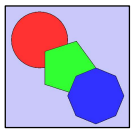
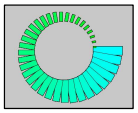
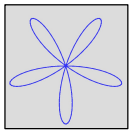
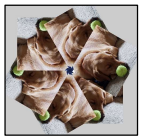
Programming with Processing !

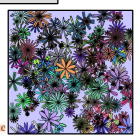
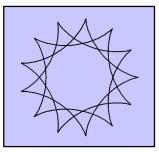
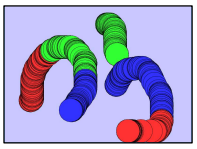
1

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

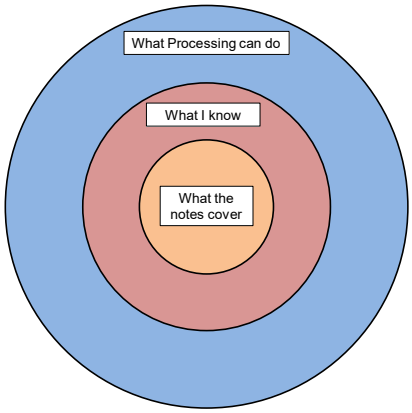
ProcessingStemAcademy.pptx

mjb – July 21, 2023

1

A Warning about the Note Coverage

2



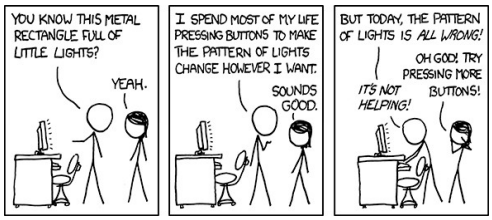
Oregon State University
Computer Graphics

mjb – July 21, 2023

2

Why Do We Have These Notes?

3



http://xkcd.com

Processing has thousands of "buttons" you can press. These notes are here to show you what certain combinations of buttons do in order to learn them the first time, and to remind you later when you've forgotten.

Oregon State University
Computer Graphics

mjb – July 21, 2023

3

Programming Through the Ages

4

Oregon State University

A "program" is a set of instructions that you can store and playback later. This sounds like a computer-thing, but the idea of a "program" has been around for hundreds of years.

The earliest known "program" is (apparently) a mechanical music playback device developed in Baghdad in the 9th century. (https://en.wikipedia.org/wiki/Music_box) You can find a similar device in Oregon today…

Oregon State University
Computer Graphics

mjb – July 21, 2023

4

Music Box Programming

… at the Albany (Oregon) Carousel and Museum

5

Oregon State University
Computer Graphics

mjb – July 21, 2023

5

Music Box Programming

6

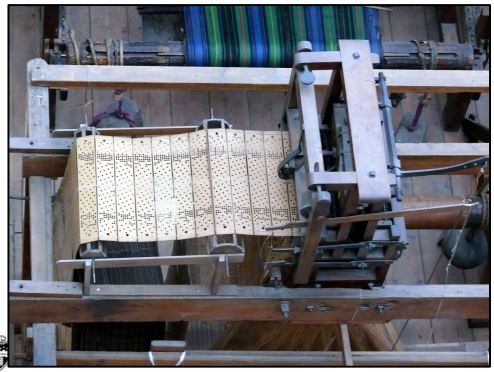
Albany Carousel and Museum

Oregon State University
Computer Graphics

mjb – July 21, 2023

6

Another Historic Example is Textile Programming



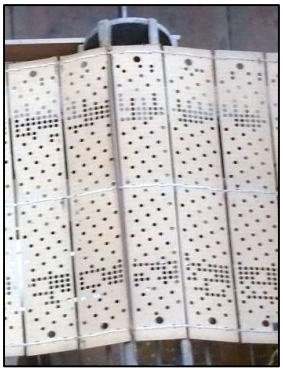
Jacquard Loom, circa 1804

Oregon State University Computer Graphics

mjb – July 21, 2023

7

Textile Programming



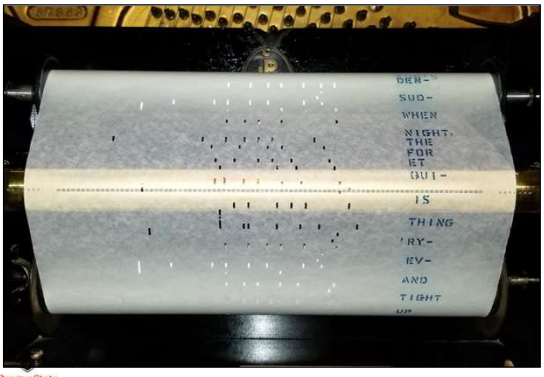
Jacquard Loom, circa 1804

Oregon State University Computer Graphics

mjb – July 21, 2023

8

And, of course, there is the Ever-fun Player Piano



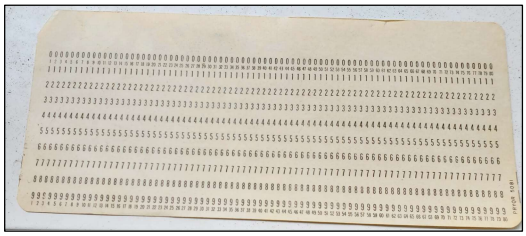
https://en.wikipedia.org/wiki/Piano_roll

Oregon State University Computer Graphics

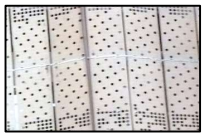
mjb – July 21, 2023

9

Computers Eventually Imitated Historic Methods using Punch Cards



circa 1972

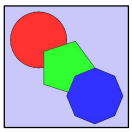


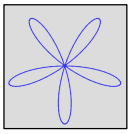
Oregon State University Computer Graphics

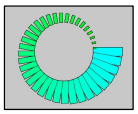
mjb – July 21, 2023

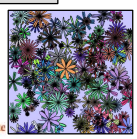
10

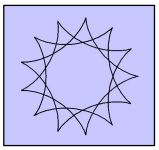
The *Processing* Programming Language

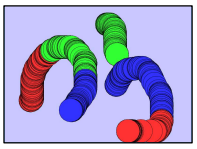












Oregon State University Computer Graphics

mjb – July 21, 2023

11

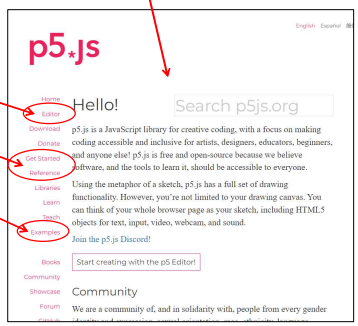
Where to Find *Processing*

In your favorite web browser, go to: https://p5js.org/
Here's what you will see:

Go here to start using *Processing*

These are good links to check out!

Processing includes a collection of *spectacular* example programs



Oregon State University Computer Graphics

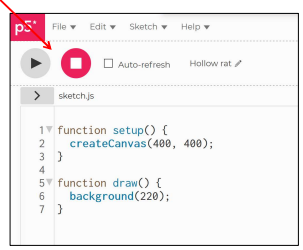
mjb – July 21, 2023

12

Running *Processing*

13

Click on the **Editor** link, or navigate to: https://editor.p5js.org/
Either way, here's what you will see:



Oregon State University Computer Graphics

mjb – July 21, 2023

13

Running *Processing*

14

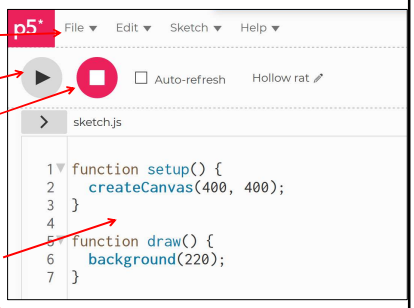
Menu headers

Run your program

Stop the program

Program-writing/editing area

Processing message area



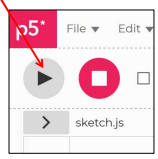
Oregon State University Computer Graphics

mjb – July 21, 2023

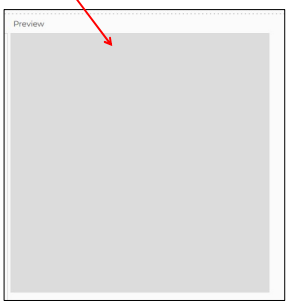
14

15

Now, click this button!



Here's what you will get



Don't worry – it will get better ☺

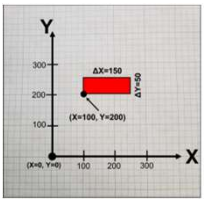
Oregon State University Computer Graphics

mjb – July 21, 2023

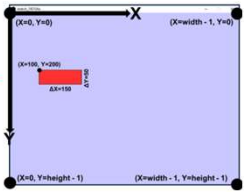
15

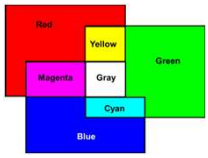
Writing *Processing* Programs

16



Oregon State University





Oregon State University Computer Graphics

mjb – July 21, 2023

16

Introduction to Writing *Processing* Programs

17

With *Processing*, you get to do real-world programming that gives you visual output. You get to make cool pictures at the same time you are learning to program. This opens up a world of opportunities for you!

Oregon State University Computer Graphics

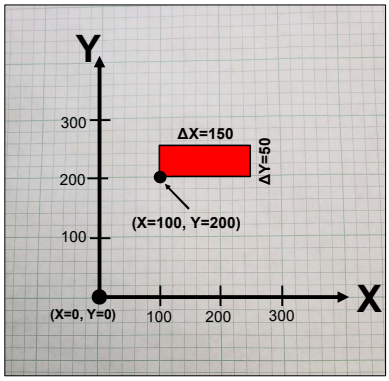
mjb – July 21, 2023

17

First, Remember How Graph Paper Works

18

The Greek letter delta, **Δ**, is the mathematics symbol for "the change in"



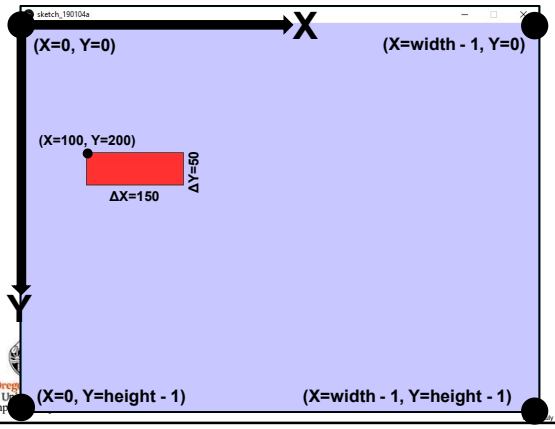
Oregon State University Computer Graphics

mjb – July 21, 2023

18

This is the "Graph Paper" System for *Processing* Programs

19



19

Colors for Computer Graphics Monitors: Additive Colors (RGB)

20

Colors are formed with combinations of **red**, **green**, and **blue**.

The smallest number you can use for each is **0**
The largest number you can use for each is **255**

	Red	Green	Blue
Black	0	0	0
White	255	255	255
Red	255	0	0
Orange	255	128	0
Yellow	255	255	0
Green	0	255	0
Cyan	0	255	255
Blue	0	0	255
Magenta	255	0	255

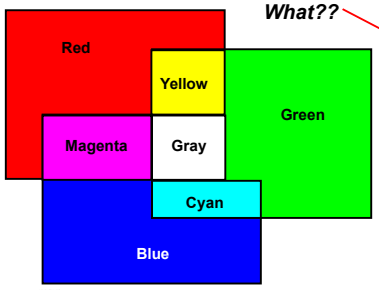
Oregon State University Computer Graphics

mjb – July 21, 2023

20

Colors for Computer Graphics Monitors: Additive Colors (RGB)

21



What??

Cyan = Green + Blue
Magenta = Red + Blue
Yellow = Red + Green
Gray = Red + Green + Blue

	Red	Green	Blue
Black	0	0	0
White	255	255	255
Red	255	0	0
Orange	255	128	0
Yellow	255	255	0
Green	0	255	0
Cyan	0	255	255
Blue	0	0	255
Magenta	255	0	255

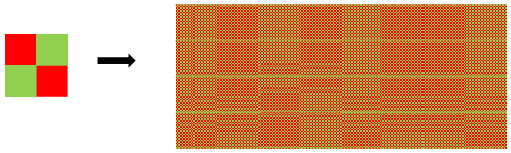
Oregon State University Computer Graphics

mjb – July 21, 2023

21

Yes, Our Vision System Really Does Mush Red and Green Together to Make Yellow!

22



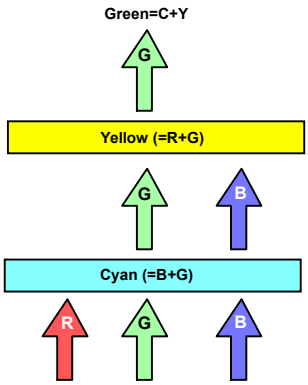
Oregon State University Computer Graphics

mjb – July 21, 2023

22

Colors for Paints, Toners, and Clear Plastic: Subtractive Colors (CMYK)

23



R = Red
G = Green
B = Blue
W = White

C = Cyan
M = Magenta
Y = Yellow
K = Black

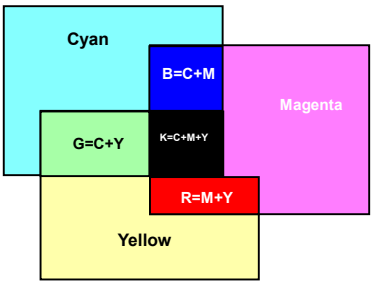
Oregon State University Computer Graphics

mjb – July 21, 2023

23

Subtractive Colors (CMYK)

24



R = Red
G = Green
B = Blue
W = White

C = Cyan
M = Magenta
Y = Yellow
K = Black

Oregon State University Computer Graphics

mjb – July 21, 2023

24

Writing a *Processing* Program – Try This!

25

```
function
setup( )
{
  createCanvas( 800, 600 );
  background( 200, 200, 255 );
  stroke( 0, 0, 0 );
  fill( 255, 50, 50 );}

function
draw( )
{
  rect( 100,  200,   150,  50 );
}
```

You don't need to type all this in – you can copy-and-paste it from these notes!

You must add code to the **setup( )** function. Processing calls this ***once*** when your program starts.

You must add code to the **draw( )** function. Processing calls this ***every time*** it wants to re-draw the scene.

Oregon State University Computer Graphics

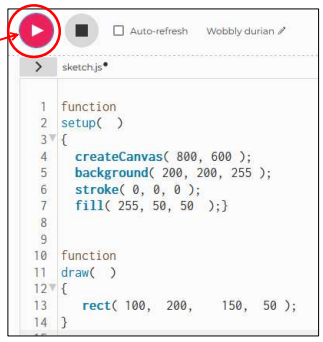
mjb – July 21, 2023

25

Running Your Processing Programs

26

Click here to run your program



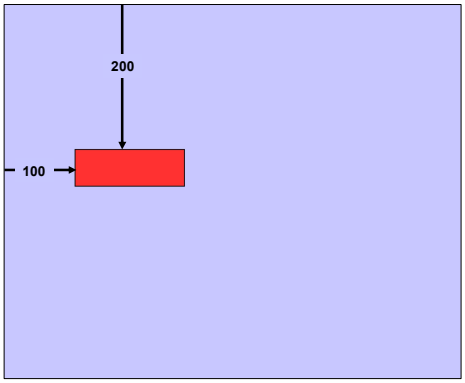
Oregon State University Computer Graphics

mjb – July 21, 2023

26

Enjoying the Output of Your First Processing Program

27



Don't worry – it will get better ☺

Oregon State University Computer Graphics

mjb – July 21, 2023

27

Some Functions to use when Writing Processing Programs

28

Color	color(r, g, b)	Set the currnt color to (r, g, b)
Drawing	fill(c)	Fill using the color c
Drawing	noFill( )	Don't do any filling
Drawing	noStroke( )	Don't do any outlining
Drawing	stroke(c)	Outline using the color c
Drawing	strokeWeight(w)	Thickness of the outline
Math	abs(f)	Absolute value
Math	lerp(v1, v2, blend)	Linearly interpolate two values
Math	map(input, lowin, highin, lowout, highout)	Lineary map the input variable from the range [lowin,highin] to [lowout,highout]
Math	max(f1, f2)	Maximum of the two numbers
Math	min(f1, f2)	Minimum of the two numbers
Printing	print(s)	Print the string into the console
Randomness	random(low, high)	Return a random number between low and high
Setup	background(r, g, b)	Set the background to r, g, b
Setup	createCanvas(w, h)	Set the size of the graphics window to w x h pixels
Setup	draw( )	The function that gets called over and over to draw your scene
Setup	setup( )	The function that gets called when your program starts
Shapes	beginShape( )	Start drawing to vertices
Shapes	ellipse(cx, cy, w, h)	Draw an ellipse
Shapes	endShape( )	Finish drawing to vertices
Shapes	line(x0, y0, x1, y1)	Draw a line
Shapes	point(x, y)	Put a dot at (x,y)
Shapes	quad(x0, y0, x1, y1, x2, y2, x3, y3)	Draw a quadrilateral
Shapes	rect(ulx, uly, w, h)	Draw a rectangle
Shapes	triangle(x0, y0, x1, y1, x2, y2)	Draw a triangle
Shapes	vertex(x, y)	(in between calls to beginShape and endShape)
Text	text(s, x, y)	Draw the text "s" on the screen at (x,y) with the current fill color
Variables	height	Screen height in pixels
Variables	PI	π
Variables	width	Screen width in pixels

Oregon State University Computer Graphics

mjb – July 21, 2023

28

Variables

29

Oregon State University

```
function
draw( )
{
  let x = 100;
  let y = 2*x;
  rect( x,  y,   150,  50 );
}
```

Arithmetic operations in programming are:

+	**Addition**
-	**Subtraction**
*****	**Multiplication**
/	**Division**
( )	**Grouping**

Oregon State University Computer Graphics

mjb – July 21, 2023

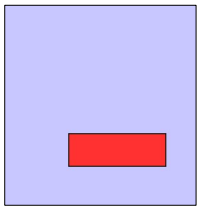
29

Variables – using symbols instead of just numbers

30

Variables are the process of replacing numbers with symbols in order to generalize a computation to work in more than one situation

```
function
draw( )
{
  let x = 100;
  let y = 200;
  rect( x,  y,   150,  50 );
}
```



"let" just says that you are defining a variable

Oregon State University Computer Graphics

mjb – July 21, 2023

30

31

Variables – using symbols instead of just numbers

We can use variables to create ***relationships***

```
function
draw( )
{
  let x = 100;
  let y = 2*x;
  rect( x,  y,   150,  50 );
}
```

When you assign a number to **x**, then **y** will automatically be twice as big as whatever you set x to be.

Arithmetic operations in programming are:

+	**Addition**
-	**Subtraction**
*****	**Multiplication**
/	**Division**
( )	**Grouping**

Oregon State University Computer Graphics

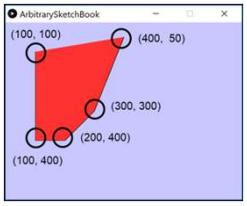
mjb – July 21, 2023

31

32

Drawing Lines and Polygons

Oregon State University



Oregon State University Computer Graphics

mjb – July 21, 2023

32

33

Rectangles are Fun, but Arbitrary Lines and Polygons are Funner

Easy – just list the coordinates:

```
beginShape( );

  vertex( x0, y0 );
  vertex( x1, y1 );
  vertex( x2, y2 );
     . . .

endShape( );
```

Oregon State University Computer Graphics

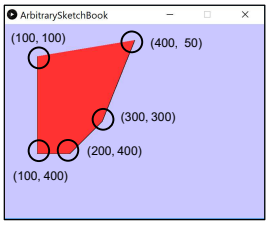
mjb – July 21, 2023

33

34

Rectangles are Fun, but Arbitrary Lines and Polygons are Funner

```
function
draw( )
{
  beginShape( );

    vertex( 100, 100 );
    vertex( 100, 400) ;
    vertex( 200, 400 );
    vertex( 300, 300 );
    vertex( 400,   50 );

  endShape( );
}
```



Oregon State University Computer Graphics

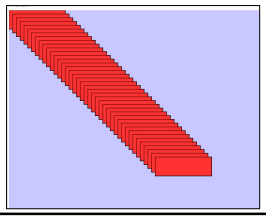
mjb – July 21, 2023

34

35

for-loops

Oregon State University



Oregon State University Computer Graphics

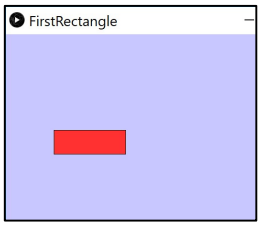
mjb – July 21, 2023

35

36

Drawing One Rectangle is Pretty Straightforward

```
rect( 100,  200,    150,  50 );
```



Oregon State University Computer Graphics

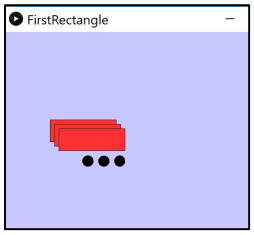
mjb – July 21, 2023

36

But, This Gets Awfully Boring if You Want to Draw 100 Rectangles!

37

```
rect( 100,  200,    150,  50 );
rect( 110,  210,    150,  50 );
rect( 120,  220,    150,  50 );
●●●
```



Oregon State University Computer Graphics

mjb – July 21, 2023

37

for-loops to the Rescue!

38

```
function
draw( )
{
  for( let x = 0 ; x < 400 ; x = x + 10 )
  {
    let y = x;
    rect( x, y,   150,  50 );
  }
}
```

Repeating a code pattern is a common theme in programming.

This line is called a "for-loop". It is very handy for repeating patterns of code.

The for-loop executes the commands **in the curly braces** a bunch of times. Using it looks like this:

1. Do this equation once at the start

2. Keep looping as long as this test is true

3. Do this at the end of one loop, but before the start of the next one

```
for( let x = 0 ; x < 400 ; x = x + 10 )
```

Yes, the semi-colons (;) are necessary!

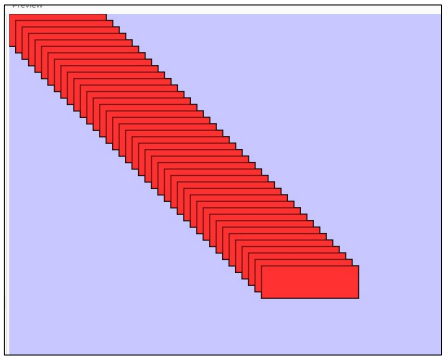
Oregon State University Computer Graphics

mjb – July 21, 2023

38

for-loops to the Rescue!

39



Oregon State University Computer Graphics

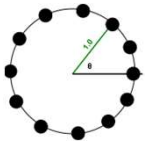
mjb – July 21, 2023

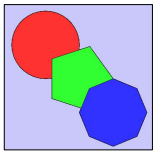
39

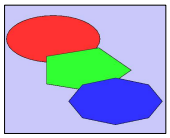
Drawing Circles and Other Regular Polygons, I

40

Oregon State University







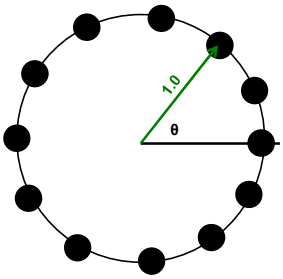
Oregon State University Computer Graphics

mjb – July 21, 2023

40

First, We Need to Understand Something about Angles

41



If a circle has a radius of 1.0, then we can march around it by simply changing the angle that we call **θ.**

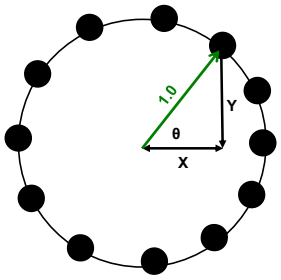
Oregon State University Computer Graphics

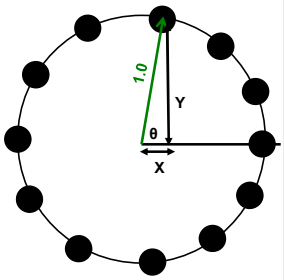
mjb – July 21, 2023

41

First, We Need to Understand Something about Angles

42





One of the things we notice is that each angle **θ** has a *unique **X** and **Y** that goes with it.*

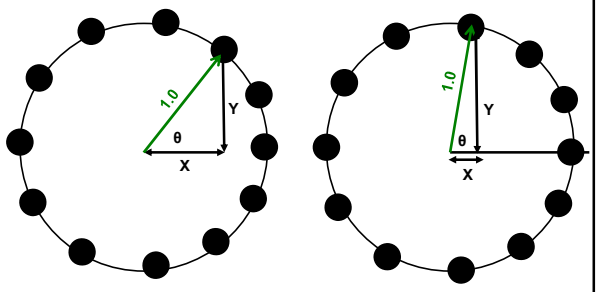
The X and Y are different for each θ.

Oregon State University Computer Graphics

mjb – July 21, 2023

42

First, We Need to Understand Something about Angles



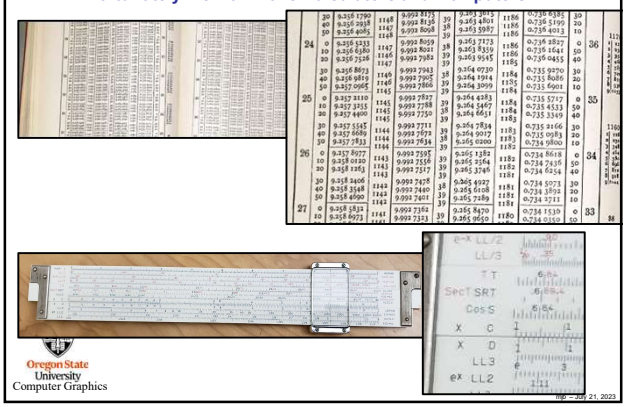
Centuries ago, people developed tables of those X and Y values as functions of θ.

They called the X values ***cosines*** and the Y values ***sines***. These are abbreviated cos and sin.

$\cos\theta = X$
$\sin\theta = Y$

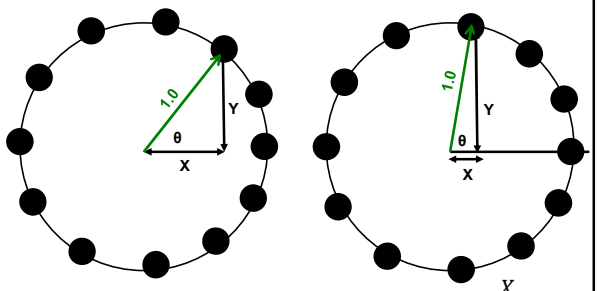
43

In Earlier Times, People Looked up Sines and Cosines in Books and on Slide Rules – Fortunately We Now Have Calculators and Computers



44

Cosines and Sines are Really Ratios



If we were to double the radius of the circle, all of the X's and Y's would also double.

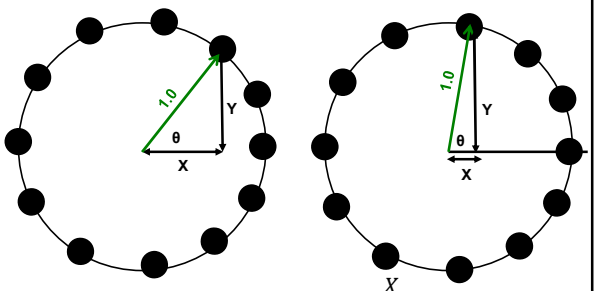
So, really the cos and sin are *ratios* of **X** and **Y** to the circle Radius

$$\cos\theta = \frac{X}{R}$$

$$\sin\theta = \frac{Y}{R}$$

45

Cosines and Sines are Really Ratios



So, if we know the circle Radius, and we march through a series of θ angles, we can determine all of the X's and Y's that we need to draw a circle.

$$\cos\theta = \frac{X}{R} \Rightarrow X = R * \cos\theta$$

$$\sin\theta = \frac{Y}{R} \qquad Y = R * \sin\theta$$

46

Processing Doesn't Include Regular Polygon-Drawing Functions, So We Add Our Own to the End of the Program

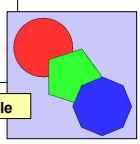

```
function
Shape( xc, yc, r, numsegs )
{
  let dang = (2.*PI) / float( numsegs );
  let ang = 0.;
  beginShape( );

  for( let i = 0; i <= numsegs; i = i + 1 )
  {
    let x = xc + r * cos(ang);
    let y = yc + r * sin(ang);
    vertex( x, y );
    ang = ang + dang;
  }

  endShape( );
}
```

numsegs is the number of line segments making up the circumference of the circle.

numsegs=36 gives a nice circle.

5 gives a pentagon.
8 gives an octagon.
4 gives you a square. Etc.

Why **2.*PI (= 2π)**?



dang **= Δang = change in *ang* as we go around the circle**

47

Why 2.*PI (= 2π)?


```
let dang = (2.*PI) / float( numsegs );
```

We commonly measure angles in **degrees**, but scientists, engineers, and computers like to measure angles in something else called **radians**.

There are 360° (degrees) in a complete circle.
There are 2π (~6.28) radians in a complete circle.

The built-in cos( ) and sin( ) functions expect angles to be given in radians.

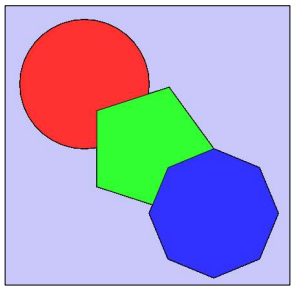
Processing has built-in functions to convert between the two:

```
let rad = radians( deg );
let deg = degrees( rad );
```


48

Circles, Pentagons, and Octagons -- oh my!

```
function
draw( )
{
  fill( 255,   50,   50 );
  Shape( 200, 200, 100, 36 );

  fill(   50, 255,   50 );
  Shape( 300, 300, 100,  5 );

  fill(   50,   50, 255 );
  Shape( 400, 400, 100,  8 );
}
```



Oregon State University
Computer Graphics

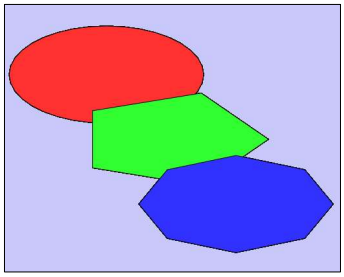
And, there is no reason the X and Y radii need to be the same…

```
function
Shape2( xc, yc, rx, ry, numsegs )
{
  let dang = (2.*PI) / float( numsegs );
  let ang = 0.;
  beginShape( );

  for( let i = 0; i <= numsegs; i = i + 1 )
  {
    let x = xc + rx * cos(ang);
    let y = yc + ry * sin(ang);
    vertex( x, y );
    ang = ang + dang;
  }

  endShape( );
}
```

Oregon State University
Computer Graphics

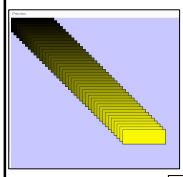
And, there is no reason the X and Y radii need to be the same…

```
function
draw( )
{
  fill( 255,   50,   50 );
  Shape2( 200, 200, 150, 75, 36 );

  fill(   50, 255,   50 );
  Shape2( 300, 300, 150, 75,  5 );

  fill(   50,   50, 255 );
  Shape2( 400, 400, 150, 75,  8 );
}
```

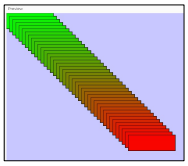


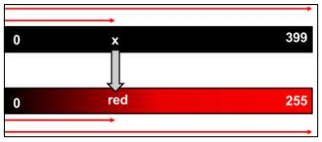
Oregon State University
Computer Graphics

The Processing *map( )* Function



Oregon State University





Oregon State University
Computer Graphics

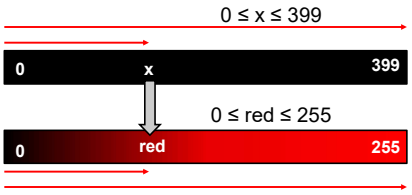
More Sophisticated Relationships:

The *map( )* function

This function takes an input value, the range of values it lives between, and the range of output values. It returns the output value that corresponds to the input value.

So, for example, if we wanted to turn an x value into a red color, we might say:

let red = int(map(x, 0, 399, 0, 255));

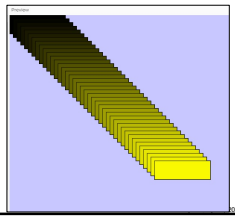


Oregon State University
Computer Graphics

More Sophisticated Relationships:

The *map( )* function

```
function
draw( )
{
  for( let x = 0 ; x < 400 ; x = x + 10 )
  {
    let y = x;                          Input Range     Output Range
    let red =   int( map( x, 0, 399,    0, 255 ) );
    let green = int( map( y, 0, 399,    0, 255 ) );
    fill( red, green, 0 );
    rect( x, y,   150,  50 );
  }
}
```



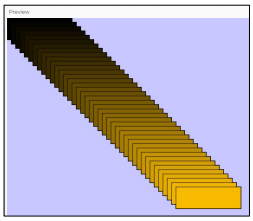
Oregon State University
Computer Graphics

More Sophisticated Relationships:

The *map( )* function

55

```
function
draw( )
{
  for( let x = 0 ; x < 400 ; x = x + 10 )
  {
    let y = x;
    let red =     int(  map( x, 0, 399,     0, 255  )  );
    let green = int(  map( y, 0, 399,     0, 255  )  );
    green = 3 * green / 4;
    fill( red, green, 0 );
    rect( x, y,   150,  50 );
  }
}
```



Oregon State University Computer Graphics

mjb – July 21, 2023

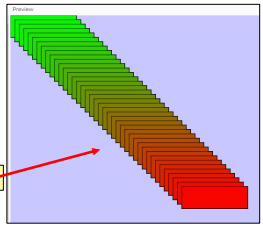
55

The *map( )* function can also do blending

56

```
function
draw( )
{
  for( let x = 0 ; x < 400 ; x = x + 10 )
  {
    let y = x;
    let red =     int(  map( x, 0, 399,     0, 255  )  );
    let green = int(  map( y, 0, 399,   255,    0  )  );
    fill( red, green, 0 );
    rect( x, y,   150,  50 );
  }
}
```

Interpolate one forward and the other one backwards



All-Green morphs into **All-Red**

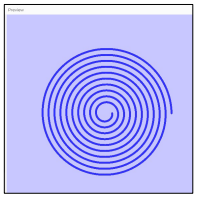
Oregon State University Computer Graphics

mjb – July 21, 2023

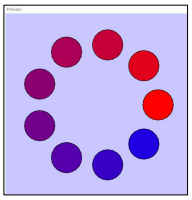
56

57

Drawing Circles and Other Regular Polygons, II



Oregon State University



Oregon State University Computer Graphics

mjb – July 21, 2023

57

58

There is also no reason we can't gradually change the radius …

```
function
Spiral( xc, yc, r1, r2, numsegs, numturns )
{
  let dang = numturns * (2.*PI) / float( numsegs );
  let ang = 0.;
  beginShape( );

  for( let i = 0; i <= numsegs; i = i + 1 )
  {
  let newrad = map( i,   0, numsegs,  r1, r2 );
    let x = xc + newrad * cos(ang);
    let y = yc + newrad * sin(ang);
    vertex( x, y );
    ang = ang + dang;
  }

  endShape( );
}
```

Oregon State University Computer Graphics

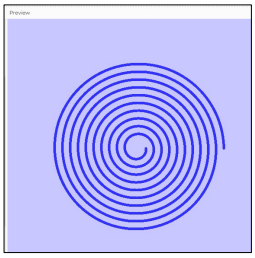
mjb – July 21, 2023

58

59

There is also no reason we can't gradually change the radius …

```
function
draw( )
{
  strokeWeight( 5 );
  noFill( );
  Spiral( 300, 300, 20, 200, 1000, 10 );
}
```



Oregon State University Computer Graphics

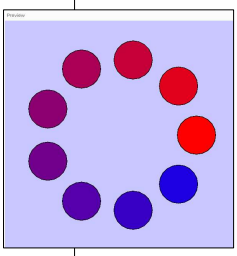
mjb – July 21, 2023

59

60

We Can Also Use This Same Idea to Arrange Things in a Circle

```
function
draw( )
{
  let numobjects = 10;
  let radius = 200.;
  let xc = 300;
  let yc = 300;
  let numsegs = 20;
  let r = 50;
  let dang = (2.*PI) / float( numobjects - 1 );
  let ang = 0.;
  for( let i = 0; i < numobjects; i = i + 1 )
  {
    let x = xc + radius * cos(ang);
    let y = yc + radius * sin(ang);
    let red   = int( map( i,   0, numobjects - 1,     0, 255 ) );
    let blue = int( map( i,   0, numobjects - 1, 255,     0 ) );
    fill( red, 0, blue );
    Shape( x, y, r, numsegs );
    ang = ang + dang;
  }
}
```



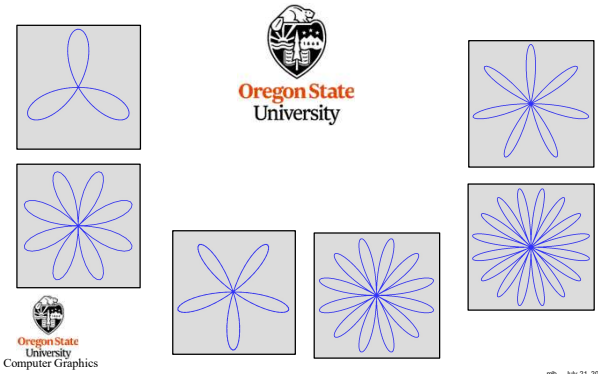
Oregon State University Computer Graphics

mjb – July 21, 2023

60

61

Polar Equations



Oregon State University
Computer Graphics

mjb – July 21, 2023

61

62

Setting the radius as a function of the angle

```
function
Polar(xc,  yc, factor, numsegs,  numturns )
{
  let dang = numturns * (2.*PI) / float( numsegs );
  let theta = 0.;
  beginShape( );

  for( let i = 0; i <= numsegs; i = i + 1 )
  {
    let r = 200. * sin(factor*theta);
    let x = xc + r * cos(theta);
    let y = yc + r * sin(theta);
    vertex( x, y );
    theta = theta + dang;
  }

  endShape( );
}
```

200 is the radius of the circle the shape fits in

sin(factor*theta) changes that radius by making it grow bigger and smaller

Oregon State University
Computer Graphics

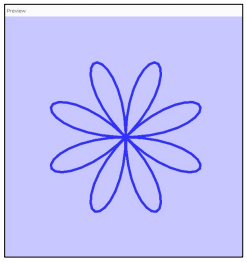
mjb – July 21, 2023

62

63

Setting the radius as a function of the angle

```
function
draw( )
{
  stroke(  50, 50, 255 );
  strokeWeight( 5 );
  noFill( );
  Polar( 300, 300,  4, 1000, 8 );
}
```



It's a lot of fun to experiment with different values for the ***factor*** variable!

Oregon State University
Computer Graphics

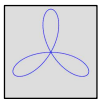
mjb – July 21, 2023

63

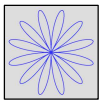
64

It's a lot of fun to experiment with different values for the *factor* variable!

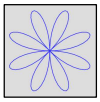
factor = 3



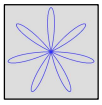
factor = 6



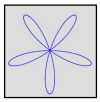
factor = 4



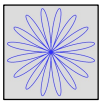
factor = 7



factor = 5



factor = 8



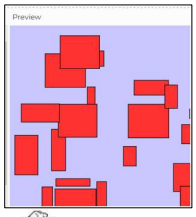
Oregon State University
Computer Graphics

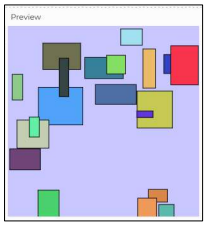
mjb – July 21, 2023

64

65

Randomness





Oregon State University
Computer Graphics

mjb – July 21, 2023

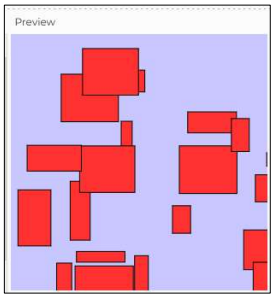
65

66

Randomness

The Processing function ***random( )*** takes in two numbers and returns a random number between them. Here it is being used to randomly position and size shapes:

```
function
setup( )
{
  createCanvas( 300, 300 );
  background( 200, 200, 255 );
  stroke(  0, 0, 0 );
  fill( 255,  50,  50 );
  noLoop( );
}

function
draw( )
{
  for( let i = 0 ; i < 20 ; i = i + 1 )
  {
    let x = random( 0, 300 );
    let y = random( 0, 300 );
    let sizex = random( 10, 70 );
    let sizey = random( 10, 70 );
    rect( x, y,  sizex, sizey );
  }
}
```



Com

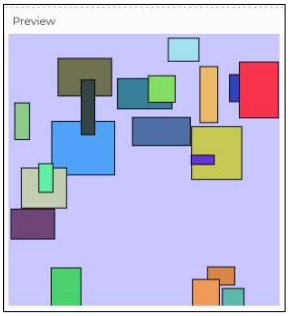
mjb – July 21, 2023

66

Randomness

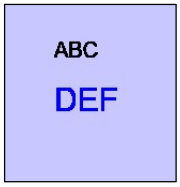
Or, also use it to pick colors:

```
function
draw( )
{
  for( let i = 0 ; i < 20 ; i = i + 1 )
  {
    let x = random( 0, 300 );
    let y = random( 0, 300 );
    let sizex = random( 10, 70 );
    let sizey = random( 10, 70 );
    let r  = random(  50, 255 );
    let g = random(   50, 255 );
    let b = random(   50, 255 );
    fill( r, g, b );
    rect( x, y,  sizex, sizey );
  }
}
```



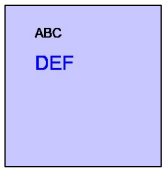
Drawing Text

Oregon State University



Setting the size and drawing the text

```
function
setup( )
{
  createCanvas( 400, 400 );
  background( 200, 200, 255 );
}

function
draw( )
{
  fill( 0, 0, 0 );
  textSize( 20 );
  text( "ABC", 50, 50 );

  fill( 0, 0, 255 );
  textSize( 30 );
  text( "DEF", 50, 100 );
}
```



Use fill( ) to set the text color

Text height in pixels

Where (x,y) to draw the text

Text to draw

Saving Your Processing Program and Getting It Back Later

Oregon State University

Processing Doesn't Save to Your Local Machine

It saves to the cloud. But it only does it if you have an account.

Fortunately, AWSEM / STEM Academy already has one.
So, go to the upper-right corner of your Processing window and click on **Log in**.
Then enter:

Username: **awsem**
Password: **corvallis72542**

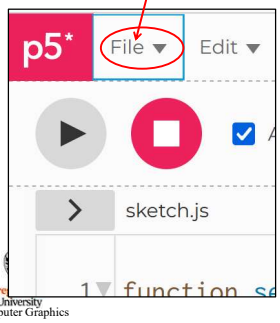
You can create your own account if you want, *but only do it with your parents' help.*

Saving Your Processing Program and Getting It Back Later

The next trick is to click here and change the goofy name it gave your program to something more sensible, preferably something with your name in it and maybe something about what you were working on.

Saving Your Processing Program and Getting It Back Later

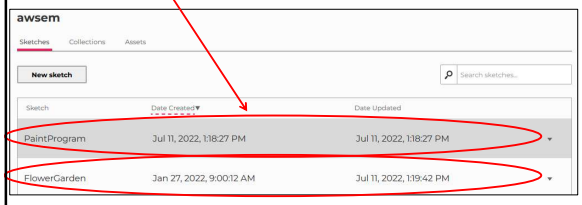
Then click **File** → **Save**



73

Getting Your Processing Programs Back Later

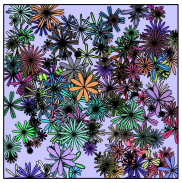
To bring back programs, click **File** → **Open**, look at the list of program names there, then click on the one you want to bring back

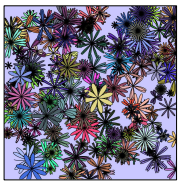


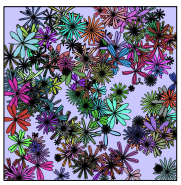
The two I put there are *FlowerGarden* and *PaintProgram.*

74

The Flower Garden







75

The Flower Garden

Be sure the call to **noLoop( )** is included in setup( ) !

```
function setup( )
{
  createCanvas( 600, 600 );
  background( 200, 200, 255 );
  stroke( 0, 0, 0 );
  noLoop( );
}

function draw( )
{
  for( let i = 0 ; i < 200 ; i = i + 1 )
  {
    let r = random( 50, 255 );
    let g = random( 50, 255 );
    let b = random( 50, 255 );
    let xc = random( 0, width );
    let yc = random( 0, height );
    let factor = random( 3, 12 );
    let size = random( 5, 40 );
    fill( r, g, b );
    Flower( xc, yc, factor, size, 150, 1 );
  }
}
```

Number of flowers to draw

Number of line segments to use

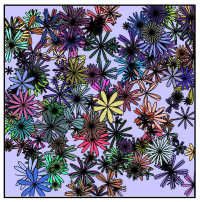
```
function Flower(xc, yc, factor, size, numsegs, numturns )
{
  let dang = numturns * (2.*PI) / float( numsegs );
  let theta = 0.;
  beginShape( );
  for( let i = 0; i <= numsegs; i = i + 1 )
  {
    let r = size * sin(factor*theta);
    let x = xc + r * cos(theta);
    let y = yc + r * sin(theta);
    vertex( x, y );
    theta = theta + dang;
  }
  endShape( );
}
```

76

The Program Randomly Chooses the Flower's Color, Position, Size, and Number of Petals

77

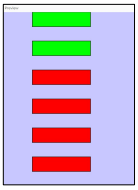
You Get a Different Garden Every Time You Run the Program!



78

79

if-statements



Oregon State University
Computer Graphics

mjb – July 21, 2023

79

80

Your Code Often Wants to Test Something and Make a Decision Based On It

```
if( condition )
{
        do this;
        do that;
}
```

These Operators Are the Possible Conditions to Test For:

<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to
&&	And
||	Or

Oregon State University
Computer Graphics

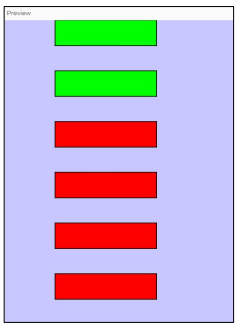
mjb – July 21, 2023

80

81

Example

```
function
draw( )
{
  let x = 100;
  fill( 0, 255, 0 );
  for( let y = 0 ; y <= 500 ; y = y + 100 )
  {
    if( y >= 200 )
    {
        fill( 255, 0, 0 );
    }
    rect( x, y, 200, 50 );
  }
}
```



Oregon State University
Computer Graphics

mjb – July 21, 2023

81

82

Your Code Often Wants to Test Something and Make a Decision Based On It or the Opposite Condition

```
if( condition )
{
        do this1;
        do this2;
}
else
{
        do that1;
        do that2;
}
```

Oregon State University
Computer Graphics

mjb – July 21, 2023

82

83

Your Code Often Wants to Test Something and Make a Decision Based On It or on Other Conditions

```
if( condition )
{
        do this;
}
else if( another_condition )
{
        do it;
}
else
{
        do that;
}
```

Oregon State University
Computer Graphics

mjb – July 21, 2023

83

84

Your Code Often Wants to Test Something and Make a Decision Based On It or Lots of Alternatives

key is a Processing built-in variable that tells you what key has been hit on the keyboard

```
if( key == 'r' )
{
        fill( 255,  50,   50 );
}
else if( key == 'g' )
{
        fill(  50,  255,   50 );
}
else if( key == 'b' )
{
        fill(  50,   50,  255 );
}
else
{
        fill( 100, 100, 100 );
}
```

Oregon State University
Computer Graphics

mjb – July 21, 2023

84

Your Code Often Wants to Test Something and Make a Decision Based On It or Lots of Alternatives -- a Better Way

```
switch( key )
{
    case 'r':
        fill( 255, 50, 50 );
        break;

    case 'g':
        fill( 50, 255, 50 );
        break;

    case 'b':
        fill( 50, 50, 255 );
        break;

    default:
        fill( 100, 100, 100 );
}
```


85

Some of Processing's Variables Already Have the Condition Built-In

```
function
setup( )
{
 createCanvas( 600, 600 );
 background( 200, 200, 255 );
 stroke( 0, 0, 0 );
 fill( 255, 255, 0 );
}

function
draw( )
{
  if( mouseIsPressed )
  {
    rect( mouseX, mouseY, 50, 20 );
  }
}
```

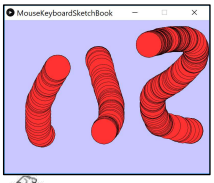
mouseIsPressed is a built-in variable that is always telling you if a mouse button is currently pressed

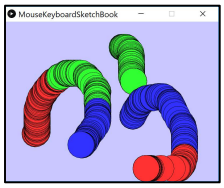
mouseX and ***mouseY*** are built-in variables that are always telling you where the mouse cursor is

86

Reacting to the Mouse and Keyboard: Creating Your Own Paint Program

Oregon State University





87

The *mouseIsPressed*, *mouseX*, and *mouseY* Variables

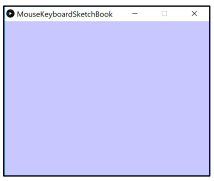
```
function
setup( )
{
 createCanvas( 600, 600 );
 background( 200, 200, 255 );
 stroke( 0, 0, 0 );
 fill( 255, 255, 0 );
}

function
draw( )
{
  if( mouseIsPressed )
  {
    ellipse( mouseX, mouseY, 50, 50 );
  }
}
```

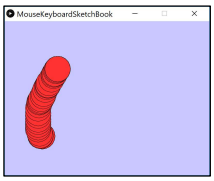
mouseIsPressed is a built-in variable that is always telling you if a mouse button is currently pressed

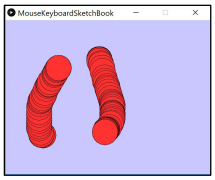
mouseX and ***mouseY*** are built-in variables that are always telling you where the mouse cursor is

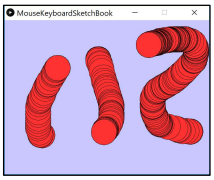
88

The *mouseIsPressed*, *mouseX*, and *mouseY* Variables









89

The *isKeyPressed* and *key* Variables

```
function
draw( )
{
  if( keyIsPressed )
  {
    switch( key )
    {
      case 'r':
          fill( 255, 50, 50 );
          break;

      case 'g':
          fill( 50, 255, 50 );
          break;

      case 'b':
          fill( 50, 50, 255 );
          break;
    }
  }

  if( mouseIsPressed )
  {
    ellipse( mouseX, mouseY, 50, 50 );
  }
}
```

keyIsPressed is a built-in variable that is always telling you if a keyboard key has been pressed

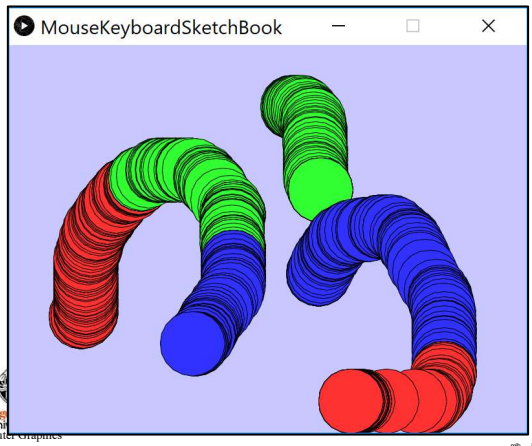
key is a built-in variable that tells you what key has been hit on the keyboard

The ***switch/case*** statements are Processing's way of checking many values without having a whole slew of if-statements

mouseIsPressed is a built-in variable that is always telling you if a mouse button has been pressed

90

91



Oregon State University
Computer Graphics

mjb – July 21, 2023

91

What if you want to read the Special Keys?

92

```
. . .

if( keyIsPressed )
{
    if( key == CODED )
    {
        switch( keyCode )
        {
            case UP:        // up-arrow
                . . .
                break;
        }

    }
}
```

Values for **keyCode** can be:
- UP
- DOWN
- LEFT
- RIGHT
- ESC
- DELETE
- BACKSPACE
- TAB
- ENTER
- RETURN

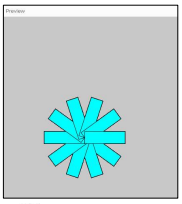
Oregon State University
Computer Graphics

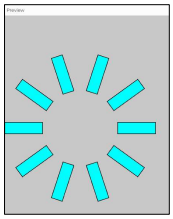
mjb – July 21, 2023

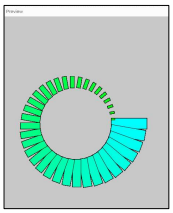
92

93

Transformations







Oregon State University

Oregon State University
Computer Graphics

mjb – July 21, 2023

93

Let's Use Our Rectangle Object as an Example of Transformations

94

```
function
setup( )
{
  createCanvas( 800, 800 );
  background( 200, 200, 200 );
  stroke( 0, 0, 0 );
  fill(   0, 255, 255 );
}

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

Oregon State University
Computer Graphics

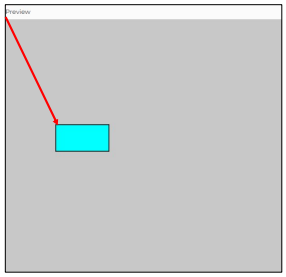
mjb – July 21, 2023

94

It is Often Nice to Transform Entire Objects at Once

95

```
function
setup( )
{
  createCanvas( 800, 800 );
  background( 200, 200, 200 );
  stroke( 0, 0, 0 );
  fill(   0, 255, 255 );
}

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



The word "translate" means to "move around"

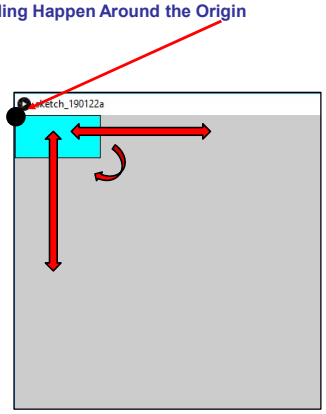
Oregon State University
Computer Graphics

mjb – July 21, 2023

95

Rotations and Scaling Happen Around the Origin

96



Oregon State University
Computer Graphics

mjb – July 21, 2023

96

Rotation

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

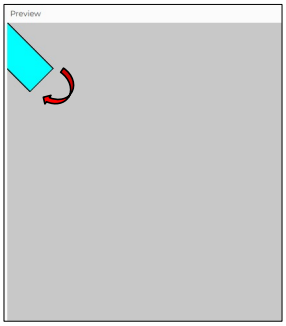
In math, science, and computer programming, angles are not given in degrees, they are given in ***radians***.

1 radian = 0.01745 degrees
1 radian = π/180. degrees

But, don't worry about this.

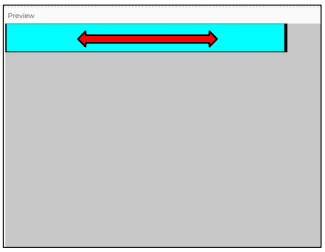
Processing gives you a function, **radians( )**, to automatically convert degrees into radians, like this:
rad = radians(deg);

Use it!



Scaling

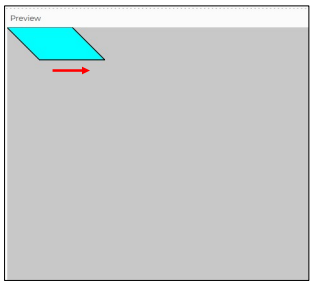
```
function
draw( )
{
  scale( 5., 1. );
  rect( 0, 0, 100, 50 );
}
```



Shearing

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

There is also a **shearY** transformation function



Transformations Accumulate!

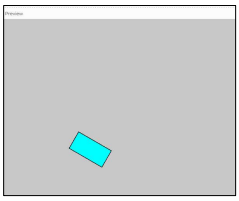
```
function
draw( )
{
  rotate( radians( 10. ) );
  rotate( radians( 10. ) );
  . . .
}
```

is the same as:

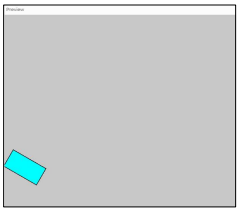
```
function
draw( )
{
  rotate( radians( 20. ) );
. . .
}
```


Transformation Order Matters!

```
function
draw( )
{
2. translate( 200, 300 );
1. rotate( radians(60.) );

  rect( 0, 0, 100, 50 );
}
```

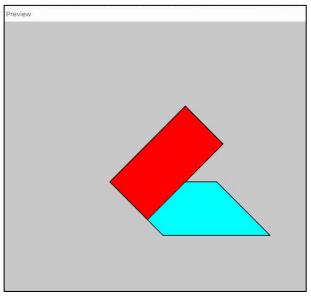



```
function
draw( )
{
2. rotate( radians(60.) );
1. translate( 200, 300 );

  rect( 0, 0, 100, 50 );
}
```



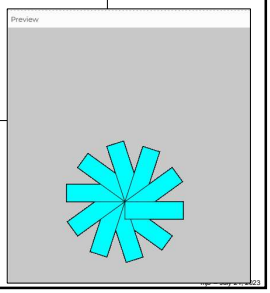
You Can Save and Restore Transformations

```
function
draw( )
{
2. translate( 200, 300 );

  push( );        "save"
1.  shearX( radians(45.) );
    rect( 0, 0, 200, 100 );
  pop( );         "restore"

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



Transformations and for-loops

```
function
draw( )
{
  translate( 200, 300 );

  for( let degrees = 0 ; degrees <= 360 ; degrees = degrees + 36 )
  {
    push( );
      rotate(  radians(degrees)  );
      rect(  0,  0,   100, 30 );
    pop( );
  }
}
```

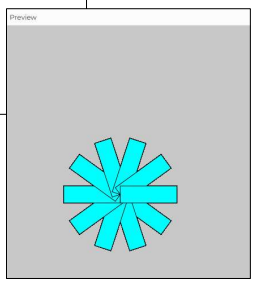


Oregon State University
Computer Graphics

103

Transformations and for-loops

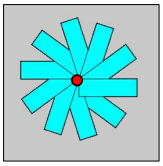
```
function
draw( )
{
  translate( 200, 300 );

  for( let degrees = 0 ; degrees <= 360 ; degrees = degrees + 36 )
  {
    push( );
      rotate(  radians(degrees)  );
      rect(  0,  -15,   100, 30 );
    pop( );
  }
}
```



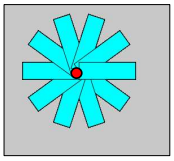
Oregon State University
Computer Graphics

104

What's the Difference?



```
push( );
  rotate(  radians(degrees)  );
  rect(  0,  0,   100, 30 );
pop( );
```



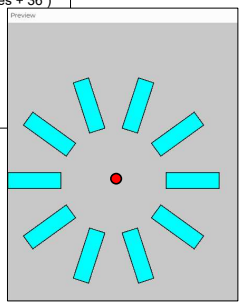
```
push( );
  rotate(  radians(degrees)  );
  rect(  0,  -15,   100, 30 );
pop( );
```

Oregon State University
Computer Graphics

105

Transformations and for-loops

```
function
draw( )
{
  translate( 200, 300 );

  for( let degrees = 0 ; degrees <= 360 ; degrees = degrees + 36 )
  {
    push( );
      rotate(  radians(degrees)  );
      rect(  100, -15,   100, 30 );
    pop( );
  }
}
```

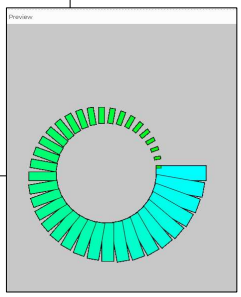


Oregon State University
Computer Graphics

106

Rotating While Changing Color and Size

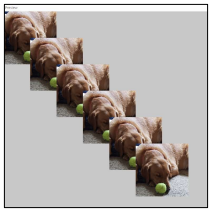
```
function
draw( )
{
  translate( 200, 300 );
  for( let degrees = 0 ; degrees <= 360 ; degrees = degrees + 10 )
  {
    push( );
      let blue = map( degrees,   0, 360,   255, 0 );
      fill( 0, 255, blue );
      rotate(  radians(degrees)  );
      let xsize = map( degrees,   0, 360,   100, 10 );
      let ysize = map( degrees,   0, 360,     30,   5 );
      rect(  100, -15,   xsize, ysize );
    pop( );
  }
}
```



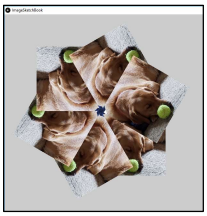
Oregon State University
Computer Graphics

107

Images in *Processing* Programming



Oregon State University



Oregon State University
Computer Graphics

108

Let's Start with a Favorite Image

109

Zelda-the-WonderDog

It can be in .jpg, .bmp, or .png format

Each pixel contains a red-green-blue, each in the range 0-255

The image has an ***aspect ratio***, which is the ratio of the *number of Y pixels : the number of X pixels* (this image's aspect ratio is 1:1)

Oregon State University Computer Graphics

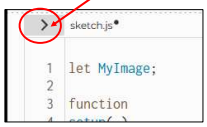
mjb – July 21, 2023

109

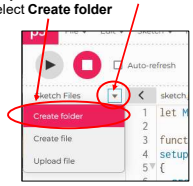
Loading Your Image into Your Program's *assets* Area

110

Step #1: Click on this arrow

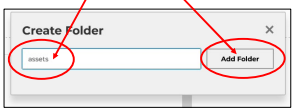


Step #2: Click on this arrow and select **Create folder**



*If you already have an **assets** folder, then you can skip steps #2 and #3.*

Step #3: Enter ***assets*** as the name of the folder and click on **Add Folder**



Oregon State University Computer Graphics

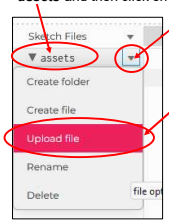
mjb – July 21, 2023

110

Loading Your Image into Your Program's Assets Area

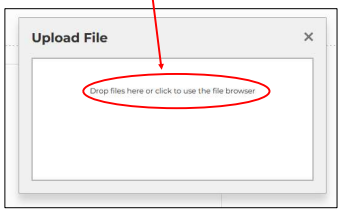
111

Step #4: Hover over the word ***assets*** and then click on this arrow



Step #5: Click on ***Upload file***

Step #6: Drag your image file into this window or click on this window to browse to the image file



Oregon State University Computer Graphics

mjb – July 21, 2023

111

Loading and Drawing an Image

112

```
let MyImage;

function
setup( )
{
  createCanvas( 800, 800 );
  MyImage = loadImage( "assets/zelda.jpg" );
  background( 200, 200, 200 );
  stroke( 0, 0, 0 );
  fill( 255, 50, 50 );
}

function
draw( )
{
  image( MyImage,  100, 100,  400, 400 );
}
```

Declaring a variable up here, ahead of everything else, makes it so that it can be seen from *anywhere* in the program.

This tells your program to look for the image in its assets area

This loads the image from the assets area into the variable called MyImage

This function draws the image from the variable called MyImage

How many pixels to use to draw the image

What X-Y to draw its upper-left corner at

Oregon State University Computer Graphics

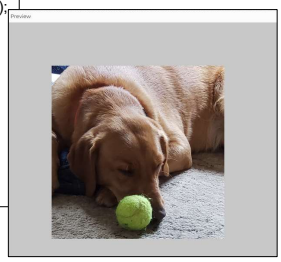
mjb – July 21, 2023

112

Loading and Drawing an Image

113

```
let MyImage;

function
setup( )
{
  createCanvas( 800, 800 );
  MyImage = loadImage( "assets/zelda.jpg" );
  background( 200, 200, 200 );
  stroke( 0, 0, 0 );
  fill( 255, 50, 50 );
}

function
draw( )
{
  image( MyImage,  100, 100,  400, 400 );
}
```



Oregon State University Computer Graphics

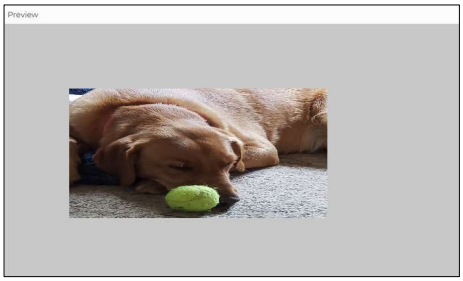
mjb – July 21, 2023

113

What Happens if You Ask For a Different Aspect Ratio?

114

```
function
draw( )
{
  image( MyImage,  100, 100,  400, 200 );
}
```



Oregon State University Computer Graphics

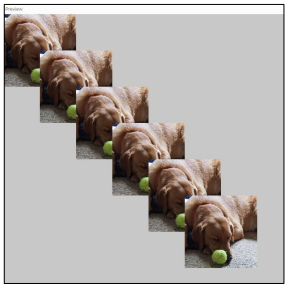
mjb – July 21, 2023

114

Translating an Image

115

```
function
draw( )
{
  for( let i = 0 ; i < 6 ; i = i + 1 )
  {
    push( );
      translate( i*100, i*100 );
      image( MyImage,  0, 0,  200, 200 );
    pop( );
  }
}
```



Notice how transforming images works just like transforming rectangles does!

Oregon State University Computer Graphics

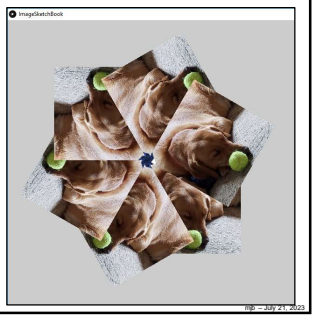
mjb – July 21, 2023

115

Rotating an Image

116

```
function
draw( )
{
  for( let i = 0 ; i < 6 ; i = i + 1 )
  {
    push( );
  2. translate( 300, 300 );
  1. rotate( radians(i*60) );
      image( MyImage,  0, 0,  200, 200 );
    pop( );
  }
}
```



Notice how transforming images works just like transforming rectangles does!

Oregon State University Computer Graphics

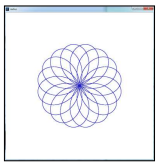
mjb – July 21, 2023

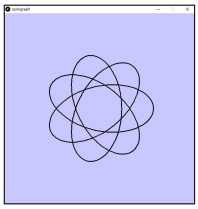
116

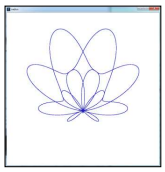
Advanced Polar Patterns

117

Oregon State University







Oregon State University Computer Graphics

mjb – July 21, 2023

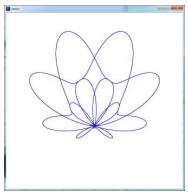
117

Some Other Polar Patterns

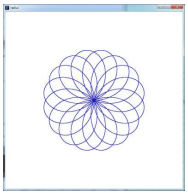
118

$$r = \sin\theta + \sin^3\left(\frac{5\theta}{2}\right)$$

Note: $x^3 = x*x*x$



$$r = \sin\left(\frac{8\theta}{5}\right)$$



James Stewart, *Single Variable Calculus*, Brooks/Cole, 2001.

Oregon State University Computer Graphics

mjb – July 21, 2023

118

Imitating a Spirograph™

119

Looks like an Oreo, but it's not. ☺

Oregon State University Computer Graphics

mjb – July 21, 2023

119

Imitating a Spirograph™

120

Oregon State University Computer Graphics

mjb – July 21, 2023

120

Imitating a Spirograph™

```
let BigR     = 200.;
let SmallR = 150.;
let D         = 120.;

function
setup( )
{
  createCanvas( 800, 800 );
  background( 200, 200, 255 );
  stroke( 0, 0, 0 );
  strokeWeight( 2 );
  noFill( );
}
```

Oregon State University
Computer Graphics

121

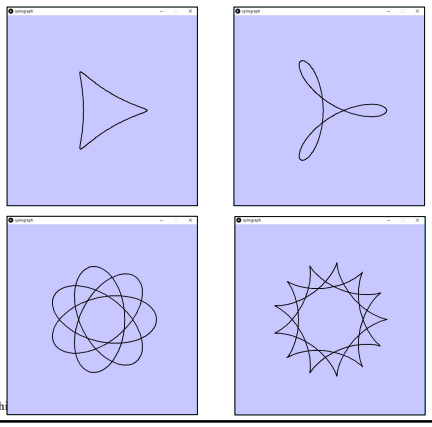
Imitating a Spirograph™

```
function
draw( )
{
  translate( 400, 400 );

  beginShape( );

  for( let t = 0; t <= 10*360; t = t + 2 )
  {
    let bigTheta = radians( t );
    let smallTheta = - ( BigR / SmallR ) * bigTheta;
    let x = ( BigR - SmallR ) * cos( bigTheta )  +  D * cos( smallTheta );
    let y = ( BigR - SmallR ) * sin( bigTheta )  +  D * sin(  smallTheta );
    vertex( x, y );
  }

  endShape( );
}
```

Oregon State University
Computer Graphics

122

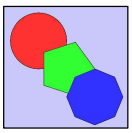
Imitating a Spirograph™

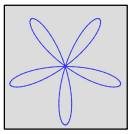


Oregon State University
Computer Graphi

123

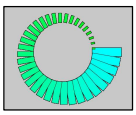
Programming with Processing !

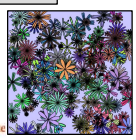


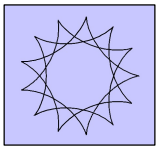


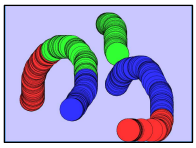
Oregon State University

Mike Bailey
mjb@cs.oregonstate.edu









Oregon State University
Computer Graphics

ProcessingSteinAcademy.pptx

124