# First Projection

# Learning Outcomes

- Review the glRect method
- Understand how to compute a window *Aspect Ratio*
- Understand the role of the glutReshapeFunc()
- Have seen an orthographic projection in action using the glOrtho() function

# Draw a Rectangle

- glColor3f() selects a color in the same manner as glClearColor, but no alpha translucency component needs to be specified (the default value for alpha is 1.0 for completely opaque)

- glRectf() function takes floating-point arguments, represent two coordinate pairs, (x1, y1) and (x2, y2)

- The first pair represents the upper-left corner of the rectangle, and the second pair represents the lower-right corner.

```
void renderScene(void)
{
  glClear( GL_COLOR_BUFFER_BIT);

  glColor3f(1.0f, 0.0f, 0.0f);

  glRectf(-25.0f, 25.0f, 25.0f, -25.0f);

  glFlush();
}
```

# Scaling to the Window

- In most windowing environments, the user can at any time change the size and dimensions of the window.

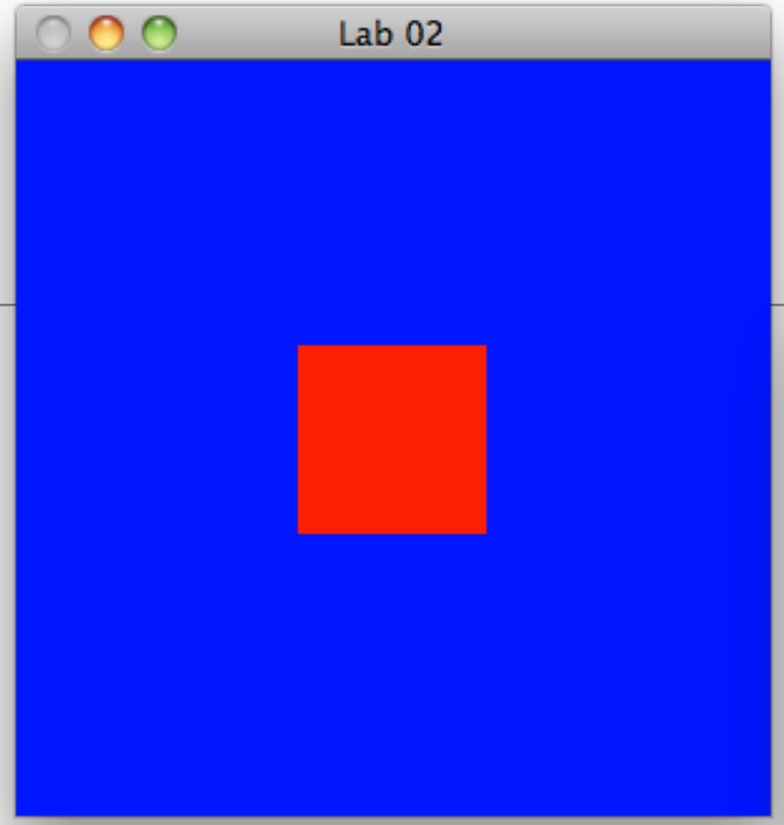
- Even if you are writing a game that always runs in fullscreen mode, the window is still considered to change size once—when it is created.

- When this happens, the window usually responds by redrawing its contents, taking into consideration the window’s new dimensions.

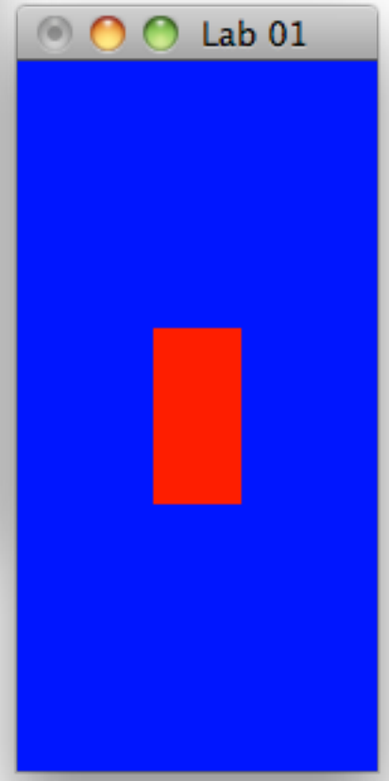
- May wish to scale the drawing to fit within the window,regardless of the size of the drawing or window.

# Aspect Ratio

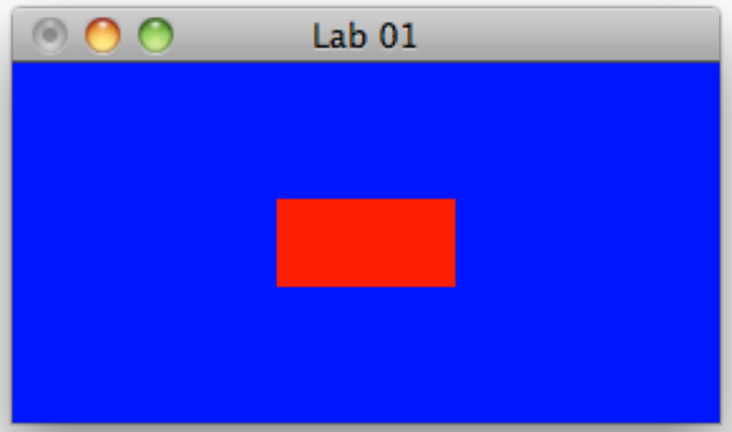
- The aspect ratio is the ratio of the number of pixels along a unit of length in the vertical direction to the number of pixels along the same unit of length in the horizontal direction.
  - I.e. The width of the window divided by the height.
- An aspect ratio of 1.0 defines a square aspect ratio.
- An aspect ratio of 0.5 indicates the width is half the height
- An aspect ratio of 2 indicates the height is half the width




1




0.5




2

# glutReshapeFunc()

- callback function for when ever the window changes size (when it is stretched, maximized, and so on)

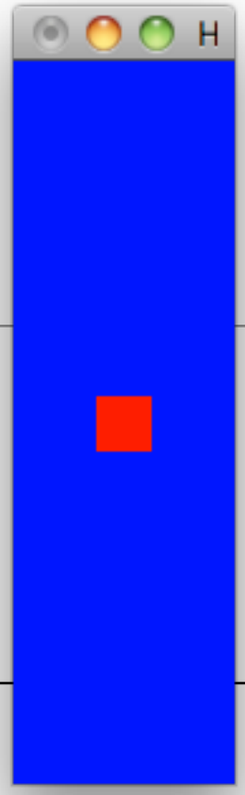
- The changeSize  function receives the new width and height whenever the window size changes.

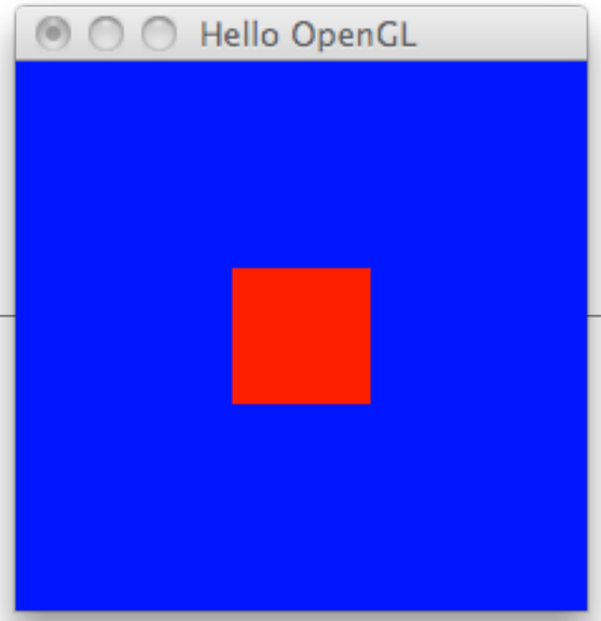
- We can use this information to modify the mapping of our desired coordinate system to real screen coordinates

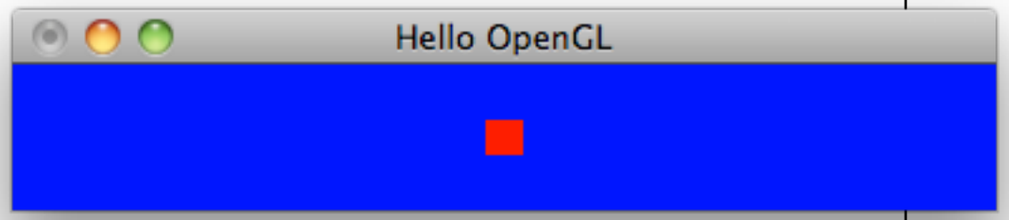
```
int main(int argc, char* argv[])
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
  glutCreateWindow("Hello OpenGL");
  glutDisplayFunc(renderScene);

  glutReshapeFunc(changeSize);

  setupRC();

  glutMainLoop();

  return 0;
}
```

```
void changeSize(int w, int h)
{
 //...
}
```

# changeSize




```
void changeSize(int w, int h)
{
  GLfloat aspectRatio;

  glViewport(0, 0, w, h);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  aspectRatio = (GLfloat) w / (GLfloat) h;
  if (w <= h)
    glOrtho(-100.0, 100.0, -100 / aspectRatio, 100.0 / aspectRatio, 1.0, -1.0);
  else
    glOrtho(-100.0 * aspectRatio, 100.0 * aspectRatio, -100.0, 100.0, 1.0, -1.0);

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

# Defining the Clipping Volume

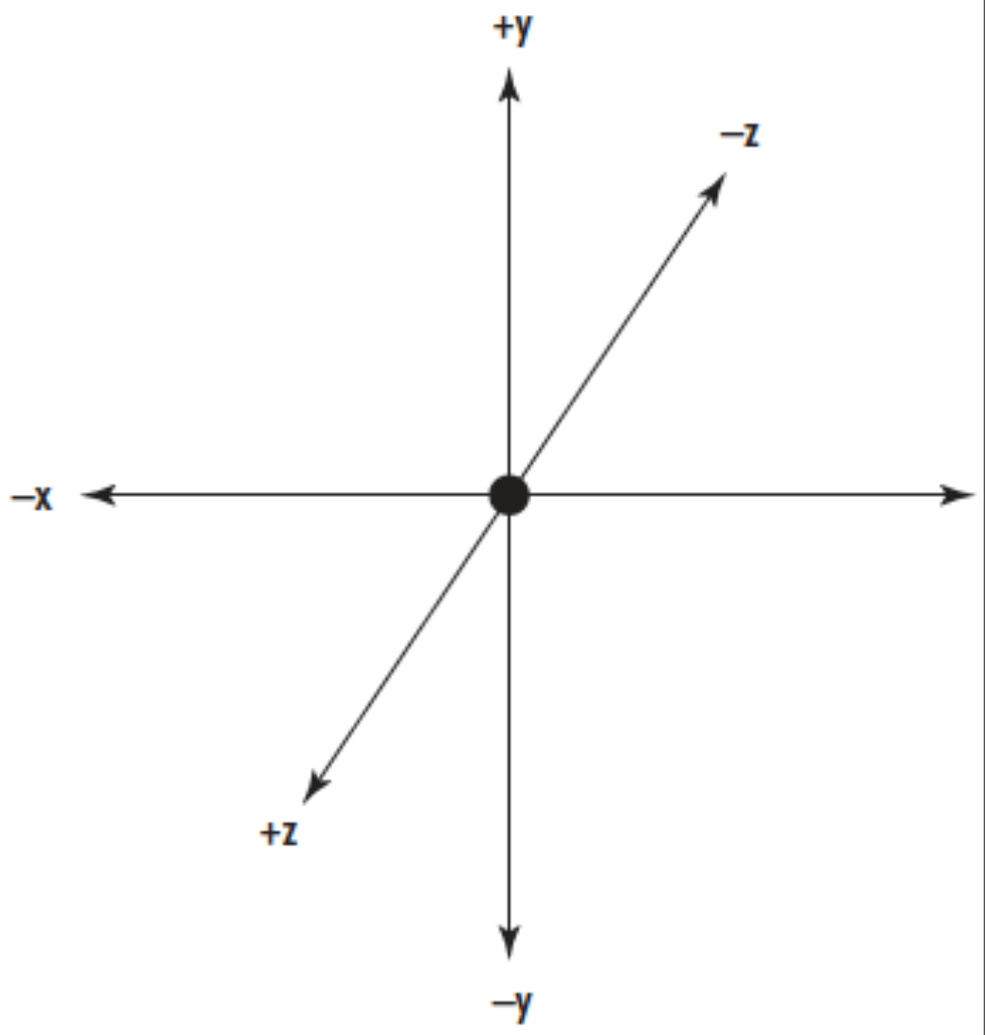
- Define the clipping volume so that the aspect ratio remains square.
- If a viewport that is not square is mapped to a square clipping volume,the image will be distorted.
- E.g, a viewport matching the window size and dimensions but mapped to a square clipping volume would cause images to appear tall and thin in tall and thin windows and wide and short in wide and short windows.

# glOrtho()

```
void changeSize(int w, int h)
{
  //...
  glOrtho(-100.0, 100.0, -100 / aspectRatio, 100.0 / aspectRatio, 1.0, -1.0);
 //..
}
```

```
void glOrtho (GLdouble left,  GLdouble right, GLdouble bottom, GLdouble top,
              GLdouble zNear, GLdouble zFar);
```

- The left and right values specify the minimum and maximum coordinate value displayed along the x-axis; bottom and top are for the y-axis.
- The near and far parameters are for the z-axis, generally with negative values extending away from the viewer
- Many graphics libraries use window coordinates(pixels) for drawing commands.
- OpenGL allows us to use a floating-point (and seemingly arbitrary) coordinate system for rendering.



# Keeping a Square Square - horizontal

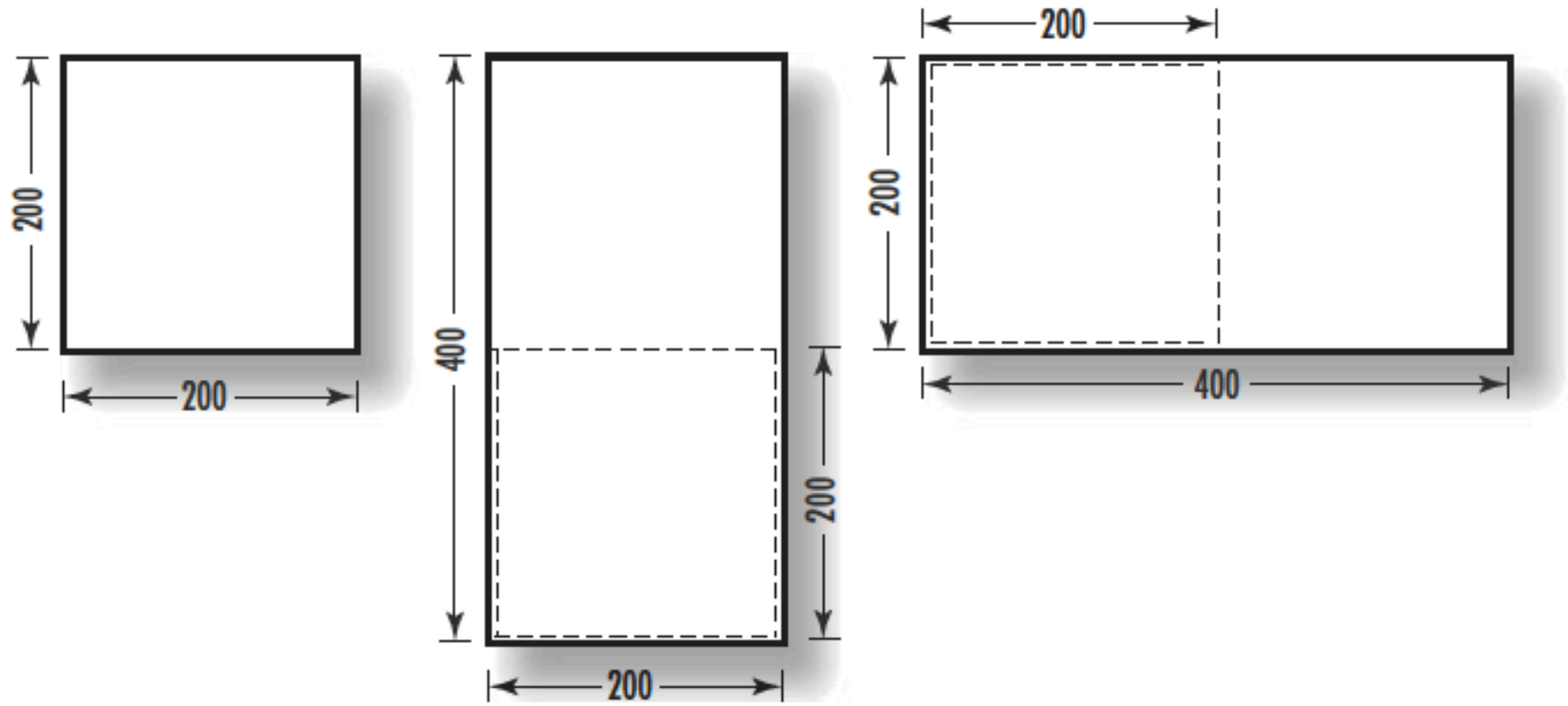
- The clipping volume (visible coordinate space) is modified so that the left side is always at x = –100 and the right side extends to 100

- Unless the window is wider than it is tall, in which case, the horizontal extent is scaled by the aspect ratio of the window.

```
void changeSize(int w, int h)
{
  //...
  aspectRatio = (GLfloat) w / (GLfloat) h;
  if (w <= h)
    glOrtho(-100.0, 100.0, -100 / aspectRatio, 100.0 / aspectRatio, 1.0, -1.0);
  else
    glOrtho(-100.0 * aspectRatio, 100.0 * aspectRatio, -100.0, 100.0, 1.0, -1.0);

  //...
}
```

# Keeping a Square Square - vertical

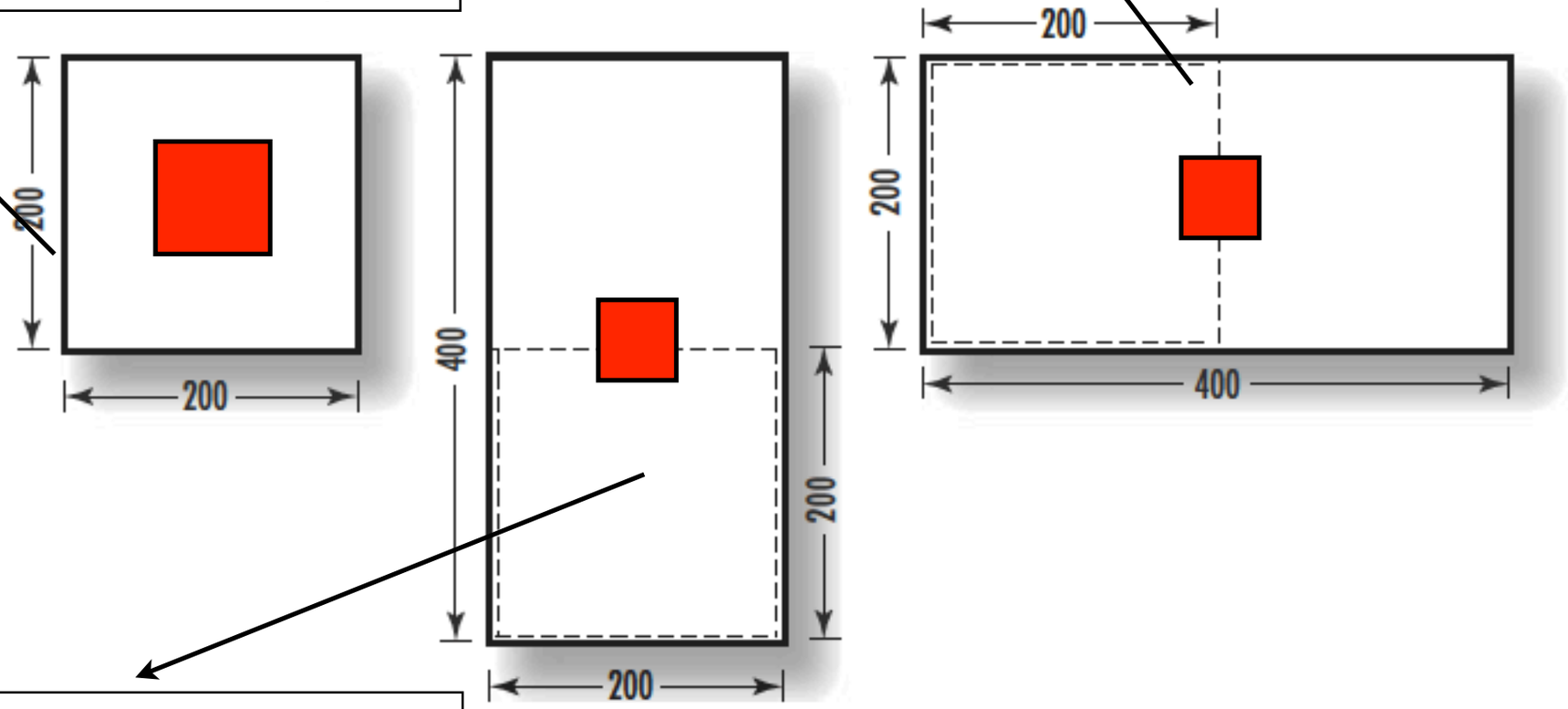
- Similarly, the bottom is always at y = –100 and extends upward to 100 unless the window is taller than it is wide.
- In that case, the upper coordinate is scaled by the inverse of the aspect ratio.
- This serves to keep a square coordinate region 200×200 available (with 0,0 in the center) regardless of the shape of the window



# Example Values

```
aspect ratio:2 :width:400 :height:200
left: -200 right:200 bottom:-100 :top-100:
```

```
aspect ratio:1 :width:200 :height:200
left:-100 :right-100: bottom:-100 top:100
```




```
aspect ratio:0.5 :width:200 :height:400
left:-100 :right-100: bottom:-200 top:200
```

```
  aspectRatio = (GLfloat) w / (GLfloat) h;
  if (w <= h)
    glOrtho(-100.0, 100.0, -100 / aspectRatio, 100.0 / aspectRatio, 1.0, -1.0);
  else
    glOrtho(-100.0 * aspectRatio, 100.0 * aspectRatio, -100.0, 100.0, 1.0, -1.0);
```

# glMatrixMode() & glLoadIdentity()

```
void changeSize(int w, int h)
{
  //...
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

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

- Matrices and the matrix stacks are part of the OpenGL Pipeline

- The projection matrix is the place where we actually define your viewing volume.

- The first call to glLoadIdentity serves to “reset” the coordinate system before any matrix manipulations are performed

- The last two calls indicate that all future transformations will affect what is about to be drawn