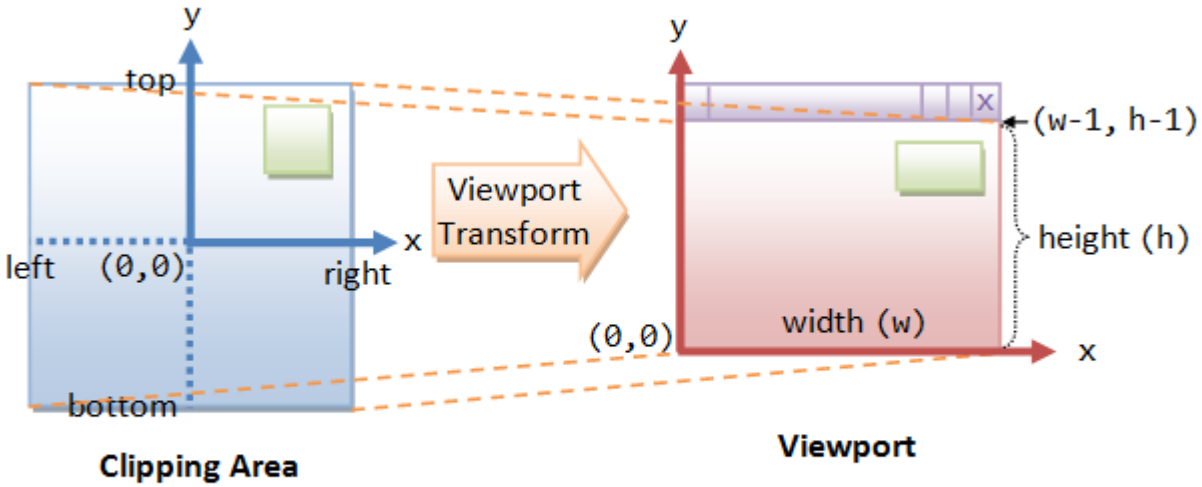
Clipping-Area & Viewport

Try dragging the corner of the window to make it bigger or smaller. Observe that all the shapes are distorted.

We can handle the re-sizing of window via a callback handler `reshape()`, which can be programmed to adjust the OpenGL clipping-area according to the window's aspect ratio.



Clipping Area and Viewport: Objects will be distorted if the aspect ratios of the clipping area and viewport are different.

Clipping Area: *Clipping area* refers to the area that can be seen (i.e., captured by the camera), measured in OpenGL coordinates.

The function `gluOrtho2D` can be used to set the clipping area of 2D orthographic view. Objects outside the clipping area will be *clipped* away and cannot be seen.

```
void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)
   // The default clipping area is (-1.0, 1.0, -1.0, 1.0) in OpenGL coordinates,
   // i.e., 2x2 square centered at the origin.
```

To set the clipping area, we need to issue a series of commands as follows: we first select the so-called *projection matrix* for operation, and reset the projection matrix to identity. We then choose the 2D orthographic view with the desired clipping area, via `gluOrtho2D()`.

```
// Set to 2D orthographic projection with the specified clipping area
glMatrixMode(GL_PROJECTION);      // Select the Projection matrix for operation
glLoadIdentity();                 // Reset Projection matrix
gluOrtho2D(-1.0, 1.0, -1.0, 1.0); // Set clipping area's left, right, bottom, top
```

Viewport: *Viewport* refers to the display area on the window (screen), which is measured in pixels in screen coordinates (excluding the title bar).

The clipping area is mapped to the viewport. We can use `glViewport` function to configure the viewport.

```
void glViewport(GLint xTopLeft, GLint yTopLeft, GLsizei width, GLsizei height)
```

Suppose the the clipping area's (left, right, bottom, top) is (-1.0, 1.0, -1.0, 1.0) (in OpenGL coordinates) and the viewport's (xTopLeft, xTopRight, width, height) is (0, 0, 640, 480) (in screen coordinates in

pixels), then the bottom-left corner (-1.0, -1.0) maps to (0, 0) in the viewport, the top-right corner (1.0, 1.0) maps to (639, 479). It is obvious that if the *aspect ratios* for the clipping area and the viewport are not the same, the shapes will be distorted.

Take note that in the earlier example, the windows' size of 320x320 has a square shape, with a aspect ratio consistent with the default 2x2 squarish clipping-area.

4.1 Example 3: Clipping-area and Viewport (`GL03Viewport.cpp`)

```
 1/*
 2 * GL03Viewport.cpp: Clipping-area and Viewport
 3 * Implementing reshape to ensure same aspect ratio between the
 4 * clipping-area and the viewport.
 5 */
 6#include <windows.h>  // for MS Windows
 7#include <GL/glut.h>  // GLUT, include glu.h and gl.h
 8
 9/* Initialize OpenGL Graphics */
10void initGL() {
11   // Set "clearing" or background color
12   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
13}
14
15void display() {
16   glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color
17
18   // Define shapes enclosed within a pair of glBegin and glEnd
19   glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
20      glColor3f(1.0f, 0.0f, 0.0f); // Red
21      glVertex2f(-0.8f, 0.1f);     // Define vertices in counter-clockwise (CCW) order
22      glVertex2f(-0.2f, 0.1f);     //  so that the normal (front-face) is facing you
23      glVertex2f(-0.2f, 0.7f);
24      glVertex2f(-0.8f, 0.7f);
25
26      glColor3f(0.0f, 1.0f, 0.0f); // Green
27      glVertex2f(-0.7f, -0.6f);
28      glVertex2f(-0.1f, -0.6f);
29      glVertex2f(-0.1f,  0.0f);
30      glVertex2f(-0.7f,  0.0f);
31
32      glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
33      glVertex2f(-0.9f, -0.7f);
34      glColor3f(1.0f, 1.0f, 1.0f); // White
35      glVertex2f(-0.5f, -0.7f);
36      glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
37      glVertex2f(-0.5f, -0.3f);
38      glColor3f(1.0f, 1.0f, 1.0f); // White
```

```
39      glVertex2f(-0.9f, -0.3f);
40   glEnd();
41
42   glBegin(GL_TRIANGLES);          // Each set of 3 vertices form a triangle
43      glColor3f(0.0f, 0.0f, 1.0f); // Blue
44      glVertex2f(0.1f, -0.6f);
45      glVertex2f(0.7f, -0.6f);
46      glVertex2f(0.4f, -0.1f);
47
48      glColor3f(1.0f, 0.0f, 0.0f); // Red
49      glVertex2f(0.3f, -0.4f);
50      glColor3f(0.0f, 1.0f, 0.0f); // Green
51      glVertex2f(0.9f, -0.4f);
52      glColor3f(0.0f, 0.0f, 1.0f); // Blue
53      glVertex2f(0.6f, -0.9f);
54   glEnd();
55
56   glBegin(GL_POLYGON);            // These vertices form a closed polygon
57      glColor3f(1.0f, 1.0f, 0.0f); // Yellow
58      glVertex2f(0.4f, 0.2f);
59      glVertex2f(0.6f, 0.2f);
60      glVertex2f(0.7f, 0.4f);
61      glVertex2f(0.6f, 0.6f);
62      glVertex2f(0.4f, 0.6f);
63      glVertex2f(0.3f, 0.4f);
64   glEnd();
65
66   glFlush();  // Render now
67}
68
69/* Handler for window re-size event. Called back when the window first appears and
70   whenever the window is re-sized with its new width and height */
71void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
72   // Compute aspect ratio of the new window
73   if (height == 0) height = 1;                // To prevent divide by 0
74   GLfloat aspect = (GLfloat)width / (GLfloat)height;
75
76   // Set the viewport to cover the new window
77   glViewport(0, 0, width, height);
78
79   // Set the aspect ratio of the clipping area to match the viewport
80   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
81   glLoadIdentity();             // Reset the projection matrix
82   if (width >= height) {
83     // aspect >= 1, set the height from -1 to 1, with larger width
```

```
 84      gluOrtho2D(-1.0 * aspect, 1.0 * aspect, -1.0, 1.0);
 85   } else {
 86      // aspect < 1, set the width to -1 to 1, with larger height
 87     gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);
 88   }
 89}
 90
 91/* Main function: GLUT runs as a console application starting at main() */
 92int main(int argc, char** argv) {
 93   glutInit(&argc, argv);          // Initialize GLUT
 94   glutInitWindowSize(640, 480);   // Set the window's initial width & height - non-squa
 95   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
 96   glutCreateWindow("Viewport Transform");  // Create window with the given title
 97   glutDisplayFunc(display);       // Register callback handler for window re-paint even
 98   glutReshapeFunc(reshape);       // Register callback handler for window re-size event
 99   initGL();                       // Our own OpenGL initialization
100   glutMainLoop();                 // Enter the infinite event-processing loop
101   return 0;
102}
```

A `reshape()` function, which is called back when the window first appears and whenever the window is re-sized, can be used to ensure consistent aspect ratio between clipping-area and viewport, as shown in the above example. The graphics sub-system passes the window's width and height, in pixels, into the `reshape()`.

```
GLfloat aspect = (GLfloat)width / (GLfloat)height;
```

We compute the aspect ratio of the new re-sized window, given its new `width` and `height` provided by the graphics sub-system to the callback function `reshape()`.

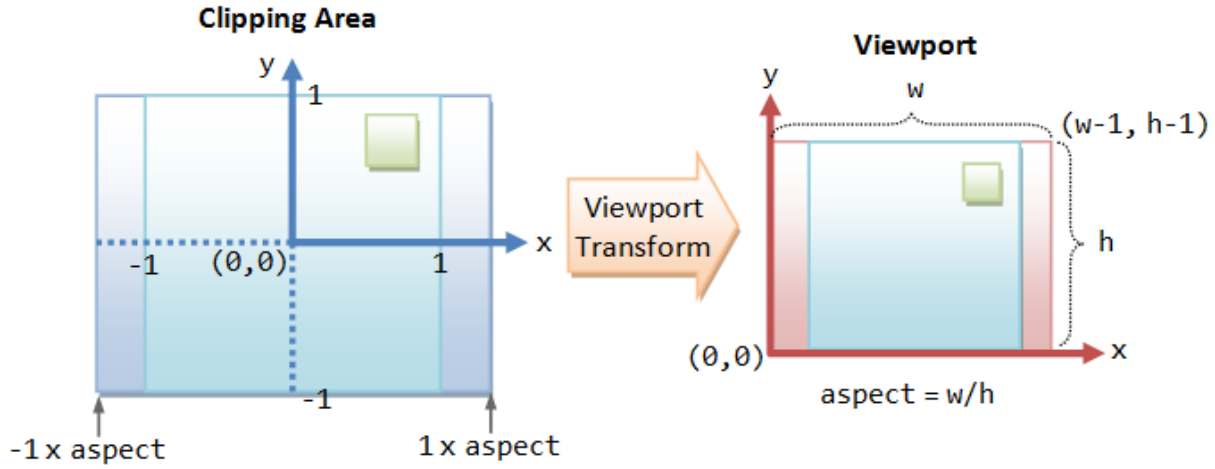
```
glViewport(0, 0, width, height);
```

We set the viewport to cover the entire new re-sized window, in pixels. Try setting the viewport to cover only a quarter (lower-right qradrant) of the window via `glViewport(0, 0, width/2, height/2)`.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (width >= height) {
   gluOrtho2D(-1.0 * aspect, 1.0 * aspect, -1.0, 1.0);
} else {
   gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);
}
```

We set the aspect ratio of the clipping area to match the viewport. To set the clipping area, we first choose the operate on the projection matrix via `glMatrixMode(GL_PROJECTION)`. OpenGL has two

matrices, a projection matrix (which deals with camera projection such as setting the clipping area) and a model-view matrix (for transforming the objects from their local spaces to the common world space). We reset the projection matrix via `glLoadIdentity()`.
Finally, we invoke `gluOrtho2D()` to set the clipping area with an aspect ratio matching the viewport. The shorter side has the range from -1 to +1, as illustrated below:



Clipping Area and Viewport: same aspect ratio for the clipping area and viewport to ensure that the objects are not distorted.

We need to register the `reshape()` callback handler with GLUT via `glutReshapeFunc()` in the `main()` as follows:

```
int main(int argc, char** argv) {
   glutInitWindowSize(640, 480);
   ......
   glutReshapeFunc(reshape);
}
```

In the above `main()` function, we specify the initial window size to `640x480`, which is non-squarish. Try re-sizing the window and observe the changes.
Note that the `reshape()` runs at least *once* when the window first appears. It is then called back whenever the window is re-shaped. On the other hand, the `initGL()` runs once (and only once); and the `display()` runs in response to window re-paint request (e.g., after the window is re-sized).