Transformations

It is Often Nice to Transform Entire Objects at Once

Translation

Rotations and Scaling Happen Around the Origin

Rotation

Scaling

In math, science, and computer programming, angles are not given in degrees, they are given in radians.

1 radian = 0.01745 degrees
1 radian = \pi/180. degrees

But, don't worry about this. Processing gives you a function, radians(), to automatically convert degrees into radians. Use it!
Shearing

There is also a shearY transformation function.

Transformations Accumulate!

```c
void draw()
{
    shear(radians(45.));
    rect(0, 0, 100, 50);
}
```

is the same as:

```c
void draw()
{
    rotate(radians(20.));
}
```

Transformation Order Matters!

```c
void draw()
{
    1. rotate(radians(20.));
    rect(0, 0, 100, 50);
    2. translate(300, 200);
    rect(0, 0, 100, 50);
}
```

You Can Save and Un-do Transformations

```c
void draw()
{
    1. draw( );
    2. translate(200, 300);
    3. draw( );
    4. rotate(radians(45.));
    5. fill(255, 0, 0);
    6. rotate(radians(-45.));
    7. rect(0, 0, 200, 100);
}
```

Transformations and for-loops

```c
void draw()
{
    translate(200, 300);
    for (int i = 0; i < 10; i++)
    {
        rotate(radians(360) / 10);
        rect(0, 0, 100, 100);
    }
}
```

Transformations and for-loops

```c
transform(300, 300);
for (int i = 0; i < 10; i++)
{
    rotate(radians(360) / 10);
    rect(0, 0, 100, 100);
}
```
What's the Difference?

Transformations and for-loops

Rotating While Changing Color and Size

And, there are even 3D Transformations

```
translate(x, y, z);
scale(x, y, z);
rotateX(radians);
rotateY(radians);
rotateZ(radians);
```

But, we will get to those later …