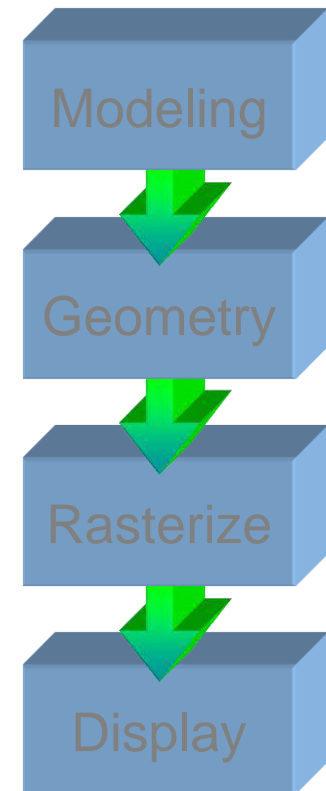


Overview of Graphics Systems

- Four major task for rendering geometric objects
 - modeling
 - geometric processing
 - transformations
 - clipping
 - shading
 - hidden-surface removal
 - projection
 - Rasterization (Scan Conversion)
 - Display



Agenda

1. Input devices
2. Hard-copy devices
3. Video display devices
4. Graphics workstations and viewing systems

Input Devices

- Input devices
 - Keyboards, button boxes, dials
 - Mouse devices
 - Trackballs and space balls
 - Joysticks
 - Data gloves
 - Digitizers
 - Image scanners
 - Touch panels
 - Light pens
 - Voice systems

Input Devices

- Keyboards, button boxes, dials
 - Standard keyboard
 - Alphanumeric
 - Function keys
 - Button box
 - set of input dials

Input Devices

- Mouse devices
 - Mechanical mouse
 - One-button
 - Rotating ball
 - Two perpendicular shafts to capture rotation
 - Optical mouse
 - Optical sensor
 - Laser
 - Grid to detect movement
 - Added widgets
 - Buttons
 - Trackball
 - Thumbwheel.

Input Devices

- Trackball
 - A ball device that can be rotated with the fingers or palm of hand
- Spaceball
 - Six degrees of freedom
 - Does not move, detects strain placed on the ball by trying to move it.

Input Devices

- Joystick
 - A small, vertical lever mounted on a base
 - Movable joystick measures motion
 - Stationary (isometric) joystick measures strain.
- Data glove
 - Used to grasp a virtual object
 - Measures hand and finger position
 - 2D or 3D
 - Can also be used as input device to detect surface

Input Devices

- Digitizers
 - Used for drawing, painting, or selecting positions
 - Graphics tablet used to input 2D coordinates by activating a hand cursor or stylus at given positions on a flat surface
 - Used to trace contours, select precise coordinate positions
 - Hand held cursor
 - Stylus
 - Electromagnetic
 - Grid of wires
 - Electromagnetic pulses send an electrical signal in stylus or cursor
 - Acoustic
 - Sound waves to detect stylus position by microphones
 - Can be 3D

Input Devices

- Image scanners
 - Used to store images on a computer
 - Hand held
 - Flatbed
 - Drum.

Input Devices

- Light pens
 - Pen-shaped device to select screen positions by detecting lights coming from points on the CRT screen
 - Used to capture position of an object or select menu options.

Input Devices

- Voice systems
 - Speech recognition systems to recognize voice commands
 - Used to activate menu options or to enter data
 - Uses a dictionary from a particular user (learning system).

Hard-copy Devices

- Hard-copy devices
 - Plotters
 - 2D moving pen with stationary paper
 - 1D pen and 1D moving paper
 - Printers
 - Impact devices
 - Inked ribbon
 - Non impact devices
 - Laser, ink-jet, xerographic, electrostatic, electrothermal.

Video Display Devices

- Cathode-ray tubes
 - Raster-scan displays
 - Random-scan displays
 - Color CRT displays
 - Direct View Storage Tubes
- Flat-panel displays
- Three-dimensional viewing devices
- Stereoscopic and virtual-reality systems

Cathode-ray tubes

- Raster-scan displays
- Random-scan displays
- Color CRT displays
- Direct View Storage Tubes

Cathode-Ray Tubes

- Classical output device is a monitor.
- Cathode-Ray Tube (CRT)
 - Invented by Karl Ferdinand Braun (1897)

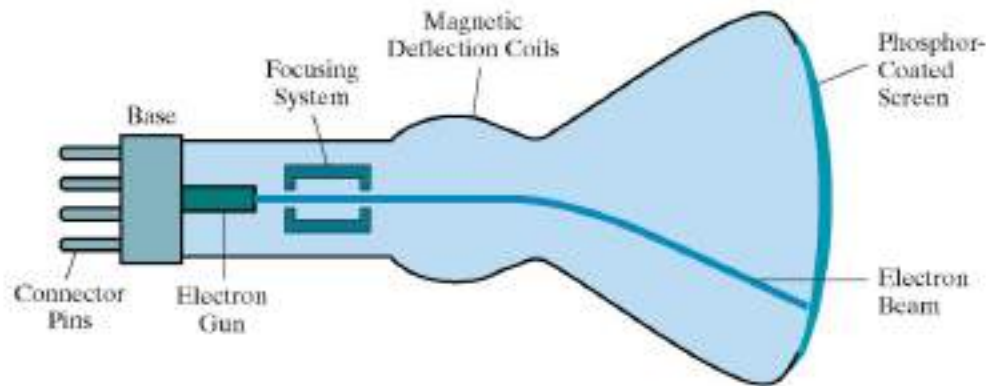


Figure 2-2

Basic design of a magnetic-deflection CRT.

(from Donald Hearn and Pauline Baker)

Cathode-Ray Tubes

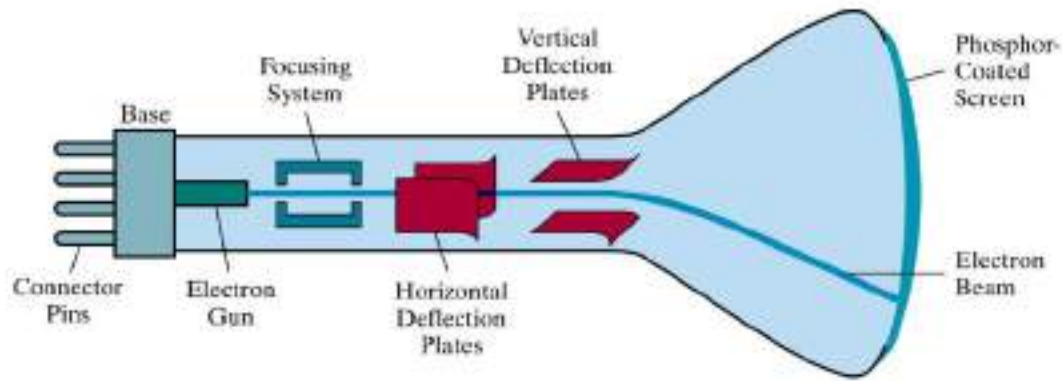


Figure 2-4

Electrostatic deflection of the electron beam in a CRT.

(from Donald Hearn and Pauline Baker)

Cathode-Ray Tubes

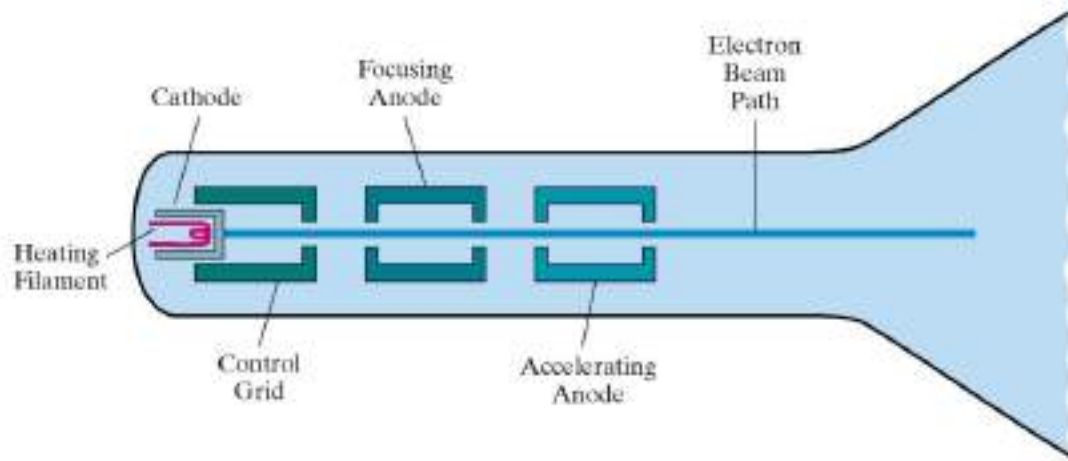


Figure 2-3

Operation of an electron gun with an accelerating anode.

(from Donald Hearn and Pauline Baker)

Cathode-Ray Tubes

1. Working of CRT

- Beam of electrons directed from cathode (-) to phosphor-coated (fluorescent) screen (anode (+))
- Directed by magnetic focusing and deflection coils (anodes) in vacuum filled tube
- Phosphor emits photon of light, when hit by an electron, of varied persistence (long 15-20 ms for texts / short < 1 ms for animation)
- Refresh rate (50-60 Hz / 72-76 Hz) to avoid flicker / trail
- Phosphors are organic compounds characterized by their persistence and their color (blue, red, green).

Cathode-Ray Tubes

- Horizontal deflection and vertical deflection direct the electron beam to any point on the screen
- Intensity knob: regulates the flow of electrons by controlling the voltage at the control grid (high voltage reduces the electron density and thus brightness)
- Accelerating voltage from positive coating inside screen (anode screen) or an accelerating anode

2. Image maintenance

- Charge distribution to store picture information OR
- Refresh CRT: refreshes the display constantly to maintain phosphor glow.

Cathode-Ray Tubes

3. Focusing

- *Focusing* forces the electron beam to converge to a point on the monitor screen
- Can be electrostatic (lens) or magnetic (field)

4. Deflection

- *Deflection* directs the electron beam horizontally and/or vertically to any point on the screen
- Can be controlled by electric (deflection plates, slide 9) or magnetic fields (deflection coils, slide 5)
- Magnetic coils: two pairs (top/bottom, left/right) of tube neck
- Electric plates: two pairs (horizontal, vertical)

Cathode-Ray Tubes

Characteristics of Cathode-Ray Tube (CRT)

1. **Intensity** is proportional to the number of electrons repelled in beam per second (**brightness**)
2. **Resolution** is the maximum number of points that can be displayed without overlap; is expressed as number of horizontal points by number of vertical points; points are called pixels (picture elements); example: resolution 1024 x 768 pixels. Typical resolution is 1280 x 1024 pixels.
 - High-definition systems: high resolution systems.

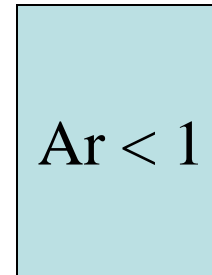
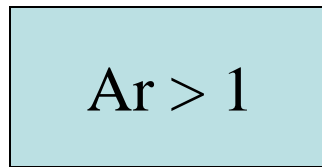
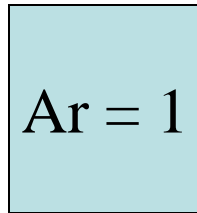
Cathode-Ray Tubes

3. **Persistence** is defined as the time taken by the emitted light to decay one tenth of its original intensity.
 - Max persistence 1 Sec, Min Persistence 10-60 μ sec
 - Higher persistence ~ Low refresh rate ~ complex images
 - Lower persistence ~ High refresh rate ~ Animations
4. **Refresh Rate** (Hz) number of times screen drawn or refreshed per second.
 - Usually 60 Hz (Why)
 - Depends upon persistence
5. **Pixel Picture Element**
 - Mapping of phosphorus element to pixel
 - Bit for monochrome
 - Byte for 256 color levels
 - 3 Bytes to produce more than 16.7 million colors

Cathode-Ray Tubes

Aspect ratio

- *Aspect ratio* is the ratio of vertical pixels to horizontal pixels for an equal length line.
- A square plotted with same number of pixels with different aspect ratios will look as:



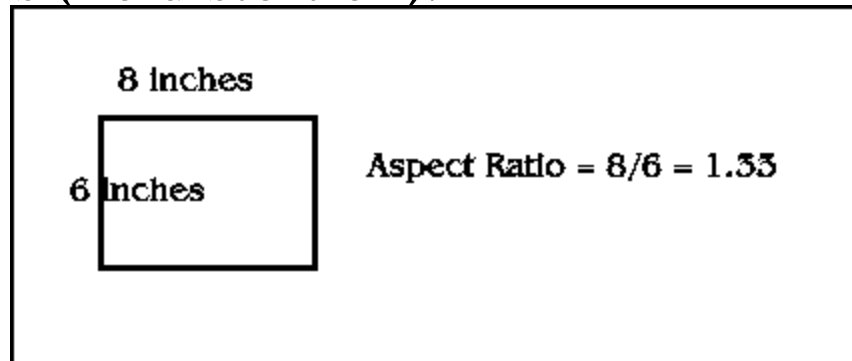
Cathode-Ray Tubes

– It is also defined as the ratio of the vertical dimension over the horizontal dimension. If and resolution of 640 x 480 pixels:

→ Horizontal $640/8 = 80$ pixels / inch

→ Vertical $480/6 = 80$ pixels / inch

Square pixels (no distortion).



Cathode-ray tubes

- **Raster-scan displays**
- Random-scan displays
- Color CRT displays
- Direct View Storage Tubes

Raster-scan Displays

1. Introduction

- Raster-scan display is the most common type of monitor using a CRT.
- A *raster* is a matrix of pixels covering the screen area and is composed of raster lines.
- The electron beam scans the screen from top to bottom one row at a time. Each row is called a scan line.
- The electron beam is turned on and off to produce a collection of dots painted one row at a time. These will form the image.

Raster-scan Displays

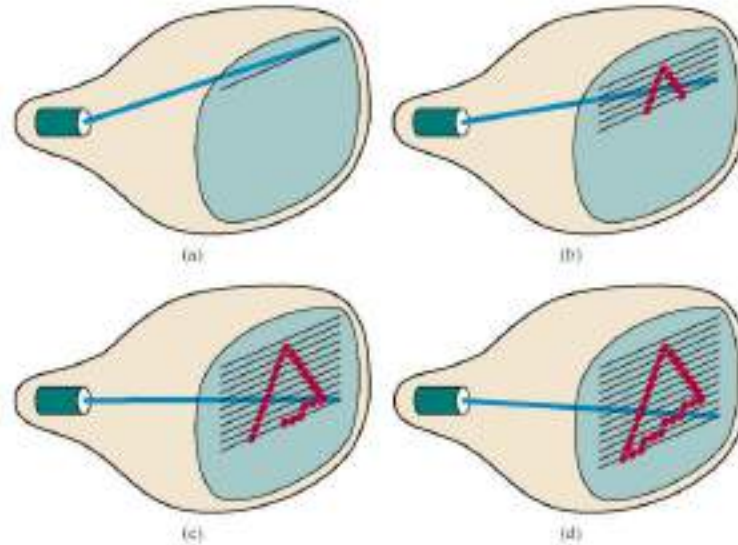


Figure 2-7

A raster-scan system displays an object as a set of discrete points across each scan line.

(from Donald Hearn and Pauline Baker)

Raster-scan Displays

2. Refresh Procedure

- Retracing
 - Horizontal retrace – beam returns to left of screen
 - Vertical retrace – beam returns to top left corner of screen
- Blanking
 - Horizontal Retrace Blanking
 - Vertical Retrace Blanking
- Interlacing
 - display first even-numbered lines, then odd-numbered lines
 - permits to see the image in half the time
 - useful for slow refresh rates (30 Hz shows as 60 Hz).

Raster-scan Displays

– Over scanning

- Scan lines extended beyond visibility edge as there is limit on speed of sweep generator
- Avoid cracking at borders and distortion
- Top and Bottom Vertical Over scanning
- Left and Right Horizontal Over scanning

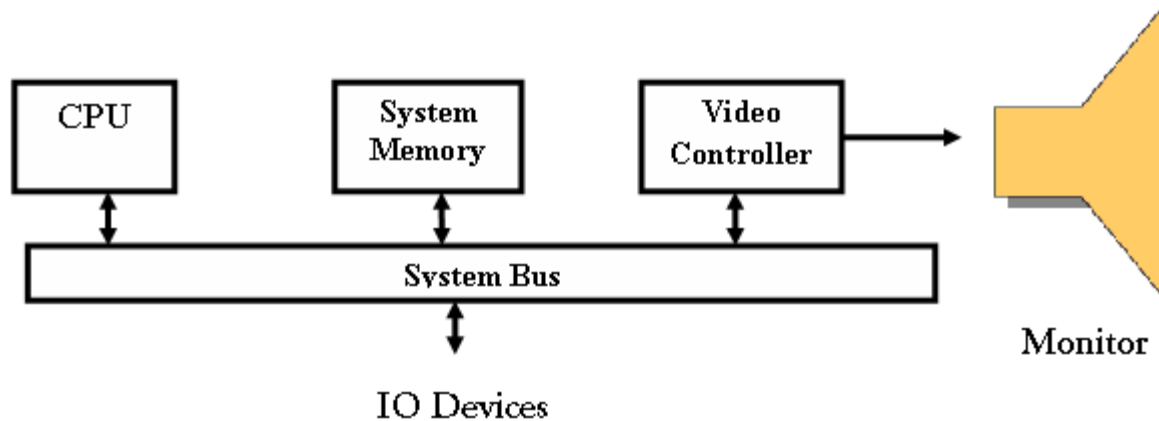
– Refresh rate

- 24 is a minimum to avoid flicker, corresponding to 24 Hz (1 Hz = 1 refresh per second)
- Current raster-scan displays have a refresh rate of at least 60 frames (60 Hz) per second, up to 120 (120 Hz).

Raster-scan Displays

3. Architecture of Raster Scan System

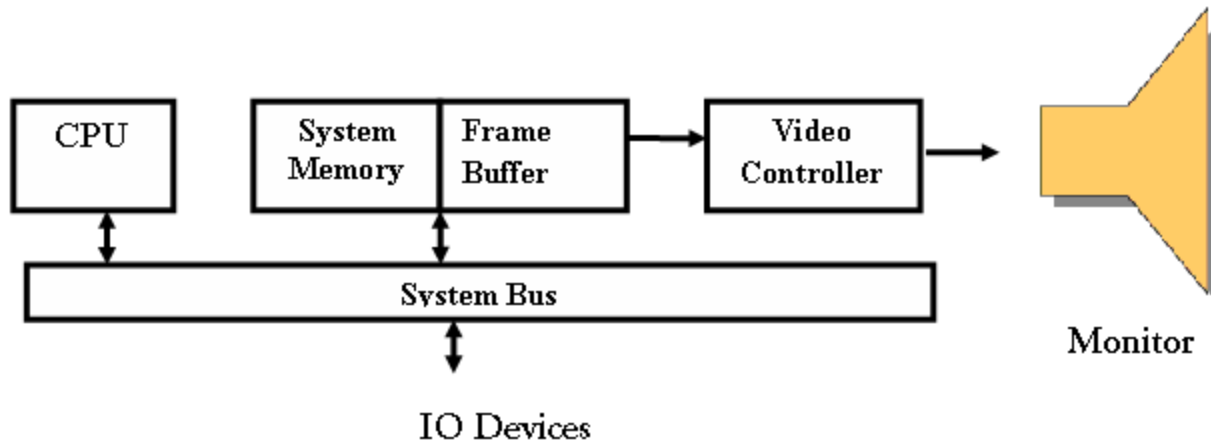
- Simple architecture



(from Donald Hearn and Pauline Baker)

Raster-scan Displays

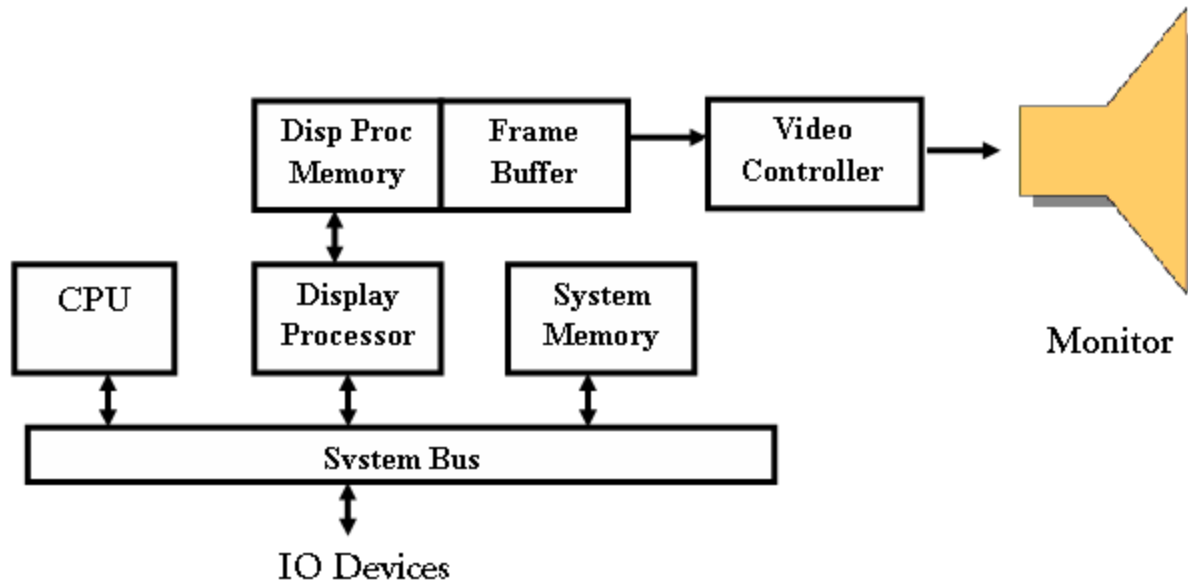
- Architecture with reserved Frame Buffer



(from Donald Hearn and Pauline Baker)

Raster-scan Displays

- Architecture with reserved Frame Buffer and separate Display Processor



(from Donald Hearn and Pauline Baker)

Raster-scan Displays

3.1 Frame Buffer

- Also called Refresh Buffer, contains picture definition
- The image is stored in a *frame buffer* containing the total screen area and where each memory location corresponds to a pixel.
- Consider it as 2-D memory array
- E.g. Frame buffer
 - size 8x8
 - Color depth 8 (values 0-7)
- **Uses large memory:**
640x480 → 307200 bits → 38 kB

3	0	0	4	2	0	0	1
2	7	5	0	6	0	0	0
1	5	6	0	4	0	0	0
0	6	7	1	1	1	0	0
0	5	1	1	1	1	1	0
3	3	3	3	3	3	3	3
0	0	3	2	5	5	4	1
6	4	5	3	2	6	5	2

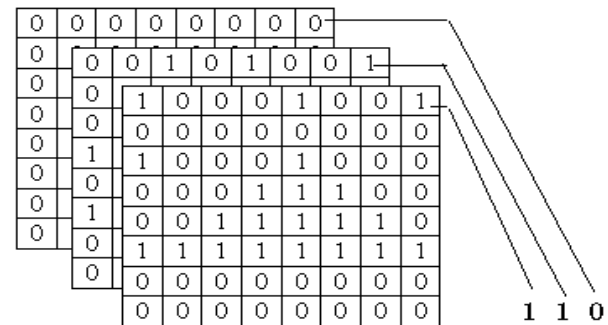
Raster-scan Displays

- ***Bitmap*** In a monochrome system, each bit is 1 or 0 for the corresponding pixel to be on or off making frame a bitmap.
- The display processor scans the frame buffer to turn electron beam on/off depending if the bit is 1 or 0.
- Example Bitmap

1	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
0	0	0	1	1	1	0	0
0	0	1	1	1	1	1	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Raster-scan Displays

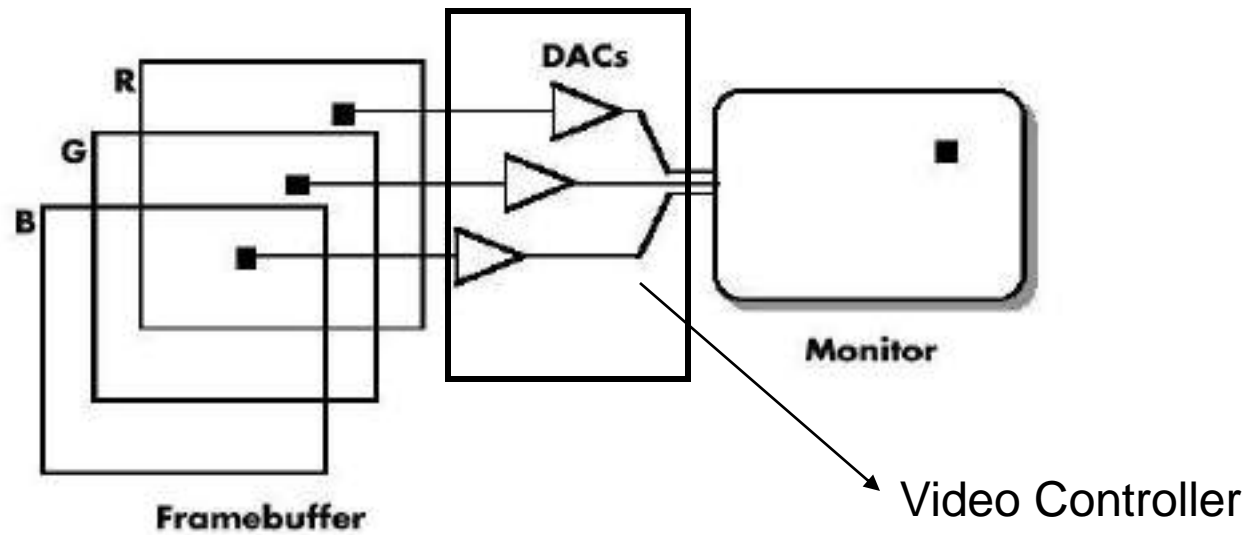
- *Pixmap* for color monitors, the frame buffer also contains the color of each pixel (color buffer) as well as other characteristics of the image (gray scale, ...).
- Depth of the buffer area is the number of bits per pixel (bit planes), up to 24. 8 bits/pixel \rightarrow 0..255
- Examples: television panels, printers, PC monitors
- 8 level Pixmap



Raster-scan Displays

3.2 Video Controller

- Also called scan controller
- display an image onto screen
- Read the frame buffer and display on screen



Raster-scan Displays

- How each pixel value in the frame buffer is sent to the right place on the display surface

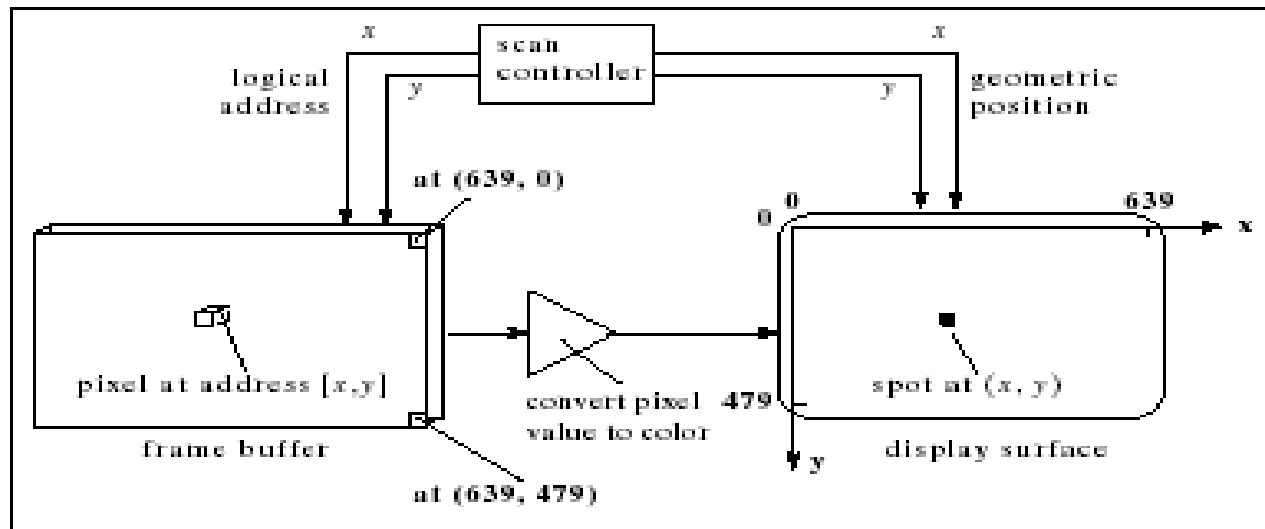
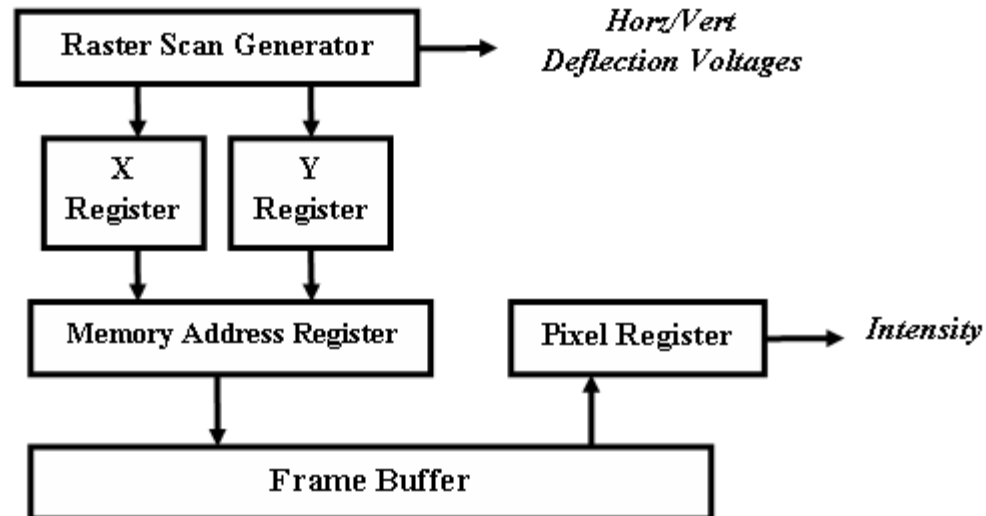


FIGURE 1.37 Scanning out an image from the frame buffer to the display surface.

Raster-scan Displays

- Basic Operation of Video Controller/Scan Controller



(from Donald Hearn and Pauline Baker)

Raster-scan Displays

3.3 Display Processor

- Relieves CPU from graphics chores
- It digitize the picture definition in the application program
- It does SCAN CONVERSION
- Define Graphic objects and characters to be displayed

Cathode-ray tubes

- Raster-scan displays
- Random-scan displays**
- Color CRT displays
- Direct View Storage Tubes

Random-scan Displays

1. Introduction

- Random scan systems are also called
 - Vector Displays
 - stroke-writing, or
 - calligraphic displays.
- The electron beam directly draws the picture in any specified order.
- A pen plotter is an example of such a system.

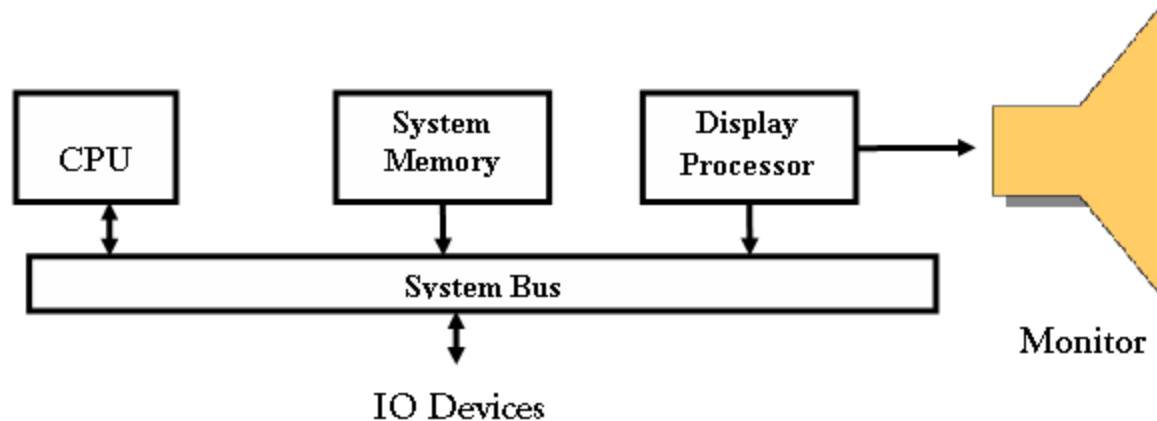
Random-scan Displays

- Picture is stored in a display list, refresh display file, vector file, or display program as a set of line drawing commands.
- Refresh rate depends upon the size of the file.
- Refreshes by scanning the list 30 to 60 times per second.
- More suited for line-drawing applications such as architecture and manufacturing

Random-scan Displays

2. Architecture of Random Scan System

- Simple architecture



(from Donald Hearn and Pauline Baker)

Random-scan Displays

3. Advantages:

- Good quality lines
- No need of scan conversion
- Easy animation and requires little memory

4. Disadvantages:

- Requires intelligent electron beam (processor controlled)
 - Limited screen density, limited to simple, line-based images
 - Limited color capability.
- Improved in the 1960's by the Direct View Storage Tube (DVST) from Tektronix.

Raster vs. Random-scan Displays

	RASTER	RANDOM
DISPLAY MECHANISM	E-beam traces entire screen from upper left corner to bottom right	E-beam can highlight random positions on the screen
DRAWING UNIT	Pixel	Line
IMAGE STORAGE	Frame Buffer	Display File
IMAGE TYPES	Can display very complex images with greater accuracy	Wire Frame modeling
IMAGE QUALITY	<ul style="list-style-type: none"> •May be Jagged due to digitization •Diagonal Lines are produced with lower intensity 	<ul style="list-style-type: none"> •Smooth lines as e-beam directly follows the line path •Diagonal Lines are produced with equal intensity
REFRESHING	Entire Screen has to be refreshed	Only selected portions are redrawn
REFRESH RATE	Maximum 80 Hz	Higher refresh rates.
ANIMATIONS	Supported	Not supporting
COLORS	Higher Color Depth	Lesser colors and shades
COLOR TECHNIQUE	Shadow Masking	Beam Penetration

Cathode-ray tubes

- Raster-scan displays
- Random-scan displays
- Color CRT displays**
- Direct View Storage Tubes

Color CRT Monitor

1. Introduction

- Uses different phosphors, a combination of Red, Green, and Blue, to produce any color.
- Two methods:
 - Beam penetration
 - Shadow Masking

Color CRT Monitor

2. Beam Penetration Method

- Random scan systems uses beam penetration.
- 2 layers (Red, Green) phosphors; low speed electrons excite Red, high speed electrons excite Green.
- Intermediate speed excite both to get yellow and orange.
- Color is controlled by electron beam voltage.
- It is inexpensive
- Only produces a restricted set of colors.
- Quality of Picture is low

Color CRT Monitor

3. Shadow Masking Method

- Raster scan systems uses a shadow mask with three electron guns: Red, Green, and Blue (RGB color model).
- Color is produced by adjusting the intensity level of each electron beam.
- Produces a wide range of colors, from 8 to several millions.
- The arrangement of color components can be
 - Delta-Delta arrangement
 - In line arrangement

Color CRT Monitor

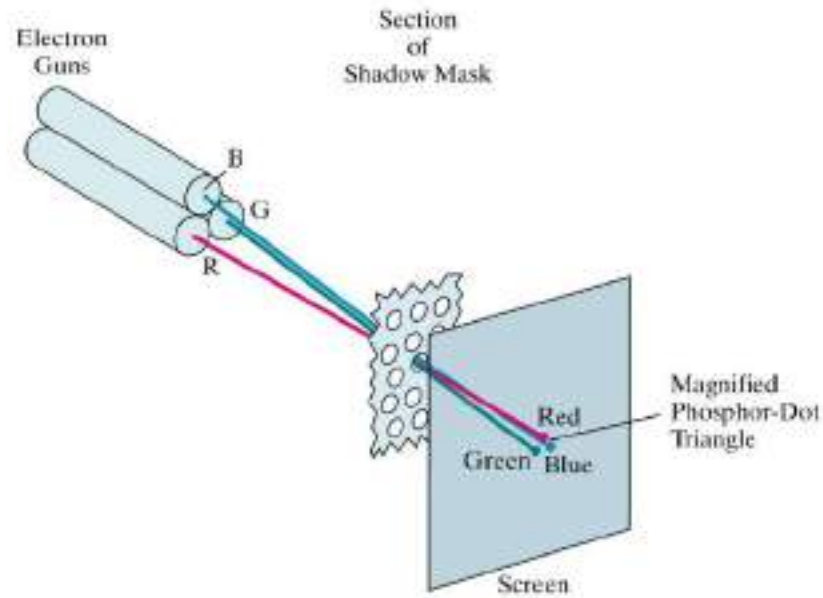


Figure 2-10

Operation of a delta-delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

(from Donald Hearn and Pauline Baker)

Color CRT Monitor

R G B color

0 0 0 black

0 0 1 blue

0 1 0 green

0 1 1 cyan

1 0 0 red

1 0 1 magenta

1 1 0 yellow

1 1 1 white

Color CRT Monitor

- Color CRT's are designed as RGB monitors also called full-color system or true-color system.
- Use shadow-mask methods with intensity from each electron gun (red, green, blue) to produce any color directly on the screen without preprocessing.
- Frame buffer contains 24 bits per pixel, for 256 voltage settings to adjust the intensity of each electron beam, thus producing a choice of up to 17 million colors for each pixel (256^3).

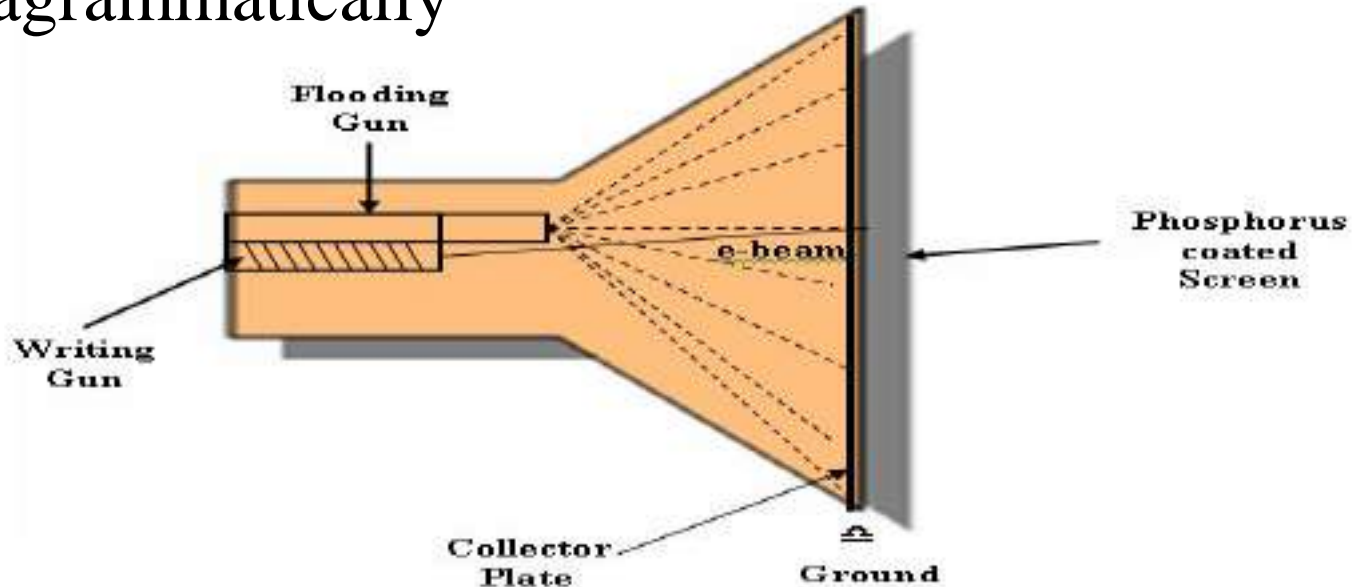
Cathode-ray tubes

- Raster-scan displays
- Random-scan displays
- Color CRT displays
- Direct View Storage Tubes**

DVST Displays

1. Introduction

- Picture is stored inside the CRT
- No refreshing required
- Diagrammatically



DVST Displays

2. Components

- *Flooding Gun* to flood the entire screen and charge the collector plate
- *Writing Gun* is same as e-gun in CRT having heating filament, cathode, focusing anode and deflection yokes
- *Collector Plate* partly energized by the flooding gun, has background charge to keep fired phosphorus illuminated
- *Phosphorus Screen* higher persistence CRT screen
- *Ground* to discharge the collector to erase the screen

DVST Displays

3. Advantages/Disadvantages

- No Refreshing required
- It can draw complex images with higher resolution
- Does not display colors
- Selected part of the picture cannot be erased
- Animation not supported

Flat Panel Displays

Flat Panel Displays

1. Introduction

- Flat panel displays are video devices that are thinner, lighter, and require less power than CRT's.
- Examples: wall frames, pocket notepads, laptop computer screens, ...

2. Types of Flat Panel Displays

- **Emissive panels** convert electrical energy into light: plasma panels, thin-film electroluminescent display device, light-emitting diodes.
- **Non-emissive** convert light into graphics using optical effects:
liquid-crystal device (LCD).

Flat Panel Displays

2.1 Plasma-panel display:

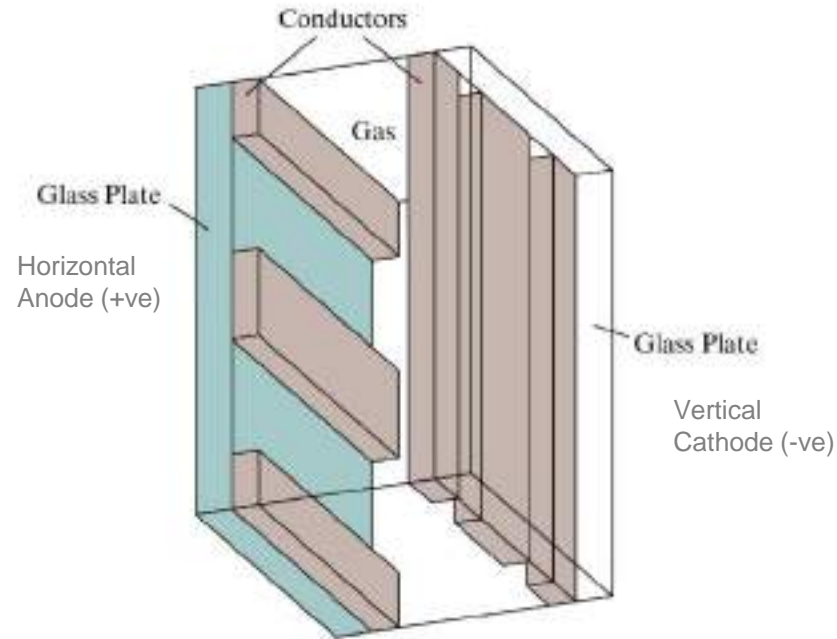


Figure 2-11

Basic design of a plasma-panel display device.

(from Donald Hearn and Pauline Baker)

Flat Panel Displays

Components of Plasma-panel displays

- **Cathode** Fine wires attached to glass plates deliver –ve voltage to gas cells on vertical axis
- **Fluorescent cells** Small pockets of gas liquid or solids to emit light in excited state
- **Anode** Fine wires attached to glass plates deliver +ve voltage to gas cells on horizontal axis
- **Glass Plates** to act as capacitors to maintain sustaining voltage

Flat Panel Displays

Working of Plasma-panel displays

- An array of small fluorescent gas lights
- Constructed by filling a mixture of gases (usually neon) between two glass plates
- vertical conducting ribbons are placed in one plate, and horizontal conducting ribbons are placed in the other plate
- voltage is applied to the two ribbons to transform gas into glowing plasma of electrons and ions.

Flat Panel Displays

- Two voltage levels
 - Firing Voltage 60
 - Sustaining Voltage 40

Advantages/Disadvantages:

- No need of refreshing
- Provides Fairly High resolution
- However MONOCHROME with few grey levels

Flat Panel Displays

2.2 Thin-film electroluminescent display:

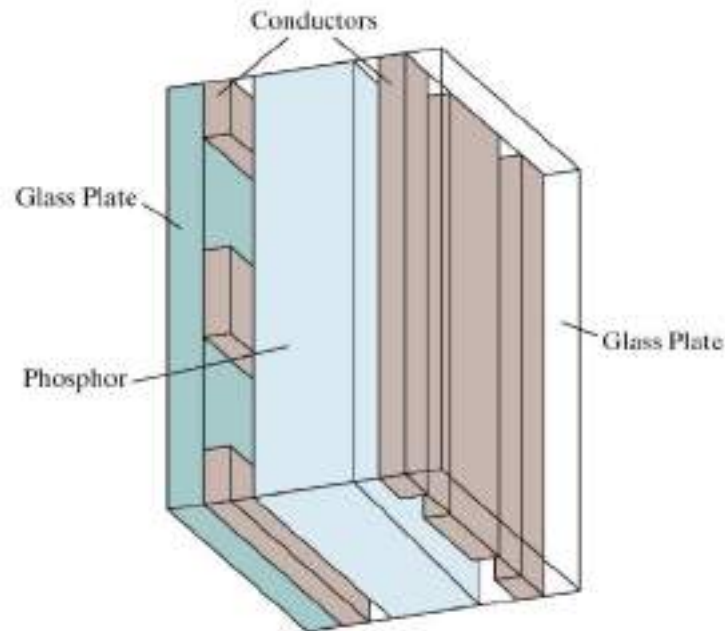


Figure 2-13

Basic design of a thin-film electroluminescent display device.

Flat Panel Displays

Thin-film electroluminescent displays are

- similar devices except that the region between the plates is filled with phosphor instead of gas.

Example: zinc sulfide with manganese

voltage applied between the plates moves electrons to the manganese atoms that release photons of light.

Flat Panel Displays

2.3 Light-emitting diode:

- a matrix of diodes, one per pixel
- apply voltage stored in the refresh buffer
- convert voltage to produce light in the display.

Flat Panel Displays

2.4 Liquid-crystal displays (LCD):

- LCD screens are often used in small devices such as calculators and laptop monitors.
- non-emissive types of displays
- the picture produced by passing light from a light source through liquid-crystal material
- Liquid-crystal material contains crystals within a liquid nematic (thread-like) liquid-crystals have rod shape that can either align to with the light direction or not when voltage is applied to conductors.
- Liquid-crystal material can be programmed to either let the light through or not

Flat Panel Displays

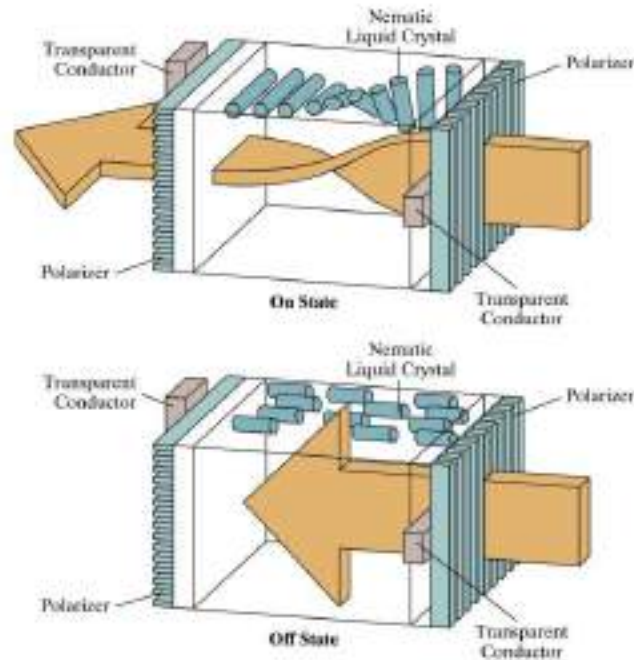


Figure2-15

The light-twisting, shutter effect used in the design of most liquid-crystal display devices.

(from Donald Hearn and Pauline Baker)

Flat Panel Displays

Components of Liquid Crystal Displays

- **Glass Plates** contains the liquid crystal and serve as bonding surface for conductive coating.
- **Transparent Conductor** To apply voltage to two ribbons (across liquid crystals) to make plasma glow.
- **Liquid Crystals** Substance that polarize light when voltage is applied.
- **Two Polarized Films** Transparent sheet that polarize light.
- **Reflectors**

Flat Panel Displays

- ON STATE when light is twisted
- OFF STATE when block the light
- Passive matrix LCD have refresh buffer
screen refreshed at 60 frames per second
- Active matrix LCD transistor stored at each
pixel prevents charge from leaking out of
liquid-crystals
- Temperature dependent, sluggish
- Require little power to operate

3-D Viewing Devices

- For the display of 3D scenes.
- Often using a vibrating, flexible mirror.
- Scan alternate images in alternate frames.
- Multiple stereo images (time multiplexing)

Stereoscopic and Virtual-Reality Systems

- Another technique for the display of 3D scenes.
- Not true 3D images, but provides a 3D effect.
- Uses two views of a scene along the lines of right and left eye. Gives perception of a scene depth when right view is seen from right eye and left scene is seen from left eye (stereoscopic effect). Display each view at alternate refresh cycles.

Stereoscopic and Virtual-Reality Systems

- Stereoscopic systems are used in virtual reality systems:
 - Augmented reality
 - Immersive reality
 - Headset generates stereoscopic views
 - Input devices (gloves, helmet, ...) capture motion
 - Sensing system in headset tracks user's position
- Scene projected on an arrangement of walls

Graphics Workstations

- Graphics monitors use raster-scan displays (CRT or flat-panel monitors).
- Graphics workstations provide more powerful graphics capability:
 - Screen resolution 1280 x 1024 to 1600 x 1200.
 - Screen diagonal > 18 inches.
- Specialized workstations (medical imaging, CAM):
 - Up to 2560 x 2048.
 - Full-color.
- 360 degrees panel viewing systems.

2-D Transformations

Contents

- 1. Homogeneous coordinates**
2. Matrices
3. Transformations
4. Geometric Transformations
5. Inverse Transformations
6. Coordinate Transformations
7. Composite transformations

Homogeneous Coordinates

- There are three types of co-ordinate systems
 1. *Cartesian Co-ordinate System*
 - *Left Handed Cartesian Co-ordinate System*(Clockwise)
 - *Right Handed Cartesian Co-ordinate System* (Anti Clockwise)
 2. *Polar Co-ordinate System*
 3. *Homogeneous Co-ordinate System*

We can always change from one co-ordinate system to another.

Homogeneous Coordinates

- A point (x, y) can be re-written in **homogeneous coordinates** as (x_h, y_h, h)
- The **homogeneous parameter** h is a non-zero value such that:

$$x = \frac{x_h}{h} \quad y = \frac{y_h}{h}$$

- We can then write any point (x, y) as (hx, hy, h)
- We can conveniently choose $h = 1$ so that (x, y) becomes $(x, y, 1)$

Homogeneous Coordinates

Advantages:

1. Mathematicians use homogeneous coordinates as they allow scaling factors to be removed from equations.
2. All transformations can be represented as 3×3 matrices making homogeneity in representation.
3. Homogeneous representation allows us to use matrix multiplication to calculate transformations extremely efficient!
4. Entire object transformation reduces to single matrix multiplication operation.
5. Combined transformations are easier to build and understand.

Contents

1. Homogeneous coordinates
- 2. Matrices**
3. Transformations
4. Geometric Transformations
5. Inverse Transformations
6. Coordinate Transformations
7. Composite transformations

Matrices

- **Definition:** A *matrix* is an $n \times m$ array of scalars, arranged conceptually in n rows and m columns, where n and m are positive integers. We use A , B , and C to denote matrices.
- If $n = m$, we say the matrix is a **square matrix**.
- We often refer to a matrix with the notation $\mathbf{A} = [\mathbf{a(i,j)}]$, where $a(i,j)$ denotes the scalar in the i th row and the j th column
- Note that the text uses the typical mathematical notation where the i and j are subscripts. We'll use this alternative form as it is easier to type and it is more familiar to computer scientists.

Matrices

- **Scalar-matrix multiplication:**

$$\alpha A = [\alpha a(i,j)]$$

- **Matrix-matrix addition:** A and B are both $n \times m$

$$C = A + B = [a(i,j) + b(i,j)]$$

- **Matrix-matrix multiplication:** A is $n \times r$ and B is $r \times m$

$$C = AB = [c(i,j)] \quad \text{where} \quad c(i,j) = \sum_{k=1}^r a(i,k) b(k,j)$$

Matrices

- **Transpose:** A is $n \times m$. Its transpose, A^T , is the $m \times n$ matrix with the rows and columns reversed.
- **Inverse:** Assume A is a square matrix, i.e. $n \times n$. The identity matrix, I_n has 1s down the diagonal and 0s elsewhere. The inverse A^{-1} does not always exist. If it does, then

$$A^{-1} A = A A^{-1} = I$$

Given a matrix A and another matrix B , we can check whether or not B is the inverse of A by computing AB and BA and seeing that $AB = BA = I$

Matrices

- Each point $P(x,y)$ in the homogenous matrix form is represented as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{3 \times 1}$$

- Recall matrix multiplication takes place:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}_{3 \times 3} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{3 \times 1} = \begin{bmatrix} a * x + b * y + c * z \\ d * x + e * y + f * z \\ g * x + h * y + i * z \end{bmatrix}_{3 \times 1}$$

Matrices

- Matrix multiplication does NOT *commute*:

$$\mathbf{M N} \neq \mathbf{N M}$$

- (unless one or the other is a uniform scale)

- Matrix composition works *right-to-left*.

- Compose:

$$\mathbf{M} = \mathbf{A B C}$$

- Then apply it to a column matrix \mathbf{v} :

$$\mathbf{v}' = \mathbf{M v}$$

$$\mathbf{v}' = (\mathbf{A B C}) \mathbf{v}$$

$$\mathbf{v}' = \mathbf{A} (\mathbf{B} (\mathbf{C v}))$$

- It first applies \mathbf{C} to \mathbf{v} , then applies \mathbf{B} to the result, then applies \mathbf{A} to the result of that.

Contents

1. Homogeneous coordinates
2. Matrices
- 3. Transformations**
4. Geometric Transformations
5. Inverse Transformations
6. Coordinate Transformations
7. Composite transformations

Transformations

- A *transformation* is a function that maps a point (or vector) into another point (or vector).
- An *affine transformation* is a transformation that maps lines to lines.
- Why are affine transformations "nice"?
 - We can define a polygon using only points and the line segments joining the points.
 - To move the polygon, if we use affine transformations, we only must map the points defining the polygon as the edges will be mapped to edges!
- We can model many objects with polygons---and should---for the above reason in many cases.

Transformations

- Any affine transformation can be obtained by applying, in sequence, transformations of the form
 - Translate
 - Scale
 - Rotate
 - Reflection
- So, to move an object all we have to do is determine the sequence of transformations we want using the 4 types of affine transformations above.

Transformations

- **Geometric Transformations:** In Geometric transformation an object itself is moved relative to a stationary coordinate system or background. The mathematical statement of this view point is described by geometric transformation applied to each point of the object.
- **Coordinate Transformation:** The object is held stationary while coordinate system is moved relative to the object. These can easily be described in terms of the opposite operation performed by Geometric transformation.

Transformations

- What does the transformation do?
- What matrix can be used to transform the original points to the new points?
- Recall--- moving an object is the same as changing a frame so we know we need a 3×3 matrix
- It is important to remember the form of these matrices!!!

Contents

1. Homogeneous coordinates
2. Matrices
3. Transformations
- 4. Geometric Transformations**
5. Inverse Transformations
6. Coordinate Transformations
7. Composite transformations

Geometric Transformations

- In Geometric transformation an object itself is moved relative to a stationary coordinate system or background. The mathematical statement of this view point is described by geometric transformation applied to each point of the object. Various Geometric Transformations are:

- **Translation**
- Scaling
- Rotation
- Reflection
- Shearing

Geometric Transformations

–**Translation**

–Scaling

–Rotation

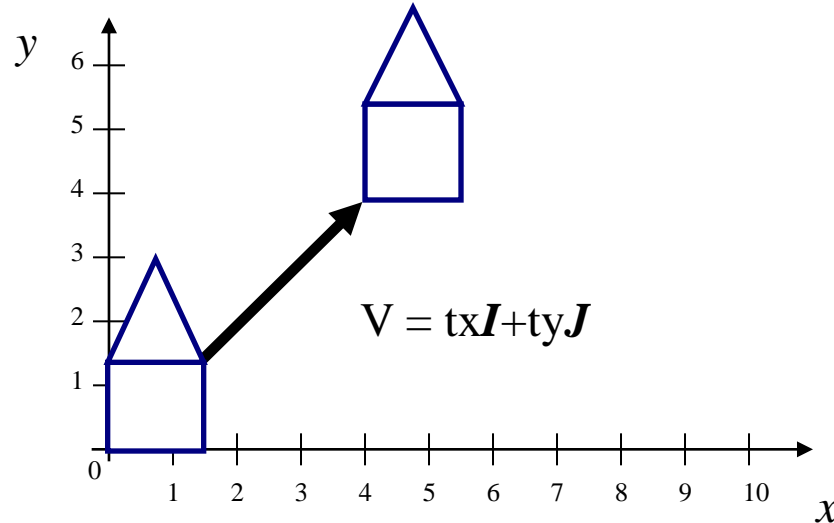
–Reflection

–Shearing

Geometric Translation

- Is defined as the displacement of any object by a given distance and direction from its original position. In simple words it moves an object from one position to another.

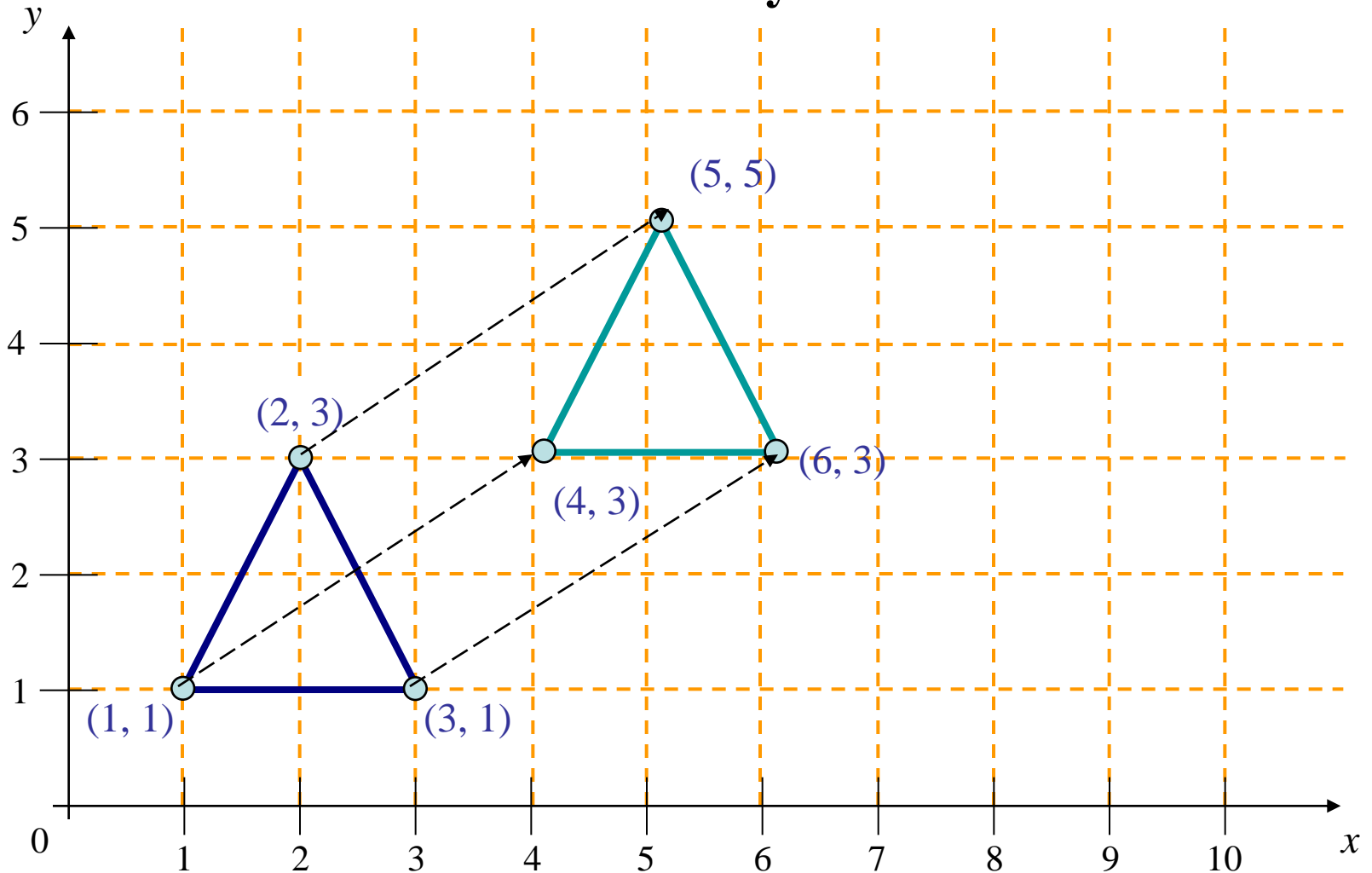
$$x' = x + tx \qquad y' = y + ty$$



Note: House shifts position relative to origin

Geometric Translation Example

Translation by $3\mathbf{i}+2\mathbf{j}$



Geometric Translation

- To make operations easier, 2-D points are written as homogenous coordinate column vectors
- The translation of a point $P(x,y)$ by (tx, ty) can be written in matrix form as:

$$P' = T_v(P) \quad \text{where } v = tx\vec{I} + ty\vec{J}$$

$$T_v = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Geometric Translation

- Representing the point as a homogeneous column vector we perform the calculation as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1*x + 0*y + tx*1 \\ 0*x + 1*y + ty*1 \\ 0*x + 0*y + 1*1 \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \\ 1 \end{bmatrix}$$

on comparing

$$x' = x + tx$$

$$y' = y + ty$$

Geometric Transformations

–Translation

–**Scaling**

–Rotation

–Reflection

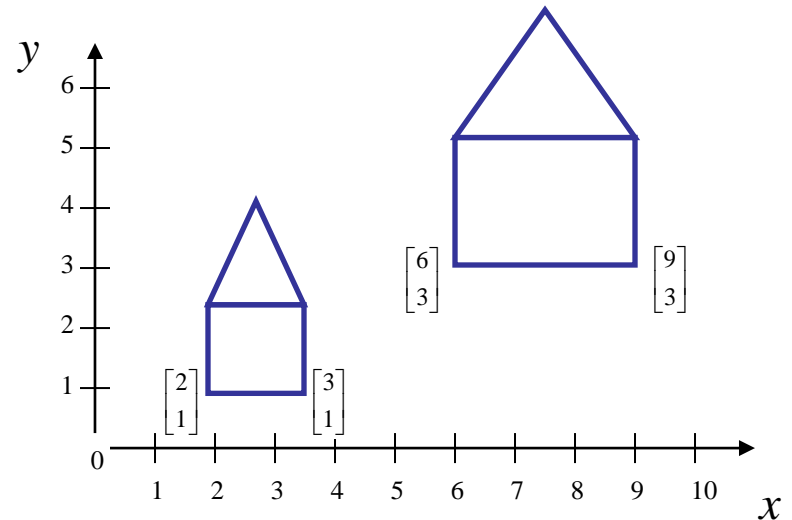
–Shearing

Geometric Scaling

- Scaling is the process of expanding or compressing the dimensions of an object determined by the scaling factor.
- Scalar multiplies all coordinates

$$x' = Sx \times x \quad y' = Sy \times y$$

- **WATCH OUT:**
 - Objects grow and move!



Note: House shifts position relative to origin

Geometric Scaling

- The scaling of a point $P(x,y)$ by scaling factors S_x and S_y about origin can be written in matrix form as:

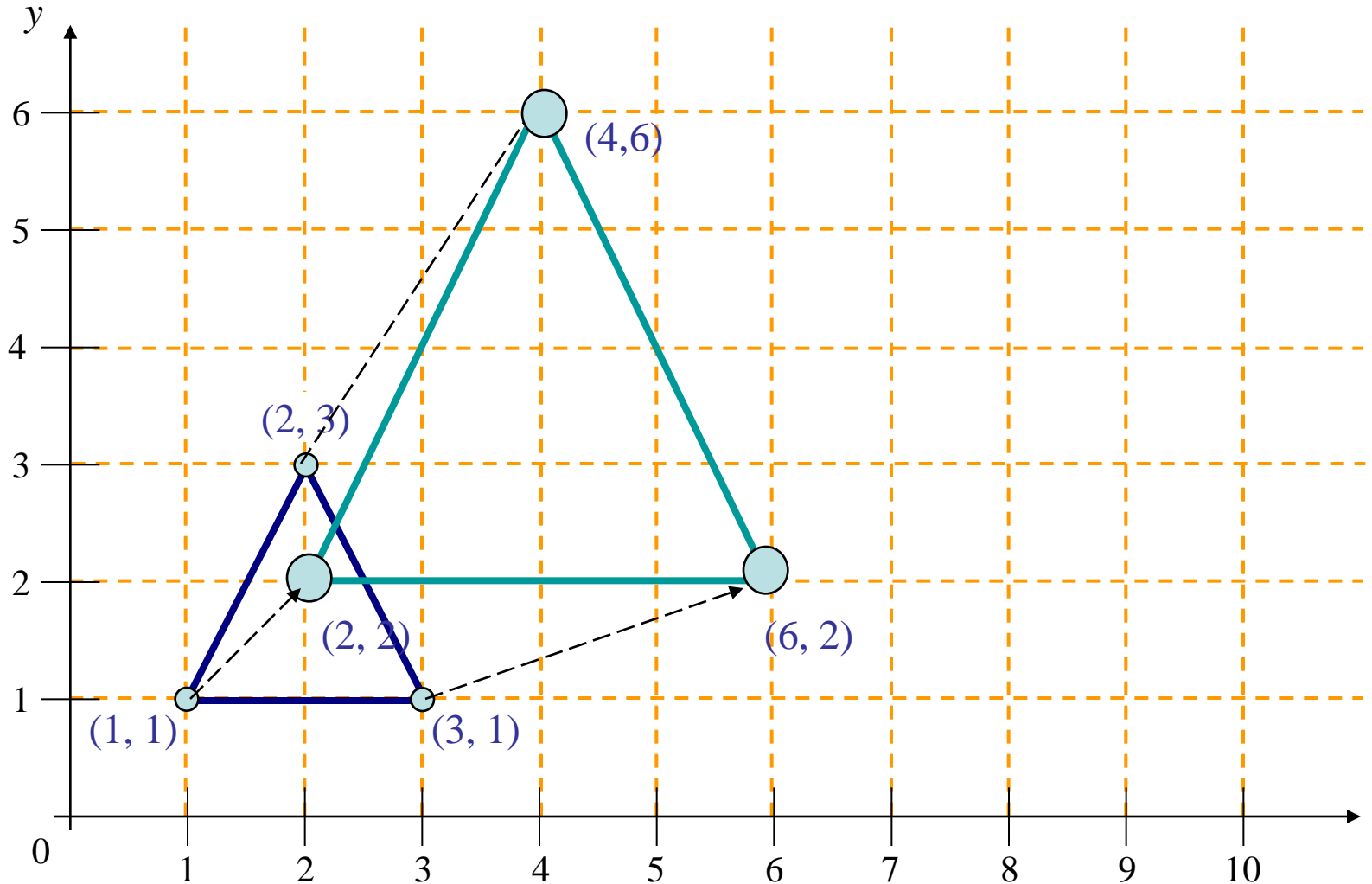
$$P' = S_{s_x, s_y}(P) \quad \text{where}$$

$$S_{s_x, s_y} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \times x \\ s_y \times y \\ 1 \end{bmatrix}$$

Geometric Scaling Example

Scale by $(2, 2)$



Geometric Transformations

–Translation

–Scaling

–**Rotation**

–Reflection

–Shearing

Geometric Rotation

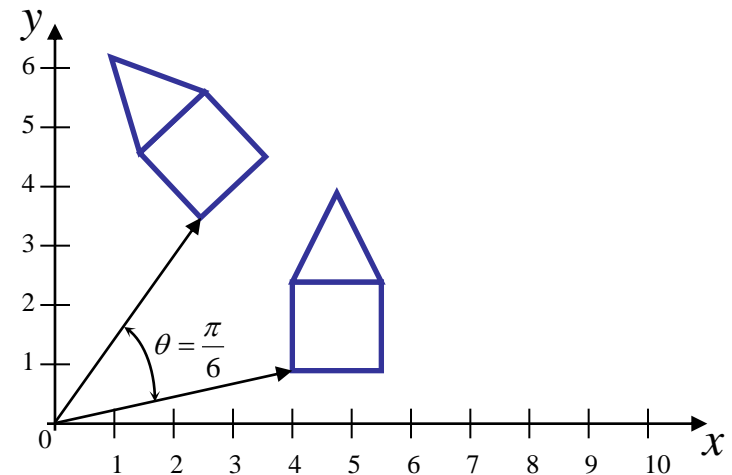
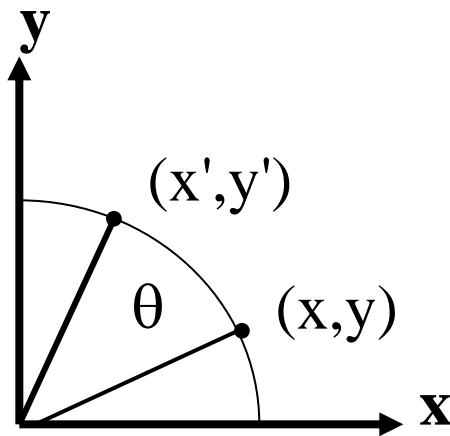
- The rotation of a point $P(x,y)$ *about origin*, by specified angle θ (>0 counter clockwise) can be obtained as

$$x' = x \times \cos\theta - y \times \sin\theta$$

$$y' = x \times \sin\theta + y \times \cos\theta$$

Let us derive these equations

- To rotate an object we have to rotate all coordinates



Geometric Rotation

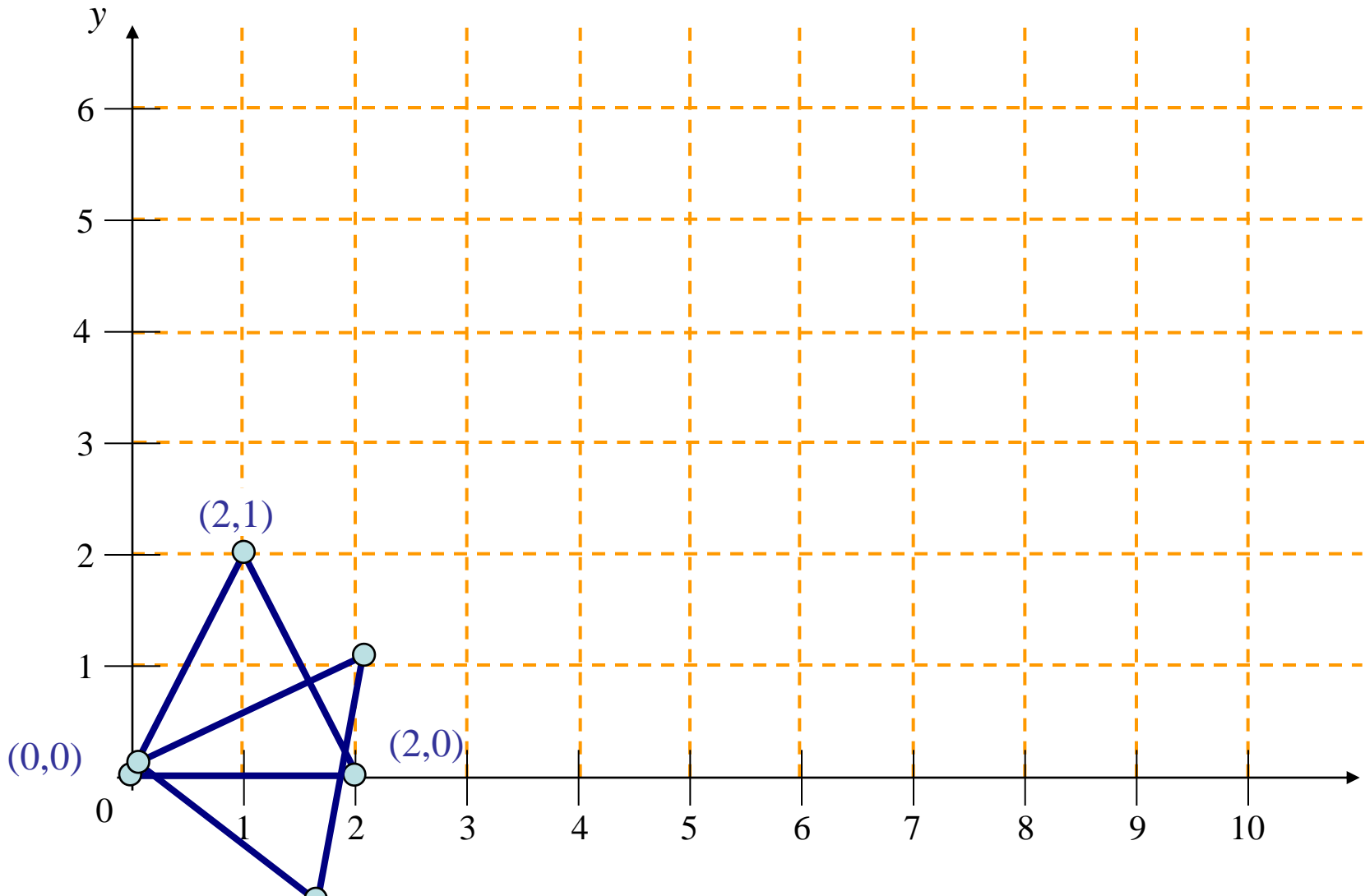
- The rotation of a point $P(x,y)$ by an angle θ about origin can be written in matrix form as:

$$P' = R_{\theta}(P) \quad \text{where}$$

$$R_{\theta} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta \times x - \sin \theta \times y \\ \sin \theta \times x + \cos \theta \times y \\ 1 \end{bmatrix}$$

Geometric Rotation Example



Geometric Transformations

–Translation

–Scaling

–Rotation

–**Reflection**

–Shearing

Geometric Reflection

- Mirror reflection is obtained about X-axis

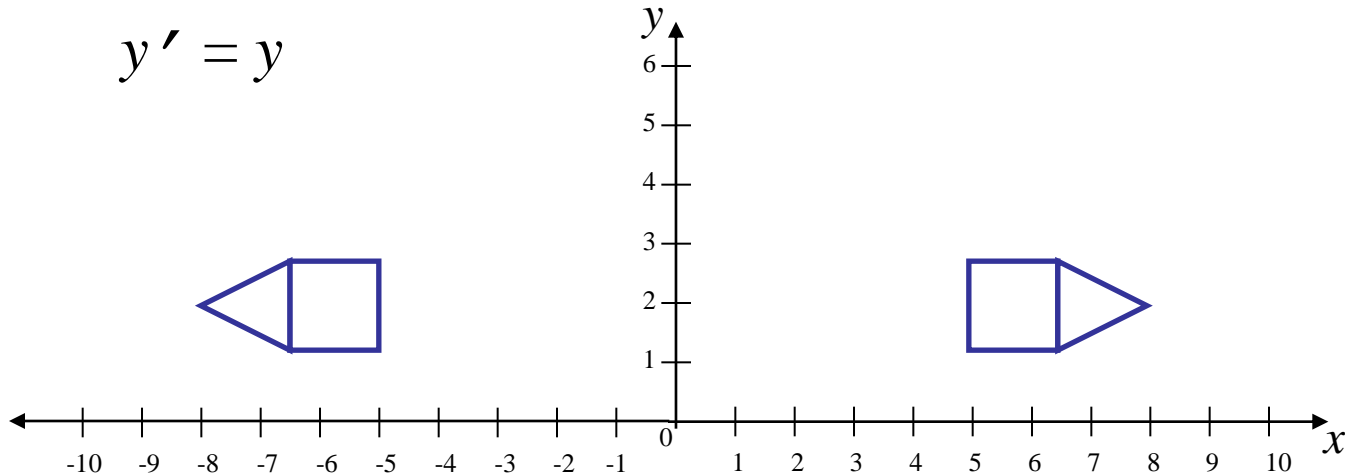
$$x' = x$$

$$y' = -y$$

- Mirror reflection is obtained about Y-axis

$$x' = -x$$

$$y' = y$$



Geometric Reflection

- The reflection of a point $P(x,y)$ about X-axis can be written in matrix form as:

$$P' = M_x(P) \quad \text{where}$$

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ -y \\ 1 \end{bmatrix}$$

Geometric Reflection

- The reflection of a point $P(x,y)$ about Y-axis can be written in matrix form as:

$$P' = M_y(P) \quad \text{where}$$

$$M_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ y \\ 1 \end{bmatrix}$$

Geometric Reflection

- The reflection of a point $P(x,y)$ about origin can be written in matrix form as:

$$P' = M_{xy}(P) \quad \text{where}$$

$$M_{xy} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -x \\ -y \\ 1 \end{bmatrix}$$

Geometric Transformations

–Translation

–Scaling

–Rotation

–Reflection

–**Shearing**

Geometric Shearing

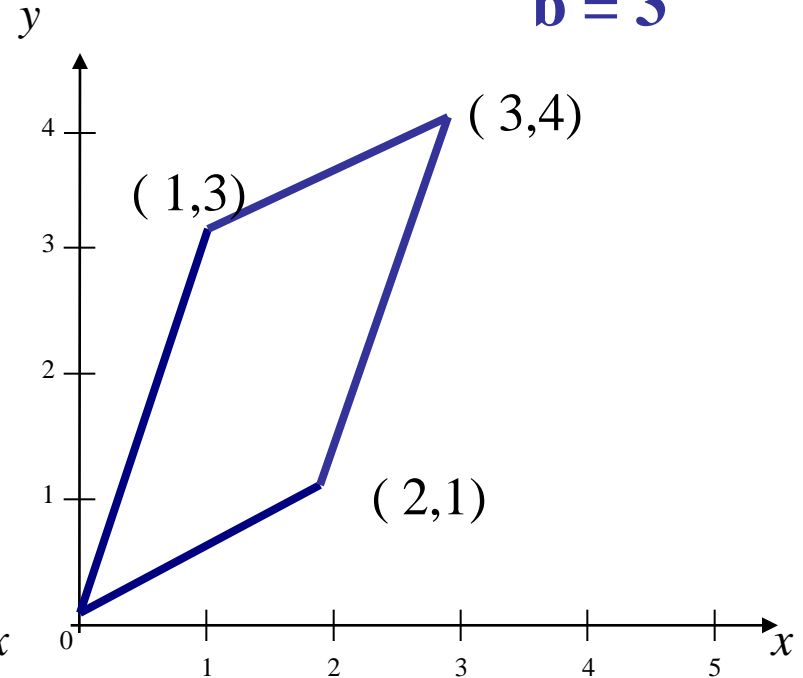
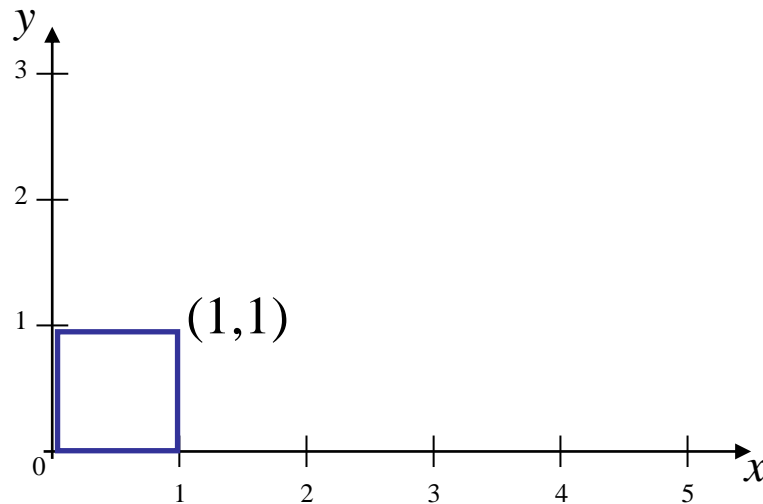
- It is defined as tilting in a given direction
- Shearing about y-axis

$$x' = x + ay$$

$$y' = y + bx$$

$$a = 2$$

$$b = 3$$



Geometric Shearing

- The shearing of a point $P(x,y)$ in general can be written in matrix form as:

$$P' = Sh_{a,b}(P) \quad \text{where}$$

$$Sh_{a,b} = \begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

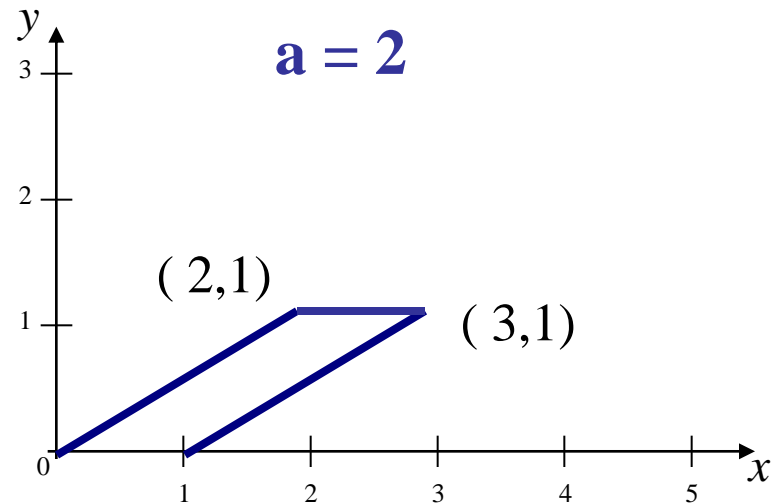
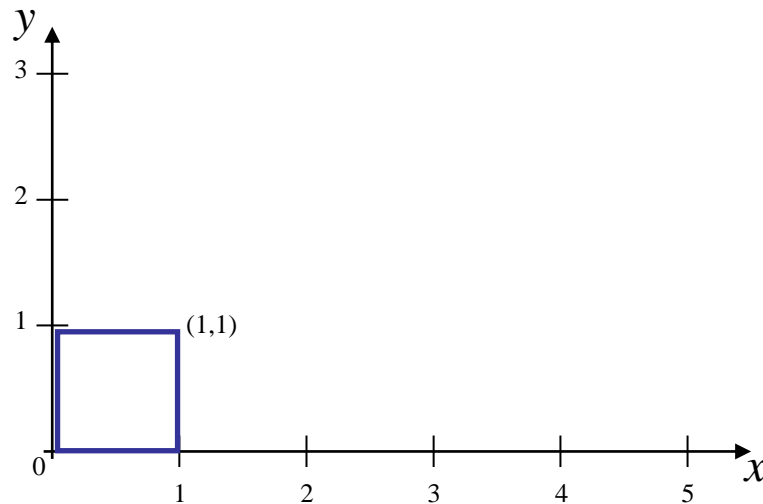
$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + ay \\ y + bx \\ 1 \end{bmatrix}$$

Geometric Shearing

– If $b = 0$ becomes Shearing about X-axis

$$x' = x + ay$$

$$y' = y$$



Geometric Shearing

- The shearing of a point $P(x,y)$ about X-axis can be written in matrix form as:

$$P' = Sh_{a,0}(P) \quad \text{where}$$

$$Sh_{a,0} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

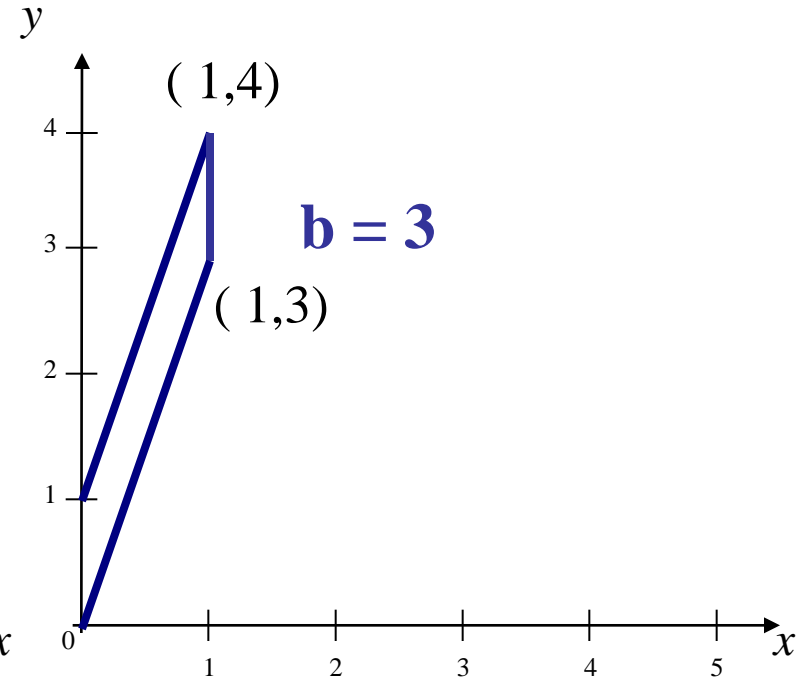
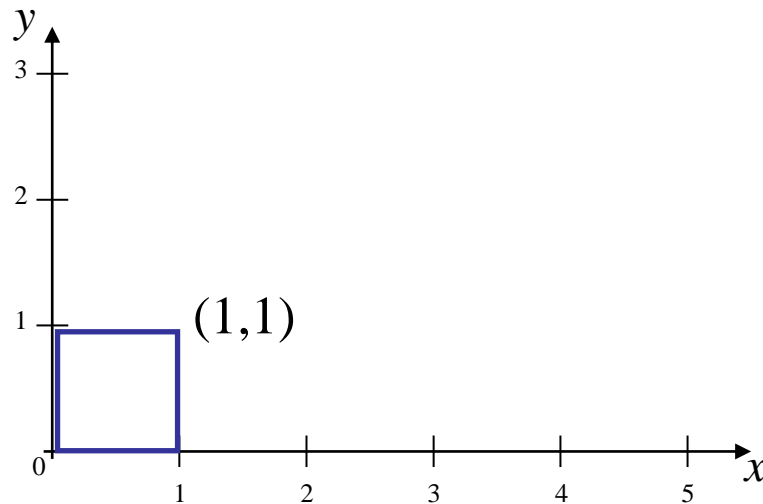
$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + ay \\ y \\ 1 \end{bmatrix}$$

Geometric Shearing

- If $a = 0$ it becomes Shearing about y -axis

$$x' = x$$

$$y' = y + bx$$



Geometric Shearing

- The shearing of a point $P(x,y)$ about Y-axis can be written in matrix form as:

$$P' = Sh_{0,b}(P) \quad \text{where}$$

$$Sh_{0,b} = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{such that} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y + bx \\ 1 \end{bmatrix}$$

Contents

1. Homogeneous coordinates
2. Matrices multiplications
3. Transformations
4. Geometric Transformations
- 5. Inverse Transformations**
6. Coordinate Transformations
7. Composite transformations

Inverse Transformations

- **Inverse Translation:** Displacement in direction of $-V$

$$T_v^{-1} = T_{-v} = \begin{bmatrix} 1 & 0 & -tx \\ 0 & 1 & -ty \\ 0 & 0 & 1 \end{bmatrix}$$

- **Inverse Scaling:** Division by S_x and S_y

$$S_{sx, sy}^{-1} = S_{1/sx, 1/sy} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverse Transformations

- **Inverse Rotation:** Rotation by an angle of $-\theta$

$$R_{\theta}^{-1} = R_{-\theta} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Inverse Reflection:** Reflect once again

$$M_x^{-1} = M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Contents

1. Homogeneous coordinates
2. Matrices multiplications
3. Transformations
4. Geometric Transformations
5. Inverse Transformations
- 6. Coordinate Transformations**
7. Composite transformations

Transformations

- **Geometric Transformations:** In Geometric transformation an object itself is moved relative to a stationary coordinate system or background. The mathematical statement of this view point is described by geometric transformation applied to each point of the object.
- **Coordinate Transformation:** The object is held stationary while coordinate system is moved relative to the object. These can easily be described in terms of the opposite operation performed by Geometric transformation.

Coordinate Transformations

- **Coordinate Translation:** Displacement of the coordinate system origin in direction of $-V$

$$\bar{T}_v = T_{-v} = \begin{bmatrix} 1 & 0 & -tx \\ 0 & 1 & -ty \\ 0 & 0 & 1 \end{bmatrix}$$

- **Coordinate Scaling:** Scaling an object by S_x and S_y or reducing the scale of coordinate system.

$$\bar{S}_{sx,sy} = S_{1/sx,1/sy} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate Transformations

- **Coordinate Rotation:** Rotating Coordinate system by an angle of $-\theta$

$$\overline{R}_\theta = R_{-\theta} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Coordinate Reflection:** Same as Geometric Reflection (why?)

$$\overline{M}_x = M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Contents

1. Homogeneous coordinates
2. Matrices multiplications
3. Transformations
4. Geometric Transformations
5. Inverse Transformations
6. Coordinate Transformations
7. **Composite transformations**

Composite Transformations

- A number of transformations can be combined into one matrix to make things easy
 - Allowed by the fact that we use homogenous coordinates
- Matrix composition works *right-to-left*.

Compose:

$$\mathbf{M} = \mathbf{A} \mathbf{B} \mathbf{C}$$

Then apply it to a point:

$$\mathbf{v}' = \mathbf{M} \mathbf{v}$$

$$\mathbf{v}' = (\mathbf{A} \mathbf{B} \mathbf{C}) \mathbf{v}$$

$$\mathbf{v}' = \mathbf{A} (\mathbf{B} (\mathbf{C} \mathbf{v}))$$

It first applies **C** to **v**, then applies **B** to the result, then applies **A** to the result of that.

Composite Transformations

- Matrix multiplication does NOT *commute*:

$$\mathbf{M N} \neq \mathbf{N M}$$

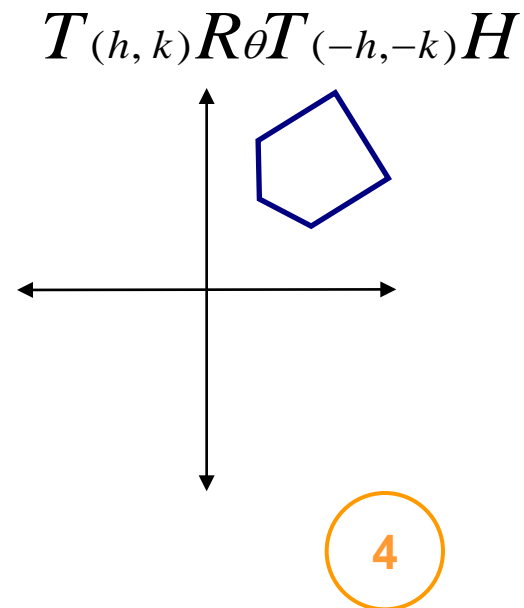
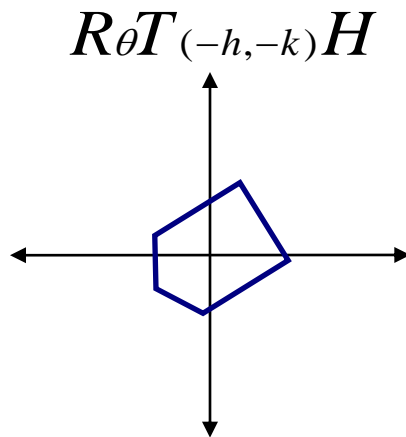
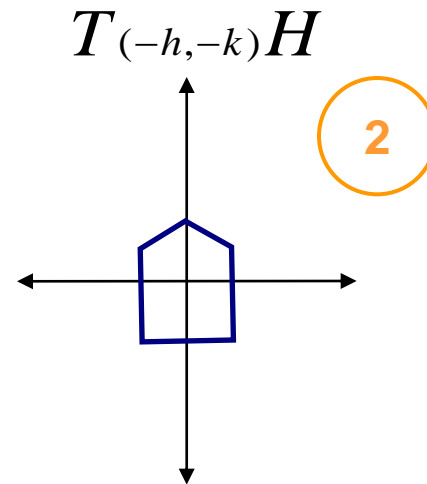
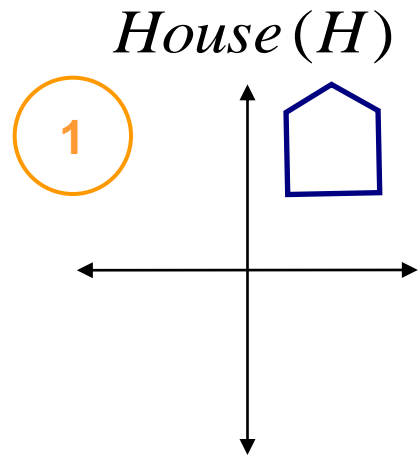
- (unless one or the other is a uniform scale)
- Try this: rotate 90 degrees about x then 90 degrees about y, versus rotate 90 degrees about y then 90 degrees about x.

Composite Transformations

Rotation about Arbitrary Point (h,k)

- Imagine rotating a polygon around a point (h,k) other than the origin
 - Transform to centre point to origin
 - Rotate around origin
 - Transform back to centre point

Composite Transformations



Composite Transformations

- The three transformation matrices are combined as follows

$$\begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T_{(h, k)} R_{\theta} T_{(-h, -k)} (P)$$

REMEMBER: Matrix multiplication is not commutative so order matters

Composite Transformations

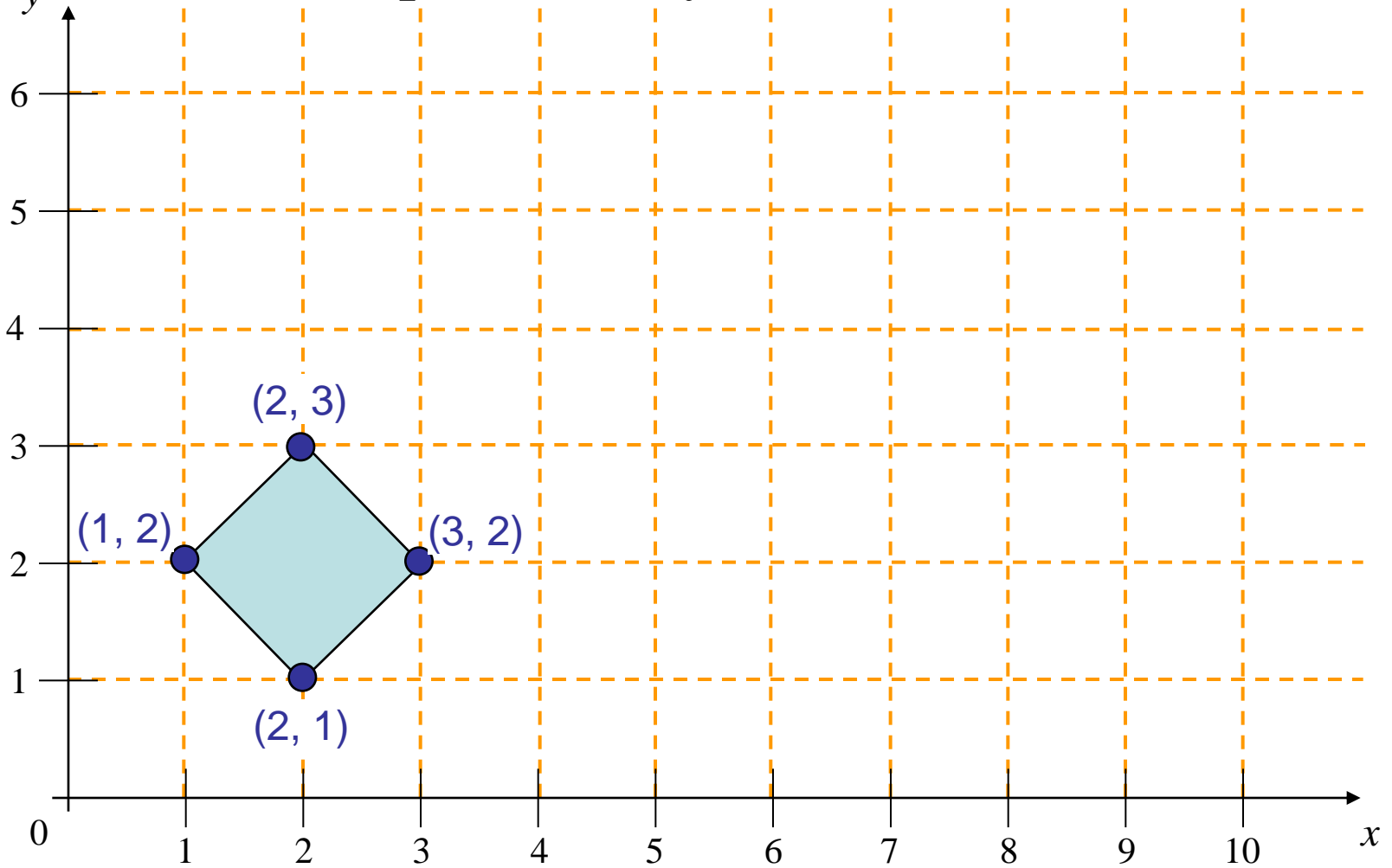
– The composite Transformation is

$$R_{\theta, (h, k)} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos \theta & -\sin \theta & -h \cos \theta + k \sin \theta + h \\ \sin \theta & \cos \theta & -h \sin \theta - k \cos \theta + k \\ 0 & 0 & 1 \end{bmatrix}$$

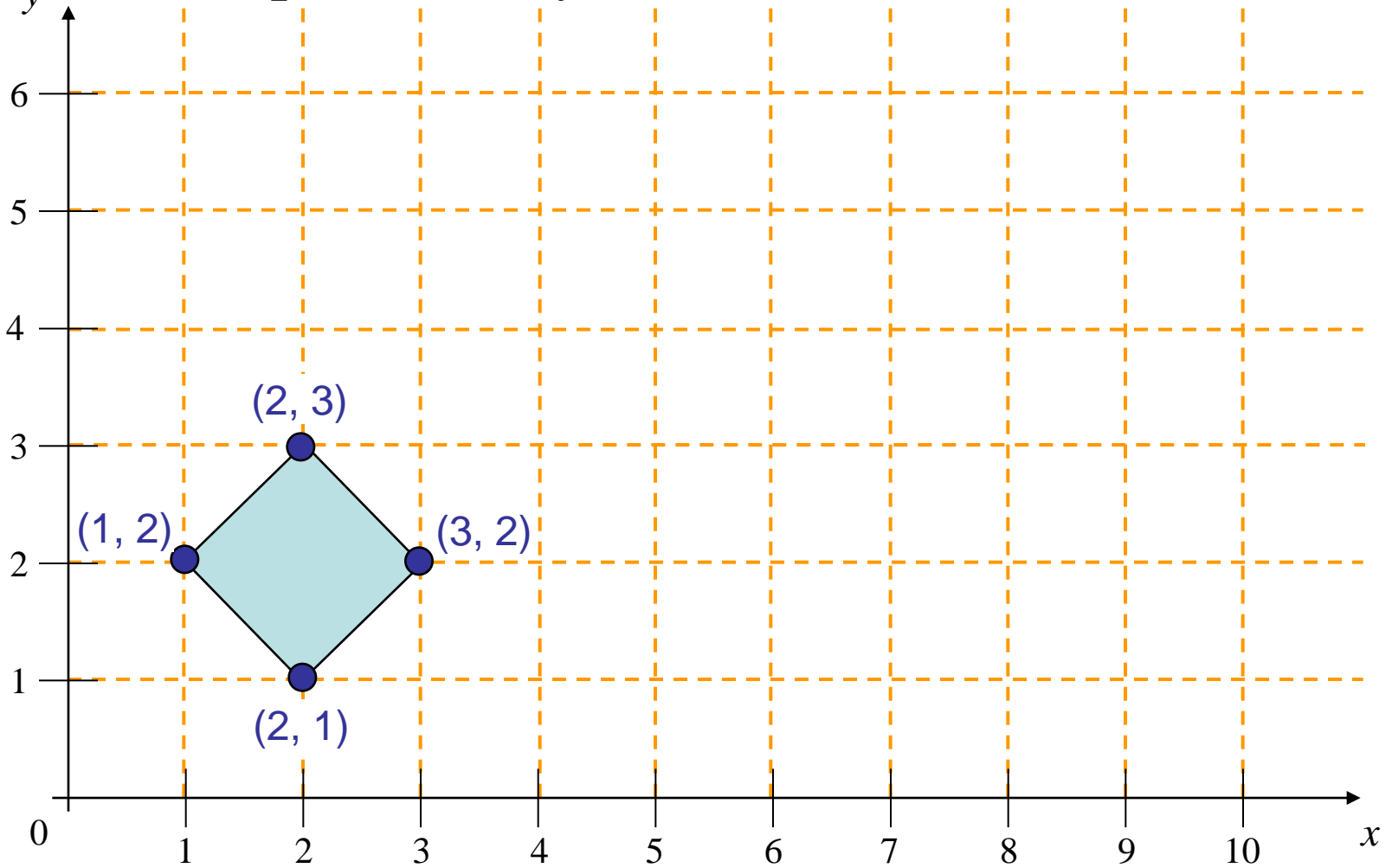
Exercises 1

Translate the shape below by $(7, 2)$



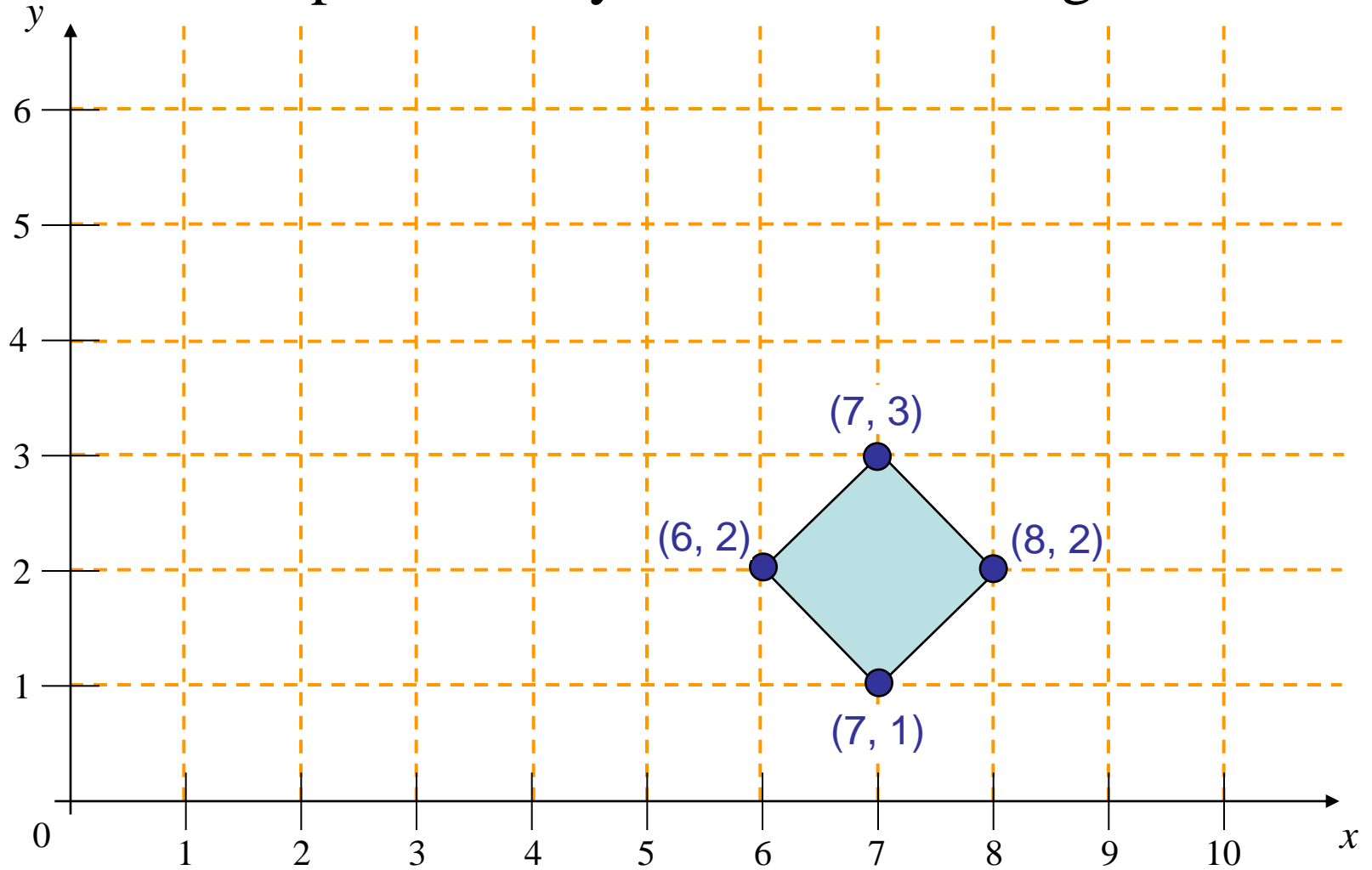
Exercises 2

Scale the shape below by 3 in x and 2 in y



Exercises 3

Rotate the shape below by 30° about the origin



Exercise 4

Write out the homogeneous matrices for the previous three transformations

Translation

$$\begin{bmatrix} _ & _ & _ \\ _ & _ & _ \\ _ & _ & _ \end{bmatrix}$$

Scaling

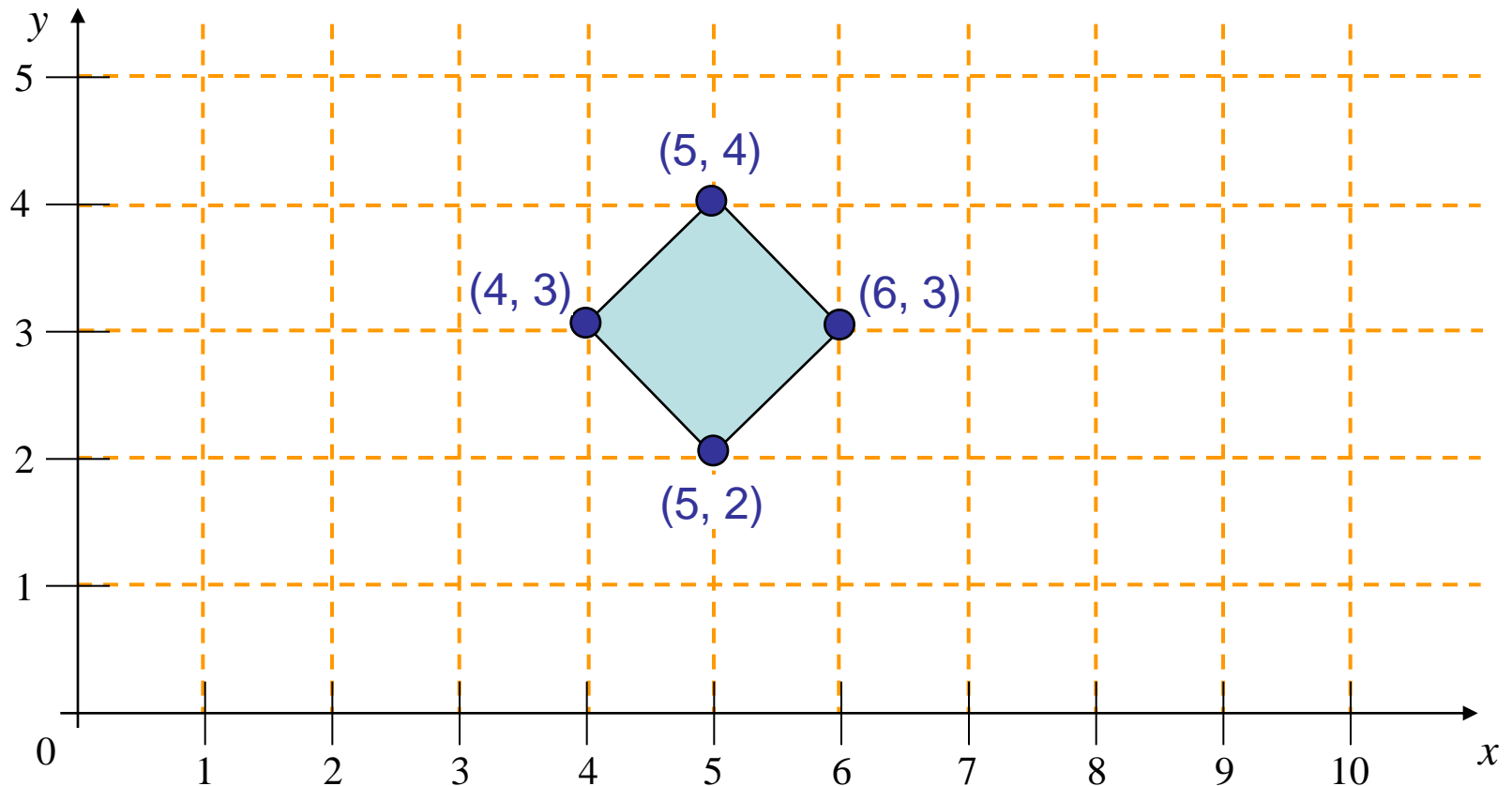
$$\begin{bmatrix} _ & _ & _ \\ _ & _ & _ \\ _ & _ & _ \end{bmatrix}$$

Rotation

$$\begin{bmatrix} _ & _ & _ \\ _ & _ & _ \\ _ & _ & _ \end{bmatrix}$$

Exercises 5

Using matrix multiplication calculate the rotation of the shape below by 45° about its centre $(5, 3)$



Exercise 6

Rotate a triangle ABC A(0,0), B(1,1), C(5,2) by 45°

1. About origin (0,0)
2. About P(-1,-1)

$$R_{45} = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[ABC] = \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$R_{45, (-1, -1)} = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & -1 \\ \sqrt{2}/2 & \sqrt{2}/2 & \sqrt{2}-1 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 7

Magnify a triangle ABC A(0,0), B(1,1), C(5,2) twice keeping point C(5,2) as fixed.

$$[ABC] = \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$S_{2,2,(5,2)} = \begin{bmatrix} 2 & 0 & -5 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[A'B'C'] = \begin{bmatrix} -5 & -3 & 5 \\ -2 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Exercise 8

Describe transformation M_L which reflects an object about a
Line L: $y=m*x+b$.

$$M_L = \begin{bmatrix} \frac{1-m^2}{1+m^2} & \frac{2m}{1+m^2} & \frac{-2bm}{1+m^2} \\ \frac{2m}{1+m^2} & \frac{m^2-1}{1+m^2} & \frac{2b}{1+m^2} \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 9

Reflect the diamond shaped polygon whose vertices are A(-1,0) B(0,-2) C(1,0) and D(0,2) about

1. Horizontal Line $y=2$

2. Vertical Line $x = 2$

3. Line L: $y=x+2$.

$$M_{y = x + 2} = \begin{bmatrix} 0 & 1 & -2 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{y = 2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{x = 2} = \begin{bmatrix} -1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 10

Obtain reflection about Line $y = x$

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

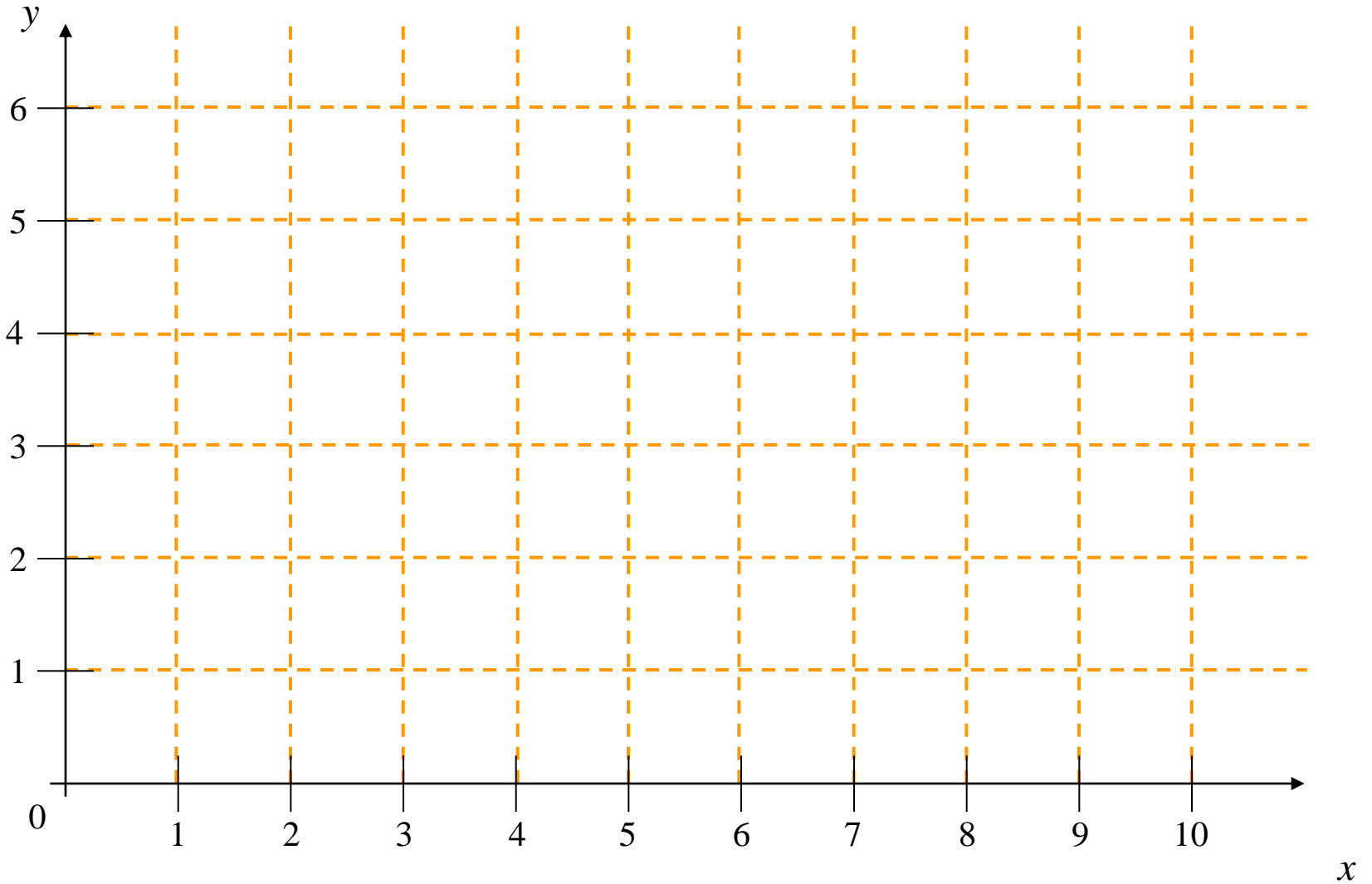
Exercise 11

Prove that

1. Two successive translations are additive /commutative.
2. Two successive rotations are additive /commutative.
3. Two successive Scaling are multiplicative /commutative.
4. Two successive reflections are nullified /Invertible.

Is Translation followed by Rotation equal to Rotation followed by translation ?

Scratch



**2D Viewing Transformation
&
2D Clipping**

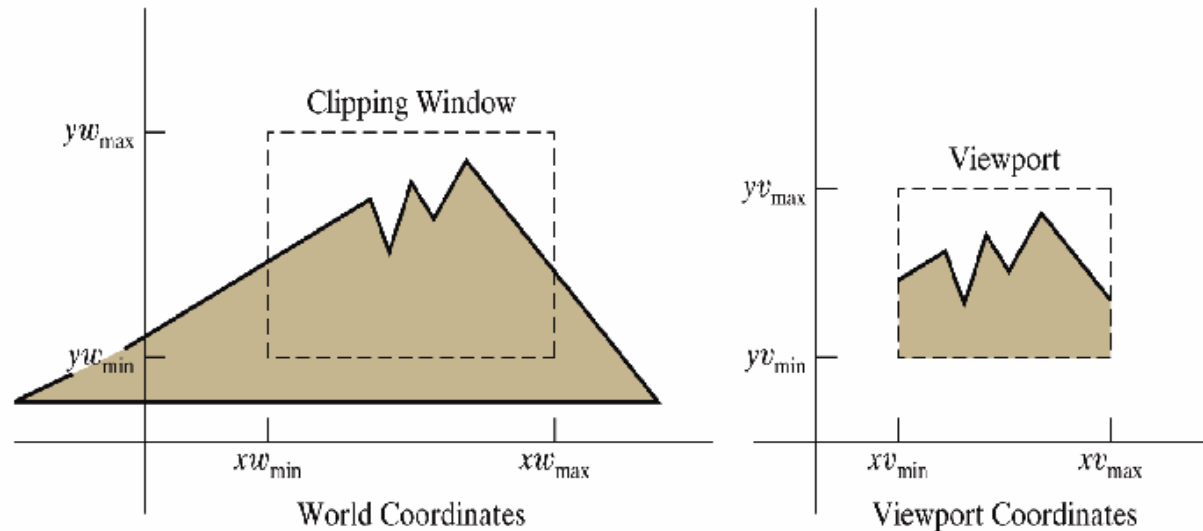
2D Viewing Transformation

1. **Definition**
2. 2D Viewing Pipeline
3. Approaches
4. Aspect Ratio

2D Viewing Transformation

1. Definitions: It is defined as a process for displaying views of a two-dimensional picture on an output device:

- Specify which parts of the object to display (clipping window, or world window, or viewing window)
- Where



2D Viewing Transformation

- **World Co – Ordinate System** is a right handed Cartesian coordinate system in which picture is actually defined.
- **Physical Device Co – Ordinate System** is a coordinate system that correspond to output device or work stations where image to be displayed. E.g. with our monitor it is a Left handed Cartesian coordinate system.
- **Normalized Co – Ordinate System** It is a right handed coordinate system in which display area of virtual display device correspond to the unit square (1x1).

2D Viewing Transformation

- **Window** or Clipping window is the selected section of a scene that is displayed on a display window. It is a finite region from World Coordinate System.
- **View port** is the window where the object is viewed on the output device. It is a finite region from Device Coordinate System.

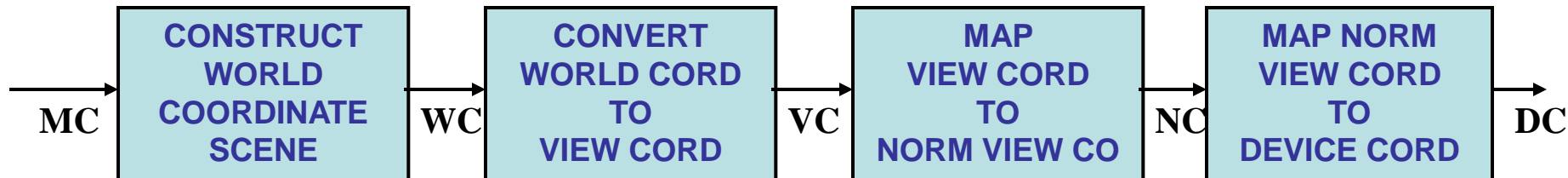
2D Viewing Transformation

1. Definition
- 2. 2D Viewing Pipeline**
3. Approaches
4. Aspect Ratio

2D Viewing Transformation

2. 2D viewing pipeline

- Construct world-coordinate scene using modeling-coordinate transformations
- Convert world-coordinates to viewing coordinates
- Transform viewing-coordinates to normalized-coordinates (ex: between 0 and 1, or between -1 and 1)
- Map normalized-coordinates to device-coordinates.



2D Viewing Transformation

1. Definition
2. 2D Viewing Pipeline
- 3. Approaches**
4. Aspect Ratio

2D Viewing Transformation

3. Approaches

- Two main approaches to 2D viewing Transformation are

1. **Direct Approach**

2. Normalized Approach

2D Viewing Transformation

3.1. Direct Approach

- Mapping the clipping window into a view port which may be normalized
- Its involves
 - translating the origin of the clipping window to that of the view port
 - scaling the clipping window to the size of the view port
- We can drive it by three methods

2D Viewing Transformation

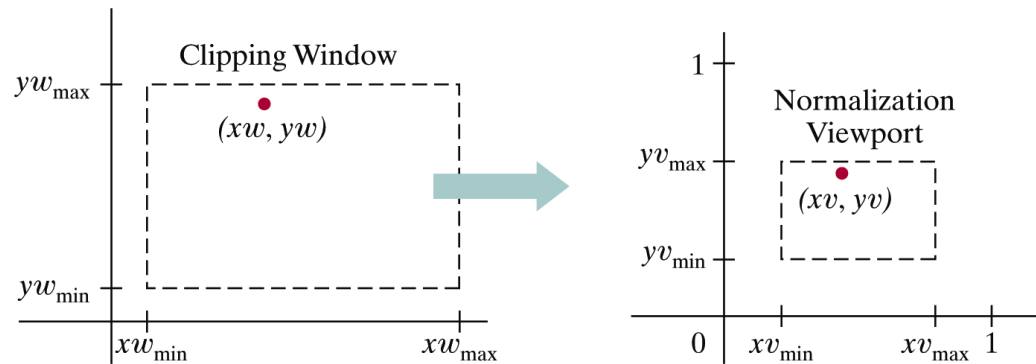


Figure 6-7

A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv) , within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

(from Donald Hearn and Pauline Baker)

2D Viewing Transformation

Method 1: Let $P(x_w, y_w)$ be any point in the window, which is to be mapped to $P'(x_v, y_v)$ in the associated view port. The two steps required are:

1. Translation T_v that takes $(x_{w_{\min}}, y_{w_{\min}})$ to $(x_{v_{\min}}, y_{v_{\min}})$, where
$$v = (x_{v_{\min}} - x_{w_{\min}})I + (y_{v_{\min}} - y_{w_{\min}})J$$

2. Scaling by following scaling factors about point $(x_{v_{\min}}, y_{v_{\min}})$

$$s_x = \frac{x_{v_{\max}} - x_{v_{\min}}}{x_{w_{\max}} - x_{w_{\min}}} \quad \text{and} \quad s_y = \frac{y_{v_{\max}} - y_{v_{\min}}}{y_{w_{\max}} - y_{w_{\min}}}$$

2D Viewing Transformation

$$\begin{aligned} V &= S_{s_x, s_y, (xv_{\min}, yv_{\min})} \cdot T_v \\ &= \begin{bmatrix} s_x & 0 & s_x - xv_{\min} + xv_{\min} \\ 0 & s_y & s_y - yv_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & xv_{\min} - xw_{\min} \\ 0 & 1 & yv_{\min} - yw_{\min} \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} s_x & 0 & -s_x * xw_{\min} + xv_{\min} \\ 0 & s_y & -s_y * yw_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Method

2D Viewing Transformation

Method 2: We get the same result if we perform

$$\begin{aligned} V &= T_{(xv_{\min}, yv_{\min})} S_{s_x, s_y} \cdot T_{(-xw_{\min}, -yw_{\min})} \\ &= \begin{bmatrix} 1 & 0 & xv_{\min} \\ 0 & 1 & yv_{\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xw_{\min} \\ 0 & 1 & -yw_{\min} \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} s_x & 0 & -s_x * xw_{\min} + xv_{\min} \\ 0 & s_y & -s_y * yw_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

2D Viewing Transformation

Method 3: Let P (x_w, y_w) be any point in the window, which is to be mapped to P' (x_v, y_v) in the associated view port. To maintain the same relative placement in the view port as in window, we require that

$$\frac{x_v - x_{v_{\min}}}{x_{v_{\max}} - x_{v_{\min}}} = \frac{x_w - x_{w_{\min}}}{x_{w_{\max}} - x_{w_{\min}}} \quad \text{and}$$

$$\frac{y_v - y_{v_{\min}}}{y_{v_{\max}} - y_{v_{\min}}} = \frac{y_w - y_{w_{\min}}}{y_{w_{\max}} - y_{w_{\min}}}$$

2D Viewing Transformation

Which means

$$xv = xv_{\min} + (xw - xw_{\min}) * \frac{(xv_{\max} - xv_{\min})}{xw_{\max} - xw_{\min}} \quad \text{and}$$

$$yv = yv_{\min} + (yw - yw_{\min}) * \frac{(yv_{\max} - yv_{\min})}{yw_{\max} - yw_{\min}}$$

Put these equations in the matrix form.

2D Viewing Transformation

3.2 Normalized Approach

- Mapping the clipping window into a normalized square
- It involves
 - transforming the clipping window into a normalized square
 - Then clipping in normalized coordinates (ex: -1 to 1) or (ex: 0 to 1)
 - Then transferring the scene description to a view port specified in screen coordinates.
 - Finally positioning the view port area in the display window.
- Thus $V = W.N$, where N is a transformation that maps window to normalized view port and W maps Normalized points to view port.

2D Viewing Transformation

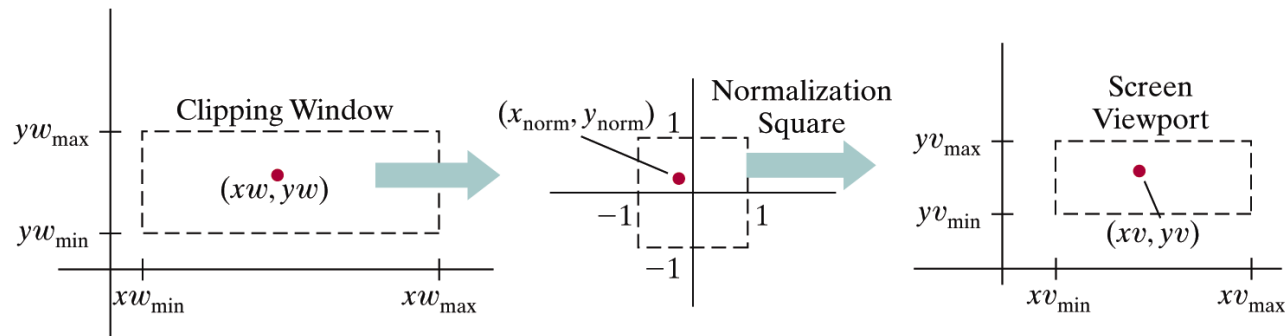


Figure 6-8

A point (x_w, y_w) in a clipping window is mapped to a normalized coordinate position $(x_{\text{norm}}, y_{\text{norm}})$, then to a screen-coordinate position (x_v, y_v) in a viewport. Objects are clipped against the normalization square before the transformation to viewport coordinates.

(from Donald Hearn and Pauline Baker)

2D Viewing Transformation

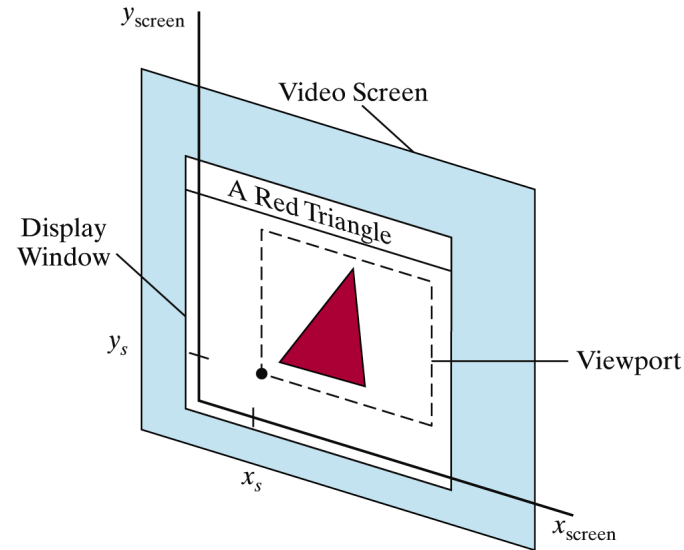


Figure 6-9

A viewport at coordinate position (x_s, y_s) within a display window.

(from Donald Hearn and Pauline Baker)

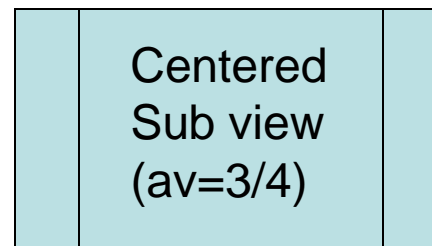
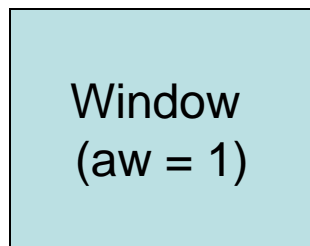
2D Viewing Transformation

1. Definition
2. 2D Viewing Pipeline
3. Approaches
4. **Aspect Ratio**

2D Viewing Transformation

4. Aspect Ratio

- Since viewing involves scaling, so undesirable distortions may be introduced when $s_x \neq s_y$
- Aspect ratio is defined as $(y_{\max} - y_{\min}) / (x_{\max} - x_{\min})$
- If $a_w = a_v \Rightarrow$ no distortion
- If $a_w > a_v \Rightarrow$ Horizontal spanning
- If $a_w < a_v \Rightarrow$ Vertical spanning



Exercise 1

Find the normalization transformation that maps a window defined by (0,0) to (3,5) on to

1. The view port that is entire normalized device.

$$V = \begin{bmatrix} 1/2 & 0 & -1/2 \\ 0 & 1/4 & -1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

2. Has lower left corner (0,0) and upper right corner as (1/2,1/2)

$$V = \begin{bmatrix} 1/4 & 0 & -1/4 \\ 0 & 1/8 & -1/8 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 2

Find the complete viewing transformation that

1. First maps a window defined by (1,1) to (10,10) on to a view port of size(1/4,0) to (3/4,1/2) in normalized device space
2. Then maps a window of (1/4,1/4) to (1/2,1/2) in normalized device space to view port of (0,0) to (10,10)

$$N = \begin{bmatrix} 1/18 & 0 & 7/36 \\ 0 & 1/18 & -1/18 \\ 0 & 0 & 1 \end{bmatrix} \quad W = \begin{bmatrix} 36 & 0 & -8 \\ 0 & 36 & -8 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V = W.N = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 3

Find the normalization transformation that maps a window defined by (0,0) to (4,3) on to normalized device screen keeping aspect ratio preserved.

Sol: $aw = 3/4$

$$av = 1$$

In normalized device we will keep
x extent 0 to 1 and y extent 0 to $3/4$

$$S_x = (1-0) / (4-0) = 1/4$$

$$S_y = (3/4 - 0) / (3-0) = 1/4$$

$$N = \begin{bmatrix} 1/4 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 4

Find the normalization transformation that maps a window defined by A(1,1) B(5,3) C(4,5) and D(0,3) on to normalized device screen.

$$\begin{aligned} V &= N \cdot R_{-\theta, (1,1)} \\ &= \begin{bmatrix} 1/2\sqrt{5} & 0 & -1/2\sqrt{5} \\ 0 & 1/\sqrt{5} & -1/\sqrt{5} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} & 1-3/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} & 1-1/\sqrt{5} \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1/5 & 1/10 & -3/10 \\ -1/5 & 2/5 & -1/10 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

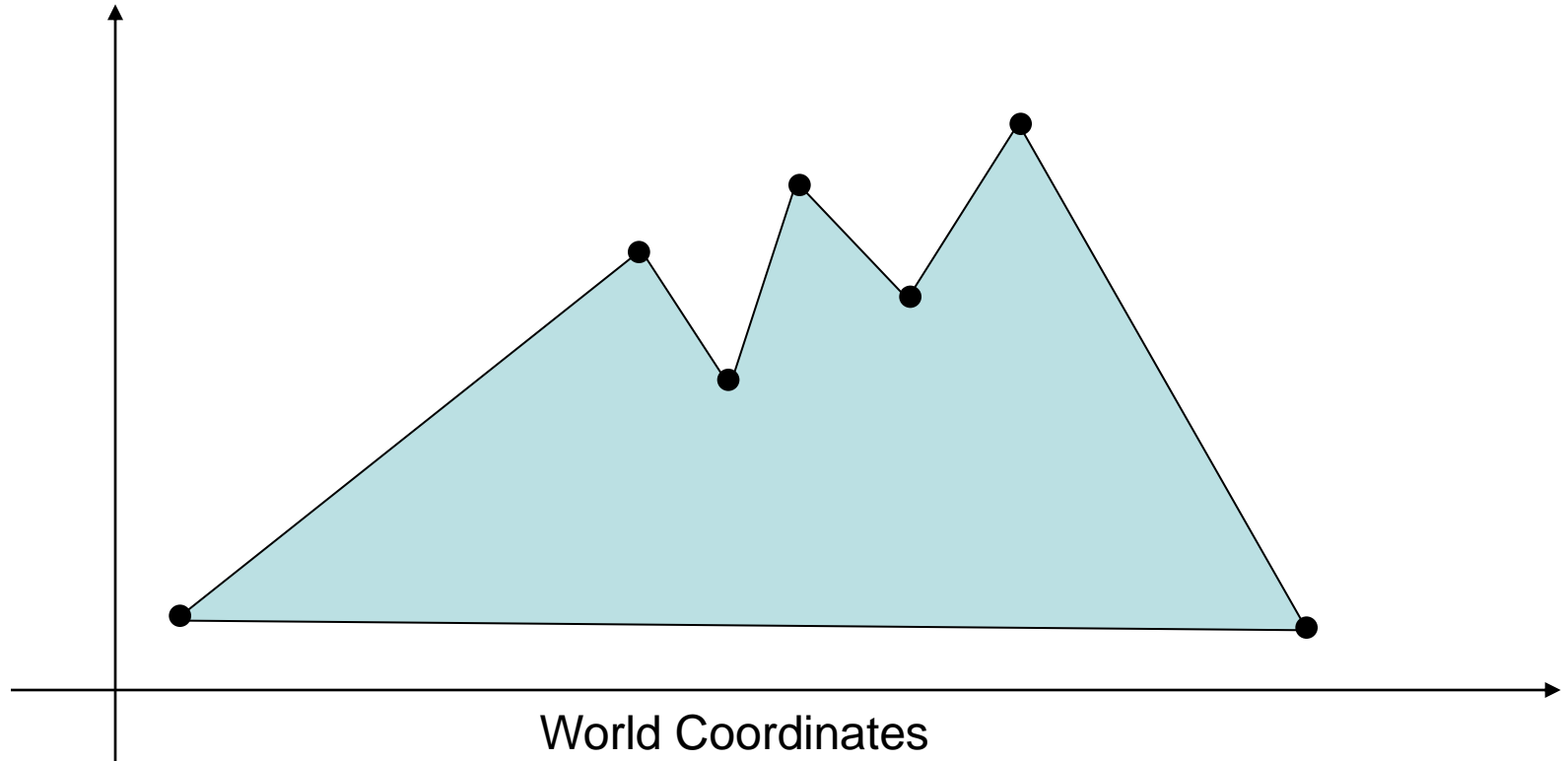
2D Clipping

- 1. Introduction**
2. Point Clipping
3. Line Clipping
4. Polygon/Area Clipping
5. Text Clipping

2D Clipping

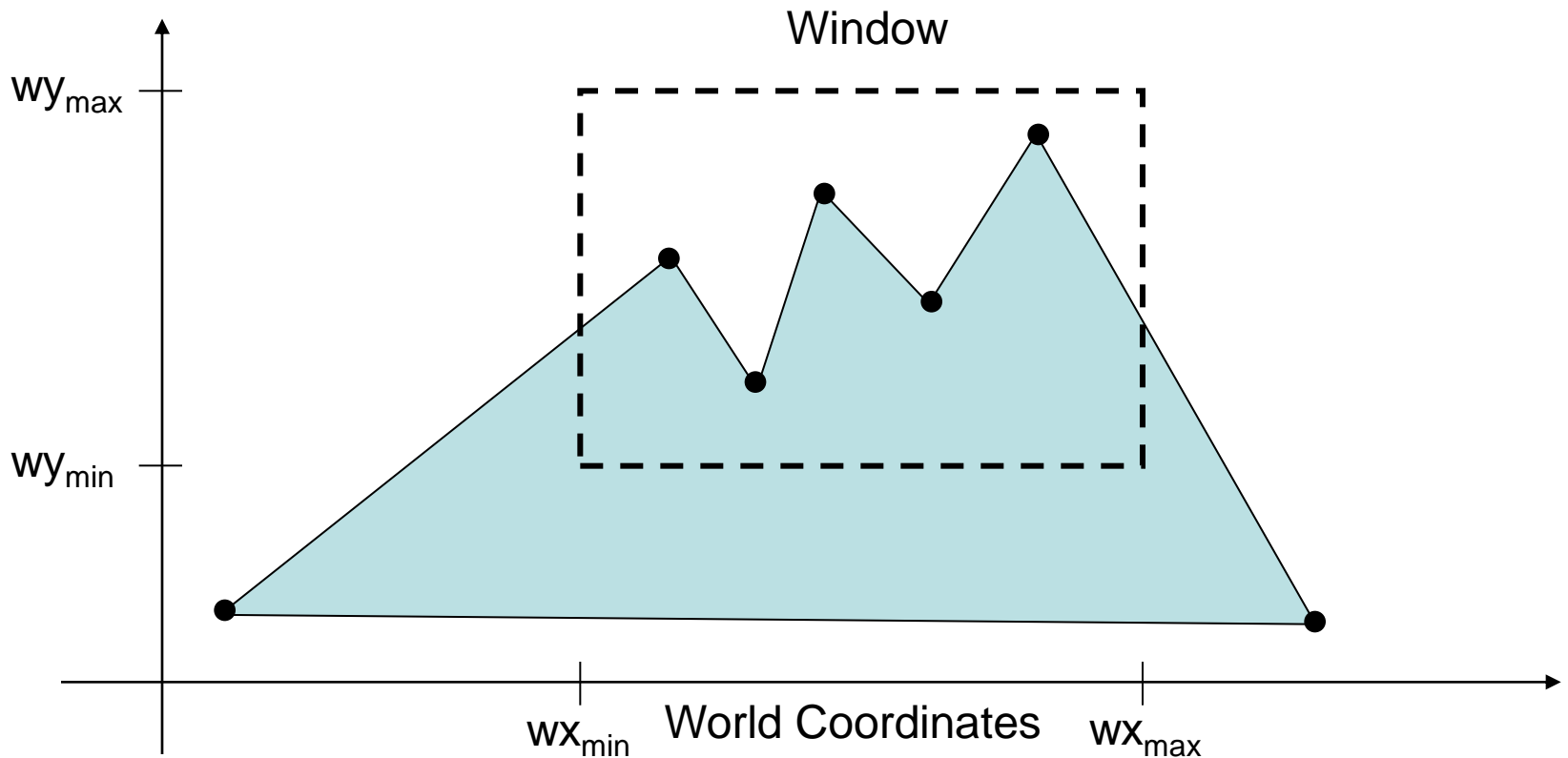
1. Introduction:

A scene is made up of a collection of objects specified in world coordinates



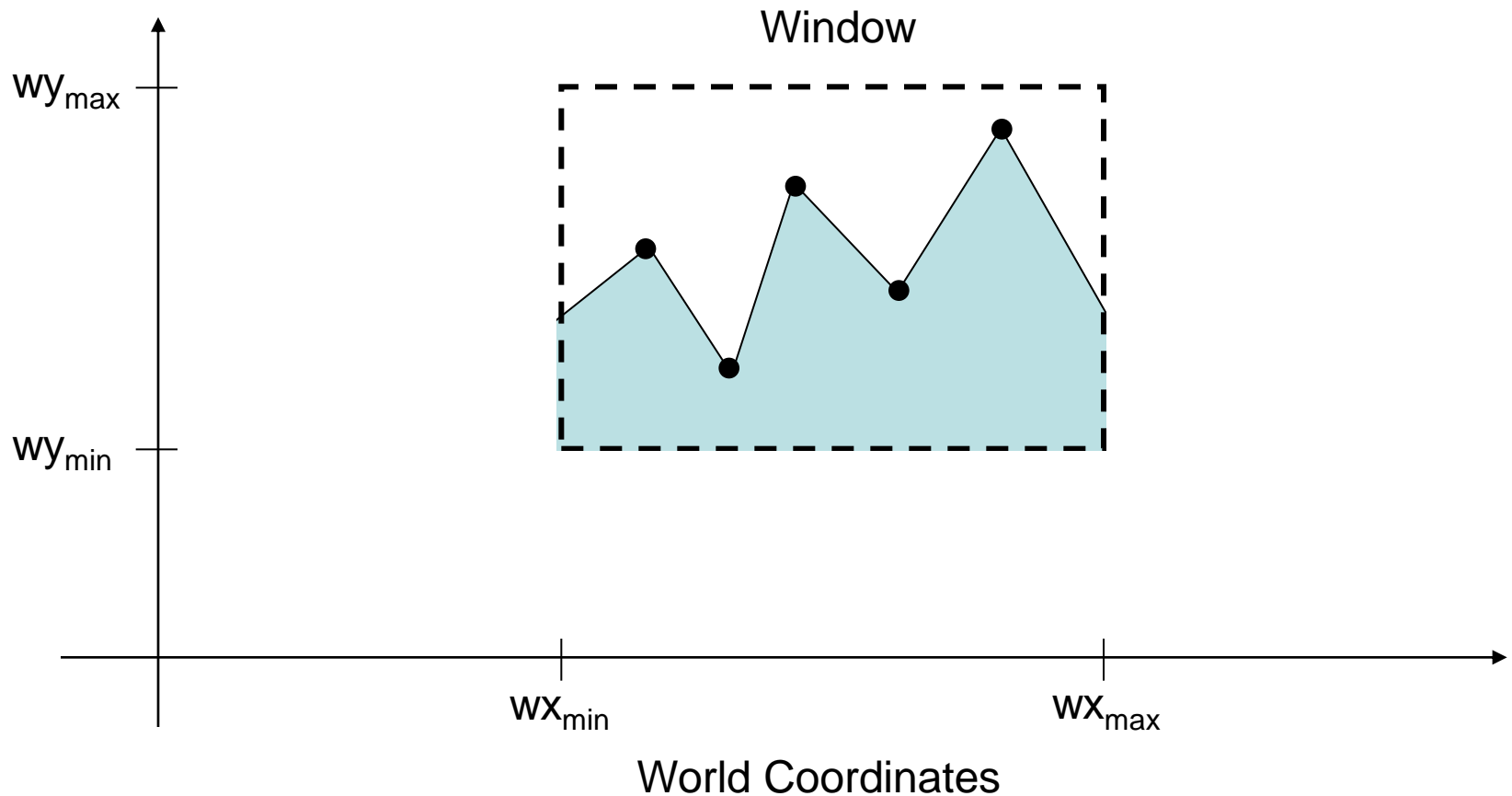
2D Clipping

When we display a scene only those objects within a particular window are displayed



2D Clipping

Because drawing things to a display takes time we *clip* everything outside the window



2D Clipping

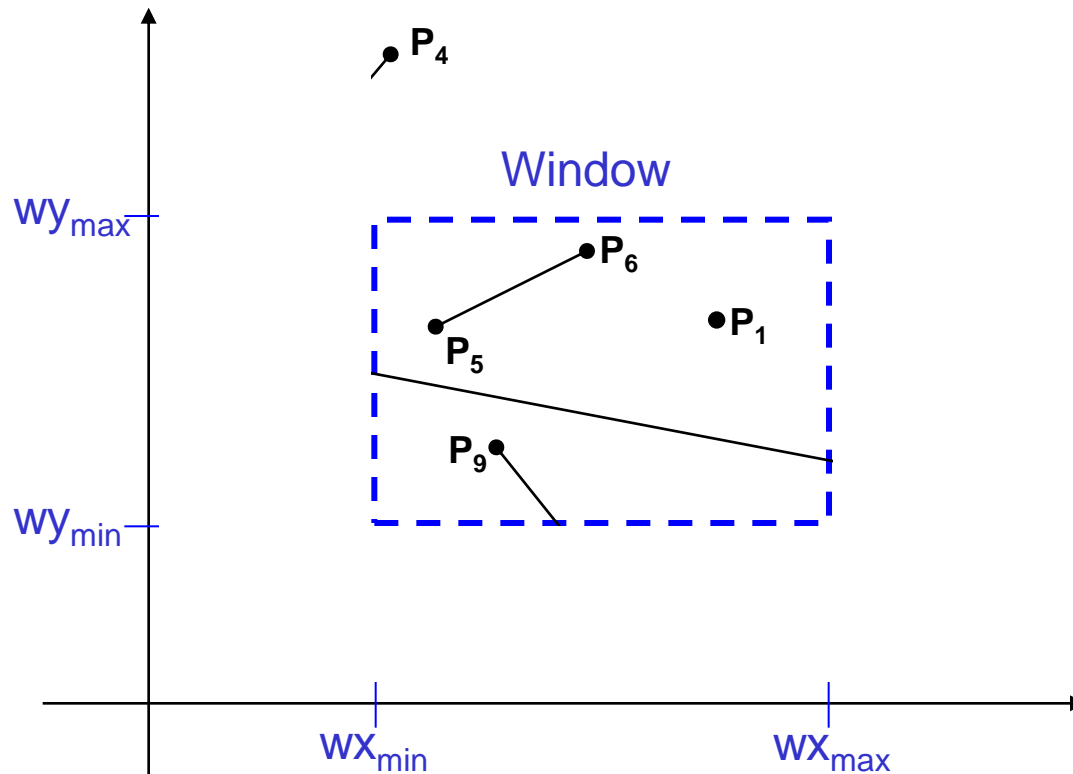
1.1 Definitions:

- *Clipping* is the process of determining which elements of the picture lie inside the window and are visible.
- *Shielding* is the reverse operation of clipping where window act as the block used to abstract the view.
- By default, the “*clip window*” is the entire canvas
 - not necessary to draw outside the canvas
 - for some devices, it is damaging (plotters)
- Sometimes it is convenient to restrict the “clip window” to a smaller portion of the canvas
 - partial canvas redraw for menus, dialog boxes, other obscuration

2D Clipping

1.2 Example:

For the image below consider which lines and points should be kept and which ones should be clipped against the clipping window



2D Clipping

1.3 Applications:

- Extract part of a defined scene for viewing.
- Drawing operations such as erase, copy, move etc.
- Displaying multi view windows.
- Creating objects using solid modeling techniques.
- Anti-aliasing line segments or object boundaries.
- Identify visible surfaces in 3D views.

2D Clipping

1.4 Types of clipping:

- Three types of clipping techniques are used depending upon when the clipping operation is performed

a. Analytical clipping

- Clip it before you scan convert it
- used mostly for lines, rectangles, and polygons, where clipping algorithms are simple and efficient

2D Clipping

b. Scissoring

- Clip it during scan conversion
- a brute force technique
 - scan convert the primitive, only write pixels if inside the clipping region
 - easy for thick and filled primitives as part of scan line fill
 - if primitive is not much larger than clip region, most pixels will fall inside
 - can be more efficient than analytical clipping.

2D Clipping

c. Raster Clipping

- Clip it after scan conversion
- render everything onto a temporary canvas and copy the clipping region
 - wasteful, but simple and easy,
 - often used for text

2D Clipping

Foley and van Dam suggest the following:

- for floating point graphics libraries, try to use analytical clipping
- for integer graphics libraries
 - analytical clipping for lines and polygons
 - others, do during scan conversion
- sometimes both analytical and raster clipping performed

2D Clipping

1.5 Levels of clipping:

- Point Clipping
- Line Clipping
- Polygon Clipping
- Area Clipping
- Text Clipping
- Curve Clipping

2D Clipping

1. Introduction
- 2. Point Clipping**
3. Line Clipping
4. Polygon/Area Clipping
5. Text Clipping

Point Clipping

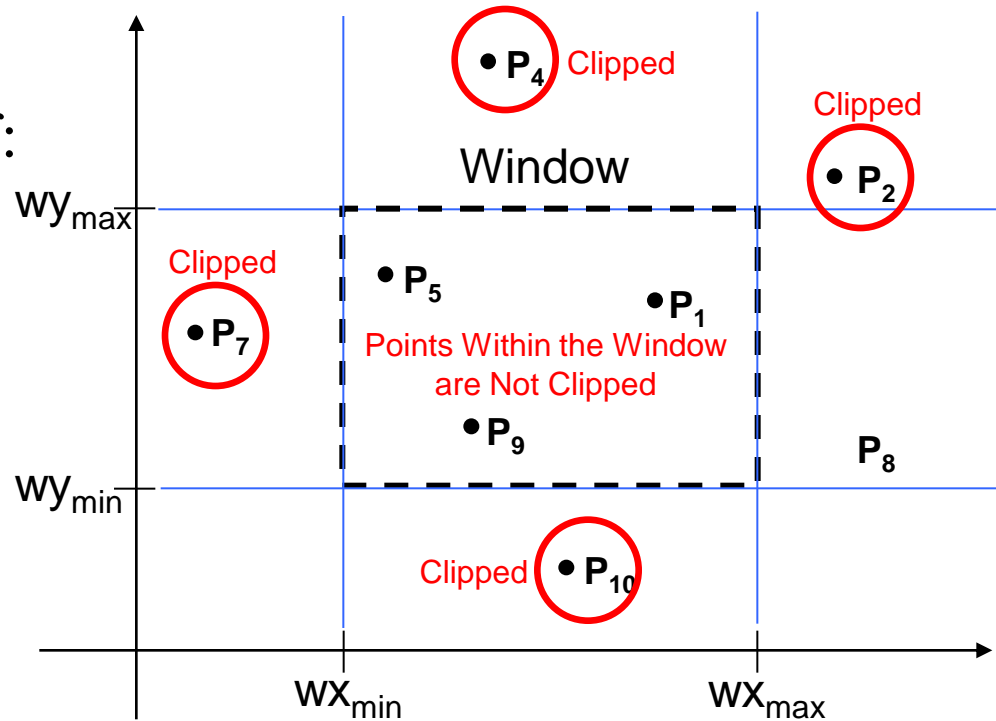
- Simple and Easy
- a point (x,y) is not clipped if:

$$wx_{min} \leq x \leq wx_{max}$$

&

$$wy_{min} \leq y \leq wy_{max}$$

- otherwise it is clipped

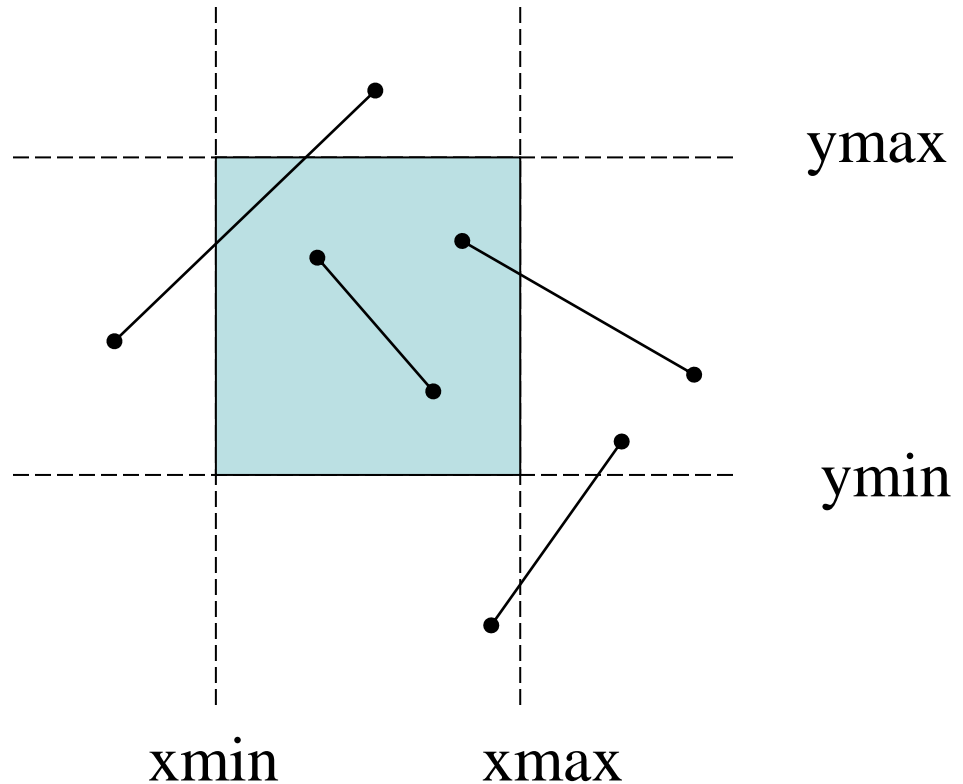


2D Clipping

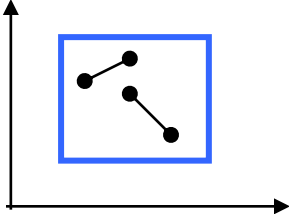
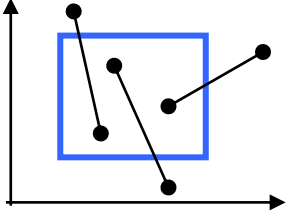
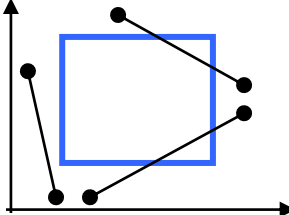
1. Introduction
2. Point Clipping
3. **Line Clipping**
4. Polygon/Area Clipping
5. Text Clipping

Line Clipping

- It is Harder than point clipping
- We first examine the endpoints of each line to see if they are in the window or not
 - Both endpoints inside, line trivially accepted
 - One in and one out, line is partially inside
 - Both outside, might be partially inside
 - What about trivial cases?



Line Clipping

Situation	Solution	Example
Both end-points inside the window	Don't clip	
One end-point inside the window, one outside	Must clip	
Both end-points outside the window	Don't know!	

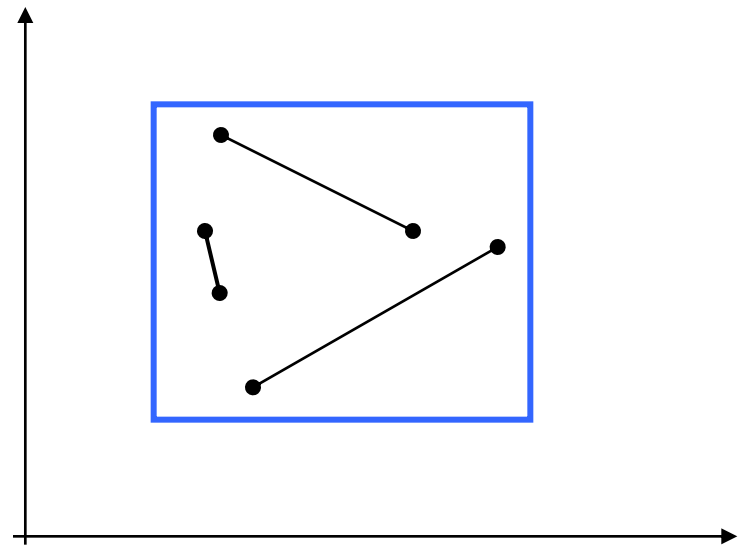
2D Line Clipping Algorithms

1. **Analytical Line Clipping**
2. Cohen Sutherland Line Clipping
3. Liang Barsky Line Clipping

Analytical Line Clipping

Also called *Brute force* line clipping can be performed as follows:

1. Don't clip lines with both end-points within the window



Analytical Line Clipping

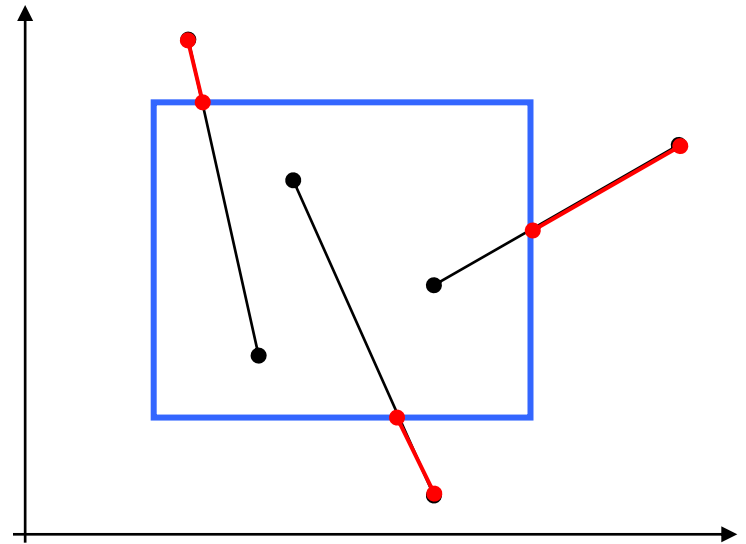
2. For lines with one end-point inside the window and one end-point outside, calculate the intersection point (using the equation of the line) and clip from this point out

– Use parametric equations

$$x = x_0 + t(x_1 - x_0)$$

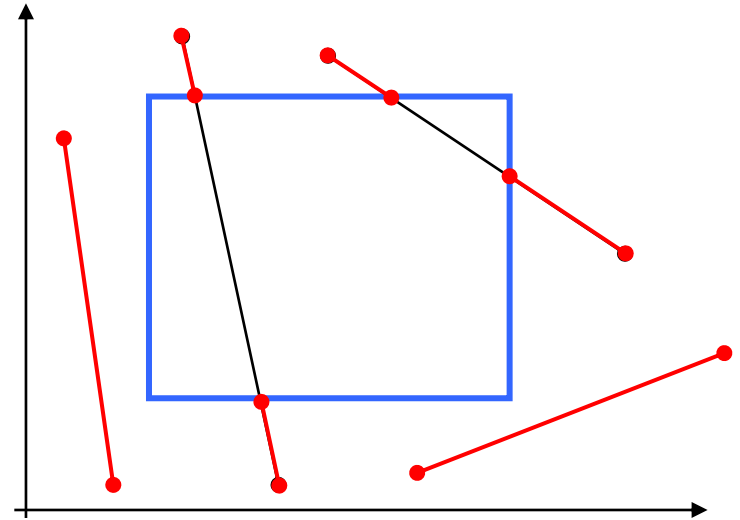
$$y = y_0 + t(y_1 - y_0)$$

– Intersection if $0 \leq t \leq 1$



Analytical Line Clipping

3. For lines with both end-points outside the window test the line for intersection with all of the window boundaries, and clip appropriately



+ **Very Simple method**

- However, calculating line intersections is computationally expensive
- Because a scene can contain so many lines, the brute force approach to clipping is much too slow

2D Line Clipping Algorithms

1. Analytical Line Clipping
2. **Cohen Sutherland Line Clipping**
3. Liang Barsky Line Clipping

Cohen-Sutherland Line Clipping

- An efficient line clipping algorithm
- The key advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated.



Dr. Ivan E. Sutherland co-developed the Cohen-Sutherland clipping algorithm. Sutherland is a graphics giant and includes amongst his achievements the invention of the head mounted display.



Cohen is something of a mystery – can anybody find out who he was?

Cohen-Sutherland Line Clipping

- Two phases Algorithm

Phase I: Identification Phase

All line segments fall into one of the following categories

1. Visible: Both endpoints lies inside
2. Invisible: Line completely lies outside
3. Clipping Candidate: A line neither in category 1 or 2

Phase II: Perform Clipping

Compute intersection for all lines that are candidate for clipping.

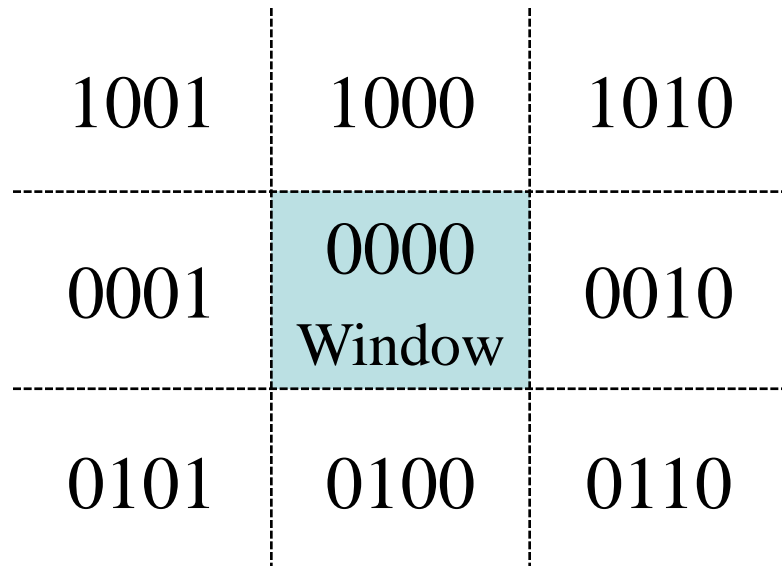
Cohen-Sutherland Line Clipping

Phase I: Identification Phase: World space is divided into regions based on the window boundaries

- Each region has a unique four bit region code
- Region codes indicate the position of the regions with respect to the window

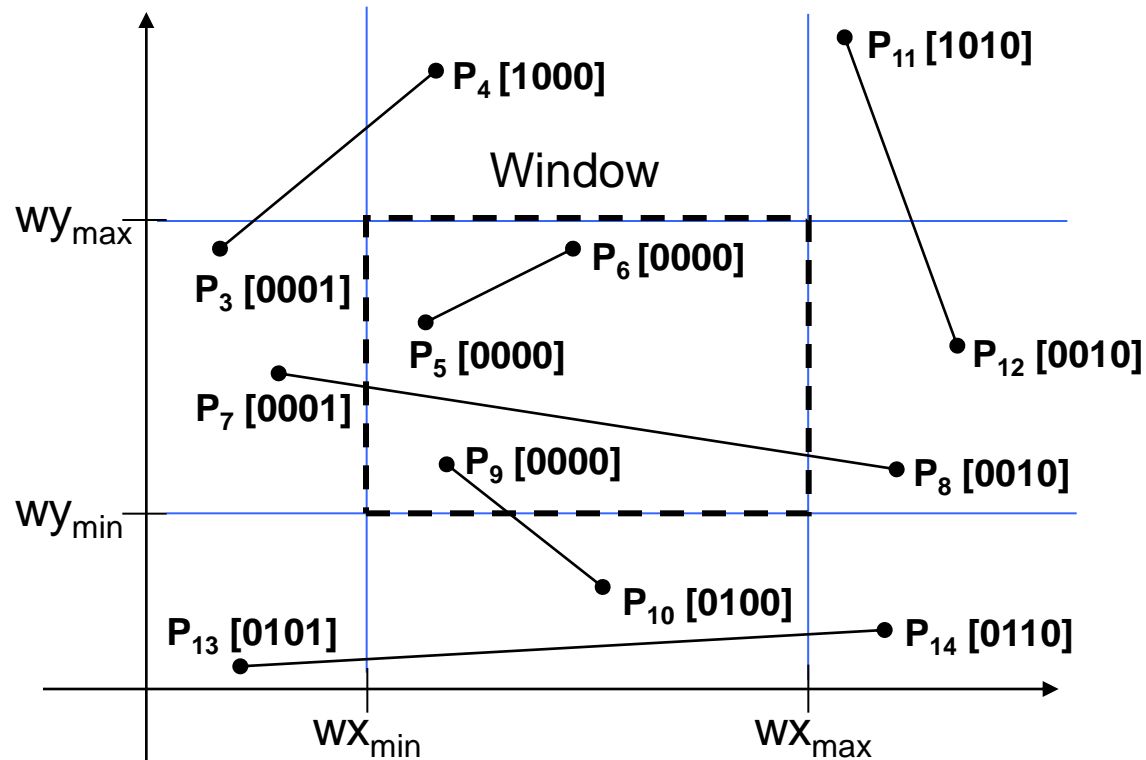
3	2	1	0
above	below	right	left

Region Code Legend



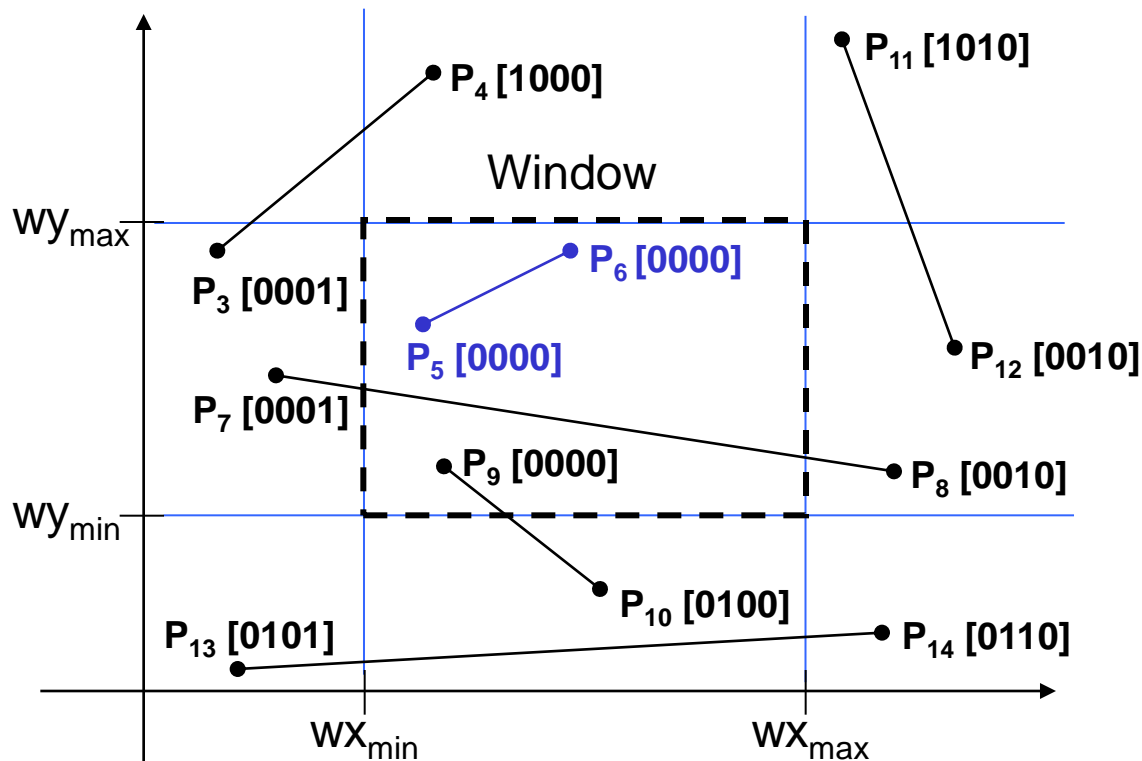
Cohen-Sutherland Line Clipping

Every end-point is labelled with the appropriate region code



Cohen-Sutherland Line Clipping

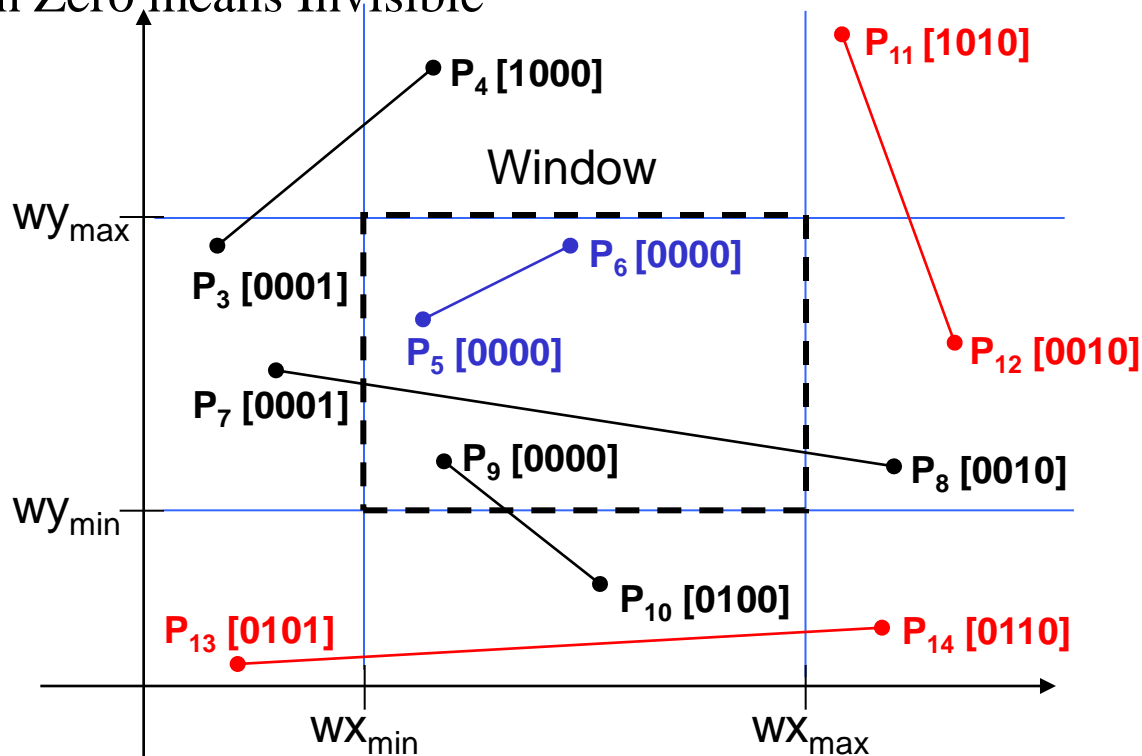
Visible Lines: Lines completely contained within the window boundaries have region code [0000] for both end-points so are not clipped



Cohen-Sutherland Line Clipping

Invisible Lines: Any line with a common set bit in the region codes of both end-points can be clipped completely

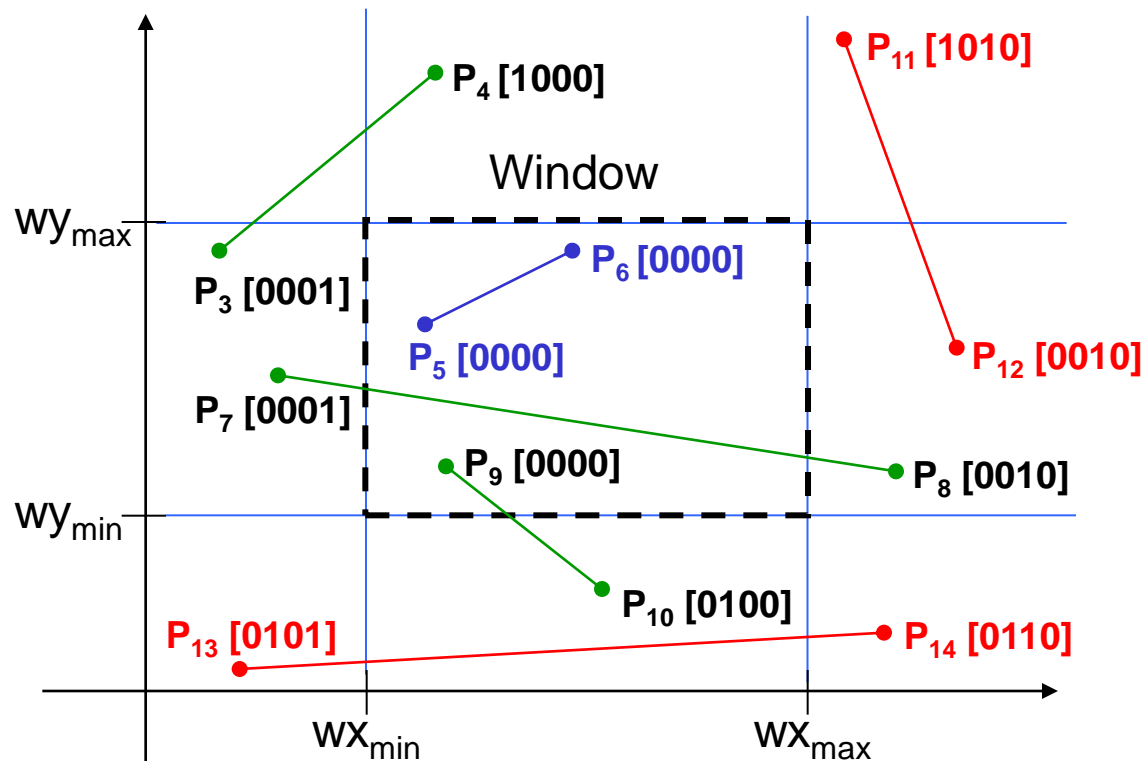
- The AND operation can efficiently check this
- Non Zero means Invisible



Cohen-Sutherland Line Clipping

Clipping Candidates: Lines that cannot be identified as completely inside or outside the window may or may not cross the window interior. These lines are processed in Phase II.

- If AND operation result in 0 the line is candidate for clipping



Cohen-Sutherland Clipping Algorithm

Phase II: Clipping Phase: Lines that are in category 3 are now processed as follows:

- Compare an end-point outside the window to a boundary (choose any order in which to consider boundaries e.g. left, right, bottom, top) and determine how much can be discarded
- If the remainder of the line is entirely inside or outside the window, retain it or clip it respectively
- Otherwise, compare the remainder of the line against the other window boundaries
- Continue until the line is either discarded or a segment inside the window is found

Cohen-Sutherland Line Clipping

- Intersection points with the window boundaries are calculated using the line-equation parameters
 - Consider a line with the end-points (x_1, y_1) and (x_2, y_2)
 - The y-coordinate of an intersection with a vertical window boundary can be calculated using:

$$y = y_1 + m (x_{boundary} - x_1)$$

where $x_{boundary}$ can be set to either wx_{min} or wx_{max}

- The x-coordinate of an intersection with a horizontal window boundary can be calculated using:

$$x = x_1 + (y_{boundary} - y_1) / m$$

where $y_{boundary}$ can be set to either wy_{min} or wy_{max}

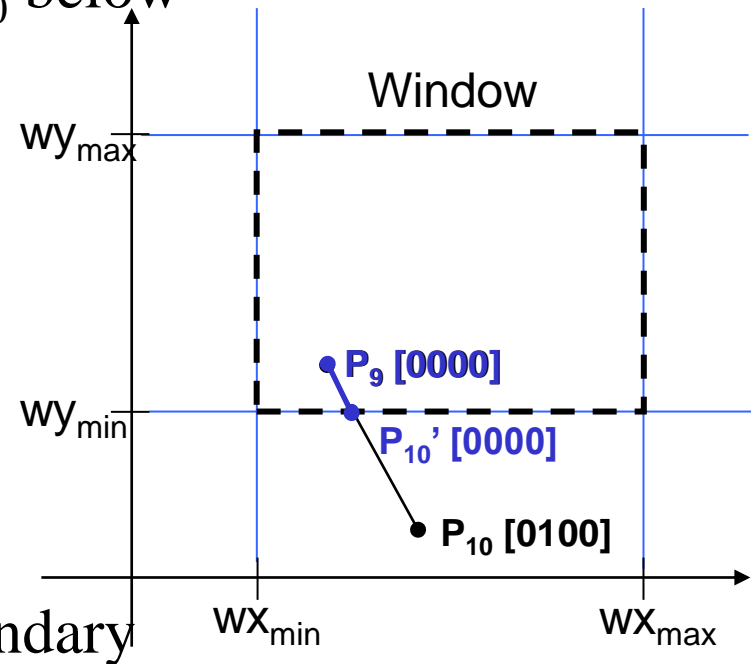
Cohen-Sutherland Line Clipping

- We can use the region codes to determine which window boundaries should be considered for intersection
 - To check if a line crosses a particular boundary we compare the appropriate bits in the region codes of its endpoints
 - If one of these is a 1 and the other is a 0 then the line crosses the boundary.

Cohen-Sutherland Line Clipping

Example 1: Consider the line P_9 to P_{10} below

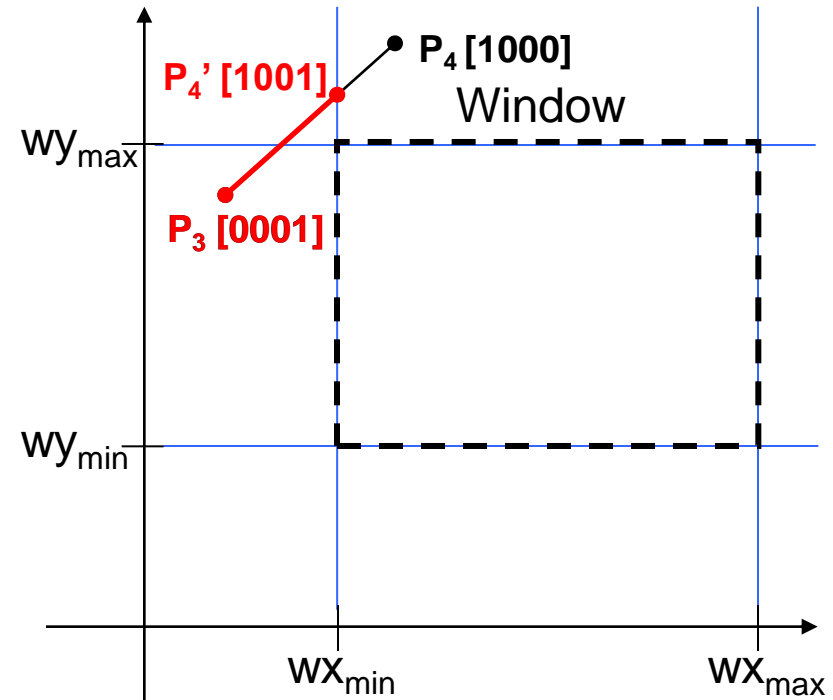
- Start at P_{10}
- From the region codes of the two end-points we know the line doesn't cross the left or right boundary
- Calculate the intersection of the line with the bottom boundary to generate point P_{10}'
- The line P_9 to P_{10}' is completely inside the window so is retained



Cohen-Sutherland Line Clipping

Example 2: Consider the line P_3 to P_4 below

- Start at P_4
- From the region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_4'

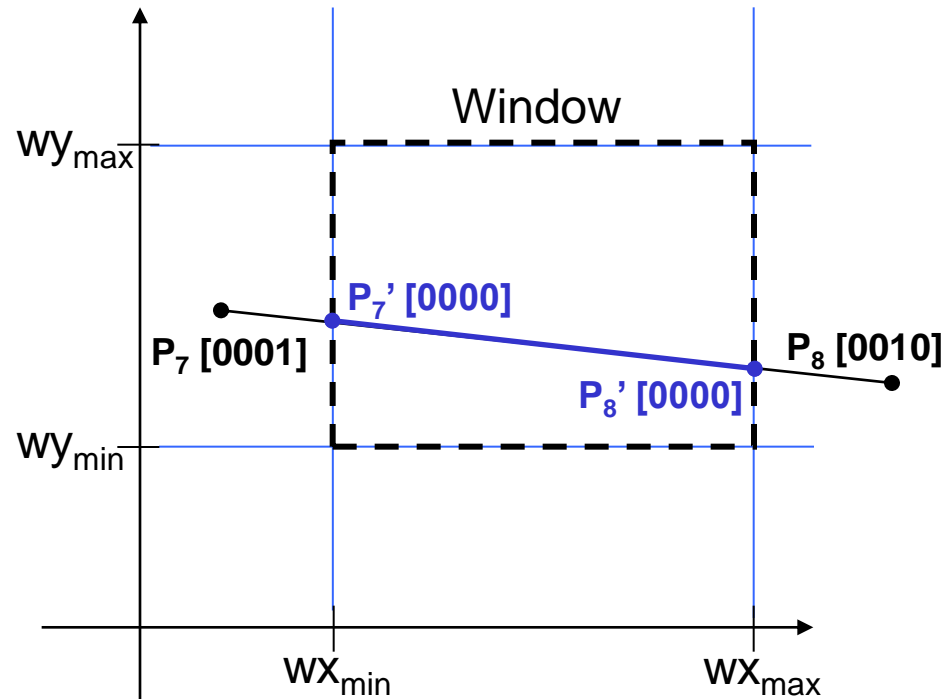


- The line P_3 to P_4' is completely outside the window so is clipped

Cohen-Sutherland Line Clipping

Example 3: Consider the line P_7 to P_8 below

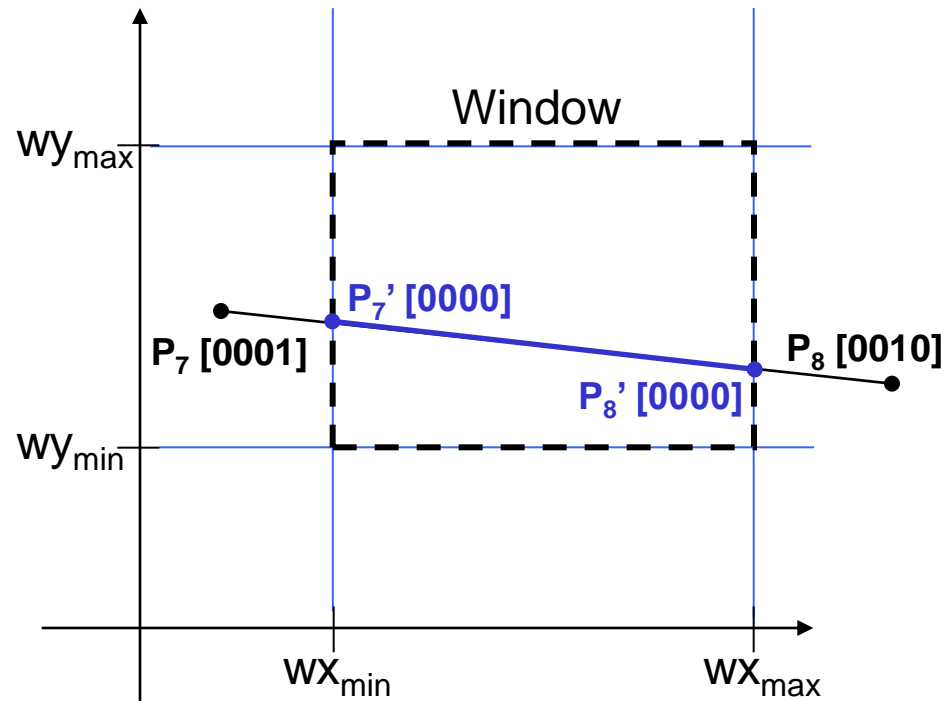
- Start at P_7
- From the two region codes of the two end-points we know the line crosses the left boundary so calculate the intersection point to generate P_7'



Cohen-Sutherland Line Clipping

Example 4: Consider the line P_7' to P_8

- Start at P_8
- Calculate the intersection with the right boundary to generate P_8'
- P_7' to P_8' is inside the window so is retained



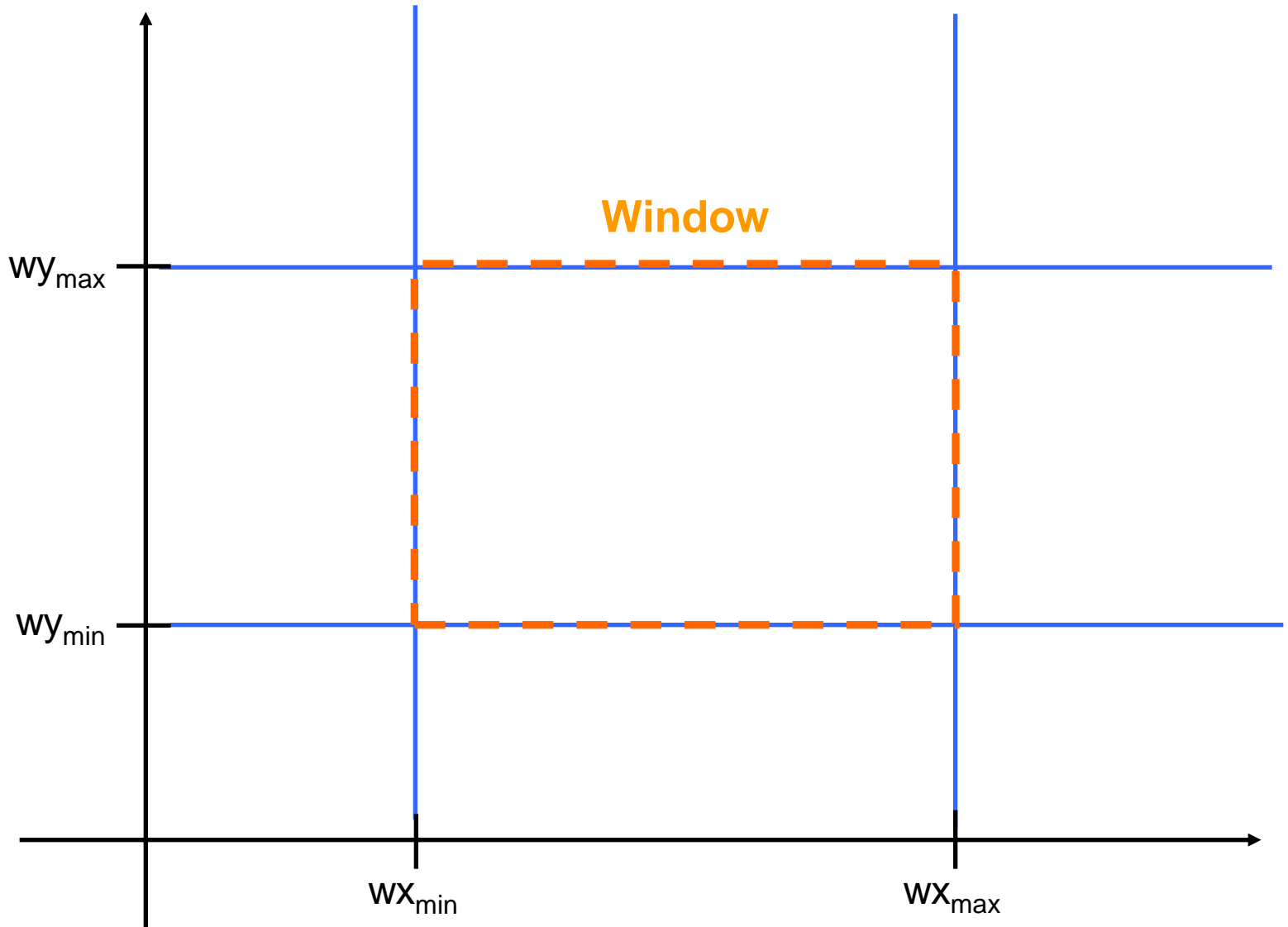
Cohen-Sutherland Line Clipping

Mid-Point Subdivision Method

– Algorithm

1. Initialise the list of lines to all lines
2. Classify lines as in *Phase I*
3. Remove all lines from the list in category 1 or 2;
4. Divide all lines of category 3 into two smaller segments at mid-point (x_m, y_m) where $x_m = (x_1 + x_2)/2$ and $y_m = (y_1 + y_2)/2$
5. Remove the original line from list and enter its two newly created segments.
6. Repeat step 2-5 until list is null.

Cohen-Sutherland Line Clipping



Cohen-Sutherland Line Clipping

Mid-Point Subdivision Method

- Integer Version
- Fast as Division by 2 can be performed by simple shift right operation
- For $N \times N$ max dimension of line number of subdivisions required $\log_2 N$.
- Thus a 1024×1024 raster display require just 10 subdivisions.....

2D Line Clipping Algorithms

1. Analytical Line Clipping
2. Cohen Sutherland Line Clipping
3. **Liang Barsky Line Clipping**

Liang-Barsky Line Clipping

Introduction:

- Cohen-Sutherland sometimes performs a lot of fruitless clipping due to external intersections, but oldest, widely published, most common
- Cyrus-Beck (1978) and Liang-Barsky (1984) are more efficient
- C-B is a parametric line-clipping algorithm
- L-B is based on C-B algorithm. It adds efficient trivial rejection tests
 - can be used for 2D line clipping against arbitrary convex polygons

Liang-Barsky Line Clipping

- Using parametric equations, compute line segment intersections (actually, just values of \mathbf{u}) with clipping region edges
- Determine if the four values of \mathbf{u} actually correspond to real intersections
- Then calculate x and y values of the intersections
- L-B examines values of \mathbf{u} for earlier reject

Liang-Barsky Line Clipping

- Parametric definition of a line:
 - $x = x_1 + u\Delta x$
 - $y = y_1 + u\Delta y$
 - $\Delta x = (x_2 - x_1)$, $\Delta y = (y_2 - y_1)$, $0 \leq u \leq 1$
- Goal: find range of u for which x and y *both* inside the viewing window

Liang-Barsky Line Clipping

– Mathematically:

$$x_{\min} \leq x_1 + u\Delta x \leq x_{\max}$$

$$y_{\min} \leq y_1 + u\Delta y \leq y_{\max}$$

– Rearranged

$$1: u^*(-\Delta x) \leq (x_1 - x_{\min})$$

$$2: u^*(\Delta x) \leq (x_{\max} - x_1)$$

$$3: u^*(-\Delta y) \leq (y_1 - y_{\min})$$

$$4: u^*(\Delta y) \leq (y_{\max} - y_1)$$

– In general

$$u^*(p_k) \leq (q_k) \quad k = 1, 2, 3, 4$$

where

$p_1 = -\Delta x$	$q_1 = x_1 - x_{\min}$ (left)
$p_2 = \Delta x$	$q_2 = x_{\max} - x_1$ (right)
$p_3 = -\Delta y$	$q_3 = y_1 - y_{\min}$ (bottom)
$p_4 = \Delta y$	$q_4 = y_{\max} - y_1$ (top)

Liang-Barsky Line Clipping

– Rules:

- 1) $p_k = 0$: the line is parallel to boundaries
 - If for that same k , $q_k < 0$, it's outside
 - Otherwise it's inside
- 2) $p_k < 0$: the line starts outside this boundary
 - $r_k = q_k/p_k$
 - $u_1 = \max(0, r_k, u_1)$
- 3) $p_k > 0$: the line starts inside the boundary
 - $r_k = q_k/p_k$
 - $u_2 = \min(1, r_k, u_2)$
- 4) If $u_1 > u_2$, the line is completely outside

Liang-Barsky Line Clipping

1. A line parallel to a clipping window edge has $p_k = 0$ for that boundary.
2. If for that k , $q_k < 0$, the line is completely outside and can be eliminated.
3. When $p_k < 0$ the line proceeds outside to inside the clip window and when $p_k > 0$, the line proceeds inside to outside.
4. For nonzero p_k , $u = q_k / p_k$ gives the intersection point.
5. For each line, calculate u_1 and u_2 . For u_1 , look at boundaries for which $p_k < 0$ (outside \rightarrow in). Take u_1 to be the largest among $(0, q_k / p_k)$. For u_2 , look at boundaries for which $p_k > 0$ (inside \rightarrow out). Take u_2 to be the minimum of $(1, q_k / p_k)$. If $u_1 > u_2$, the line is outside and therefore rejected.

Liang-Barsky Line Clipping

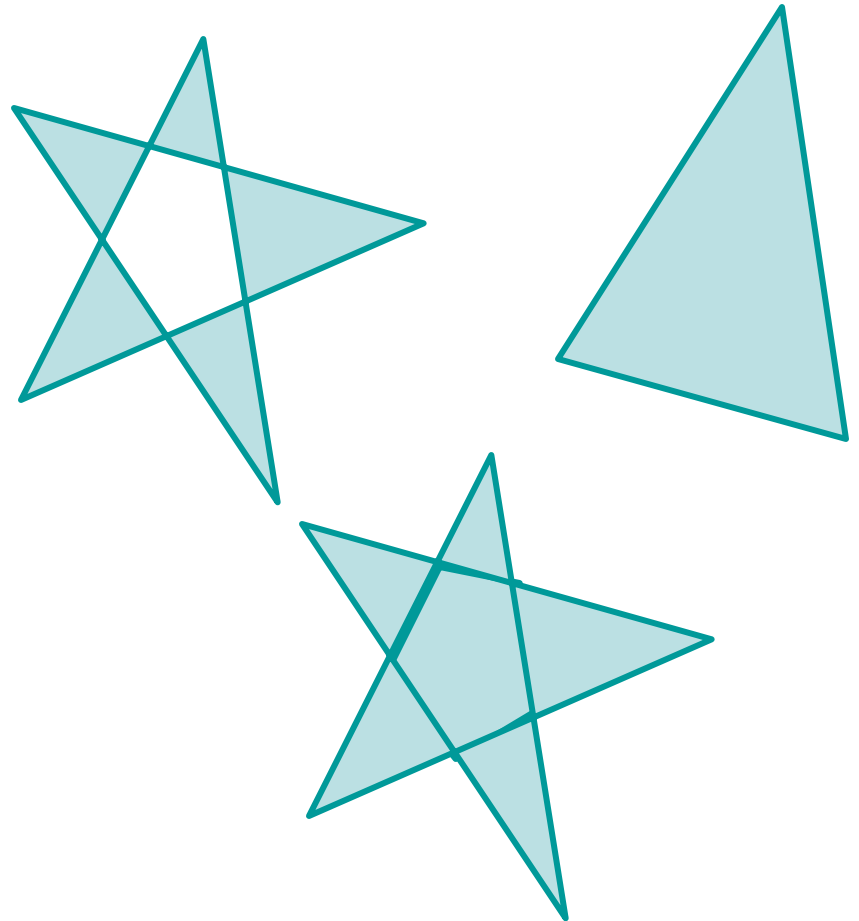
- Faster than Cohen-Sutherland, does not need to iterate
- can be used for 2D line clipping against arbitrary convex polygons
- Also extends to 3D
 - Add $z = z_1 + u\Delta z$
 - Add 2 more p's and q's
 - Still only 2 u's

2D Clipping

1. Introduction
2. Point Clipping
3. Line Clipping
4. **Polygon / Area Clipping**
5. Text Clipping

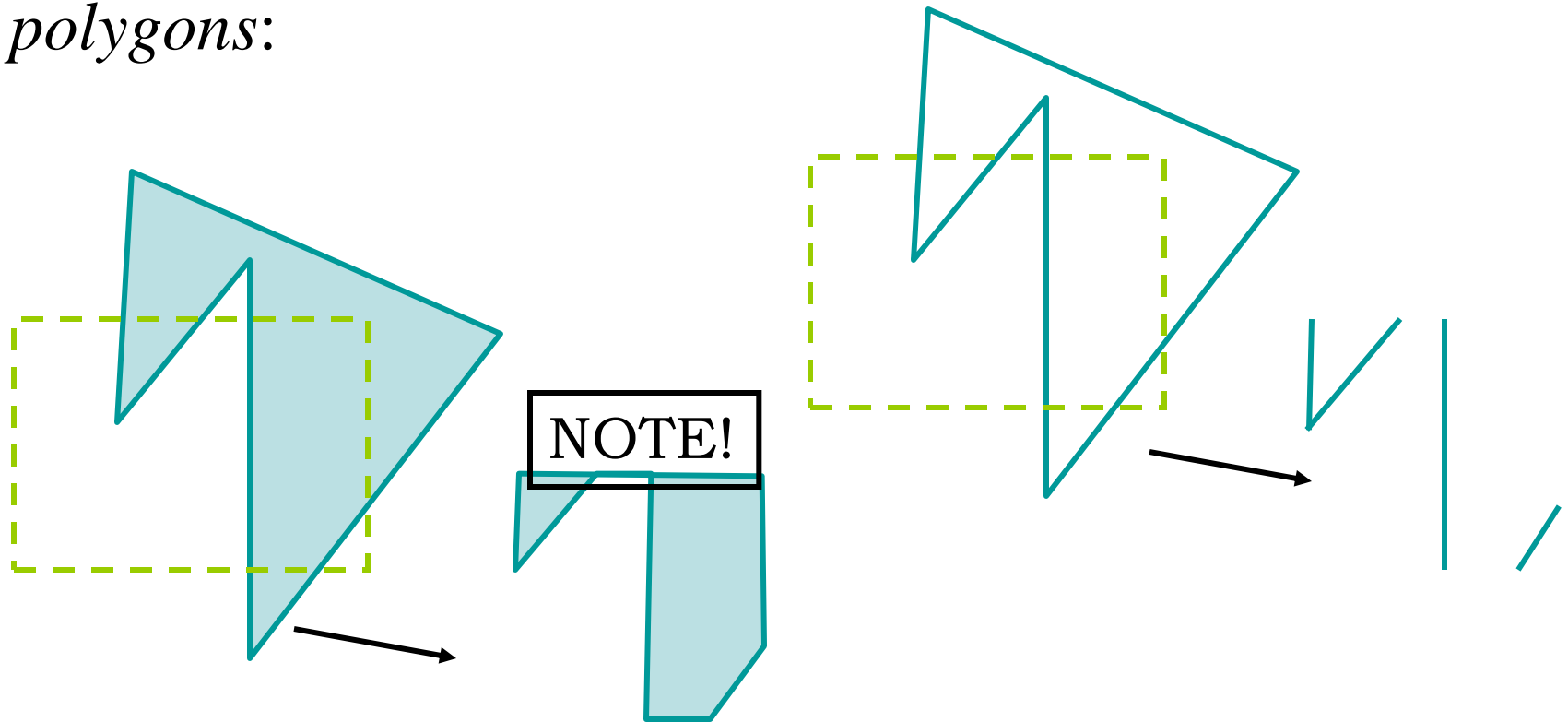
Polygon Clipping

- Polygons have a distinct *inside* and *outside*...
- Decided by
 - Even/Odd
 - Winding Number



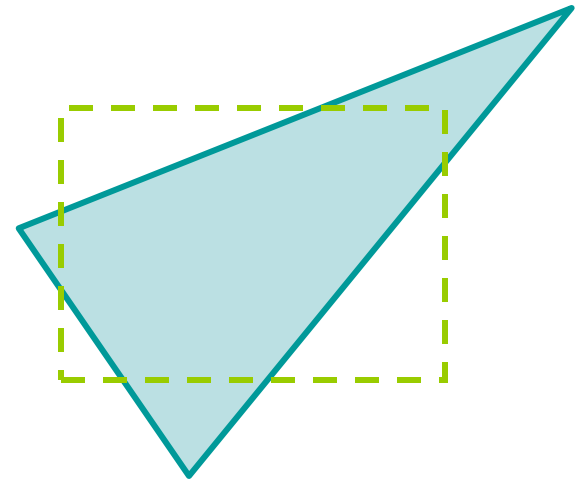
Polygon Clipping

- Note the difference between clipping *lines* and *polygons*:



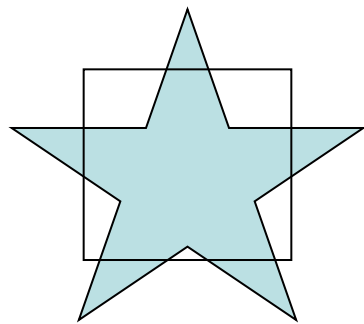
Polygon Clipping

- Some difficulties:
 - Maintaining correct inside/outside
 - Variable number of vertices
 - Handle screen corners correctly

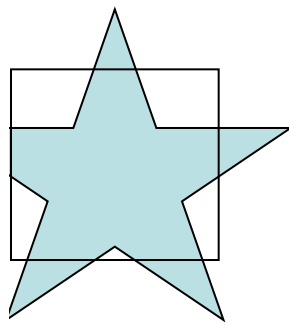


Sutherland-Hodgman Area Clipping

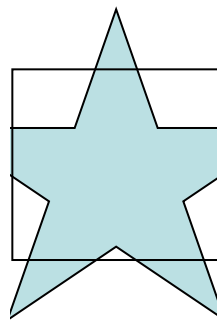
- A technique for clipping areas developed by Sutherland & Hodgman
- Put simply the polygon is clipped by comparing it against each boundary in turn



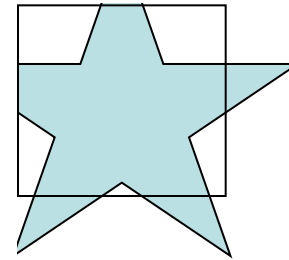
Original Area



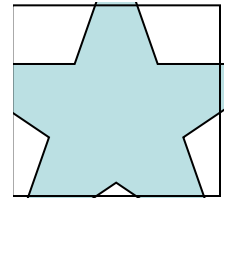
Clip Left



Clip Right

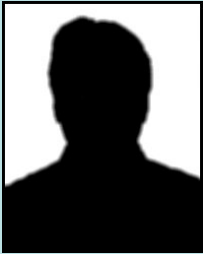


Clip Top



Clip Bottom

Sutherland turns up again. This time with Gary Hodgman with whom he worked at the first ever graphics company Evans & Sutherland



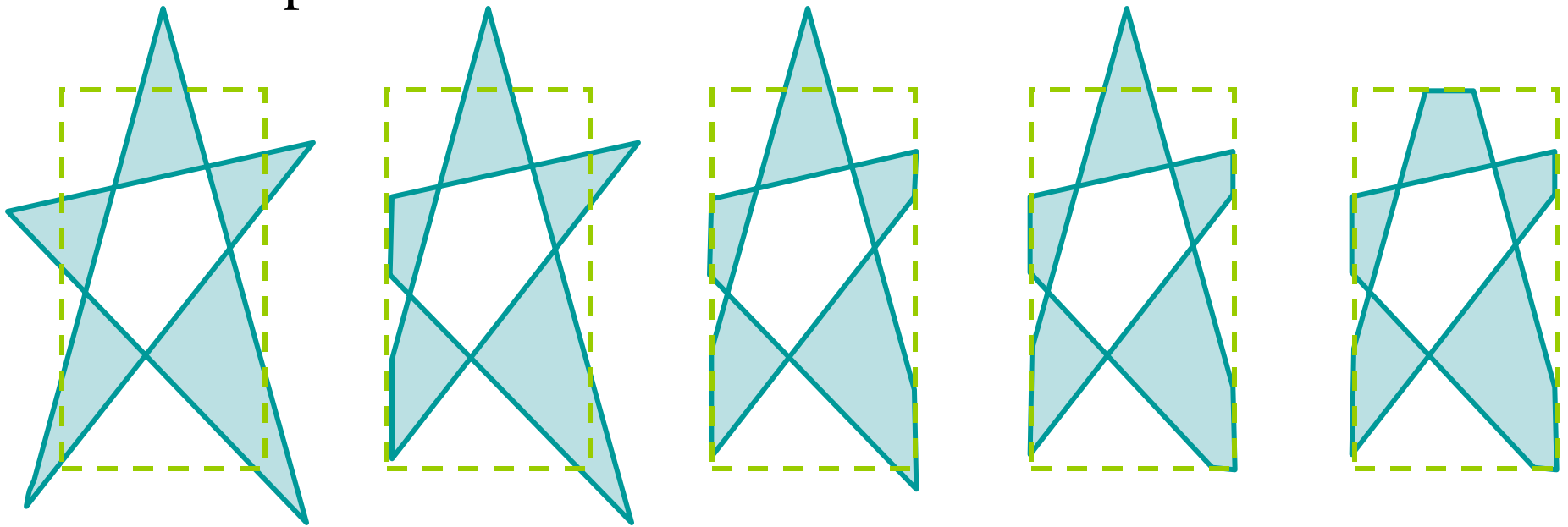
Sutherland-Hodgeman Polygon Clipping

1. Basic Concept:

- Simplify via separation
- Clip whole polygon against one edge
 - Repeat with output for other 3 edges
 - Similar for 3D
- You can create intermediate vertices that get thrown out

Sutherland-Hodgeman Polygon Clipping

- Example



Start

Left

Right

Bottom

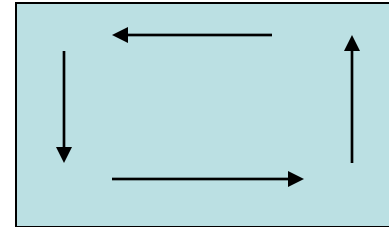
Top

Note that the point one of the points added when clipping on the right gets removed when we clip with bottom

Sutherland-Hodgeman Polygon Clipping

2. Algorithm:

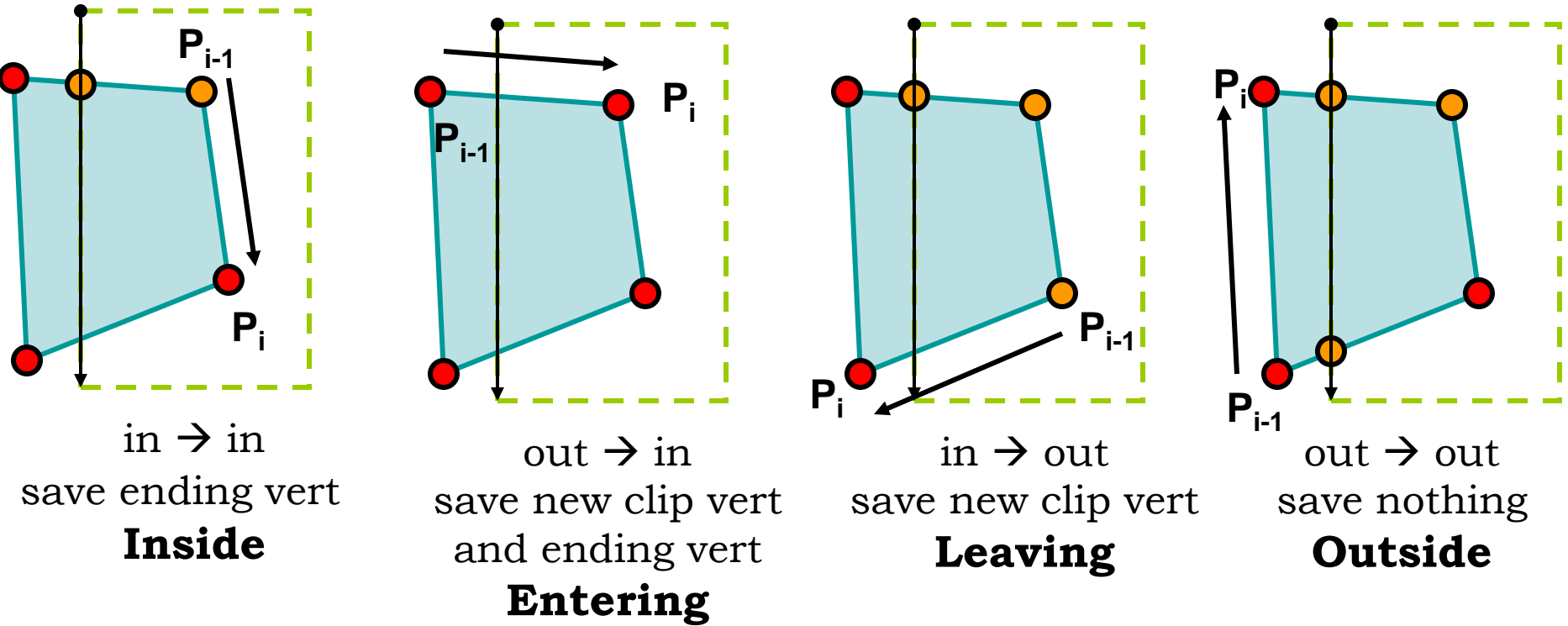
Let (P_1, P_2, \dots, P_N) be the vertex list of the Polygon to be clipped and E be the edge of *positively oriented, convex clipping window*.



We clip each edge of the polygon in turn against each window edge E , forming a new polygon whose vertices are determined as follows:

Sutherland-Hodgeman Polygon Clipping

Creating New Vertex List

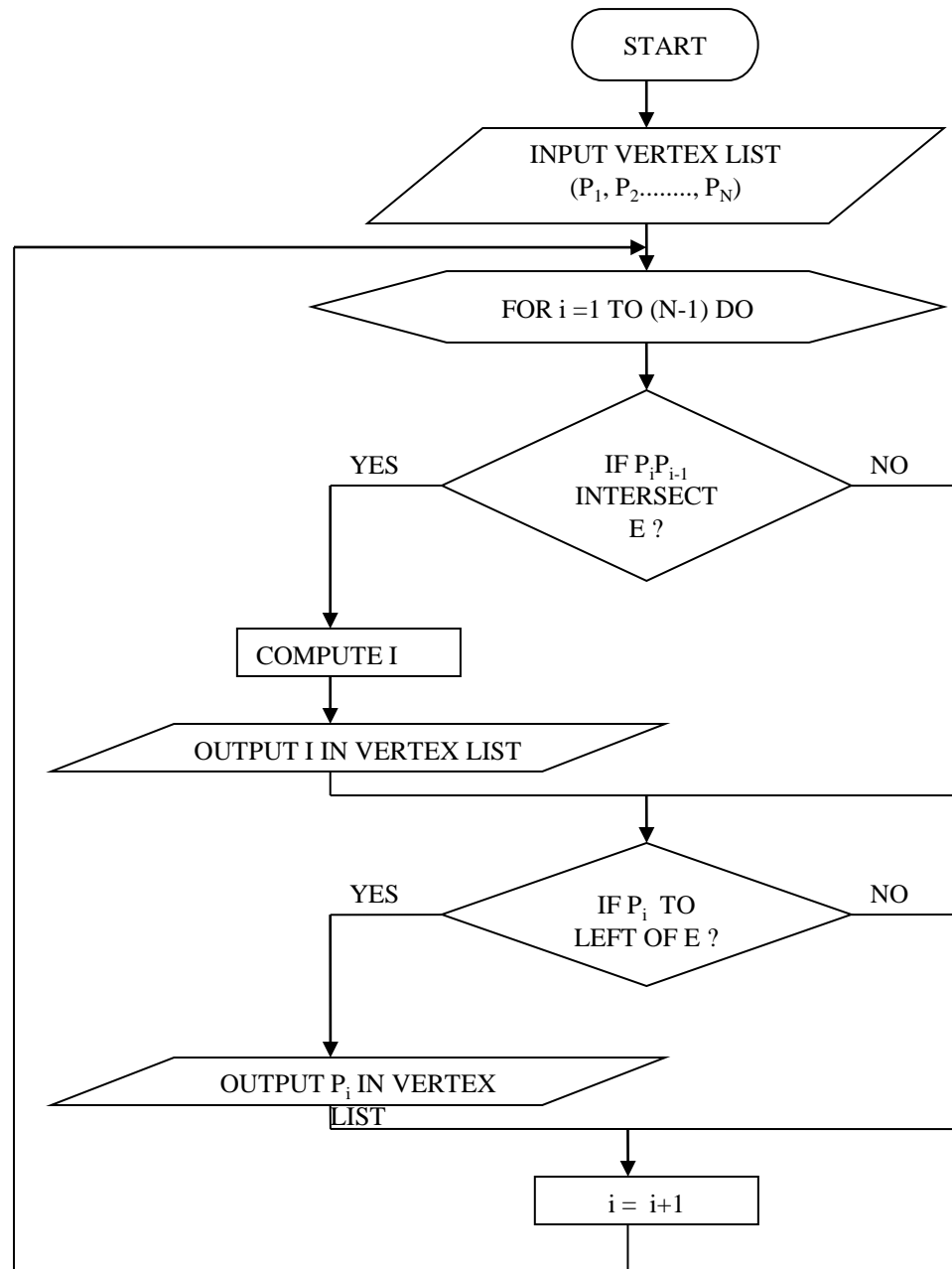


Sutherland-Hodgeman Polygon Clipping

Four cases

1. **Inside:** If both P_{i-1} and P_i are to the left of window edge vertex then P_i is placed on the output vertex list.
2. **Entering:** If P_{i-1} is to the right of window edge and P_i is to the left of window edge vertex then intersection (I) of P_{i-1} P_i with edge E and P_i are placed on the output vertex list.
3. **Leaving:** If P_{i-1} is to the left of window edge and P_i is to the right of window edge vertex then only intersection (I) of P_{i-1} P_i with edge E is placed on the output vertex list.
4. **Outside:** If both P_{i-1} and P_i are to the right of window edge nothing is placed on the output vertex list.

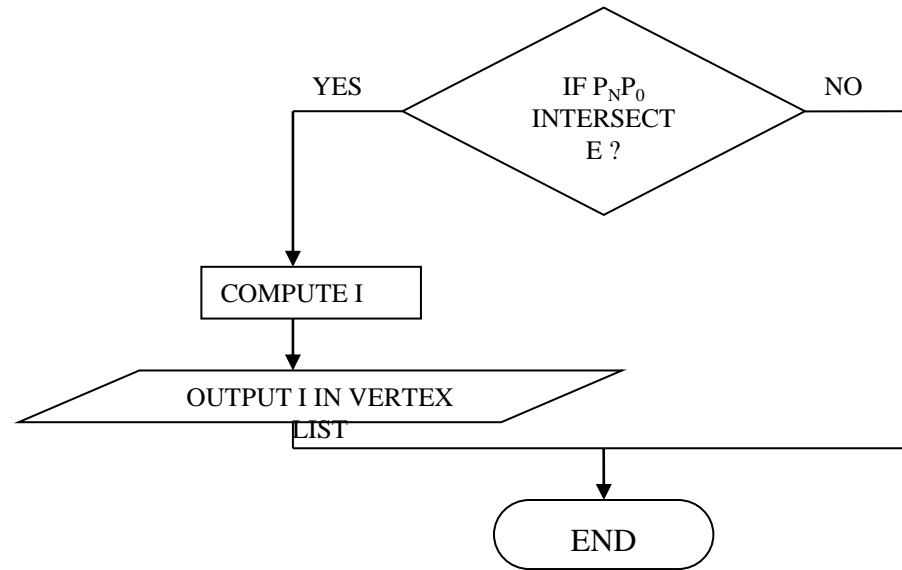
Flow Chart



Special case for
first Vertex

Flow Chart

Special case for
first Vertex



YOU CAN ALSO APPEND AN ADDITIONAL VERTEX
 $P_{N+1} = P_1$ AND AVOID SPECIAL CASE FOR FIRST
VERTEX

Sutherland-Hodgeman Polygon Clipping

Inside/Outside Test:

Let $P(x,y)$ be the polygon vertex which is to be tested against edge E defined from $A(x_1, y_1)$ to $B(x_2, y_2)$. Point P is to be said to the left (inside) of E or AB iff

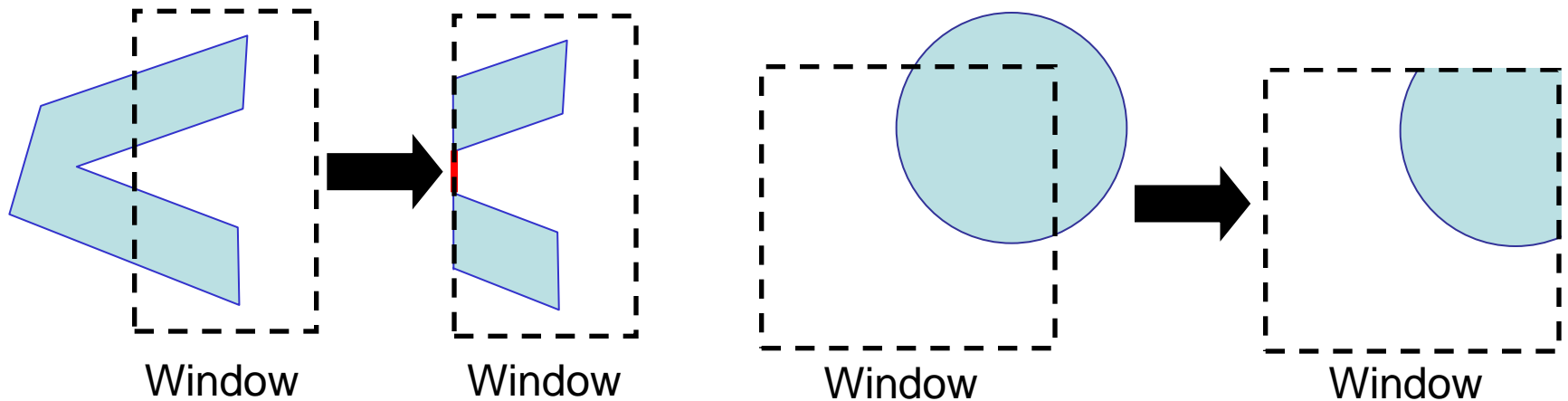
$$\frac{y - y_1}{y_2 - y_1} - \frac{x - x_1}{x_2 - x_1} > 0$$

or $C = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) > 0$

otherwise it is said to be the right/Outside of edge E

Other Area Clipping Concerns

- Clipping concave areas can be a little more tricky as often superfluous lines must be removed



- Clipping curves requires more work
 - For circles we must find the two intersection points on the window boundary

2D Clipping

1. Introduction
2. Point Clipping
3. Line Clipping
4. Polygon/Area Clipping
- 5. Text Clipping**

Text Clipping

Text clipping relies on the concept of bounding rectangle

TYPES

– All or None String Clipping: The bounding rectangle is on the word.

STRING

– All or None Character Clipping: The bounding rectangle is on the character.

S T R I N G

– Character Clipping **STR**ING → **STI**

Text Clipping

METHODS

- Point Clipping in case of Bit Mapped Fonts
- Curve, Line or Polygon clipping in case of Outlined fonts

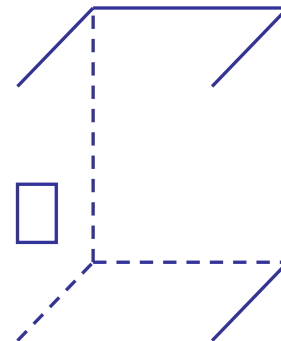
Visible-Surface Detection Methods

Contents

- **Abstract**
- **Introduction**
- **Back-Face Detection**
- **Depth-Buffer Method**
- **A-Buffer Method**
- **Scan-Line Method**
- **Depth-Sorting Method**
- ◆ **Area-Subdivision Method**
- ◆ **Octree Method**
- ◆ **Ray-Casting Method**
- ◆ **Image-Space Method vs. Object-Space Method**
- ◆ **Curved Surfaces**
- ◆ **Wireframe**

Abstract

- Hidden-surface elimination methods
- Identifying visible parts of a scene from a viewpoint
- Numerous algorithms
 - More memory - storage
 - More processing time – execution time
 - Only for special types of objects - constraints
- Deciding a method for a particular application
 - Complexity of the scene
 - Type of objects
 - Available equipment
 - Static or animated scene



Introduction

Classification of Visible-Surface Detection Algorithms

- ***Object-space methods*** vs. ***Image-space methods***
 - Object definition directly vs. their projected images
 - Most visible-surface algorithms use image-space methods
 - Object-space can be used effectively in some cases
 - Ex) Line-display algorithms
- Object-space methods
 - Compares objects and parts of objects to each other
- Image-space methods
 - Point by point at each pixel position on the projection plane

Sorting and Coherence Methods

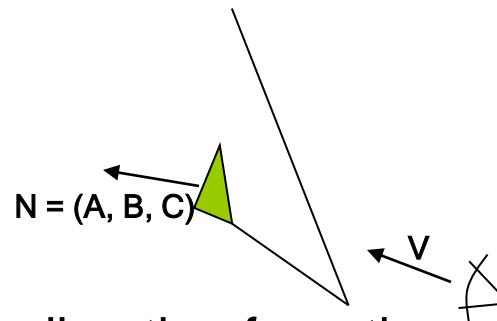
- To improve performance
- ***Sorting***
 - Facilitate depth comparisons
 - Ordering the surfaces according to their distance from the viewplane
- ***Coherence***
 - Take advantage of regularity
 - Epipolar geometry
 - Topological coherence

Back-Face Detection

Inside-outside test

- A point (x, y, z) is “inside” a surface with plane parameters $A, B, C,$ and D if
- The polygon is a back face if $Ax + By + Cz + D < 0$

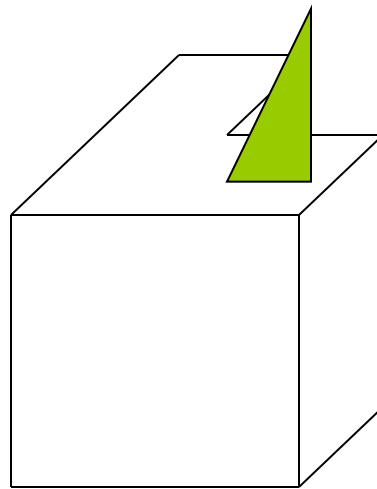
$$V \cdot N > 0$$



- V is a vector in the viewing direction from the eye(camera)
- N is the normal vector to a polygon surface

Advanced Configuration

- In the case of ***concave polyhedron***
 - Need more tests
 - Determine faces totally or partly obscured by other faces
 - In general, back-face removal can be expected to eliminate *about half of the surfaces* from further visibility tests

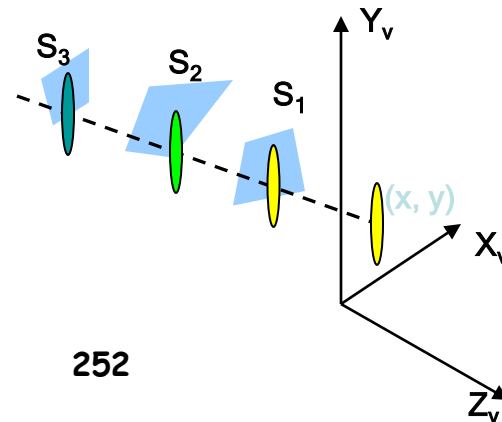


<View of a concave polyhedron with
one face partially hidden by other surfaces>

Depth-Buffer Method

Characteristics

- Commonly used image-space approach
- Compares depths of each pixel on the projection plane
 - Referred to as the *z-buffer* method
- Usually applied to scenes of polygonal surfaces
 - Depth values can be computed very quickly
 - Easy to implement



Depth Buffer & Refresh Buffer

- Two buffer areas are required
 - Depth buffer
 - Store depth values for each (x, y) position
 - All positions are initialized to minimum depth
 - Usually 0 – most distant depth from the viewplane
 - Refresh buffer
 - Stores the intensity values for each position
 - All positions are initialized to the background intensity

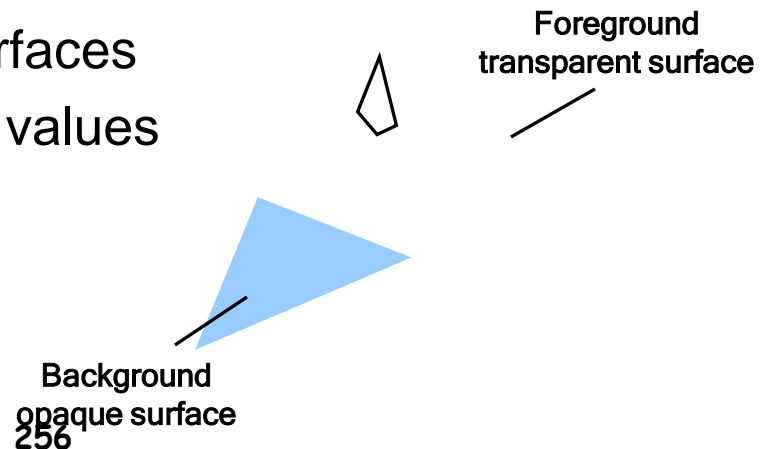
Algorithm

- Initialize the depth buffer and refresh buffer
$$\text{depth}(x, y) = 0, \quad \text{refresh}(x, y) = I_{\text{backgnd}}$$
- For each position on each polygon surface
 - Calculate the depth for each (x, y) position on the polygon
 - If $z > \text{depth}(x, y)$, then set
$$\text{depth}(x, y) = z, \quad \text{refresh}(x, y) = I_{\text{surf}}(x, y)$$
- Advanced
 - With resolution of 1024 by 1024
 - Over a million positions in the depth buffer
 - Process one section of the scene at a time
 - Need a smaller depth buffer
 - The buffer is reused for the next section

A-Buffer Method

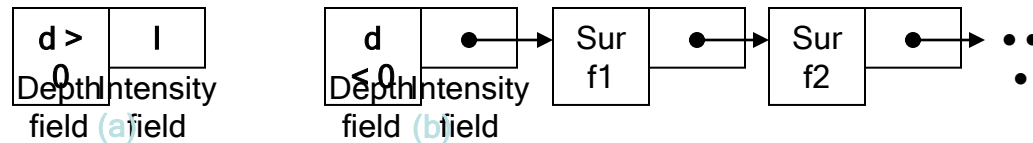
Characteristics

- An extension of the ideas in the depth-buffer method
- The origin of this name
 - At the other end of the alphabet from “z-buffer”
 - Antialiased, area-averaged, accumulation-buffer
 - Surface-rendering system developed by ‘Lucasfilm’
 - REYES(Renders Everything You Ever Saw)
- A drawback of the depth-buffer method
 - Deals only with opaque surfaces
 - Can’t accumulate intensity values for more than one surface



Algorithm(1 / 2)

- Each position in the buffer can reference a linked list of surfaces
 - Several intensities can be considered at each pixel position
 - Object edges can be antialiased
- Each position in the A-buffer has two fields
 - Depth field
 - Stores a positive or negative real number
 - Intensity field
 - Stores surface-intensity information or a pointer value



<Organization of an A-buffer pixel position : (a) single-surface overlap (b) multiple-surface overlap>

Algorithm(2 / 2)

- If the depth field is positive
 - The number at that position is the depth
 - The intensity field stores the RGB
- If the depth field is negative
 - Multiple-surface contributions to the pixel
 - The intensity field stores a pointer to a linked list of surfaces
 - Data for each surface in the linked list

- RGB intensity components
- Opacity parameters(percent of transparency)
- Depth
- Percent of area coverage
- Surface identifier
- Pointers to next surface

Scan-Line Method

Characteristics

- Extension of the scan-line algorithm for filling polygon interiors
 - For all polygons intersecting each scan line
 - Processed from left to right
 - Depth calculations for each overlapping surface
 - The intensity of the nearest position is entered into the refresh buffer

Tables for The Various Surfaces

- Edge table
 - Coordinate endpoints for each line
 - Slope of each line
 - Pointers into the polygon table
 - Identify the surfaces bounded by each line
- Polygon table
 - Coefficients of the plane equation for each surface
 - Intensity information for the surfaces
 - Pointers into the edge table

Active List & Flag

- Active list
 - Contain only edges across the current scan line
 - Sorted in order of increasing x
- Flag for each surface
 - Indicate whether inside or outside of the surface
 - At the leftmost boundary of a surface
 - The surface flag is turned on
 - At the rightmost boundary of a surface
 - The surface flag is turned off

Example

- Active list for scan line 1

- Edge table

- AB, BC, EH, and FG

- Between AB and BC, only

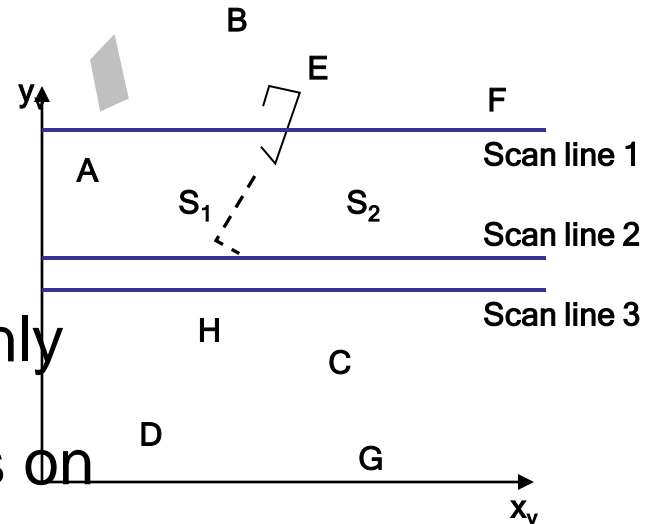
the flag for surface S_1 is on

- No depth calculations are necessary

- Intensity for surface S_1 is entered into the refresh buffer

- Similarly, between EH and FG, only the flag for S_2

is on

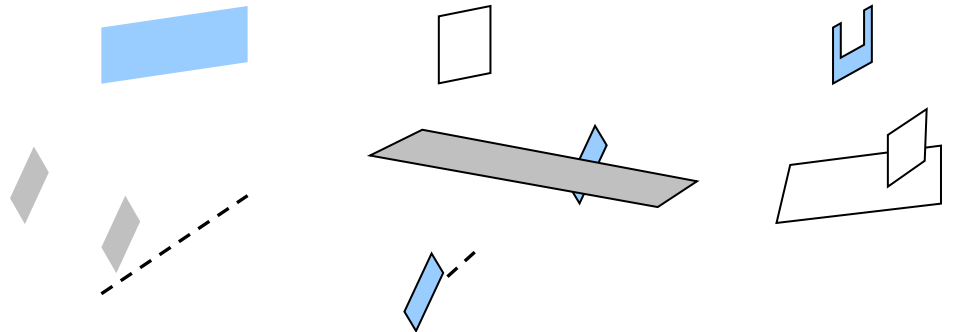


Example(cont.)

- For scan line 2, 3
 - AD, EH, BC, and FG
 - Between AD and EH, only the flag for S_1 is on
 - Between EH and BC, the flags for both surfaces are on
 - Depth calculation is needed
 - Intensities for S_1 are loaded into the refresh buffer until BC
 - Take advantage of coherence
 - Pass from one scan line to next
 - Scan line 3 has the same active list as scan line 2
 - Unnecessary to make depth calculations between EH and BC

Drawback

- Only if surfaces don't cut through or otherwise cyclically overlap each other
 - If any kind of cyclic overlap is present
 - Divide the surfaces



Depth-Sorting Method

Operations

- Image-space and object-space operations
 - Sorting operations in both image and object-space
 - The scan conversion of polygon surfaces in image-space
- Basic functions
 - Surfaces are sorted in order of decreasing depth
 - Surfaces are scan-converted in order, starting with the surface of greatest depth

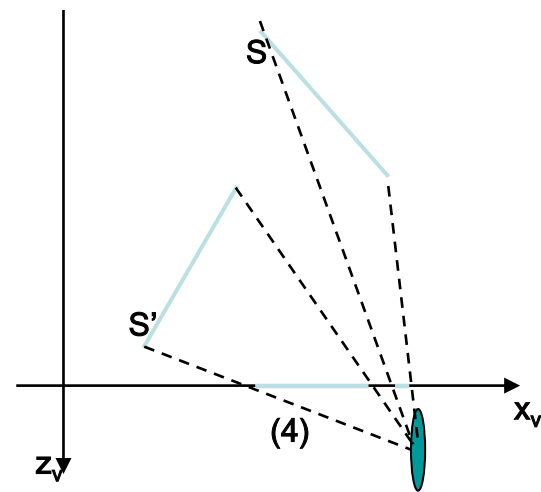
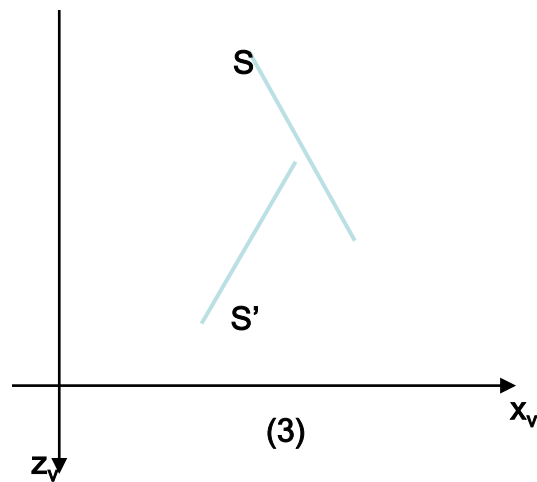
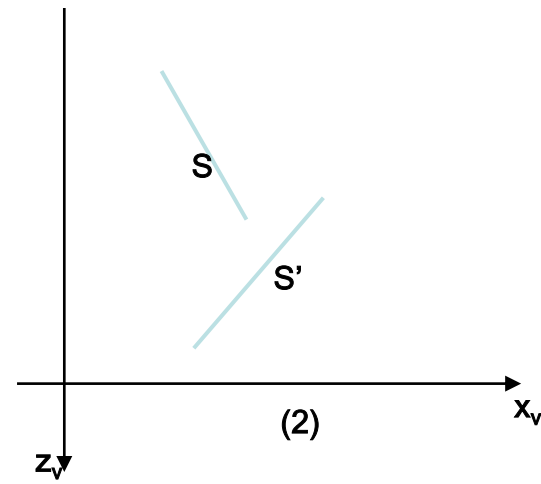
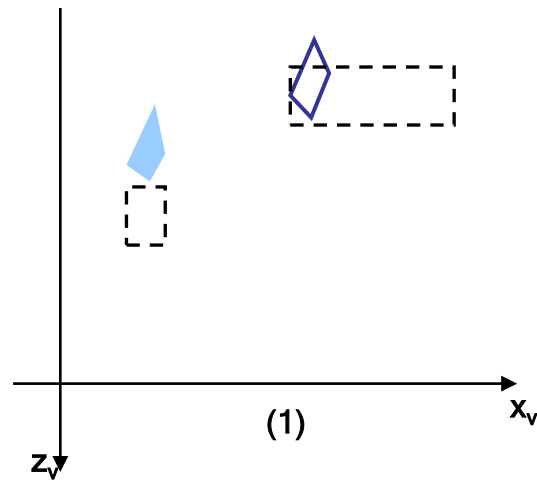
Algorithm

- Referred to as the *painter's algorithm*
 - In creating an oil painting
 - First paints the background colors
 - The most distant objects are added
 - Then the nearer objects, and so forth
 - Finally, the foregrounds are painted over all objects
 - Each layer of paint covers up the previous layer
- Process
 - Sort surfaces according to their distance from the viewplane
 - The intensities for the farthest surface are then entered into the refresh buffer
 - Taking each succeeding surface in decreasing depth order

Overlapping Tests

- Tests for each surface that overlaps with S
 - The bounding rectangle in the xy plane for the two surfaces do not overlap (1)
 - Surface S is completely behind the overlapping surface relative to the viewing position (2)
 - The overlapping surface is completely in front of S relative to the viewing position (3)
 - The projections of the two surfaces onto the viewplane do not overlap (4)
- If all the surfaces pass at least one of the tests, none of them is behind S
 - No reordering is then necessary and S is scan converted

Overlapping Test Examples



Surface Reordering

- If all four tests fail with S'
 - Interchange surfaces S and S' in the sorted list
 - Repeat the tests for each surface that is reordered in the list



Drawback

- If two or more surfaces alternately obscure each other
 - Infinite loop
 - Flag any surface that has been reordered to a farther depth
 - It can't be moved again
 - If an attempt to switch the surface a second time
 - Divide it into two parts to eliminate the cyclic loop
 - The original surface²⁷² is then replaced by the two

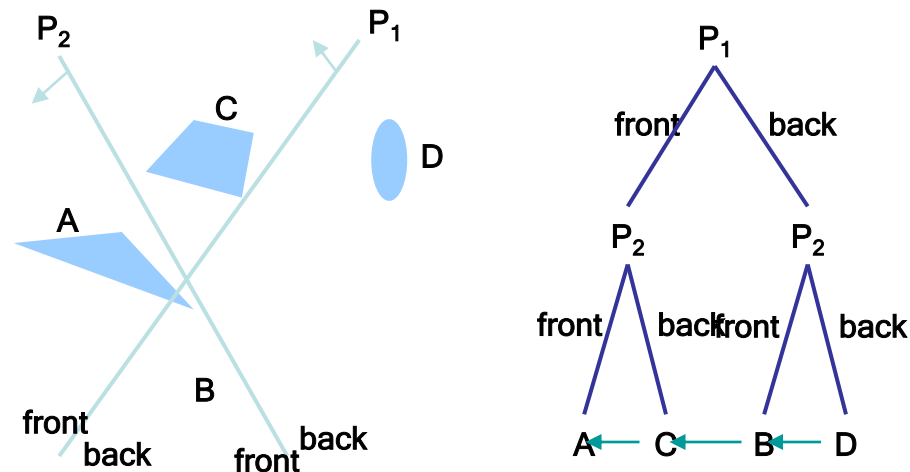
BSP-Tree Method

Characteristics

- Binary Space-Partitioning(BSP) Tree
- Determining object visibility by painting surfaces onto the screen from back to front
 - Like the painter's algorithm
- Particularly useful
 - The view reference point changes
 - The objects in a scene are at fixed positions

Process

- Identifying surfaces
 - “inside” and “outside” the partitioning plane
- Intersected object
 - Divide the object into two separate objects(A, B)



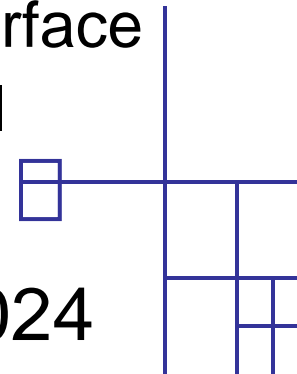
Area-Subdivision Method

Characteristics

- Takes advantage of area coherence
 - Locating view areas that represent part of a single surface
 - Successively dividing the total viewing area into smaller rectangles
 - Until each small area is the projection of part of a single visible surface or no surface
 - Require tests
 - Identify the area as part of a single surface
 - Tell us that the area is too complex to analyze easily
- Similar to constructing a *quadtree*

Process

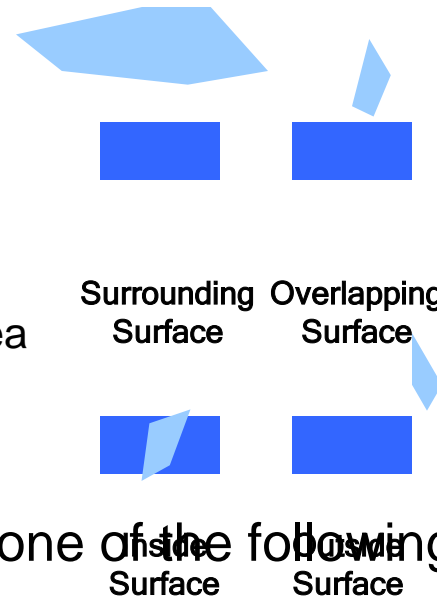
- Staring with the total view
 - Apply the identifying tests
 - If the tests indicate that the view is sufficiently complex
 - Subdivide
 - Apply the tests to each of the smaller areas
 - Until belonging to a single surface
 - Until the size of a single pixel
- Example
 - With a resolution 1024×1024
 - 10 times before reduced to a point



Identifying Tests

- Four possible relationships

- Surrounding surface
 - Completely enclose the area
- Overlapping surface
 - Partly inside and partly outside the area
- Inside surface
- Outside surface



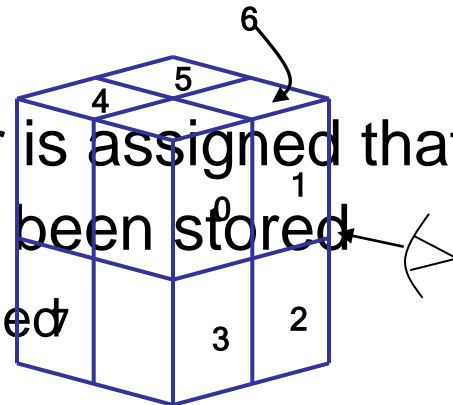
- No further subdivisions are needed if one of the following conditions is true

- All surface are outside surfaces with respect to the area
- Only one inside, overlapping, or surrounding surface is in the area
- A surrounding surface obscures all other surfaces within the area boundaries → from depth sorting, plane equation

Octree Method

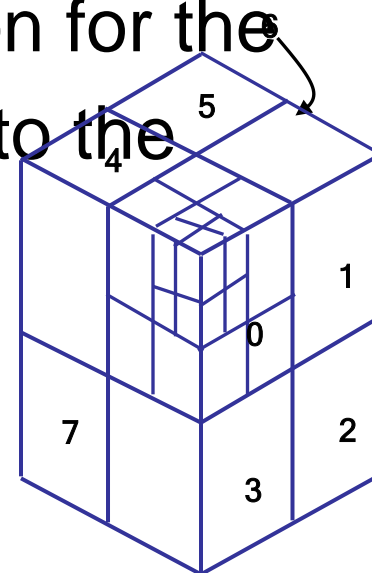
Characteristics

- Extension of *area-subdivision method*
- Projecting octree nodes onto the viewplane
 - Front-to-back order \leftrightarrow Depth-first traversal
 - The nodes for the front suboctants of octant 0 are visited before the nodes for the four back suboctants
 - The pixel in the framebuffer is assigned that color if no values have previously been stored
 - Only the front colors are loaded



Displaying An Octree

- Map the octree onto a quadtree of visible areas
 - Traversing octree nodes from front to back in a recursive procedure
 - The quadtree representation for the visible surfaces is loaded into the framebuffer



Octants in Space

Ray-Casting Method

Characteristics

- Based on geometric optics methods
 - Trace the paths of light rays
 - Line of sight from a pixel position on the viewplane through a scene
 - Determine which objects intersect this line
 - Identify the visible surface whose intersection point is closest to the pixel
 - Infinite number of light rays
 - Consider only rays that pass through pixel positions
 - Trace the light-ray paths backward from the pixels
- Effective visibility-detection method
 - For scenes with curved surfaces

Image-Space Method vs. Object-Space Method

- Image-Space Method
 - Depth-Buffer Method
 - A-Buffer Method
 - Scan-Line Method
 - *Area-Subdivision Method*
- ◆ Object-Space Method
 - Back-Face Detection
 - BSP-Tree Method
 - *Area-Subdivision Method*
 - Octree Methods
 - Ray-Casting

Curved Surfaces

Abstract

- Effective methods for curved surfaces
 - Ray-casting
 - Octree methods
- Approximate a curved surface as a set of plane, polygon surfaces
 - Use one of the other hidden-surface methods
 - More efficient as well as more accurate than using ray casting and the curved-surface equation

Curved-Surface Representations

- Implicit equation of the form

$$f(x, y, z) = 0$$

- Parametric representation
- Explicit surface equation

$$z = f(x, y)$$

- Useful for some cases
 - A height function over an xy ground plane
- Scan-line and ray-casting algorithms
 - Involve numerical approximation techniques

Surface Contour Plots

- Display a surface function with a set of contour lines that show the surface shape
 - Useful in math, physics, engineering, ...
- With an explicit representation
 - Plot the visible-surface contour lines
 - To obtain an xy plot $\rightarrow z = f(x, y)$
 - Plotted for values of z
 - Using a specified interval Δz



<Color-coded surface contour plot>

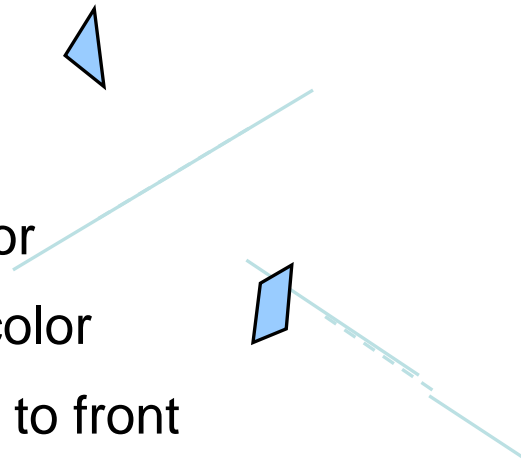
Wireframe Methods

Characteristics

- In wireframe display
 - Visibility tests are applied to surface edges
 - Visible edge sections are displayed
 - Hidden edge sections can be eliminated or displayed differently from the visible edges
- Procedures for determining visibility of edges
 - Wireframe-visibility (Visible-line detection, Hidden-line detection) methods

Wireframe Visibility Methods

- Compare each line to each surface
 - Direct approach to identifying the visible lines
 - Depth values are compared to the surfaces
 - Use coherence methods
 - No actual testing each coordinate
- With depth-sorting
 - Interiors are in the background color
 - Boundaries are in the foreground color
 - Processing the surfaces from back to front
 - Hidden lines are erased by the nearer surfaces



Comparison(1 / 2)

- Back-face detection methods
 - Fast and effective as an initial screening
 - Eliminate many polygons from further visibility tests
 - In general, this can't completely identify all hidden surfaces
- Depth-buffer(z-buffer) method
 - Fast and simple
 - Two buffers
 - Refresh buffer for the pixel intensities
 - Depth buffer for the depth of the visible surface

Comparison(2 / 2)

- A-buffer method
 - An improvement on the depth-buffer approach
 - Additional information
 - Antialiased and transparent surfaces
- Other visible-surface detection schemes
 - Scan-line method
 - Depth-sorting method(painter's algorithm)
 - BSP-tree method
 - Area subdivision method
 - Octree methods
 - Ray casting

References

- Donald Hearn, M. Pauline Baker(Computer Graphics)
- Junglee graphics laboratory, korea university