World windows and viewports

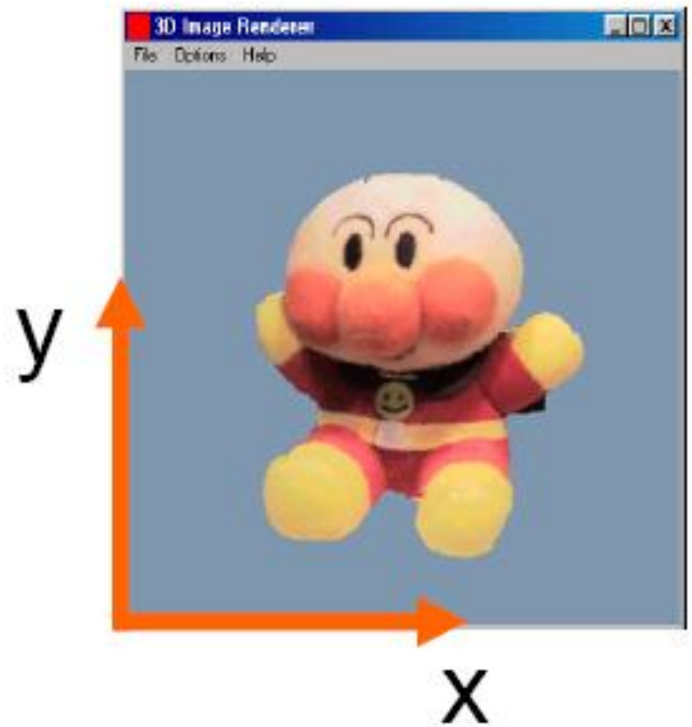
Lecture 3

Outline

- Screen coordinates
- World coordinate
- To introduce viewports and clipping
- To develop the window-to-viewport transformation

Screen coordinate systems

GLUT
Windows API
X Windows under Unix/Linux
Apple QuickDraw



OpenGL

Screen coordinates

- Basic coordinate system of the **screen window** uses coordinates that are in pixels, extending from
  – 0 to screenWidth-1 in x
  – 0 to screenHeight-1 in y

This means that we can use only positive values of x and y

Screen coordinates

- In a given problem, however, we may not want to think in terms of pixels

- It may be much more natural to think in terms of x varying from, say, -1 to 1, and y varying from -100.0 to 200.0

World coordinates

- We develop methods that let the programmer or user describe objects in whatever coordinate system best fits the problem

- The space in which objects are described is called **world coordinates**, *which are the usual Cartesian xy-coordinates used in mathematics,* based on whatever units are convenient.

World window

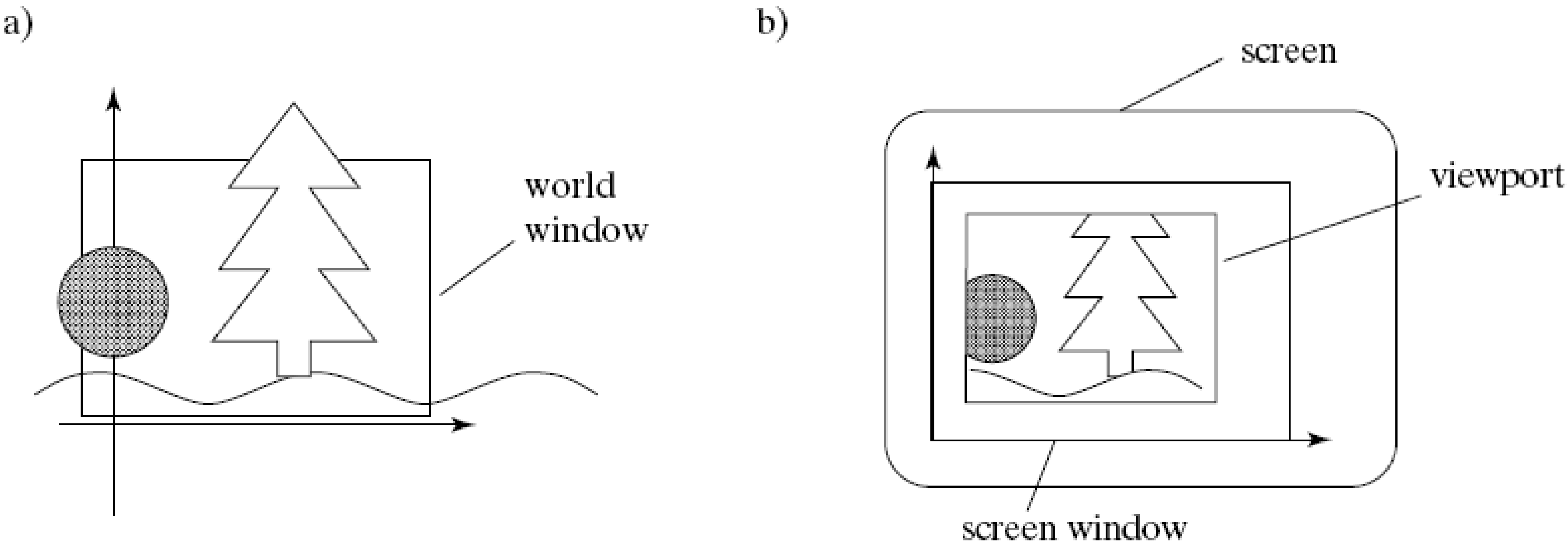
- We define a rectangular **world window** in world coordinates.
  - The world window specifies which part of the "world" should be drawn.
  - Whatever lies inside the window should be drawn and whatever lies outside should be clipped away or not drawn.

Viewport

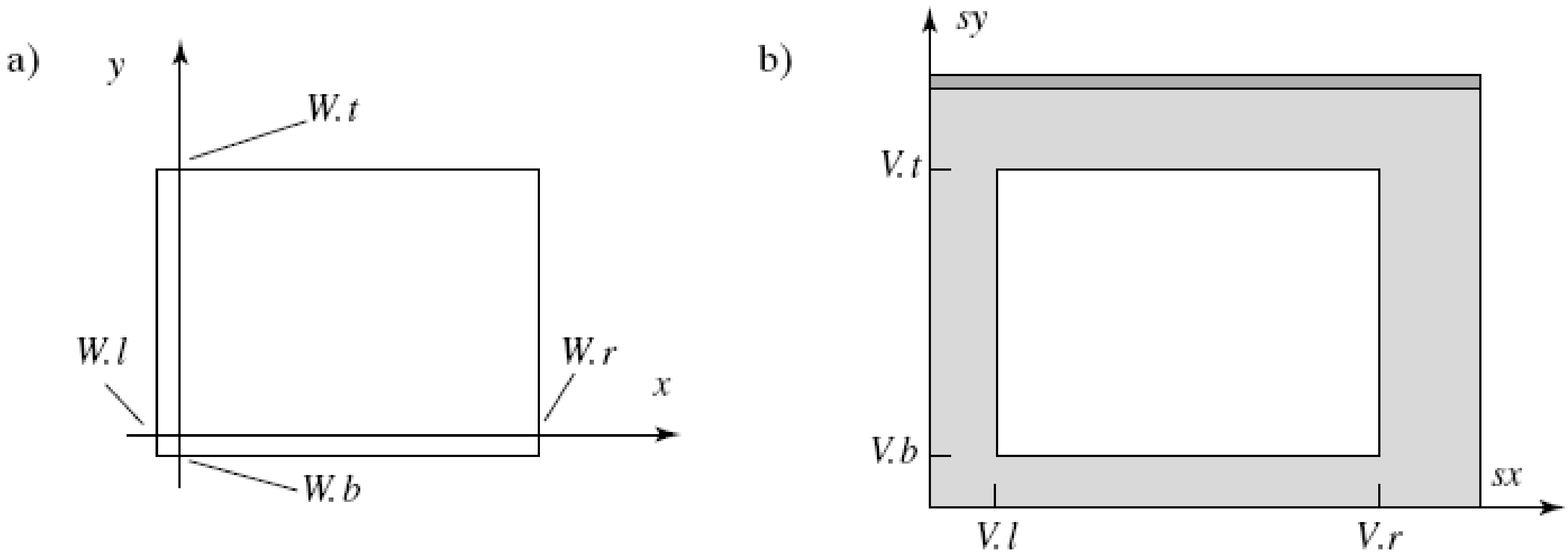
- In addition, we define a rectangular **viewport** in the screen window.

- When **mapping between the world window and the viewport**, the parts that lie inside the world window are automatically mapped to the inside of the viewport.

World window and viewport

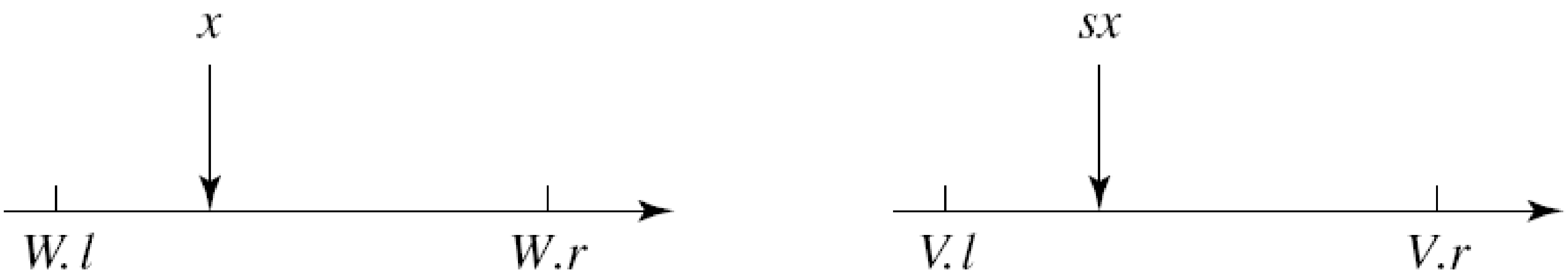


The mapping from the Window to the Viewport



Mapping 1/3

- Mapping or transformation called **window-to-viewport mapping**.



$$sx = Ax + C$$
$$sy = By + D$$

How can A, B, C, and D be determined?

Mapping 2/3

$$\frac{sx - V.l}{V.r - V.l} = \frac{x - W.l}{W.r - W.l}$$

$$sx = \underbrace{\frac{V.r - V.l}{W.r - W.l}}_{A} x + \underbrace{V.l - \frac{V.r - V.l}{W.r - W.l} W.l}_{C}$$

$$C = V.l - A\ W.l$$

Mapping 3/3

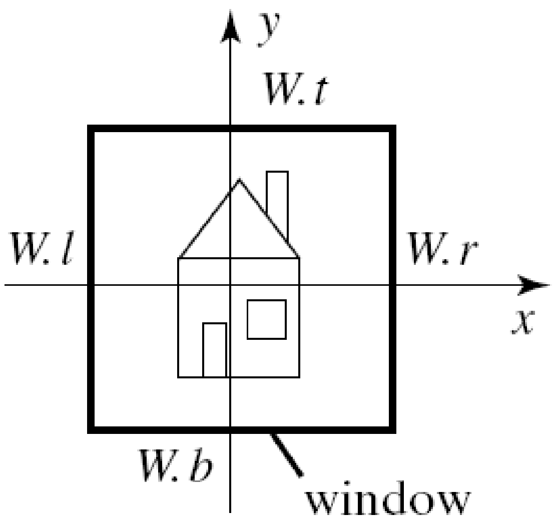
$$\frac{sy - V.b}{V.t - V.b} = \frac{y - W.b}{W.t - W.b}$$

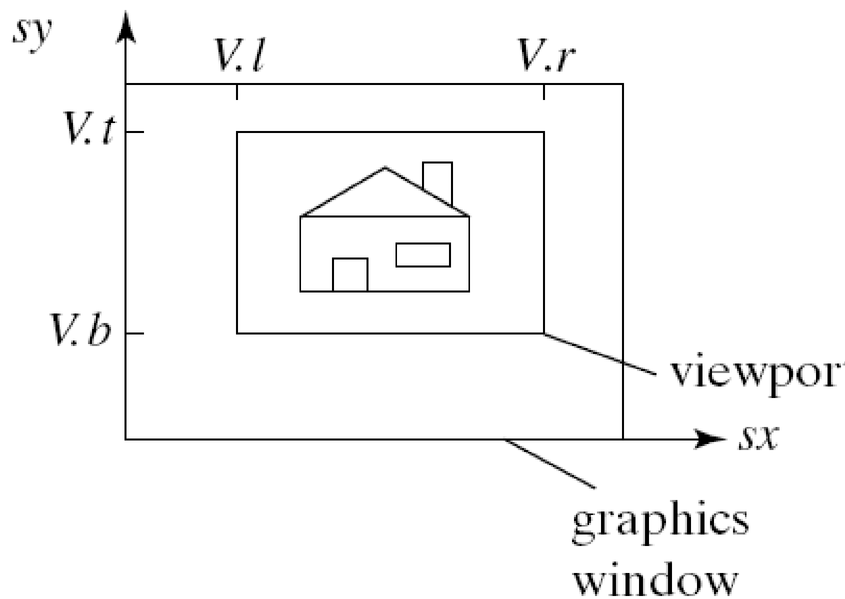
$$B = \frac{V.t - V.b}{W.t - W.b}$$

$$D = V.b - B\ W.b$$

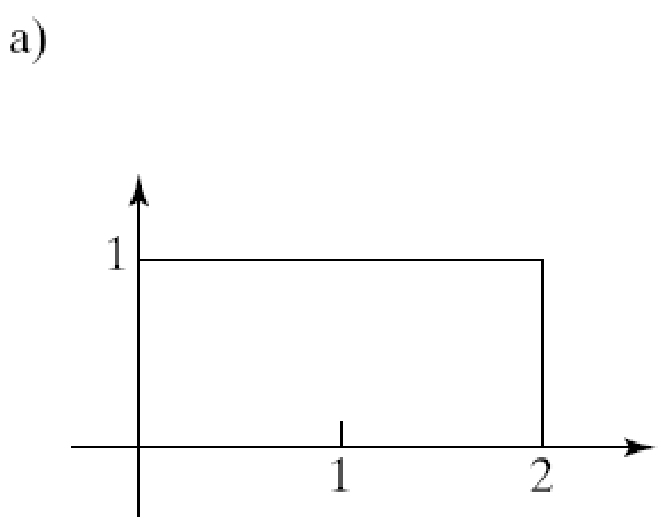
The mapping from the Window to the Viewport

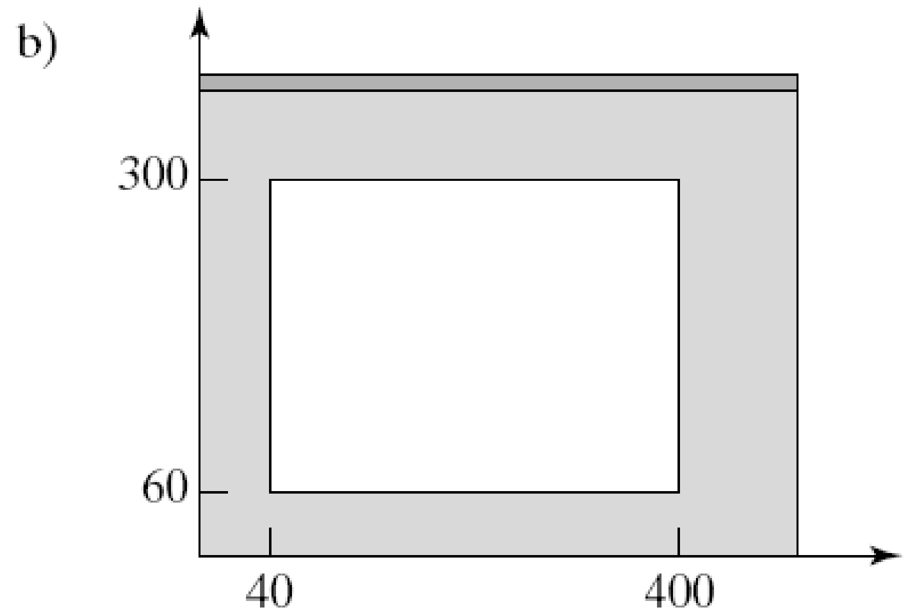
- The world window and viewport do not have to have the same aspect ratio





Example





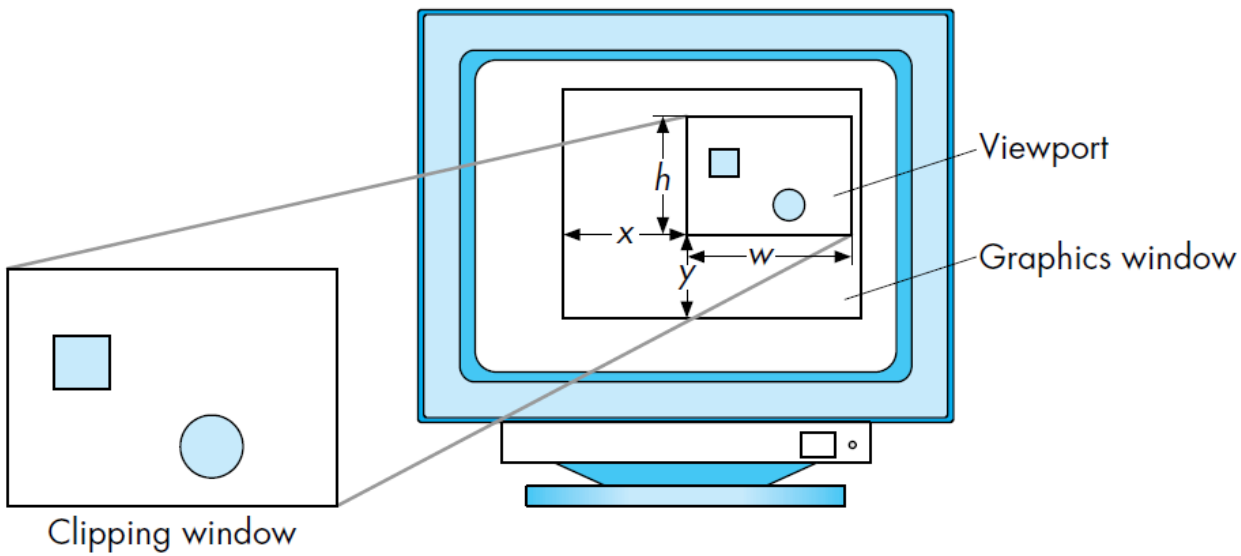

```
(W.l, W.r, W.b, W.t) = (0, 2.0, 0, 1.0)
(V.l, V.r, V.b, V.t) = (40, 400, 60, 300)
```

Using the formula A=180, C=40, B=240, and D=60

```
sx = 180x + 40
sy = 240y + 60
```

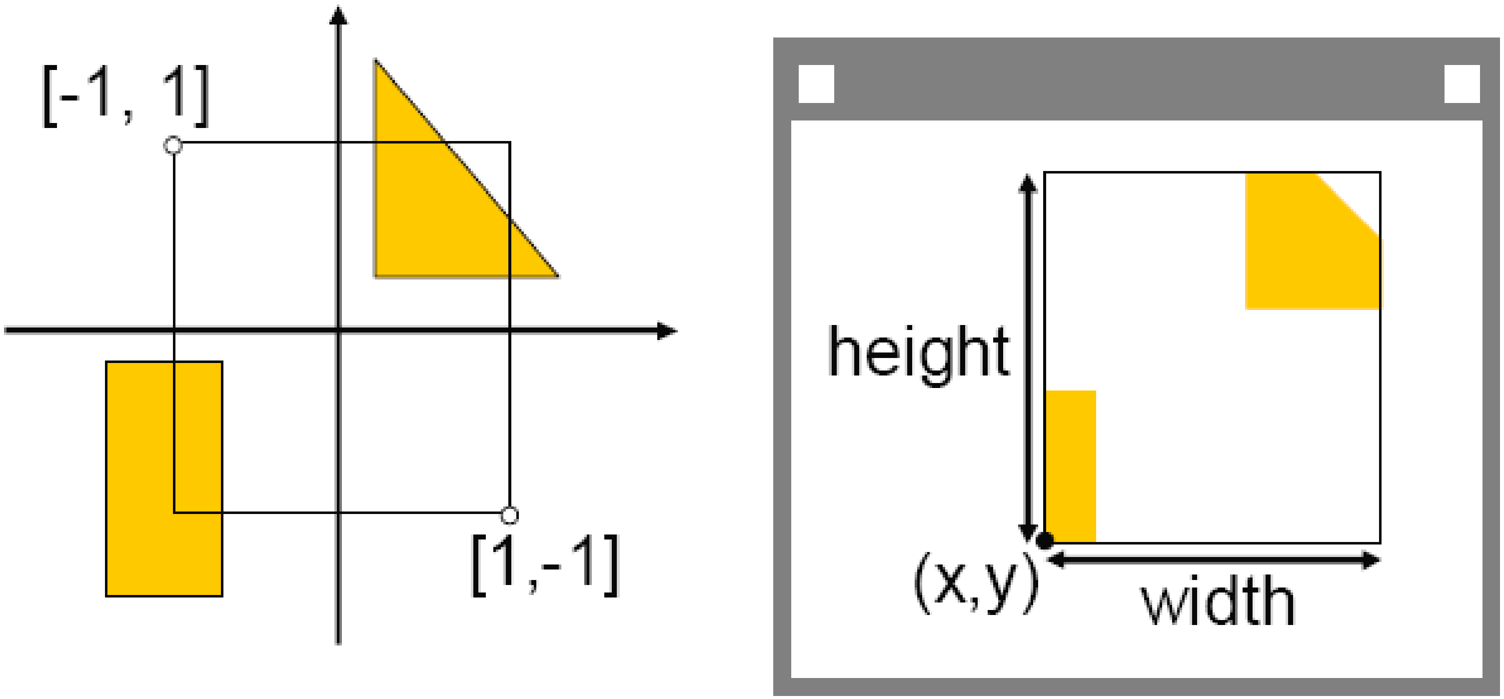
Doing it in OpenGL

- For 2D drawing, the world window is set by the function **gluOrtho2D()** and the viewport is set by the function **glViewport()**



Doing it in OpenGl

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



Clip Space

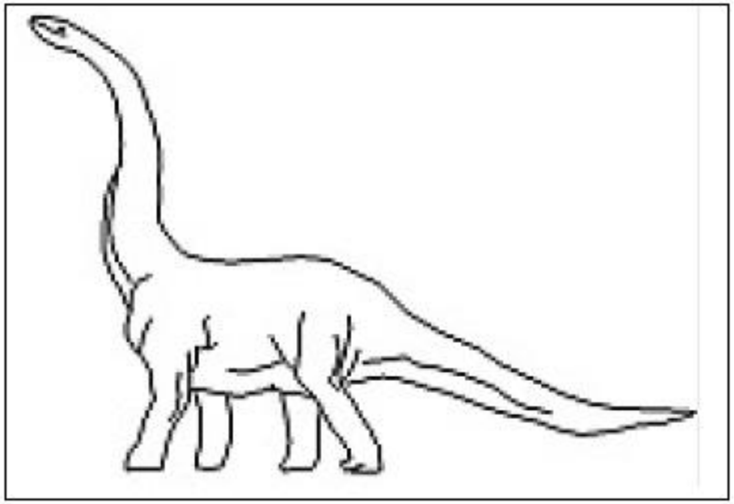
Viewport Space

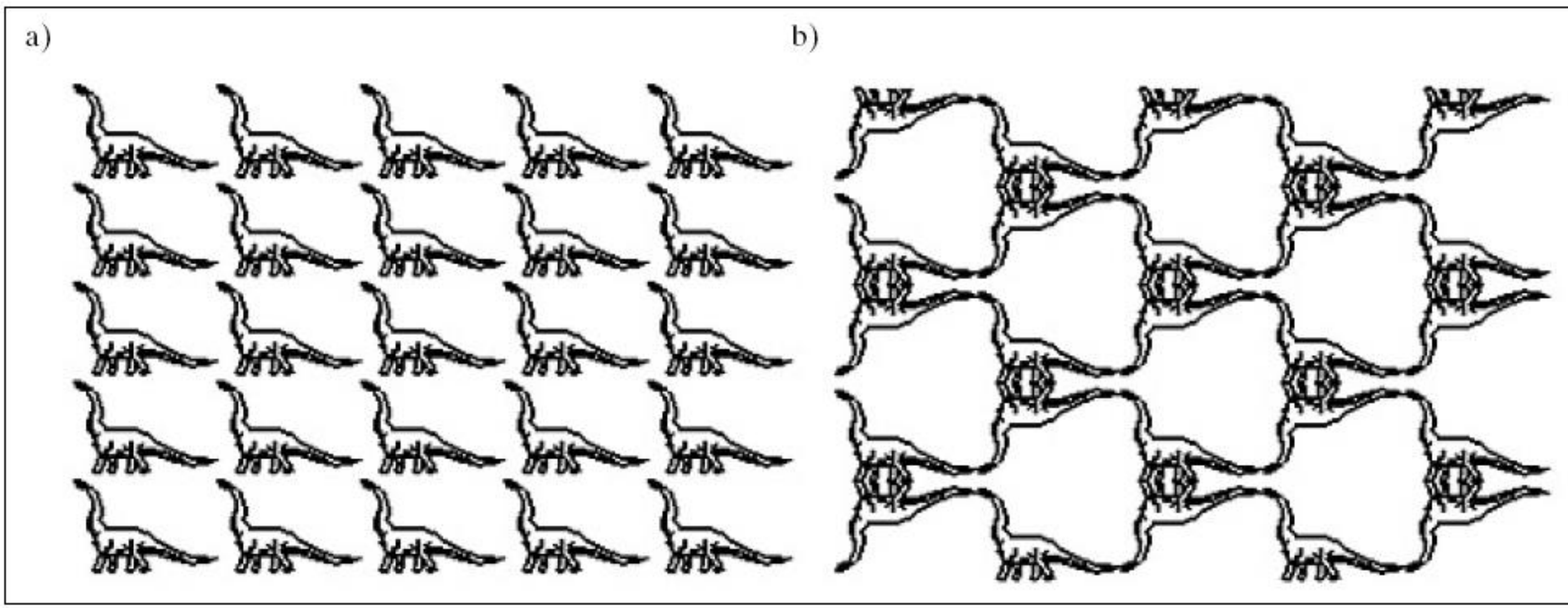
```
gluOrtho2D(left, right, bottom, top)
```

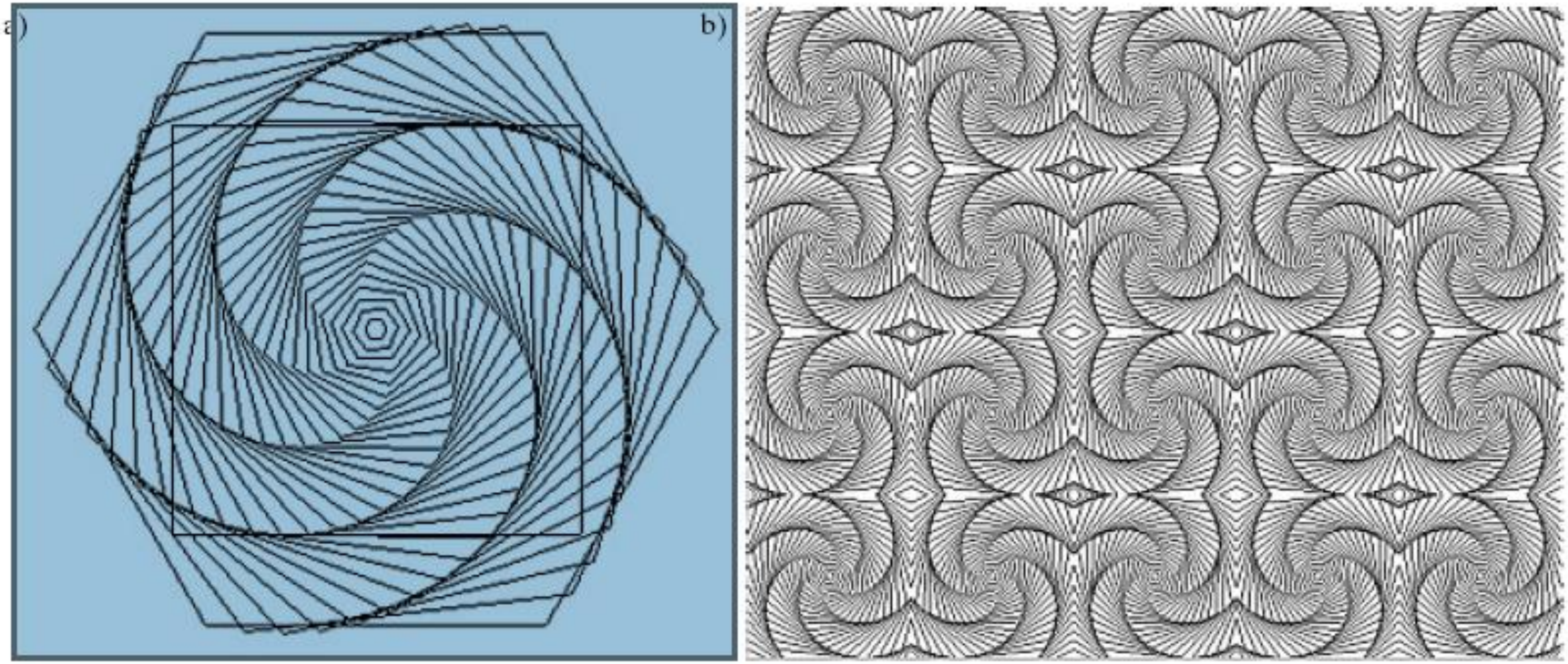
glutReshapeFunc(resize);

```
void resize (GLsizei w, GLsizei h) {
    if (h==0) h=1;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w<=h) {
        winHeight=250.0f*h/w;
        winWidth=250.0f;
    } else {
        winWidth=250.0f*w/h;
        winHeight=250.0f;
    }
    glOrtho(0.0f, winWidth, 0.0f, winHeight, 1.0f, -1.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```




a)
b)


a)
b)

```
void Displaysinex ( void ) {
GLfloat degtorads, x;
int i;
degtorads=3.14159265/180.0; //degrees to radians
glClear( GL_COLOR_BUFFER_BIT);
for(i=0; i<5; i++) {
        //5 different viewports
        glViewport ( rand() % 100,rand() % 50,
                                    100 , 50 );
        glBegin(GL_LINE_STRIP);
        for(x=0.0; x<=360.0; x +=1.0) {
                //sine curve between 0 and 2π
                glColor3f(1.0f, (1.0- (float) x /360.0), 0.0f);
                glVertex2f(x, sin(x*degtorads));
        }
        glEnd();
}
glFlush();
}
```