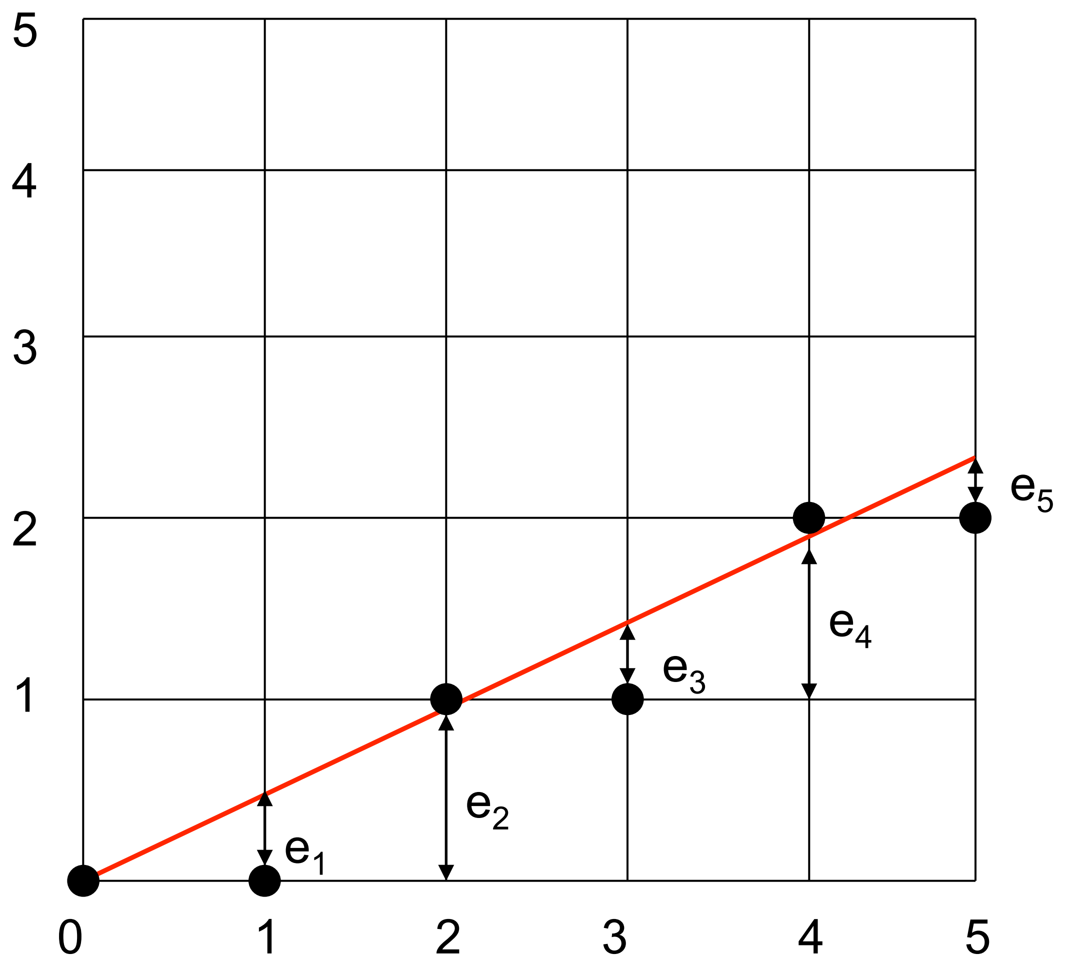
Lines

- Today we will look at a very efficient and accurate raster scan conversion technique for lines in this week's lecture and laboratory
- The technique is called "Bresenham's line drawing algorithm"
- It is an algorithm that uses only incremental integer calculations which are very cheap from a computational point of view
- We will only plot lines with positive slope < 1
- It is important you know how the algorithm is derived
- Understanding the algorithm is the hard part
- Implementing the algorithm is the easy part

Bresenham's Line drawing Algorithm

If e < 0.5, y remains the same, else y++ (for 0<m<1)

$$m = \frac{y_f - y_0}{x_f - x_0}$$



$e_0 = 0$

$e_1 = e_0 + m$

$e_2 = e_1 + m$

$e_3 = e_2 + m - 1$

$e_4 = e_3 + m$

$e_5 = e_4 + m - 1$

Can you see the pattern in d

No re-evaluation of the line equation for a change in X

Bresenham’s Line drawing Algorithm

- This algorithm can be coded efficiently using only incremental calculations.
- But it still has floating point values – now we will remove them
- We can do this by applying the scaling factor, $c = 2(x_f - x_0)$ to all values used in the calculation of the next y value e.g.
- $M = cm = 2(y_f - y_0)$

What else should the scale factor be applied to in the algorithm?????????

- After the scale factor is applied to all the necessary values/variables there will be no floating point calculations left

Output primitives - Circles

Do you remember this equation?

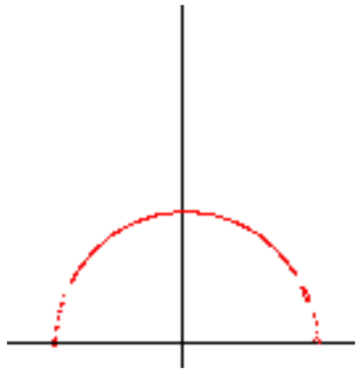
$$f_{circle}(x, y) = (x - x_c)^2 + (y - y_c)^2 = r^2$$

Or with circle centre at the origin?

$$f_{circle}(x, y) = x^2 + y^2 - r^2$$

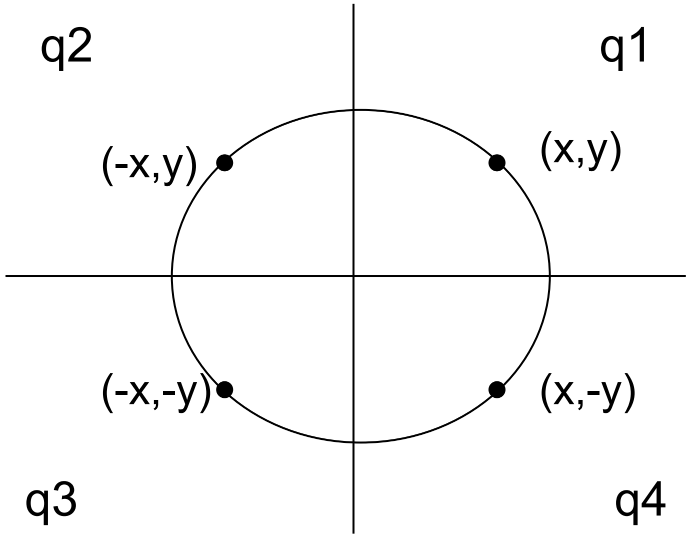
Can it be used to draw a circle on raster scan device?

$$y = \sqrt{r^2 - x^2}$$



What about symmetry?

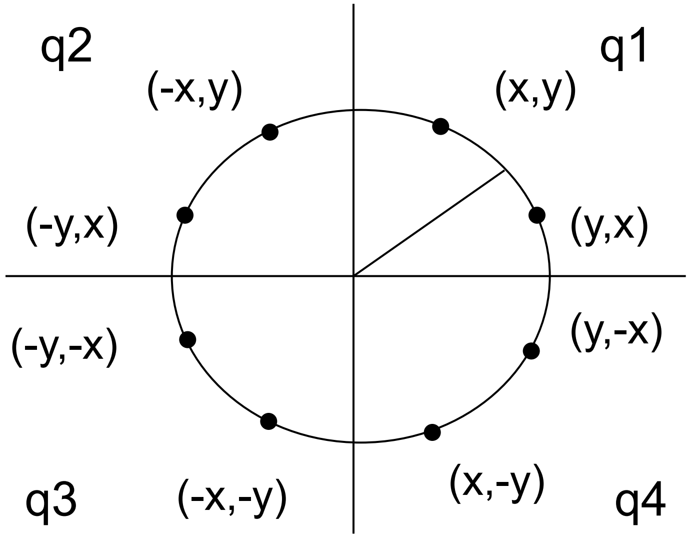
- The shape of the circle is **similar in each quadrant**



- We can use this symmetry so we only need to calculate the **positions on the circumference in one quadrant**

What about symmetry?

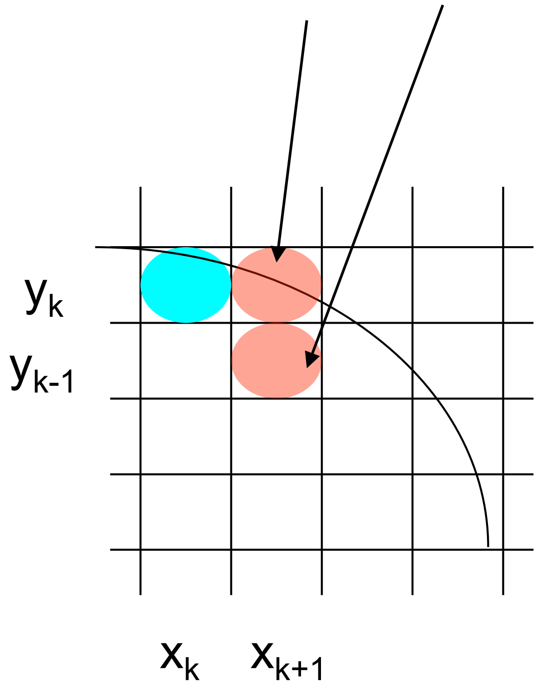
- But there is also **symmetry about adjacent octants** in a quadrant about axes 45 degrees from the positive and negative x axis



- We can use this symmetry so we **only need to calculate the positions on the circumference in one octant**

Calculating the points of the octant x=0 to x=y

- All that needs to be calculated are **the points on one octant** and symmetry used to find the others
- We will use the **Bresenham's circle drawing algorithm** to do this
- Assuming we know the first point on the circle (x_k,y_k) below the next x point is x_{k+1} and the next y value is either y_k or y_{k-1}



Calculating the points of the octant x=0 to x=y

- All points inside the circle have **circle equation less than zero**, **all points on the boundary zero** and **all points outside the circle greater than zero**
- So if we substitute **(x_{k+1}, y_k) and (x_{k+1}, y_{k-1})** into the circle equation, the one who's absolute value || is smaller is closer to the true circle

$$\arg\min(\|f_{circle}(x_{k+1}, y_k)\|, \|f_{circle}(x_{k+1}, y_{k-1})\|)$$

- Just like the line drawing algorithm, the circle algorithm uses incremental integer calculations – this means no floating point numbers and no multiplications
- But evaluating the equation above means substituting points into the circle equation

$$f_{circle}(x, y) = x^2 + y^2 - r^2$$

And this involves squaring x, y and r, but it can be simplified!