




# Transformations



**Oregon State University**  
Mike Bailey  
mjb@cs.oregonstate.edu

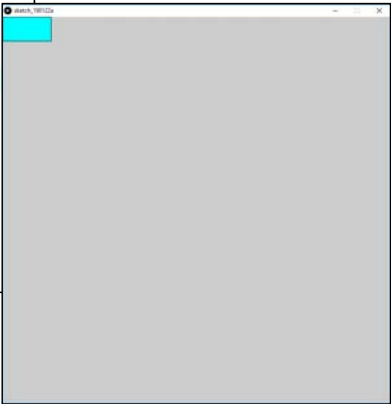

Oregon State University  
Computer Graphics

transformations.pptx mjb - January 28, 2019

# It is Often Nice to Transform Entire Objects at Once

```
void
setup( )
{
  size( 800, 800 );
  stroke( 0, 0, 0 );
  fill( 0, 255, 255 );
}

void
draw( )
{
  rect( 0, 0, 100, 50 );
}
```

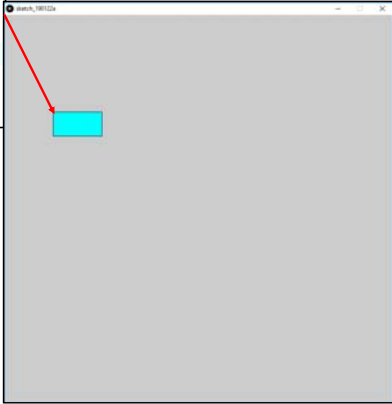




Oregon State University  
Computer Graphics

mjb - January 28, 2019

# Translation

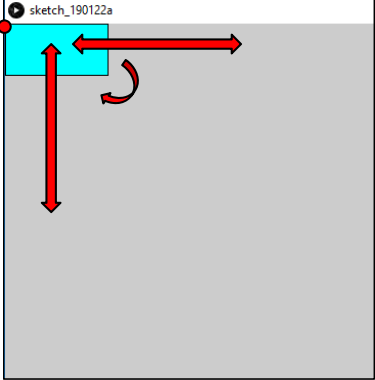

```
void
draw( )
{
  translate( 100, 200 );
  rect( 0, 0, 100, 50 );
}
```

Oregon State University  
Computer Graphics

mjb - January 28, 2019

# Rotations and Scaling Happen Around the Origin

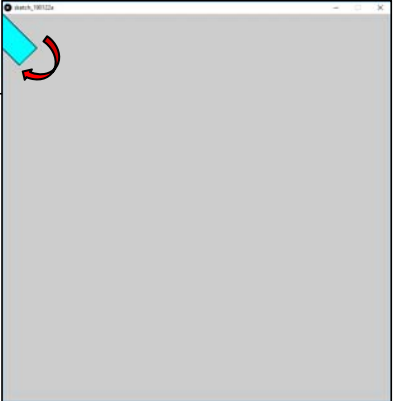



Oregon State University  
Computer Graphics

mjb - January 28, 2019

### Rotation

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



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.

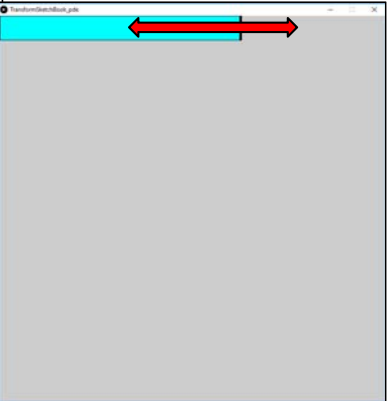
Use it!

Oregon State University  
Computer Graphics

mjb - January 28, 2019

### Scaling

```
void
draw( )
{
  scale( 5., 1. );
  rect( 0, 0, 100, 50 );
}
```

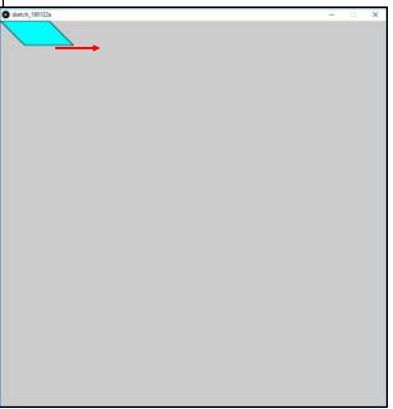


Oregon State University  
Computer Graphics

mjb - January 28, 2019

### Shearing

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



There is also a **shearY** transformation function

Oregon State University  
Computer Graphics

mjb - January 28, 2019

### Transformations Accumulate!

```
void
draw( )
{
  rotate( radians( 10. ) );
  rotate( radians( 10. ) );
  ...
}
```

is the same as:

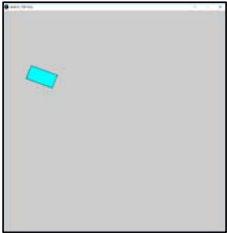
```
void
draw( )
{
  rotate( radians( 20. ) );
  ...
}
```

Oregon State University  
Computer Graphics

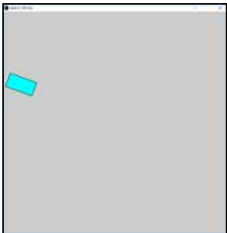
mjb - January 28, 2019

### Transformation Order Matters!

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



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



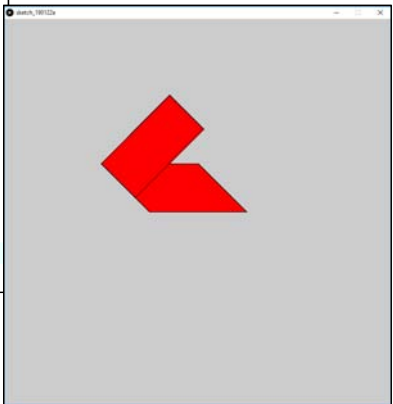
Oregon State University Computer Graphics

mjb - January 28, 2019

### You Can Save and Un-do Transformations

```
void
draw( )
{
  2. translate( 200, 300 );
  pushMatrix();
  1. shearX( radians(45.) );
  rect( 0, 0, 200, 100 );
  popMatrix();

  fill( 255, 0, 0 );
  1. rotate( radians( -45. ) );
  rect( 0, 0, 200, 100 );
}
```

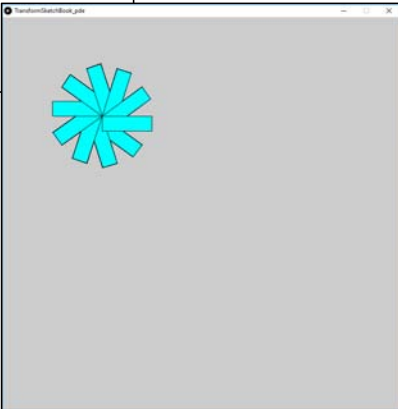


Oregon State University Computer Graphics

mjb - January 28, 2019

### Transformations and for-loops

```
void
draw( )
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360; degrees = degrees + 36 )
  {
    pushMatrix();
    rotate( radians( degrees ) );
    rect( 0, 0, 100, 30 );
    popMatrix();
  }
}
```

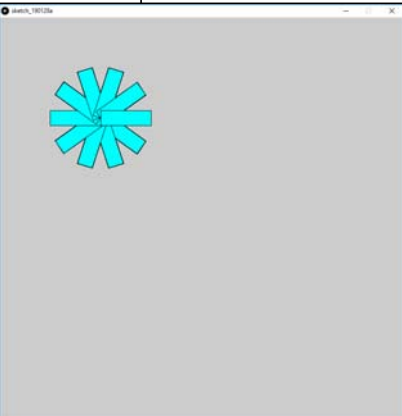


Oregon State University Computer Graphics

2019

### Transformations and for-loops

```
void
draw( )
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360; degrees = degrees + 36 )
  {
    pushMatrix();
    rotate( radians( degrees ) );
    rect( 0, -15, 100, 30 );
    popMatrix();
  }
}
```

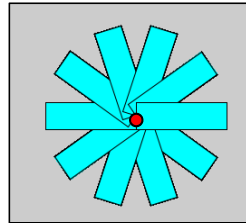
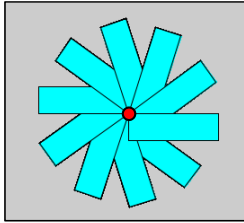


Oregon State University Computer Graphics

19

### What's the Difference?

13



```
void
draw()
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360;
  {
    pushMatrix();
    rotate( radians( degrees ) );
    rect( 0, 0, 100, 30 );
    popMatrix();
  }
}
```

```
void
draw()
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360;
  {
    pushMatrix();
    rotate( radians( degrees ) );
    rect( 0, -15, 100, 30 );
    popMatrix();
  }
}
```

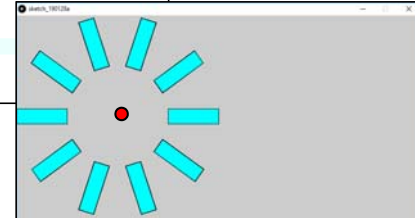
Oregon State  
University  
Computer Graphics

mjb - January 28, 2019

### Transformations and for-loops

14

```
void
draw()
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360; degrees = degrees + 36 )
  {
    pushMatrix();
    rotate( radians( degrees ) );
    rect( 100, -15, 100, 30 );
    popMatrix();
  }
}
```



Oregon State  
University  
Computer Graphics

mjb - January 28, 2019

### Rotating While Changing Color and Size

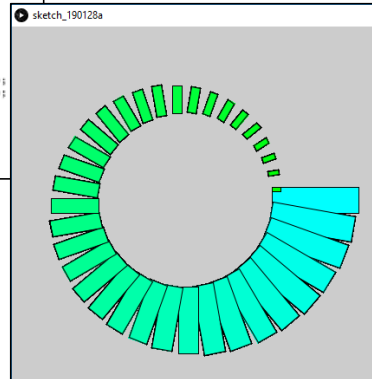
15

```
void
draw()
{
  translate( 200, 200 );
  for( int degrees = 0; degrees <= 360; degrees = degrees + 30 )
  {
    pushMatrix();
    // map color from cyan to green
    int blue = int( map( degrees, 0, 360, 255, 0 ) );
    fill( 0, 255, blue );

    // transform by rotating:
    rotate( radians( degrees ) );

    // change rectangle size:
    int xsz = int( map( degrees, 0, 360, 100, 10 ) );
    int ysz = int( map( degrees, 0, 360, 30, 5 ) );

    // draw rectangle away from the origin:
    rect( 100, -15, xsz, ysz );
    popMatrix();
  }
}
```



Oregon State  
University  
Computer Graphics

mjb - January 28, 2019

### And, there are even 3D Transformations

16

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

But, we will get to those later ...



Oregon State  
University  
Computer Graphics

mjb - January 28, 2019