Computational Fabrication

CS 491 and 591, Special Topics in Computer Science
Professor: Leah Buechley
https://handandmachine.org/classes/computational_fabrication

WordPress FYI & Introduction

Create an account
use your full name and UNM email
add your full name to your account if you haven't

FERPA
All student activity is private
including all course work and comments

Syllabus, Schedule, Assignments, Resources are public

WordPress Creating a Post

Posts

Categories

Inserting images & other files

HTML

Small Assignment

Due Tuesday

Small Assignment 1: Introduce yourself

Find and share and example of computational design work you love

Put in “Small Assignment - Introductions” subcategory under “Student Work 2023” category

Due Thursday

Comment on at least three peer posts

questions?

Processing

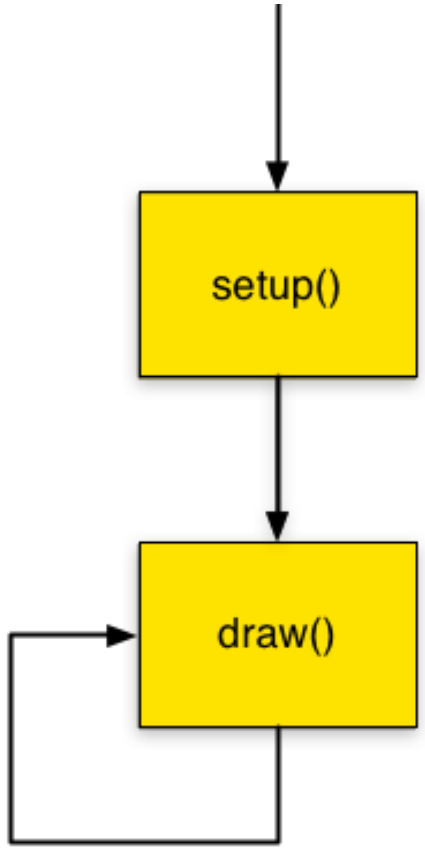
Processing + P5JS

- Java based programming environment +
- Javascript based programming environment
- Lots of built in libraries
- Excellent documentation
- Vibrant community of artists and designers
- Makes programming of visual and interactive elements quick an easy
- Open Processing: https://openprocessing.org/

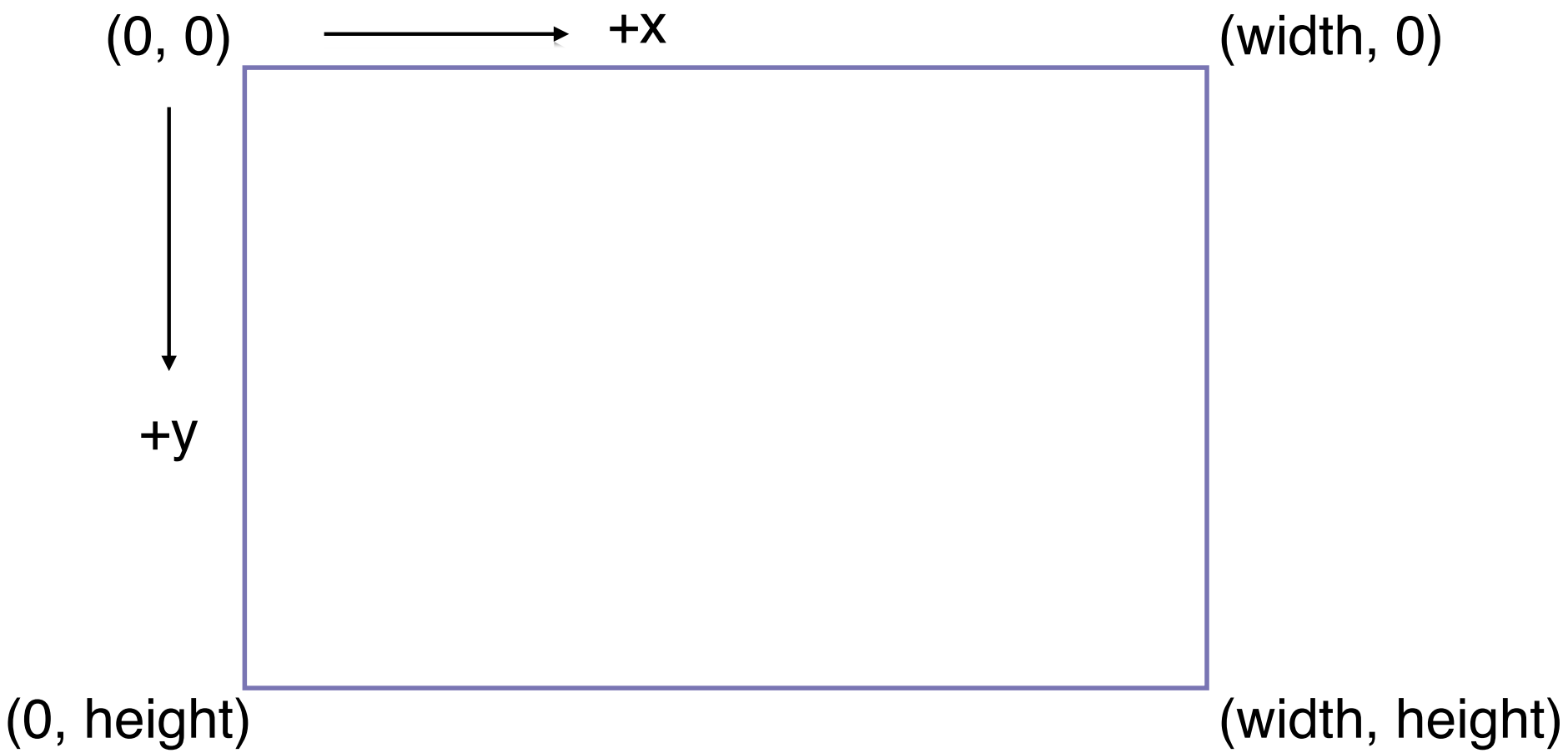
open up Processing &
Processing reference

Program Structure

```
//this runs once
void setup () {
}

//this loops forever
void draw () {
}
```



Coordinate System



Simple Program

```
void setup() {
    size(700,500);
    background(255);
}

void draw() {
    line(0,0,100,100);
}
```

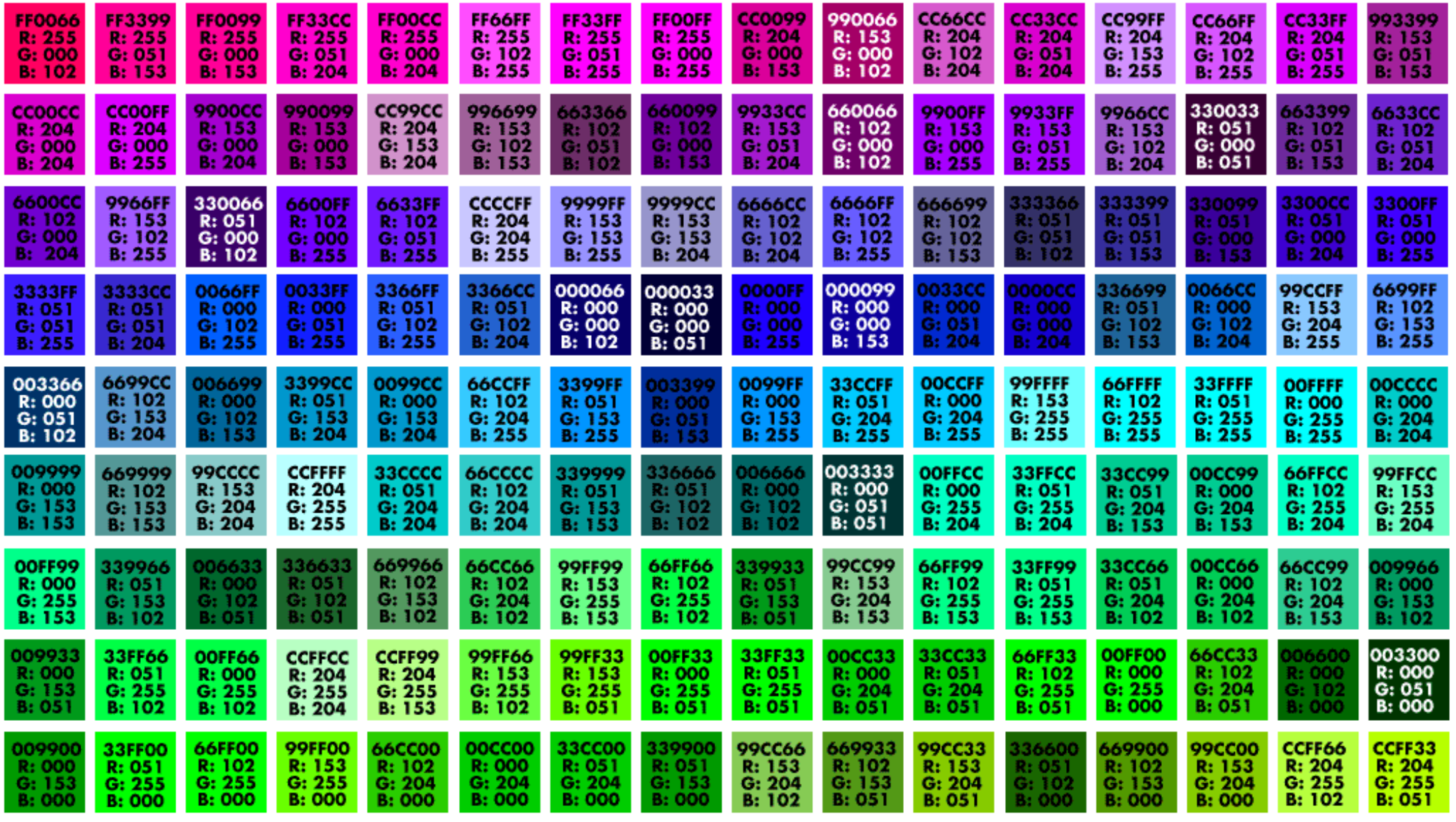
Save your “Sketch”

Built-in Variables

```
void setup() {
    size(700,500);
    background(255);
}


void draw() {
    line(width/2,height/2,width,height);
}
```

Built-in Variables, Interactivity

```
void setup() {
    size(700,500);
    background(255);
}

void draw() {
    line(0,0,mouseX,mouseY);
}
```

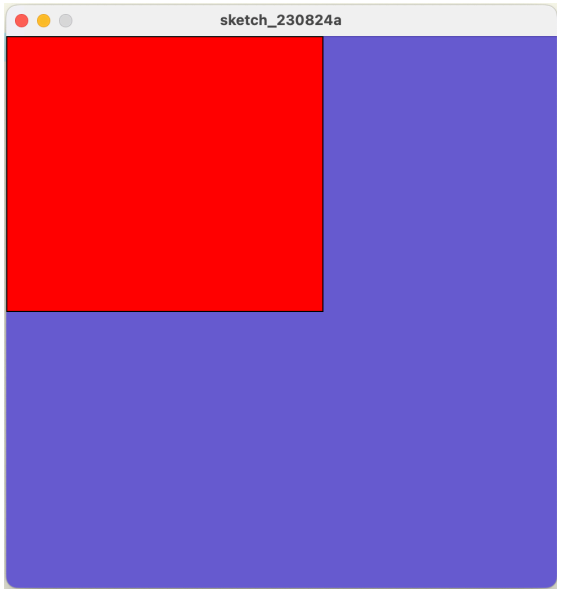

FF0066 R: 255 G: 000 B: 102
FF3399 R: 255 G: 051 B: 153
FF0099 R: 255 G: 000 B: 153
FF33CC R: 255 G: 051 B: 204
FF00CC R: 255 G: 000 B: 204
FF66FF R: 255 G: 102 B: 255
FF33FF R: 255 G: 051 B: 255
FF00FF R: 255 G: 000 B: 255
CC0099 R: 204 G: 000 B: 153
990066 R: 153 G: 000 B: 102
CC66CC R: 204 G: 102 B: 204
CC33CC R: 204 G: 051 B: 204
CC99FF R: 204 G: 153 B: 255
CC66FF R: 204 G: 102 B: 255
CC33FF R: 204 G: 051 B: 255
993399 R: 153 G: 051 B: 153
CC00CC R: 204 G: 000 B: 204
CC00FF R: 204 G: 000 B: 255
9900CC R: 153 G: 000 B: 204
990099 R: 153 G: 000 B: 153
CC99CC R: 204 G: 153 B: 204
996699 R: 153 G: 102 B: 153
663366 R: 102 G: 051 B: 102
660099 R: 102 G: 000 B: 153
9933CC R: 153 G: 051 B: 204
660066 R: 102 G: 000 B: 102
9900FF R: 153 G: 000 B: 255
9933FF R: 153 G: 051 B: 255
9966CC R: 153 G: 102 B: 204
330033 R: 051 G: 000 B: 051
663399 R: 102 G: 051 B: 153
6633CC R: 102 G: 051 B: 204
6600CC R: 102 G: 000 B: 204
9966FF R: 153 G: 102 B: 255
330066 R: 051 G: 000 B: 102
6600FF R: 102 G: 000 B: 255
6633FF R: 102 G: 051 B: 255
CCCCFF R: 204 G: 204 B: 255
9999FF R: 153 G: 153 B: 255
9999CC R: 153 G: 153 B: 204
6666CC R: 102 G: 102 B: 204
6666FF R: 102 G: 102 B: 255
666699 R: 102 G: 102 B: 153
333366 R: 051 G: 051 B: 102
333399 R: 051 G: 051 B: 153
330099 R: 051 G: 000 B: 153
3300CC R: 051 G: 000 B: 204
3300FF R: 051 G: 000 B: 255
3333FF R: 051 G: 051 B: 255
3333CC R: 051 G: 051 B: 204
0066FF R: 000 G: 102 B: 255
0033FF R: 000 G: 051 B: 255
3366FF R: 051 G: 102 B: 255
3366CC R: 051 G: 102 B: 204
000066 R: 000 G: 000 B: 102
000033 R: 000 G: 000 B: 051
0000FF R: 000 G: 000 B: 255
000099 R: 000 G: 000 B: 153
0033CC R: 000 G: 051 B: 204
0000CC R: 000 G: 000 B: 204
336699 R: 051 G: 102 B: 153
0066CC R: 000 G: 102 B: 204
99CCFF R: 153 G: 204 B: 255
6699FF R: 102 G: 153 B: 255
003366 R: 000 G: 051 B: 102
6699CC R: 102 G: 153 B: 204
006699 R: 000 G: 102 B: 153
3399CC R: 051 G: 153 B: 204
0099CC R: 000 G: 153 B: 204
66CCFF R: 102 G: 204 B: 255
3399FF R: 051 G: 153 B: 255
003399 R: 000 G: 051 B: 153
0099FF R: 000 G: 153 B: 255
33CCFF R: 051 G: 204 B: 255
00CCFF R: 000 G: 204 B: 255
99FFFF R: 153 G: 255 B: 255
66FFFF R: 102 G: 255 B: 255
33FFFF R: 051 G: 255 B: 255
00FFFF R: 000 G: 255 B: 255
00CCCC R: 000 G: 204 B: 204
009999 R: 000 G: 153 B: 153
669999 R: 102 G: 153 B: 153
99CCCC R: 153 G: 204 B: 204
CCFFFF R: 204 G: 255 B: 255
33CCCC R: 051 G: 204 B: 204
66CCCC R: 102 G: 204 B: 204
339999 R: 051 G: 153 B: 153
336666 R: 051 G: 102 B: 102
006666 R: 000 G: 102 B: 102
003333 R: 000 G: 051 B: 051
00FFCC R: 000 G: 255 B: 204
33FFCC R: 051 G: 255 B: 204
33CC99 R: 051 G: 204 B: 153
00CC99 R: 000 G: 204 B: 153
66FFCC R: 102 G: 255 B: 204
99FFCC R: 153 G: 255 B: 204
00FF99 R: 000 G: 255 B: 153
339966 R: 051 G: 153 B: 102
006633 R: 000 G: 102 B: 051
336633 R: 051 G: 102 B: 051
669966 R: 102 G: 153 B: 102
66CC66 R: 102 G: 204 B: 102
99FF99 R: 153 G: 255 B: 153
66FF66 R: 102 G: 255 B: 102
339933 R: 051 G: 153 B: 051
99CC99 R: 153 G: 204 B: 153
66FF99 R: 102 G: 255 B: 153
33FF99 R: 051 G: 255 B: 153
33CC66 R: 051 G: 204 B: 102
00CC66 R: 000 G: 204 B: 102
66CC99 R: 102 G: 204 B: 153
009966 R: 000 G: 153 B: 102
009933 R: 000 G: 153 B: 051
33FF66 R: 051 G: 255 B: 102
00FF66 R: 000 G: 255 B: 102
CCFFCC R: 204 G: 255 B: 204
CCFF99 R: 204 G: 255 B: 153
99FF66 R: 153 G: 255 B: 102
99FF33 R: 153 G: 255 B: 051
00FF33 R: 000 G: 255 B: 051
33FF33 R: 051 G: 255 B: 051
00CC33 R: 000 G: 204 B: 051
33CC33 R: 051 G: 204 B: 051
66FF33 R: 102 G: 255 B: 051
00FF00 R: 000 G: 255 B: 000
66CC33 R: 102 G: 204 B: 051
006600 R: 000 G: 102 B: 000
003300 R: 000 G: 051 B: 000
009900 R: 000 G: 153 B: 000
33FF00 R: 051 G: 255 B: 000
66FF00 R: 102 G: 255 B: 000
99FF00 R: 153 G: 255 B: 000
66CC00 R: 102 G: 204 B: 000
00CC00 R: 000 G: 204 B: 000
33CC00 R: 051 G: 204 B: 000
339900 R: 051 G: 153 B: 000
99CC66 R: 153 G: 204 B: 102
669933 R: 102 G: 153 B: 051
99CC33 R: 153 G: 204 B: 051
336600 R: 051 G: 102 B: 000
669900 R: 102 G: 153 B: 000
99CC00 R: 153 G: 204 B: 000
CCFF66 R: 204 G: 255 B: 102
CCFF33 R: 204 G: 255 B: 051

Color

```
void setup() {
    size(700,500);
}

void draw() {
    background(100,90,200);
    line(0,0,mouseX,mouseY);
}
```

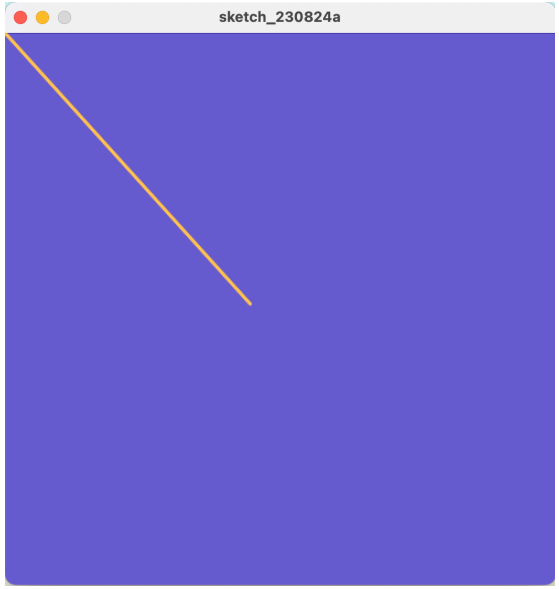
Color, Fill

```
void setup() {
    size(700,500);
    fill(255,0,0);
}

void draw() {
    background(100,90,200);
    rect(0,0,mouseX,mouseY);
}
```



Color, Stroke, Stroke Weight

```
void setup() {
    size(700,500);
    fill(255,0,0);
    stroke(255,200,100);
    strokeWeight(3);
}

void draw() {
    background(100,90,200);
    line(0,0,mouseX,mouseY);
}
```



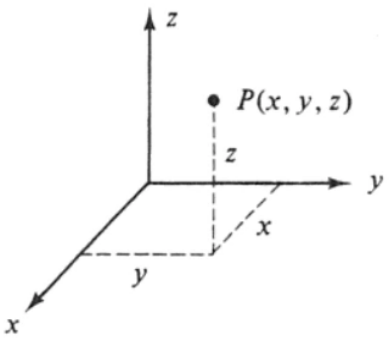
questions?

Math Review
trigonometry & coordinate systems

Coordinate systems

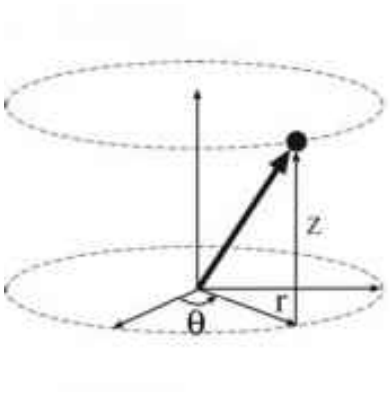
Cartesian

x,y,z



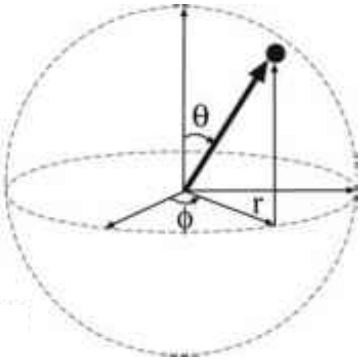
Polar/Cylindrical

r,θ,z



Spherical

r,θ,ϕ



Create a new Sketch

Line: Cartesian coordinate system

```
void setup() {
  size (300,300);
  background(220,200,100);
}

void draw() {
   line(width/2,height/2,mouseX,mouseY);
}
```

Line

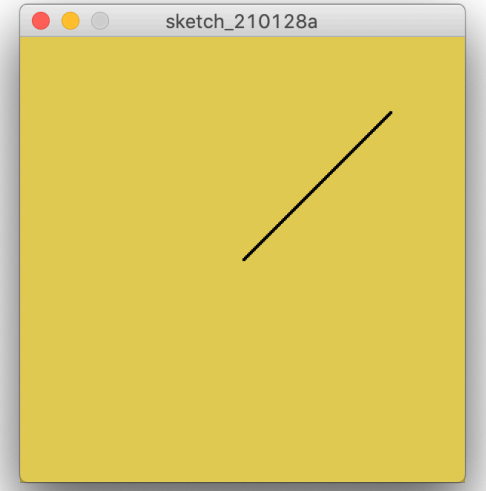
```
void setup() {
  size (300,300);
  background(220,200,100);
}

void draw() {
  line(width/2,height/2,width/2+100,height/2-100);
}
```

Line

```
void setup() {
  size (300,300);
  background(220,200,100);
}

void draw() {
  int x0,y0,x1,y1;
  x0 = width/2;
  y0 = height/2;
  x1 = x0+100;
  y1 = y0-100;
  line(x0,y0,x1,y1);
}
```

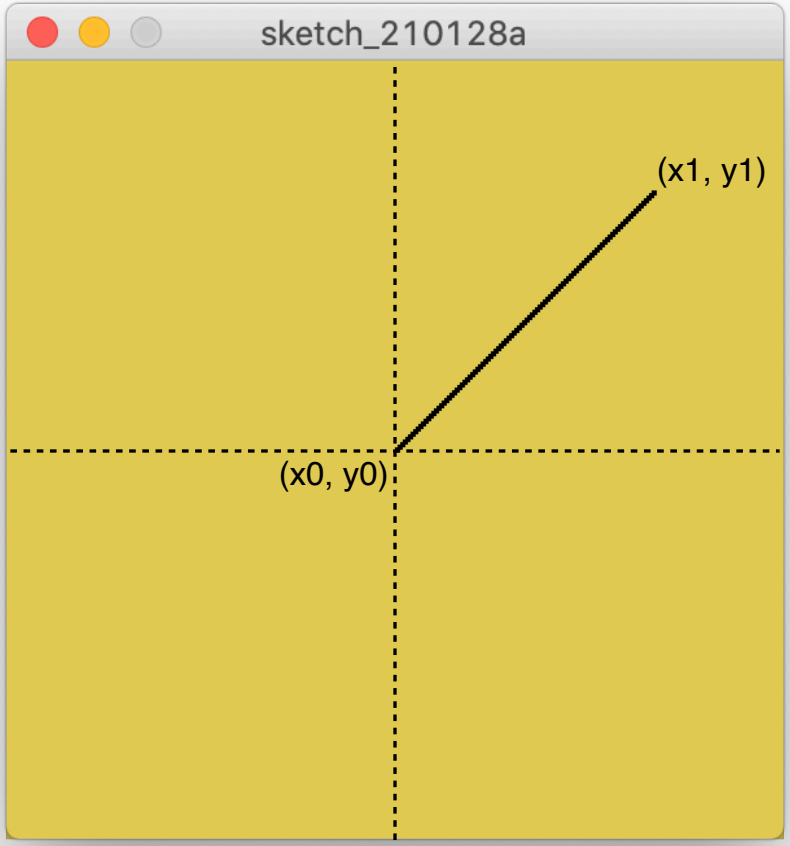
Line from x and y

```
void cartesianLine(float x0, float y0, float x1, float y1) {
  line(x0,y0,x1,y1);
}
```

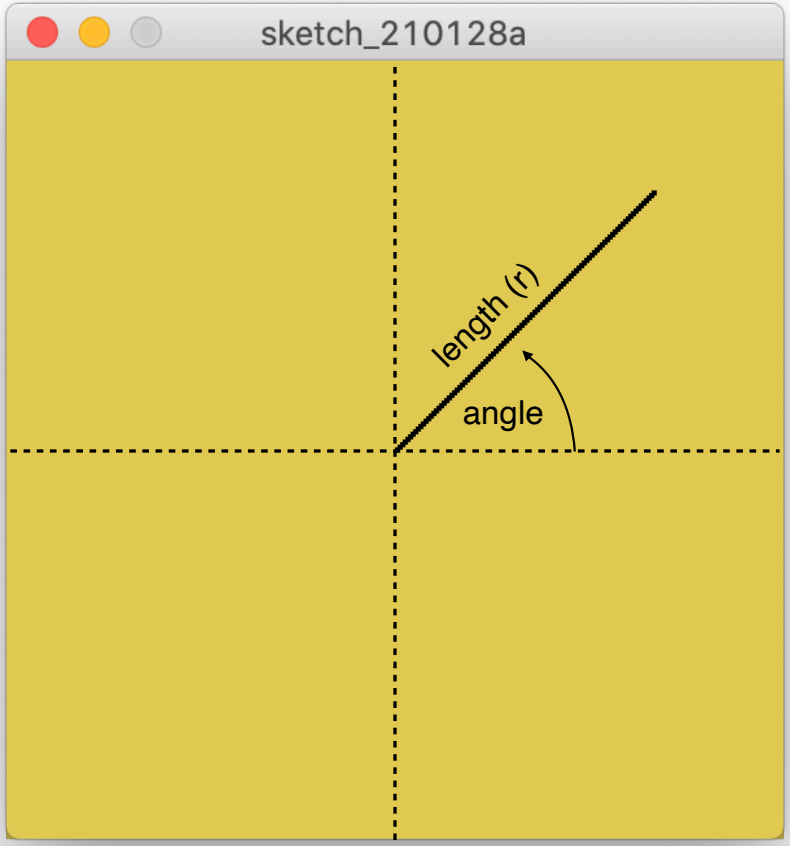


Line: Polar coordinate system?

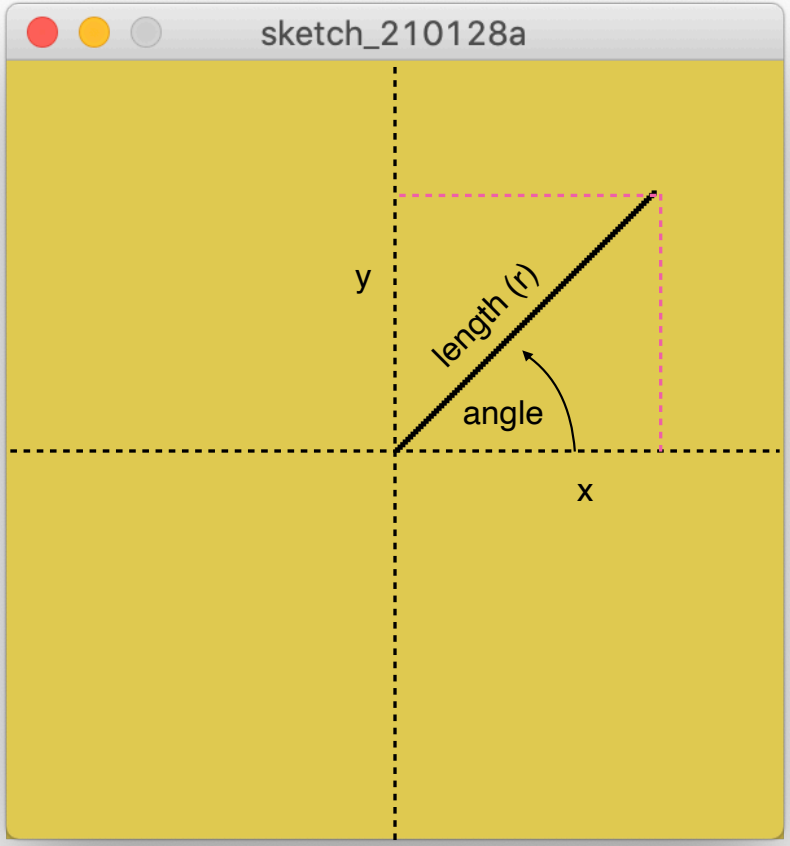
Line



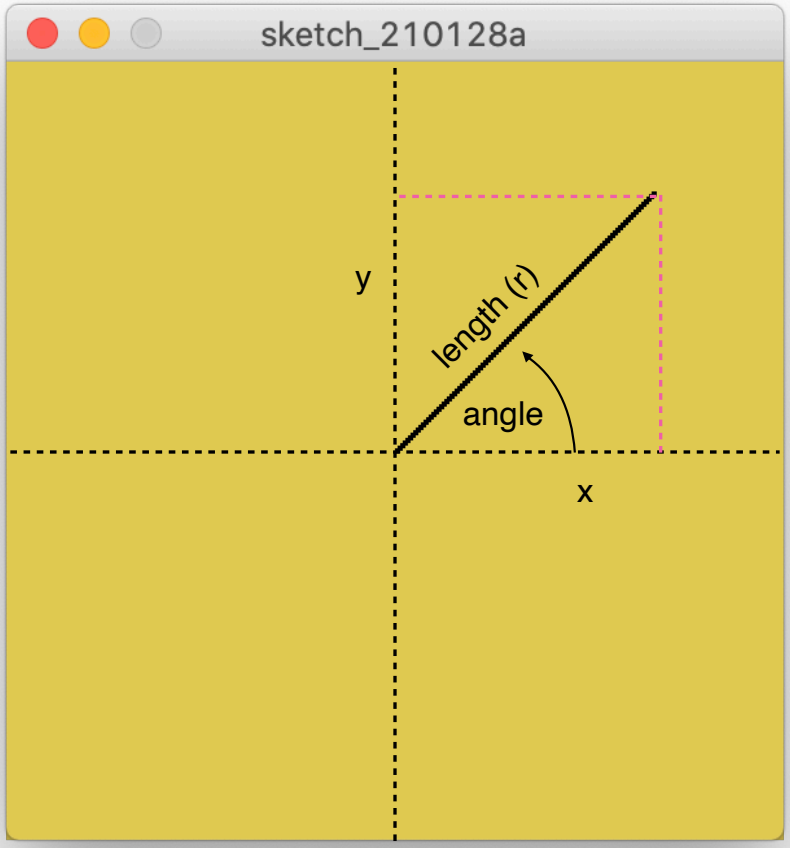
Line



Line



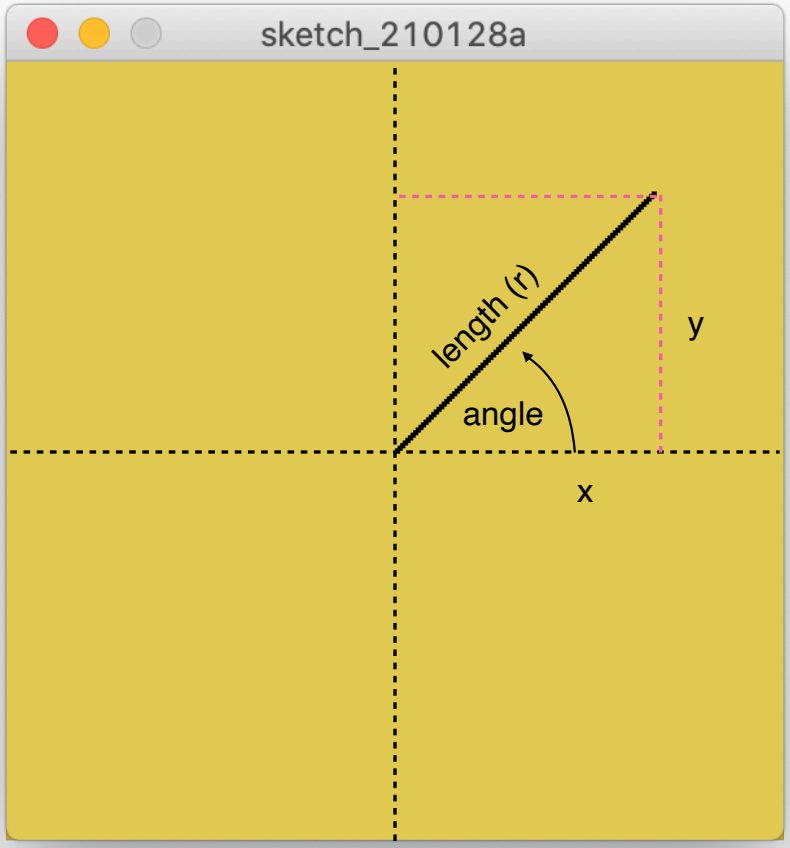
Line



length ?

r ?

Line

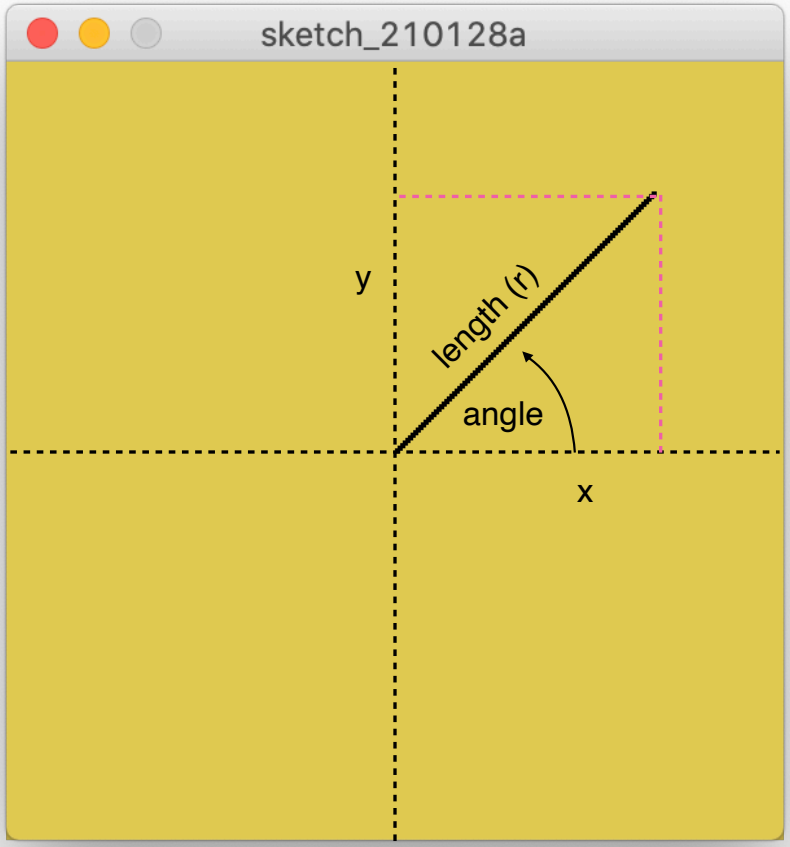


$$c^2 = a^2 + b^2$$

$$r^2 = x^2 + y^2$$

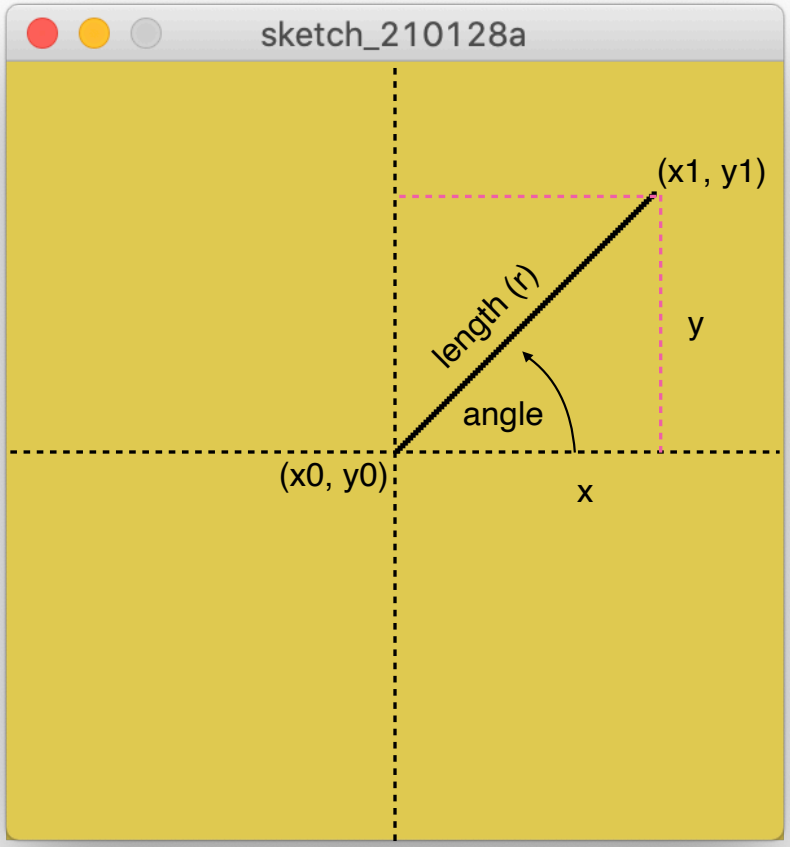
$$r = \sqrt{(x^2 + y^2)}$$

Line



angle ?

Line

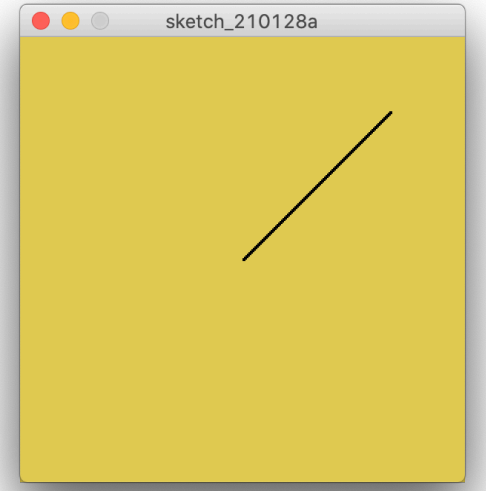


$$x = r * cos(angle)$$
$$y = r * sin(angle)$$

$$x1 = x0 + r * cos(angle)$$
$$y1 = y0 + r * sin(angle)$$

Line from r and θ

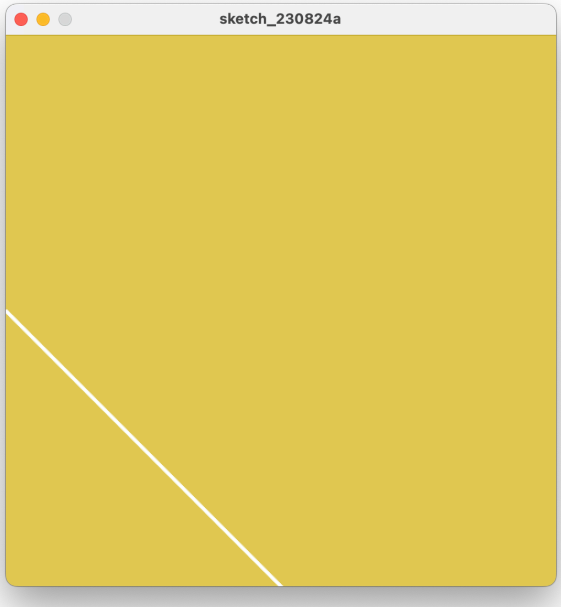
```
void polarLine(float x0, float y0, float r, float angle) {
  float x1 = x0 + r*cos(radians(angle));
  float y1 = y0 - r*sin(radians(angle));
  line(x0,y0,x1,y1);
}
```



questions?

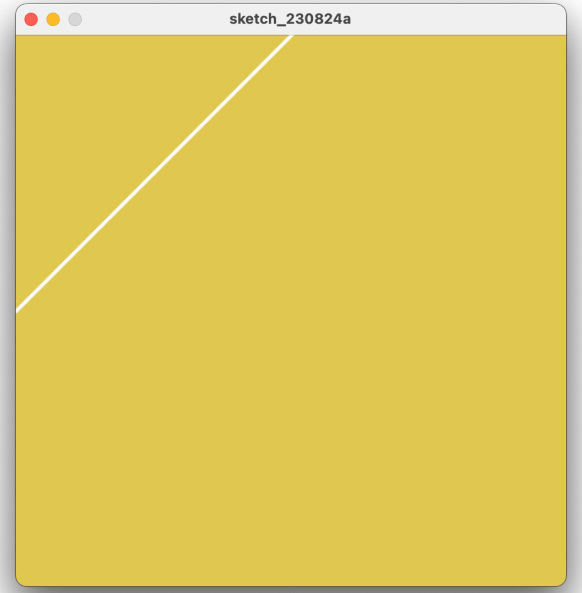
Cartesian Plots

```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 = x1;     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



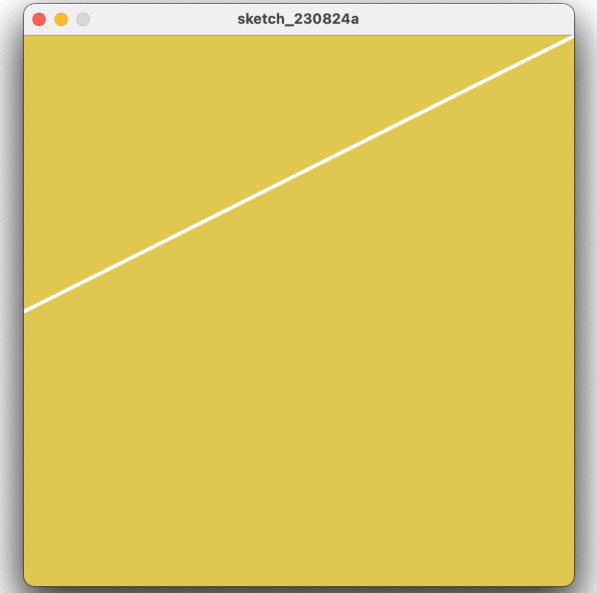
Cartesian Plots

```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 = -x1;     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



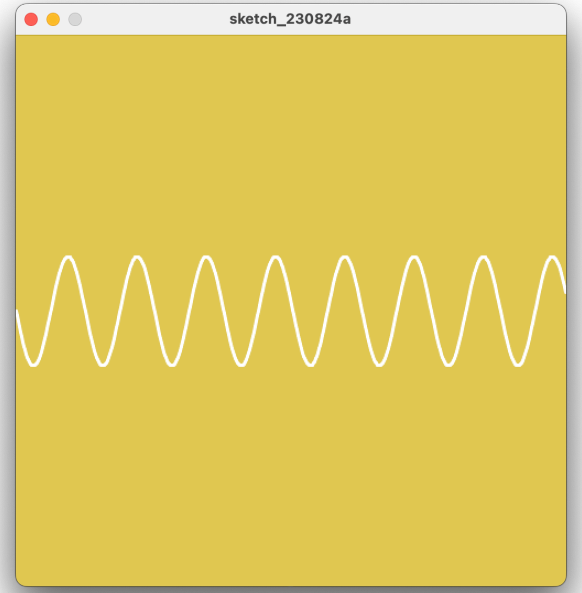
Cartesian Plots

```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 = -x1/2;     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



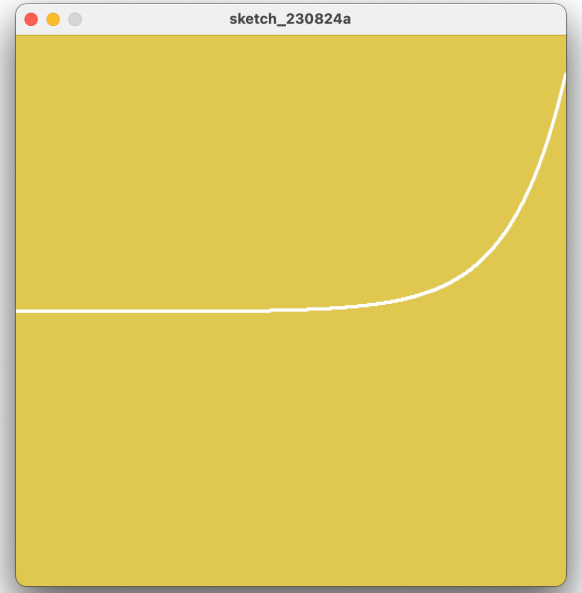
Cartesian Plots

```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 =(int)(sin(x1/10.0)*50);     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



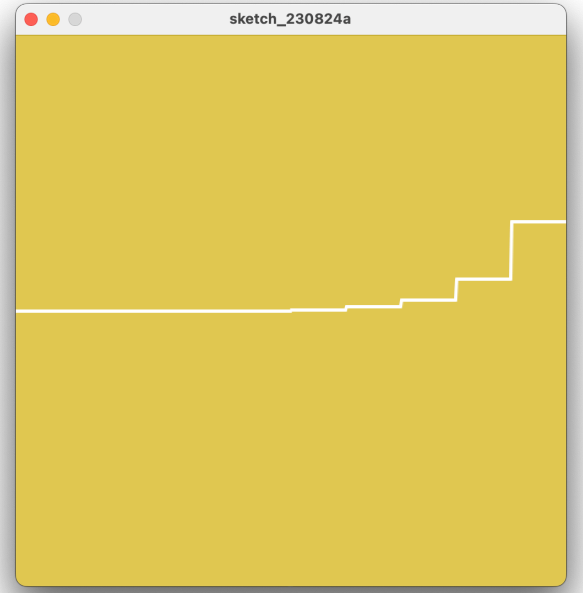
Cartesian Plots

```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 =-(int)(exp(x1/50.0)/100);     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



ints & floats, Be careful!

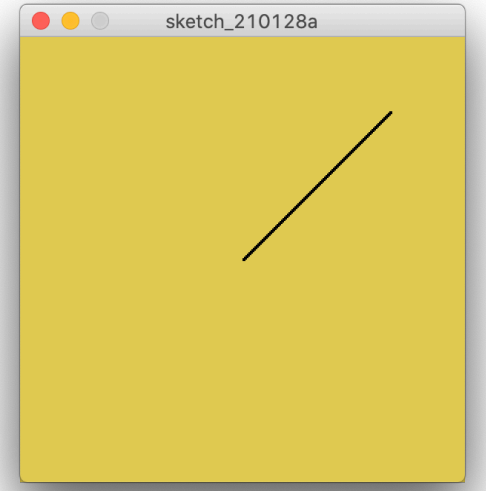
```
void cartesian_plot() {
  int x0,y0,x1,y1,y_offset;
  x0 = 0;
  y0 = 0;
  y_offset = width/2;
  for (x1=0;x1<width;x1++) {
    y1 =-(int)(exp(x1/50)/100);     //y = f(x)
    cartesianLine(x0,y0+y_offset,x1,y1+y_offset);
    x0 = x1;
    y0 = y1;
  }
}
```



questions?

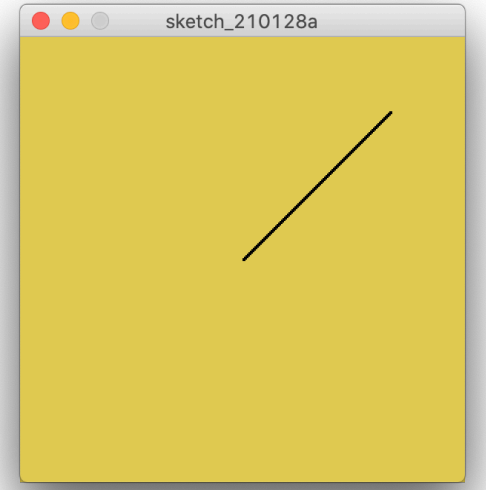
Polar Plots

```
void polarLine(float x0, float y0, float r, float angle) {
  float x1 = x0 + r*cos(radians(angle));
  float y1 = y0 - r*sin(radians(angle));
  line(x0,y0,x1,y1);
}
```



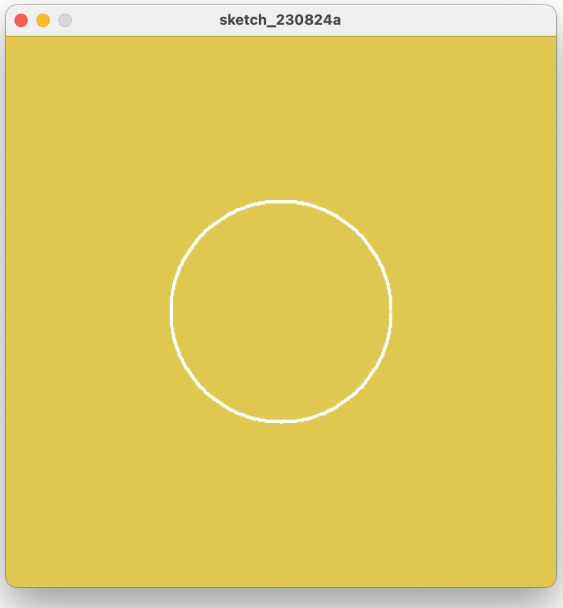
Point from from r and θ

```
float[] polarPoint(float r, float theta) {
  float [] point = new float[2];
  point[0] = width/2+r*cos(radians(theta));
  point[1] = height/2+r*sin(radians(theta));
  return point;
}
```



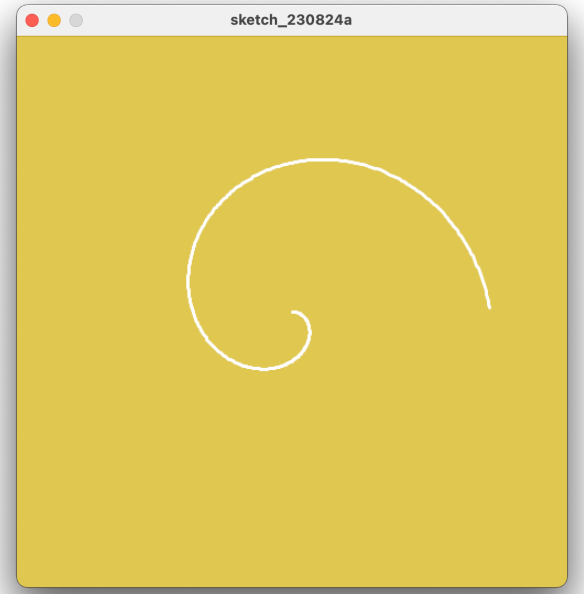
Polar Plots

```
void polar_plot () {
  float r;
  beginShape();
  for (int i=0; i<361; i++) {
    r = 100;    // r = f(theta)
    float[] point = polarPoint(r, i);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```

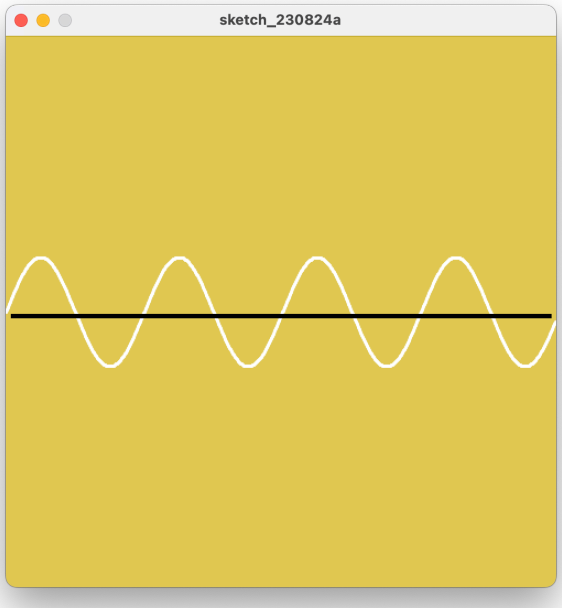


Polar Plots

```
void polar_plot () {
  float r;
  beginShape();
  for (int i=0; i<361; i++) {
    r = theta/2;     // r = f(theta)
    float[] point = polarPoint(r, i);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```

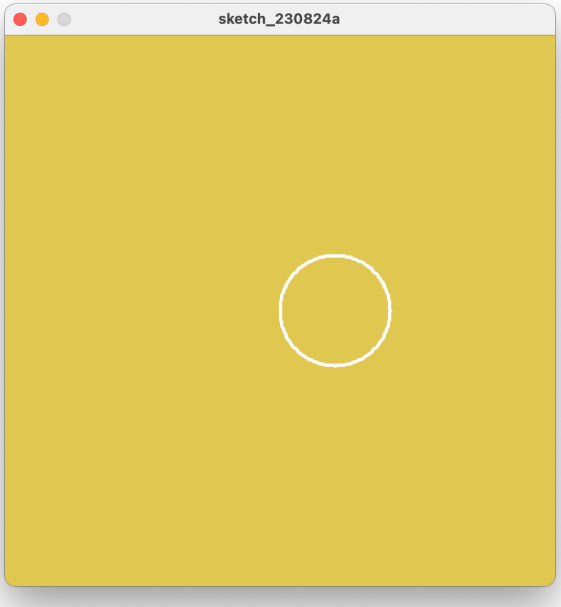


sin or cosine in polar coordinates?



Polar Plots

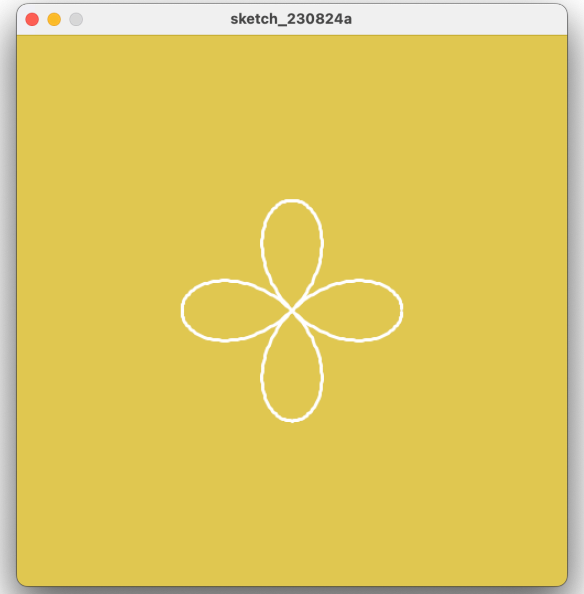
```
void polar_plot () {
  float r;
  float theta;
  beginShape();
  for (theta=0; theta<361; theta++) {
    r = cos(radians(theta))*100;    // r = f(theta)
    float[] point = polarPoint(r, theta);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```



one period, 0-2π, 0-360°

Polar Plots

```
void polar_plot () {
  float r;
  float theta;
  beginShape();
  for (theta=0; theta<361; theta++) {
    r = cos(radians(theta*2.0))*100;     // r = f(theta)
    float[] point = polarPoint(r, theta);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```

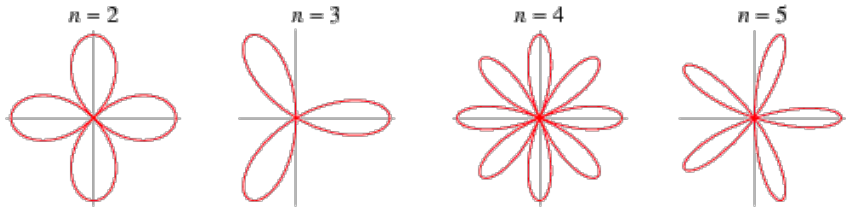


two periods, 0-4π, 0-720°

questions?

Polar Rose Equations

$$r = a\cos(n\theta)$$



a = amplitude of oscillation around circle

n = number of oscillations

Polar Rose Equation Extension

$$r = c + a\cos(n\theta)$$

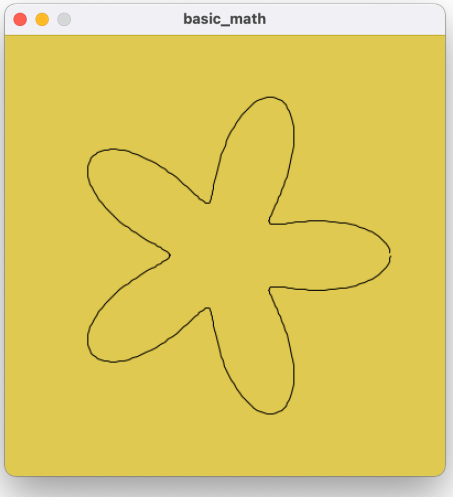
c = starting radius (opens up the shape if c > a)

a = amplitude of oscillation around circle

n = number of oscillations

Polar Plots

```
void plot () {
  float r;
  beginShape();
  for (int i=0; i<360; i++) {
    r = 100+50*cos(radians(i*5));
    float[] point = polarPoint(r, i);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```



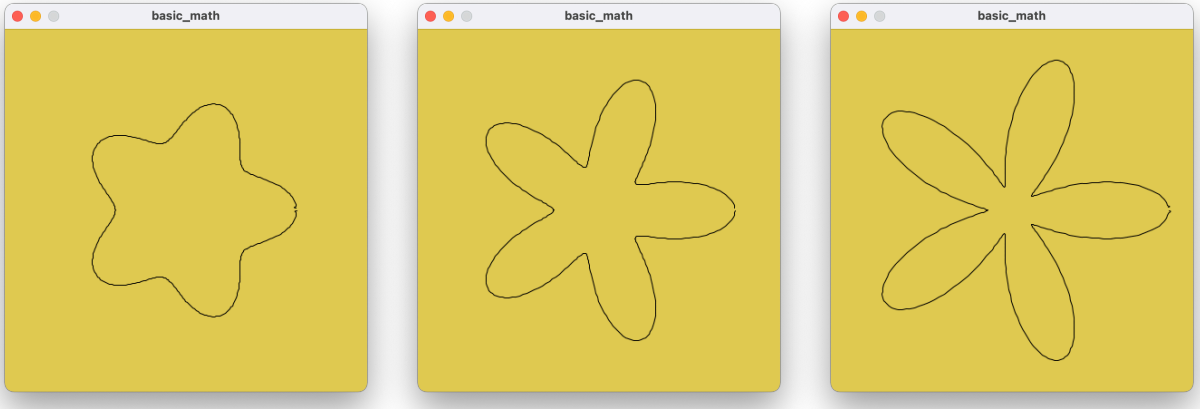
c = 100

a = 50

n = 5

Interactive Polar Plot

```
void plot () {
  float r;
  beginShape();
  for (int i=0; i<360; i++) {
    r = 100+mouseX/5*cos(radians(i*5));
    float[] point = polarPoint(r, i);
    curveVertex(point[0],point[1]);
  }
  endShape();
}
```



c = 100

a = mouseX/5

n = 5

questions?

For Next Class

First Small Assignment

Reading: Turtle Geometry

Install Turtle Library

Order your 3D printers

questions?

Thank you!

CS 491 and 591, Special Topics in Computer Science
Professor: Leah Buechley
https://handandmachine.org/classes/computational_fabrication