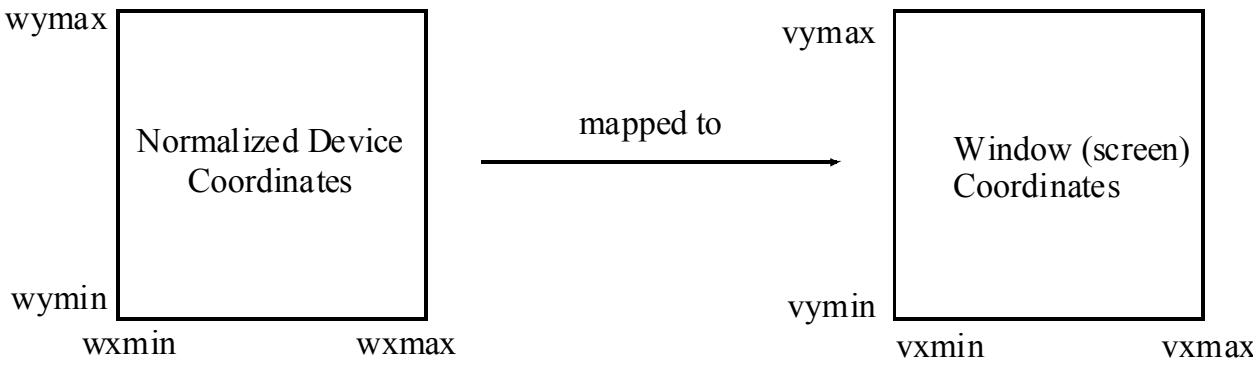
8. Viewport Transformations

The viewport transformation maps Normalized device coordinates into *window (screen)* coordinates

- The viewport is the rectangular region of the window where the image is drawn
- Normally the viewport is measured in window coordinates relative to the lower-left corner of the window
- The viewport transformation determines the actual size in pixels of the displayed object



To map (wx, wy) in normalized device coordinates to (vx, vy) in window coordinates:

vx = ((vxmax - vxmin)/(wxmax - wxmin))*(wx - wxmin) + vxmin
(8.1)

vy = ((vymax - vymin)/(wymax - wymin))*(wy - wymin) + vymin
(8.2)

In order to avoid distortion, the aspect ratio in normalized device coordinates should equal the aspect ratio of the viewport coordinates:

$$\frac{(wx\max - wy\min)}{(wy\max - wx\min))} = \frac{(vx\max - vx\min)}{(vy\max - vy\min))} \qquad (8.3)$$

- Unequal aspect ratios will cause objects to appear stretched or squished
- In order to preserve the aspect ratio, it may be necessary have a viewport which is smaller than the window

In OpenGL, by default, the viewport is set to the entire client area of the drawing window

void glViewport(GLint *x*, GLint *y*, GLsizei *width*, GLsizei *height*);

is used to

- explicitly set the viewport
- define multiple viewports in a single window

Parameters

- (*x, y*) specifies the lower left corner of the viewport
- width and height specify the width and height of the viewport in pixels

Example 8.1

Maintaining equal aspect ratios

```
void COpenGLView::OnSize( UINT nType, int cx, int cy )
{
   CView::OnSize( nType, cx, cy );

      winSizeY = (cy == 0) ? 1 : cy;
      winSizeX = (cx == 0) ? 1 : cx;

      // Set viewport to entire client area
      glViewport(0, 0, winSizeX, winSizeY);

      // Projection matrix stack
      glMatrixMode (GL_PROJECTION);

      // Reset the projection marix stack
      glLoadIdentity();

      // Set up a perspective projection matrix
      gluPerspective(45.0f,
                (GLfloat)winSizeX/(GLfloat)winSizeY,
                1.0f, 50.0f);

      // Modelview matrix stack
      glMatrixMode (GL_MODELVIEW);            /

} // end OnSize
```

Example 8.2

Drawing two different views in a split window.

```
void COpenGLView::OnDraw(CDC* pDC)
{
     glViewport( 0, 0, sizex/2, sizey );

     // Projection matrix stack
     glMatrixMode (GL_PROJECTION);

     // Reset the projection marix stack
     glLoadIdentity();

     gluPerspective(45.0f,
           (GLfloat)(winSizeX/2.0)/(GLfloat)winSizeY,
           1.0f, 50.0f);

     drawRightSideView();

     glViewport( sizex/2, 0, sizex/2, sizey );

     drawLeftSideView();

     glFlush();
     SwapBuffers( m_pDC->GetSafeHdc() );

     // Invalidate the client area
     // so another OnDraw message
     // will be posted
     InvalidateRect( 0, FALSE );
     GetParent() -> PostMessage( WM_PAINT );

} // end OnDraw
```