Scaling

Hide tutorial navigation

The final coordinate system transformation is scaling, which changes the size of the grid. Take a look at this program, which draws a square, then scales the grid to twice its normal size, and draws it again.

```
background(255, 255, 255);



stroke(120, 120, 120);

rect(20, 20, 40, 40);



pushMatrix();

scale(2.0);

stroke(0, 0, 0);

rect(20, 20, 40, 40);

popMatrix();
```

First, you can see that the square appears to have moved. It hasn't, of course. Its upper left corner is still at (20, 20) on the scaled-up grid, but that point is now twice as far away from the origin as it was in the original coordinate system.

If you wanted the large square to start at the same corner as the small square, you could translate first, and then scale:

```
background(255, 255, 255);



stroke(120, 120, 120);

fill(0, 0, 255, 100);

rect(20, 20, 40, 40);



pushMatrix();

translate(20, 20);

scale(2.0);

fill(255, 0, 0, 100);

stroke(0, 0, 0);
```

```
rect(0, 0, 40, 40);

popMatrix();
```

You might also notice from both of those programs that the lines are thicker on the large squares. That's no optical illusion—the lines really are twice as thick, because the coordinate system has been scaled to double its size. You could explicitly change the `strokeWeight()` to account for that, or avoid using `scale()` all together.

The `scale()` command optionally accepts two parameters instead of just one, so you can scale the x and y dimensions separately. Try using `scale(3.0, 0.5)` in the above program to make the x dimension three times its normal size and the y dimension only half its normal size.