Computational Fabrication

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

Small Assignment 1

Upcoming Deadlines

Thursday 8/31
Intro to the Laser Cutter, Arts Lab, + Hand and Machine Lab
led by Fiona Bell and Valery Estabrook
due: comments on introduction posts

Tuesday 9/5
L-Systems
due: reading: L-Systems

3D Printers

Order now!

I’ll supply filament.

questions?

Turtle Geometry & LOGO

LOGO

An embodied approach to geometry

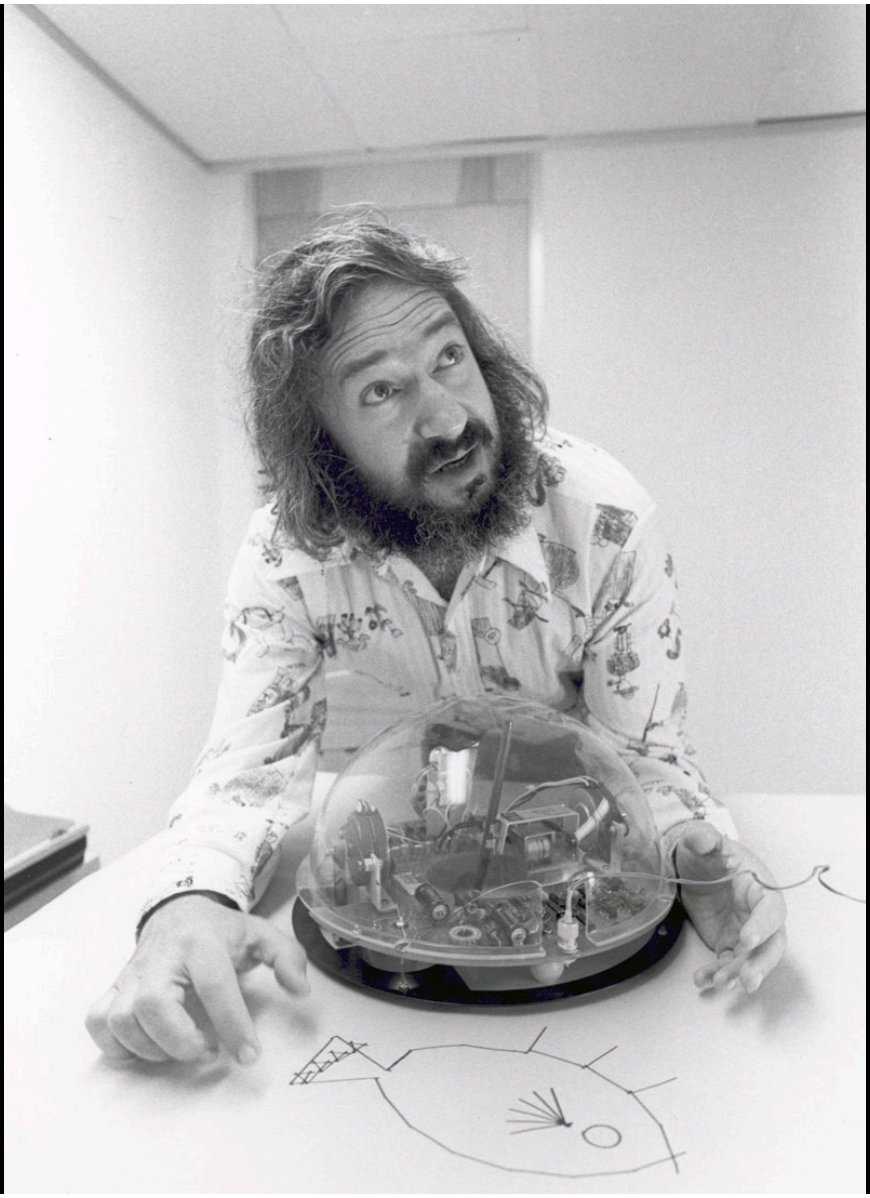
You direct a “turtle” to move around giving it simple directions.

Developed in 1967 by
Seymour Papert, Cynthia Solomon, and Wally Feurzeig

A language designed to enable children to explore
mathematics and computers

Turtle robot in 1969

Mindstorms published in 1980
Turtle Geometry published in 1981



Seymour Papert and students, 1969

open up Processing

Turtle Libaray needs to be installed

http://leahbuechley.com/Turtle/index.html

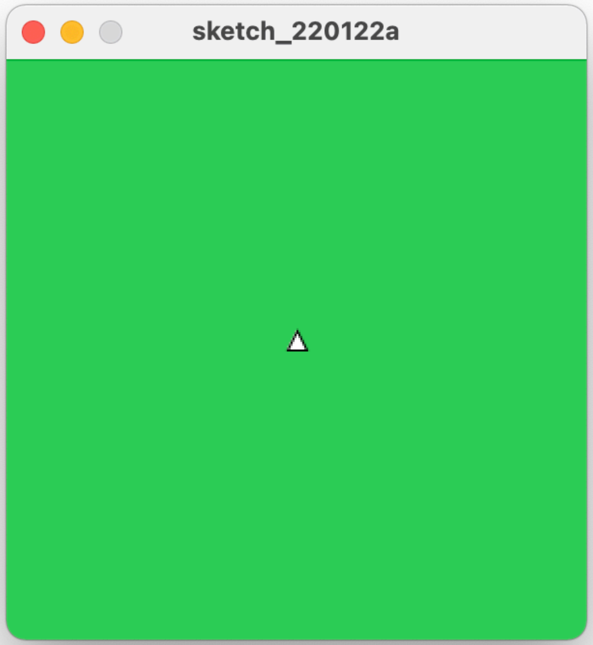
Download and put unzipped folder in Processing Sketchbook folder, default location:
Documents/Processing/Libraries

Basic Turtle Setup

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  t.drawTurtle();
}
```

Create turtle after setting window size.

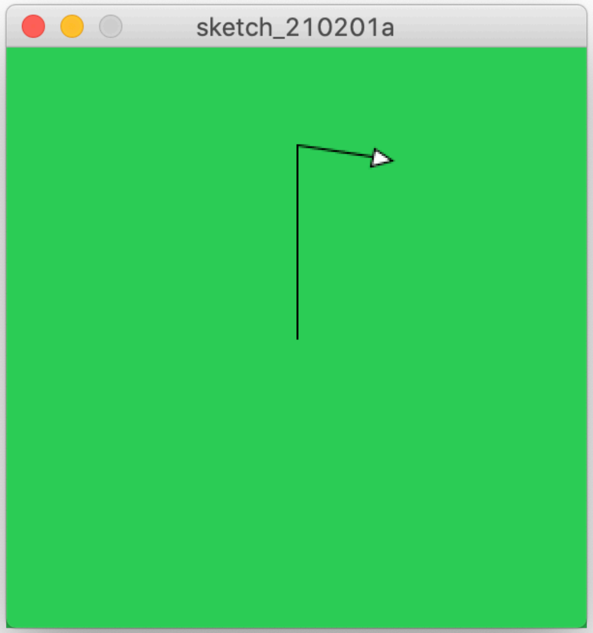
Turtle starts in center of window facing up.



Save your “Sketch”

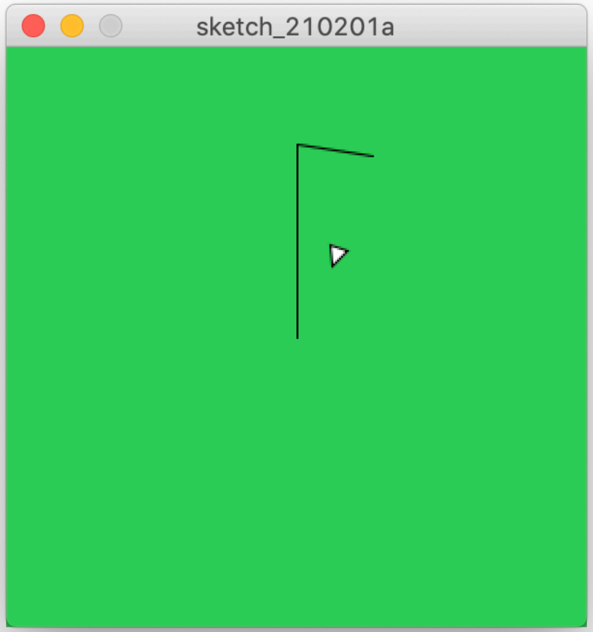
What the turtle can do

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);

  t.forward(100);
  t.right(100);
  t.forward(40);
  t.drawTurtle();
}
```



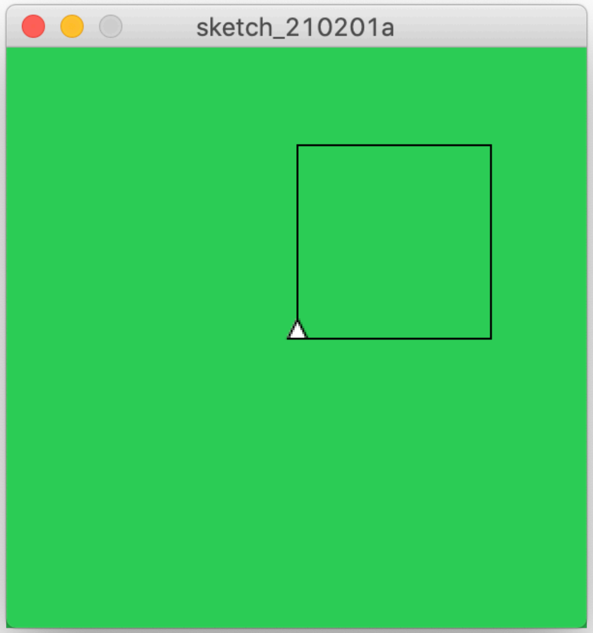
What the turtle can do

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);

  t.forward(100);
  t.right(100);
  t.forward(40);
  t.right(100);
  t.penUp();
  t.forward(50);
  t.drawTurtle();
}
```



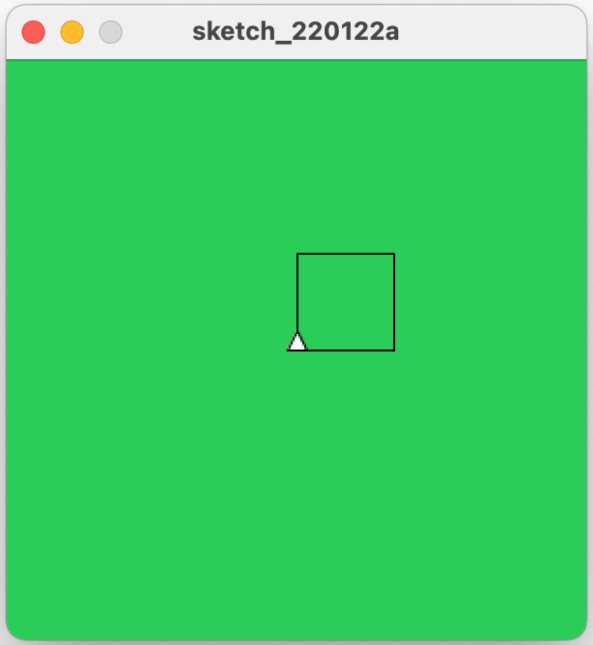
A Square

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  square();
  t.drawTurtle();
}

void square() {
  for (int i=0;i<4;i++) {
    t.forward(100);
    t.right(90);
  }
}
```



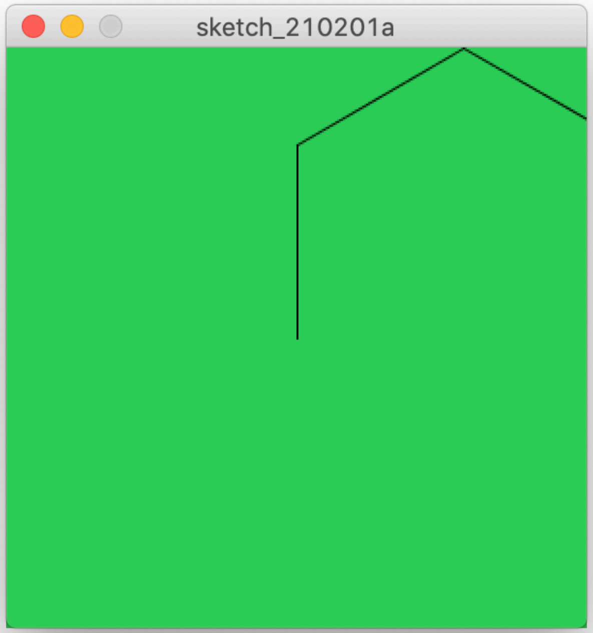
A Square

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  square(50);
  t.drawTurtle();
}

void square(int size) {
  for (int i=0;i<4;i++) {
    t.forward(size);
    t.right(90);
  }
}
```

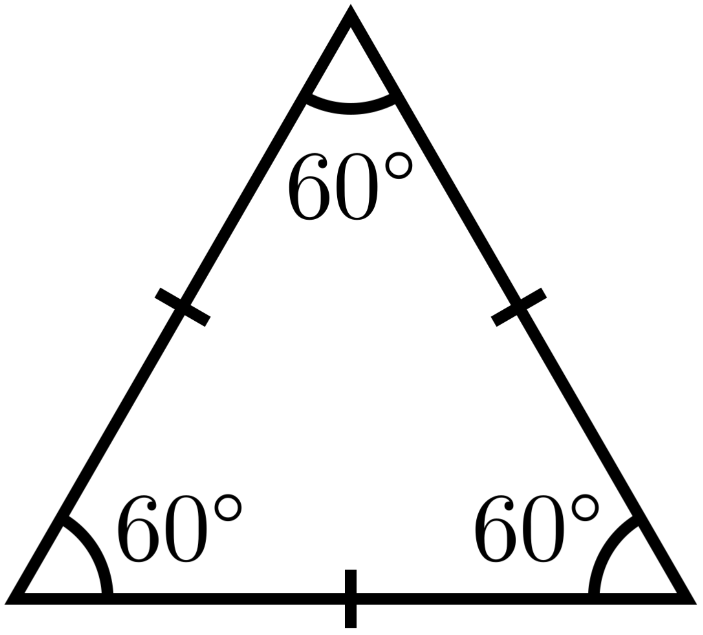


A Triangle?

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  triangle(100);
  t.drawTurtle();
}

void triangle(int size) {
    for (int i=0;i<3;i++) {
    t.forward(size);
    t.right(60);
  }
}
```



A Triangle, Traditional Geometry

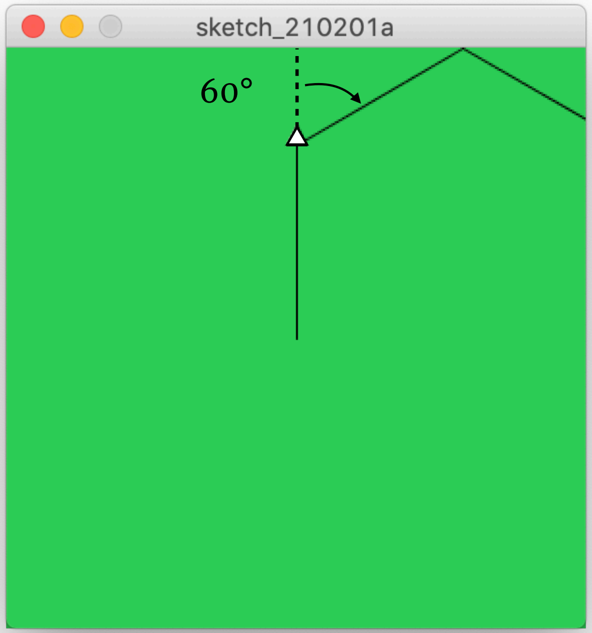


Equilateral Triangle

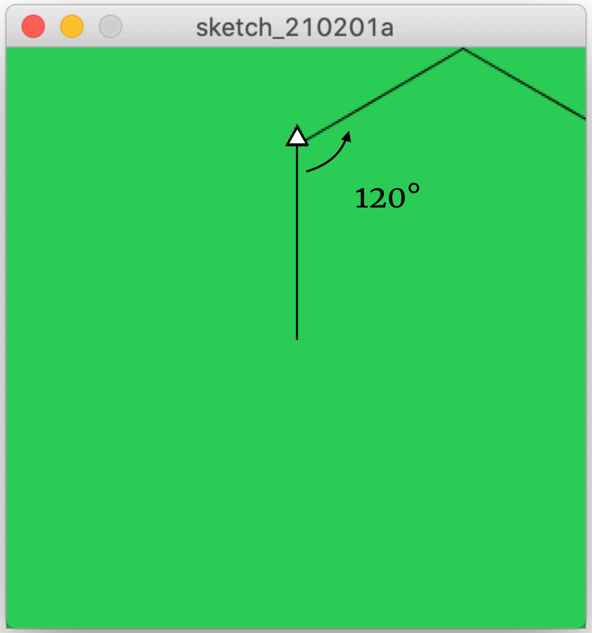
All sides equal length

All internal angles equal to 60°

External vs. Internal Angles



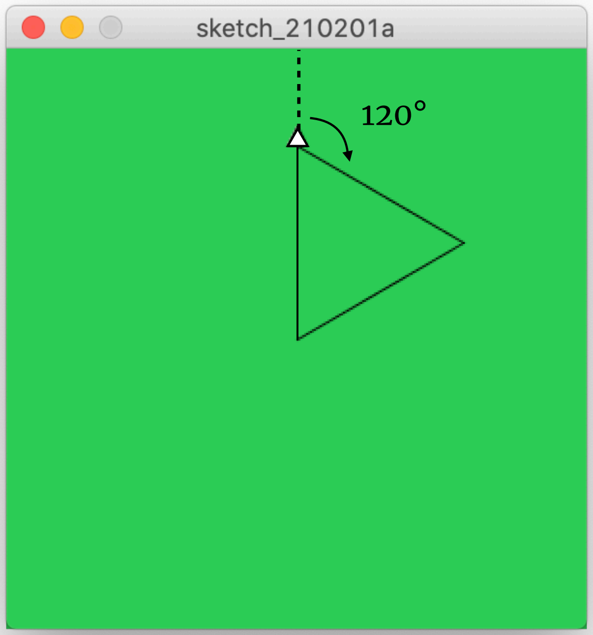
external



internal

A Triangle Using the External (Turtle) Angle

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  triangle(100);
  t.drawTurtle();
}

void triangle(int size) {
    for (int i=0;i<3;i++) {
    t.forward(size);
    t.right(120);
  }
}
```



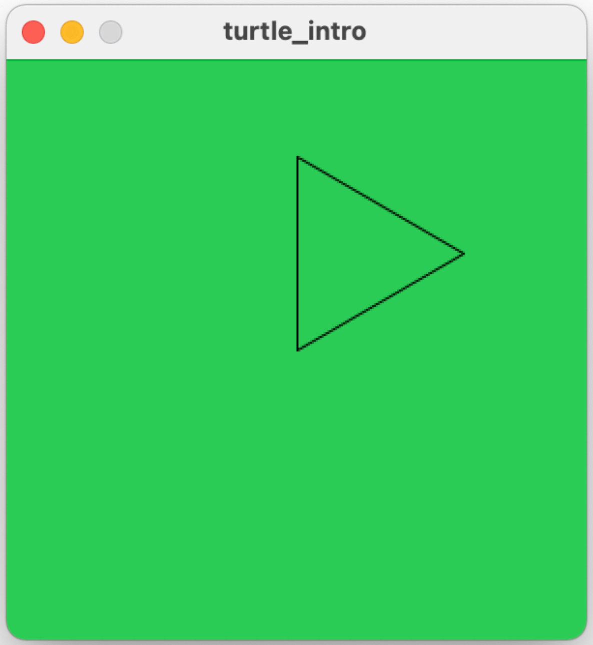
questions?

Add a Draw Method

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
}

void draw() {

}
```

Another Way to Draw a Triangle

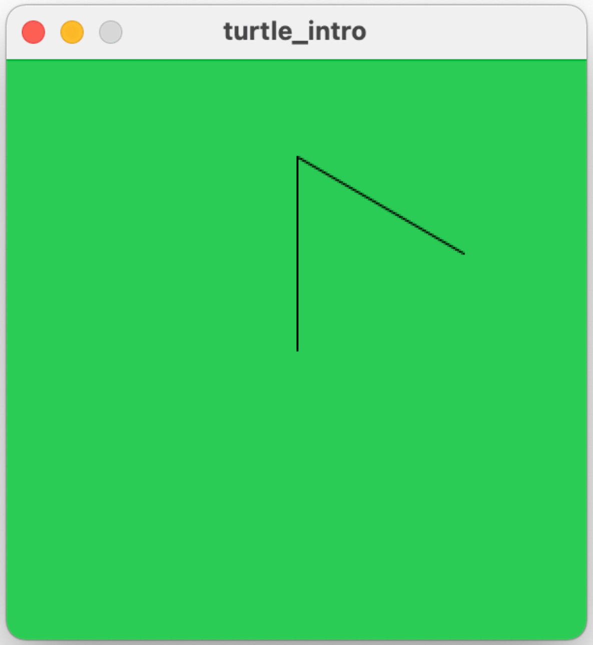
```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
}

void draw() {
  t.forward(100);
  t.right(120);
}
```



Will keep retracing its steps.

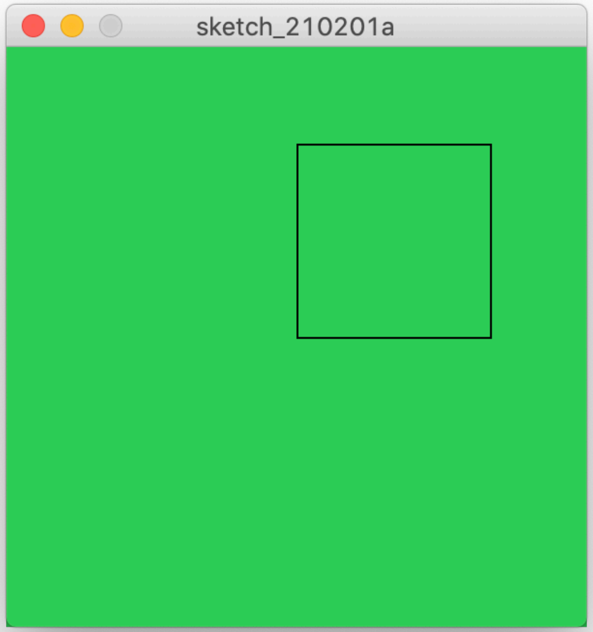
Animating Drawing: Adjust Framerate

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  frameRate(3);
}

void draw() {
  t.forward(100);
  t.right(120);
}
```



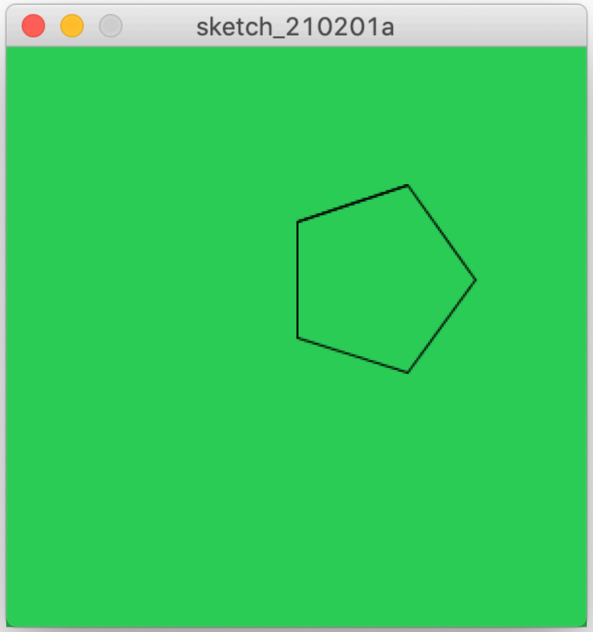
A Square

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle (this);
  frameRate(3);
}

void draw() {
  t.forward(100);
  t.right(90);
}
```



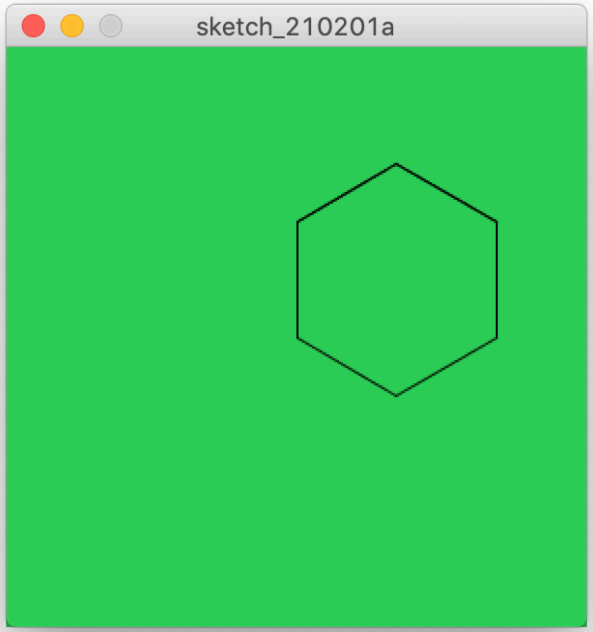
Generalizing: PolygonStep

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(60,72);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



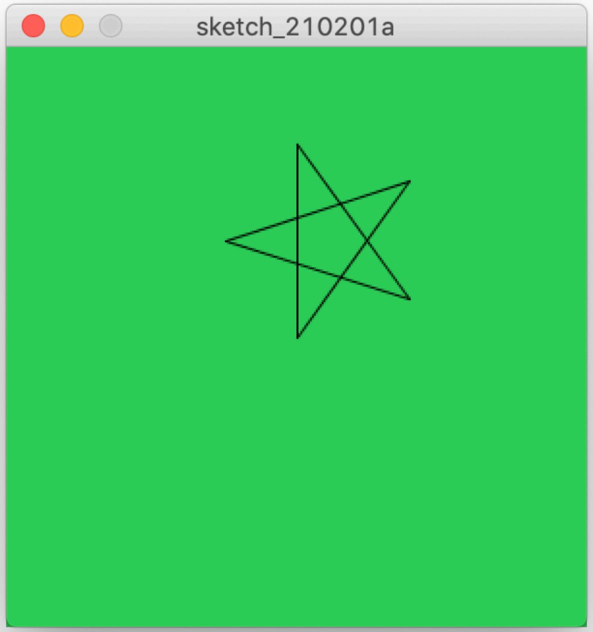
Draw and Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(60,60);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



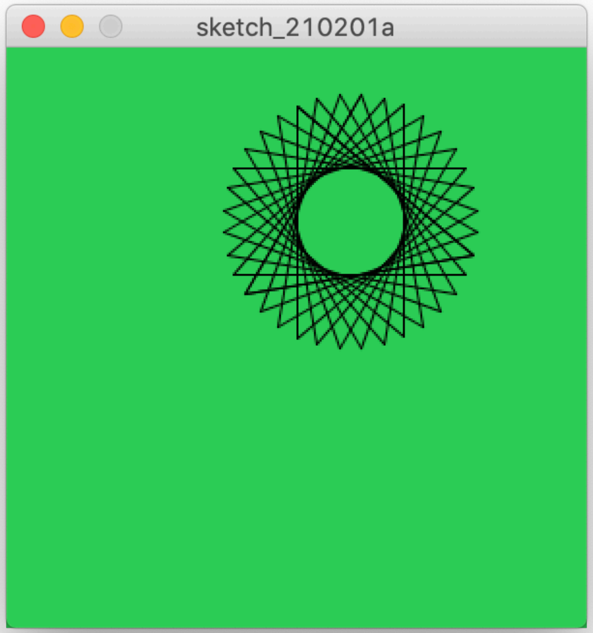
Draw and Polygons

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(100,144);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Draw and Polygons

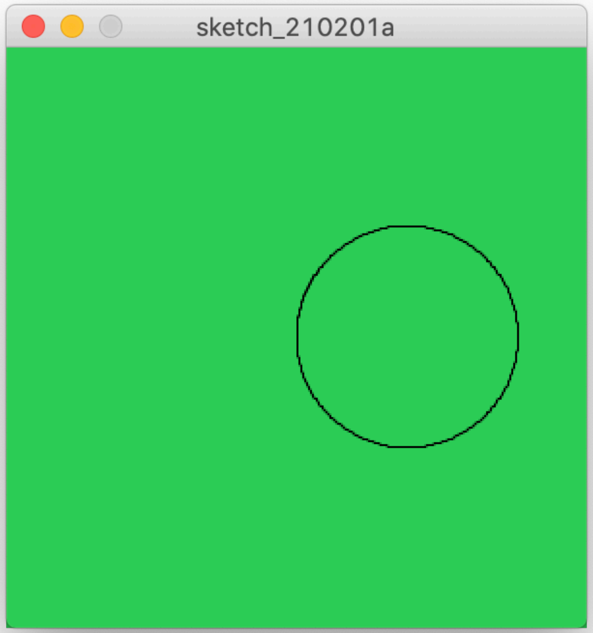
```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(100,130);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



How do you draw a circle?

A Circle

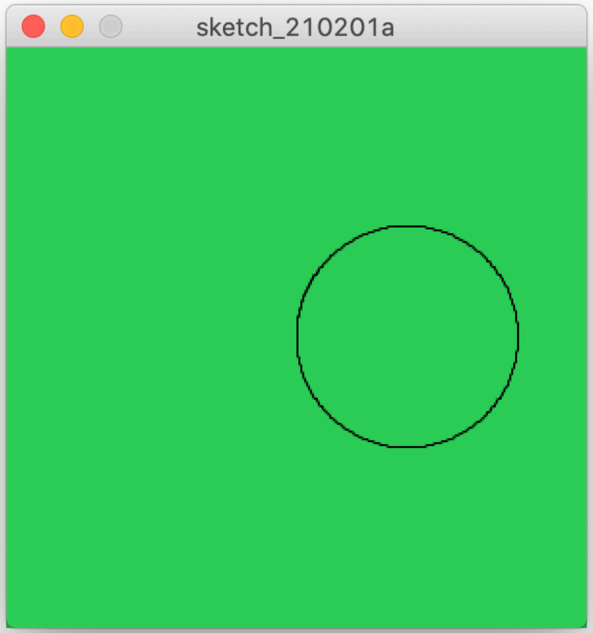
```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(50);
}

void draw() {
  polygonStep(1,1);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Playing with parameters…

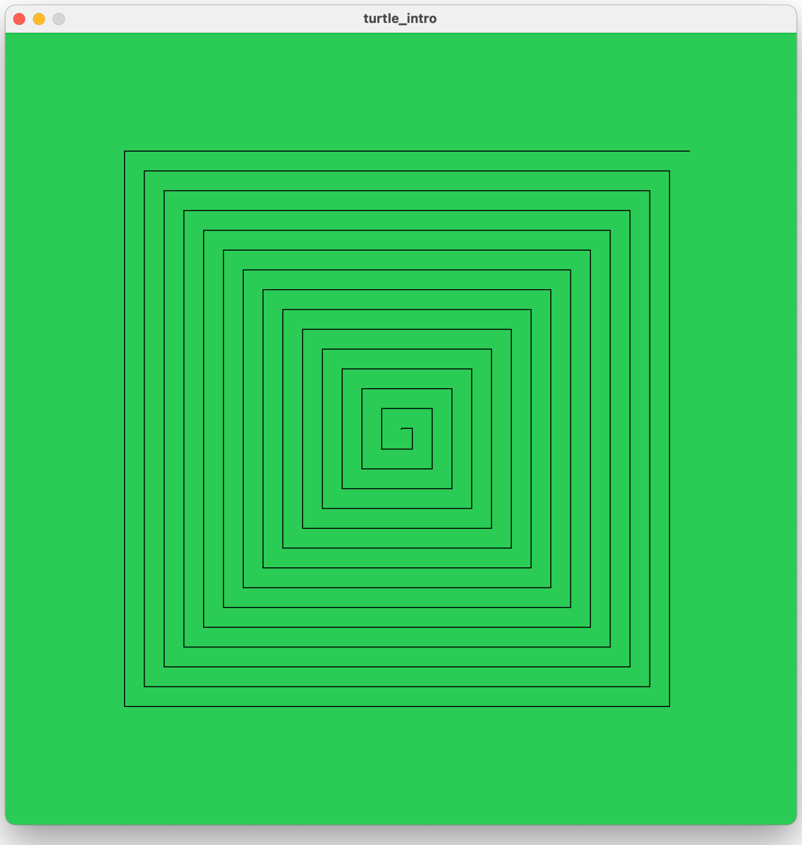
Add Size and Angle Parameters

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 1;
}

void draw() {
  polygonStep(size,angle);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



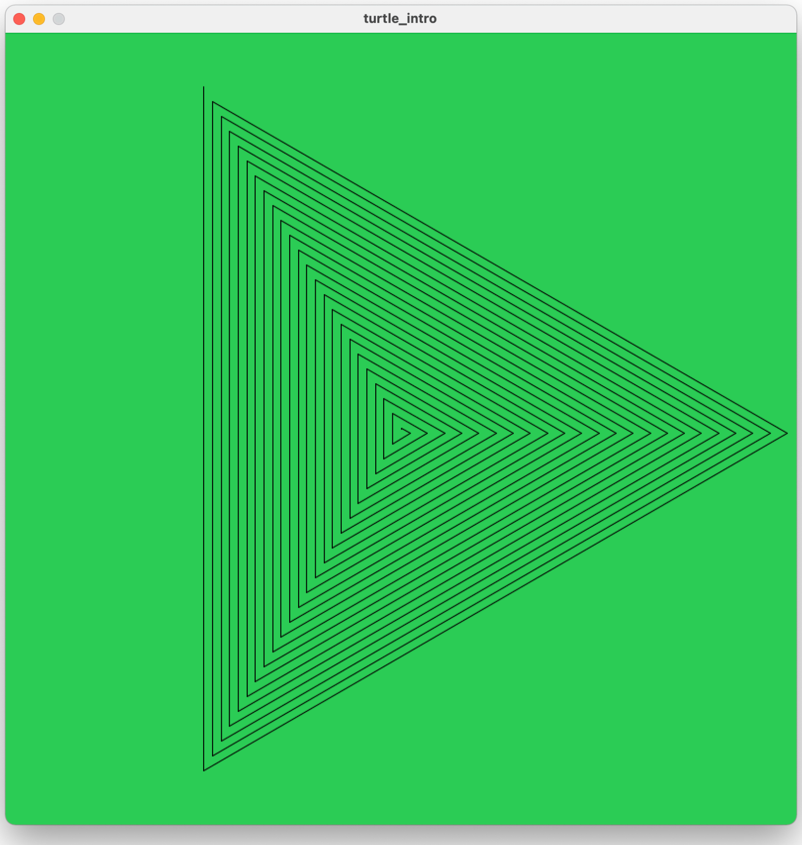
Changing Size in Draw

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 90;
}

void draw() {
  polygonStep(size,angle);
  size = size + 10;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



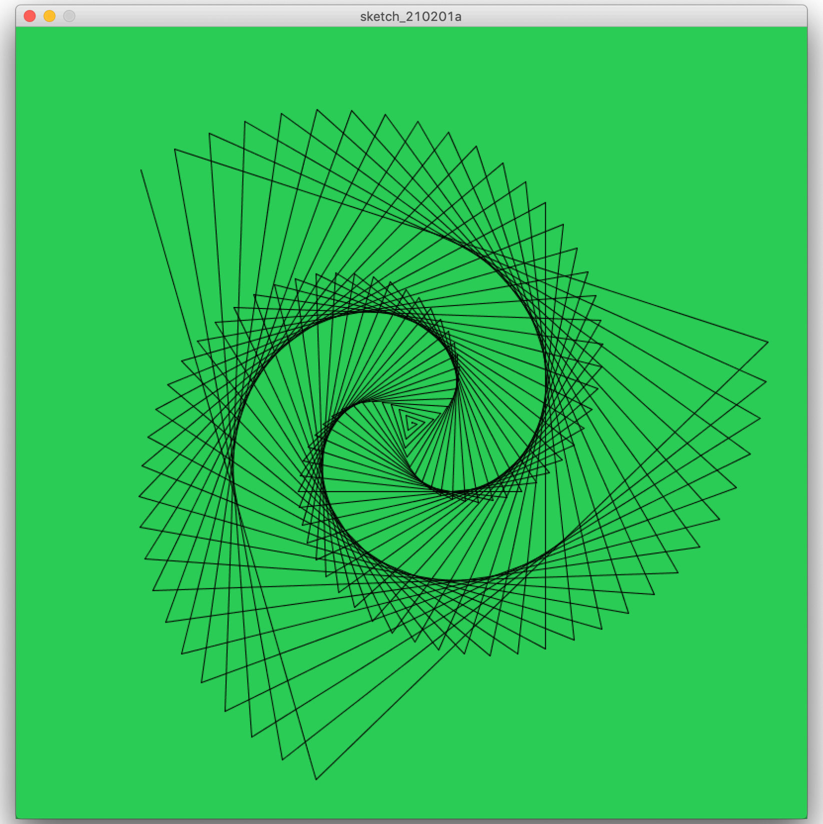
Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 120;
}

void draw() {
  polygonStep(size,angle);
  size = size + 10;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



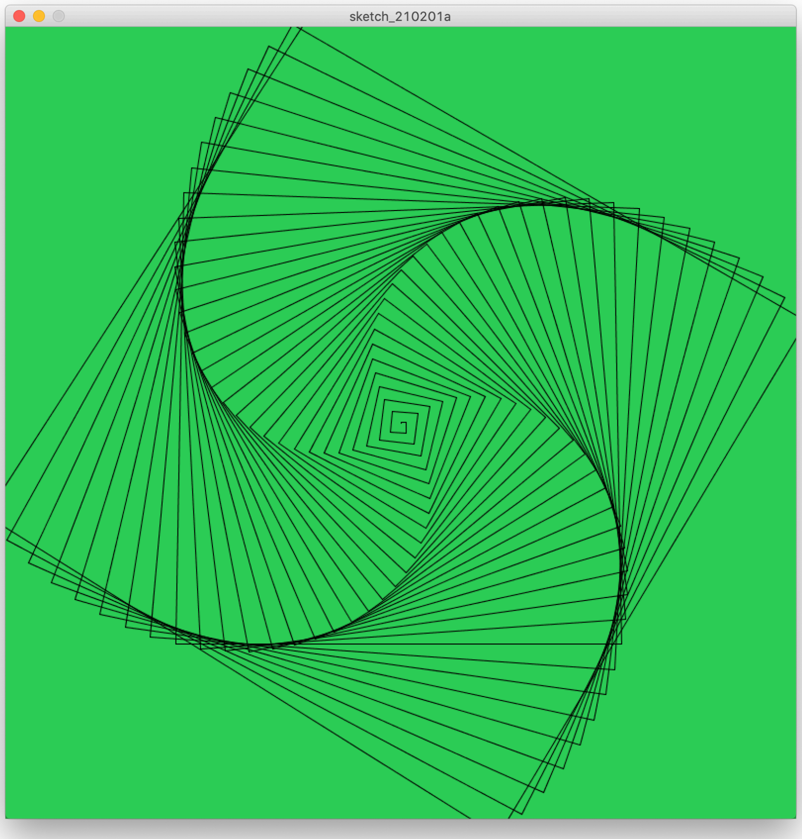
Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 118;
}

void draw() {
  polygonStep(size,angle);
  size = size + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



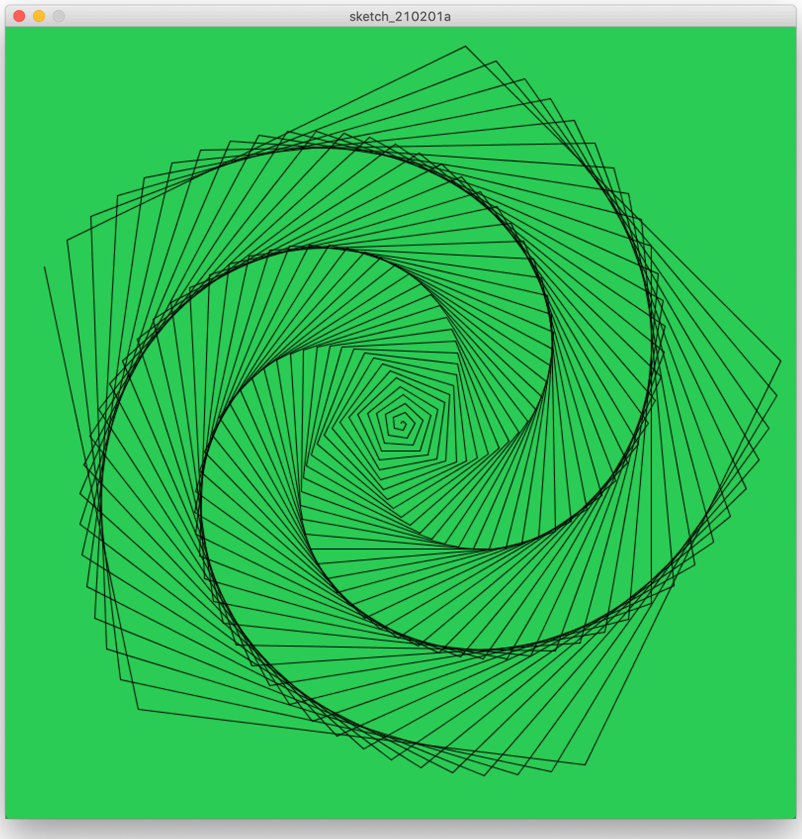
Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 91;
}

void draw() {
  polygonStep(size,angle);
  size = size + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



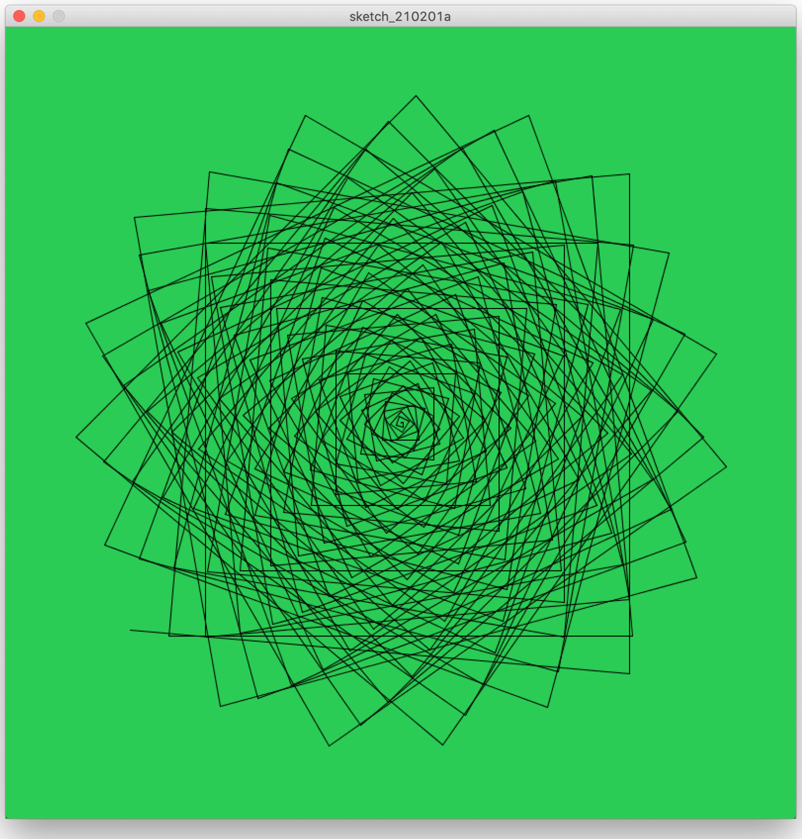
Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 73;
}

void draw() {
  polygonStep(size,angle);
  size = size + 2;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Changing Size

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(10);
  size = 1;
  angle = 95;
}

void draw() {
  polygonStep(size,angle);
  size = size + 2;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```

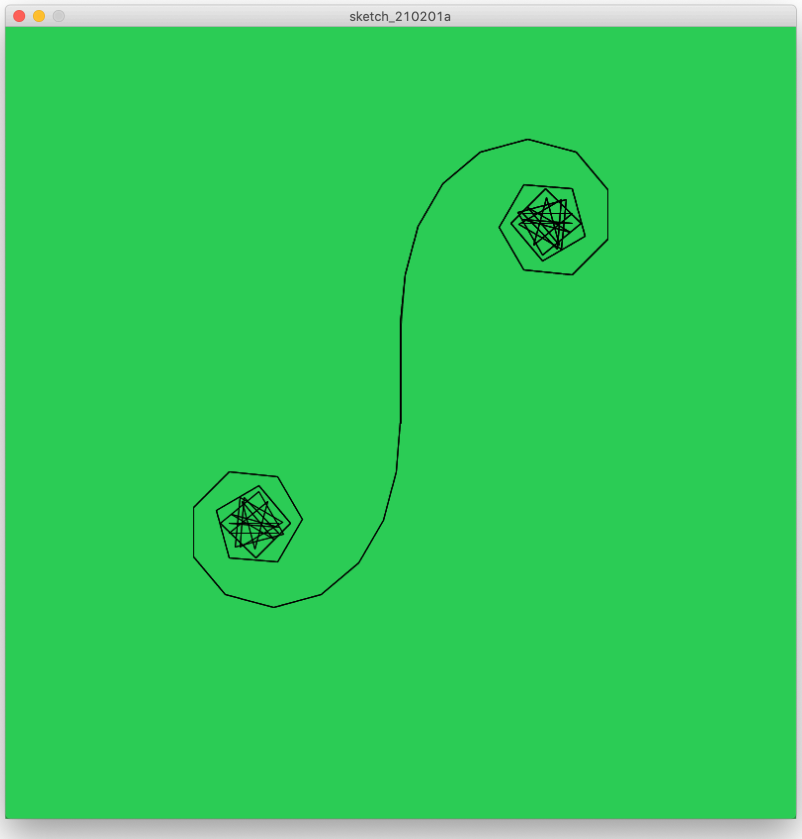


questions?

Playing with parameters…

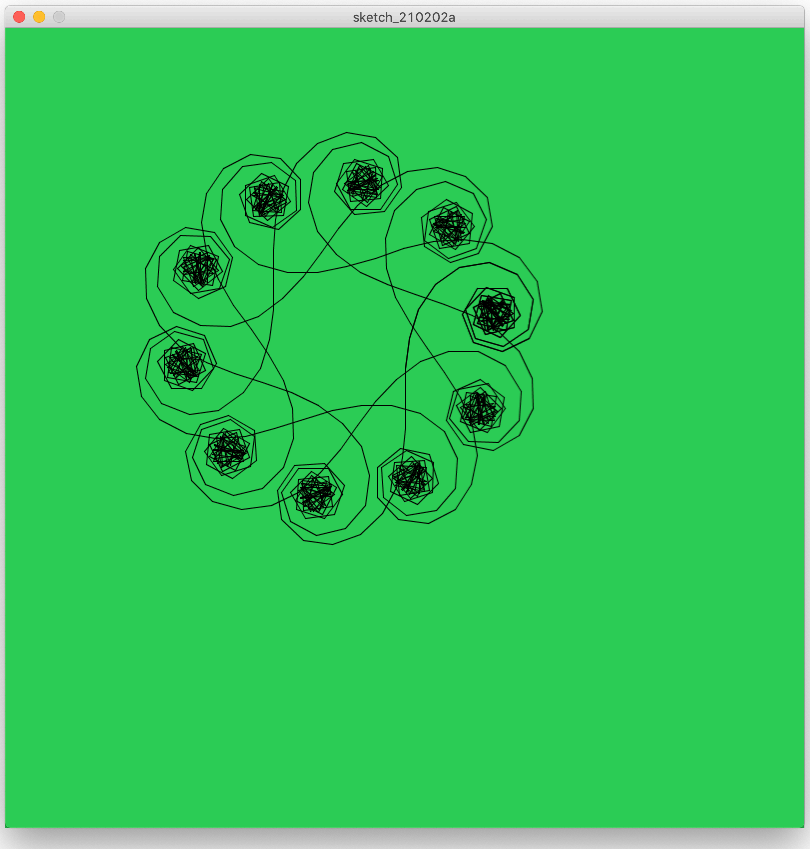
Changing Angle (Increment)

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 50;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



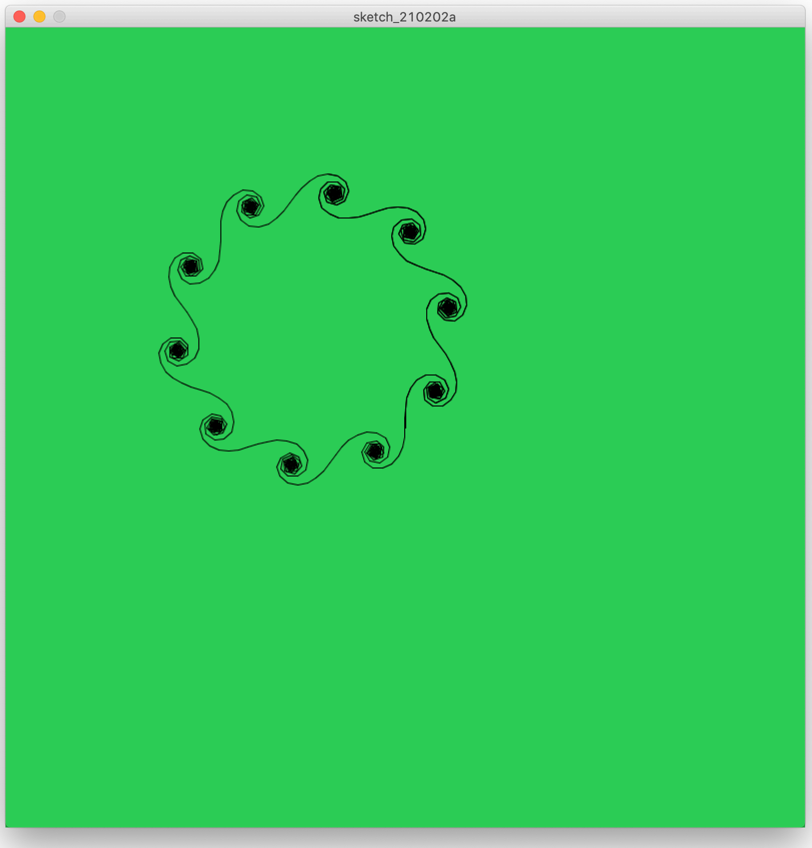
Changing (Initial) Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 30;
  angle = 1;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



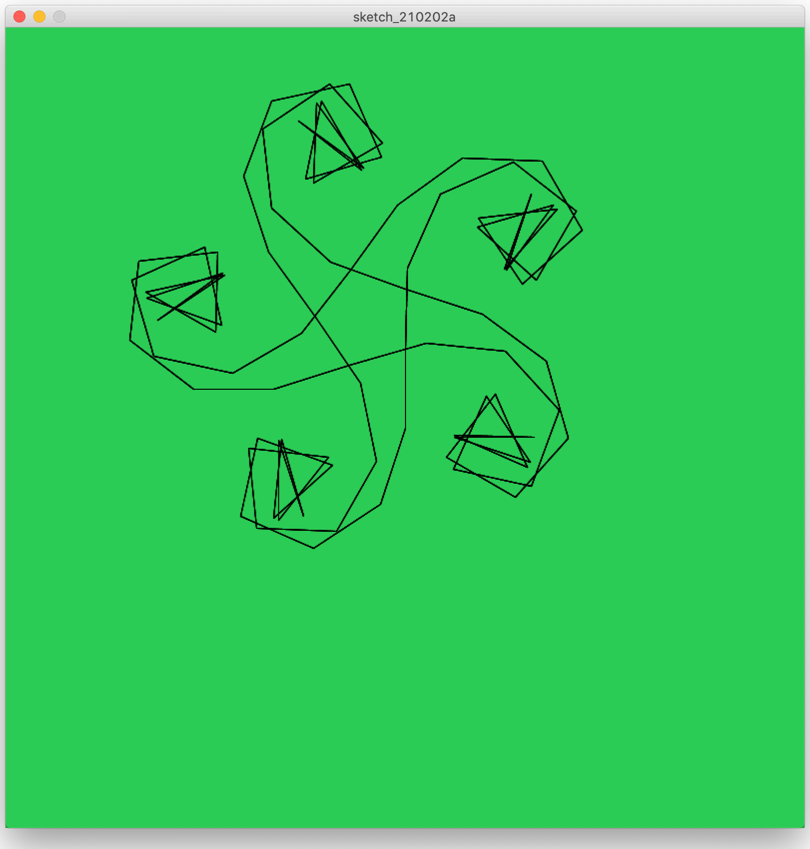
Changing (Initial) Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 10;
  angle = 2;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 5;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



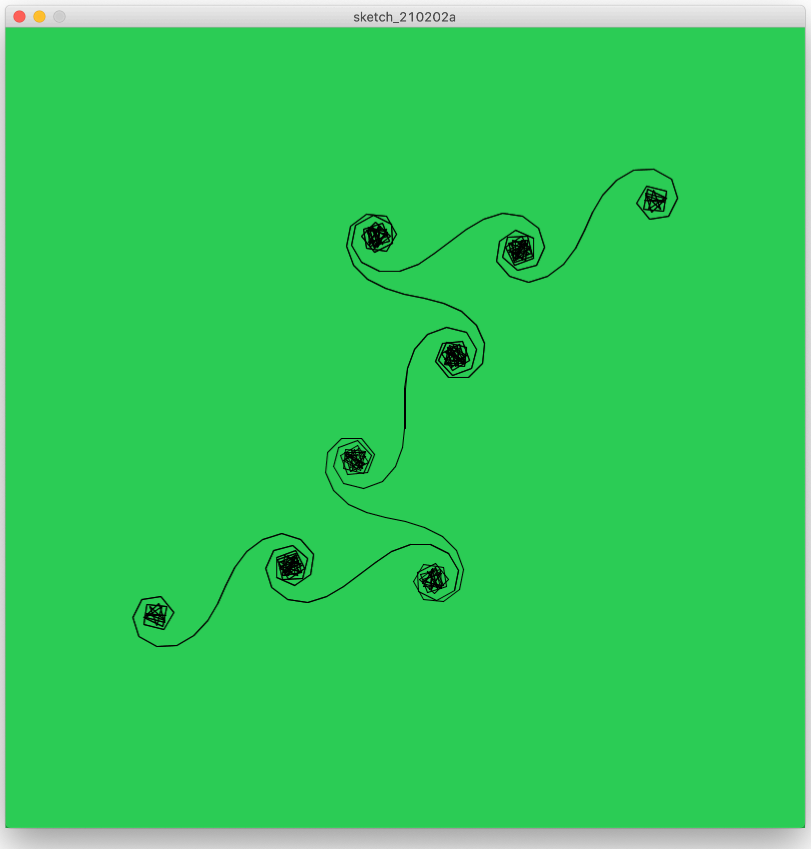
Changing Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 80;
  angle = 2;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 20;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



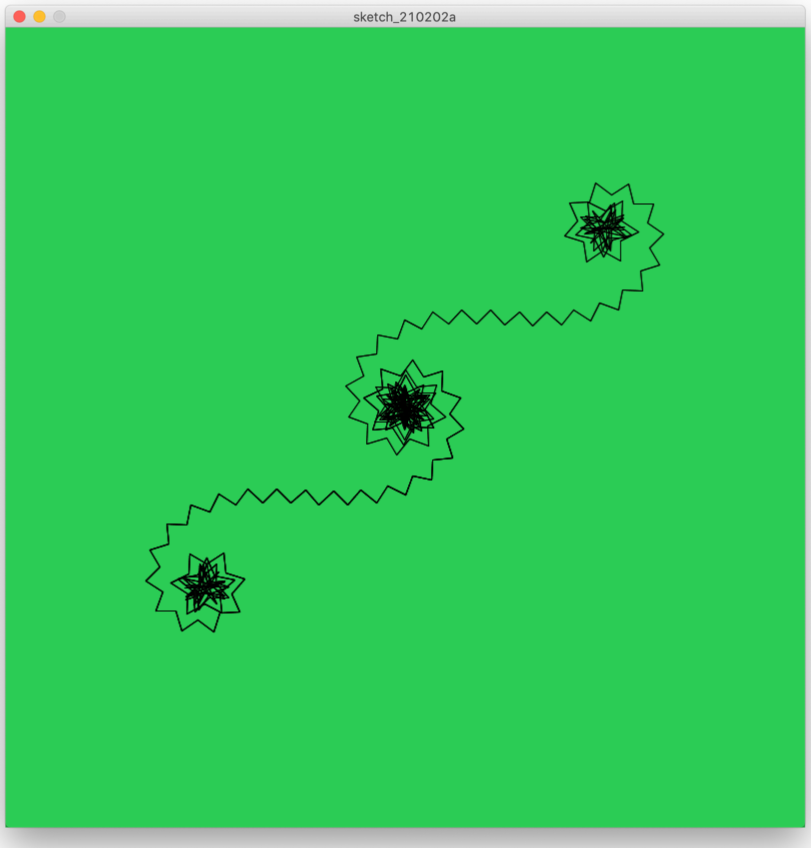
Changing Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 20;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 7;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



Changing Angle

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  size = 20;
  angle = 0;
}

void draw() {
  polygonStep(size,angle);
  angle= angle + 179;
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```



questions?

All of these programs run forever.

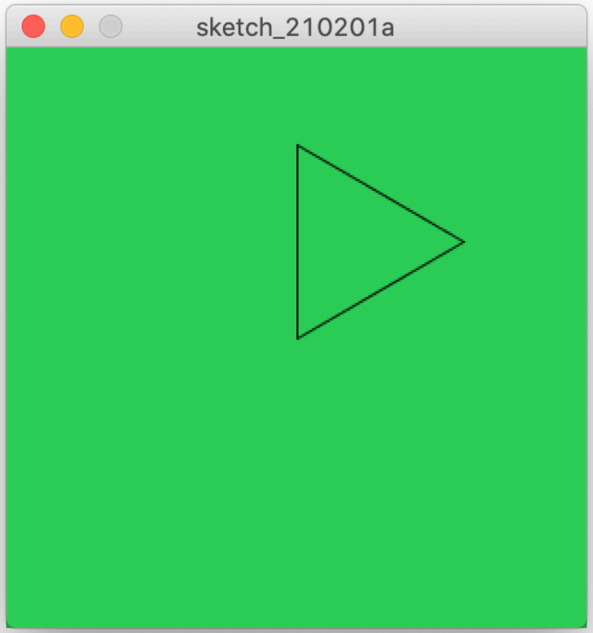
Let’s write a polygon function that halts

All the turtle knows is its current heading.

How can the turtle know when to stop?

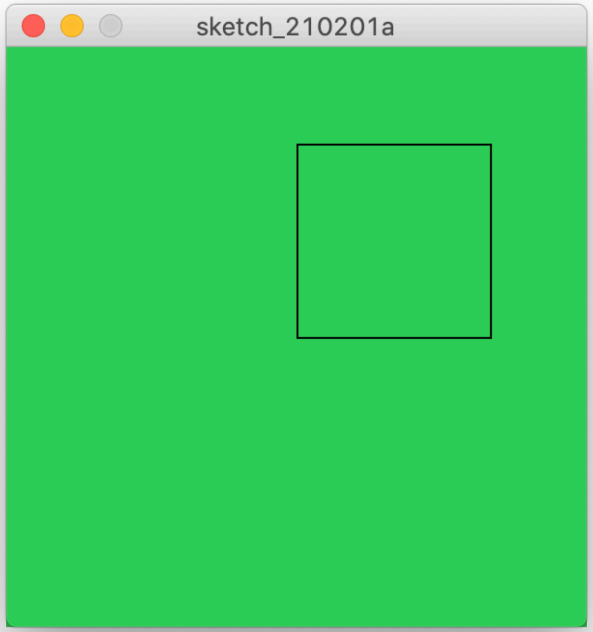
Halting Polygon Attempt 1

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 120;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int sides = 360/angle;
  for (int i=0; i<sides; i++) {
    t.forward(size);
    t.right(angle);
  }
}
```



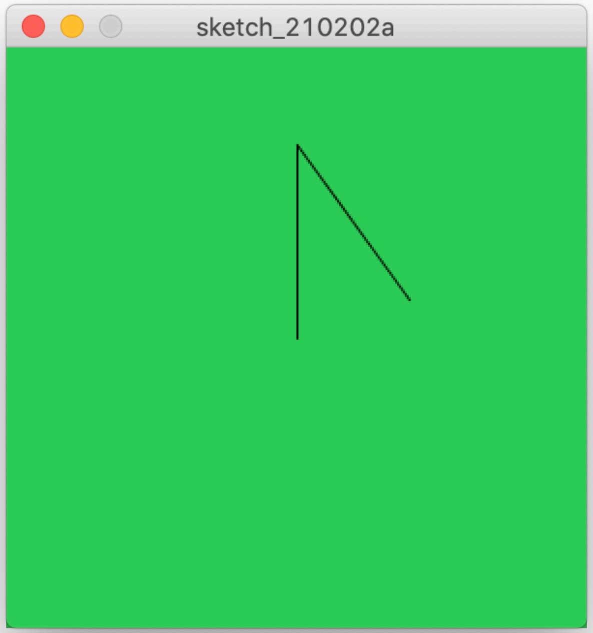
Halting Polygon Attempt 1

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 90;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int sides = 360/angle;
  for (int i=0; i<sides; i++) {
    t.forward(size);
    t.right(angle);
  }
}
```



A Case Where This Fails :(

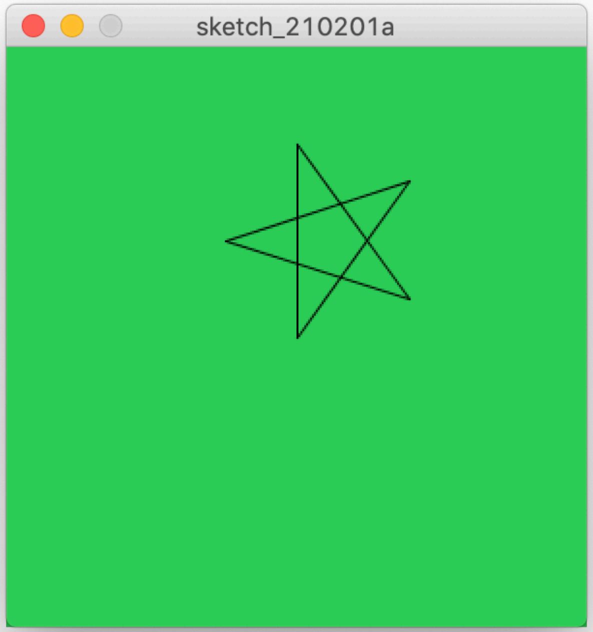
```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 144;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int sides = 360/angle;
  for (int i=0; i<sides; i++) {
    t.forward(size);
    t.right(angle);
  }
}
```



We want to preserve the original behavior;
we want to produce a closed turtle path.

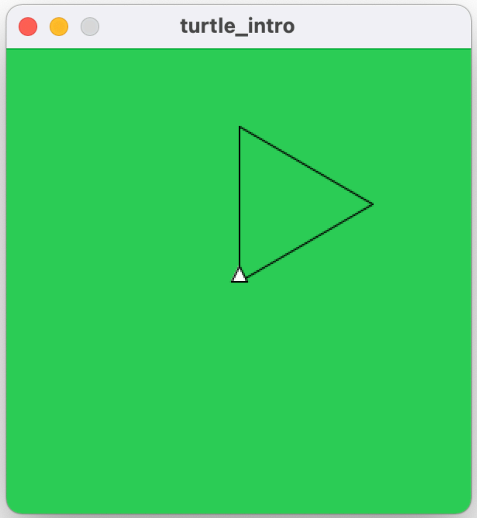
Original Behavior for Angle = 144

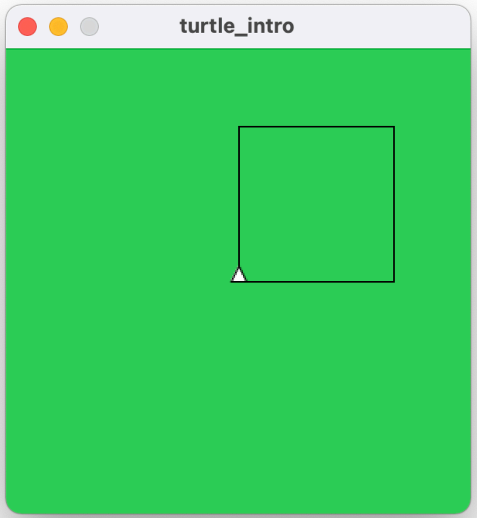
```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(100,200,100);
  t = new Turtle(this);
  frameRate(3);
}

void draw() {
  polygonStep(100,144);
}

void polygonStep(int size, int angle) {
    t.forward(size);
    t.right(angle);
}
```

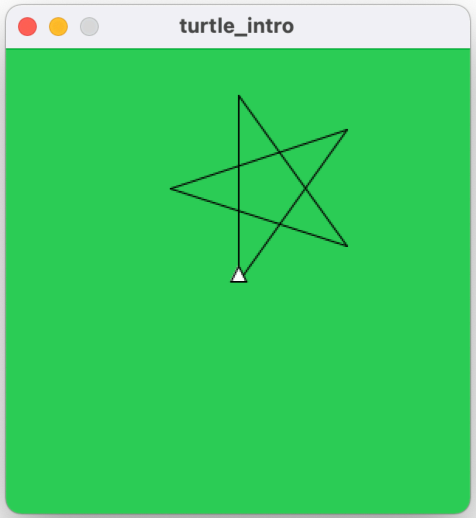


How?

Where should the turtle be at the end of polygonHalt? What should its heading be?







Exactly where it started!
0° (360°)

Halting Polygon Attempt 2

```
void polygonHalt(int size, int angle) {
  int rotation = 0;
  t.forward(size);
  t.right(angle);
  rotation = rotation + angle;
  while( ?? ) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
}
```

Add a variable **rotation** to keep track of the turtle's heading

When should the loop end?

Halting Polygon Attempt 2

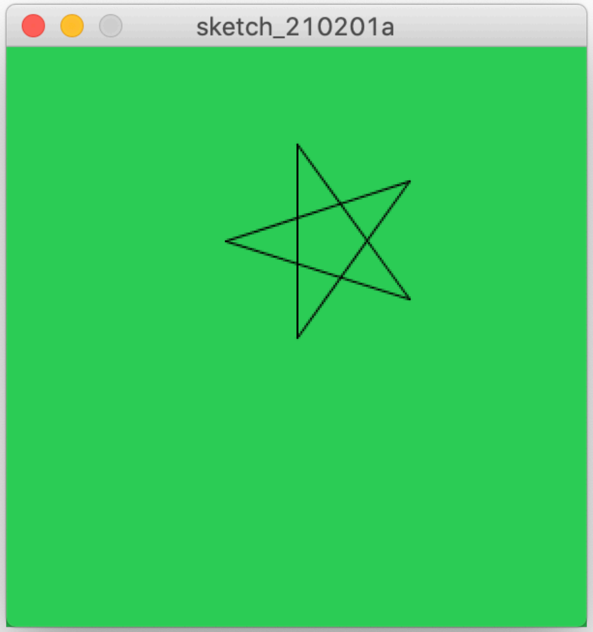
```
void polygonHalt(int size, int angle) {
  int rotation = 0;
  t.forward(size);
  t.right(angle);
  rotation = rotation + angle;
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
}
```

Loop until the turtle is back where it started

End when **rotation** is back to 0, back to 360

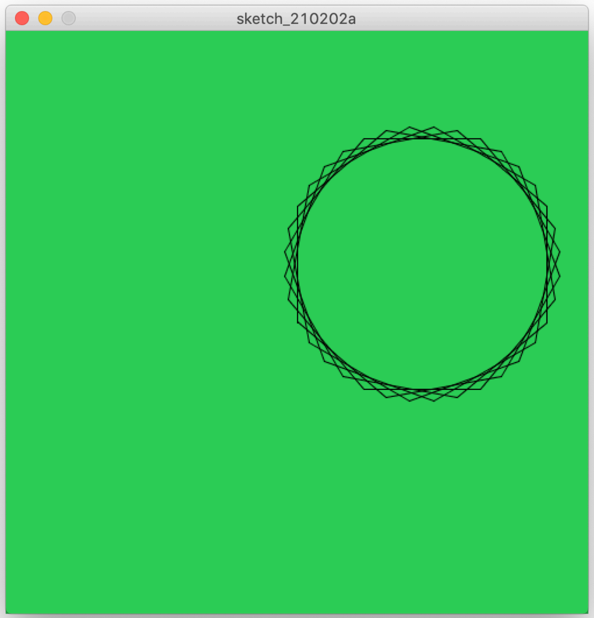
Halting Polygon Attempt 2

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 144;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int rotation = 0;
  t.forward(size);
  t.right(angle);
  rotation = rotation + angle;
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
}
```



Halting Polygon Attempt 2

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 50;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int rotation = 0;
  t.forward(size);
  t.right(angle);
  rotation = rotation + angle;
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
}
```



questions?

Let’s formalize this understanding

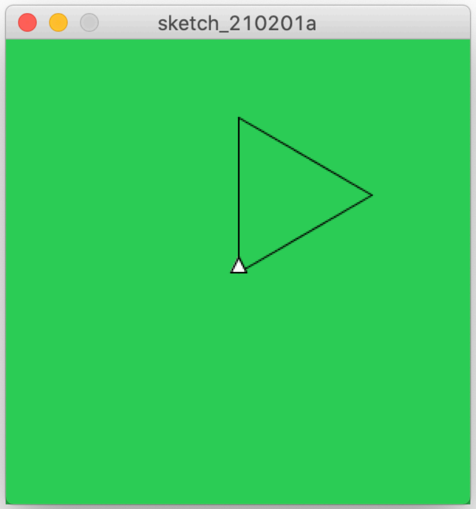
Closed-Path Theorm

The total rotation along any closed path is an integer multiple of 360°

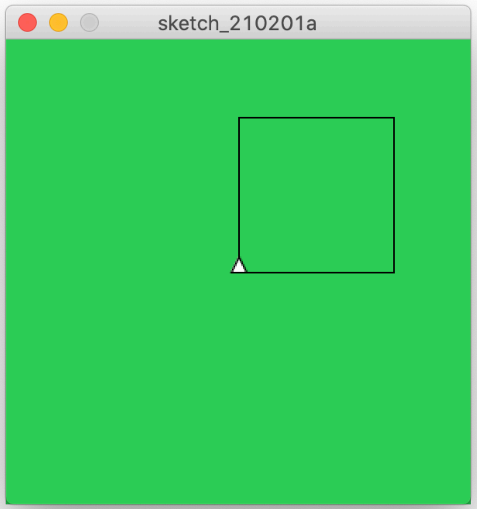
Definition of “closed”

Turtle returns to the same position **and** same heading

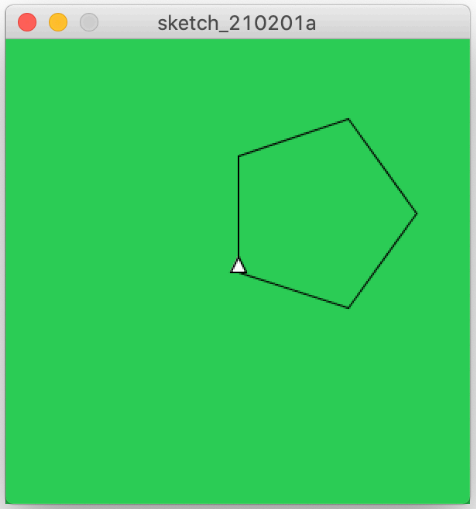
Number of Rotations = 1



120*3 = 360

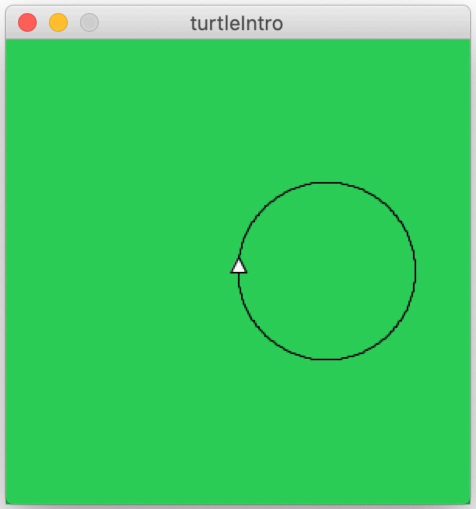


90*4 = 360



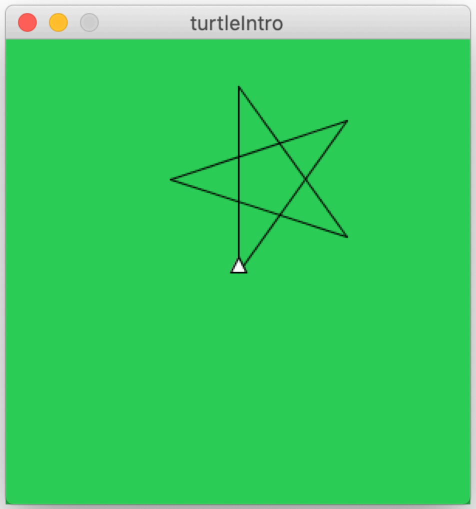
72*5 = 360

Rotations = 1



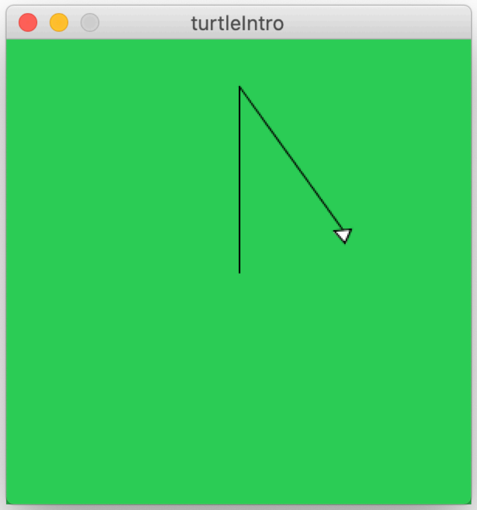
360*1 = 360

Rotations = ?

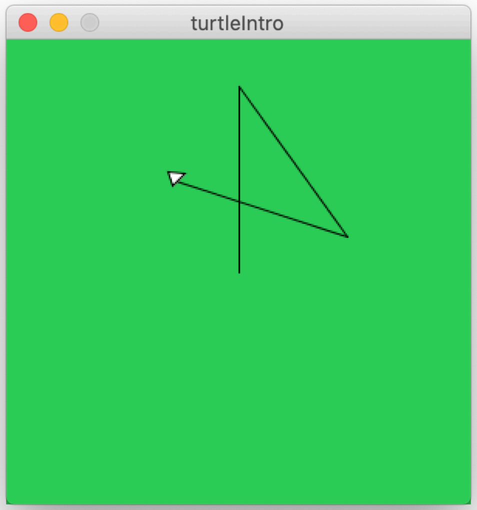


angle = 144

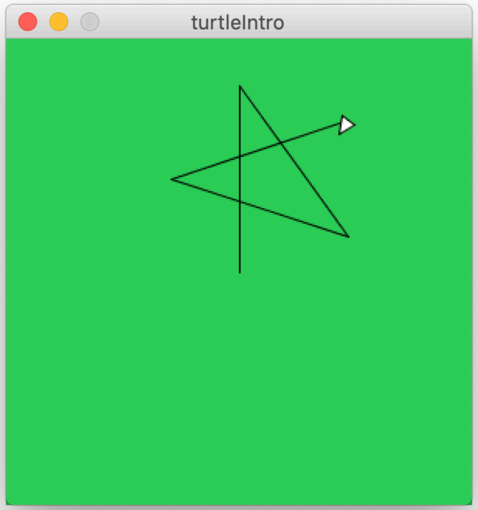
Rotations = ?



144*2 = 288

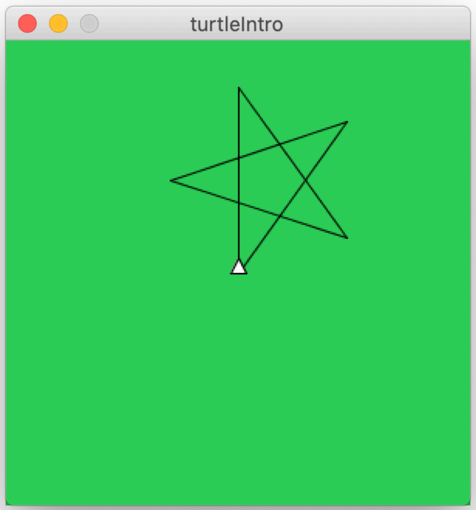


144*3 = 432



144*4 = 576

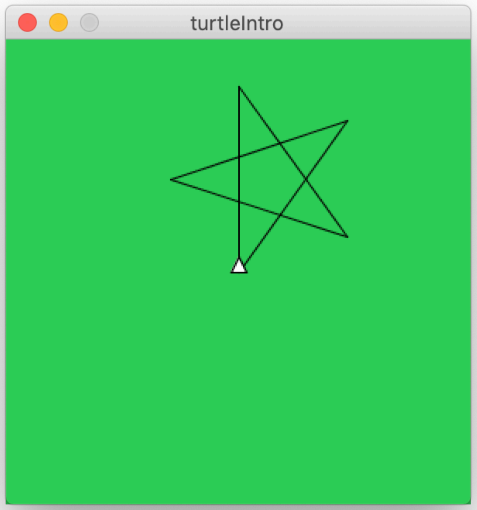
Rotations = ?



144*5 = 720

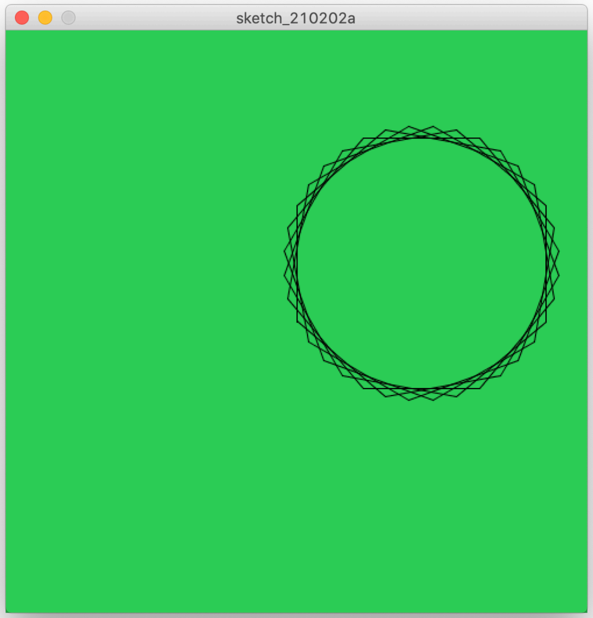
Rotations = 2

```
import Turtle.*;
Turtle t;
int size, angle;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
  noLoop();
  size = 100;
  angle = 50;
}

void draw() {
  polygonHalt(size, angle);
}

void polygonHalt(int size, int angle) {
  int rotation = angle;
  t.forward(size);
  t.right(angle);
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
  println("Total rotation = " +rotation);
  println("Number of rotations = " +rotation/360);
}
```



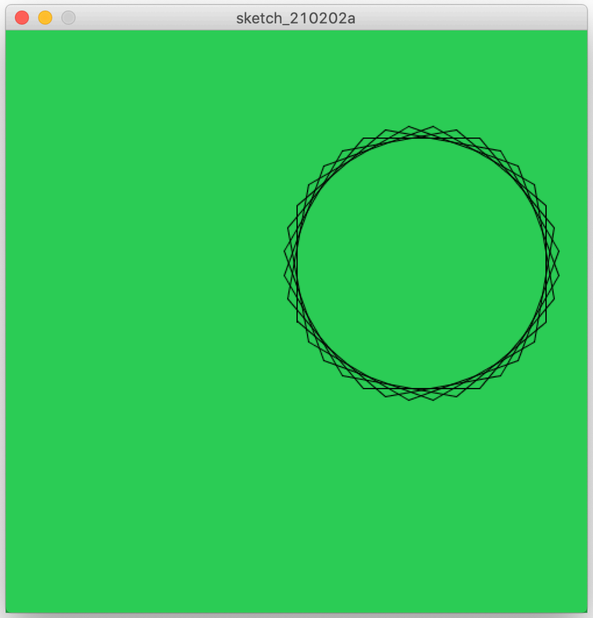
144*5 = 720
rotations = 720/360 = 2

Rotations = ?



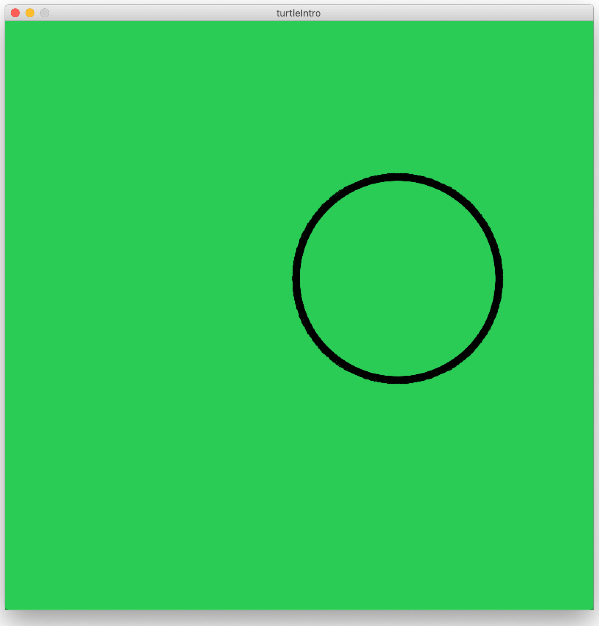
angle = 50

Rotations = 5



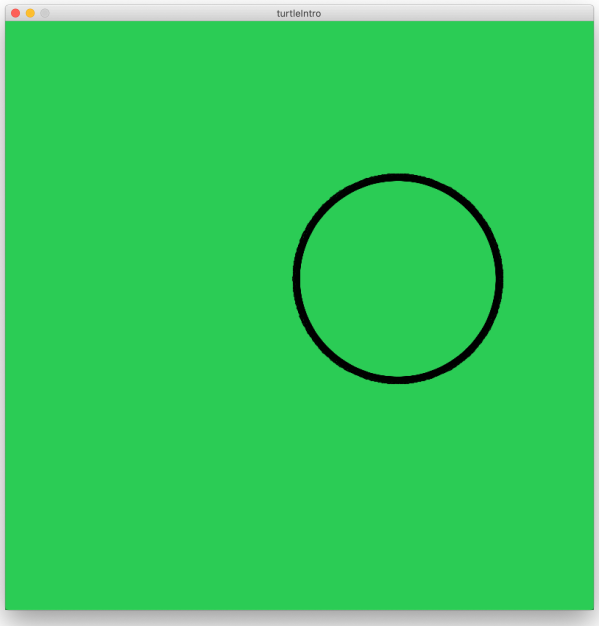
angle = 50
50 * 36 = 1800
rotations = 1800/360 = 5

Rotations = ?



angle = 41

Rotations = 41



angle = 41
rotation = 41 * 360 = 14760
14760/360 = 41

Closed-Path Theorm

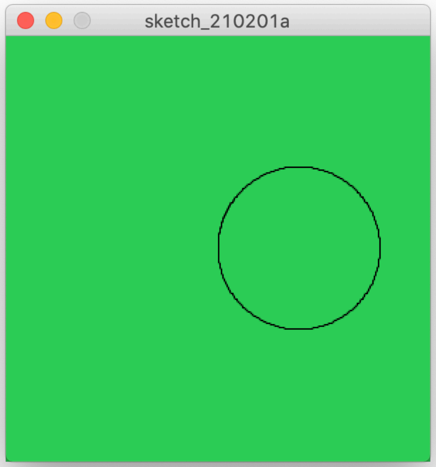
The total rotation along any closed path is an integer multiple of 360°

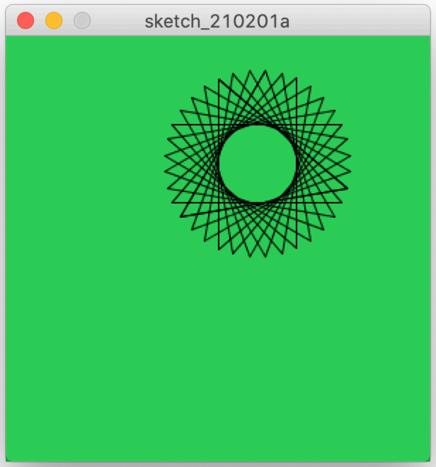
PolygonHalt Closing Theorem

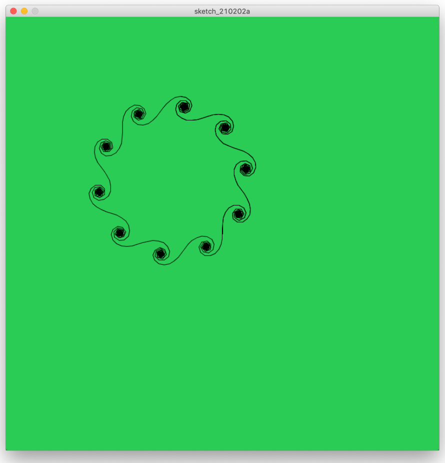
A path drawn by the polygonHalt procedure will close precisely when the **rotation** reaches a multiple of 360°

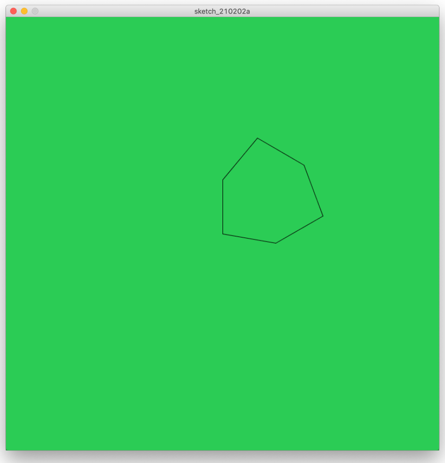
exception: angle = 0°, 360°, 720°, etc.

More examples of closed Polygons







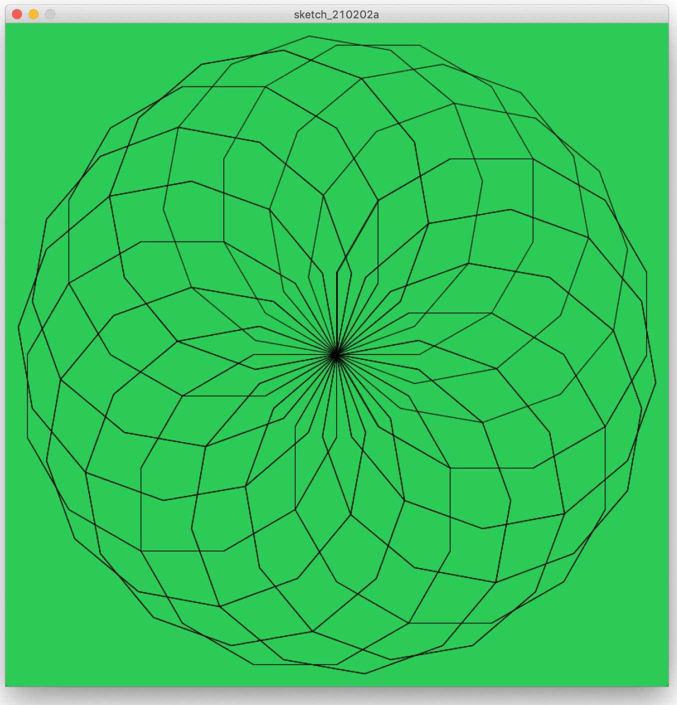


questions?

Playing…

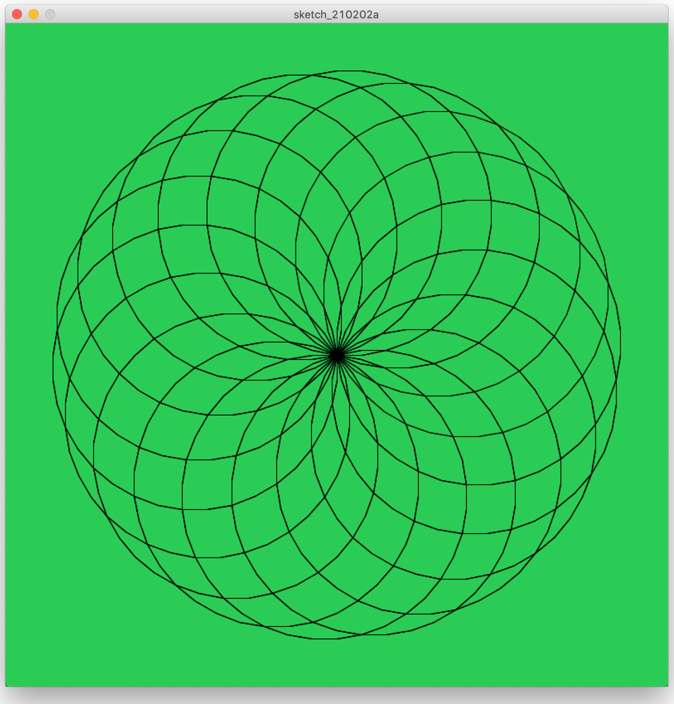
Play with Draw

```
import Turtle.*;
Turtle t;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
}

void draw() {
  polygonHalt(100, 30);
  t.right(20);
}

void polygonHalt(int size, int angle) {
  int rotation = angle;
  t.forward(size);
  t.right(angle);
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
  println("Total rotation = " +rotation);
  println("Number of rotations = " +rotation/360);
}
```



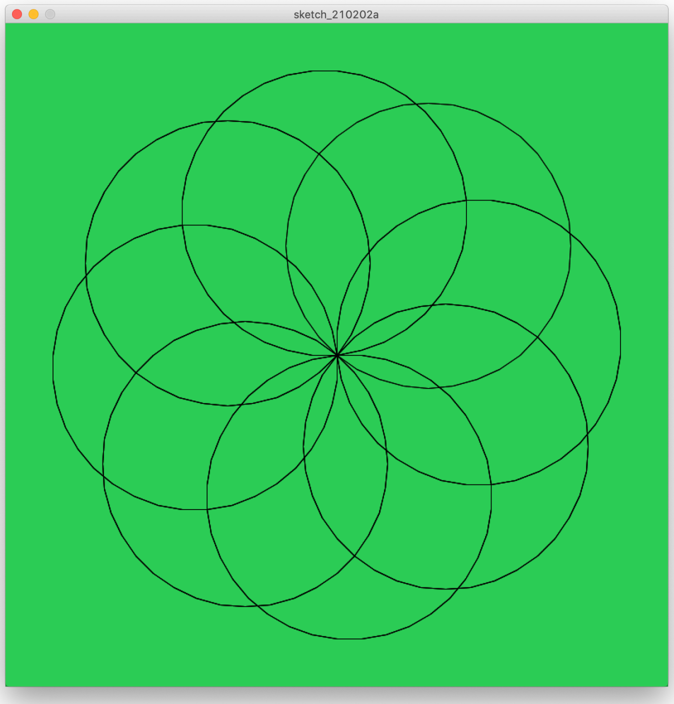
Play with Draw

```
import Turtle.*;
Turtle t;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
}

void draw() {
  polygonHalt(30, 10);
  t.right(20);
}

void polygonHalt(int size, int angle) {
  int rotation = angle;
  t.forward(size);
  t.right(angle);
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
  println("Total rotation = " +rotation);
  println("Number of rotations = " +rotation/360);
}
```



Play with Draw

```
import Turtle.*;
Turtle t;

void setup() {
  size(800,800);
  background(100,200,100);
  t = new Turtle(this);
}

void draw() {
  polygonHalt(30, 10);
  t.right(45);
}

void polygonHalt(int size, int angle) {
  int rotation = angle;
  t.forward(size);
  t.right(angle);
  while(rotation%360 > 0) {
    t.forward(size);
    t.right(angle);
    rotation = rotation + angle;
  }
  println("Total rotation = " +rotation);
  println("Number of rotations = " +rotation/360);
}
```



questions?

Turtle Geometry Chapter 2 Mini Intro

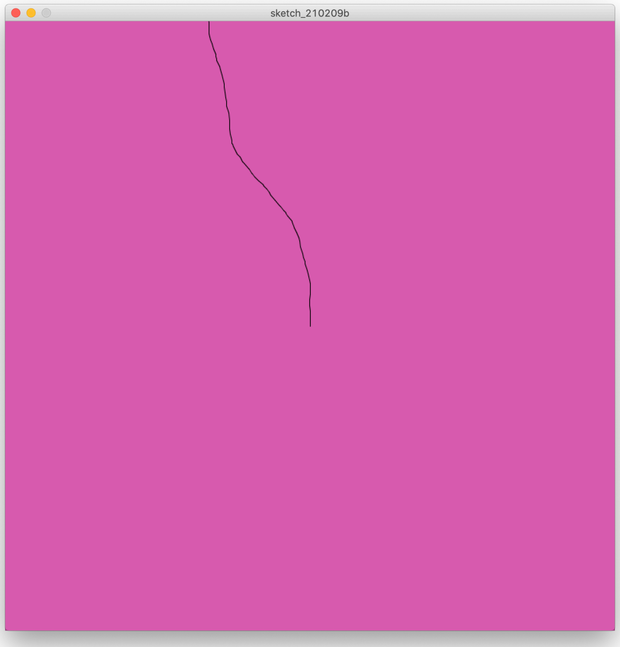
Create a new program

Basic Turtle Setup

```
import Turtle.*;
Turtle t;

void setup() {
  size(300,300);
  background(200,100,170);
  t = new Turtle(this);
}
```

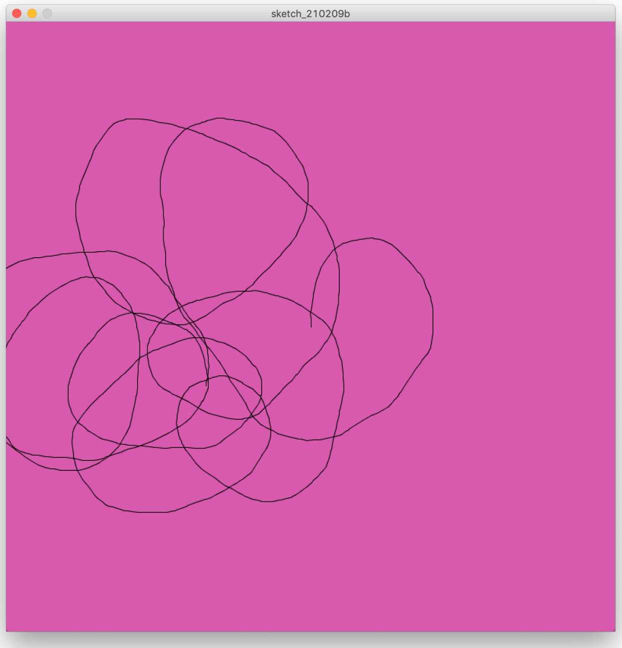
Turtle as Creature

```
import Turtle.*;
Turtle t;

void setup() {
  size(800,800);
  background(200,100,170);
  t = new Turtle(this);
  frameRate(10);
}

void draw() {
  randomMove();
}

void randomMove() {
  t.forward(random(1,10));
  t.right(random(-10,10));
}
```



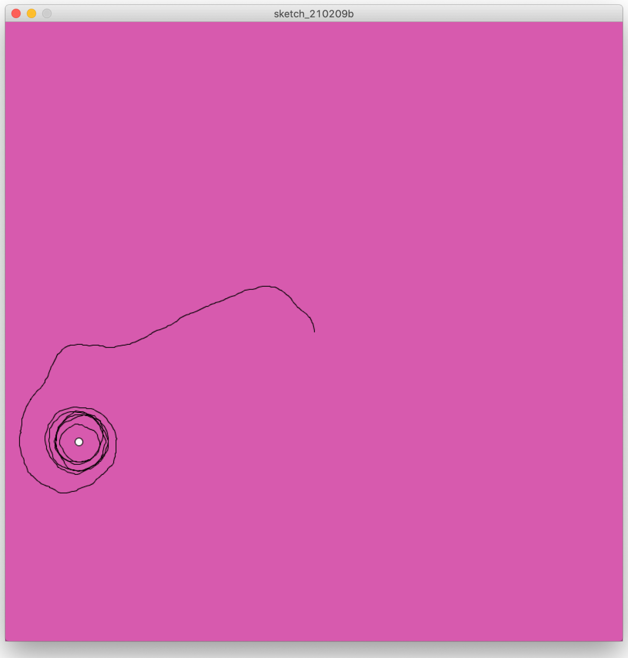
Random Motion

```
import Turtle.*;
Turtle t;

void setup() {
  size(800,800);
  background(200,100,170);
  t = new Turtle(this);
  frameRate(30);
}

void draw() {
  randomMove(-5, 10);
}

void randomMove(float minAngle, float maxAngle) {
  t.forward(random(1,10));
  t.right(random(minAngle,maxAngle));
}
```



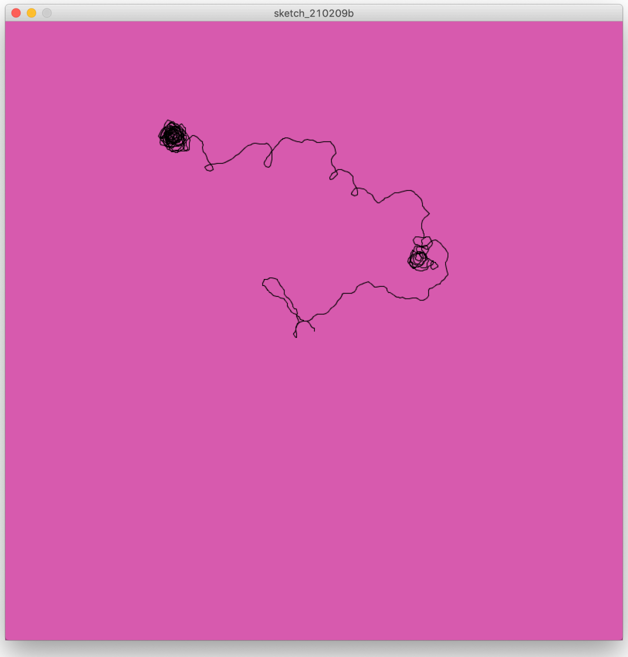
Move towards a “Smell”

```
import Turtle.*;
Turtle t;
float smellX, smellY;
float distToSmell, prevDistToSmell;

void setup() {
  size(800,800);
  background(200,100,170);
  t = new Turtle(this);
  smellX = random(10, width-10);
  smellY = random(10, height-10);
  ellipse(smellX, smellY, 10,10);
}

void draw() {
  findBySmell();
}

void randomMove(float minAngle, float maxAngle) {
  t.forward(random(1,10));
  t.right(random(minAngle,maxAngle));
}

void findBySmell() {
  t.forward(random(0,5));
  t.right(random(-10,10));
  prevDistToSmell = distToSmell;
  distToSmell = dist(t.getX(), t.getY(), smellX, smellY);
  if (distToSmell > prevDistToSmell) {
    t.left(random(0,20));
  }
}
```



Move towards a “Smell”

```
import Turtle.*;
Turtle t;
float smellX, smellY;
float distToSmell, prevDistToSmell;

void setup() {
  size(800,800);
  background(200,100,170);
  t = new Turtle(this);
  smellX = random(10, width-10);
  smellY = random(10, height-10);
}

void draw() {
  findBySmell();
  smellX = mouseX;
  smellY = mouseY;
}

void randomMove(float minAngle, float maxAngle) {
  t.forward(1);
  t.right(random(minAngle,maxAngle));
}

void findBySmell() {
  t.forward(random(0,5));
  t.right(random(-30,30));
  prevDistToSmell = distToSmell;
  distToSmell = dist(t.getX(), t.getY(), smellX, smellY);
  if (distToSmell > prevDistToSmell) {
    t.left(random(0,60));
  }
}
```



Upcoming Deadlines

Thursday 1/27
comments on introduction posts
reading: L-Systems

Tuesday 2/1
Small Assignment 2
reading: shape grammars

Thank you!

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