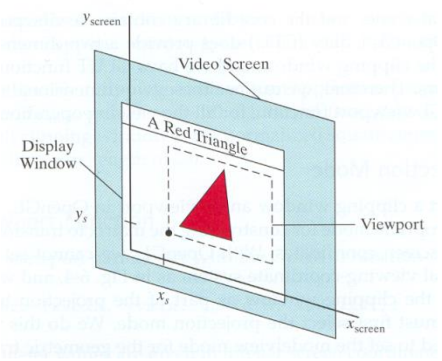
Universidade de Aveiro
Departamento de Electrónica,
Telecomunicações e Informática

2D Visualization



Beatriz Sousa Santos, Joaquim Madeira

Overview

- Recap – 2D transformations
- Recap – 2D visualization pipeline
- The window-to-viewport transformation revisited
- 2D viewing in OpenGL / WebGL

RECAP
- 2D TRANSFORMATIONS

2D Transformations

- Position, orientation and scaling for objects in XOY
- Basic transformations
  - Translation / Displacement
  - Rotation relative to the coordinates' origin
  - Scaling relative to the coordinates' origin
- Representation using **3 x 3 matrices**
  - **Homogeneous coordinates**
- Complex transformations
  - **Decompose** into a sequence of basic transformations

Rotating a rectangle – Questions ?

- Rectangle is defined by vertices

  A=(0,0), B=(2,0), C=(2,4) and D=(0,4)
- Rotate the rectangle around its center
- Rotation angle is –45 degrees

- Decompose into basic transformations
- Multiply them to get the global transformation matrix
- Compute the coordinates of the transformed vertices

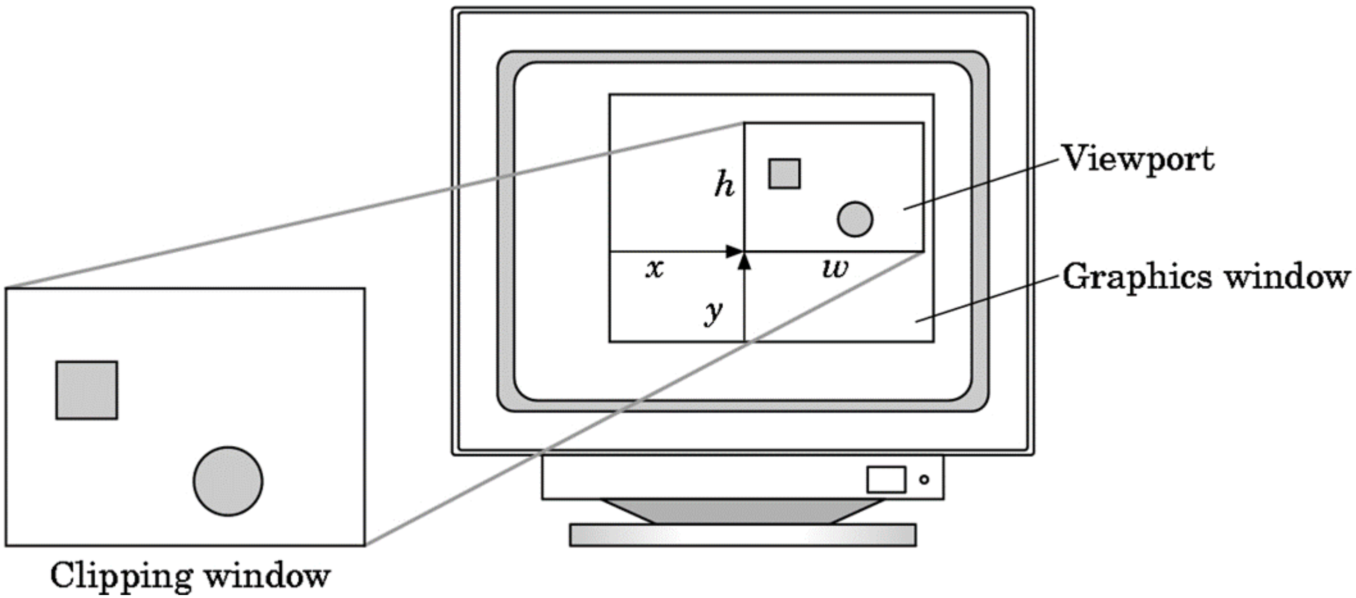
Additional problems (see PDF) – Questions ?

1- Given the square, defined by the vertices (2, 2), (3, 2), (3, 3) and (2, 3), it is to be rotated around its center by an angle of 90 degrees.

2- Given the triangle, defined by the vertices (2, 0), (4, 2) and (-1, 5), determine the triangle resulting from applying a symmetry transformation relative to the $y = x$ straight-line.

RECAP
- 2D VIEWING

2D Viewing



[Angel]

Recap – The visualization pipeline

- Model the **scene**
  - Model instatiation: position, orientation and size
  - World coordinates

- Select a **view**
  - Area in the XOY plane containing the scene or part of
  - Clipping window

- That view is shown at a given **location** in the display device
  - Viewport

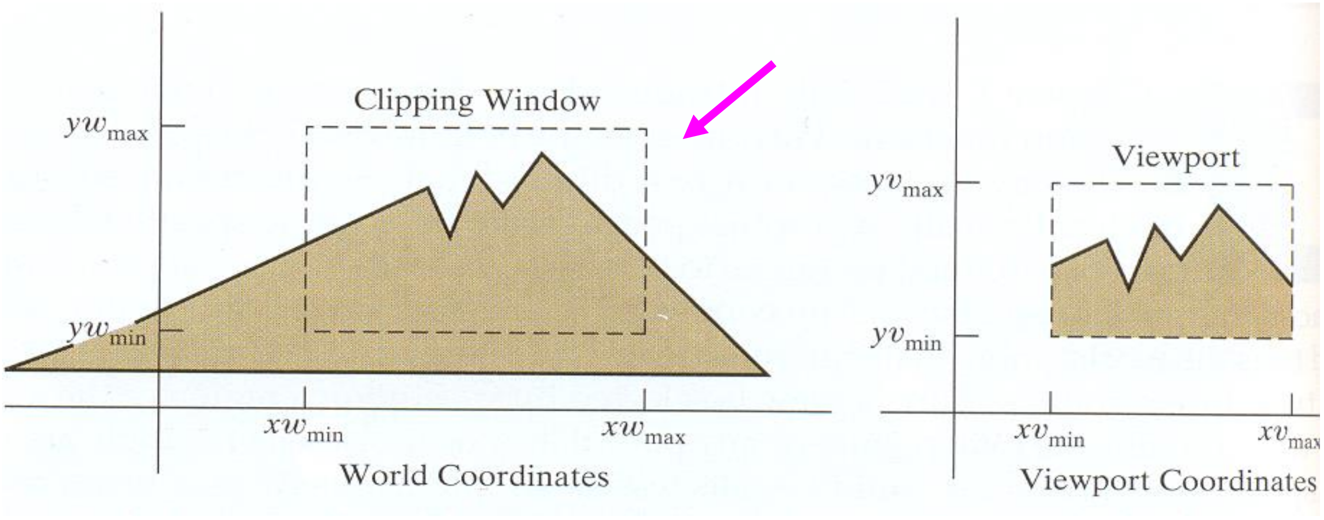
Recap – The visualization pipeline

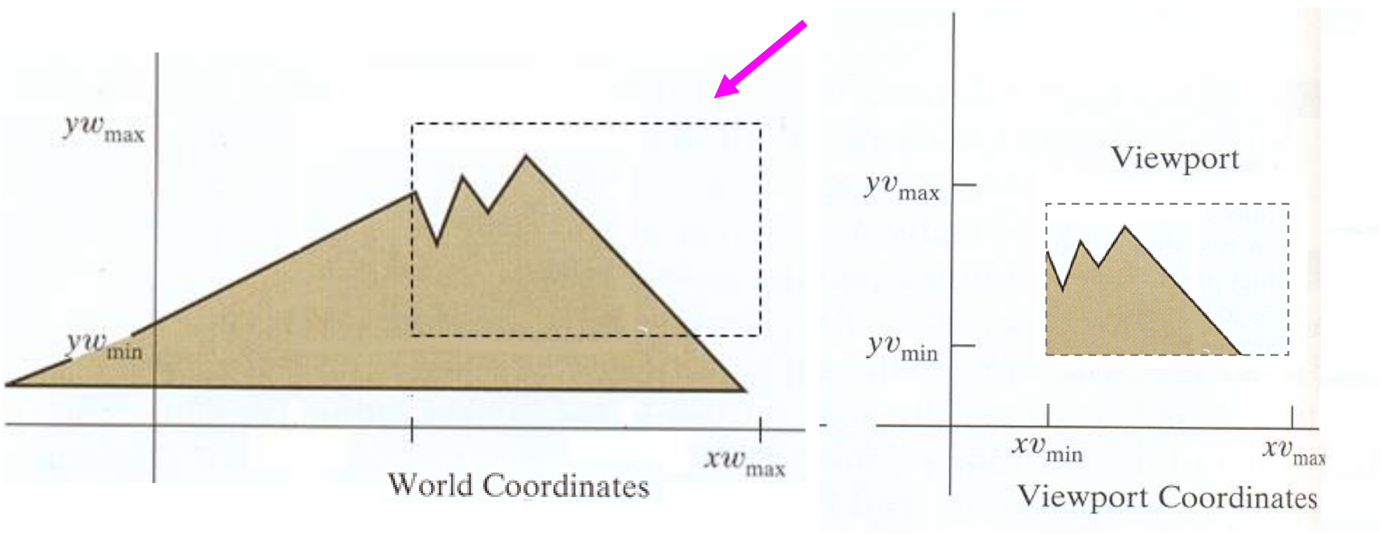
- Transformation between world coordinates and normalized device coordinates of the display

- In general, that transformation comprises
  - Translations
  - Rotations – When ?
  - Scalings

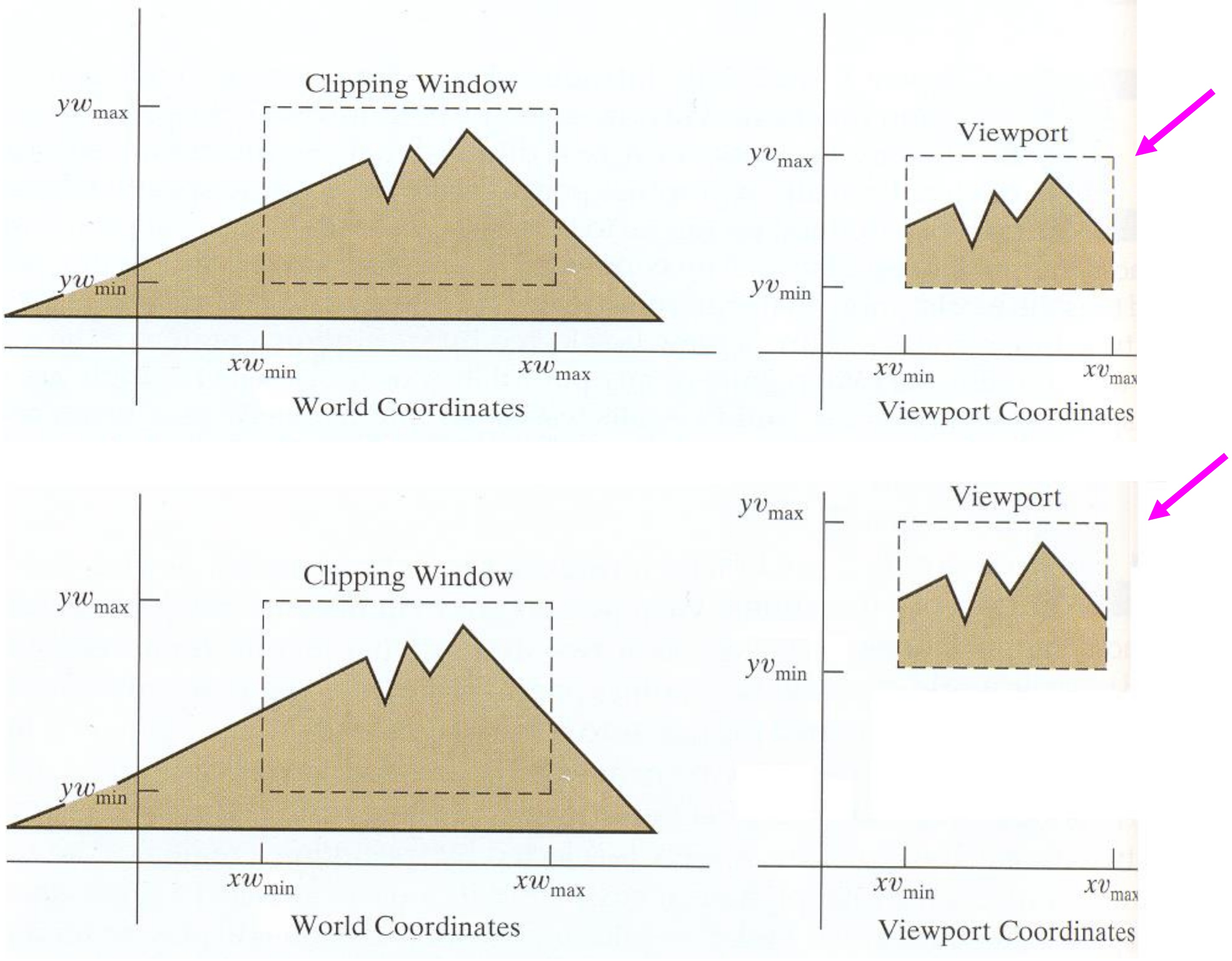
- Procedures are also carried out to remove parts of the scene that are outsider of the clipping window – clipping

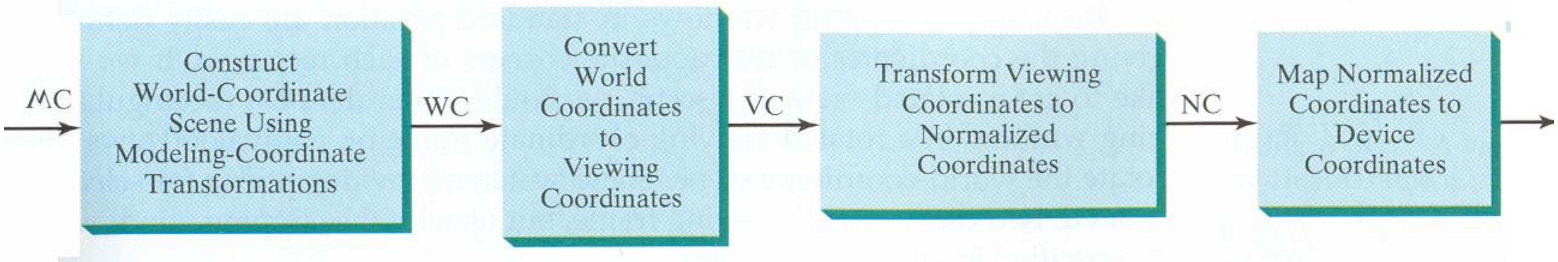
Moving the clipping window





Moving the viewport





- The transformation:

world coordinates → display device coordinates

is called 2D visualization transformation

or window-to-viewport transformation

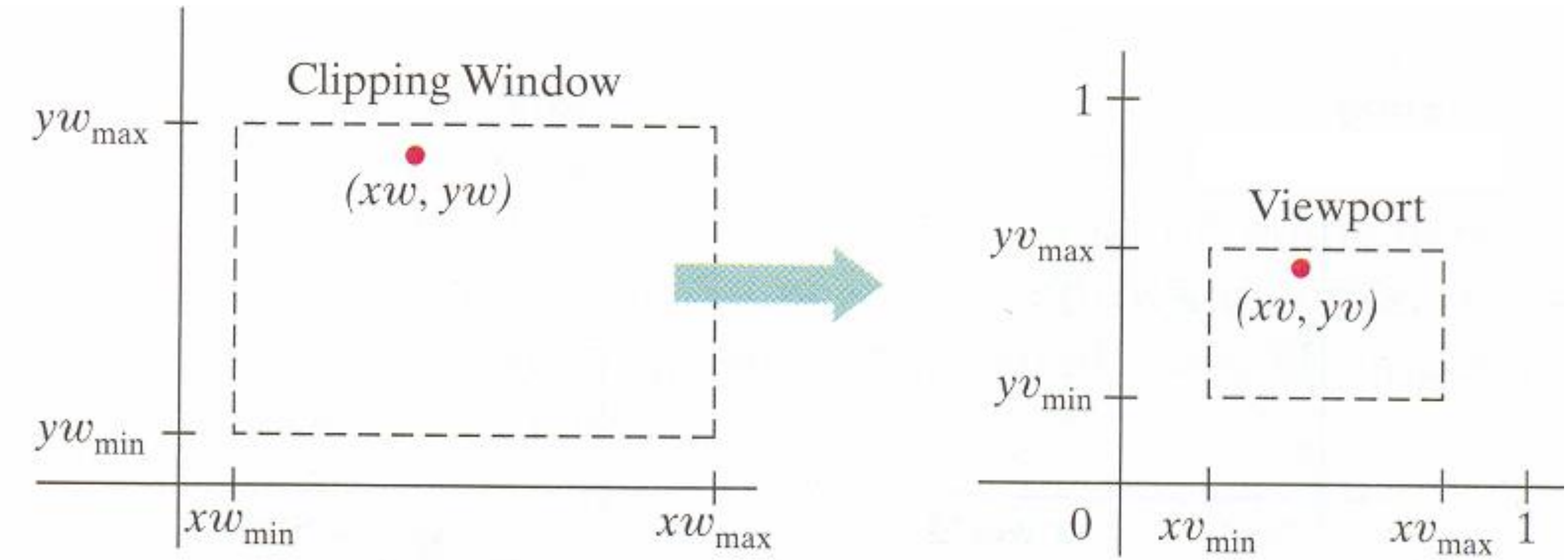
- Normalized coordinates allow for display device independence

- The clipping is usually carried out in normalized coordinates

THE WINDOW-TO-VIEWPORT TRANSFORMATION

Window-to-viewport transformation

- Given a clipping window and a viewport



- Point ***(xw, yw)*** is transformed into point ***(xv, yv)***

- Taking into account the distance ratios:

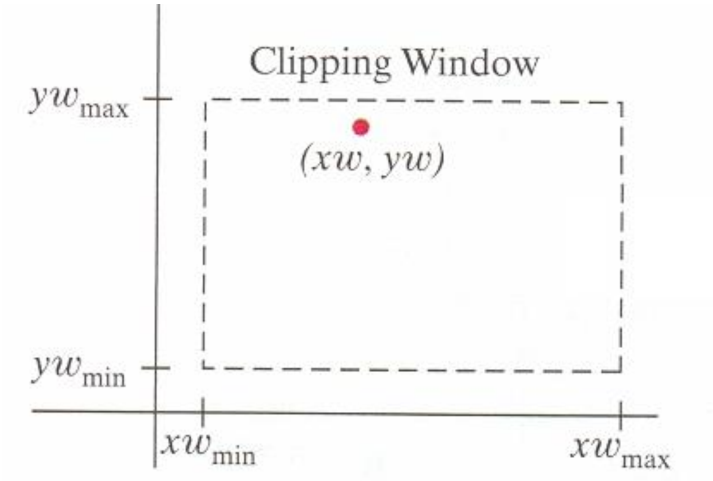
$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

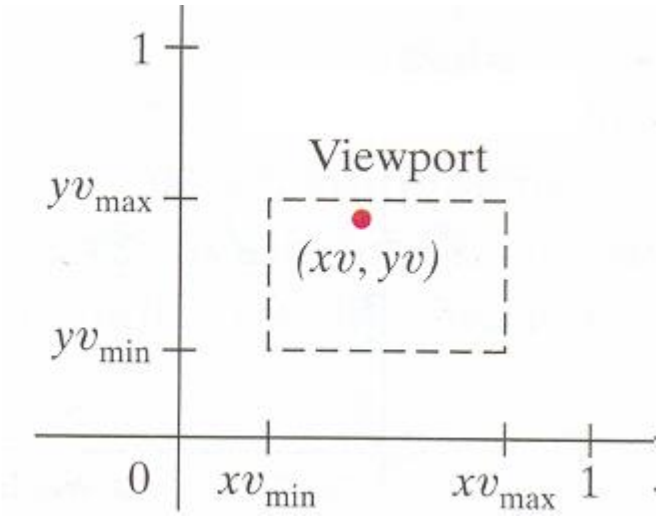
$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

- Solving for the unknowns:

$$xv = s_x xw + t_x$$

$$yv = s_y yw + t_y$$





- The scaling factors are:

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} \qquad sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

- Global displacements are given by:

$$t_x = \frac{xw_{max}xv_{min} - xw_{min}xv_{max}}{xw_{max} - xw_{min}}$$

$$t_y = \frac{yw_{max}yv_{min} - yw_{min}yv_{max}}{yw_{max} - yw_{min}}$$

- As an alternative, the window-to-viewport tranasformationcan be defined as a sequence of transformations which converts a rectangular clipping window into a rectangular viewport

1- scaling of the clipping window to the size of the viewport,

with fixed point ***(xwmin, ywmin)***

2- translation of ***(xwmin, ywmin)*** to ***(xvmin, yvmin)***

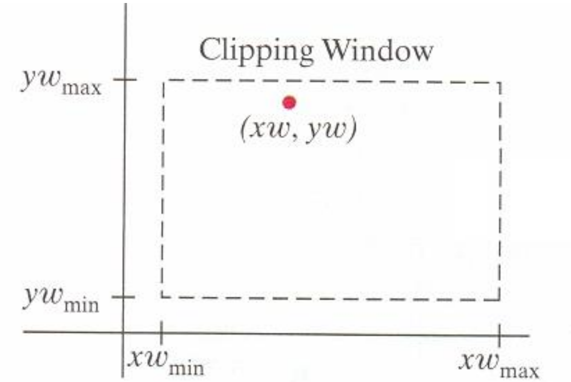
$$\mathbf{S} = \begin{bmatrix} s_x & 0 & xw_{\min}(1 - s_x) \\ 0 & s_y & yw_{\min}(1 - s_y) \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} 1 & 0 & xv_{\min} - xw_{\min} \\ 0 & 1 & yv_{\min} - yw_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{\text{window, normviewp}} = \mathbf{T} \cdot \mathbf{S} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- If ***sx*** = ***sy*** relative sizes are kept

- Another alternative, to define the window-to-viewport transformation considers carrying out a sequence of basic transformations:

which ones?

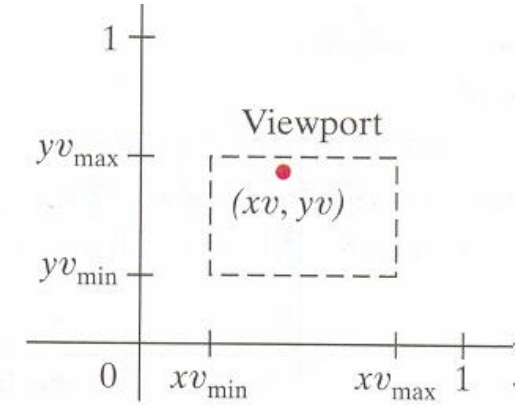


?

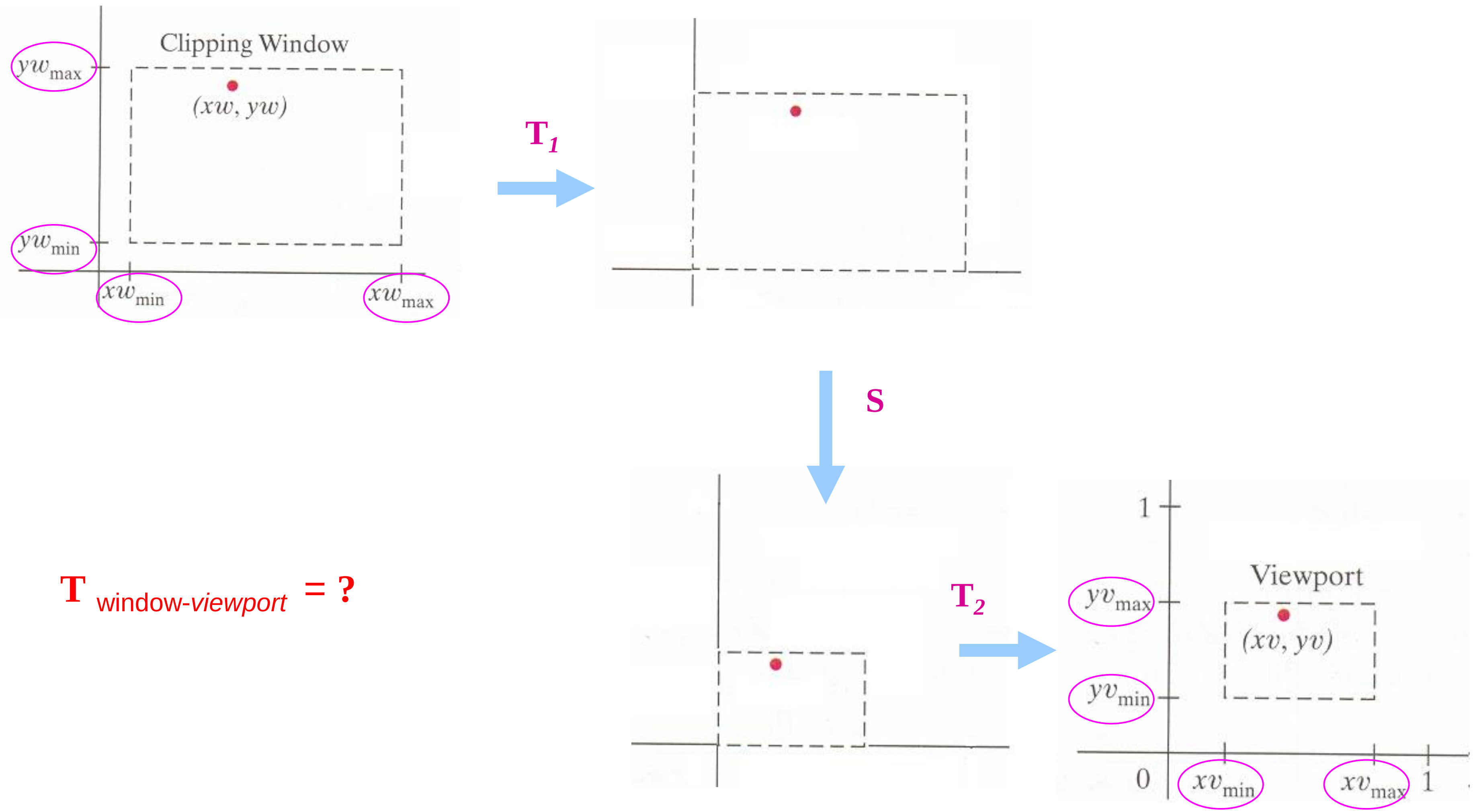
?

$\mathbf{T}_{\text{window-}\textit{viewport}} = ?$

?

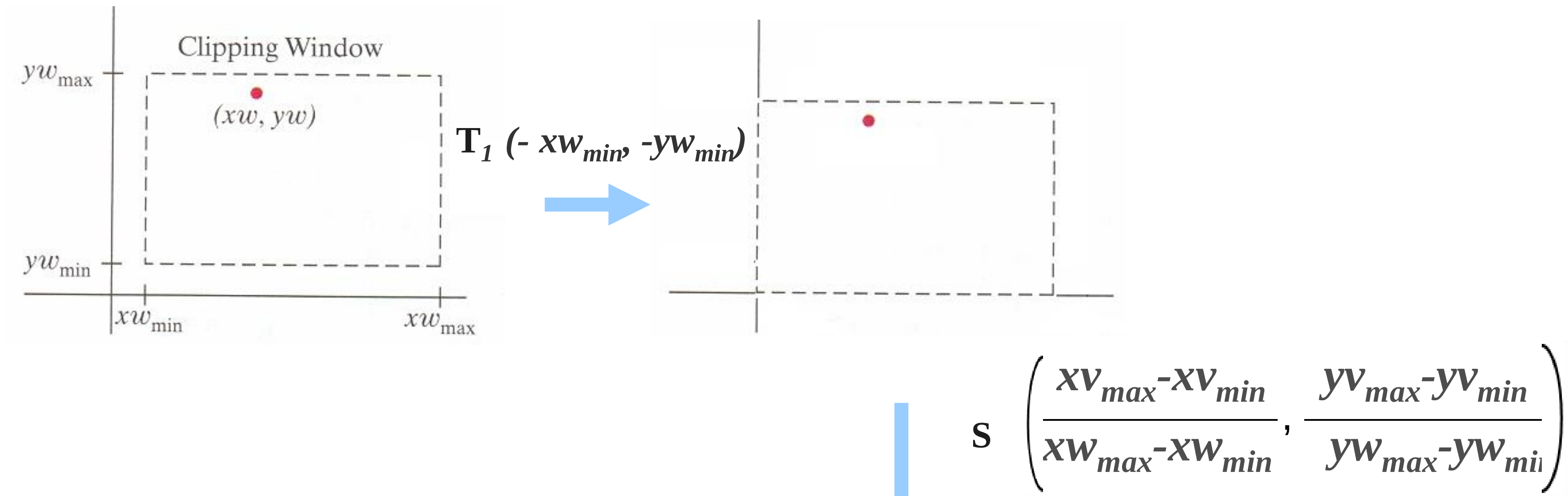


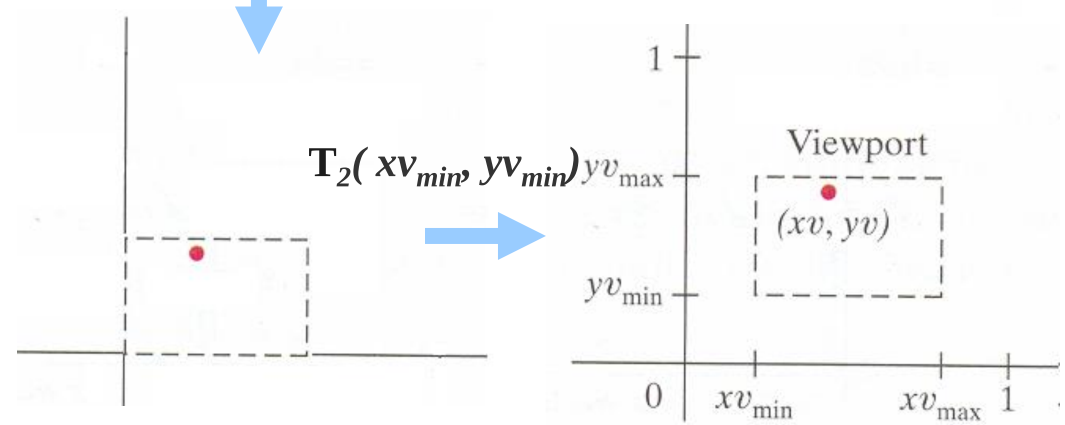
- Those basic transformations can be easily defined:



$\mathbf{T}_{\text{window-}\textit{viewport}} = \mathbf{?}$

- The global window-to-viewport transformation:



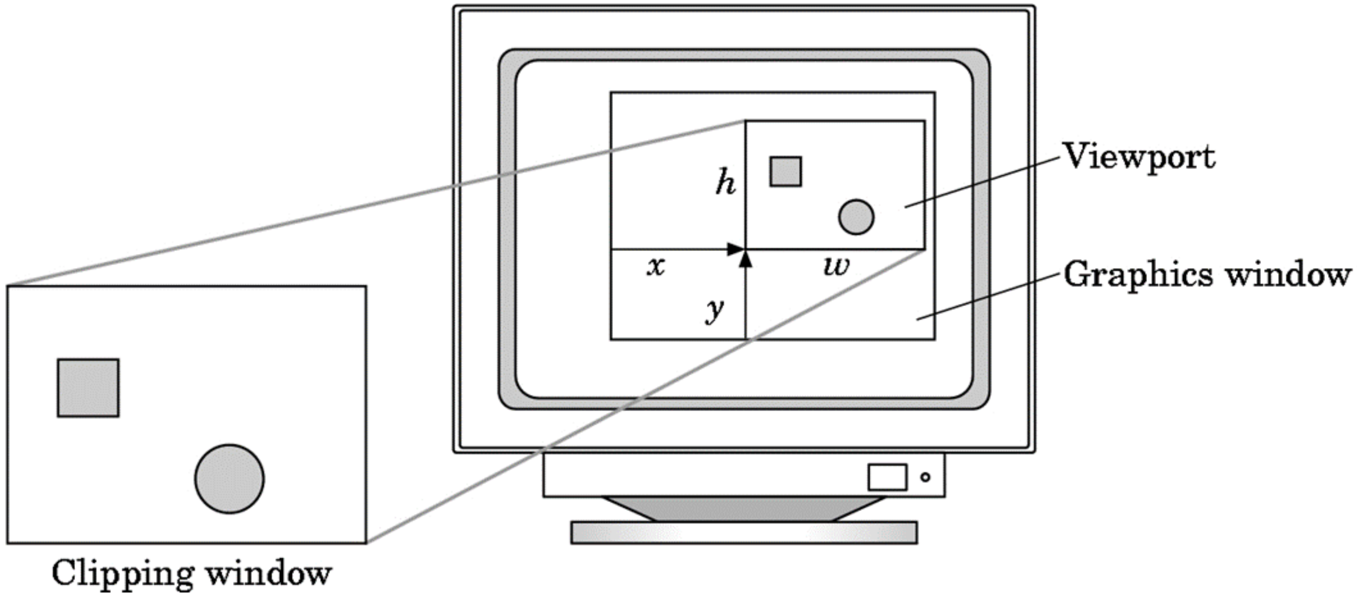


$\mathbf{T}_{\text{window-}\textit{viewport}} = \mathbf{T}_2 \,.\, \mathbf{S} \,.\, \mathbf{T}_1$

What remains to be done ?

- The final transformation:
  - From normalized device coordinates to the coordinate system of the canvas / output window
- Questions
  - Where is the origin of output window coordinates system?
  - What is the axes orientation?
  - Carry out a scaling and a displacement !!

2D Viewing



[Angel]

OpenGL (Pre-3.1 !!)

- Set the features of the clipping window and the viewport
  - The transformations matrices are instantiated !
- Clipping window

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
gluOrtho2D( xInf, xSup, yInf, ySup );
```

- Viewport
  - Set the transformation matrix from normalized coordinates to the coordinates of the output window

```
glViewport( x, y, width, height );
```

- What are the default values ?

WebGL

- Default clipping window
  - Square with corners (-1,-1) and (1,1)

- Default viewport uses the whole canvas !!

- Might not use the entire canvas for displaying

```
gl.viewport( x, y, width, height );
```

- Example

```
var canvas = document.getElementById('canvas1');
var gl = canvas.getContext('webgl');
canvas.width = newWidth;
canvas.height = newHeight;
gl.viewport(0, 0, gl.drawingBufferWidth,
                  gl.drawingBufferHeight);
```

[WebGL Specification]

REFERENCES

References

- D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*, 3rd Ed., Addison-Wesley, 2004

- E. Angel and D. Shreiner, *Introduction to Computer Graphics,* 6th Ed., Pearson Education, 2012

- J. Foley et al., *Introduction to Computer Graphics*, Addison-Wesley, 1993