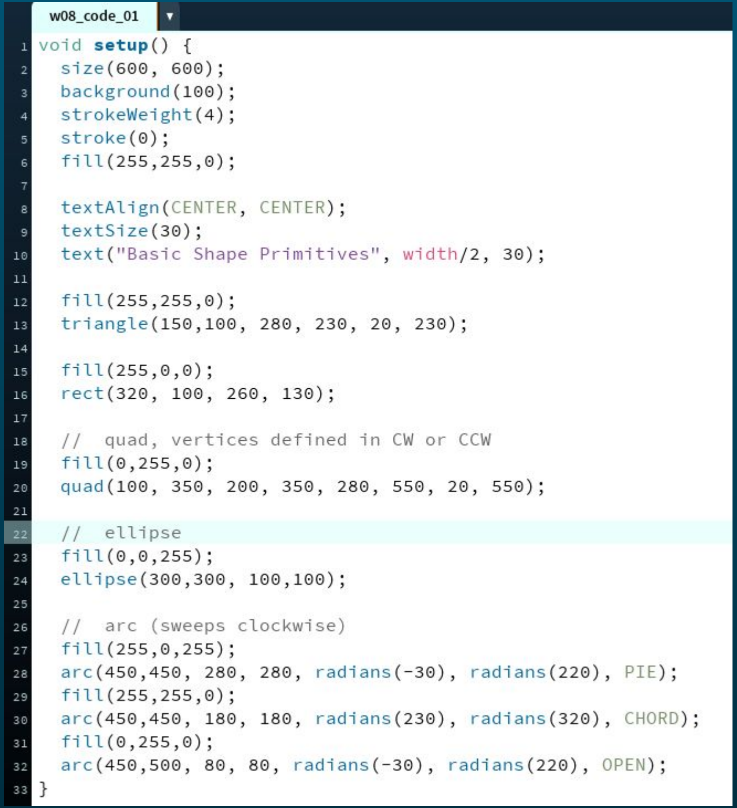
Week 08
Shape, Motion and Deformation

Quick Review of basic shape primitives

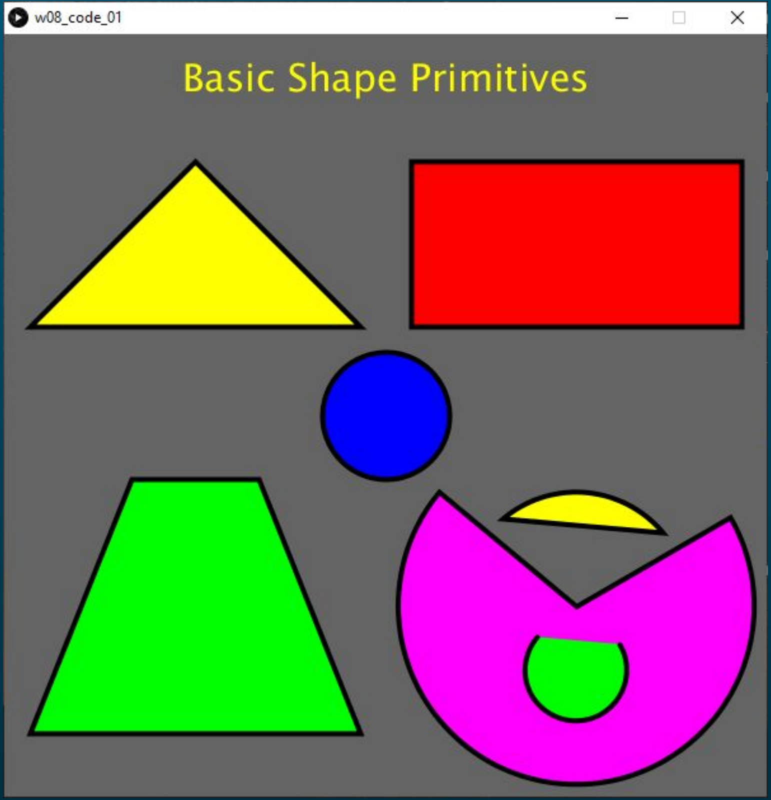
primitives	Description
`triangle()`	**Triangular** shape
`rect()`	**Rectangular** shape
`quad()`	**Quadrilateral**, a free-form **4-sided** polygon
`ellipse()`	**Ellipse** (Oval) shape
`arc()`	**Arc** with different closing modes

Example 1 - Shape Primitives



w08_code_01

```
void setup() {
  size(600, 600);
  background(100);
  strokeWeight(4);
  stroke(0);
  fill(255,255,0);

  textAlign(CENTER, CENTER);
  textSize(30);
  text("Basic Shape Primitives", width/2, 30);

  fill(255,255,0);
  triangle(150,100, 280, 230, 20, 230);

  fill(255,0,0);
  rect(320, 100, 260, 130);

  //  quad, vertices defined in CW or CCW
  fill(0,255,0);
  quad(100, 350, 200, 350, 280, 550, 20, 550);

  //  ellipse
  fill(0,0,255);
  ellipse(300,300, 100,100);

  //  arc (sweeps clockwise)
  fill(255,0,255);
  arc(450,450, 280, 280, radians(-30), radians(220), PIE);
  fill(255,255,0);
  arc(450,450, 180, 180, radians(230), radians(320), CHORD);
  fill(0,255,0);
  arc(450,500, 80, 80, radians(-30), radians(220), OPEN);
}
```

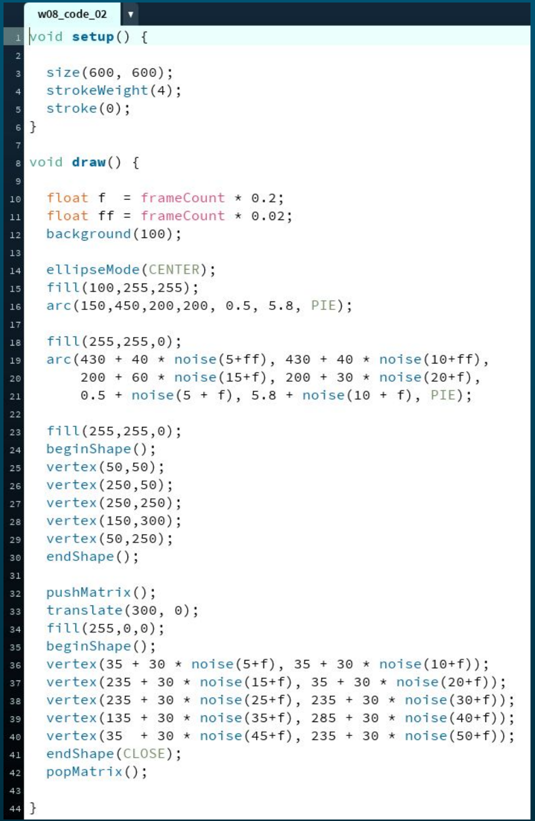


Simple N-sided polygon via `vertex()`

Simple N-sided polygon may be defined via a series of `vertex()` statements using the `beginShape()` and `endShape()` constructs.

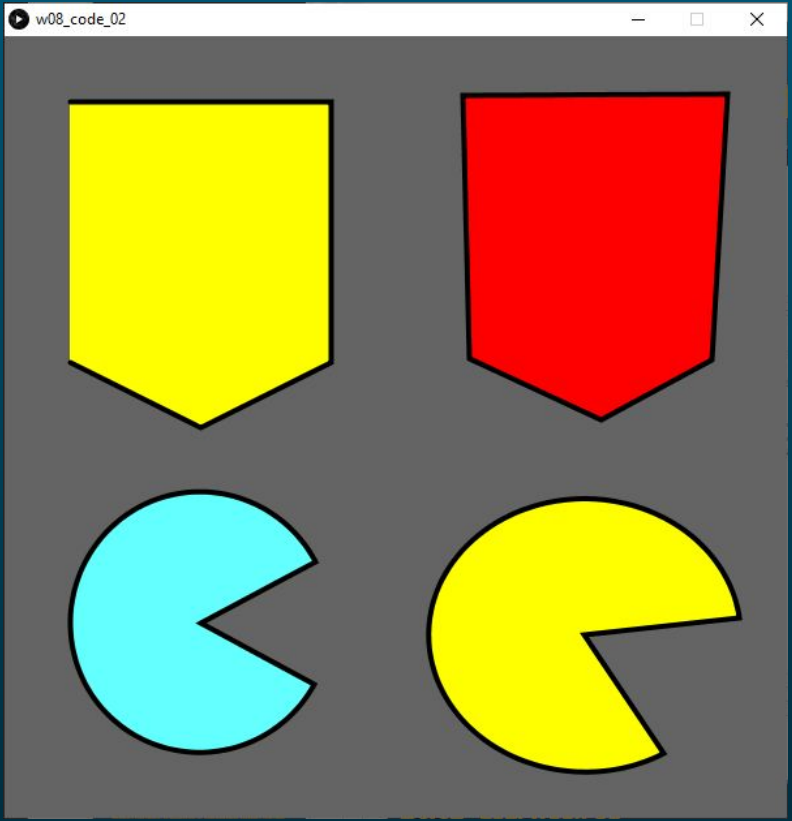
```
// Each vertex(x,y) represents a corner of the polygon
beginShape();
vertex(10, 10);
vertex(10, 20);
vertex(0, 20);
...
endShape(CLOSE);
```

Example 2 - Wiggling Shapes



w08_code_02

```
void setup() {

  size(600, 600);
  strokeWeight(4);
  stroke(0);
}

void draw() {

  float f  = frameCount * 0.2;
  float ff = frameCount * 0.02;
  background(100);

  ellipseMode(CENTER);
  fill(100,255,255);
  arc(150,450,200,200, 0.5, 5.8, PIE);

  fill(255,255,0);
  arc(430 + 40 * noise(5+ff), 430 + 40 * noise(10+ff),
      200 + 60 * noise(15+f), 200 + 30 * noise(20+f),
      0.5 + noise(5 + f), 5.8 + noise(10 + f), PIE);

  fill(255,255,0);
  beginShape();
  vertex(50,50);
  vertex(250,50);
  vertex(250,250);
  vertex(150,300);
  vertex(50,250);
  endShape();

  pushMatrix();
  translate(300, 0);
  fill(255,0,0);
  beginShape();
  vertex(35 + 30 * noise(5+f), 35 + 30 * noise(10+f));
  vertex(235 + 30 * noise(15+f), 35 + 30 * noise(20+f));
  vertex(235 + 30 * noise(25+f), 235 + 30 * noise(30+f));
  vertex(135 + 30 * noise(35+f), 285 + 30 * noise(40+f));
  vertex(35  + 30 * noise(45+f), 235 + 30 * noise(50+f));
  endShape(CLOSE);
  popMatrix();

}
```



Datatype for shapes: PShape

Shapes created using PShape can be displayed quickly using the shape() function. It is similar to the image() function designed for PImage data. An example to create a simple shape:

```
PShape sh;
sh = createShape();
sh.beginShape();
sh.noStroke();
sh.fill(255,255,0);
sh.vertex(100,100);
sh.vertex(200,200);
sh.vertex(0,200);
sh.endShape(CLOSE);
```

To display PShape shapes

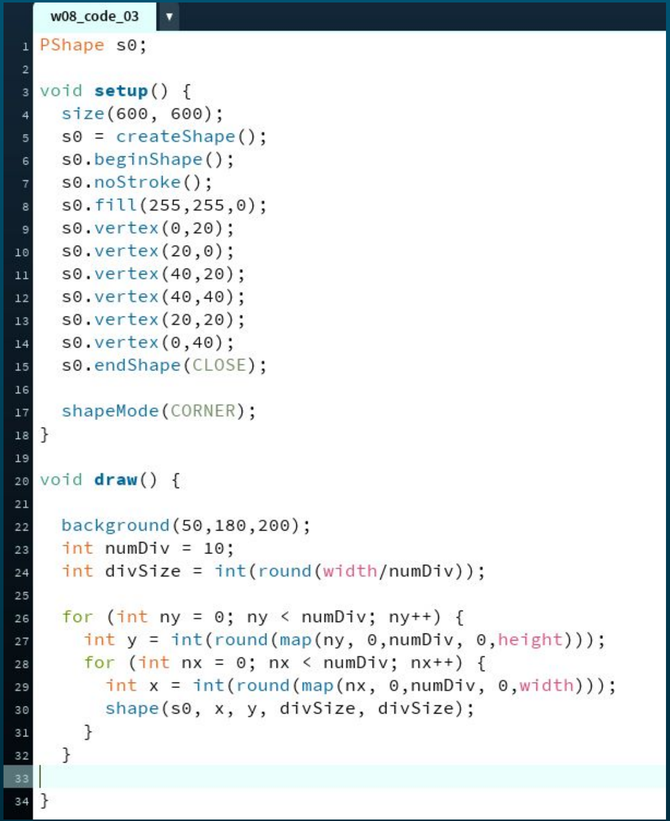
`shape()` function displays `PShape` shape with parameters for controlling the position and size. `shapeMode()` function defines the roles of the parameters.

```
shape(<shapeVar>, <x>, <y>, [w], [h]);
```

Example

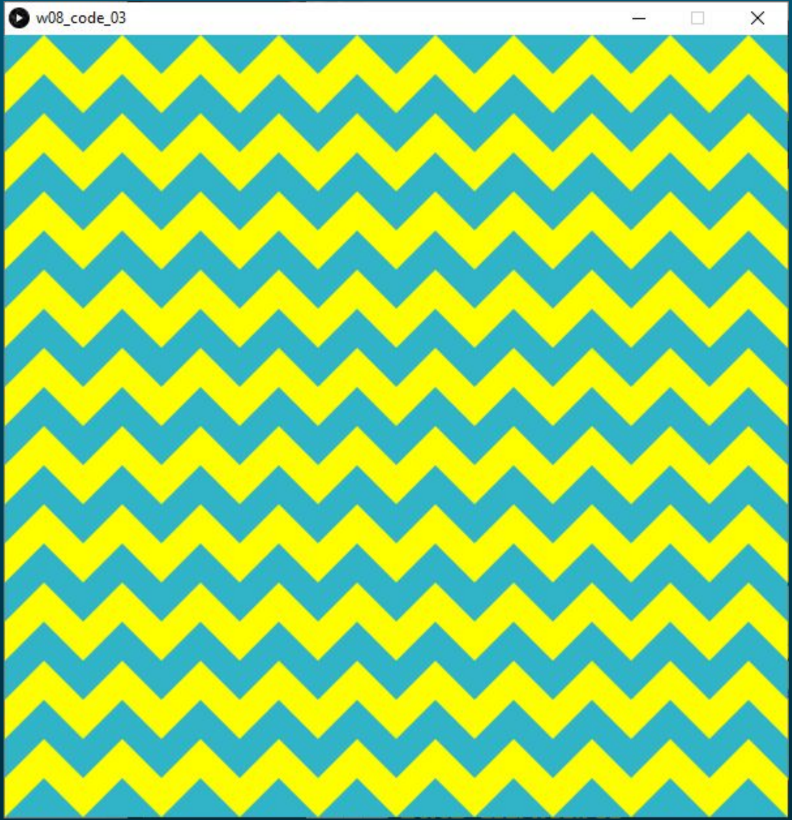
```
    shapeMode(CENTER);
    shape(sh, 100, 100, 200, 200);
```

Example 3 - PShape shapes



w08_code_03

```
PShape s0;

void setup() {
  size(600, 600);
  s0 = createShape();
  s0.beginShape();
  s0.noStroke();
  s0.fill(255,255,0);
  s0.vertex(0,20);
  s0.vertex(20,0);
  s0.vertex(40,20);
  s0.vertex(40,40);
  s0.vertex(20,20);
  s0.vertex(0,40);
  s0.endShape(CLOSE);

  shapeMode(CORNER);
}

void draw() {

  background(50,180,200);
  int numDiv = 10;
  int divSize = int(round(width/numDiv));

  for (int ny = 0; ny < numDiv; ny++) {
    int y = int(round(map(ny, 0,numDiv, 0,height)));
    for (int nx = 0; nx < numDiv; nx++) {
      int x = int(round(map(nx, 0,numDiv, 0,width)));
      shape(s0, x, y, divSize, divSize);
    }
  }

}
```

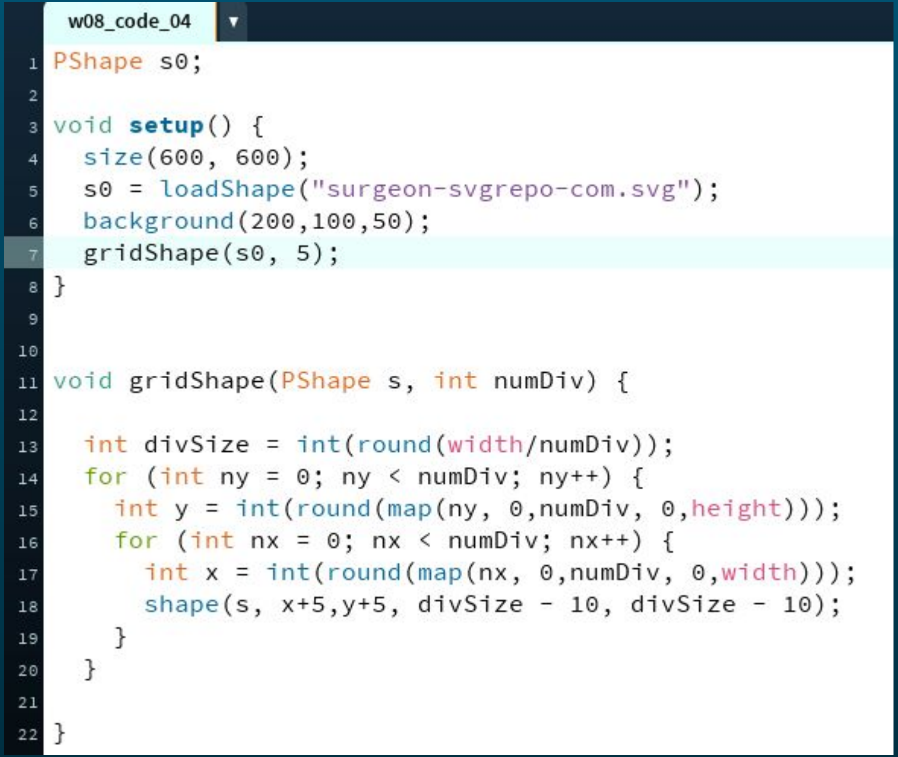


Load .SVG files via `loadShape()`

The function `loadShape()` loads .SVG files as `PShape` such that we may use .SVG artworks in our sketches.

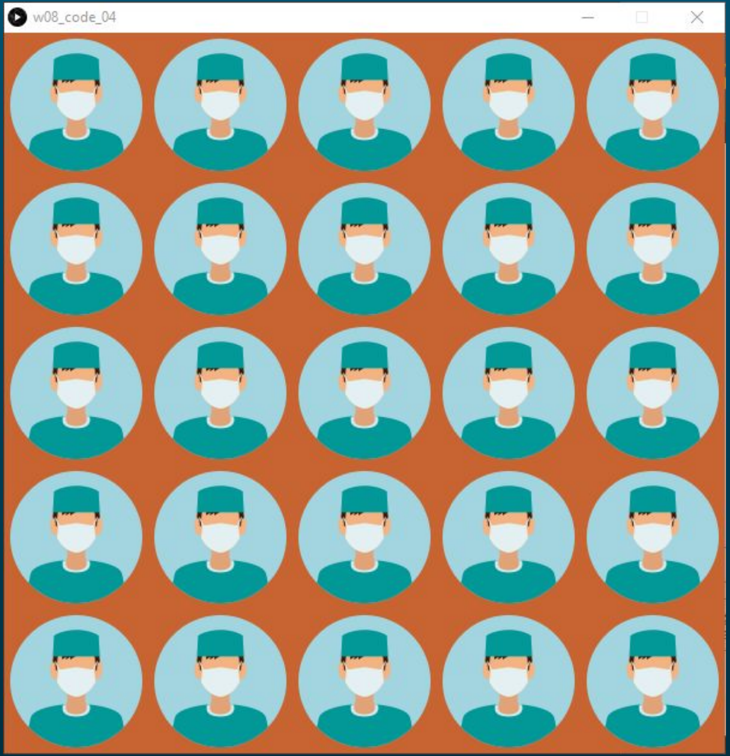
```
PShape sh;
sh = loadShape("some_shape.svg");
shape(sh, 100,100, 200,200);
```

Example 4 - using .SVG



w08_code_04

```
PShape s0;

void setup() {
  size(600, 600);
  s0 = loadShape("surgeon-svgrepo-com.svg");
  background(200,100,50);
  gridShape(s0, 5);
}


void gridShape(PShape s, int numDiv) {

  int divSize = int(round(width/numDiv));
  for (int ny = 0; ny < numDiv; ny++) {
    int y = int(round(map(ny, 0,numDiv, 0,height)));
    for (int nx = 0; nx < numDiv; nx++) {
      int x = int(round(map(nx, 0,numDiv, 0,width)));
      shape(s, x+5,y+5, divSize - 10, divSize - 10);
    }
  }

}
```



Use `sin()` & `cos()` to drive motion

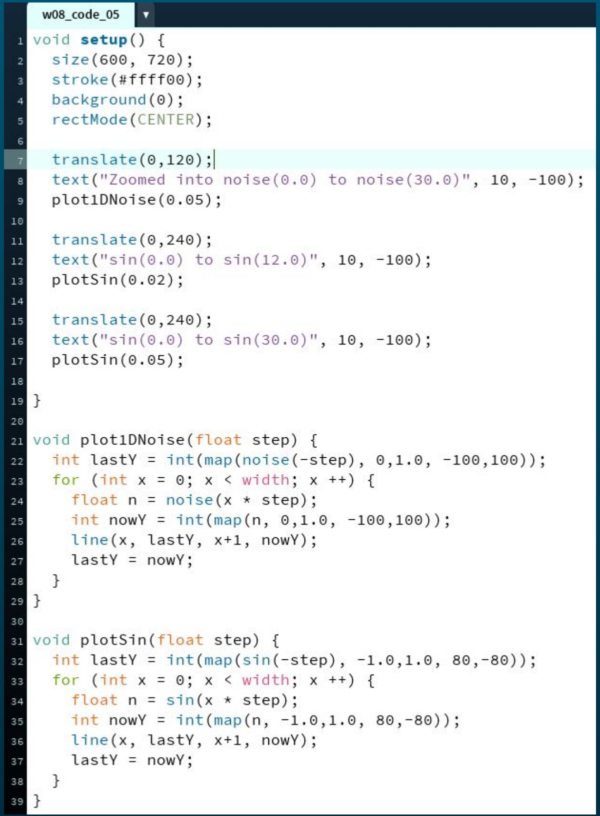
Apart from using `noise()`, `sin()` and `cos()` functions are also great for driving motion or displacement. These functions take an **angle** as input (angle defined in **radians**). Their outputs are fixed in the range of `-1.0` to `1.0`.

```
sin(<angle in rad>);
cos(<angle in rad>);
```

Processing pre-defined a few common angles in radians:

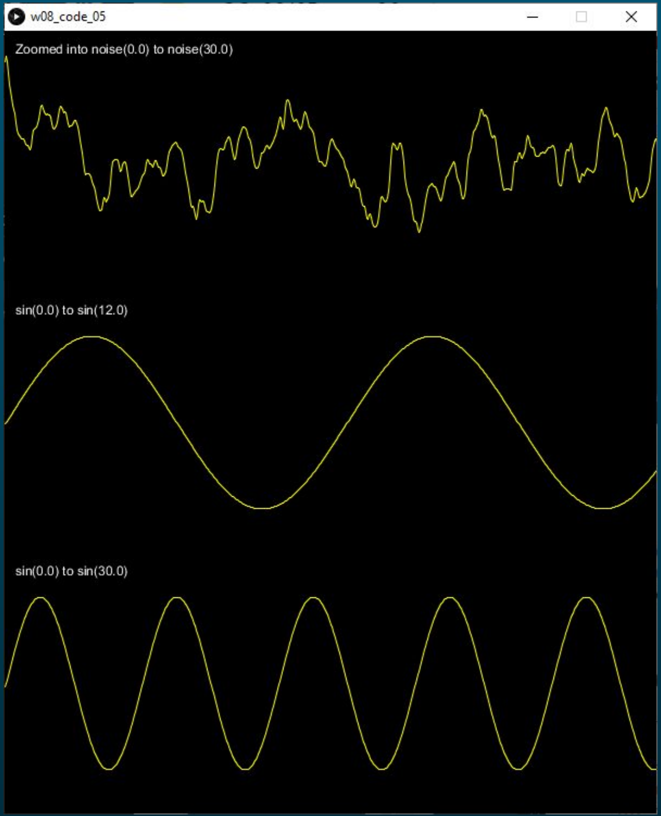
`PI` (180-deg), `TWO_PI` (360-deg)
`HALF_PI` (90-deg) & `QUARTER_PI` (45-deg)

Example 5 - simple plot

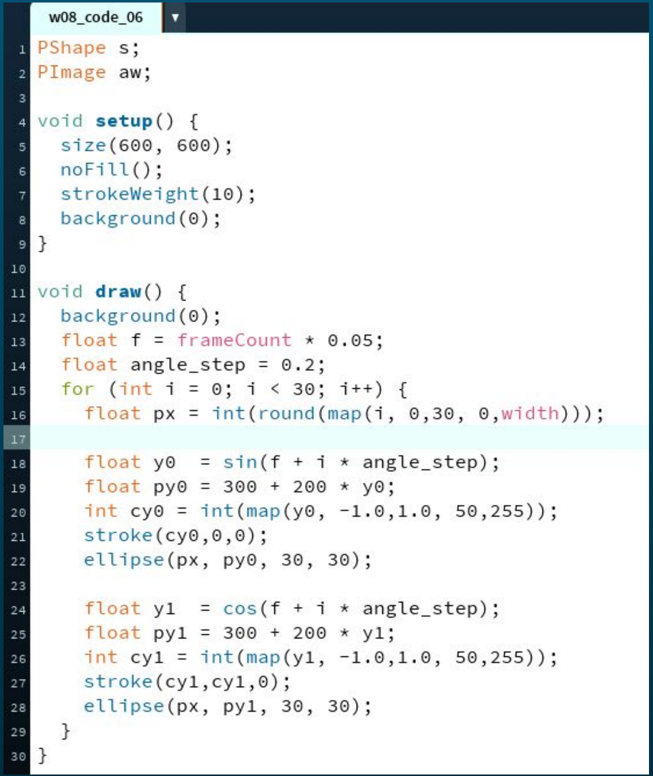


w08_code_05

```
void setup() {
  size(600, 720);
  stroke(#ffff00);
  background(0);
  rectMode(CENTER);

  translate(0,120);
  text("Zoomed into noise(0.0) to noise(30.0)", 10, -100);
  plot1DNoise(0.05);

  translate(0,240);
  text("sin(0.0) to sin(12.0)", 10, -100);
  plotSin(0.02);

  translate(0,240);
  text("sin(0.0) to sin(30.0)", 10, -100);
  plotSin(0.05);

}

void plot1DNoise(float step) {
  int lastY = int(map(noise(-step), 0,1.0, -100,100));
  for (int x = 0; x < width; x ++) {
    float n = noise(x * step);
    int nowY = int(map(n, 0,1.0, -100,100));
    line(x, lastY, x+1, nowY);
    lastY = nowY;
  }
}

void plotSin(float step) {
  int lastY = int(map(sin(-step), -1.0,1.0, 80,-80));
  for (int x = 0; x < width; x ++) {
    float n = sin(x * step);
    int nowY = int(map(n, -1.0,1.0, 80,-80));
    line(x, lastY, x+1, nowY);
    lastY = nowY;
  }
}
```

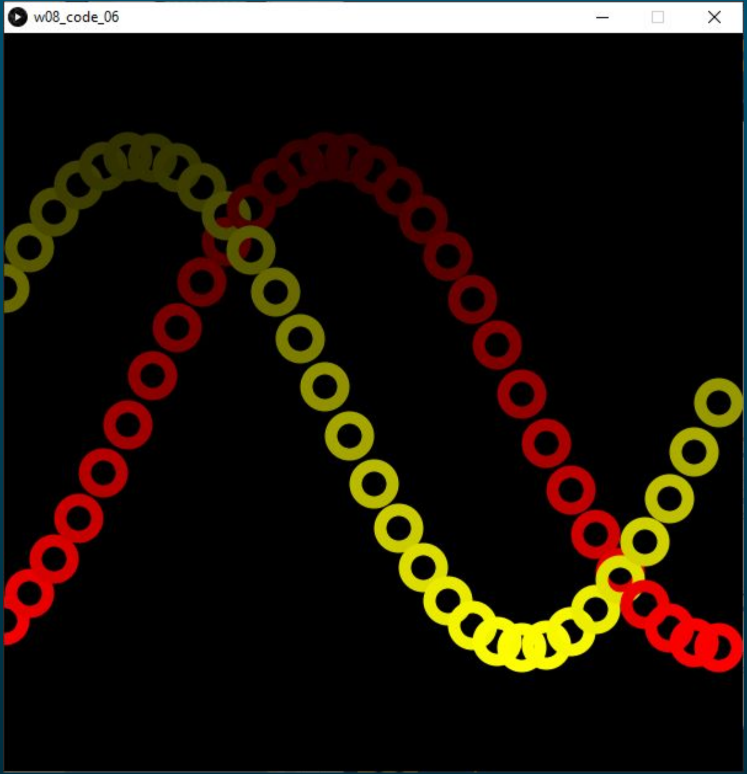


Example 5 - simple sin/cos motion



w08_code_06

```
PShape s;
PImage aw;

void setup() {
  size(600, 600);
  noFill();
  strokeWeight(10);
  background(0);
}

void draw() {
  background(0);
  float f = frameCount * 0.05;
  float angle_step = 0.2;
  for (int i = 0; i < 30; i++) {
    float px = int(round(map(i, 0,30, 0,width)));

    float y0  = sin(f + i * angle_step);
    float py0 = 300 + 200 * y0;
    int cy0 = int(map(y0, -1.0,1.0, 50,255));
    stroke(cy0,0,0);
    ellipse(px, py0, 30, 30);

    float y1  = cos(f + i * angle_step);
    float py1 = 300 + 200 * y1;
    int cy1 = int(map(y1, -1.0,1.0, 50,255));
    stroke(cy1,cy1,0);
    ellipse(px, py1, 30, 30);
  }
}
```



Example 7 - What's possible ?