Computer Graphics

(SE Computer Engineering 2019-course)

Introduction

Raise your words, not your voice.
It is rain that grows flowers,
not thunder.

Dr. Nuzhat F. Shaikh, Wadia College of Engineering, Pune

Savitribai Phule Pune University Second Year of Computer Engineering (2019 Course) 210244: Computer Graphics		
Teaching Scheme:	**Credit**	**Examination Scheme:**
TH: 03 Hours/Week	**03**	**Mid_Semester(TH): 30 Marks** **End_Semester(TH): 70 Marks**
Prerequisite Courses, if any: Basic Mathematics		
Companion Course, if any: OOP, OOP&CG Lab		

210247: OOP and Computer Graphics Laboratory		
Teaching Scheme **Practical : 04 Hours/Week**	**Credit 02**	**Examination Scheme:** **Term Work: 25 Marks** **Practical: 25Marks**
Companion Course, if any: 210243: **OOP**, 210244: **Computer Graphics**		

Course Objectives:

The Computer Graphics course prepares students for activities involving the design, development and testing of modeling, rendering, and animation solutions to a broad variety of problems found in entertainment, sciences, and engineering.

- Remembering: To acquaint the learner with the basic concepts of Computer Graphics.
- Understanding: To learn the various algorithms for generating and rendering graphical figures.
- Applying: To get familiar with mathematics behind the graphical transformations.
- Understanding: To understand and apply various methods and techniques regarding projections, animation, shading, illumination and lighting.
- Creating: To generate Interactive graphics using OpenGL.

Course Outcomes:

On completion of the course, learner will be able to–

CO1: Identify the basic terminologies of Computer Graphics and interpret the mathematical foundation of the concepts of computer graphics.

CO2: Apply mathematics to develop Computer programs for elementary graphic operations.

CO3: Describe the concepts of windowing and clipping and apply various algorithms to fill and clip polygons.

CO4: Understand and apply the core concepts of computer graphics, including transformation in two and three dimensions, viewing and projection.

CO5: Understand the concepts of color models, lighting, shading models and hidden surface elimination.

CO6: Describe the fundamentals of and implement curves, fractals, animation and gaming.

Learning Resources

Text Books: **1. S. Harrington, "Computer Graphics", 2nd Edition, McGraw-Hill Publications, 1987, ISBN 0 – 07 – 100472 – 6.** 2. Donald D. Hearn and Baker, "Computer Graphics with OpenGL", 4th Edition, ISBN-13: 9780136053583. 3. D. Rogers, "Procedural Elements for Computer Graphics", 2nd Edition, Tata McGraw-Hill Publication, 2001, ISBN 0 – 07 – 047371 – 4
Reference Books: 1. J. Foley, V. Dam, S. Feiner, J. Hughes, "Computer Graphics Principles and Practice", 2nd Edition, Pearson Education, 2003, ISBN 81 – 7808 – 038 – 9. 2. D. Rogers, J. Adams, "Mathematical Elements for Computer Graphics", 2nd Edition, Tata McGraw Hill Publication, 2002, ISBN 0 – 07 – 048677 – 8.

Learning Resources contd...

e-Books:

- https://open.umn.edu/opentextbooks/textbooks/introduction-to-computer-graphics
- http://www2.cs.uidaho.edu/~jeffery/courses/324/lecture.html

MOOC/ Video Lectures available at:

- https://nptel.ac.in/courses/106/106/106106090/
- https://nptel.ac.in/courses/106/102/106102065/

UNIT I

Unit	Lect.	Content details as per syllabus
I Graphics Primitives and Scan Conversion Algorithms	1	Introduction, graphics primitives - pixel, resolution, aspect ratio, frame buffer. Display devices, applications of computer graphics.
	2,3	Introduction to OpenGL - OpenGL architecture, primitives and attributes, simple modelling and rendering of two- and three-dimensional geometric objects, GLUT, interaction, events and call-backs picking.(Simple Interaction with the Mouse and Keyboard)
	4	Scan conversion: Line drawing algorithms: Digital Differential Analyzer (DDA),
	5	Scan conversion: Line drawing algorithms: Bresenham.
	6,7	Scan conversion: Circle drawing algorithms: DDA, Bresenham, and Midpoint.
Exemplar/Case Studies		Study about OpenGL Architecture Review Board (ARB)
Course Outcomes		CO1, CO2

UNIT II

Unit	Lect.	Content details as per syllabus
II Polygon, Windowing and Clipping	1	**Polygons: Introduction to polygon, types: convex, concave and complex. Inside test.**
	2	**Polygon Filling: flood fill, seed fill, scan line fill.**
	3	**scan line fill.**
	4	**Windowing and clipping: viewing transformations,**
	5	**2-D clipping:** **Cohen – Sutherland algorithm - line Clipping algorithm,**
	6	**Sutherland Hodgeman Polygon clipping algorithm,**
	7	**Weiler Atherton Polygon Clipping algorithm.**
Exemplar/Case Studies		**Study Guard Band Clipping Technique and its use in various rendering softwares, use of 3d pipeline/polygon modelling and applications**
Course Outcomes		**CO2, CO3**

Course Outcomes:

On completion of the course, learner will be able to–

CO1: Identify the basic terminologies of Computer Graphics and interpret the mathematical foundation of the concepts of computer graphics.

CO2: Apply mathematics to develop Computer programs for elementary graphic operations.

CO3: Describe the concepts of windowing and clipping and apply various algorithms to fill and clip polygons.

CO4: Understand and apply the core concepts of computer graphics, including transformation in two and three dimensions, viewing and projection.

CO5: Understand the concepts of color models, lighting, shading models and hidden surface elimination.

CO6: Describe the fundamentals of and implement curves, fractals, animation and gaming.

Basic Terminologies

- **Pixel**
- **Aspect Ratio**
- **Resolution**
- **Frame Buffer**
- Raster Display
- Screen Space
- Object Space
- Refresh Rate

monochrome display:
512 x 512 x 1bit (bit is either 0=off, or 1=on.)
total -32768 bytes

8 bit grey-scale display:
512 x 512 x 8bit
Each of the 512 x 512 pixels can be one of 256 shades of grey (from black to white.)
total -262,144 bytes

7680 x 4320 pixels
About 8K **Resolution**:
8K **resolution** measures in at 7680 x 4320 pixels and is currently the **highest monitor resolution** currently available.

24 bit color display:
512 x 512 x 24bit (each pixel has 8 bits for red, 8 bits for green, and 8 bits for blue.)
total - 786,432 bytes

Recommended resolution & aspect ratios
For the default 16:9 aspect ratio, encode at these resolutions:
2160p: 3840x2160
1440p: 2560x1440
1080p: 1920x1080
720p: 1280x720
480p: 854x480
360p: 640x360
240p: 426x240

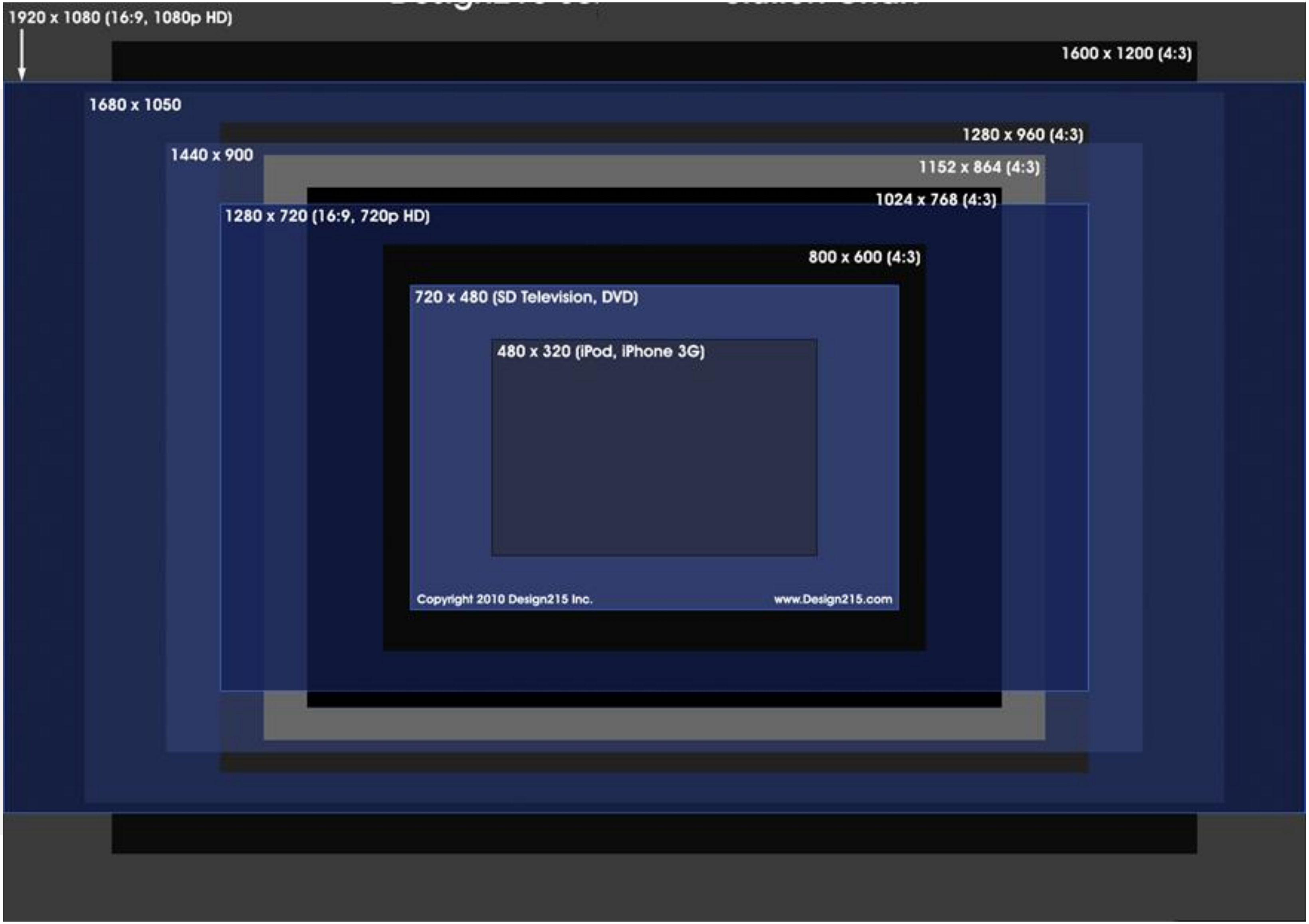
Basic Terminologies

- **Pixel**
- **Aspect Ratio**
- **Resolution**
- **Frame Buffer**
- Screen Space
- Object Space
- Refresh Rate

monochrome display:
512 x 512 x 1bit (bit is either 0=off, or 1=on.)
total -32768 bytes

8 bit grey-scale display:
512 x 512 x 8bit
Each of the 512 x 512 pixels can be one of 256 shades of grey (from black to white.)
total -262,144 bytes

24 bit color display:
512 x 512 x 24bit (each pixel has 8 bits for red, 8 bits for green, and 8 bits for blue.)
total - 786,432 bytes


1920 x 1080 (16:9, 1080p HD)
1600 x 1200 (4:3)
1680 x 1050
1280 x 960 (4:3)
1440 x 900
1152 x 864 (4:3)
1024 x 768 (4:3)
1280 x 720 (16:9, 720p HD)
800 x 600 (4:3)
720 x 480 (SD Television, DVD)
480 x 320 (iPod, iPhone 3G)
Copyright 2010 Design215 Inc.
www.Design215.com

Computer Graphics

(SE Computer Engineering 2019-course)

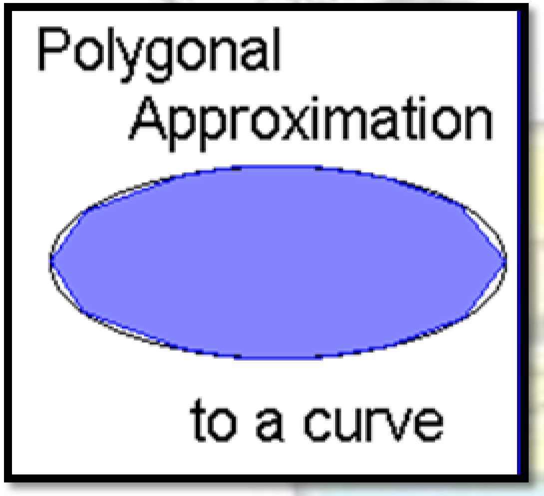
UNIT II Polygon, Windowing and Clipping

Dr. Nuzhat F. Shaikh, Wadia College of Engineering, Pune

Polygons

- What is a polygon?
- How to represent a polygon?
- Types of Polygons.
- Inside – Outside test for a point
  - Even-odd test
  - Winding no. test

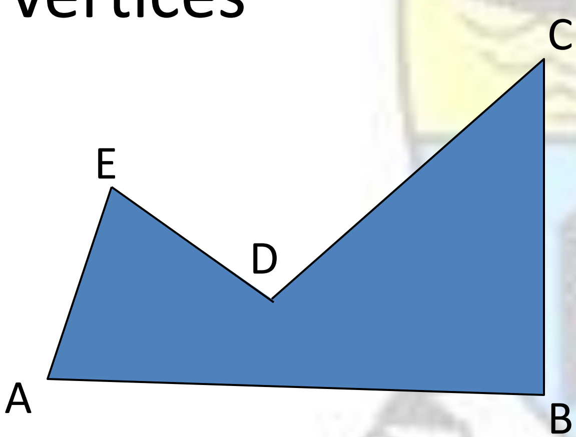
Why Polygons???

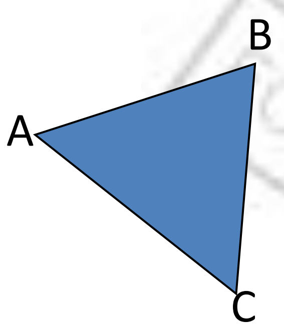


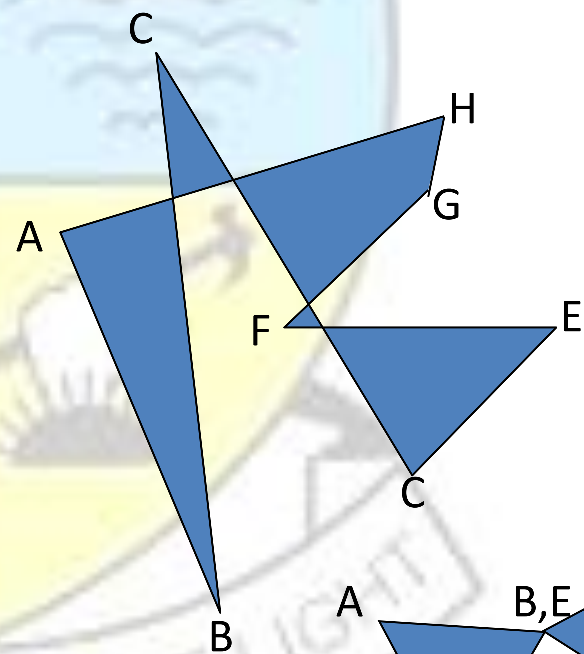
Types of Polygons

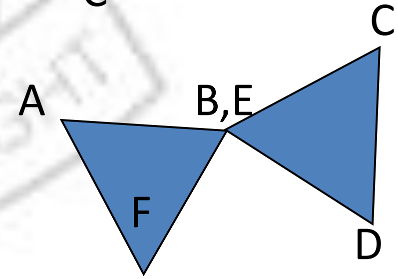
Polygons: Complex vs Simple

- A simple polygon – edges only intersect at vertices, no coincident vertices
- A complex polygon – edges intersect and/or coincident vertices



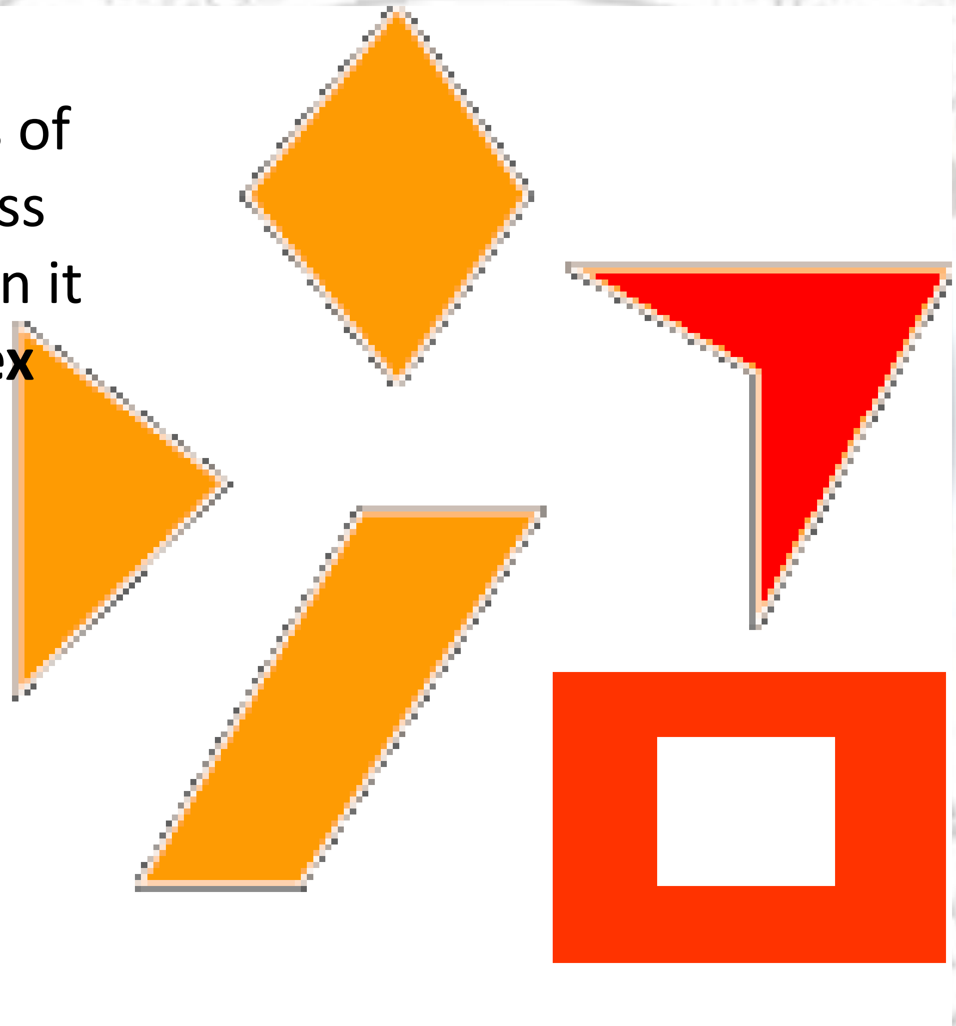






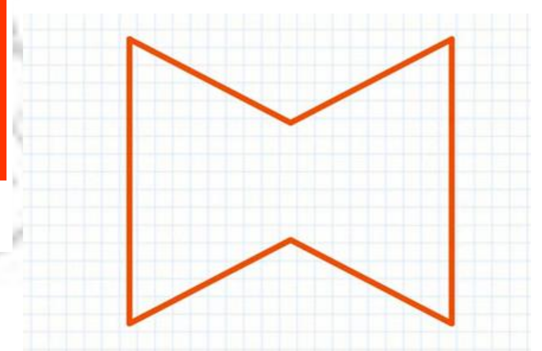
Types of Polygons

Convex polygon:
If each of the interior angles of a polygon is less than 180°, then it is called **convex polygon**.



Concave polygon:
If at least one angle of a polygon is more than 180°, then it is called a **concave polygon**.

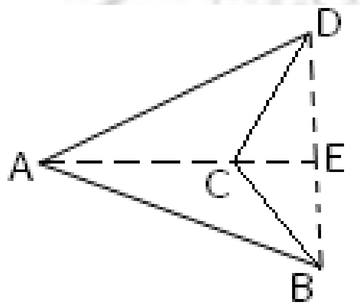
CONCAVE



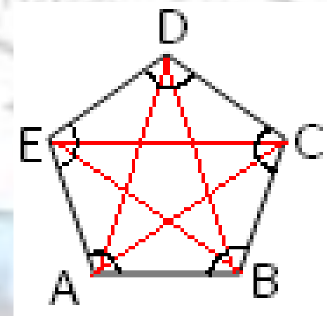
CONVEX

Classify the following as convex or concave

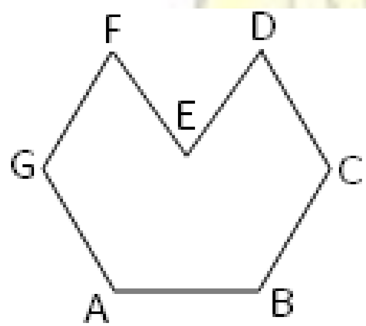
1.



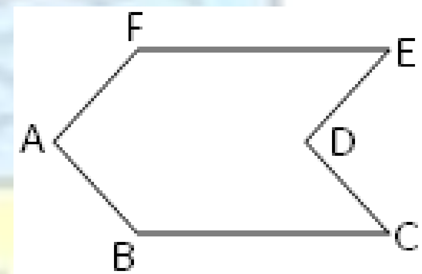
2.



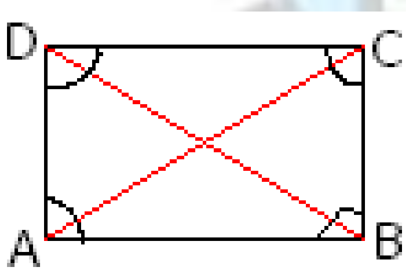
3.



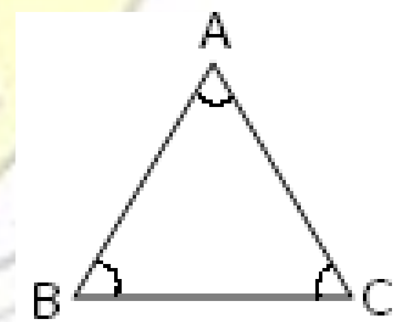
4.



5.



6.

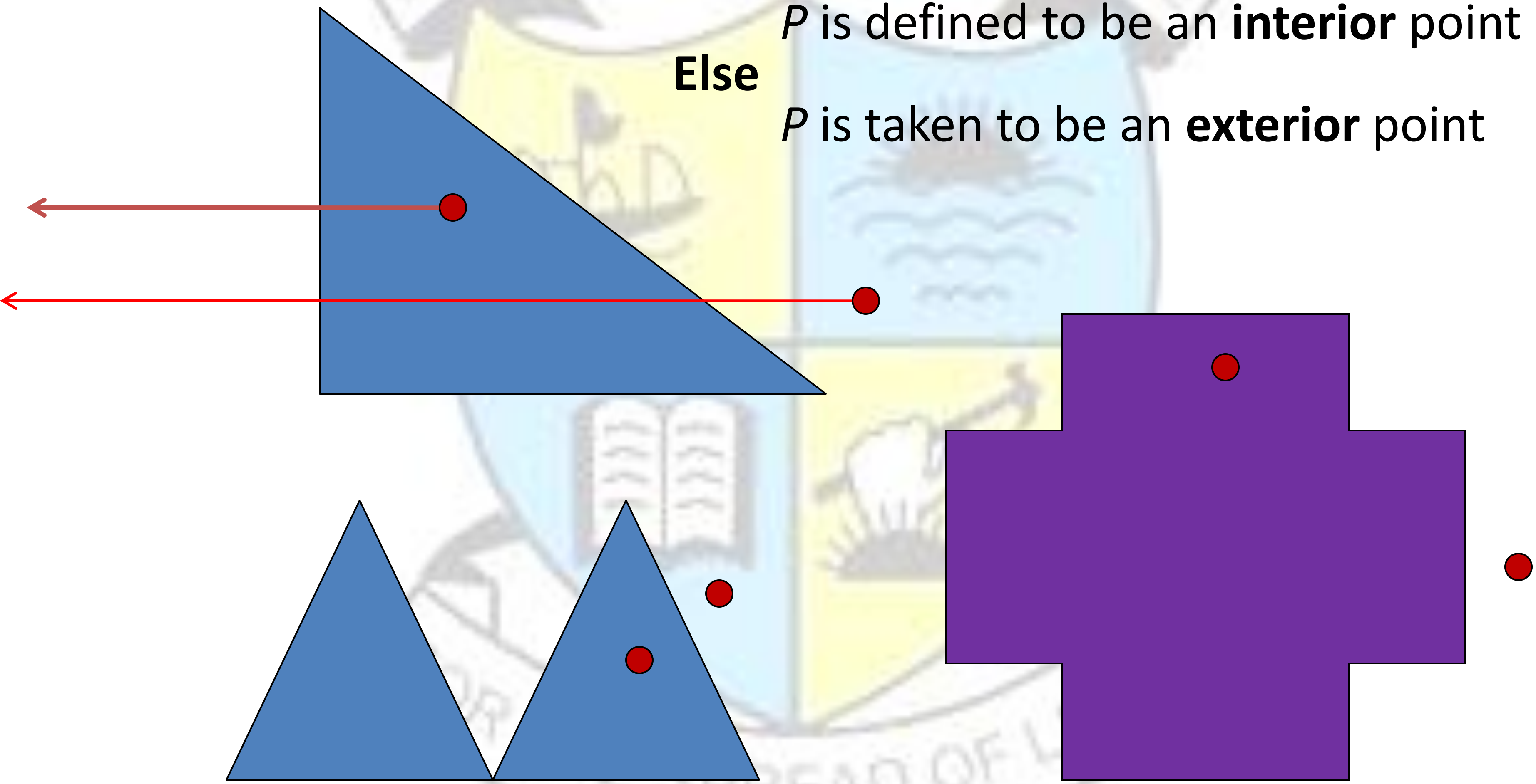


Convex…2,5,6 Concave…1,3,4

Inside – outside Test

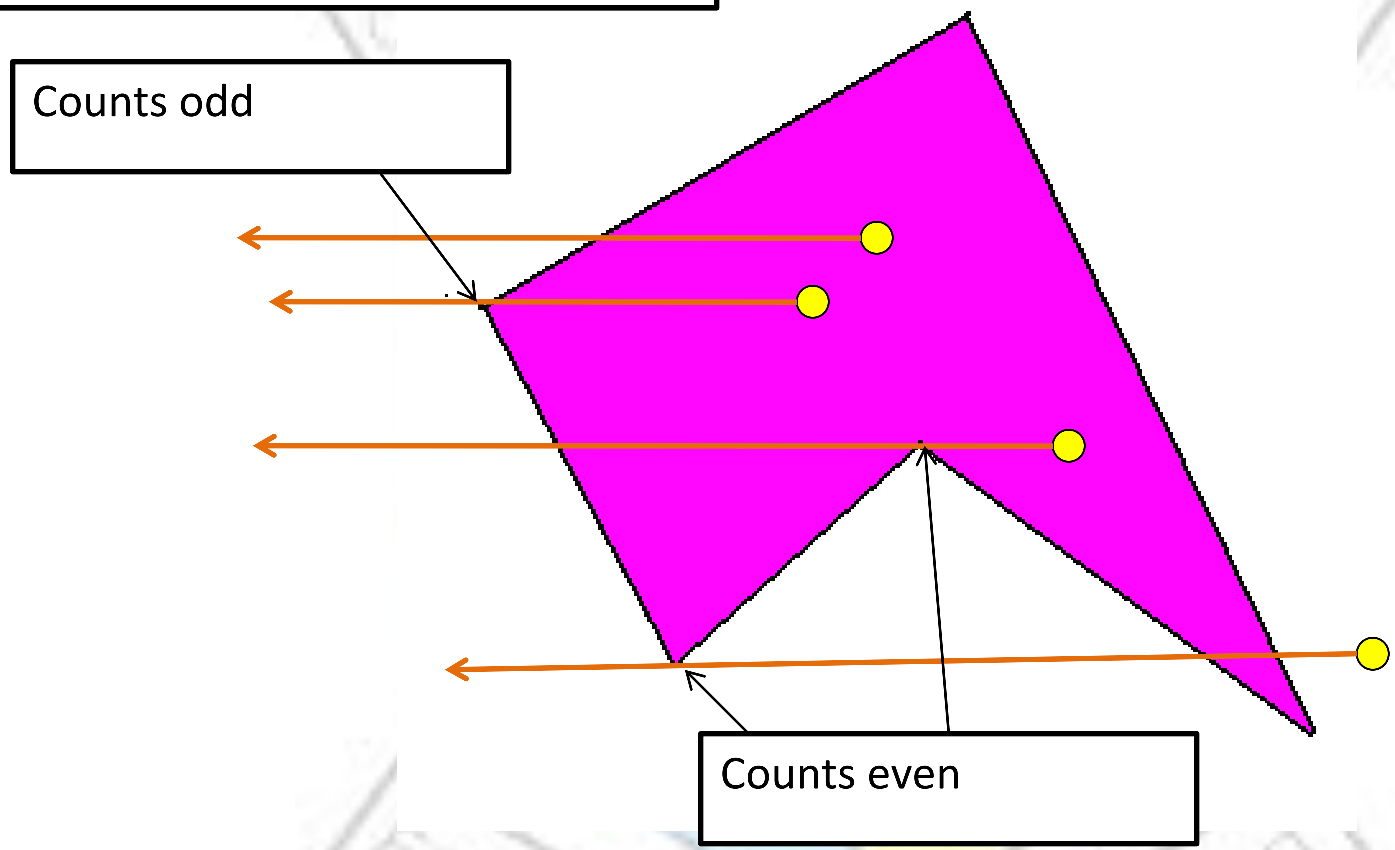
1. Even-odd

If the number of intersections is **odd**, then
P is defined to be an **interior** point
Else
P is taken to be an **exterior** point



Inside – outside Test

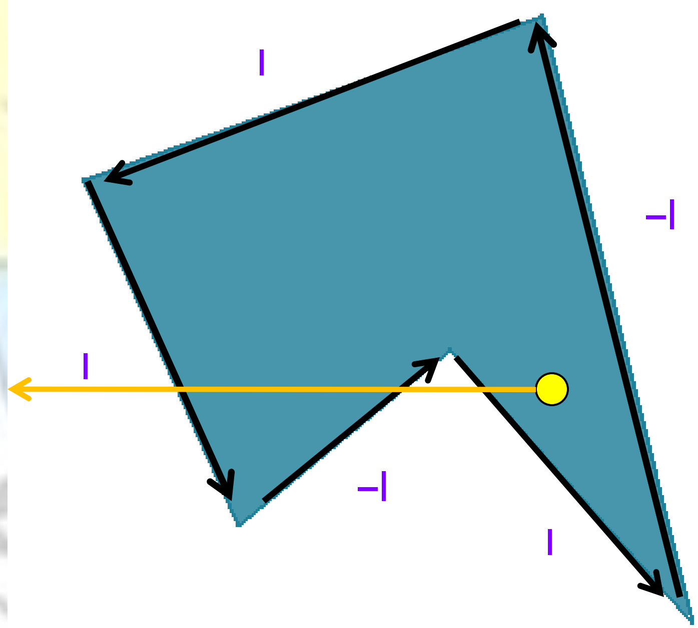
Vertices as special cases



If the number of intersections is **odd**, then
P is defined to be an **interior** point
Else
P is taken to be an **exterior** point

Inside – outside Test

2. Winding Number Test

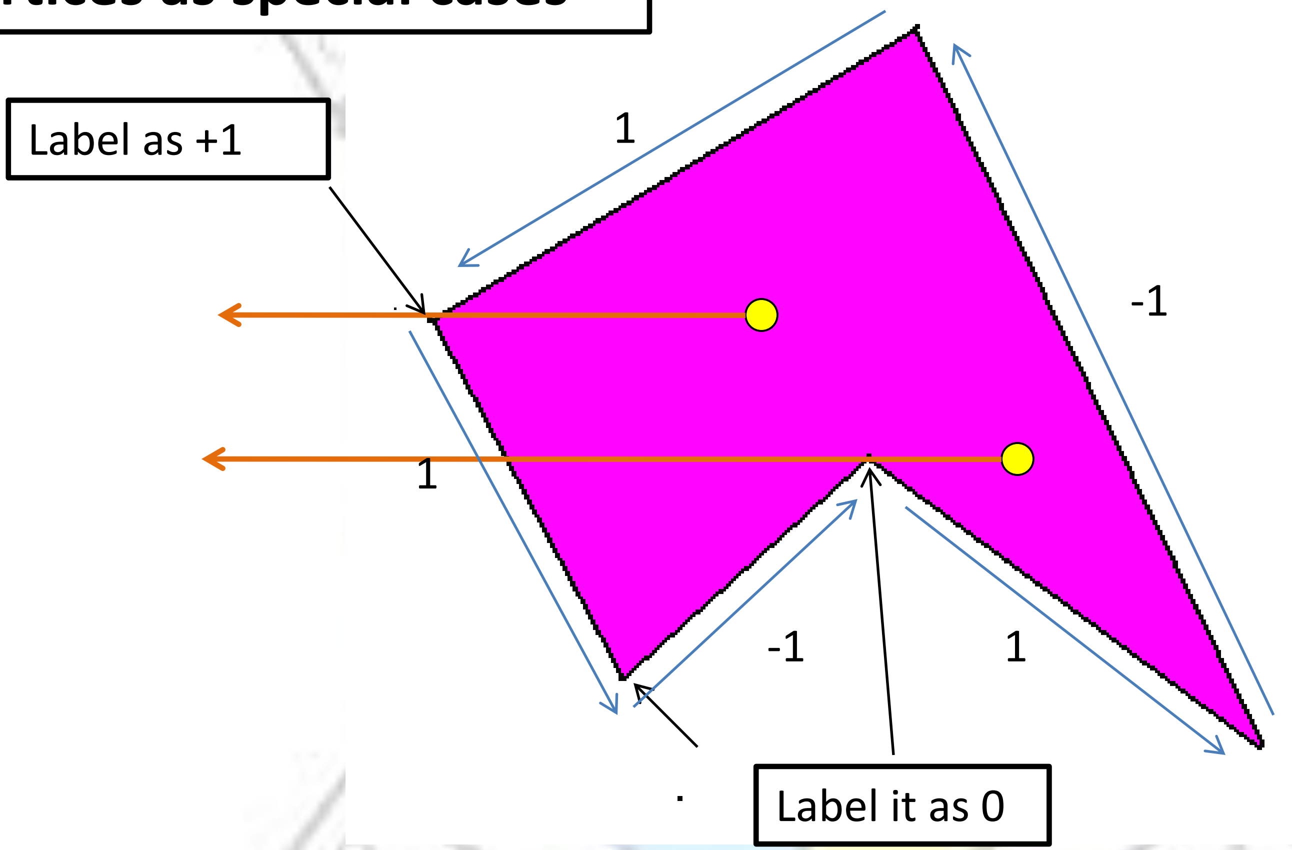


If the winding number is **nonzero**, then
P is defined to be an **interior** point
Else
P is taken to be an **exterior** point

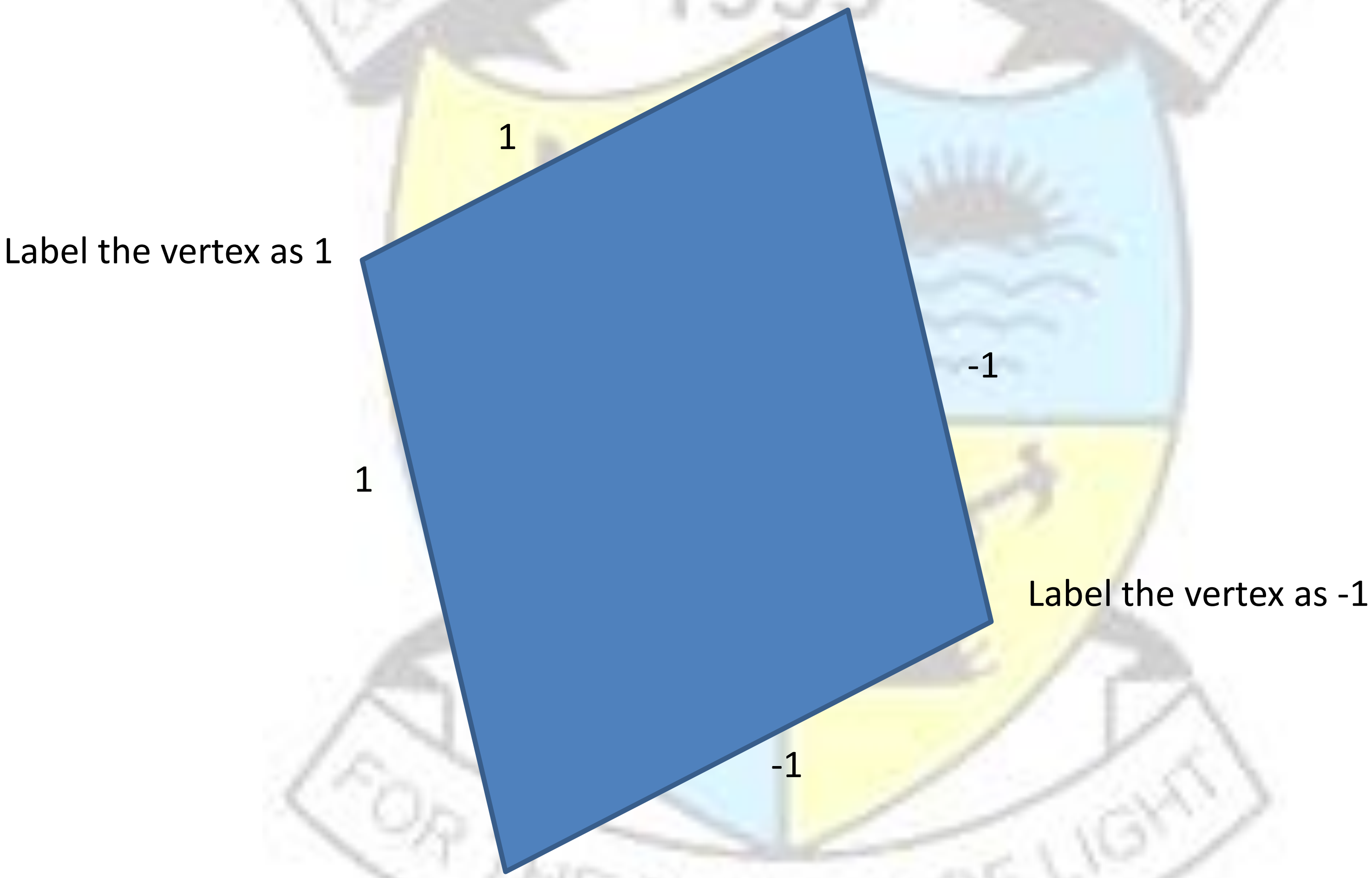
Inside – outside Test

2. Winding Number Test

Vertices as special cases



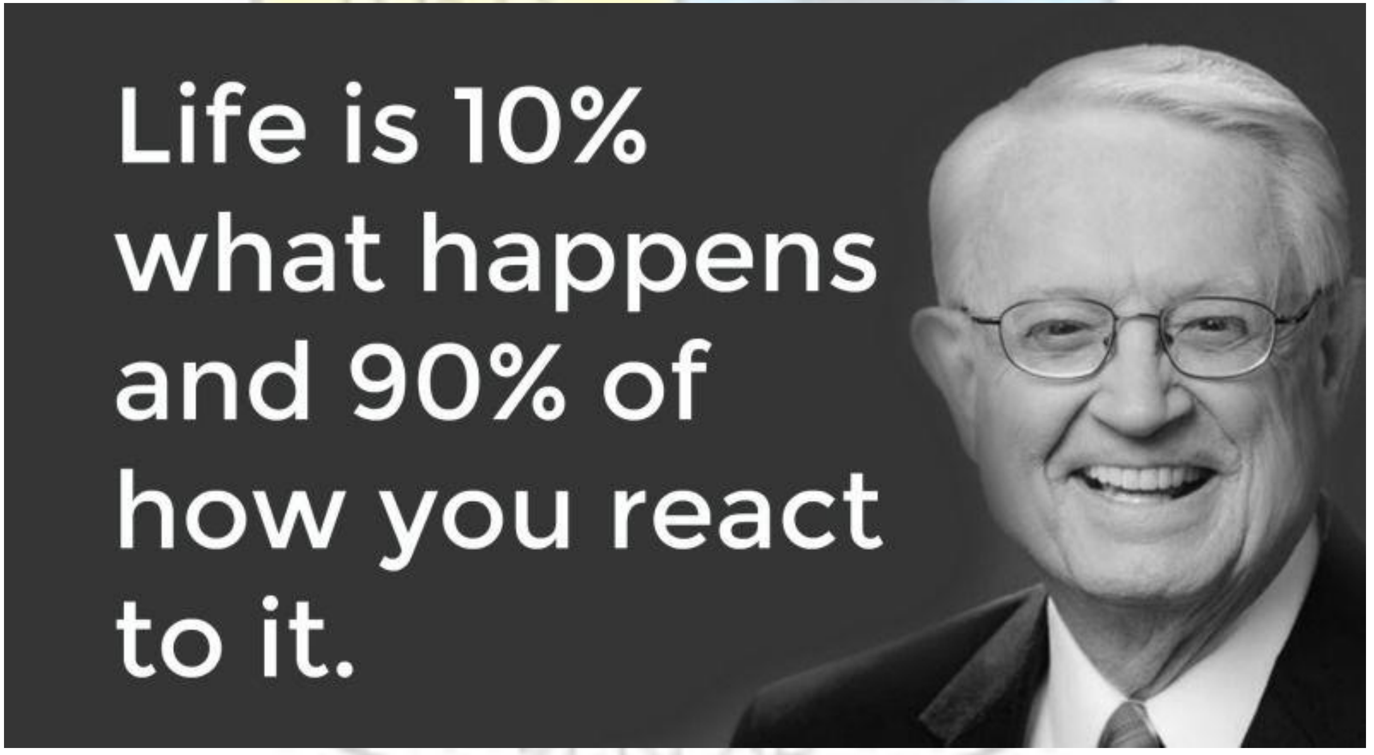
If the winding number is **nonzero**, then
P is defined to be an **interior** point
Else
P is taken to be an **exterior** point


1
Label the vertex as 1
-1
1
Label the vertex as -1
-1

Computer Graphics

(SE Computer Engineering 2019-course)

UNIT II Polygon Filling



UNIT II

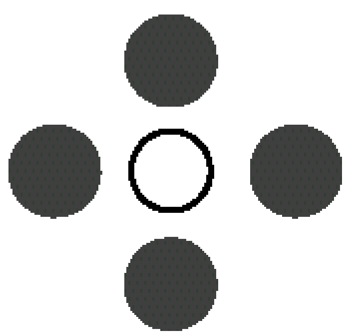
Unit	Lect.	Content details as per syllabus
II Polygon, Windowing and Clipping	1	**Polygons: Introduction to polygon, types: convex, concave and complex. Inside test.**
	2	**Polygon Filling: flood fill, seed fill, scan line fill.**
	3	**scan line fill.**
	4	**Windowing and clipping: viewing transformations,**
	5	**2-D clipping:** **Cohen – Sutherland algorithm line Clipping algorithm,**
	6	**Sutherland Hodgeman Polygon clipping algorithm,**
	8	**Weiler Atherton Polygon Clipping algorithm.**
Exemplar/Case Studies		**Study Guard Band Clipping Technique and its use in various rendering softwares, use of 3d pipeline/polygon modelling and applications**
Course Outcomes		**CO2, CO3**

Region (Polygon) Filling

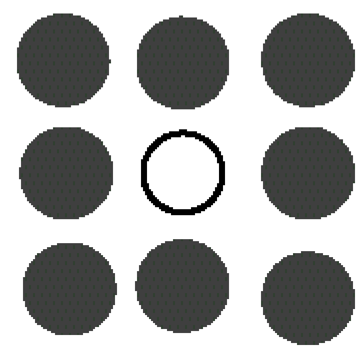
- Given the edges defining a polygon, a color for the polygon, we need to fill all the pixels inside the polygon.
- 2 classes of algorithms
- Seed Fill Approach
  - 2 algorithms: Boundary Fill and Flood Fill
  - works at pixel level
  - suitable for interactive painting applications
- Scanline Fill Approach
  - works at polygon level
  - better performance

Seed Fill Algorithms: Connectedness

- 4-connected region: From a given pixel, the region that you can get to by a series of 4 way moves (N, S, E and W)
- 8-connected region: From a given pixel, the region that you can get to by a series of 8 way moves (N, S, E, W, NE, NW, SE, and SW)



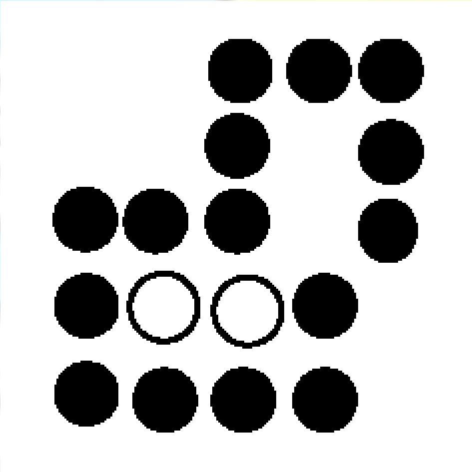
4-connected



8-connected

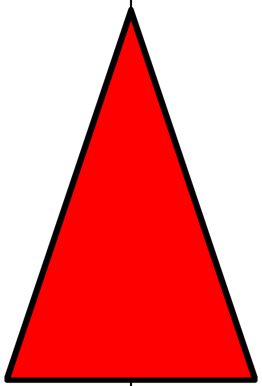
Boundary Fill Algorithm

- Start at a point inside a region
- Paint the interior outward to the edge
- The edge must be specified in a single color
- Fill the 4-connected or 8-connected region



Boundary Fill Algorithm (cont.)

```
void BoundaryFill4(int x, int y,
                          color newcolor, color edgecolor)
{
  int current;
  current = ReadPixel(x, y);
  if(current != edgecolor && current != newcolor)
  {
      BoundaryFill4(x+1, y, newcolor, edgecolor);
      BoundaryFill4(x-1, y, newcolor, edgecolor);
      BoundaryFill4(x, y+1, newcolor, edgecolor);
      BoundaryFill4(x, y-1, newcolor, edgecolor);
  }
}
```



```
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
       getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill4(x + 1, y, fill_color, boundary_color);
        boundaryFill4(x, y + 1, fill_color, boundary_color);
        boundaryFill4(x - 1, y, fill_color, boundary_color);
        boundaryFill4(x, y - 1, fill_color, boundary_color);
    }
}
```

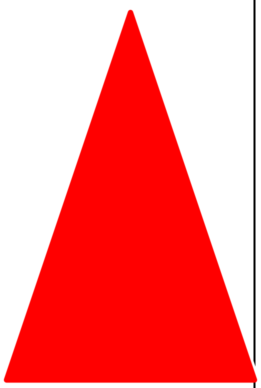
```
void boundaryFill8(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
       getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill8(x + 1, y, fill_color, boundary_color);
        boundaryFill8(x, y + 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y, fill_color, boundary_color);
        boundaryFill8(x, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
    }
}
```

Flood Fill Algorithm

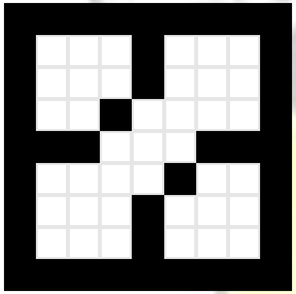
- Used when an area defined with multiple color boundaries
- Start at a point inside a region
- Replace a specified interior color (old color) with fill color
- Fill the 4-connected or 8-connected region until all interior points being replaced

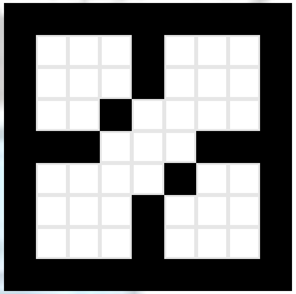
Flood Fill Algorithm (cont.)

```
void FloodFill4(int x, int y, color newcolor, color oldColor)
{
  if(ReadPixel(x, y) == oldColor)
  {
      FloodFill4(x+1, y, newcolor, oldColor);
      FloodFill4(x-1, y, newcolor, oldColor);
      FloodFill4(x, y+1, newcolor, oldColor);
      FloodFill4(x, y-1, newcolor, oldColor);
  }
}
```



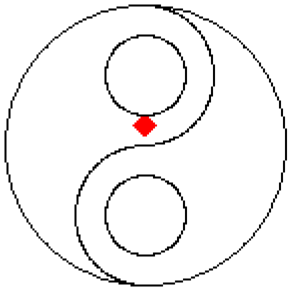
Flood Fill Algorithm (cont.)

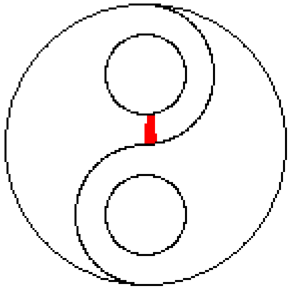




4 connected regions

8 connected regions





Four-way flood fill using a queue for storage

Four-way flood fill using a stack for storage

Computer Graphics

(SE Computer Engineering 2019-course)

UNIT II Polygon Filling

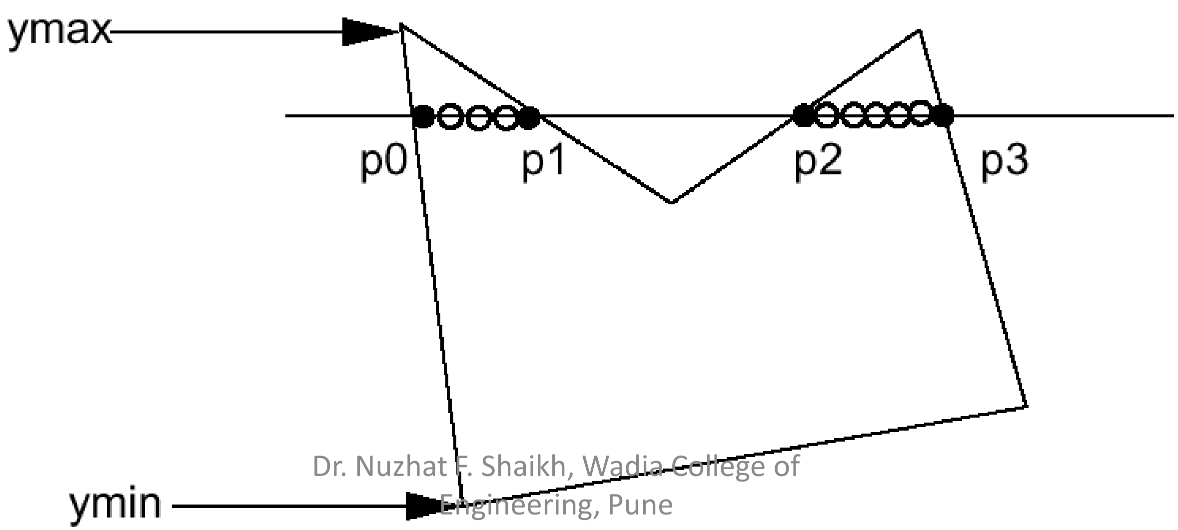
"For every minute you are angry you lose sixty seconds of happiness."

— *Ralph Waldo Emerson*

Dr. Nuzhat F. Shaikh, Wadia College of Engineering, Pune

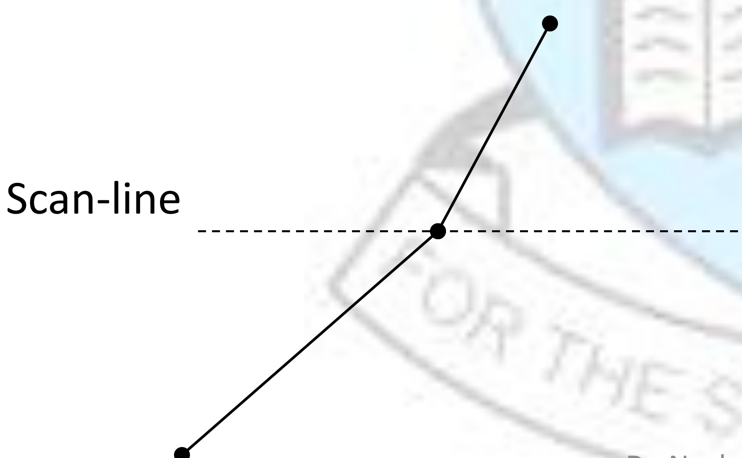
Scanline Fill Algorithm

- Intersect scanline with polygon edges
- Fill between pairs of intersections
- Basic algorithm:
  For y = ymin to ymax
  1) intersect scanline y with each edge
  2) sort intersections in increasing x [p0,p1,p2,p3]
  3) fill pairwise (p0->p1, p2->p3, ….)
- Advantage of scan-line fill: It fills in the same order as rendering, and so can be pipelined.

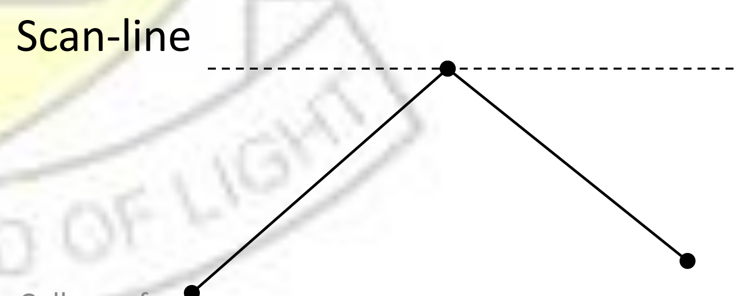


Scan Line Fill: What happens at edge end-point?

- Edge endpoint is duplicated.
- In other words, when a scan line intersects an edge endpoint, it intersects two edges.
- Two cases:
  - Case A: edges are monotonically increasing or decreasing
  - Case B: edges reverse direction at endpoint
- In Case A, we should consider this as only ONE edge intersection
- In Case B, we should consider this as TWO edge intersections



Case A



Case B

Speeding up Scan-Line Algorithm

1. Parallel algorithm: process each scan line in one processor. Each scan line is independent

2. From edge intersection with one scan line, derive edge intersection with next scan line (see next slide)

3. Active edge list (see later slides)

Use coherence property from scan line to scan line .

Method 2: Derive next intersection

- Suppose that slope of the edge is

  $m = \Delta y/\Delta x$

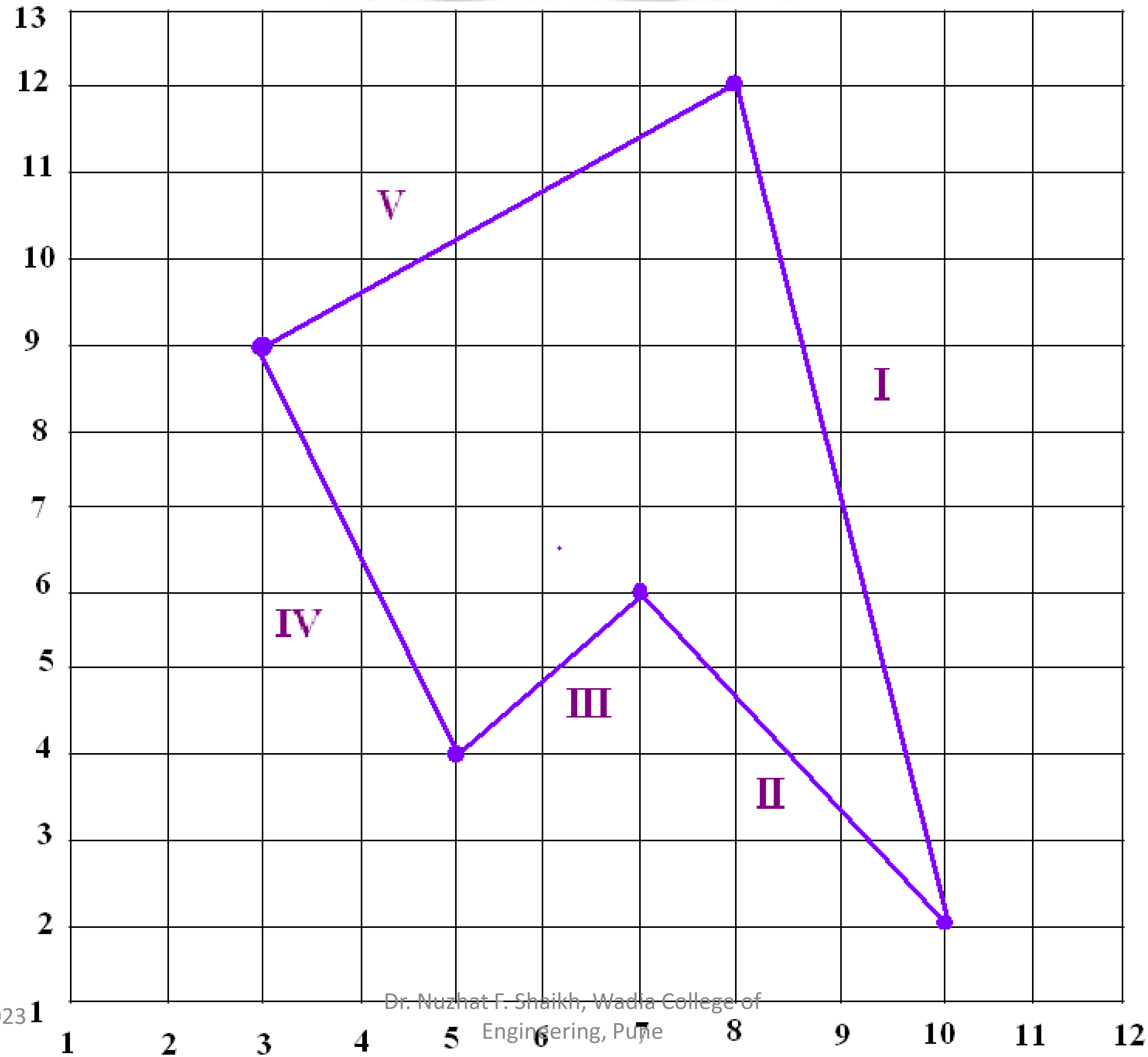
- Let x_k be the x intercept of the current scan line, and x_{k+1} be the x intercept of the next scan line, then

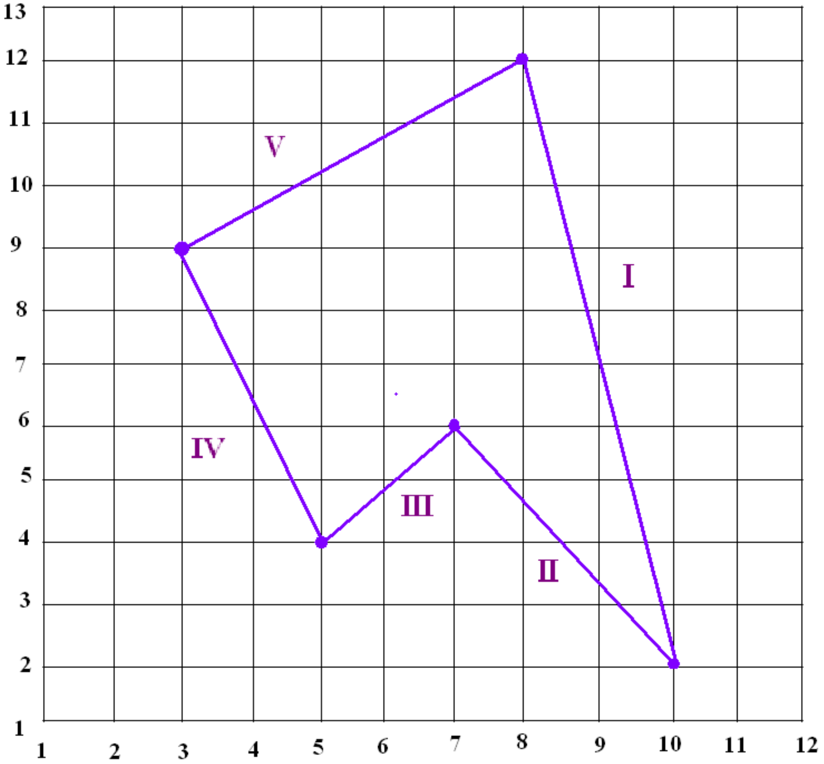
  $x_{k+1} = x_k + \Delta x/\Delta y$

  $x_{k+1} = x_k + 1/m$

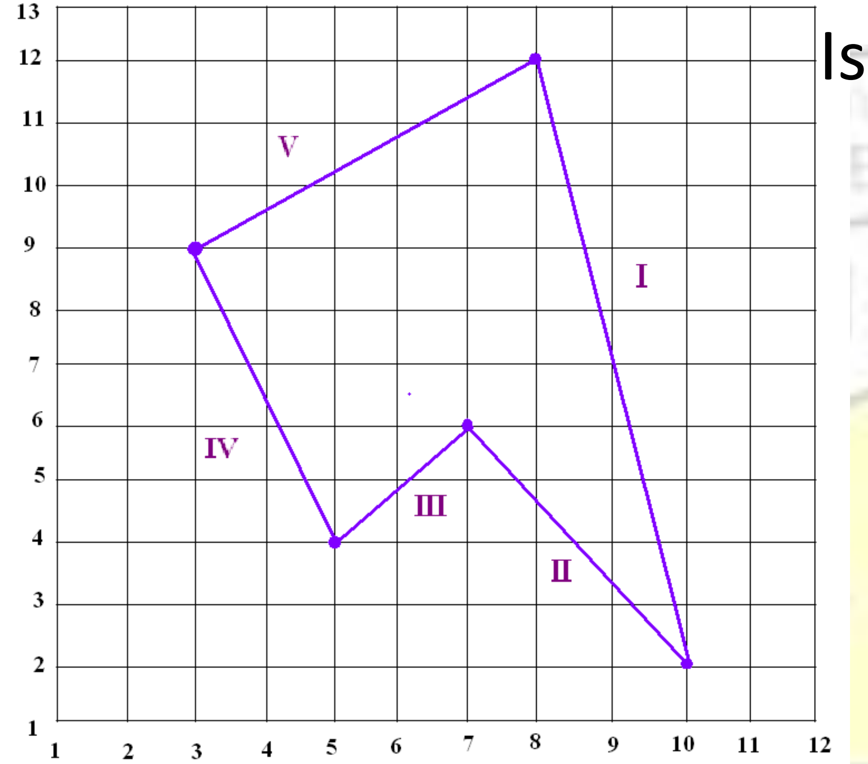
Active Edge List

- Start with the sorted edge table.
  - In this table, there is an entry for each edge.
  - Add only the non-horizontal edges into the table.
    - For each edge entry, store (1) the x-intercept with the scan-line, (2) the two y-values of the edge, and (3) the inverse of the slope.(-1/m)
  - Each scan-line entry thus contains a sorted list of edges. The edges are sorted top to bottom (hence -1/m).
- Next, we process the scan lines from top to bottom.
  - **We maintain an active edge list for the current scan-line.**
  - **The active edge list contains all the edges crossed by that scan line. As we move down, update the active edge list by the sorted edge table if necessary.**
  - **Use iterative coherence calculations to obtain edge intersections quickly.**


13
12
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
I
II
III
IV
V



	X1	Y1	X2	Y2
I	8	12	10	2
II	10	2	7	6
III	7	6	5	4
IV	5	4	3	9
V	3	9	8	12



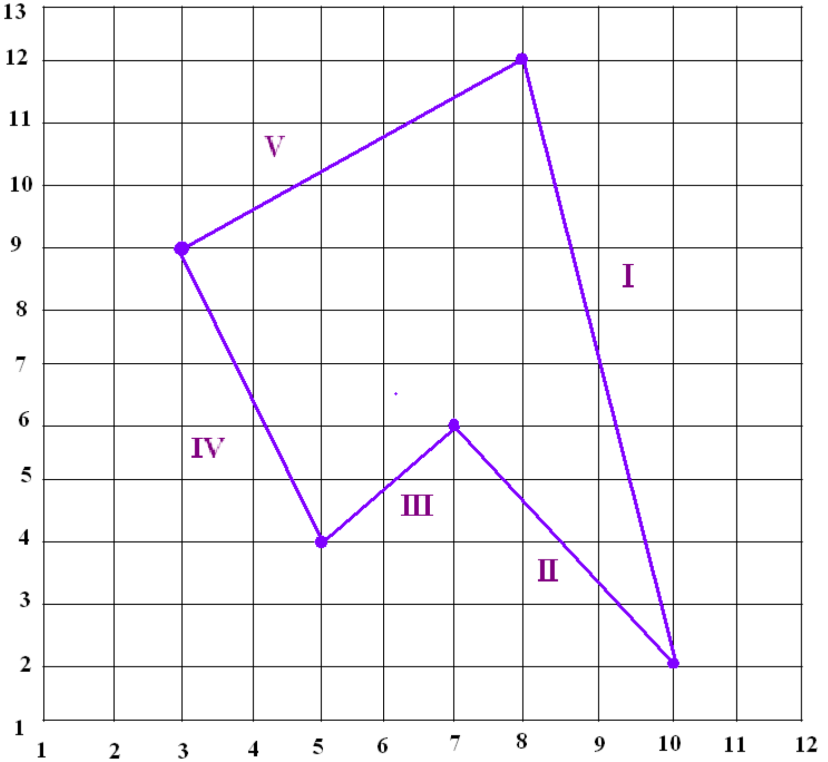
Is y1> y2?

Yes don’t do anything

No. Interchange x1,y1 and x2,y2

so that each edge is represented as starting from higher y value to lower y value

	X1	Y1	X2	Y2
I	8	12	10	2
II	10	2	7	6
III	7	6	5	4
IV	5	4	3	9
V	3	9	8	12



- All y1s are greater than y2

	X1	Y1	X2	Y2
I	8	12	10	2
II	7	6	10	2
III	7	6	5	4
IV	3	9	5	4
V	8	12	3	9

- Sort on y1
- For ease of finding active edges

	X1	Y1	X2	Y2
I	8	12	10	2
II	7	6	10	2
III	7	6	5	4
IV	3	9	5	4
V	8	12	3	9

- Sorted on y1

	X1	Y1	X2	Y2
I	8	12	10	2
V	8	12	3	9
IV	3	9	5	4
II	7	6	10	2
III	7	6	5	4

- Find slope and store -1/m

	X1	Y1	Y2	-1/m
I	8	12	2	1/5
V	8	12	9	-5/3
IV	3	9	4	2/5
II	7	6	2	3/4
III	7	6	4	-1

- Find slope and store -1/m

	X1	Y1	Y2	-1/m
I	8	12	2	0.2
V	8	12	9	-1.66
IV	3	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Consider one scan line at a time from y(max) to <y(min) decrementing by 1 at each iteration
- y(max) = 12 and y(min) = 3

Iteration 1: y = 12

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and V

Find x2=x1+(-1/m)

For I 8+0.2=8.2 (f)

For V 8-1.66=6.34(c)

Pair up intersecting points as

(7,12) and (8,12)

	X1	Y1	Y2	-1/m
I	8	12	2	0.2
V	8	12	9	-1.66
IV	3	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Iteration 2: y = 11

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and V

Find x2=x1-1/m

For I 8.2+0.2=8.4 (f)

For V 6.34-1.66=4.68 (c)

Pair up intersecting points as (5,11) and (8,11)

	X1	Y1	Y2	-1/m
I	8.2	12	2	0.2
V	6.34	12	9	-1.66
IV	3	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Iteration 3: y = 10

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and V

Find x2=x1-1/m

For I 8.4+0.2=8.6 (f)

For V 4.68-1.66=3.02 (c)

Pair up intersecting points as (4,10) and (8,10)

	X1	Y1	Y2	-1/m
I	8.4	12	2	0.2
V	4.68	12	9	-1.66
IV	3	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Iteration 4: y = 9

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and IV

Find x2=x1-1/m

For I 8.6+0.2=8.8 (f)

For IV 3+0.4=3.4 (c)

Pair up intersecting points as (4,9) and (8,9)

	X1	Y1	Y2	-1/m
I	8.6	12	2	0.2
V	3.02	12	9	-1.66
IV	3	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Iteration 5: y = 8

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and IV

Find x2=x1-1/m

For I 8.8+0.2=9

For IV 3.4+0.4=3.8

Pair up intersecting

points as

(4,8) and (9,8)

- Iteration 6: y = 7

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and IV

Find x2=x1-1/m

For I 9+0.2=9.2

For IV 3.8+0.4=4.2

Pair up intersecting

points as

(5,7) and (9,7)

- Iteration 7: y = 6

for all edges satisfying the condition (y>y2 && y<=y1)

edge I , IV, II and III

Find x2=x1-1/m

For I 9.2+0.2=9.4

For IV 4.2+0.4=4.6

For II 7+0.75=7.75

For III 7-1=6

Pair up intersecting points as

(5,6) and (6,6)

(8,6) and (9,6)

	X1	Y1	Y2	-1/m
I	9.2	12	2	0.2
V	3.02	12	9	-1.66
IV	4.2	9	4	0.4
II	7	6	2	0.75
III	7	6	4	-1

- Iteration 8: y = 5

for all edges satisfying the condition (y>y2 && y<=y1)

edge I , IV, II and III

Find x2=x1-1/m

For I 9.4+0.2=9.6

For IV 4.6+0.4=5

For II 7.75+0.75=8.50

For III 6-1=5

Pair up intersecting points as

(5,5) and (5,5)

(9,5) and (9,5)

	X1	Y1	Y2	-1/m
I	9.6	12	2	0.2
V	3.02	12	9	-1.66
IV	5	9	4	0.4
II	8.5	6	2	0.75
III	5	6	4	-1

- Iteration 9: y = 4

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and II

Find x2=x1-1/m

For I 9.6+0.2=9.8

For II 8.5+0.75=9.25

Pair up intersecting points as

(9 ,4) and (10,4)

	X1	Y1	Y2	-1/m
I	9.8	12	2	0.2
V	3.02	12	9	-1.66
IV	5	9	4	0.4
II	9.25	6	2	0.75
III	5	6	4	-1

- Iteration 10: y = 3

for all edges satisfying the condition (y>y2 && y<=y1)

edge I and II

Find x2=x1-1/m

For I 9.8+0.2=10

For II 9.25+0.75=10

Pair up intersecting points as (10,3) and (10,3)

	X1	Y1	Y2	-1/m
I	10	12	2	0.2
V	3.02	12	9	-1.66
IV	5	9	4	0.4
II	10	6	2	0.75
III	5	6	4	-1

SUMMARY

What is a polygon?

How to represent a polygon?

Types of Polygons.

Inside – Outside test for a point

- Even-odd test
- Winding no. test

Polygon Filling

- Seed Fill Approaches
  - Boundary Fill
  - Flood Fill
- Scan-line Fill Approach

Computer Graphics

(SE Computer Engineering 2019-course)

UNIT II WINDOWING AND CLIPPING

"Do not listen with the intent to reply,

but…

with the intent to understand…"

Dr. Nuzhat F. Shaikh, Wadia College of Engineering, Pune

UNIT II

Unit	Lect.	Content details as per syllabus
II Polygon, Windowing and Clipping	1&2	Polygons: Introduction to polygon, types: convex, concave and complex. Inside test.
	3	Polygon Filling: flood fill, seed fill, scan line fill.
	4&5	scan line fill.
	6	Windowing and clipping: viewing transformations,
	7	2-D clipping: Cohen – Sutherland algorithm line Clipping algorithm,
	8	Sutherland Hodgeman Polygon clipping algorithm,
	9	Weiler Atherton Polygon Clipping algorithm.
Exemplar/Case Studies		Study Guard Band Clipping Technique and its use in various rendering softwares, use of 3d pipeline/polygon modelling and applications
Course Outcomes		CO2, CO3

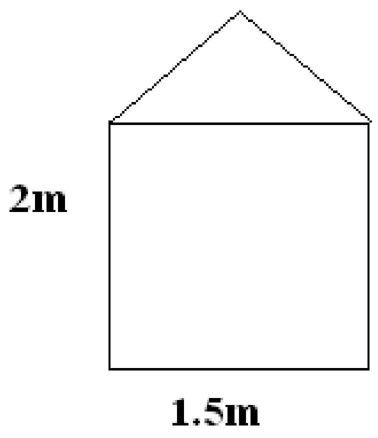
Contents

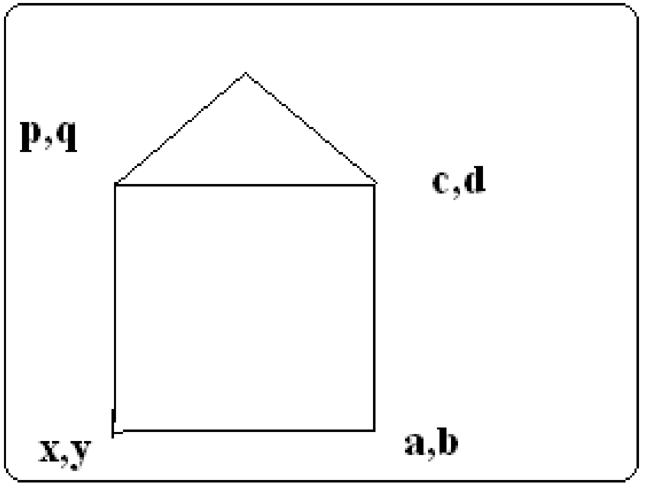
- Windowing Concepts
  - Object Space
  - Image Space
  - Window
  - Viewport
- Viewing Transformations
- Clipping
  - Introduction
  - Brute Force
  - Cohen-Sutherland Line Clipping Algorithm
- Area Clipping
  - Sutherland-Hodgman Area Clipping Algorithm
  - Weiler Atherton Area Clipping

Windowing I

- In object space the unit of measurement is any physical unit of length
- In image space the measurement is done using world co-ordinate system
- The area of an image, that we are interested in displaying is defined by the window
- The area on the screen space where we want to display the window is termed as the viewport

Windowing II

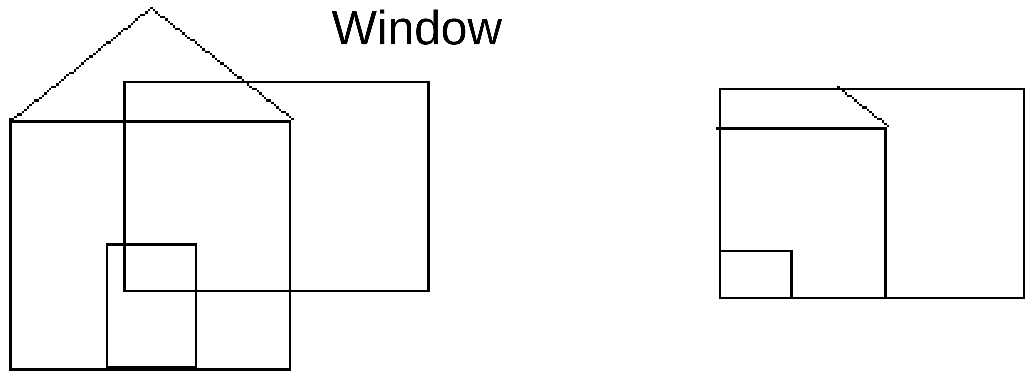




Object Space

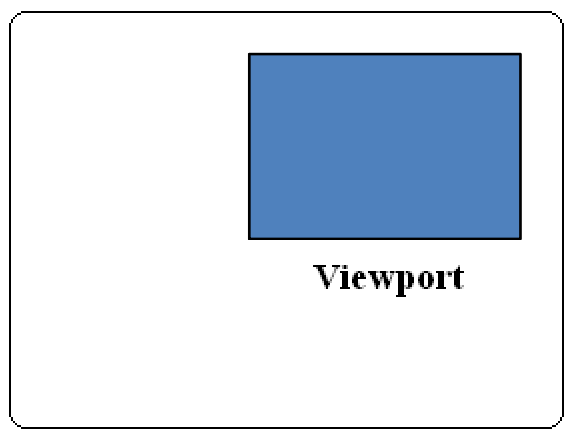
Image Space

Windowing III

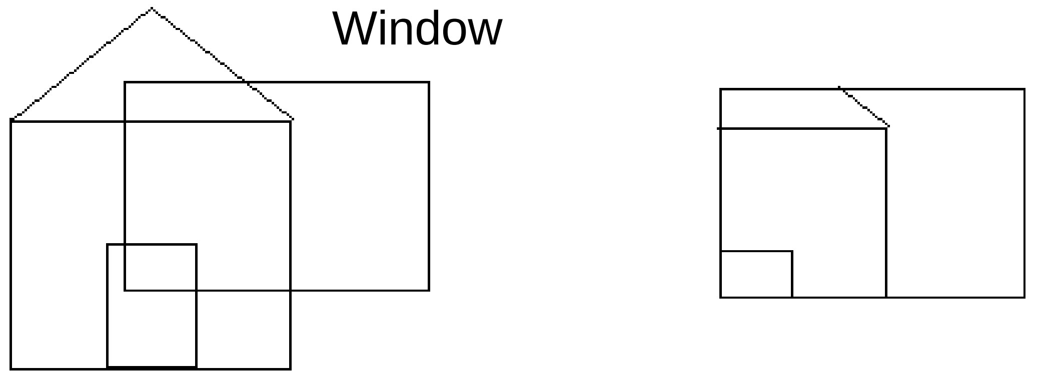


Object Space

Image Space

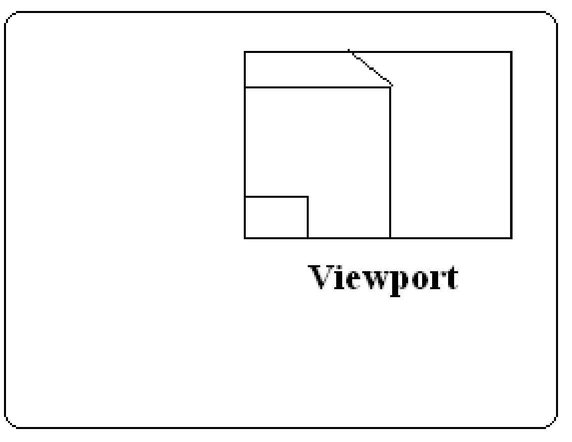


Windowing III

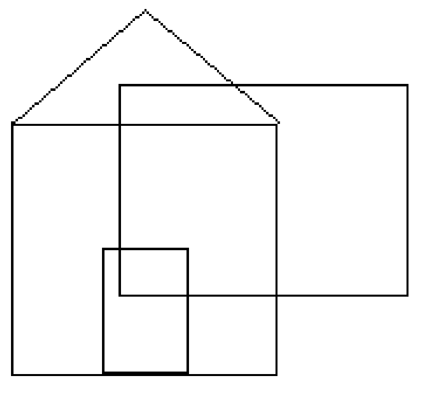


Object Space

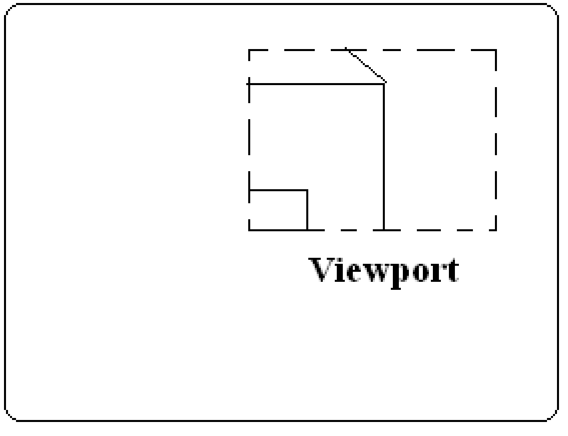
Image Space (Window mapped to viewport)

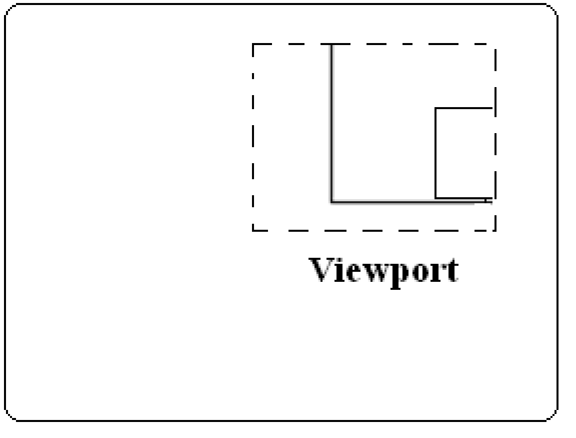


Windowing IV

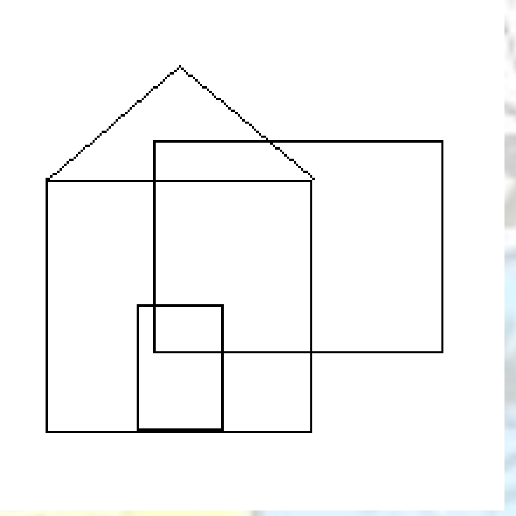


Different windows same viewport

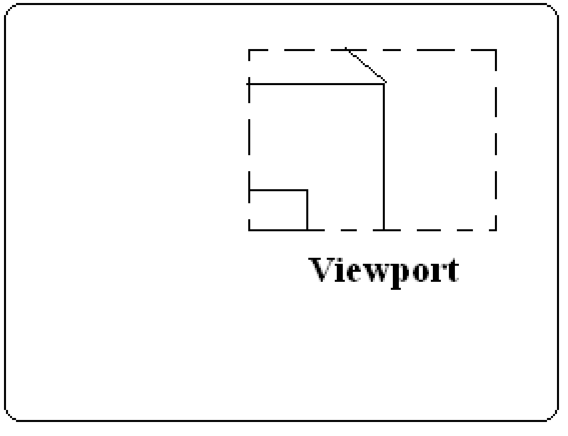


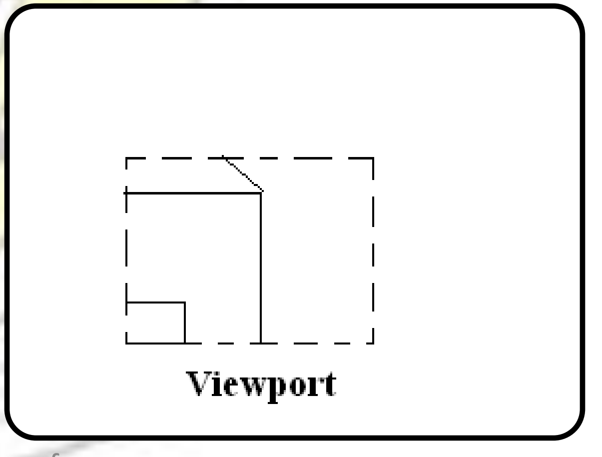


Windowing V



Same window different viewports





Computer Graphics

(SE Computer Engineering 2019-course)

UNIT II WINDOWING AND CLIPPING

"Life is like riding a bicycle. To keep your balance, you must keep moving."

Viewing Transformation

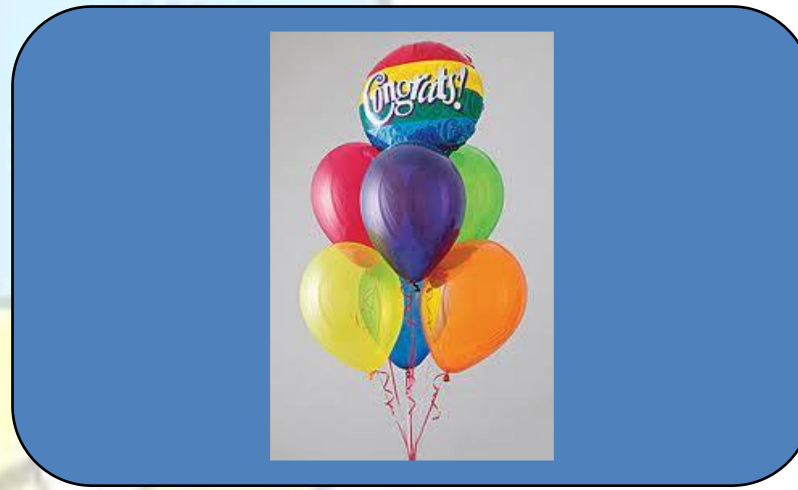
Window to Viewport to Physical Device

The 2D viewing transformation performs the mapping from the window (WCS) to the viewport (NDC) to the physical output device (PDCS). Usually all objects are clipped to the window before the viewing transformation is performed. The viewing transformation which maps WCS to PDCS can be achieved by the following transformations

- Normalization Transformation
- Workstation Transformation

Normalization Transformation(N)

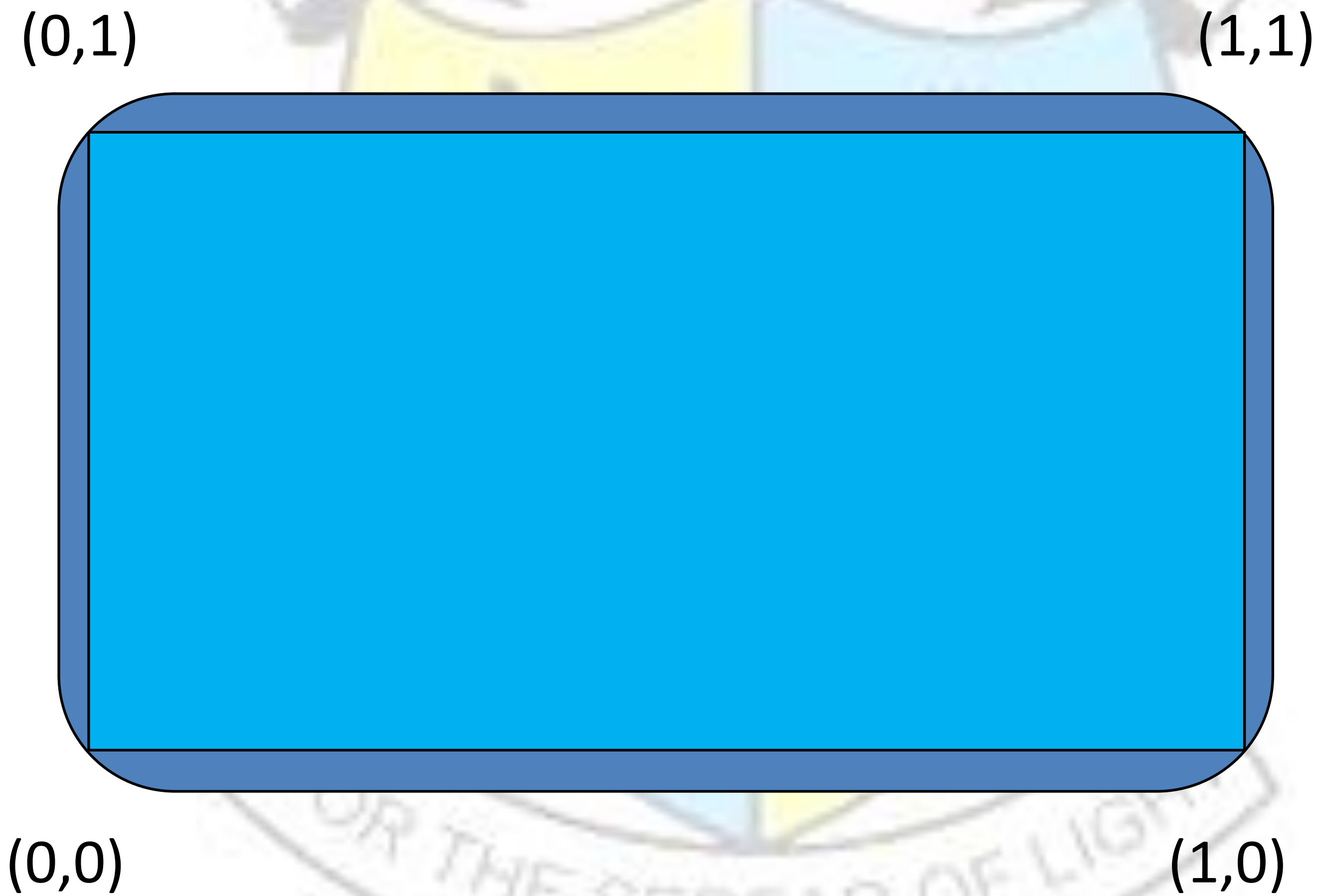
More resolution



Less resolution

Picture definition in pixels

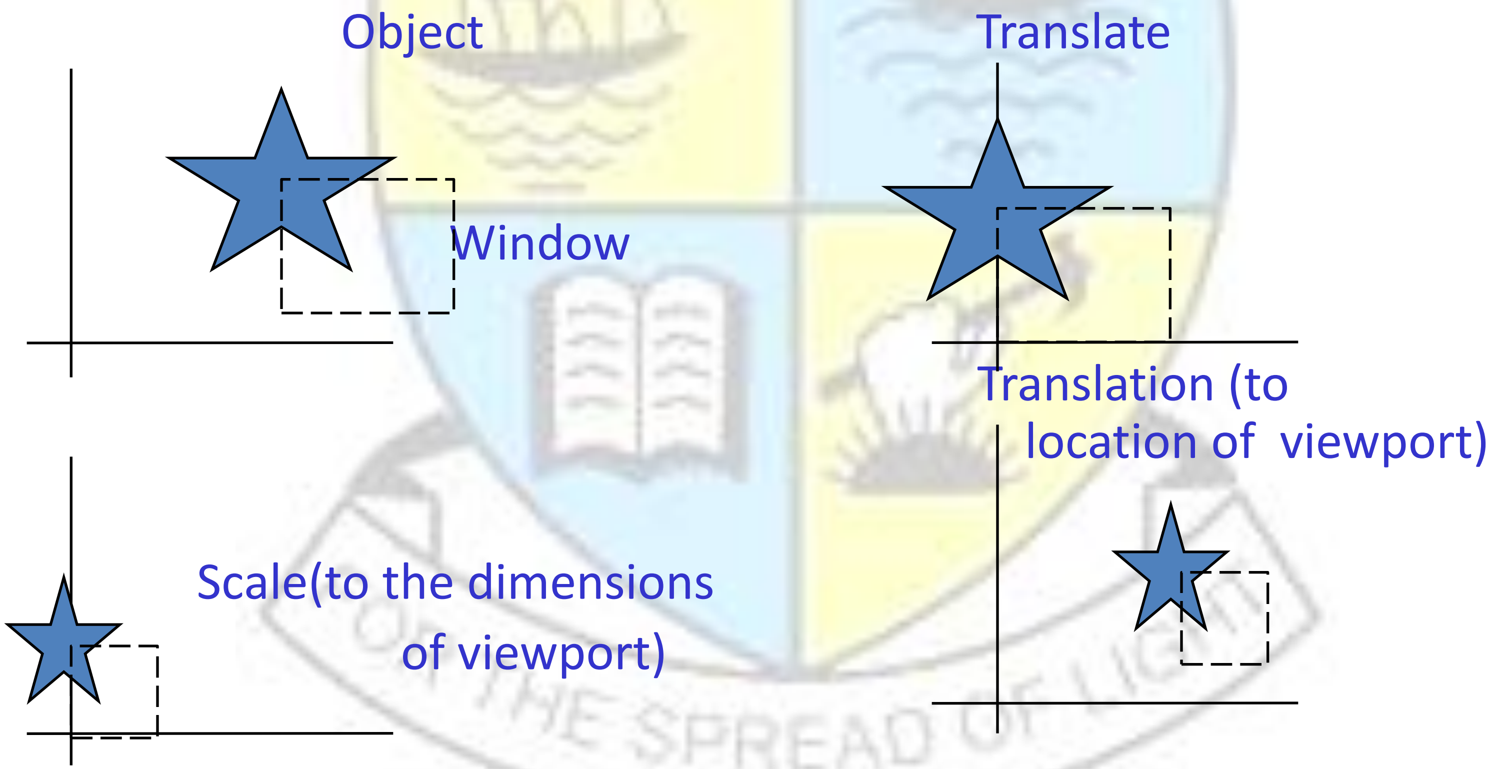
Normalization Transformation(N)



Picture definition in normalized device co-ordinates

Workstation transformation(W)

- The transformation that maps the normalized device coordinates to physical co-ordinates.
- Window to viewport co-ordinate transformation is known as workstation transformation

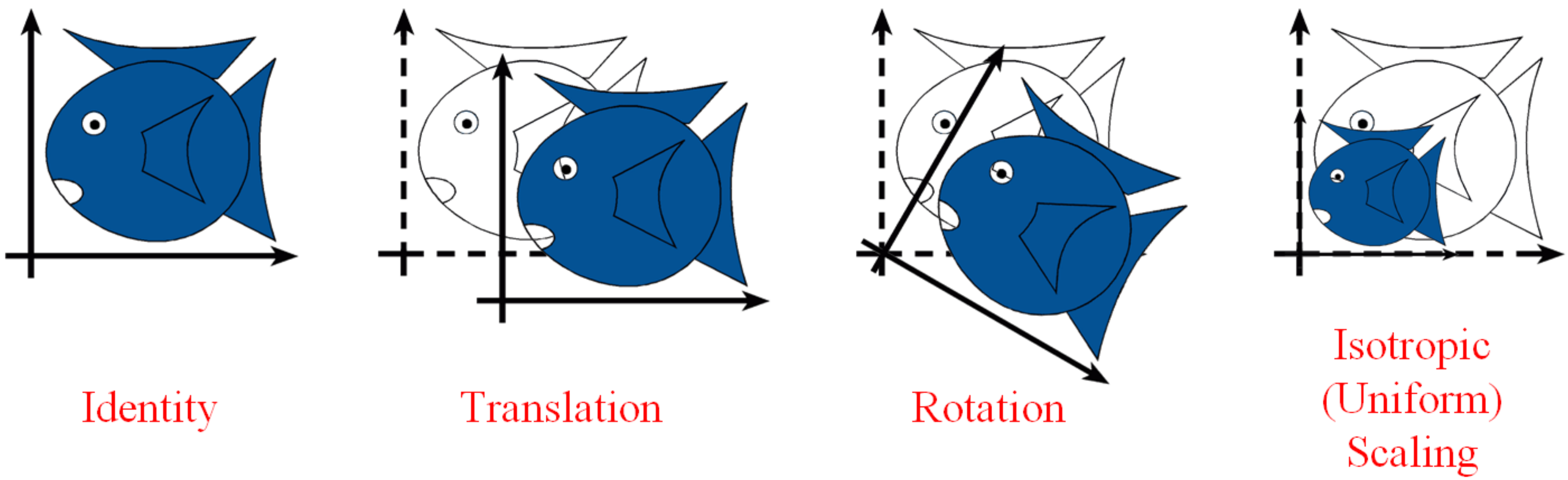


Transformation

Maps points (x, y) in one coordinate system to points (x', y') in another coordinate system

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

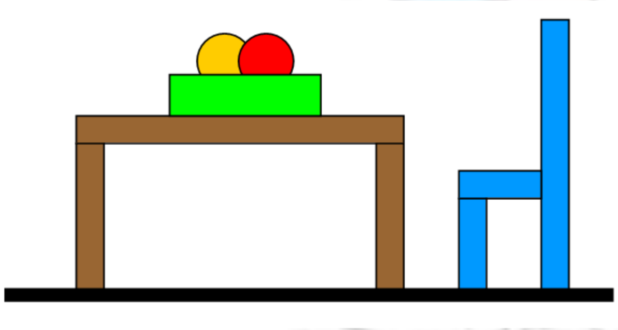
Simple Transformations

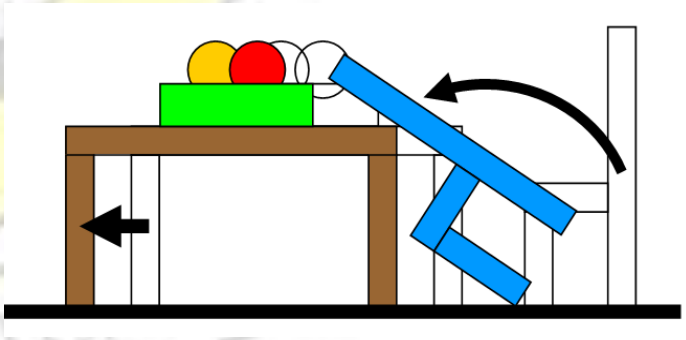


- Can be combined
- Are these operations reversible?

Transformations are used to

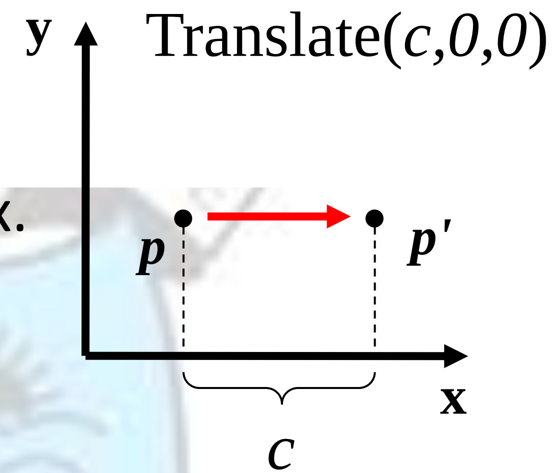
- Position objects in a scene (modelling)
- Change the shape of objects
- Create multiple copies of objects
- Projection for virtual cameras
- Animations





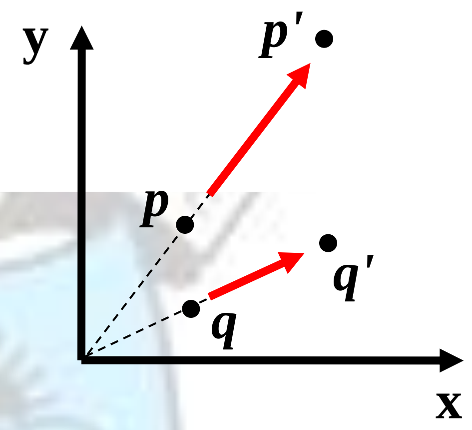
How are Transforms Represented

- translations can be encoded using a matrix.



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

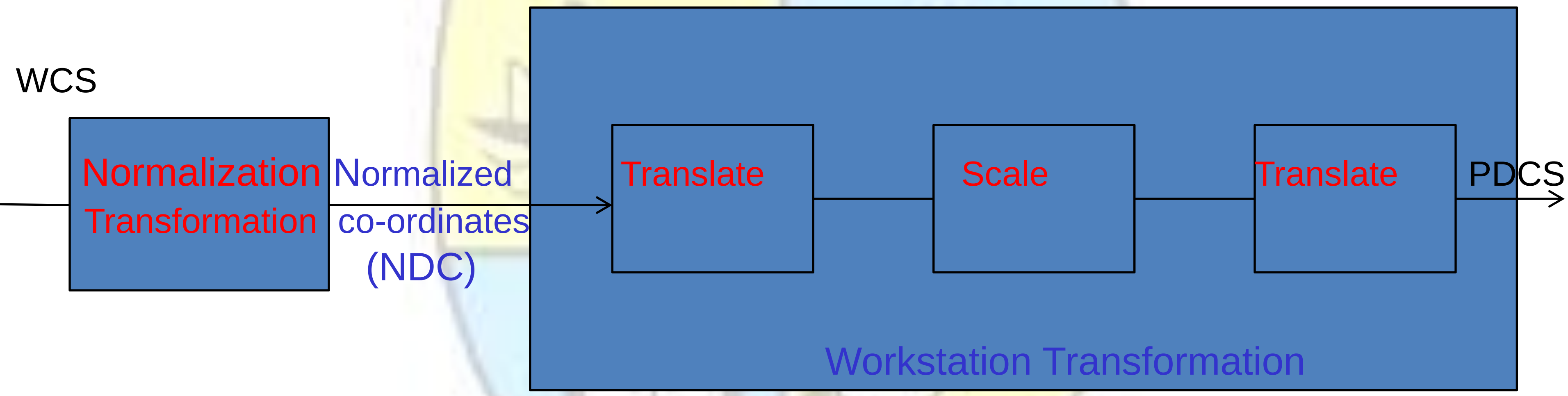
Scale (*sx, sy, sz*)



- Isotropic (uniform) scaling: $s_x = s_y = s_z$

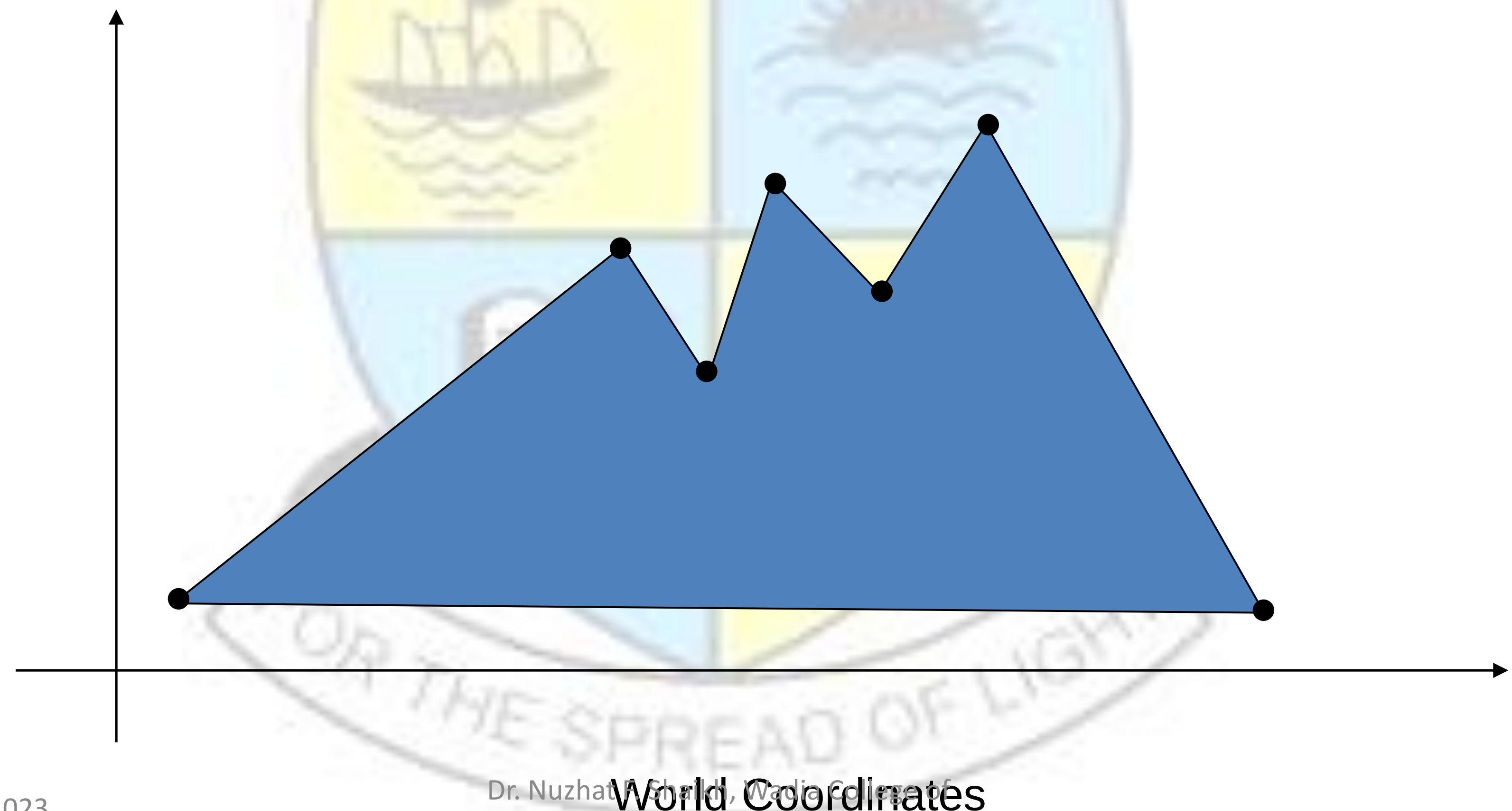
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Viewing Transformation



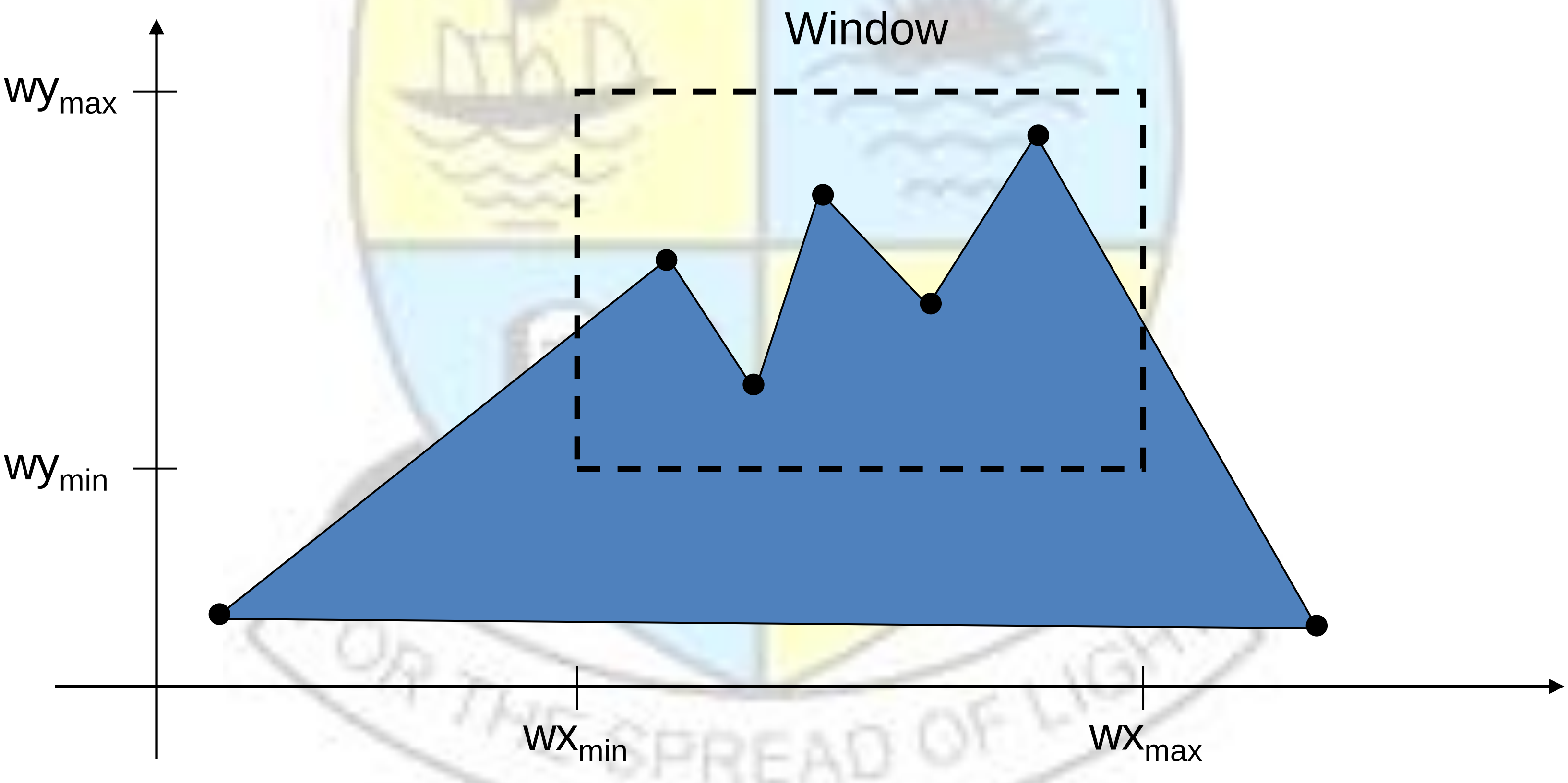
Windowing

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



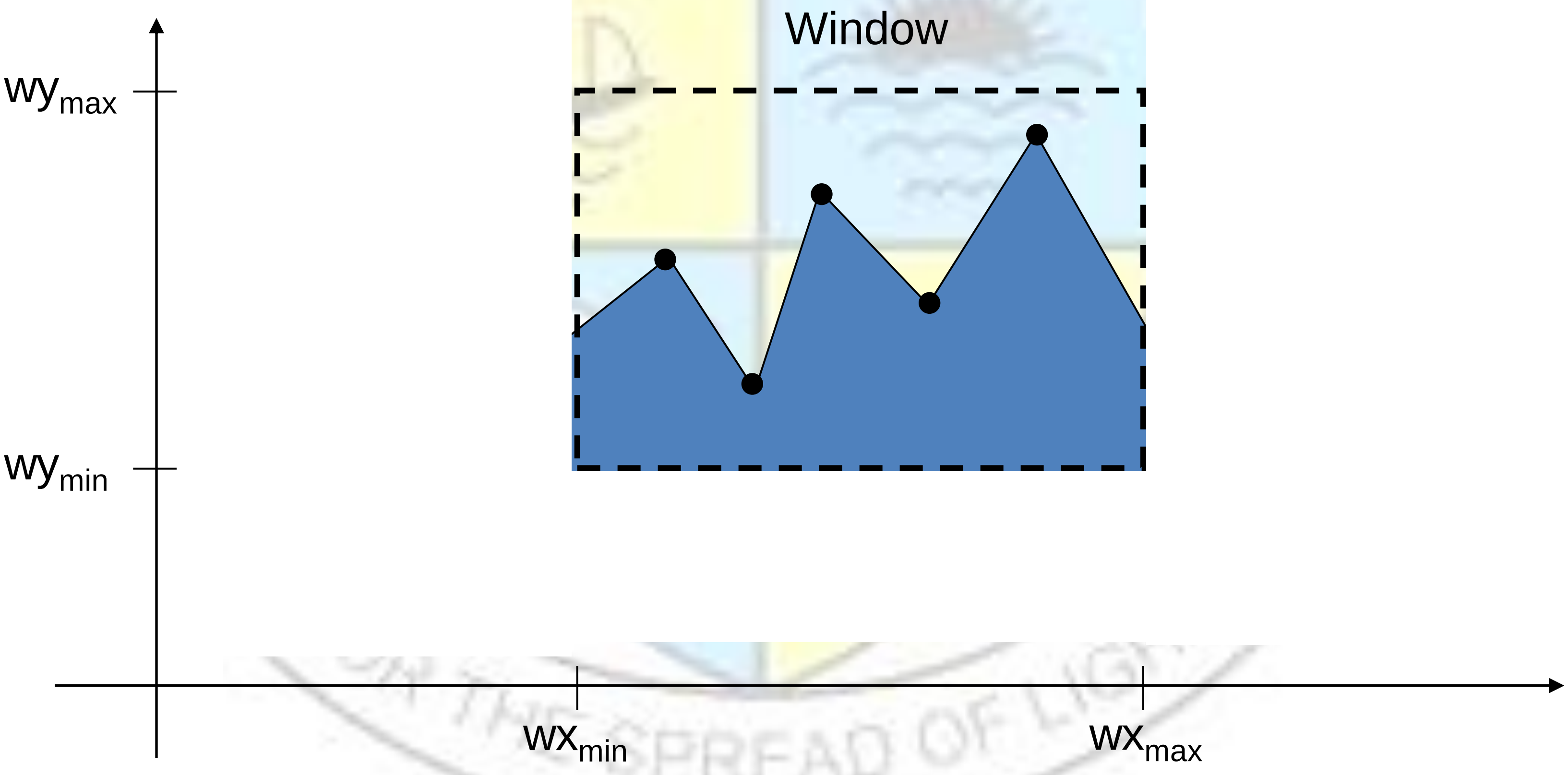
Windowing

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



Clipping

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

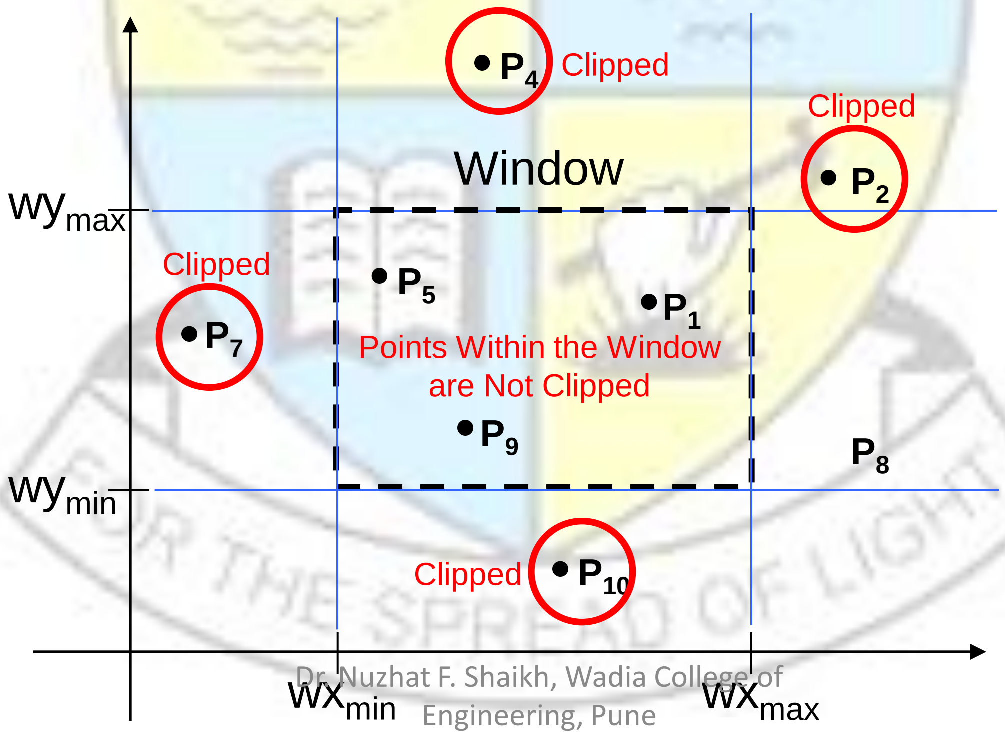


Point Clipping

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

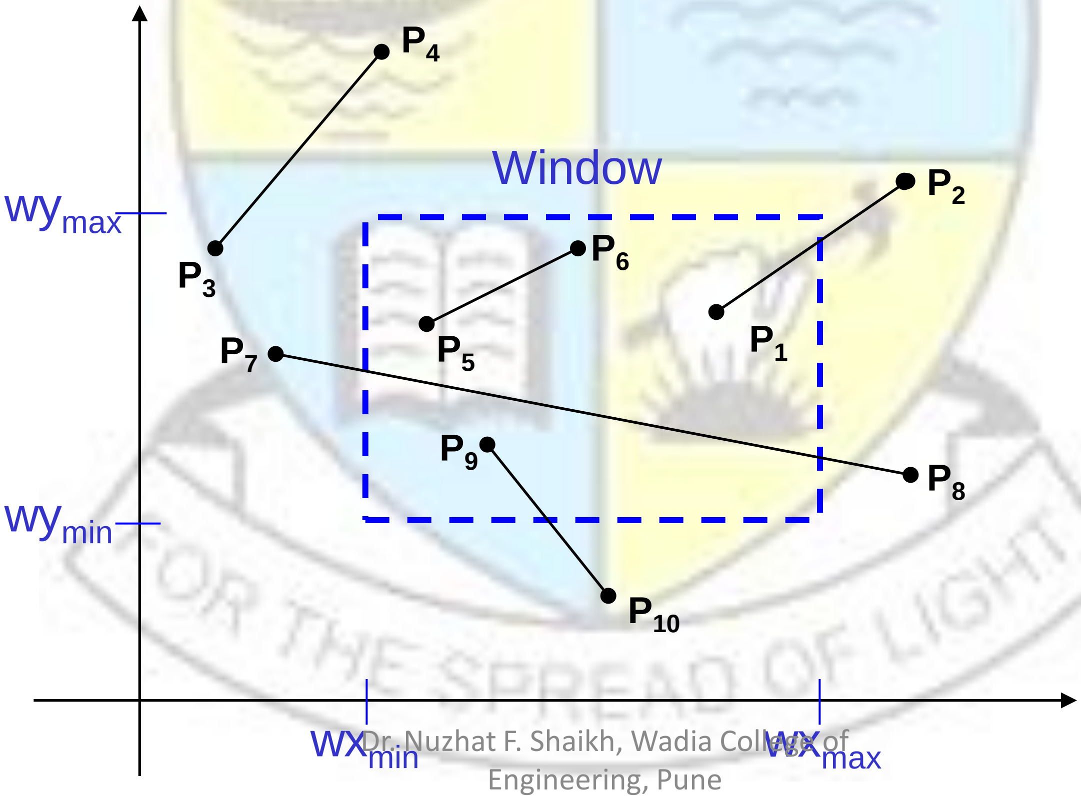
$$wx_{min} \leq x \leq wx_{max} \text{ AND } wy_{min} \leq y \leq wy_{max}$$

otherwise it is clipped



Line Clipping

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



Computer Graphics

(SE Computer Engineering 2019-course)

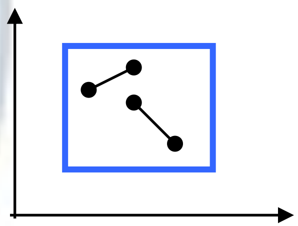
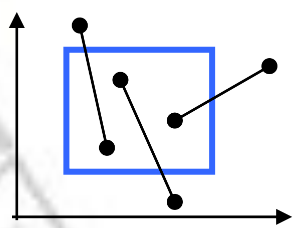
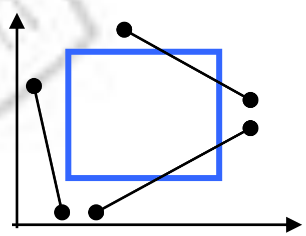
UNIT II WINDOWING AND CLIPPING

"Opportunities don't happen, you create them."

Dr. Nuzhat F. Shaikh, Wadia College of Engineering, Pune

Line Clipping

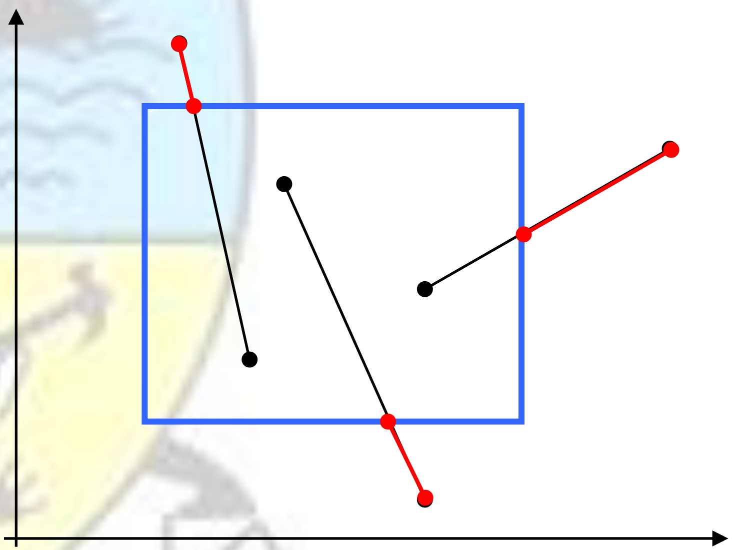
Harder - examine the end-points of each line to see if they are in the window or not

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!	

Brute Force Line Clipping

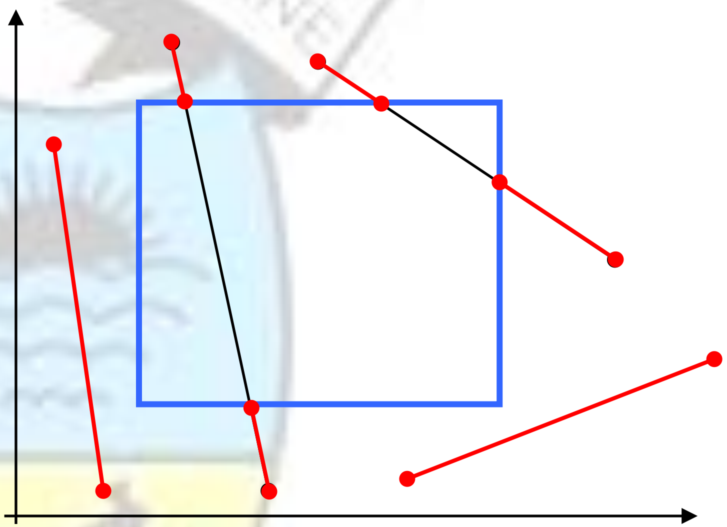
Brute force line clipping can be performed as follows:

- Don’t clip lines with both end-points within the window
- 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



Brute Force Line Clipping (cont…)

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



However, calculating line intersections is computationally expensive

Because a scene can contain so many lines, the brute force approach to clipping is much slow

Cohen-Sutherland Clipping Algorithm

An efficient line clipping algorithm

The key advantage of the algorithm is that it 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-Sutherland: World Division

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

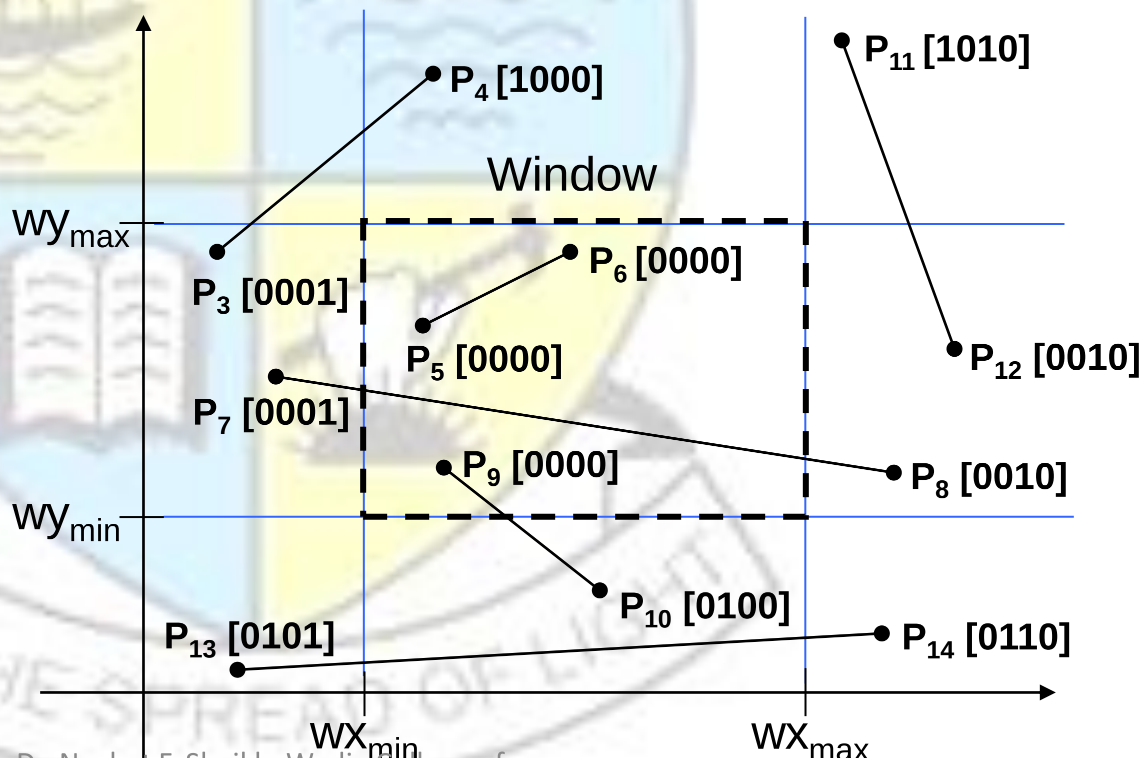
1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

Cohen-Sutherland: Labelling

Every end-point is labelled with the appropriate region code

3	2	1	0
above	below	right	left

Region Code Legend

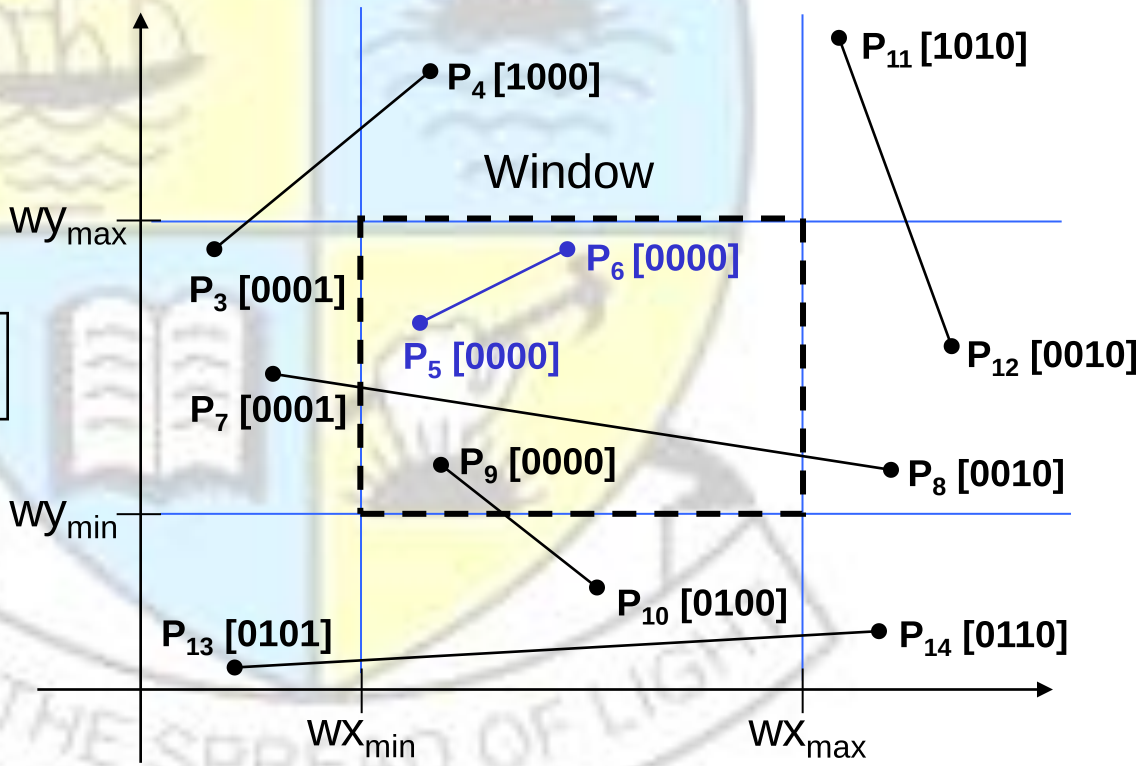


Cohen-Sutherland: Lines In The Window

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

3	2	1	0
above	below	right	left

Region Code Legend

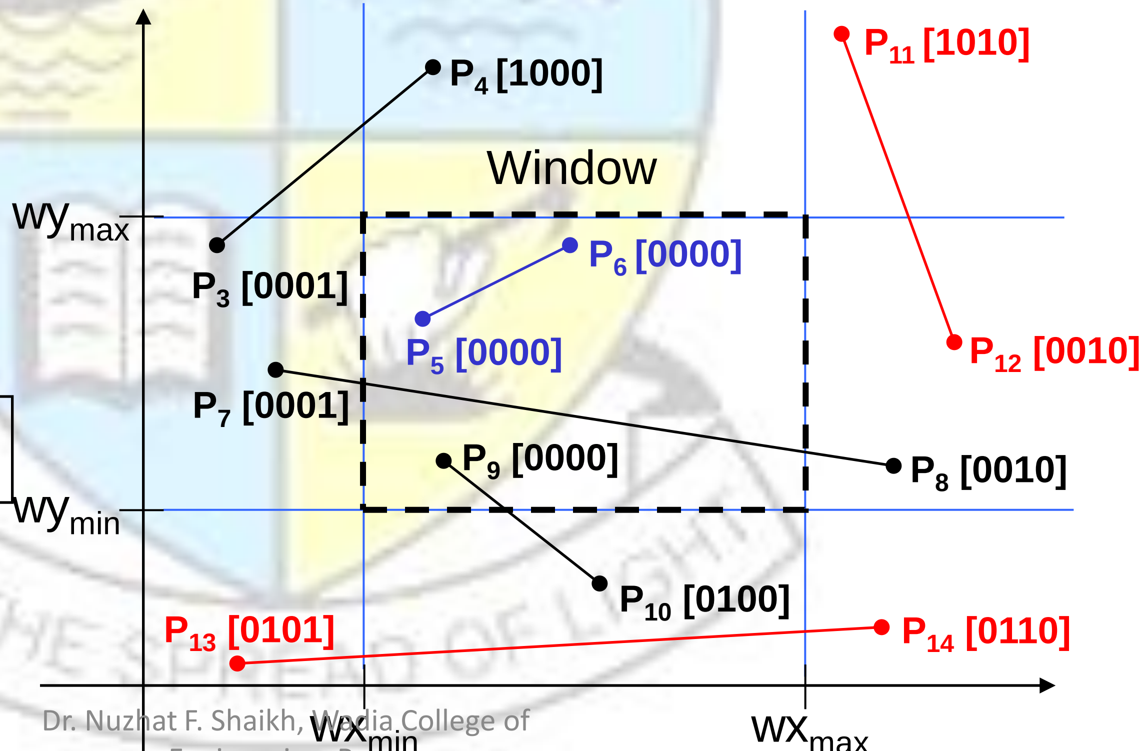


Cohen-Sutherland: Lines Outside The Window

- Any lines with a common set bit in the region codes of both end-points can be clipped
- The AND operation can efficiently check this
- If Logical ANDing is zero – Partially visible
- If Logical ANDing is non-zero – completely invisible

3	2	1	0
above	below	right	left

Region Code Legend



Cohen-Sutherland: Other Lines

Lines that cannot be identified as completely inside or completely outside the window may or may not cross the window interior

These lines are 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: Other Lines (cont…)

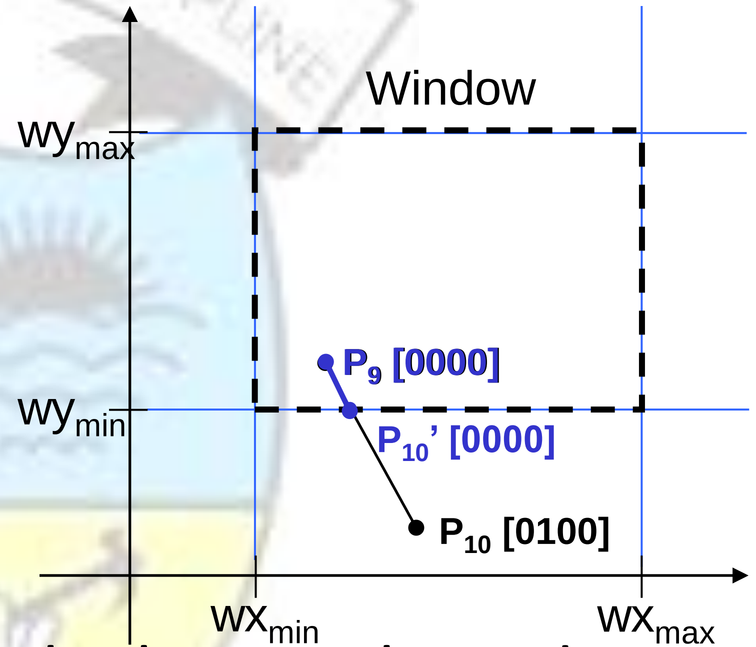
We can use the region codes to determine which window boundaries should be considered for intersection

- To check if a line crosses a particular boundary we compare the appropriate bits in the region codes of its end-points
- If one of these is a 1 and the other is a 0 then the line crosses the respective boundary

Cohen-Sutherland Examples

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



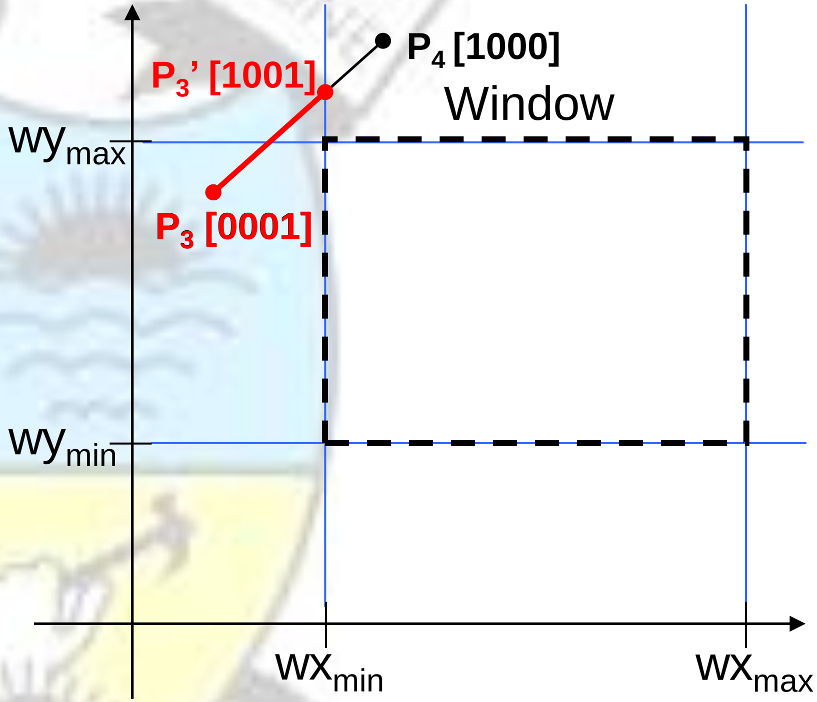
AND Z P.V.
NZ C.I.

Cohen-Sutherland Examples

Consider the line P_3 to P_4 below

– Start at P_3

– 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_3'



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

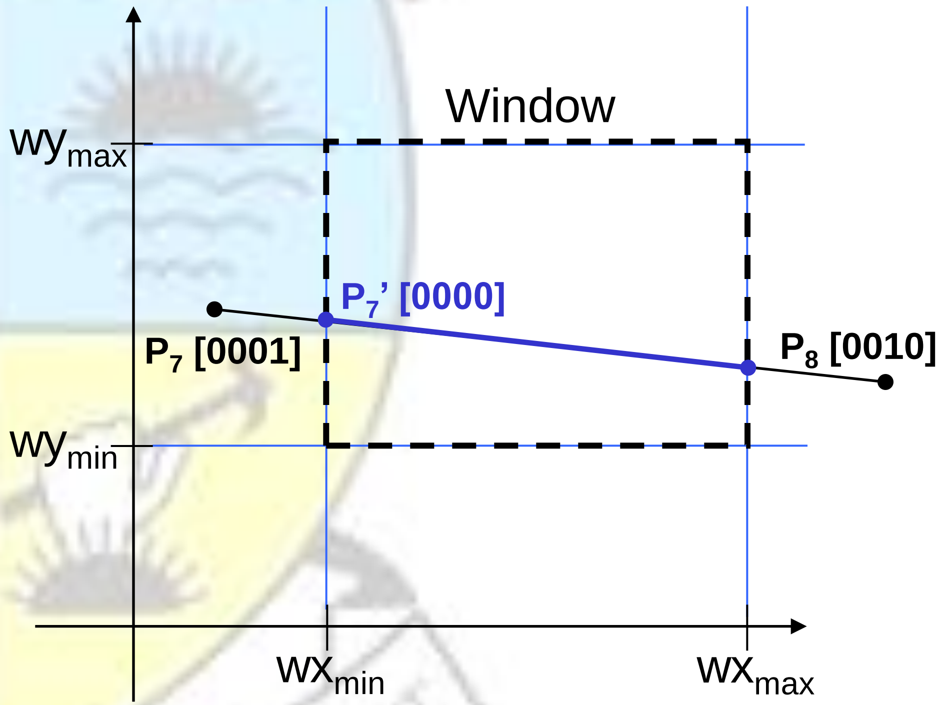
3	2	1	0
above	below	right	left

AND Z P.V.
NZ C.I.

Cohen-Sutherland Examples

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'

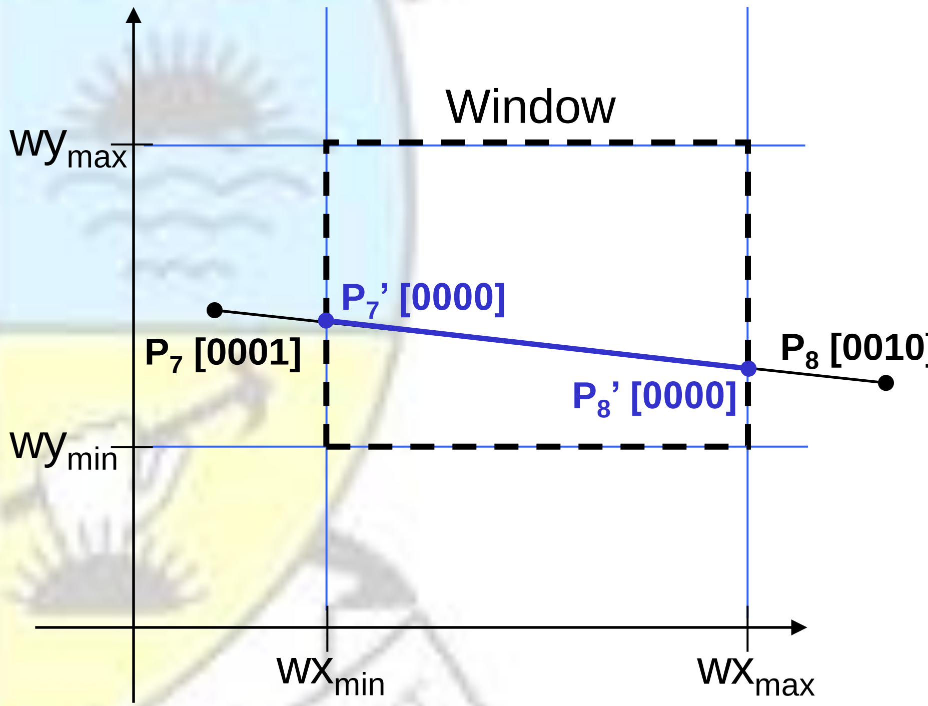


AND Z P.V.
NZ C.I.

Cohen-Sutherland Examples

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



Calculating Line Intersections

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}

- m is the slope of the line in question and can be calculated as $m = (y_2 - y_1) / (x_2 - x_1)$

Calculating Line Intersections

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

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

where $y_{boundary}$ can be set to either wy_{min} or wy_{max}
- m is the slope of the line in question and can be calculated as $m = (y_2 - y_1) / (x_2 - x_1)$

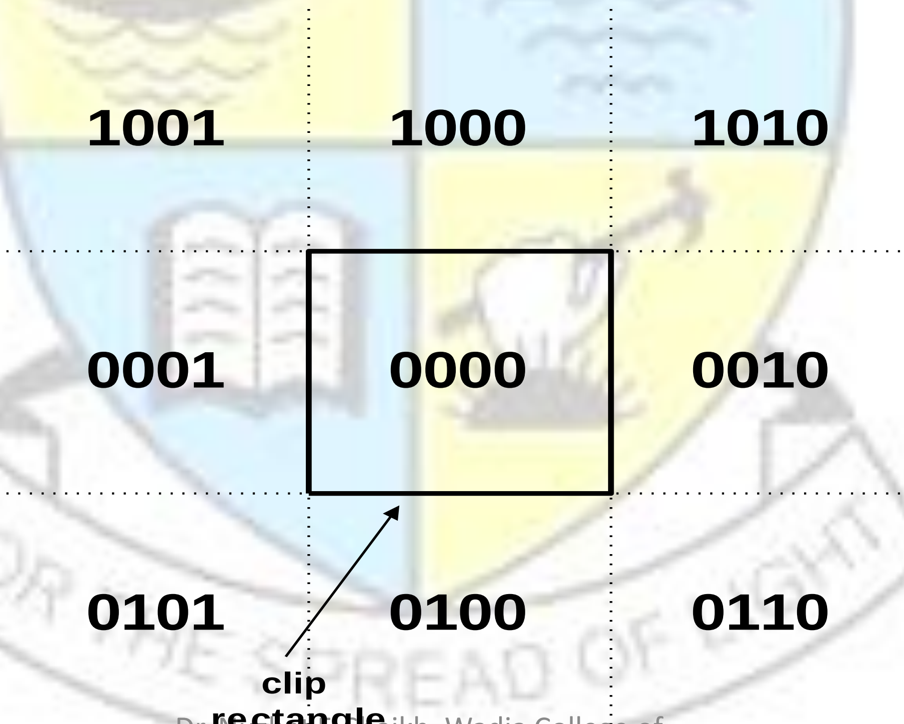
Cohen Sutherland Algorithm

- The algorithm employs an efficient procedure for finding the category of a line. It proceeds in Two Steps.

1) Assign a 4-bit Code to each endpoint of the line. The code is determined according to which of the following nine regions of the plane the endpoint lies in.

bit 0 : $y > y_{max}$

bit 1 : $y < y_{min}$



bit 2 : $x > x_{max}$

bit 3 : $x < x_{min}$

2) i)The Line is Visible if both the region codes are 0000

ii)Not Visible if the bitwise logical AND of the codes is Not 0000.

iii)Candidate for Clipping if the bitwise Logical AND of the region Codes is 0000.

- For the line in Category (iii) we proceed to find the intersection point of the line with one of the boundaries of the clipping window.
- If bit 1and 2 is 1 then intersect with Line y=ymax and y= ymin.
- If bit 3and 4 is 1 then intersect with Line x=xmax and x= xmin.
- The co-ordinates of the intersection points are:
  - xi= xmin or xmax (if the boundary line is vertical:
  - Yi=y1+m(xi-x1);

  Or

  xi= x1+(yi-y1)/m (if the boundary line is horizontal)

  Yi=ymin or ymax;

  Where m is slope = (y2-y1)/(x2-x1)

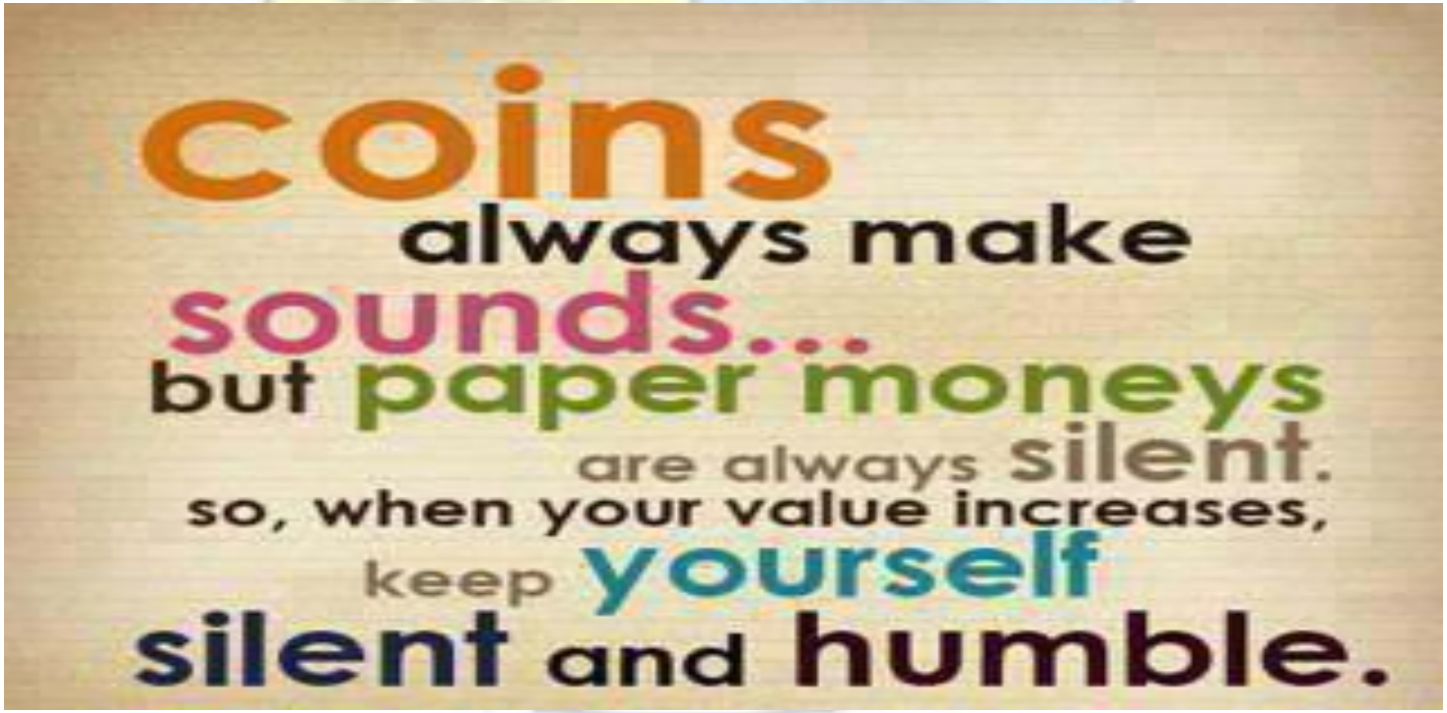
Content beyond the syllabus

- Cyrus Beck generalized line clipping
- Liang Barsky line clipping

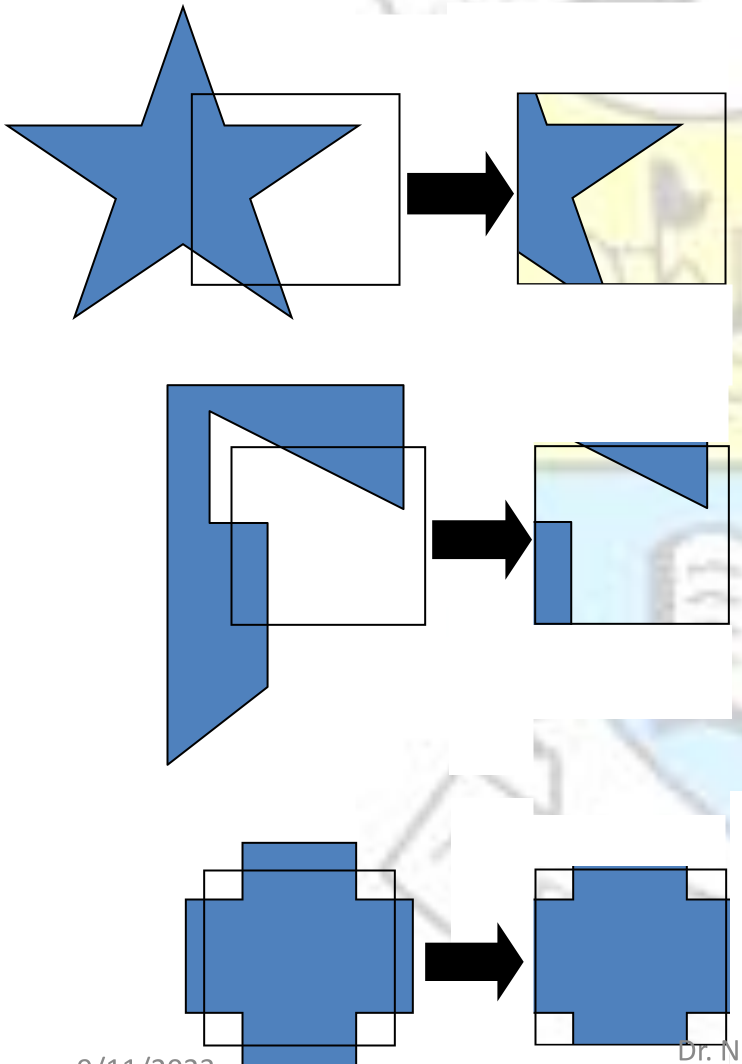
Computer Graphics

(SE Computer Engineering 2019-course)

UNIT II WINDOWING AND CLIPPING



Area Clipping



Similar to lines, areas must be clipped to a window boundary

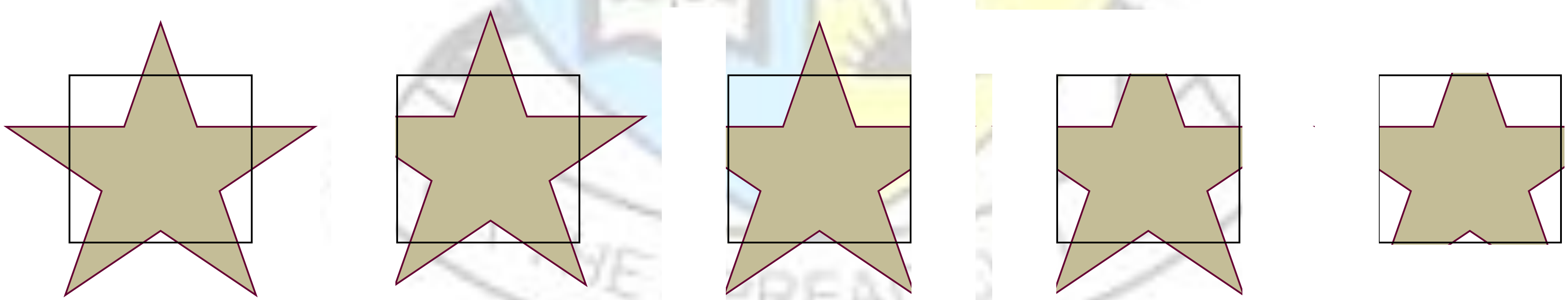
Care must be taken as to which portions of the area must be clipped

Sutherland-Hodgman Area Clipping Algorithm

A technique for clipping areas was developed by Sutherland & Hodgman

Put simply, the polygon is clipped by comparing it against each boundary in turn

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



Original Area | Clip Left | Clip Right | Clip Top | Clip Bottom

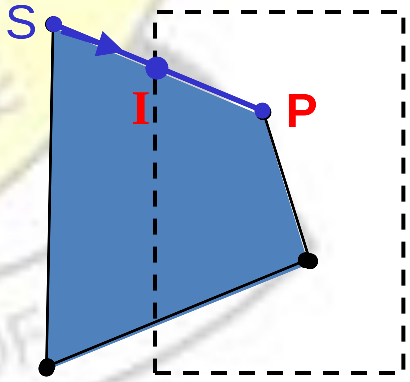
Sutherland-Hodgman Area Clipping Algorithm

To clip an area against an individual boundary:

- Consider each edge in turn against the boundary
- If the first vertex is outside the window and the second inside, then the intersection and the second vertex are added to the output vertex list

OUTPUT VERTEX LIST

{I,P}



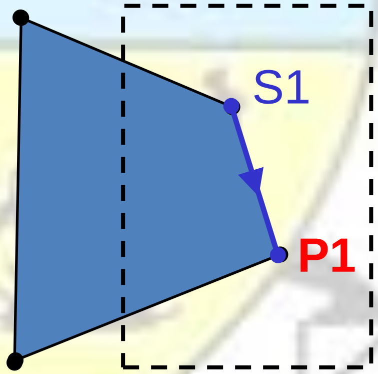
Save Points I & P

Sutherland-Hodgman Area Clipping Algorithm

- If both the vertices of the edge are inside the window only the second is added to the output vertex list

OUTPUT VERTEX LIST

{I,P,P1}



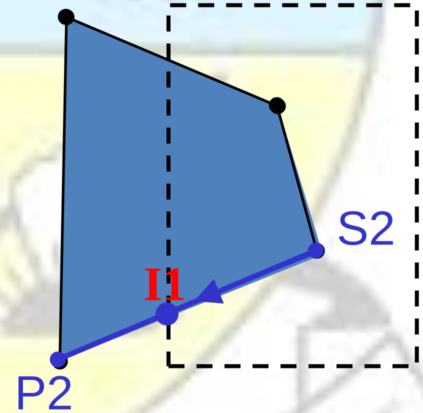
Save Point P1

Sutherland-Hodgman Area Clipping Algorithm

- If the first vertex is inside and the second outside the window boundary, only the edge intersection with the window boundary is added to the output vertex list.

OUTPUT VERTEX LIST

{I,P,P1,I1}



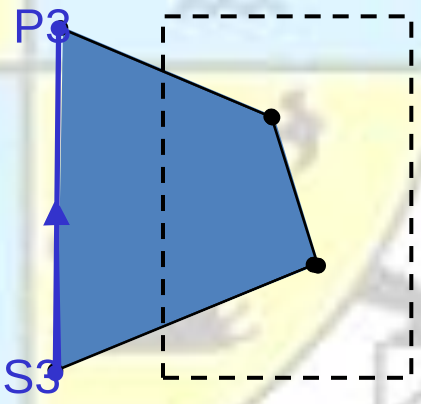
Save Point I1

Sutherland-Hodgman Area Clipping Algorithm

- If both the vertices are outside the window boundary, nothing is added to the output vertex list

OUTPUT VERTEX LIST

{I,P,P1,I1}

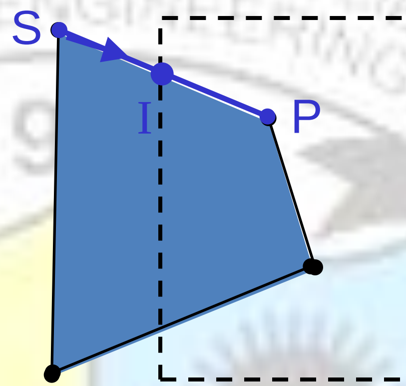


No Points Saved

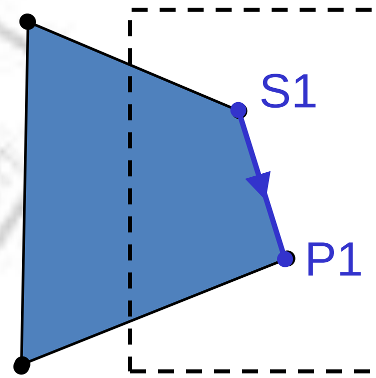
Sutherland-Hodgman Area Clipping Algorithm

Each example shows the point being processed (P) and the previous point (S)

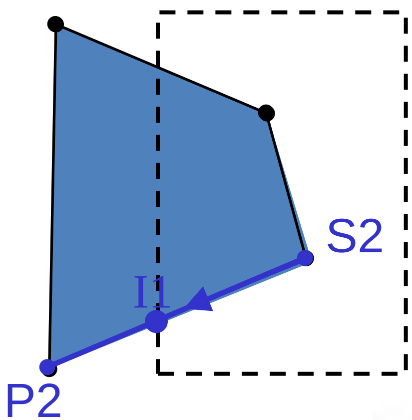
Saved points define area clipped to the boundary in question



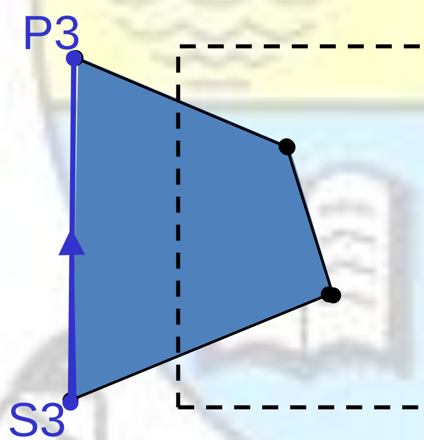
Save Points I & P



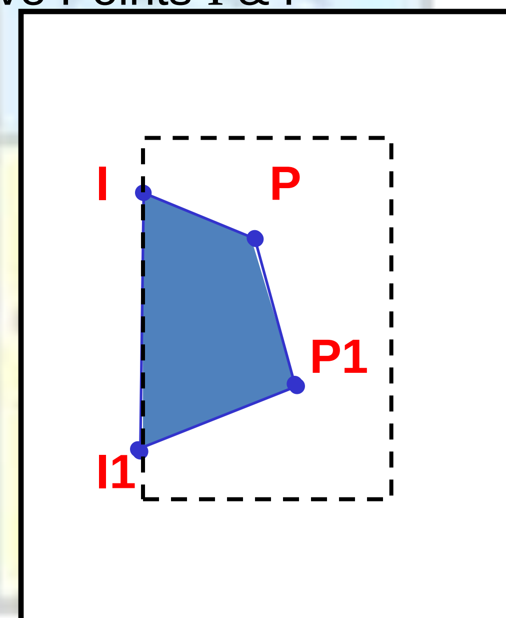
Save Point P1



Save Point I1

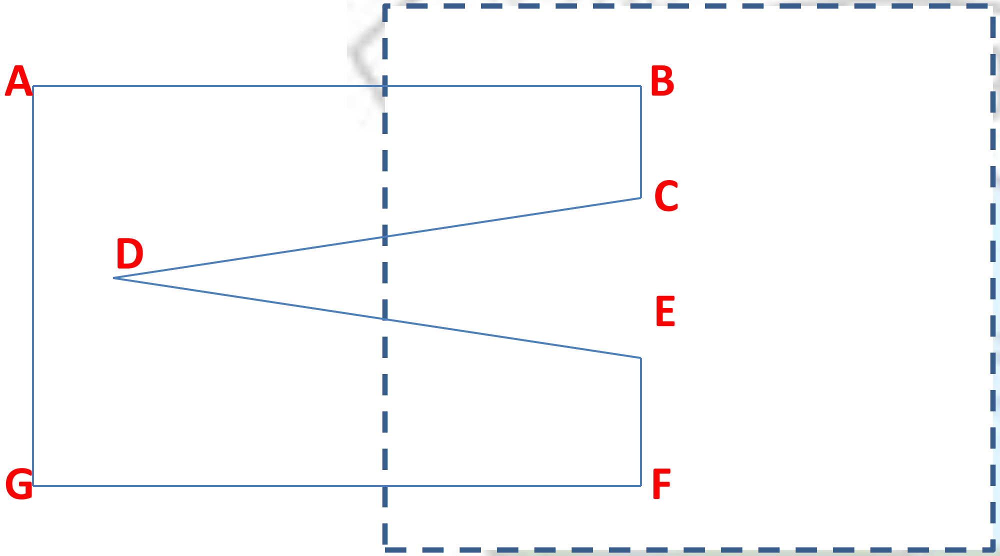


No Points Saved

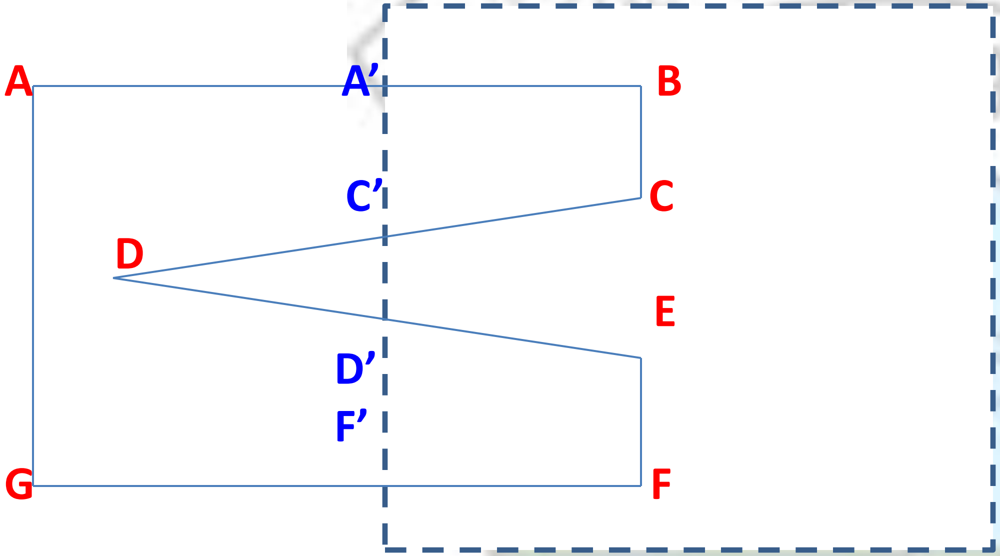


OUTPUT VERTEX LIST
{I,P,P1,I1}

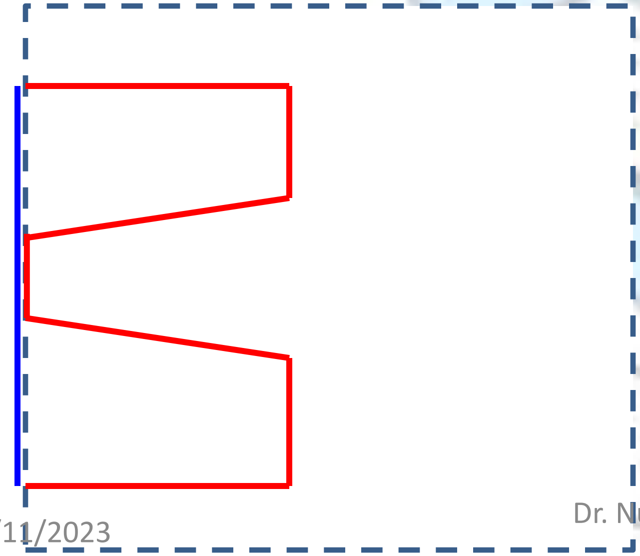
Disadvantage of Sutherland-Hodgman Area Clipping Algorithm



Disadvantage of Sutherland-Hodgman Area Clipping Algorithm



A'	B	C	C'	D'	E	F	F'



Sutherland Hodgeman
-Convex (Best suited)
-Concave (Extraneous lines)

Weiler Atherton Algorithm

Before being applied to a polygon, the algorithm requires several preconditions to be fulfilled:

- Candidate polygons need to be oriented clockwise.
- Candidate polygons should not be self-intersecting (i.e. re-entrant).

Weiler Atherton Algorithm

Given polygon A as the clipping region and polygon B as the subject polygon to be clipped, the algorithm consists of the following steps:

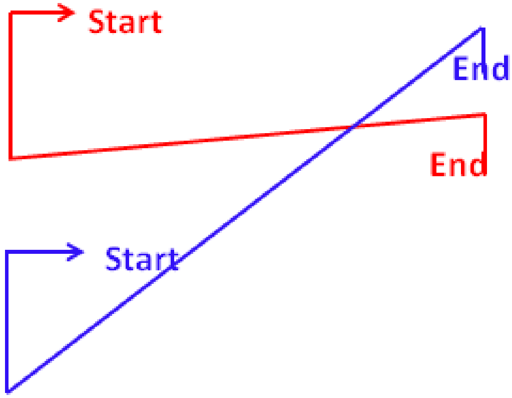
1. List the vertices of the clipping-region polygon A and those of the subject polygon B.
2. Label the listed vertices of subject polygon B as either inside or outside of clipping region A.
3. Find all the polygon intersections and insert them into both lists, linking the lists at the intersections.
4. Generate a list of "inbound" intersections – the intersections where the vector from the intersection to the subsequent vertex of subject polygon B begins inside the clipping region.
5. Follow each intersection clockwise around the linked lists until the start position is found.

Weiler Atherton Algorithm

If there are no intersections then one of three conditions must be true:

i. A is inside B – return A for clipping, B for merging.
ii. B is inside A – return B for clipping, A for merging.
iii. A and B do not overlap – return None for clipping or A & B for merging.

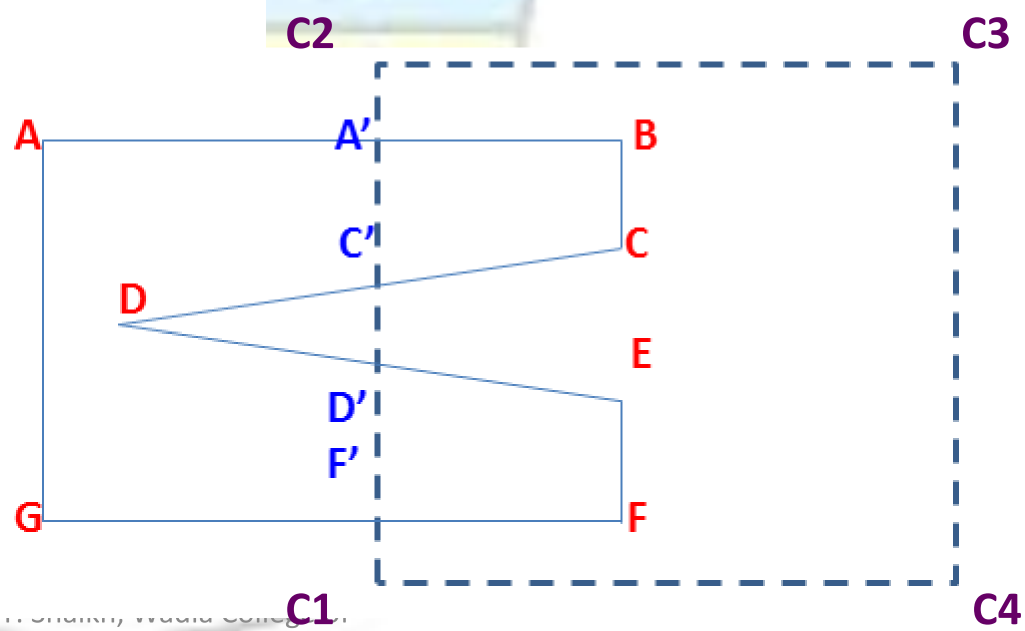
Subject Polygon List
A
A’
B
C
C’
D
D’
E
F
F’
G
A

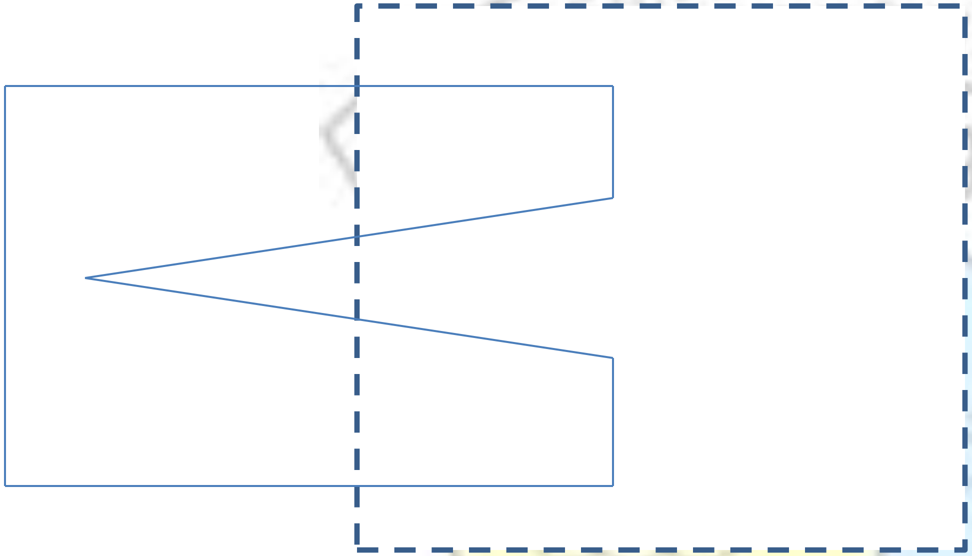


Clip Polygon List
C1
F’
D’
C’
A’
C2
C3
C4

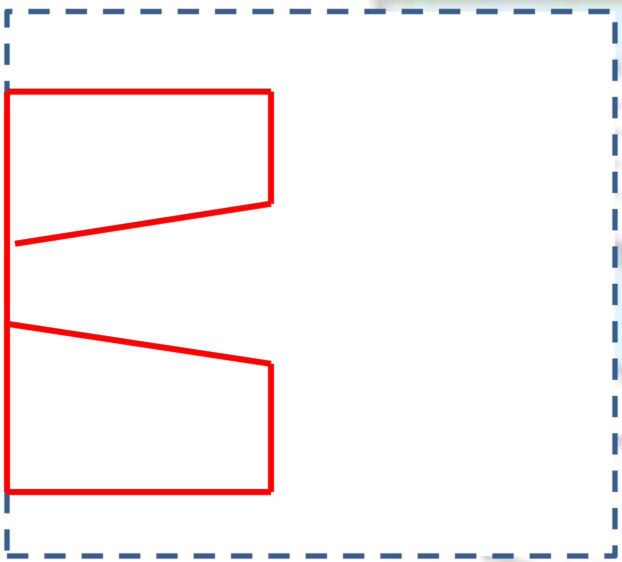
A’	B	C	C’	A’

D’	E	F	F’	D’

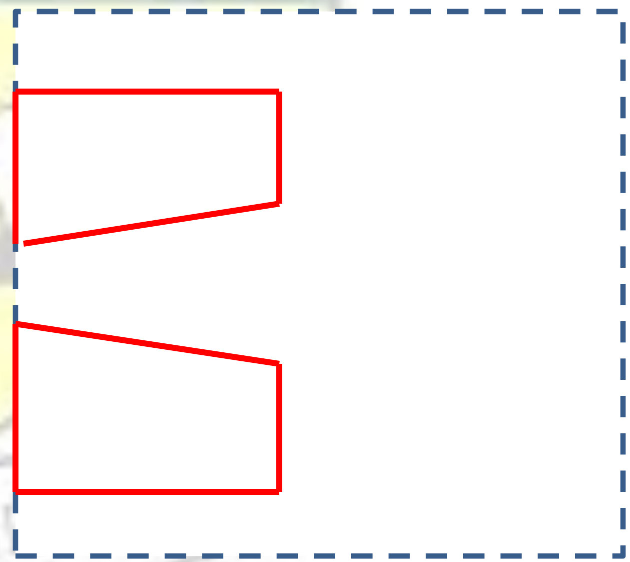




Sutherland Hodgeman



Weiler Atherton



Summary

Objects within a scene must be clipped to display the scene in a window

Because there are can be so many objects clipping must be extremely efficient

The Cohen-Sutherland algorithm can be used for line clipping

The Sutherland-Hodgman and Weiler Atherton algorithm can be used for area clipping

Assignment

1. Explain the methods used for testing whether a point is inside or outside a polygon.
2. What is seed fill algorithm.
3. Explain boundary fill and flood fill.
4. Explain scan line fill algorithm.
5. Define the following concepts: Window, viewport, image space, object space.
6. Explain viewing transformation with the help of a block diagram.
7. What is need to clip a polygon.
8. Explain Sutherland-hodgeman area clipping algorithm.
9. Explain Cohen Sutherland line clipping algorithm.
10. What are the disadvantages of Sutherland Hodgeman algorithm. Explain Weiler Atherton Area clipping algo.