Code For the Unit Circle Program, the fractal program, and the other programs:

```
int howMuchTheLinesRepeat, howMuchTheInitialCircleRepeats,
howManyTrianglesHaveBeenDrawn, imageRepeat, triangleStay,
amountOfTimesSpacePressed;
boolean delayBoolean, startCircle, secondTime, makingButterfly, sinButterfly,
cosButterfly,neutralButterfly, startFullButterfly, changeFractalBackground;
PImage cat, otter, fox,coyote,flower5,jaguar,cat2,geometry,flower9, fractal, img;
ArrayList<PImage> currentImages;
float howMuchTheCircleChanges, angle, num, amount, timesTheImagesChange; ;
PFont hi, boldHI;
color theTreesColor1, theTreesColor2;
int changeX = 50;
int changeY = 70;
float transparencyDegree;
void setup(){

   timesTheImagesChange = 0;
  amount = 3;
  amountOfTimesSpacePressed=-8;
  background(255);
  size(800,800, P3D);
  delayBoolean=true;
  secondTime=false;
  hi = loadFont("AgencyFB-Reg-48.vlw");
  //boldHI = loadFont("BodoniMT-BoldItalic-48.vlw");
  startFullButterfly=false;
   background(#E5FBFF);
}


void draw(){
 float howTransparent = map(mouseX,0,width,0.0,1);
 float glowRadius = changeX/2*howTransparent;
 switch(amountOfTimesSpacePressed) {
case -6:
justShowThePIAngles();
for(int i=0; i<2; i++){
 if(i==0){
   pushMatrix();
          translate(width,height);
   rotate(0.785398);
   rotate(PI/2);
   rotate(PI/4);
```

```
        for(int y=changeY; y<width; y+=changeX){
        for(float z=0; z<glowRadius; z++){
        stroke(#624BE8,255*(1-(z/glowRadius)));
        line(0,y+z,y+z,height);
        line(0,y-z,y-z,height);
 }
  }
        popMatrix();

 }
 else{
  for(int y=changeY; y<width; y+=changeX){
        for(float z=0; z<glowRadius; z++){

        stroke(#624BE8,255*(1-(z/glowRadius)));

        line(0,y+z,y+z,height);
        line(0,y-z,y-z,height);
 }
  }
 }

}
        break;


        case -5:
        noFill();
        ellipse(width/2,height/2,width,height);
        textFont(hi);
        switch(frameCount%140){
        case 1:

        background(#E6ECF7);
        letsDelayThis();
        showTheTrianglesGoingAround(30);
        noStroke();
        fill(255);
        rect(0, -225, 100, -50);
        fill(#5946C6);
        text("π/6",0, -225);
        noFill();
        ellipse(width/2,height/2,width,height);
        break;
```

```
case 2:
//ellipse(width/2,height/2,width,height);
letsDelayThis();
showTheTrianglesGoingAround(45);
        fill(255);
rect(0, -225, 100, -50);
fill(#5946C6);
text("π/4",0, -225);

break;
case 3:
//ellipse(width/2,height/2,width,height);
letsDelayThis();
showTheTrianglesGoingAround(60);
        fill(255);
rect(0, -225, 100, -50);
fill(#5946C6);
text("π/3",0, -225);
break;
case 4:
letsDelayThis();
showTheTrianglesGoingAround(120);
        fill(255);
rect(0, -225, 100, -50);
fill(#5946C6);
text("2π/3",0, -225);
break;
case 5:
letsDelayThis();
showTheTrianglesGoingAround(135);
        fill(255);
rect(0, -225, 100, -50);
fill(#5946C6);
text("3π/4",0, -225);
break;
case 6:
letsDelayThis();
showTheTrianglesGoingAround(150);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("5π/6",0, -225);
break;
case 7:
```

```
letsDelayThis();
showTheTrianglesGoingAround(210);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("7π/6",0, -225);
break;
case 8:
letsDelayThis();
showTheTrianglesGoingAround(225);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("5π/4",0, -225);
break;
case 9:
letsDelayThis();
showTheTrianglesGoingAround(240);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("4π/3",0, -225);
break;
case 10:
letsDelayThis();
showTheTrianglesGoingAround(300);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("5π/3",0, -225);
break;
case 11:
letsDelayThis();
showTheTrianglesGoingAround(315);
        fill(255);
rect(0, -225, 120, -50);
fill(#5946C6);
text("7π/4",0, -225);
break;
case 12:
letsDelayThis();
showTheTrianglesGoingAround(330);
        fill(255);
rect(0, -225, 120, -50);
```

```
        fill(#5946C6);
        text("11π/6",0, -225);
        //background(255);
        case 13:
        stroke(#5946C6);
        line(0,0, 20,-200);
        line(0,0, -20,-200);
        ellipse(-20,-200, 5,5);
        ellipse(20,-200, 5,5);
        letsDelayThis();

        break;
        }
        //background(255);
        sinButterfly=true;
        break;
        case -4:
        weirdCircleTransition();
        textFont(hi);
        fill(#FF9F46);
        textSize(80);
        text("    Sin (the height) \n greatest when divisible by 3 : \n π/3, 2π/3, 4π/3, 5π/3",0,
height/2-100);
        break;
        case -3:
        background(#E6ECF7);
        amountOfTimesSpacePressed++;
        break;
        case -2:

        noStroke();
        if(sinButterfly){
        background(#E6ECF7);
        sinButterfly=false;
        }
        showSinButterfly();
        break;
        case -1:
        weirdCircleTransition();
        textFont(hi);
        fill(#FF9F46);
        textSize(80);
        text("    Cos (the width) \n greatest when divisible by 6 : \n π/6, 5π/6, 7π/6, 11π/",0,
height/2-100);
```

```
        break;
        case 0:
        background(#E6ECF7);
        amountOfTimesSpacePressed++;
        break;
        case 1:
        noStroke();
        if(cosButterfly){
        background(255);
        cosButterfly=false;
        }
        showCosButterfly();
        break;


        case 2:
        weirdCircleTransition();
        textFont(hi);
        fill(#FF9F46);
        textSize(80);
        text("   Cos and Sin(width and height)\n the same when divisible by 4: \n π/4, 3π/4,
5π/4, 7π/4",0, height/2-100);
        break;
        case 3:
        background(#E6ECF7);
        amountOfTimesSpacePressed++;
        break;
        case 4:
        noStroke();
        if(neutralButterfly){
        background(255);
        neutralButterfly=false;
        }
        showNeutralButterfly();
        break;


        case 5:
        if(!startFullButterfly){
        background(255);
        startFullButterfly=true;
        }
        if(!secondTime){
```

```
        makeButterfly();
        }
        break;

        case 6:
        background(100);
        makingButterfly=false;
        showTheTrianglesGoingAround(30);
        //fill(70);//makes the fill a grey
        // translate(width, height);
        //noStroke(); //no outline
        //rect(0,0,width,height);//makes a rectangle that fills the screen
        //fill(125);//fills something else with 125
        //letsDelayThis();
        break;
  case 7:
        showTheTrianglesGoingAround(45);
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #0A0808, #0A458B);
        break;
  case 8:
        showTheTrianglesGoingAround(60);
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #2A458B, #2A458B);
        break;
  case 9:
        showTheTrianglesGoingAround(120);
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #2A458B, #2A458B);
        break;
  case 10:
        showTheTrianglesGoingAround(135);
        break;
  case 11:
        showTheTrianglesGoingAround(150);
        break;
  case 12:
        showTheTrianglesGoingAround(210);
        break;
  case 13:
        showTheTrianglesGoingAround(225);
        break;
  case 14:
        showTheTrianglesGoingAround(240);
```

```
        break;
 case 15:
        showTheTrianglesGoingAround(300);
        break;
 case 16:
        showTheTrianglesGoingAround(315);
        break;
 case 17:
 showTheTrianglesGoingAround(330);
 noFill();
 stroke(100,100,900);
 strokeWeight(4);
 ellipse(0,0,width,height);
 if(!secondTime){
 line(0,0, 20,-200);
 line(0,0, -20,-200);
 ellipse(-20,-200, 5,5);
 ellipse(20,-200, 5,5);
 break;
 }
  break;
 case 18:
        if (!secondTime) {
        amountOfTimesSpacePressed=6;
        }
        if(secondTime){
        background(100);
        }
        secondTime=true;
        break;

case 19:
// frameRate(10);
 background(#7587A5);
float x = width;
num = 20;
noStroke();
 translate(width/2,height/2);
 for(float  a=0; a<360; a+=45){
        rotate(radians(a));
  // push();
 for(int i=0; i<num; i++){
  scale(0.95);
  rotate(radians(angle/15));
```

```
fill(random(200,230),random(200,230),random(200,230),map(cos(radians(frameCount/3)),-1,1,
50,100));
   ellipse(x,0,map(mouseX, 0, width, 0, 300),map(mouseX, 0, width, 0, 300));
  }
  for(int i=0; i<num; i++){
   scale(0.95);
   rotate(-radians(angle/12));
   ellipse(x,0,map(mouseX, 0, width, 0, 300),map(mouseX, 0, width, 0, 300));
  }
  }
 // pop();
  angle+=1;

  break;

  case 20:
          //surface.setLocation(width/2, height/2);
          translate(width/2, height/2);
   //background(#8AB6FC);
   rotate(radians(frameCount/3));
   makeUnitCircle();
  break;
  case 21:
          translate(width/2, height/2);
          background(#8AB6FC);
   // rotate(radians(frameCount/3));
          makeUnitCircle();
   // System.out.print("HI!");
  break;

case 22:
background(255);
translate(-width/2, -height/2);
//showTheTrianglesGoingAround(30);
break;

case 23:
fill(#372C79);
//rect(0,0,100,100);
 drawTree(random(0,width), random(0,height), random(0,200), random(3,5), random(0,2),
#070417, #2A458B);
 changeFractalBackground=true;
```

```
 break;

 case 24:

 if(changeFractalBackground){
 background(255);
 }
 drawTree(width/2, height/2, 100,map(mouseX,0,width,0,365), 1, #070417, #2A458B);
 break;

 case 25:
 //System.out.print("HI!");
 if(changeFractalBackground){
 background(255);
 }
  drawTree(width/2, height/1.5, 100,7, 1, #070417, #2A458B);

 break;

 /*
 case 20:


 //drawTree(width/2, height/2, 100, -7, 1, #0C0B0F, #0C0B0F);

 break;

 case 21:
 System.out.print("Hi!");
 background(#070417);

 break;

/* case 21:
 if(changeFractalBackground){
 background(255);

 }
 drawTree(width/2, height/2, 100, 7, 1, #0C0B0F, #0C0B0F);
 break;
 */
/*

          case 18:showTheTrianglesGoingAround(45);
```

```
        System.out.print("HI!");
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #0A0808, #0A458B);
        break;
        case 19:showTheTrianglesGoingAround(60);
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #2A458B, #2A458B);
        break;
        case 20:showTheTrianglesGoingAround(120);
        // drawTree(random(0,width), random(0,height), random(0,200), random(3,5),
random(0,2), #2A458B, #2A458B);
        break;
        case 21:showTheTrianglesGoingAround(135);
        break;
        case 22:showTheTrianglesGoingAround(150);
        break;
        case 23:showTheTrianglesGoingAround(210);
        break;
        case 24:showTheTrianglesGoingAround(225);
        break;
        case 25:showTheTrianglesGoingAround(240);
        break;
        case 26:showTheTrianglesGoingAround(300);
        break;
        case 27:showTheTrianglesGoingAround(315);
        break;
        case 28:showTheTrianglesGoingAround(330);
        if(!secondTime){
        amountOfTimesSpacePressed=3;
        }
        secondTime=true;
        break;
        */
}
}

 // else if(amountOfTimesSpacePressed==3){
  //  background(100);
  // drawTree(random(0,width), random(0,height), random(0,200), random(3,5), random(0,2),
#2A458B, #2A458B);
 // }
```

```
/*
void draw3() {
  switch(amountOfTimesSpacePressed) {

}
*/

 void letsDelayThis(){//pauses for 1.5 seconds
  if(delayBoolean==true){
  delay(1500);
}
  }


  void makeUnitCircle(){
  fill(0);
  howMuchTheCircleChanges =map(mouseX, 0, width, 0,width);
  stroke(#8594D8);
  strokeWeight(2);
  //translate(width/2,height/2);
  ellipse(0,0, howMuchTheCircleChanges,howMuchTheCircleChanges);

  for(int i=0; i<4; i++){
   rotate(radians(90));
   line(0,0, howMuchTheCircleChanges/2,0);
  }
  for(int i=0; i<2; i++){
  rotate(radians(-30));
  line(0,0, howMuchTheCircleChanges/2,0);
  rotate(radians(-15));
          line(0,0, howMuchTheCircleChanges/2,0);
          rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
  rotate(radians(-60));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-30));
 }
  }

void keyPressed(){
```

```
  if(amountOfTimesSpacePressed<30){
  amountOfTimesSpacePressed++;
  }

}
        void showTheTriangleLines(int x){
   // letsDelayThis();//delays for 2 seconds
        fill(70);
        //rect(0,0,width,height);
        zoomIntoTheIndividualTriangles2(x);
 }
 void showTheTrianglesGoingAround(int x){
 showTheTriangleLines(x);
 if(howMuchTheLinesRepeat%3==0){
 }
}

  void zoomIntoTheIndividualTriangles(int x){
        if(amountOfTimesSpacePressed<100){
        push();
   translate(width/2,width/2); //moves everything by the width and height
   if(secondTime){
        fill(100);
        ellipse(0,0,width,height); //makes an ellipse the size of the box
   }
        rotate(radians(-x)); //rotates back
        strokeWeight(4); //makes the stroke a bit bold
        stroke(100, 900, 1000); //makes the stroke in a random color
        line(0,0, 400,0);//first line
        rotate(radians(x));  //rotates
        line(0,0,cos(radians(x))*400,0); // draws the second
        line(cos(radians(x))*400, 0, cos(radians(x))*400, -sin(radians(x))*400); //draws the third
line

        fill(100,10,900);
        if(x>0&&x<90){
        rect(cos(radians(x))*400,0,-20,-20);
        }
        else if(x>90 && x<180){
        rect(cos(radians(x))*400,0,20,-20);
        }
        else if(x>180 && x<270){
        rect(cos(radians(x))*400,0,20,20);
        }
```

```
        else{
        rect(cos(radians(x))*400,0,-20,20);
        }
        pop();
        }

        fill(1000,150,150);
        textSize(37);
        textFont(hi);
  translate(width/2, height/2);
  if(secondTime){
        switch(x){
        case 30: text(x+"degrees  "+x+ "*2π/180 =  π/6 radians",-400, 50);//texts the degrees
and such
        //fill(70);//makes the fill a grey
        // translate(width, height);
                //noStroke(); //no outline
                //rect(0,0,width,height);//makes a rectangle that fills the screen
                //fill(125);//fills something else with 125
        //letsDelayThis();
        break;
        case 45: text(x+"  degrees  = π/4 radians",-400, 50);
        break;
        case 60: text(x+"  degrees = π/3 radians",-400, 50);
        break;
        case 120: text(x+" degrees = 2π/3 radians",-400, 50);
        break;
        case 135:text(x+"  degrees = 3π/4 radians",-400, 50);
        break;
        case 150:text(x+"  degrees = 5π/6 radians",-400, 50);
        break;
        case 210:text(x+"  degrees = 7π/6 radians",-300, -50);
        break;
        case 225:text(x+"  degrees  = 5π/4 radians",-300, -50);
        break;
        case 240:text(x+"  degrees = 4π/3 radians",-400, -50);
        break;
        case 300:text(x+"  degrees = 5π/3 radians",-400, -50);
        break;
        case 315:text(x+"  degrees = 7π/4 radians",-400, -50);
        break;
        case 330:text(x+"  degrees = 11π/6 radians",-400, -50);
        break;
  }
```

```
  }
 }
 void drawTree(float startX, float startY, float len, float angle, float branchWidth, color a, color b){
  theTreesColor1 =a;
  theTreesColor2 = b;
  strokeWeight(branchWidth);
  stroke(a);
  fill(b);
  translate(startX, startY);
  rotate(radians(angle));
  line(0,0,0,-len);
  if(len<1){
  }
  else{
          drawTree(0,-len,len*0.75, angle+7, branchWidth, a, b);
          drawTree(0,-len,len*0.75, angle+7, branchWidth, b, a);

  }
}
void makeButterfly(){
 makingButterfly=true;
 //showTheTrianglesGoingAround(30);
  switch(frameCount%39) {
          case 1:
          showTheTrianglesGoingAround(30);
          break;
  case 2:
          showTheTrianglesGoingAround(45);
          break;
  case 3:
          showTheTrianglesGoingAround(60);
          break;
  case 4:
          showTheTrianglesGoingAround(120);
          break;
  case 5:
          showTheTrianglesGoingAround(135);
          break;
  case 6:
          showTheTrianglesGoingAround(150);
          break;
  case 7:
          showTheTrianglesGoingAround(210);
          break;
```

```
  case 8:
          showTheTrianglesGoingAround(225);
          break;
  case 9:
          showTheTrianglesGoingAround(240);
          break;
  case 10:
          showTheTrianglesGoingAround(300);
          break;
  case 11:
          showTheTrianglesGoingAround(315);
          break;
  case 12:
          showTheTrianglesGoingAround(330);
          break;
}

}
void showATriangleWithLengths(){
  text("  The hypotenuse of the triangle is 1(the radius) and the other sides are the height sin(of
the angle) & cos the length(of the angle)", -400, 50);
  if( amountOfTimesSpacePressed==15){
  }
  else if(amountOfTimesSpacePressed==16){

  }
}
void justShowThePIAngles(){
  background(#B8C3F0);
  makeUnitCircle2();
   fill(#FCE8C2);
   textSize(50);
   text(" Angles rotate from 0 - 2π",10,width/2);
          text("              counterclockwise",10,width/2+50);

   text("           (angle*π)/180", 10, width/2+100);

}
void showSinButterfly(){
          switch(frameCount%140){
          case 1:
          letsDelayThis();
          showTheTrianglesGoingAround(60);
                  fill(255);
```

```
		rect(0, -225, 100, -50);
		fill(#5946C6);
		text("π/3",0, -225);
		break;
		case 2:
		letsDelayThis();
		showTheTrianglesGoingAround(120);
			fill(255);
		rect(0, -225, 100, -50);
		fill(#5946C6);
		text("2π/3",0, -225);
		break;
		case 3:
		letsDelayThis();
		showTheTrianglesGoingAround(240);
			fill(255);
		rect(0, -225, 120, -50);
		fill(#5946C6);
		text("4π/3",0, -225);
		break;
		case 4:

		letsDelayThis();

		showTheTrianglesGoingAround(300);
		stroke(#5946C6);
		line(0,0, 20,-200);
		line(0,0, -20,-200);
		ellipse(-20,-200, 5,5);
		ellipse(20,-200, 5,5);
			fill(255);
		rect(0, -225, 120, -50);
		fill(#5946C6);
		text("5π/3",0, -225);
		break;
		}
}
void showCosButterfly(){
		switch(frameCount%140){
		case 1:
		letsDelayThis();
		showTheTrianglesGoingAround(30);
			fill(255);
		rect(0, -225, 100, -50);
```

```
        fill(#5946C6);
        text("π/6",0, -225);
        break;
        case 2:
        letsDelayThis();
        showTheTrianglesGoingAround(150);
                fill(255);
        rect(0, -225, 100, -50);
        fill(#5946C6);
        text("5π/6",0, -225);
        break;
        case 3:
        letsDelayThis();
        showTheTrianglesGoingAround(210);
                fill(255);
        rect(0, -225, 120, -50);
        fill(#5946C6);
        text("7π/6",0, -225);
        break;
        case 4:
        letsDelayThis();

        showTheTrianglesGoingAround(330);
        stroke(#5946C6);
        line(0,0, 20,-200);
        line(0,0, -20,-200);
        ellipse(-20,-200, 5,5);
        ellipse(20,-200, 5,5);
                fill(255);
        rect(0, -225, 120, -50);
        fill(#5946C6);
        text("11π/6",0, -225);
        break;
        }
        }
void showNeutralButterfly(){
        switch(frameCount%140){
        case 1:
        letsDelayThis();
        showTheTrianglesGoingAround(45);
                fill(255);
        rect(0, -225, 100, -50);
        fill(#5946C6);
        text("π/4",0, -225);
```

```
        break;
        case 2:
        letsDelayThis();
        showTheTrianglesGoingAround(135);
                fill(255);
        rect(0, -225, 100, -50);
        fill(#5946C6);
        text("3π/4",0, -225);
        break;
        case 3:
        letsDelayThis();
        showTheTrianglesGoingAround(225);
                fill(255);
        rect(0, -225, 120, -50);
        fill(#5946C6);
        text("5π/4",0, -225);
        break;
        case 4:
        letsDelayThis();

        showTheTrianglesGoingAround(315);
        stroke(#5946C6);
        line(0,0, 20,-200);
        line(0,0, -20,-200);
        ellipse(-20,-200, 5,5);
        ellipse(20,-200, 5,5);
                fill(255);
        rect(0, -225, 120, -50);
        fill(#5946C6);
        text("7π/4",0, -225);
        break;
        }
        }



void makeUnitCircle2(){
  fill(#545D83);
  howMuchTheCircleChanges =0;
  stroke(#8594D8);
  strokeWeight(2);
  //translate(width/2,height/2);
  ellipse(height/2,width/2,width,height);
```

```
  for(int i=0; i<4; i++){
   rotate(radians(90));
   line(0,0, howMuchTheCircleChanges/2,0);
  }
  for(int i=0; i<2; i++){
  rotate(radians(-30));
  line(0,0, howMuchTheCircleChanges/2,0);
  rotate(radians(-15));
          line(0,0, howMuchTheCircleChanges/2,0);
          rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
  rotate(radians(-60));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-15));
   line(0,0, howMuchTheCircleChanges/2,0);
   rotate(radians(-30));
 }
  }



void zoomIntoTheIndividualTriangles2(int x) {
  if (amountOfTimesSpacePressed<30) {
          push();
          translate(width/2, width/2);//moves everything by the width and height
          if (secondTime) {
          fill(100);
          ellipse(0, 0, width, height); //makes an ellipse the size of the box
          }
          //rotate(radians(-x)); //rotates back
          strokeWeight(4); //makes the stroke a bit bold
          stroke(map(mouseX, 0, width, 50, 225),map(mouseX, 0, width, 200, 225),255); //makes
the stroke based on
   // line(0, 0, 400, 0);//first line
          if(makingButterfly){
          fill(map(mouseX, 0, width, 200,250),map(mouseX, 0, width, 200, 225),random(200,255));
   // triangle(400,0,  cos(radians(x))*400, 0,cos(radians(x))*400, -sin(radians(x))*400);
   triangle(0,0,  cos(radians(x))*400, 0, cos(radians(x))*400, -sin(radians(x))*400);
   line(0,0, 20,-200);
   line(0,0, -20,-200);
   ellipse(-20,-200, 5,5);
   ellipse(20,-200, 5,5);
```

```
 }
 else{
        //rotate(radians(-x)); //rotates back
        //line(0, 0, 400, 0);//first line
        //line(0, 0, cos(radians(x))*400, 0); // draws the second
        //line(cos(radians(x))*400, 0, cos(radians(x))*400, -sin(radians(x))*400); //draws the third
line
        fill(map(mouseX, 0, width, 200,250),map(mouseX, 0, width, 200, 225),random(200,255));
        triangle(0,0,  cos(radians(x))*400, 0, cos(radians(x))*400, -sin(radians(x))*400);
        //rotate(radians(x));  //rotates
  //line(0, 0, cos(radians(x))*400, 0); // draws the second
  // line(cos(radians(x))*400, 0, cos(radians(x))*400, -sin(radians(x))*400); //draws the third line
 }

        fill(map(mouseX, 0, width, 0, 225));
        if (x>0&&x<90&&!makingButterfly) {
        rect(cos(radians(x))*400, 0, -20, -20);
        } else if (x>90 && x<180&&!makingButterfly) {
        rect(cos(radians(x))*400, 0, 20, -20);
        } else if (x>180 && x<270&&!makingButterfly) {
        rect(cos(radians(x))*400, 0, 20, 20);
        } else if(!makingButterfly) {
        rect(cos(radians(x))*400, 0, -20, 20);
        }
        pop();
 }
// }
 fill(map(mouseX, 0, width, 0, 225));
 textSize(37);
 textFont(hi);
 translate(width/2, height/2);
 if (secondTime) {
        switch(x) {
        case 30:
        text(x+"degrees  "+x+ "*π/180 =  π/6 radians", -400, 50);//texts the degrees and such
        //fill(70);//makes the fill a grey
        // translate(width, height);
        //noStroke(); //no outline
        //rect(0,0,width,height);//makes a rectangle that fills the screen
        //fill(125);//fills something else with 125
        //letsDelayThis();
        break;
        case 45:
```

```
        text(x+"  degrees  = π/4 radians", -400, 50);
        break;
        case 60:
        text(x+"  degrees = π/3 radians", -400, 50);
        break;
        case 120:
        text(x+" degrees = 2π/3 radians", -400, 50);
        break;
        case 135:
        text(x+"  degrees = 3π/4 radians", -400, 50);
        break;
        case 150:
        text(x+"  degrees = 5π/6 radians", -400, 50);
        break;
        case 210:
        text(x+"  degrees = 7π/6 radians", -300, -50);
        break;
        case 225:
        text(x+"  degrees  = 5π/4 radians", -300, -50);
        break;
        case 240:
        text(x+"  degrees = 4π/3 radians", -400, -50);
        break;
        case 300:
        text(x+"  degrees = 5π/3 radians", -400, -50);
        break;
        case 315:
        text(x+"  degrees = 7π/4 radians", -400, -50);
        break;
        case 330:
        text(x+"  degrees = 11π/6 radians", -400, -50);
        break;
        }
  }
}


void makeGrid(int e, int f,int g, PImage img){
        push();
  translate(e*width/amount,e*height/amount);
//  rotateY(radians(frameCount));
        float divisions = map(mouseX, 0, 900, 0, 400);
        float amountOfDevisions = (width/amount)/divisions;
```

```
        for (int x = 0; x < divisions; x++) {
        for (int y = 0; y < divisions; y++) {
        color c = img.get(int(x*amountOfDevisions),int(y*amountOfDevisions));
        float b = map(brightness(c),0,255,1,0);
        // float z = map(b,0,1,-150,150);
        push();

        translate(x*amountOfDevisions - (f*width/amount), y*amountOfDevisions -
(g*height/amount));
        //fill(random(10), random(20),random(30));
        fill(random(0,200),random(100,255),random(150,255));
        sphere(amountOfDevisions*b*0.8);
        pop();
  }
}
  pop();
}

void weirdCircleTransition(){
  fill(0,10);
  //hi.update();
// frameRate(20);
 // background(random(0,125),random(0,125),random(0,125));
 fill(random(0,125), random(0,125), random(0,125));
  ellipse(random(0,width),random(0,height), random(150,200),random(150,200));
  noFill();
  stroke(0);
   ellipse(random(0,width),random(0,height), random(150,200),random(150,200));
   stroke(255);
   ellipse(random(0,width),random(0,height), random(150,200),random(150,200));

  stroke(random(100,125));

}
```

Code for the sin, cos, tan, program.
Credits Go to:

Rodenbroker, T. R. [ tim rodenbröker creative coding]. (2021, June 17). *Exploring Wave-Figures in Processing* [Video]. YouTube. https://www.youtube.com/watch?v=Th7m7QEeUbI&t=947s

```
int  amountOfTimesSpacePressed;
boolean changeBackground2,changeBackground3,changeBackground4;

void setup(){
  frameRate(30);
 amountOfTimesSpacePressed = 0;
  size(900,900,P3D);


}


void draw(){
  translate(width/2, height/2);
  float mag = 400;
  float s = 10;
  noStroke();
  switch(amountOfTimesSpacePressed) {

case 0:
 for(int i = 0; i<360; i++){

        float wave1 = map(tan(radians(frameCount*0.8)), -1, 1, -100, 100);
        float wave2 = map(cos(radians(frameCount)), -1, 1, -400,400);
        float c = map(tan(radians(frameCount*10)), -1,1,0,255);
        fill(c);
        rect(wave1, wave2, 20,20);

  }
changeBackground2=true;
break;
```

```
case 1:
if(changeBackground2){
 background(255);
 changeBackground2 = false;

}
 for( int i = 0; i<360; i++){//repeats 100 times
          float w = map(sin(radians(frameCount)), -1,1,-100,100);
          float wave1 = map(tan(radians(frameCount*0.8+i+w)), -1, 1, -100, 100);
          float wave2 = map(tan(radians(frameCount+i)), -1, 1, -width/2, height/2);
          float c= map(tan(radians(frameCount*5+i)), -1,1, 0,255); //between shades of grey
          fill(c); //fills it between greys
          ellipse(wave1, wave2, s, s);//makes a rectange that is between -100 and 100
}
changeBackground3 = true;

break;
case 2:
if(changeBackground3){
 background(255);
 changeBackground3 = false;

}


for(int i = 0; i<360; i++){
          float wave1 = map(tan(radians(frameCount*0.8+i)), -1, 1, -100, 100);
          float wave2 = map(cos(radians(frameCount+i)), -1, 1, -450,450);
          float c = map(tan(radians(frameCount*10)), -1,1,0,255);
          fill(c);
          rect(wave1, wave2, 20,20);
  }
  changeBackground4 = true;

break;


case 3:
if(changeBackground4){
 background(255);
 changeBackground4 = false;

}
```

```
 for(int i = 0; i<360; i++){

          float wave1 = map(tan(radians(frameCount*0.8+i)), -1, 1, -400, 400);
          float wave2 = map(tan(radians(frameCount+i)), -1, 1, -400,400);
          float c = map(tan(radians(frameCount*5)), -1,1,0,255);
          fill(c);
          rect(wave1, wave2, 20,20);


  }

break;
   }

}

void keyPressed(){
  if(amountOfTimesSpacePressed<20){
  amountOfTimesSpacePressed++;
  }

}
```

Code For the Stippling Art:

```
PImage cat, otter, fox,coyote,flower5,jaguar,cat2,geometry,flower9, fractal, buildings;

ArrayList<PImage> currentImages;
float amount, timesTheImagesChange;
int  amountOfTimesSpacePressed;

void setup(){
  amountOfTimesSpacePressed = 0;
   timesTheImagesChange = 0;
  amount = 3;

  size(900,900,P3D);
 // img = loadImage("cat.jpg");

 // img.resize(300, 300);

  // flower1.resize(300, 300);
   cat = loadImage("cat.jpg");
   cat.resize(300, 300);
   otter = loadImage("otter.jpg");
   otter.resize(300, 300);
   fox = loadImage("fox.jpg");
   fox.resize(300, 300);
   coyote = loadImage("coyote.jpg");
   coyote.resize(300, 300);
   jaguar = loadImage("jaguar.jpg");
   jaguar.resize(300, 300);
   cat2 = loadImage("cat2.jpg");
   cat2.resize(300, 300);
   geometry = loadImage("geometry.jpg");
   geometry.resize(300, 300);
   flower9 = loadImage("flower9.jpg");
   flower9.resize(300, 300);
   fractal = loadImage("fractal.jpg");
   fractal.resize(300,300);
   currentImages =  new ArrayList<PImage>();
   currentImages.add(fox);
   currentImages.add(cat);
   currentImages.add(otter);
   currentImages.add(coyote);
```

```
   currentImages.add(jaguar);
   currentImages.add(cat2);
   currentImages.add(geometry);
   currentImages.add(flower9);
   currentImages.add(fractal) ;
   buildings = loadImage("building.jpg");

}

void draw() {
   switch(amountOfTimesSpacePressed) {
 case 0:
  background(#E5FBFF);
  fill(0);
  noStroke();
  sphereDetail(5);
  //frameRate(1);


int i = int(random(0,currentImages.size()));
makeGrid(1, 1, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(1,-1,1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(1,-1,-1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(1,1,-1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 2, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 1, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 0, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2,1,2, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2,1,0, currentImages.get(i));
break;


case 1:
background(255);
amount = 1;
cat.resize(900, 900);
```

```
makeGrid(0,0,0,buildings);
break;
case 2:
background(255);

makeGrid(0,0,0,cat);
break;

case 3:
background(255);

makeGrid(0,0,0,cat);
break;


case 4:
background(255);

makeGrid2(0,0,0,cat);
break;
  }
}

void makeGrid(int e, int f,int g, PImage img){
        push();
 translate(e*width/amount,e*height/amount);
//  rotateY(radians(frameCount));
        float divisions = map(mouseX, 0, 900, 0, 400);
        float amountOfDevisions = (width/amount)/divisions;

        for (int x = 0; x < divisions; x++) {
        for (int y = 0; y < divisions; y++) {
        color c = img.get(int(x*amountOfDevisions),int(y*amountOfDevisions));
        float b = map(brightness(c),0,255,1,0);
        // float z = map(b,0,1,-150,150);
        push();

        translate(x*amountOfDevisions - (f*width/amount), y*amountOfDevisions -
(g*height/amount));
        fill(random(10), random(20),random(30));

        if(amountOfTimesSpacePressed==3){
        fill(random(0,255),random(0,255),random(0,255));
```

```
        }
        sphere(amountOfDevisions*b*0.8);
        pop();
  }
}
  pop();
}


void makeGrid2(int e, int f,int g, PImage img){
        push();
  translate(e*width/amount,e*height/amount);
//  rotateY(radians(frameCount));
        float divisions = map(mouseX, 0, 900, 0, 400);
        float amountOfDevisions = (width/amount)/divisions;

        for (int x = 0; x < divisions; x++) {
        for (int y = 0; y < divisions; y++) {
        color c = img.get(int(x*amountOfDevisions),int(y*amountOfDevisions));
        float b = map(brightness(c),0,255,0,1);
        // float z = map(b,0,1,-150,150);
        push();

        translate(x*amountOfDevisions - (f*width/amount), y*amountOfDevisions -
(g*height/amount));
        fill(random(10), random(20),random(30));

        if(amountOfTimesSpacePressed==3){
        fill(random(0,255),random(0,255),random(0,255));

        }
        sphere(amountOfDevisions*b*0.8);
        pop();
  }
}
  pop();
}

void keyPressed(){
  if(amountOfTimesSpacePressed<20){
  amountOfTimesSpacePressed++;
  }

}
```