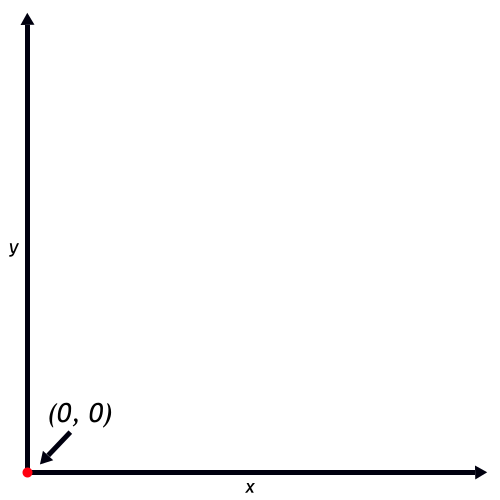
**Computer Graphics and Concepts**

**Carson Foster**

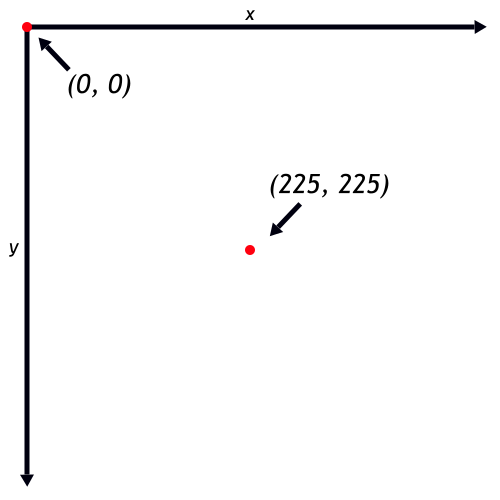
This lesson will cover the basic concepts behind 2D computer graphics. In addition, we’ll look at 2D transformations as well.

Introduction

Essentially, there are two important types of 2D graphics: raster and vector. Essentially, raster graphics deal with pixels, which are essentially small squares that can become different colors, and vector graphics deal with mathematical equations and relationships. We won’t discuss vector graphics much, as we’re mainly concerned with raster graphics. Your monitor uses a raster graphics system, with many pixels on the screen, with each pixel essentially holding a color.

In computer graphics, the most common color model is RGB or RGBA (for red, green, and blue, and sometimes alpha). In the RGB model, every color is described by three numbers, representing the amounts of red, green, and blue light in each color. For RGBA, there is an additional number, which is called the alpha. This alpha value describes the opacity of the color, with low values meaning a color is easy to see through. Each of these numbers is between 0 and 255, inclusive. Therefore, a color value of (255, 0, 0, 255) in the RGBA model means a 100% opaque, 100% red color. Likewise, (140, 0, 120, 255) in the RGBA model means a 100% opaque, reddish-purple color.

In addition to color, each pixel has a location on the screen, in the form of coordinates. Remember the Cartesian coordinate system from math class? It’s pictured to the right, in case you forgot. In this classic system, the positive directions for x and y are to the right and up, respectively. When we only look at positive x’s and y’s, we consider ourselves to be looking at Quadrant I. In the familiar coordinate system, the origin is at the bottom left of Quadrant I.

However, this is not how computers view their pixels. Instead, computer graphics commonly use a different, but similar, Cartesian coordinate system. In this modified system, the positive directions for x and y are to the right and down, respectively. What this means is that in Quadrant I, the origin is now at the top left. In this system, as x increases, it moves farther to the right, like in the normal system. However, as y increases, it moves farther down. This can take some getting used to, so bear this in mind while working with graphics.

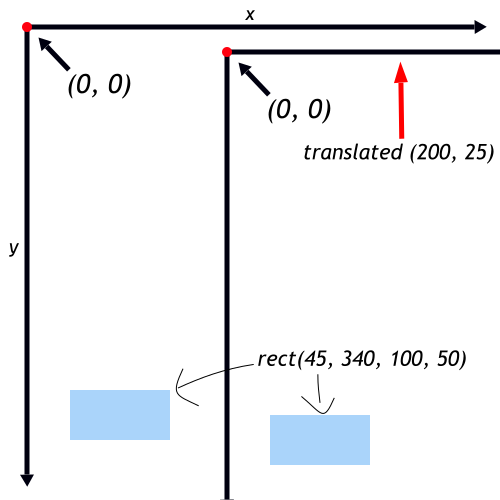
Common Representations

For the purpose of graphics, there are some common conventions and representations it will be handy to remember. The most basic useful shape is the line, which is usually specified by 4 arguments: the starting x-value, the starting y-value, the ending x-value, and the ending y-value. These 4 arguments completely describe a line in 2D space. In addition, the most common representation of a rectangle is also 4 arguments: an x- and a y-coordinate denoting the upper left corner of the rectangle, and then the width and height of that rectangle. Circles are usually three arguments: the coordinates for the center, and then the radius. Angles are almost always measured in radians, in addition.

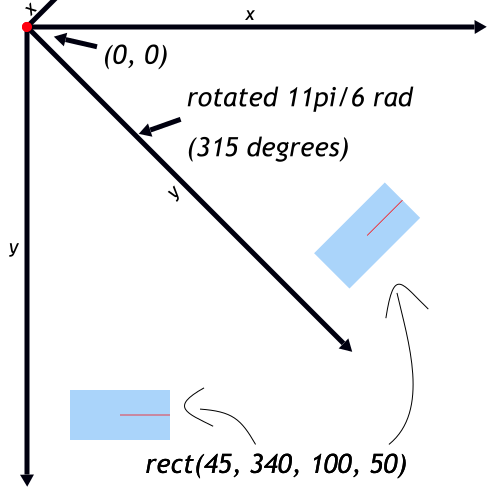
Transformations

Now, we’ll discuss the three most important 2D transformations in computer graphics: translation, rotation, and scaling. Reflection is another 2D transformation, but it isn’t used very often, and so will not be discussed.

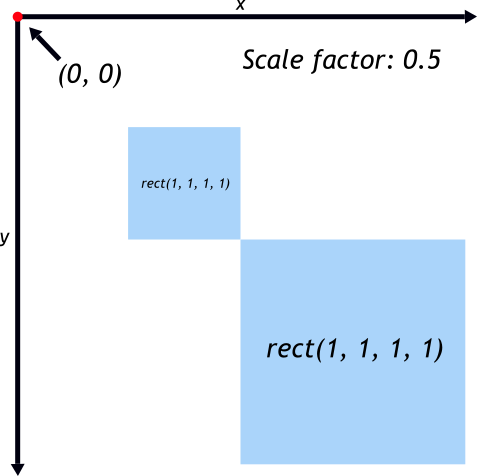
Translation

The most common type of transformation is called translation. Essentially, translation just means move. You take a shape, and then slide in any combination of directions. Any 2D translation can be represented with 2 numbers, with the first corresponding to how far the object moved in the x direction, and the second corresponding to how far the object moved in the y direction. You might remember this from early math classes, but we’ll be talking about a very similar type of translation. In Processing, instead of translating a shape or an object, you actually translate the coordinate grid and the axes. Yes, you heard me correctly: you move the axes. For example, if you translate the axes 200 pixels right and 25 pixels down, and then draw the rectangle (45, 340, 100, 50), that rectangle is going to be drawn at (245, 365) on the old axes (see graphic).

Rotation

The next type of transformation we’ll discuss is called rotation. Rotation, as the name suggests, rotates/spins an object around a point. A 2D rotation can be represented by 3 numbers, which are the x- and y-values of the pivot point and the angle to rotate. Typically, this angle is in radians and describes clockwise rotation from 0 radians (note: this is different from the angular system used in math: rotating something radians (270 degrees) clockwise is the same thing as a radian (90 degrees) rotation in math). Again, in Processing, we do not rotate objects about a point, we rotate the axes. In addition, we always rotate the axes around the same point: the origin. Thus, we can represent a rotation with one number: the angle. See the graphic to the right for an example.

Scaling

The final type of transformation is scaling. This is also called dilation in some contexts. Scaling is a little different from the other two transformations, since it changes the size/area of the object it affects. You can think of it like the handle in the corner of a picture that you drag to resize it. Usually, we represent a scaling transformation with one or two numbers, both either a decimal (called a scale factor) or a percent. With two numbers, the first tells you how much to scale the x-dimension by, and the second tells you how much to scale the y-dimension by. With one number, that just means scale both dimensions by that same amount. Scaling something by 100% in any dimension or by a scale factor of 1.0 does nothing in that dimension. Scaling a rectangle by 50% (a 0.5 scale factor) in the x-dimension, for example, cuts the horizontal length of that rectangle in half. Once more, Processing scales the axes, not objects.