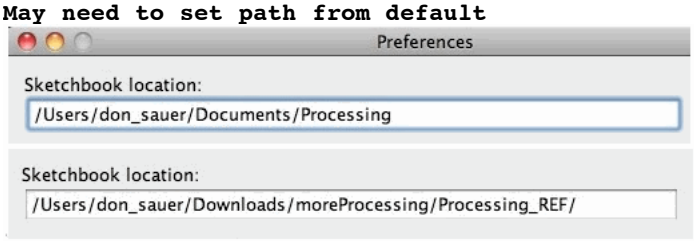
processing_templates

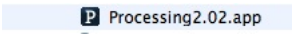
==========Copy_Paste_Code_2Run==========

The following code should run in a Processsing 2.02 environment

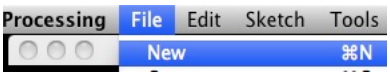
#0 ==========
May need to set path from default



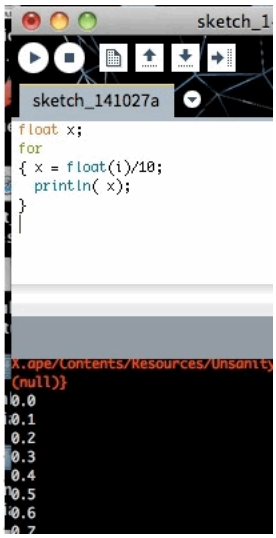
#1 ==========
Open Processing



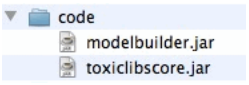
#2 ==========
Open New file



#3 ==========
Copy and Paste code into window and hit run



#4 ==========
Will be needing a code folder later



==========Calculate==========

```
float x;
for                              (int i=0; i <10;  i++)
{ x = float(i)/10;
  println( x);
}
```

```
==========result==========
0.1
0.2
0.3
```

...

==========CreatePwd==========

```
int b ;
randomSeed(                    58206);                // seed each time
for                            (int i=0; i <14;  i++)
{ b                          =  int(random(35, 123));
  if                           ( b== 39)  b = 38;
  if                           ( b== 94)  b = 75;
  if                           ( b== 95)  b = 103;
  if                           ( b== 96)  b = 119;
  print(                       char(b ));
}
```

==========result==========

```
Yn6u+P3L]ewa#
Tgl-m@9b4p@0I0
fyiDOYVww,Wo)[
X+r-X=k&E%P*kn
Tgl-m@9b4p@0I0
```

==========PRINT_ASCII==========

```
int x,y,z,w,u,v;
for                          (int i=33; i <53;  i++)
{ x                        = i;
  y                        = i+20;
  z                        = i+40;
  w                        = i+60;
  u                        = i+80;
  println(                  "i= "+x+" "+char(x)+"  i= "+y+" "+char(y)+"  i= "+z+" "+char(z)
                          +"  i= "+w+" "+char(w)+"  i= "+u+" "+char(u) );
}
```

==========result==========

```
i= 33 !  i= 53 5  i= 73 I  i= 93 ]  i= 113 q
i= 34 "  i= 54 6  i= 74 J  i= 94 ^  i= 114 r
i= 35 #  i= 55 7  i= 75 K  i= 95 _  i= 115 s
i= 36 $  i= 56 8  i= 76 L  i= 96 `  i= 116 t
i= 37 %  i= 57 9  i= 77 M  i= 97 a  i= 117 u
i= 38 &  i= 58 :  i= 78 N  i= 98 b  i= 118 v
i= 39 '  i= 59 ;  i= 79 O  i= 99 c  i= 119 w
i= 40 (  i= 60 <  i= 80 P  i= 100 d  i= 120 x
i= 41 )  i= 61 =  i= 81 Q  i= 101 e  i= 121 y
i= 42 *  i= 62 >  i= 82 R  i= 102 f  i= 122 z
i= 43 +  i= 63 ?  i= 83 S  i= 103 g  i= 123 {
i= 44 ,  i= 64 @  i= 84 T  i= 104 h  i= 124 |
i= 45 -  i= 65 A  i= 85 U  i= 105 i  i= 125 }
i= 46 .  i= 66 B  i= 86 V  i= 106 j  i= 126 ~
i= 47 /  i= 67 C  i= 87 W  i= 107 k  i= 127 □
i= 48 0  i= 68 D  i= 88 X  i= 108 l  i= 128 ?
i= 49 1  i= 69 E  i= 89 Y  i= 109 m  i= 129 ?
i= 50 2  i= 70 F  i= 90 Z  i= 110 n  i= 130 ?
i= 51 3  i= 71 G  i= 91 [  i= 111 o  i= 131 ?
i= 52 4  i= 72 H  i= 92 \  i= 112 p  i= 132 ?
```

==========Convert==========

```
String s1 = "00010000";          println(s1);              // Prints "00010000"
                                 println(unbinary(s1));    // Prints "16"
String s2 = "FF";                println(unhex(s2));       // Prints "255"
int i = 65;                      println(i);               // Prints "65"
                                 println(str(i));          // Prints "65"
                                 println(float(i));        // Prints "65.0"
                                 println(binary(i));       // Prints "00000000000000000000000001000001"
                                 println(char(i ));        // Prints "A"
String s = String.valueOf(i);    println(s);               // Prints "65"
byte b = 67;                     println(b);               // Prints "67"
                                 println(char(b ));        // Prints "C"
                                 println(int(b ));         // Prints "67"
                                 println(hex(b ));         // Prints "43"
                                 println(str(b));          // Prints "67"
char c = 'E';                    println(c);               // Prints "E"
                                 println(byte(c ));        // Prints "69"
                                 println(str(c));          // Prints "E"
float f = 32.6;                  println(f);               // Prints "32.6"
                                 println(int(f ));         // Prints "32"
float a = 90f;                   println(int(a ));         // Prints "90.0"
                                 println(PI);              // Prints "3.1415927"
                                 println(TWO_PI);          // Prints "6.2831855"
                                 println(nf(TWO_PI,1,3)); // Prints "6.283"
```

==========Math==========

```
println(abs(-3.33));           // 3.33
println(ceil(8.22));           // 9
println(constrain(100,30,70)); // 70
println(dist(0, 0, 1, 1));     // 1.4142135      dist(x1, y1, x2, y2) vector distance
println(exp(1.0));             // 2.7182817
println(floor(2.88));          // 2
println(lerp(0, 5, .3));       // 1.5            lerp(start, stop, amt)
println(log(2.7182817));       // 0.99999994
println(mag(1,1));             // 1.4142135
println(mag(1,1,1));           // 1.7320508
println(map(20, 0,60, 0, 1));  // 0.33333334     map(value, from_min, from_max, to_min, to_max)
println(max(5, 9));            // 9
println(max(-4, -12));         // -4
println(max(12.3, 230.24));    // 230.24
println(min(5, 9));            // 5
println(min(-4, -12));         // -12
println(min(12.3, 230.24));    // 12.3
println(norm(20,0,60));        // 0.33333334
println(pow(2,3));             // 8.0
```

```
println(round(9.2));          // 9
println(round(9.5));          // 10
println(sq(-5));              // 25.0
println(sqrt(625));           // 25.0
println(acos(0.9));           // 0.45102686
println(asin(.9));            // 1.1197695
println(atan(.9));            // 0.7328151
println(atan2(.6,.9));        // 0.5880026
println(cos(0.9));            // 0.62161
println(sin(.9));             // 0.7833269
println(tan(.9));             // 1.2601582
println(degrees(.9));         // 51.566196
println(radians(33.9));       // 0.59166664
```

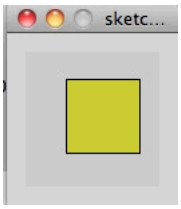
==========%(modulo)==========

```
println( 5 % 4);              // 1
println( 125 % 60);           // 5
println( 123.4567 % .01);     // 0.0067059454     123.45
//frame                       = (frame+1) % numFrames;    // Use % to cycle through frames
```

==========(bitwise OR)==========

```
int a = 205;                  // In binary: 11001101
int b = 45;                   // In binary: 00101101
int c = a | b;                 // In binary: 11101101
println(c);                   // In binary: 11101101  Prints "237"
```

==========(bitwise AND)==========




```
int a1 = 207;                 // In binary: 11001111
int b1 = 61;                  // In binary: 00111101
int c1 = a1 & b1;              // In binary: 00001101
println(c1);                  // In binary: 00001101 Prints "13"

color argb                  = color(204, 204, 51, 255);
int a                       = argb >> 24 & 0xFF;// "0xFF" is 00000000000000000000000011111111
int r                       = argb >> 16 & 0xFF;
int g                       = argb >> 8  & 0xFF;
int b                       = argb       & 0xFF;
fill(                         r, g, b, a);
rect(                         30, 20, 55, 55);
```

==========(right shift)==========

```
int m                       = 8   >> 3; println(m); // binary: 1000 to 1        // Prints "1"
int n                       = 256 >> 6; println(n); // binary: 100000000 to 100 // Prints "4"
int o                       = 16  >> 3; println(o); // binary: 10000 to 10      // Prints "2"
int p                       = 26  >> 1; println(p); // binary: 11010 to 1101    // Prints "13"
```

==========(left shift) ==========

```
int m                       = 1  << 3;  println(m); // binary: 1 to 1000        // Prints "8"
int n                       = 1  << 8;  println(n); // binary: 1 to 100000000   // Prints "256"
int o                       = 2  << 3;  println(o); // binary: 10 to 10000      // Prints "16"
int p                       = 13 << 1;  println(p); // binary: 1101 to 11010    // Prints "26"
```

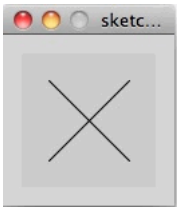
==========bigEndian==========

```
int hi                      = b[ix]   & 0xff;       // bytes -128 to 127, converts to unsigned...
int lo                      = b[ix+1] & 0xff;
int el                      = (int)((hi<<8)|lo);      // big endian!
```

==========bigEndianHiLo==========

```
int hi                       = b[ix]   & 0xff;       // bytes -128 to 127, converts to unsigned...
int lo                       = b[ix+1] & 0xff;
int el                       = (int)((hi<<8)|lo);      // big endian!
```

==========SAVE_PICTURE_FORMATS==========




```
line(20, 20, 80, 80);
save("diagonal.tif");                 // Saves a TIFF file named "diagonal.tif"
line(80, 20, 20, 80);
save("cross.tga");                    // Saves a TARGA file named "cross.tga"
                                      //TIFF (.tif), TARGA (.tga), JPEG (.jpg), or PNG (.png).

PImage img;
```

```
void setup()
{ img = loadImage("laDefense.jpg");   // Images must be in the "data" directory to load correctly
}
void draw()
{ image(img, 0, 0);                  // Loads an image into a variable of type PImage. Four types of images ( .gif, .jpg, .tga, .png)
}
```

======================Format_Numbers_For_Text_Files=============================

```
float number                = 123.456789;
float numb2                 = number - number % .01;   println( numb2 );        // 123.45
println(                      100000000*numb2);                                 // 1.23449999E10
System.out.printf(           "%7.2f",number);                                   // 123.46123.45679
String str1                 = str(number);             println( str1 );        // [0] "123.45679"
String[] str2               = split(str1, ' ');        println( str2 );        // [0] "123.45679"
saveStrings(                 "nouns2.txt",  str2);
String words                = "apple bear cat dog";
String[] list               = split(words, ' ');
saveStrings(                 "nouns.txt", list);                               // Writes strings to file on separate line
String a                    = "384762387642736842734"; println(a);            // 384762387642736842734
float f                     = Float.parseFloat(a);     println("float="+f);    // float=3.847624E20
f                           = float(a);                println("float="+f);    // float=3.847624E20
double d                    = Double.parseDouble(a);   println("double="+d);   // double=3.8476238764273684E20
int ii                      = int(f/10E15);            println("integer="+ii); // integer=38476
float numb3                 = f/10E15 ;
float numb4                 = numb3 - numb3 % .01 ;    println( numb4 );       // 38476.23
println(                      100000000*numb4);                                 // 3.84762302E12
```

======================keyCode=====================================

```
void                         draw()
{ if(keyPressed)
{ if(key     == 'w')         { println("key pressed is w ")    ;} //key pressed is w
  if(key     == 's')         { println("key pressed is " + key);} //key pressed is s
  if(keyCode == UP)          { println("key pressed is up " )  ;} //key pressed is up
}
}
```

======================keyReleased=================================

```
void                         draw()     {}         // Empty draw() needed to keep the program running
void                         keyPressed()
{ println(                  "pressed " +key + " " + int(key) + " " + keyCode ); //  pressed a 97 65
}
void                         keyTyped()
{ println(                  "typed "   +key + " " + int(key) + " " + keyCode);  // typed a 97 0
}
void                          keyReleased()
{ println(                   "released "+key + " " + int(key) + " " + keyCode);  // released a 97 65
}
```

======================keyPress_Case=================================

```
void                         keyPressed()
{ cnt++;
  if                         (cnt>14)cnt=0;
  switch                     (cnt)
{ case 0  :                  s= new DoubleCone(g,resolution,resolution);         break;
  case 1  :                  s= new EnnepersSurface(g,resolution,resolution);    break;
  case 2  :                  s= new FishSurface(g,resolution,resolution);        break;
  case 3  :                  s= new Horn(g,resolution,resolution);               break;
  case 4  :                  s= new JetSurface(g,resolution,resolution);         break;
  case 5  :                  s= new MoebiusStrip(g,resolution,resolution);       break;
  case 6  :                  s= new Pillow(g,resolution,resolution);             break;
  case 7  :                  s= new Shell(g,resolution,resolution);              break;
  case 8  :                  s= new SnailSurface(g,resolution,resolution);       break;
  case 9  :                  s= new Sphere(g,resolution,resolution);             break;
  case 10 :                  s= new Spring(g,resolution,resolution);             break;
  case 11 :                  // s= new SuperShapes(g,resolution,resolution);     break;
  case 12 :                  s= new TearDrop(g,resolution,resolution);           break;
  case 13 :                  s= new TetrahedralEllipse(g,resolution,resolution); break;
  case 14 :                  s= new WhitneyUmbrella(g,resolution,resolution);    break;
}
}
```

import java.awt.event.KeyEvent;

```
import java.awt.event.KeyEvent;

int[] ALFABET;

ALFABET = new int[27];
ALFABET[0]= KeyEvent.VK_A; // 65
println(ALFABET[0]);
ALFABET[0]= KeyEvent.VK_B; // 66
println(ALFABET[0])
char c = 'E';                  println(c);              // Prints "E"
                               println(byte(c ));       // Prints "69"
                               println(str(c));         // Prints "E"
```

======================HASHMAP=====================================

```
import                       java.util.Iterator;
import                       java.util.Map;
HashMap hm                  = new HashMap();
hm.put(                      "Ava", 1);
hm.put(                      "Cait", 35);
hm.put(                      "Casey", 36);
Iterator i                  = hm.entrySet().iterator();  // Get an iterator
```

```
while                          (i.hasNext())
{ Map.Entry me               = (Map.Entry)i.next();
  print(                       me.getKey() + " is ");
  println(                     me.getValue());
}
====================
//Ava is 1
//Cait is 35
//Casey is 36
```

======================STRING============================

```
String men                   = "Chernenko,Andropov,Brezhnev";
String[] list                = split(men, ',');              // list[0] is now "Chernenko", list[1] is "Andropov"...
String joinedNames           = join(list, " : ");
println(                       joinedNames);                 // Prints "Chernenko:Andropov:Brezhnev"

String numbers               = "8     67 5  309";
int[] nums                   = int(split(numbers, ' '));     // nums[0] is now 8, nums[1] is now 67...
String joinedNumbers         = join(nf(nums, 0), ", ");
println(                       joinedNumbers);               // Prints "8, 0, 0, 0, 0, 67, 5, 0, 309"

String s1                    = "    Somerville MA ";
String s2                    = trim(s1);
println(                       s1);                          // Prints "    Somerville   MA "
println(                       s2);                          // Prints "Somerville   MA"

String str1                  = "CCCP";
char   data[]                = {'C', 'C', 'C', 'P'};
String str2                  = new String(data);
println(                       str1);                        // Prints "CCCP" to the console
println(                       str2);                        // Prints "CCCP" to the console

String p                     = "potato";
if                             (p == "potato") {println("p == potato, yep.");  }    // This will not print
if                             (p.equals("potato")) {  println("Yes");    }         //Yes
String quoted                = "This one has \"quotes\"";
println(                        quoted);                                           // This one has "quotes"

String t                     = "a b";
String[] q                   = splitTokens(t);
println(                       q[0]);                          // Prints "a"
println(                       q[1]);                          // Prints "b"

String s                     = "a, b c ,,d ";
String[] q2                  = splitTokens(s, ", ");           // ", " means break whenever comma *or* a space
println(                       q2.length + " values found");  // Prints "4 values found"
println(                       q2[0]);  // Prints "a"         // multiple adjacent delimiters as a single break.
println(                       q2[1]);  // Prints "b"
println(                       q2[2]);  // Prints "c"
println(                       q2[3]);  // Prints "d"

int a                        = 200, b=40, c=90;
float d                      = 200.94, e = 40.2;
println( nf(a,10)+" "          +nf(b,5)+" "+nf(c,3)+" "+nf(d,10,4)+" "+nf(e, 5, 3)); //"0000000200 00040 090 0000000200.9400 00040.200"
float f                      = 42525.34343;
println(                       nfc(f, 2) );  // Prints "42,525.34"
int a2                       = 200, b2=-40, c2=90;
float d2                     = 200.94, e2 = -40.2;
println( nfs(a2, 10)+" "       +nfs(b2,5)+" "+nfs(c2,3)+" "+nfs(d2,10,4)+" "+nfs(e2,5,3)); //"0000000200 -00040  090  0000000200.9400 -
00040.200"
```

======================STRING2NUMB============================

```
float number                 = 123.456789;
float numb2                  = number - number % .01;   println( numb2 );        // 123.45
println(                       100000000*numb2);                                 // 1.23449999E10
System.out.printf(             "%7.2f",number);                                  // 123.46123.45679
String str1                  = str(number);             println( str1 );       // [0] "123.45679"
String[] str2                = split(str1, ' ');        println( str2 );       // [0] "123.45679"
saveStrings(                   "nouns2.txt",  str2);
String words                 = "apple bear cat dog";
String[] list                = split(words, ' ');
saveStrings(                   "nouns.txt", list);                               // Writes strings to file on separate line
String a                     = "384762387642736842734"; println(a);            // 384762387642736842734
float f                      = Float.parseFloat(a);     println("float="+f);    // float=3.847624E20
f                            = float(a);                println("float="+f);    // float=3.847624E20
double d                     = Double.parseDouble(a);   println("double="+d);   // double=3.8476238764273684E20
int ii                       = int(f/10E15);            println("integer="+ii); // integer=38476
                               println(PI);                                      // Prints "3.1415927"
                               println(TWO_PI);                                  // Prints "6.2831855"
                               println(nf(TWO_PI,1,3));                          // Prints "6.283"
```

======================CONTROL============================

```
for                            (int i = 0; i <= 10; i += 2)
{ if                           (i == 6)                          // If i is 60 skip to the next iteration, ############
{ continue;
} else
{ print(                       i + " ");                         // 0 2 4 8 10
}
} println();
int i                        = 0;
while                          (i < 8)
{ print(                       i + " ");                         //0 1 2 3 4 5 6 7
  i                          = i + 1;
} println();
int s                        = (i > 5)  ? 10 : 255;
println(                       "i = " + i + " s = " + s);       // i = 8 s = 10
char letter                  = 'B';
if                             (letter != 'A') println("Not_A"); // Not_A
switch(                        letter)
```

```
{ case 'A':                println("Alpha");   break;        // Does not execute
  case 'B':                println("Bravo");   break;        // Prints "Bravo"
  default :                println("Zulu") ;   break;        // Does not execute
}
```

====================TIME====================================

```
println(                   "month = "       + month());  //month = 5
println(                   "day = "         + day());    //day = 14
println(                   "year = "        + year());   //year = 2013
println(                   "hour = "        + hour());   //hour = 17
println(                   "minute = "      + minute()); //minute = 5
println(                   "second = "      + second()); //second = 50
println(                   "milliseconds = "+ millis()); //milliseconds = 371
float x;
for                        (int i=0; i <10;  i++)
{ x = float(i)/10;
  delay(4250);             // delay 4250msec
  println( x);
}
```

====================PROMPT====================================

```
int num                  = 0;
void                       setup()
{ size(                    200,200);
  println(                "Please press a number between 1-9"); //Please press a number between 1-9
}
void                       draw()
{ background(              255);
  if                       (num !=0)
{ println(                "This is Number "+ num);              //This is Number 3
  num                    = 0;
}
}
void                       keyPressed()
{ int                      keyNum = Character.getNumericValue(key);
  if                       (keyNum<=9 && keyNum>0)  num = keyNum;
}
```

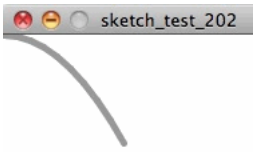
====================User_Input================================

ClickHere and type.
Hit return to save what you typed.

testing,one,two,three

```
PFont f;
String typing          = "";    // Variable to store text currently being typed
String saved           = "";    // Variable to store saved text when return is hit
void                       setup()
{ size(                    300,200);
  f                    = createFont("Arial",16,true);
}
void                       draw()
{ background(              255);
  int indent           = 25;
  textFont(                f);
  fill(0);
  text(                   "ClickHere and type. \nHit return to save what you typed. ", indent, 40);
  text(                    typing,indent,90);
  text(                    saved,indent,130);
}
void                       keyPressed()
{ if                       (key == '\n' )            // If the return key is pressed, save the String and clear it
{ saved                = typing;
  typing               = "";                       // A String can be cleared by setting it equal to ""
} else
{ typing               = typing + key;             // Otherwise, concatenate typed to end of String variable.
}
}
void                       mousePressed()
{ save(                   "pict.jpg");
}
```

====================PLOT_And_Save==========

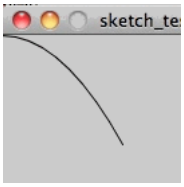


```
PrintWriter output;
float x0 = 0, y0=0;
float x1, y1;
void                                setup()
{ output                           = createWriter("commaSep.txt");   // Create a new file in the sketch directory
  size                               (200, 300);
  background                         (255);
  stroke                             (153);    // grey
  smooth                             ();
  strokeWeight(5);

  for                                (int i=0; i <10;  i++)
{ x1                               = float(i)*10;
  y1                               = x1* x1/100;
  line                               (x0, y0, x1, y1);
  output.println(                    x1 + "     ," + y1);
  x0                               = x1;
  y0                               = y1;
}
  output.flush();                   // Writes the remaining data to the file
  output.close();                   // Finishes the file
  exit();                           // Stops the program
}
```

==========commaSep.txt==========

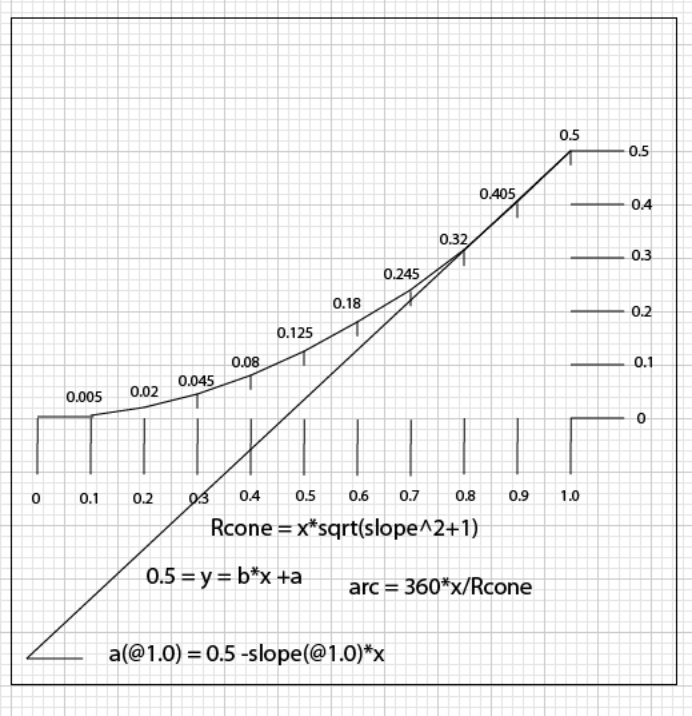
```
0.0     ,0.0
10.0    ,1.0
20.0    ,4.0
30.0    ,9.0
40.0    ,16.0
50.0    ,25.0
60.0    ,36.0
70.0    ,49.0
80.0    ,64.0
90.0    ,81.0
```

==========Comma_Separated_Text==========



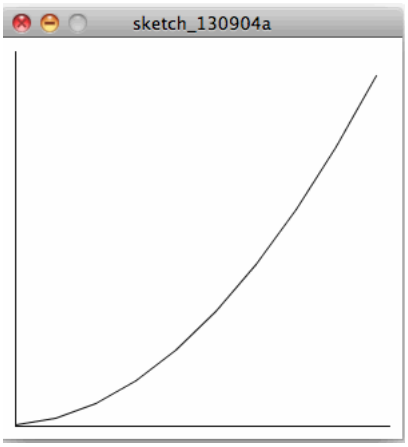

```
int[] data;
String[] grafpts;
float x0 = 0, y0=0;
float x1, y1;

void setup()
{ size(200,200);
  String[] stuff = loadStrings("/Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/data/commaSep.txt");
  //print (stuff.length);
  for (int i = 0; i < stuff.length; i ++ )
{ grafpts = split(stuff[i], ',' );
  println(grafpts[0]);
  x1                               = float(grafpts[0]);
  y1                               = float(grafpts[1]);;
  line                               (x0, y0, x1, y1);
  x0                               = x1;
  y0                               = y1;
}
}
```

==========Equations_Table==========



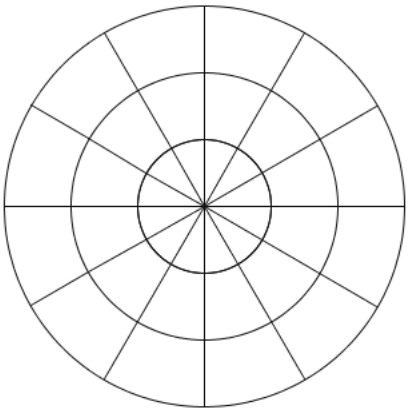

```
float x, y;
float focal =.5;
float xlast = 0 ;
float ylast = 0 ;
float len ;
float slope;
float Rcone;
float arc ;
float circm;
for                     (int i=0; i <=10;  i++)
{ x                   = float(i)/10;                                          // radius
  y                   = x*x/(4*focal);                                        // height at radius
  len                 = sqrt( (x- xlast)* (x- xlast) +(y- ylast)* (y- ylast));  // length to previous radius
  slope               = (y- ylast)/(x- xlast);                                // slope of length
  Rcone               = x*sqrt( slope*slope +1 );                             // outer cone radius
  arc                 = 360*x/Rcone;                                          // outer cone arc in degress
  circm               =  TWO_PI*x;                                            // circum at radius
  println(              "x="+x+" y="+nf(y,1,3) +" len="+nf(len,1,3) +" slope="+nf(slope,1,3)
                      + " Rcone="+nf(Rcone,1,3)+" arc="+nf(arc,1,3) +" circm="+nf(circm,1,3));
  xlast               = x ;
  ylast               = y ;
}
  println(              PI);
  println(              TWO_PI);
```

```
=========================Result============================================================
x=0.0 y=0.000 len=0.000 slope=?     Rcone=?     arc=?       circm=0.000
x=0.1 y=0.005 len=0.100 slope=0.050 Rcone=0.100 arc=359.551 circm=0.628
x=0.2 y=0.020 len=0.101 slope=0.150 Rcone=0.202 arc=356.017 circm=1.257
x=0.3 y=0.045 len=0.103 slope=0.250 Rcone=0.309 arc=349.251 circm=1.885
x=0.4 y=0.080 len=0.106 slope=0.350 Rcone=0.424 arc=339.789 circm=2.513
x=0.5 y=0.125 len=0.110 slope=0.450 Rcone=0.548 arc=328.292 circm=3.142
x=0.6 y=0.180 len=0.114 slope=0.550 Rcone=0.685 arc=315.438 circm=3.770
x=0.7 y=0.245 len=0.119 slope=0.650 Rcone=0.835 arc=301.840 circm=4.398
x=0.8 y=0.320 len=0.125 slope=0.750 Rcone=1.000 arc=288.000 circm=5.027
x=0.9 y=0.405 len=0.131 slope=0.850 Rcone=1.181 arc=274.298 circm=5.655
x=1.0 y=0.500 len=0.138 slope=0.950 Rcone=1.379 arc=261.000 circm=6.283
3.1415927
6.2831855
====================Parabola_Equations_Plot=========================================
=
```



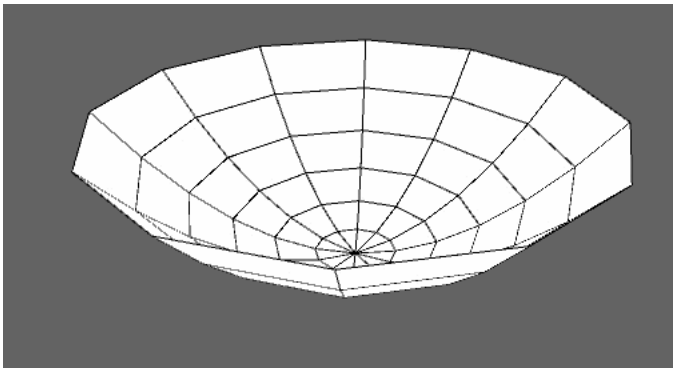

```
float x0=0, y0=0;
float x1, y1;
float xoff = 10;
float yoff = 100;

size(300, 300);
 background(255);
 stroke(15);                                // grey
 smooth();
 strokeWeight(1);
yoff = height-10 ;
x0 =  xoff;
y0 =  yoff;
line( xoff, 10,  xoff, height-10);
line( xoff,  height-10,  width-10, height-10);

for                                  (int i=0; i <10;  i++)
{ x1 = float(i)*30 + xoff ;
  y1 = -x1* x1/300 + yoff;
  line(x0, y0, x1, y1);
  x0=x1;
  y0=y1;
}
```

==========Draw_2D_segments==========



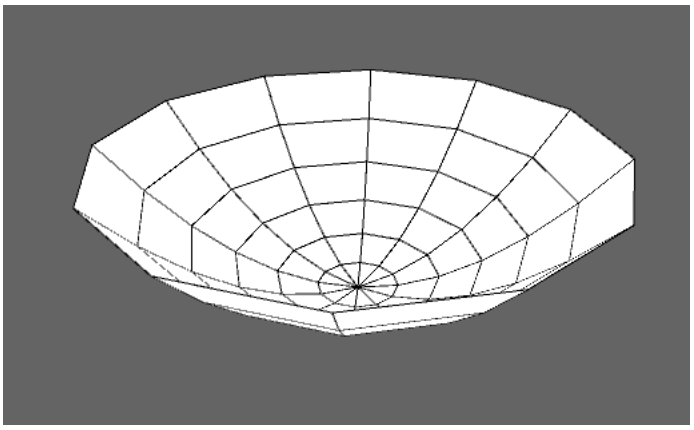
```
  float r               = 50;
  size(                   600, 600);
  background(             255);
  stroke(                 15);                              // grey
  smooth(                 );
  strokeWeight(           1);
  float cx              = width/2;
  float cy              = height/2;
  ellipse(                cx, cy , r*2, r*2);
  noFill(                 );                                 // NO FILL
  for                     (int j = 0; j < 3; j ++ )
{ ellipse(                cx, cy , r*2*(j+1), r*2*(j+1));
  for                     (int i = 0; i < 12; i ++ )
{ line(                   cx+r*j*cos(PI*i/6),cy+r*j*sin(PI*i/6),cx+r*(j+1)*cos(PI*i/6),cy+r*(j+1)*sin(PI*i/6) );
  println( PI*i/6);
}
}
```

==========Create_3D_Parabola==========



```
  PVector                 v1, v2 ,v3 , va,vb ,vc,vd;
  float                   x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
  float r               = 30;
  float focal           = 300;
  void                    setup()
{ size(                   600, 600, P3D);
  vd                    = FindNormal( 50.0, 50.0, 40.0 , 0.0,  50.0, 100.0,  0.0,  0.0,  80.0);
  println(                vd);            // [ 3000.0, -1000.0, 2500.0 ]
}
  void                    draw()
{ background(             100);
  pushMatrix(             );                        // save orginal set up
  translate(              300, 300, 0);             // translate origin to center
  rotateX(                PI-mouseY/150.0);
  rotateZ(                PI/2-mouseX/150.0);
  for                     (int j = 0; j < 6; j ++ )
{ for                     (int i = 0; i < 12; i ++ )
{ x1                    = r*j*cos(PI*i/6);
  y1                    = r*j*sin(PI*i/6);
  z1                    = sq((r*j))/focal;
  x2                    = r*j*cos(PI*(i+1)/6);
  y2                    = r*j*sin(PI*(i+1)/6) ;
  z2                    = sq((r*j))/focal;
  x3                    = r*(j+1)*cos(PI*(i+1)/6);
  y3                    = r*(j+1)*sin(PI*(i+1)/6);
  z3                    = sq((r*(j+1)))/focal;
  x4                    = r*(j+1)*cos(PI*i/6);
  y4                    = r*(j+1)*sin(PI*i/6) ;
  z4                    = sq((r*(j+1)))/focal;
  drawPlate(              x1,y1,z1,   x2,y2,z2,   x3,y3,z3,   x4,y4,z4);
}
} popMatrix();                                      // restore orginal set up
}
 void drawPlate(          float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3,float x4,float y4,float z4)
{ stroke(                 0);
  strokeWeight(           1);
  beginShape(             QUADS);
  vertex(                 x1, y1, z1);
  vertex(                 x2, y2, z2);
  vertex(                 x3, y3, z3);
  vertex(                 x4, y4, z4);
  endShape();
}

 PVector                  FindNormal(float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3)
{ v1                    = new PVector(x1, y1, z1);// println(v1); // [ 50.0, 50.0, 40.0 ]
  v2                    = new PVector(x2, y2, z2);// println(v2); // [ 0.0,  50.0, 100.0 ]
  v3                    = new PVector(x3 ,y3, z3);// println(v3); // [ 0.0,  0.0,  80.0 ]
  va                    = v1;
  va.sub(                 v2);                    // println(va); // [ 50.0, 0.0, -60.0 ]
  vb                    = v2;
  vb.sub(                 v3);                    // println(vb); // [ 0.0,  50.0, 20.0 ]
  vc                    = va.cross(vb);           // println(vc); // [ 3000.0, -1000.0, 2500.0 ]
  return                  vc;
}
```

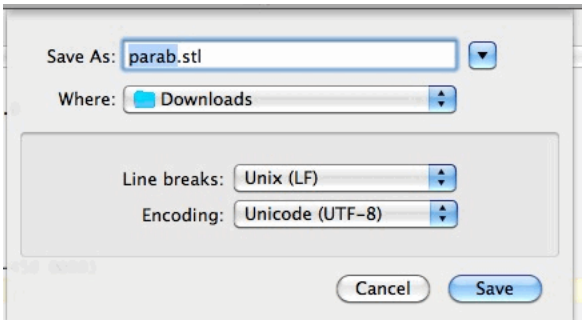
==========Create3DParabolaSTL==========

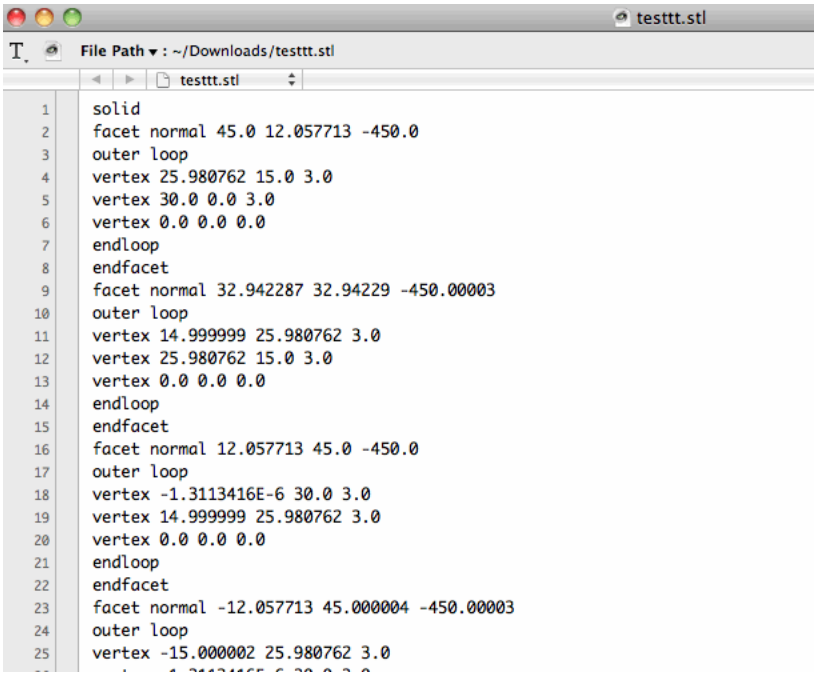


```
  PVector                v1, v2 ,v3 , va,vb ,vc,vd;
  float                  x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
  float r              = 30;
  float focal          = 300;
  void                   setup()
{ size(                  600, 600, P3D);
  println (              "solid");
  for                    (int j = 0; j < 6; j ++ )
{ for                    (int i = 0; i < 12; i ++ )
{ x1                 = r*j*cos(PI*i/6);
  y1                 = r*j*sin(PI*i/6);
  z1                 = sq((r*j))/focal;
  x2                 = r*j*cos(PI*(i+1)/6);
  y2                 = r*j*sin(PI*(i+1)/6) ;
  z2                 = sq((r*j))/focal;
  x3                 = r*(j+1)*cos(PI*(i+1)/6);
  y3                 = r*(j+1)*sin(PI*(i+1)/6);
  z3                 = sq((r*(j+1)))/focal;
  x4                 = r*(j+1)*cos(PI*i/6);
  y4                 = r*(j+1)*sin(PI*i/6) ;
  z4                 = sq((r*(j+1)))/focal;
  drawPlate(           x1,y1,z1,   x2,y2,z2,   x3,y3,z3,   x4,y4,z4);
  PrintSTL(            x1,y1,z1,   x2,y2,z2,   x3,y3,z3   );
  PrintSTL(            x3,y3,z3,   x4,y4,z4,   x1,y1,z1   );
}
}
  println (              "endsolid");
}
  void                   draw()
{ background(            100);
  pushMatrix(            );                        // save orginal set up
  translate(             300, 300, 0);             // translate origin to center
  rotateX(               PI-mouseY/150.0);
  rotateZ(               PI/2-mouseX/150.0);
  for                    (int j = 0; j < 6; j ++ )
{ for                    (int i = 0; i < 12; i ++ )
{ x1                 = r*j*cos(PI*i/6);
  y1                 = r*j*sin(PI*i/6);
  z1                 = sq((r*j))/focal;
  x2                 = r*j*cos(PI*(i+1)/6);
  y2                 = r*j*sin(PI*(i+1)/6) ;
  z2                 = sq((r*j))/focal;
  x3                 = r*(j+1)*cos(PI*(i+1)/6);
  y3                 = r*(j+1)*sin(PI*(i+1)/6);
  z3                 = sq((r*(j+1)))/focal;
  x4                 = r*(j+1)*cos(PI*i/6);
  y4                 = r*(j+1)*sin(PI*i/6) ;
  z4                 = sq((r*(j+1)))/focal;
  drawPlate(           x1,y1,z1,   x2,y2,z2,   x3,y3,z3,   x4,y4,z4);
}
} popMatrix();                                     // restore orginal set up
}
 void drawPlate(         float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3,float x4,float y4,float z4)
{ stroke(                0);
  strokeWeight(          1);
  beginShape(            QUADS);
  vertex(                x1, y1, z1);
  vertex(                x2, y2, z2);
  vertex(                x3, y3, z3);
  vertex(                x4, y4, z4);
  endShape(              );
}

 PVector                 FindNormal(float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3)
{ v1                 = new PVector(x1, y1, z1);// println(v1); // [ 50.0, 50.0, 40.0 ]
  v2                 = new PVector(x2, y2, z2);// println(v2); // [ 0.0,  50.0, 100.0 ]
  v3                 = new PVector(x3 ,y3, z3);// println(v3); // [ 0.0,  0.0,  80.0 ]
  va                 = v1;
  va.sub(                v2);                    // println(va); // [ 50.0, 0.0, -60.0 ]
  vb                 = v2;
  vb.sub(                v3);                    // println(vb); // [ 0.0,  50.0, 20.0 ]
  vc                 = va.cross(vb);             // println(vc); // [ 3000.0, -1000.0, 2500.0 ]
  return                 vc;
}
```

```
 void                    PrintSTL(float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3)
{ vd                  =  FindNormal( x1,y1,z1,x2,y2,z2,x3,y3,z3 );
  if                      ( vd.x != 0 && vd.y != 0 && vd.z != 0 )
{ println(                "facet normal " + vd.x + " " + vd.y + " " + vd.z );
  println (               "outer loop ");
  println(                "vertex " + x1 + " " + y1 + " " + z1 );
  println(                "vertex " + x2 + " " + y2 + " " + z2 );
  println(                "vertex " + x3 + " " + y3 + " " + z3 );
  println (               "endloop");
  println (               "endfacet");
}
}
```

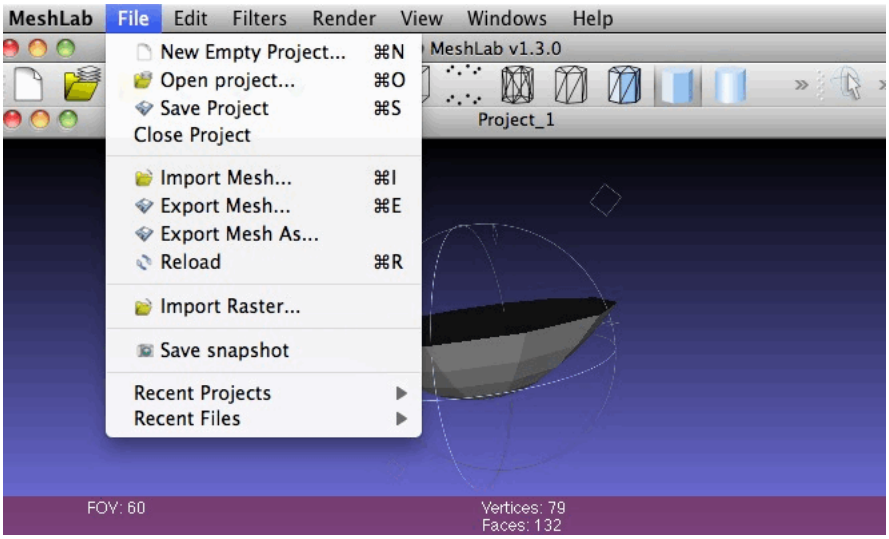
==========Copy_Output_To_A_File==========



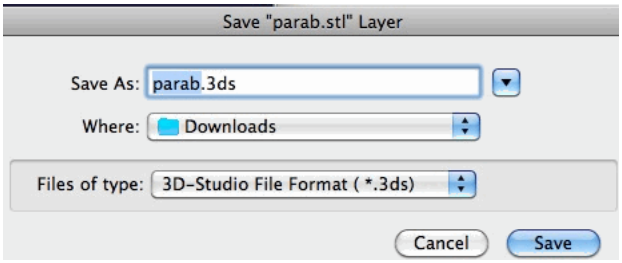



```
solid
facet normal 45.0 12.057713 -450.0
outer loop
vertex 25.980762 15.0 3.0
vertex 30.0 0.0 3.0
vertex 0.0 0.0 0.0
endloop
endfacet
facet normal 32.942287 32.94229 -450.00003
outer loop
vertex 14.999999 25.980762 3.0
vertex 25.980762 15.0 3.0
vertex 0.0 0.0 0.0
endloop
endfacet
facet normal 12.057713 45.0 -450.0
outer loop
vertex -1.3113416E-6 30.0 3.0
vertex 14.999999 25.980762 3.0
vertex 0.0 0.0 0.0
endloop
endfacet
facet normal -12.057713 45.000004 -450.00003
outer loop
vertex -15.000002 25.980762 3.0
```

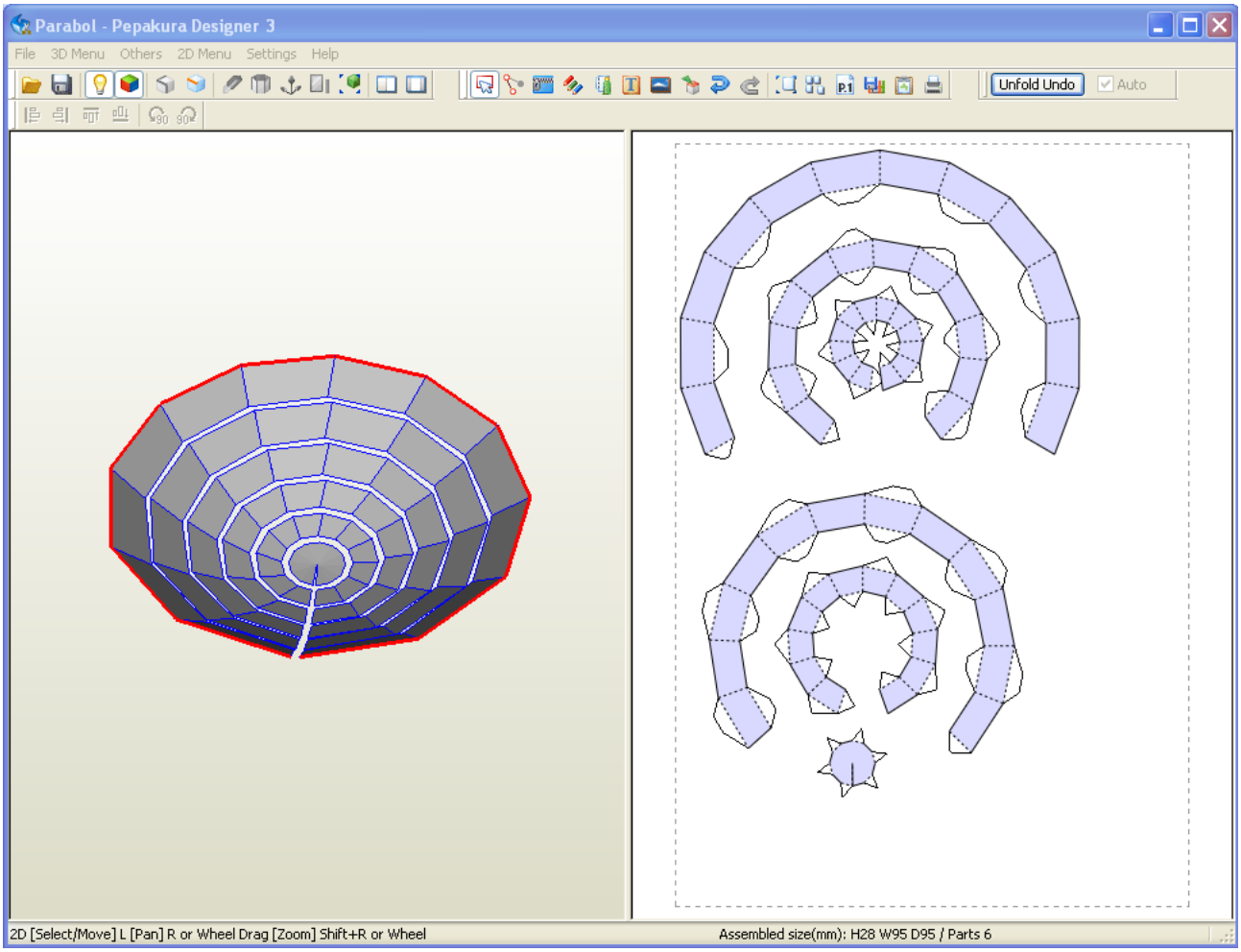
==========Import_Mesh==========



==========Convert_to_3DS==========



==========Can_Built_3ds_files==========



==========ConvertSTL2SACSII==========

```
ruby /Users/don_sauer/Downloads/2DO/sketch_130802a/convertSTL.rb /Users/don_sauer/Downloads/sketch_130815a/ribbon.stl

/Users/don_sauer/Downloads/sketch_130815a/ribbon.stl is in BINARY format,
converting to ASCII: /Users/don_sauer/Downloads/sketch_130815a/ribbon-ascii.stl
```

==========ARRAY==========

```
int[] numbers               = new int[2];
numbers[0]                  = 90;
numbers[1]                  = 150;
int[] numbers2              = { 90, 150, 30 };
numbers2                    = sort( numbers2 ) ;        println(numbers2);     // { 30, 90,150 }
String[] north              = { "OH", "IN", "MI"};      println(north);        // north[0] = "OH"
String[] sa1                = append(north , "MA");     println(sa1);          //{"OH", "IN", "MI" , "MA"}
String[] sa2                = shorten(sa1);             println(sa2);          //{"OH", "IN", "MI"}
String[] a                  = splice(sa2, "KY", 1);     println(a);            //{"OH", "KY", "IN", "MI"} Splice one value into
String[] south              = { "GA", "FL", "NC"};      println(south);        //{ "GA", "FL", "NC"}
String[] sa3                = concat(north, south);     println(sa3);          //{"OH", "IN", "MI","GA", "FL", "NC" }
String[] sa4                = subset(sa3, 1 , 2);       println(sa4);          //{ "IN", "MI" }
//String[] sa5              = arrayCopy(north, south); println(sa5);           //{ "OH", "IN", "MI"}
int[] data                  = {0, 1, 3, 4};             println(data.length);  // Prints "4"
data                        = expand(data);             println(data.length);  // Prints "8"
data                        = expand(data, 512);        println(data.length);  // Prints "512"
String sa[]                 = { "OH", "NY", "MA", "CA"};
sa                          = reverse(sa);              println(sa); //{ "CA", "MA", "NY", "OH"}
```

==========Print-1D-Array==========

```
int[] myArray           = {0,1,2,3};
println(myArray);

String[] images         = { "jewels.jpg", "bark.jpg", "dp.jpg", "jeff.jpg"};
for                       (int i=0;i<images.length;i++) println("images " +i+"  " +images[i]);
=========================================
[0] 0
[1] 1
[2] 2
[3] 3
images 0  jewels.jpg
```

```
images 1  bark.jpg
images 2  dp.jpg
images 3  jeff.jpg
```

====================Print-2D-Array=====================

```
int[][] myArray2 = { {0,1,2,3}, {3,2,1,0}, {3,5,6,1} };

int[][] myArray = {  {0, 1, 2, 3},
                     {3, 2, 1, 0},
                     {3, 8, 3, 4}  };
int     cols = myArray.length;
int     rows = myArray[0].length;
  for (int i = 0; i < cols; i++)
{ for (int j = 0; j < rows; j++)
{ print( myArray[i][j] + " ");
}
  println(" ");
}
```

```
====================================
0 1 2 3
3 2 1 0
3 8 3 4
```

====================PRINT-3D_ARRAYS=====================

```
  int   n                      = 3;
  double[][][]                   Array3D = new double[n][n][n];
  for                           (int i=0; i < Array3D.length;         i++)
{ for                           (int j=0; j < Array3D[i].length;      j++)
{ for                           (int k=0; k < Array3D[i][j].length;   k++)
{ Array3D[i][j][k]            = i*n*n + j *n  + k;     // initialize contents with double
}                                                      // for(int k=0; k < Array3D[i][j].length;   k++)
}                                                      // for(int j=0; j < Array3D[i].length;      j++)
}                                                      // for(int i=0; i < Array3D.length;         i++)
  println(                      "// Array3D[i][j][k]");
  print(                        "// \n");
  for                           (int i=0; i < Array3D.length;        i++)
{ print(                        "// i=" + i );
  for                           (int k=0; k < Array3D[i][0].length;   k++) { print( "   k=" + k ); }
  print(                        "\n");
  for                           (int j=0; j < Array3D[i].length;      j++)
{ print(                        "// j=" + j );
  for                           (int k=0; k < Array3D[i][j].length;   k++)
{ print(                        "   " + Array3D[i][j][k] );
} print(                        "\n");                              // for(int k=0; k < Array3D[i][j].length;   k++)
} print(                        "// \n");                           // for(int j=0; j < Array3D[i].length;      j++)
}                                                                  // for(int i=0; i < Array3D.length;         i++)
```

```
====================================
// Array3D[i][j][k]
//
// i=0   k=0   k=1   k=2
// j=0   0.0   1.0   2.0
// j=1   3.0   4.0   5.0
// j=2   6.0   7.0   8.0
//
// i=1   k=0    k=1    k=2
// j=0   9.0    10.0   11.0
// j=1   12.0   13.0   14.0
// j=2   15.0   16.0   17.0
//
// i=2   k=0    k=1    k=2
// j=0   18.0   19.0   20.0
// j=1   21.0   22.0   23.0
// j=2   24.0   25.0   26.0
//
```

====================PRINTARRAYS=====================

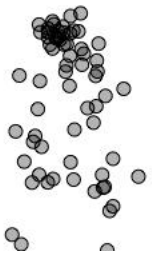
```
void                            Print1DI( int[] array )           //################################
{ for                           (int i = 0; i < array.length; i++)
  println                       ("i="+i+" val= "+ array[i]);
}
void                            Print1DS( String[]  array )       //################################
{ for                           (int i = 0; i < array.length; i++)
  println                       ("i="+i+" val= "+ array[i]);
}
void                            Print2DF( float[][] array )       //################################
{ for                           (int i = 0; i < array.length;    i++)
{ for                           (int j = 0; j < array[0].length; j++)
{ print                         (array[i][j]+" ");
}
  println();
}
}
```

====================ArrayListFunctions=====================

```
ArrayList words                 = new ArrayList<String>(50);
words.add(                      "Java");
words.add(                      "is");
println(                        "Current size: " + words.size( ));                  //Current size: 2
words.add(                      "a");
words.add(                      "powerful");
words.add(                      "Last word? ");                                     //Java is a powerful Last word?
println(                        "\nTraverse by index");                           //Traverse by index
for                             (int i=0; i<5; i++)
{ print(                        words.get(i) + " ");                               //Java is a powerful Last word?
```

```
}
print(                  "\n\n");
println(                "Reverse traverse by index");                    //Reverse traverse by index
for                     (int i=words.size( )-1; i>=0; i--)
{ print(                words.get(i) + " ");                             //Last word?  powerful a is Java
}
print(                  "\n\n");
println(                "Add values");                                   //Add values
words.add(0,            "Sun's");
words.add(4,            "very");
words.add(6,            "programming");
for                     (int i=0; i<=words.size( )-1; i++)
{ print(                words.get(i) + " ");                             //Sun's Java is a very powerful programming Last
word?
}
print(                  "\n\n");
println(                "Remove and size 1");                                  //Remove and size 1
println(                "Size: " + words.size( ));                       // Size: 8
words.remove(           0);
for                     (int i=0; i<=words.size( )-1; i++)
{ print(                words.get(i) + " ");                             //Java is a very powerful programming Last word?
}
print(                  "\n\n");
println(                "Remove and size 2");                            //Remove and size 2
println(                "Size: " + words.size( ));                       //Size: 7
words.remove(           3);
words.remove(           4);
for                     (int i=0; i<=words.size( )-1; i++)
{ print(                words.get(i) + " ");                             //Java is a powerful Last word?
}
print(                  "\n\n");
println(                "Set");                                          // Set
words.set(              0, "Suns's Java");
words.set(              4, "programming language");
words.add(              5, "but I prefer Captain Crunch");
for                     (int i=0; i<=words.size( )-1; i++)
{ print(                words.get(i) + " ");                             // Suns's Java is a powerful programming language but
I prefer Captain Crunch
}
```

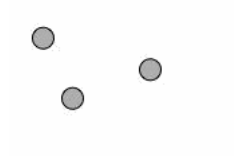
====================PARTICLE_ARRAYLIST==========================



```
ArrayList               particles;
void                    setup()
{ size(                 200,200);
  particles             = new ArrayList();
  smooth(               );
}
void                    draw()
{ particles.add(        new Particle());                        // Particle object added to ArrayList every cycle draw().
  background(           255);
  for                   (int i = 0; i < particles.size(); i++ )
{ Particle p            = (Particle) particles.get(i);          // Iterate ArrayList and get each Particle
  p.run(                );
  p.gravity(            );
  p.display(            );
}
  if                    (particles.size()>100){particles.remove(0);} // If ArrayList more than 100 delete first element
}
class                   Particle                                // A simple Particle class
{ float                 x, y,xspeed,yspeed;
Particle(               )
{ x                     = mouseX;
  y                     = mouseY;
  xspeed                = random(-1,1);
  yspeed                = random(-2,0);
}
void                    run()
{ x                     = x + xspeed;
  y                     = y + yspeed;
}
void                    gravity()
{ yspeed                += 0.1;
}
void                    display()
{ stroke(               0);
  fill(                 0,75);
  ellipse(              x,y,10,10);
```

```
}
}
void                          mousePressed()
{ save(                      "pict.jpg");
}
```

==========AppendArray==========

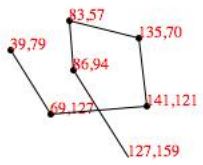


```
Ball[] balls             = new Ball[1];              // We start with an array with just one element.
float gravity            = 0.1;
void                       setup()
{ size(                    200,200);
  smooth(                  );
  balls[0]               = new Ball(50,0,16);        // Initialize ball index 0
}
void                       draw()                    // Update and display all balls
{ background(              255);
  for                      (int i = 0; i < balls.length; i ++ )
{ balls[i].gravity(        );
  balls[i].move(           );
  balls[i].display(        );
}}
void                       mousePressed()                 // A new ball object
{ Ball b                 = new Ball(mouseX,mouseY,16);    // Make a new object at the mouse location.
  balls                  = (Ball[]) append(balls,b);      // append() element to the end of the array.
}
class                      Ball
{ float                    x,y,speed, w;
Ball(                     float tempX, float tempY, float tempW)
{ x                      = tempX;
  y                      = tempY;
  w                      = tempW;
  speed                  = 0;
}
void                       gravity()
{ speed                  = speed + gravity;              // Add gravity to speed
}
void                       move()
{ y                      = y + speed;                      // Add speed to y location
  if                       (y > height)                      // If square reaches the bottom// Reverse speed
{ speed                  = speed * -0.95;
  y                      = height;
}}
void                       display()
{ fill(                    175);                           // Display the circle
  stroke(                  0);
  ellipse(                 x,y,w,w);
}}
void                       keyPressed()
{ save(                    "pict.jpg");
}
```

==========SAVEJPG==========

```
void                        keyPressed()
{ save(                    "pict.jpg");
  if                        (key=='s') {save( "pict.jpg");}
}


void                        mousePressed()
{ save(                    "pict.jpg");
}

if                         (mousePressed) { ca.drawBoxAt(mouseX,mouseY,5,1); }
```

==========CLICK_VIEW==========

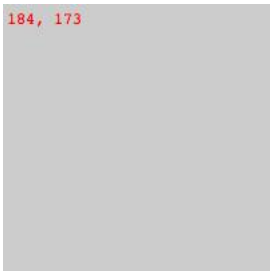



```
1 39  79
2 69  127
3 141 121
4 135 70
5 83  57
6 86  94
```

```
PFont                   font;
String                  str1,str2;
ArrayList pt          = new ArrayList();
void                    setup()
{ size(                 200, 200);
  font                =  createFont("Arial",9,true);
  //font              = loadFont("/Users/don_sauer/Downloads/more/sketch_131105a/data/Serif-12.vlw");
  textFont(             font,12);
  fill(                 255, 0, 0);
}
void                    draw()
{ strokeWeight(         5);
  background(           255);
  str1                = String.valueOf(mouseX) +","+String.valueOf(mouseY);
  text(                 str1, mouseX,  mouseY);
  if                    (pt != null)
{ for                   (int i = 0; i < pt.size(); i++ )
{ XYpt p              = (XYpt) pt.get(i);
  point(                p.x, p.y );
  str2                = String.valueOf(int(p.x)) +","+String.valueOf(int(p.y));
  text(                 str2, p.x,  p.y);
} strokeWeight(         1);
  for                   (int i = 1; i < pt.size(); i++ )
{ XYpt p1             = (XYpt) pt.get(i-1);
  XYpt p2             = (XYpt) pt.get(i);
  line(                 p1.x, p1.y , p2.x, p2.y);
} if                    (pt.size()>0)
{ XYpt p3             = (XYpt) pt.get(pt.size()-1);
  line(                 p3.x, p3.y , mouseX, mouseY);
}}
  if                    (mousePressed )
{ pt.add(               new XYpt(mouseX, mouseY));
  println(              pt.size()+" "+ mouseX+" "+mouseY);
  delay(                550);
}}
void                    keyPressed()
{ //int                 keyNum = Character.getNumericValue(key);
  //if                  (keyNum<=9 && keyNum>0)  num = keyNum;
  save(                 "pict.jpg");
}
class                   XYpt
{ float                 x;
  float                 y;
XYpt(                  float xin, float yin)
{ x                   = xin;
  y                   = yin;
}}
```

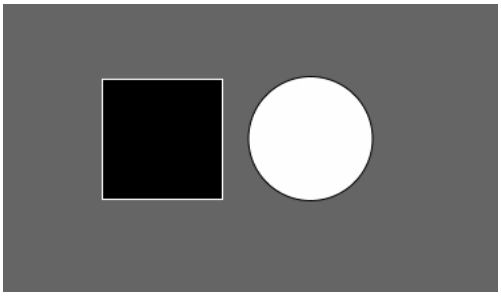
```
====================
1 39  79
2 69  127
3 141 121
4 135 70
5 83  57
6 86  94
```

==========Display-Mouse==========




```
PFont metaBold;
String str1;
void setup()
{ size(200, 200);
  //metaBold = loadFont("CourierNew36.vlw");
  metaBold =  createFont("Arial",9,true);
  textFont(metaBold);
  stroke(0);
  fill(255, 0, 0);
  textSize(12);
}
void draw()
{ background(204);
  if (mousePressed )
{ str1 =str(mouseX)+ ", "+ str(mouseY) ;
  text(str1, 3, 15);
}
}
void keyPressed() { save("pict.jpg"); }
```

==========MOUSE_OVER==========



```
int     rectX, rectY;                   // Position of square button
int     circleX, circleY;               // Position of circle button
int     rectSize               = 90;    // Diameter of rect
int     circleSize             = 93;    // Diameter of circle
boolean rectOver               = false;
boolean circleOver             = false;
color   rectColor, circleColor, baseColor;
color   rectHighlight, circleHighlight;
color   currentColor;
void                             setup()
{ size(                          640, 360);
  rectColor                    = color(0);
  rectHighlight                = color(51);
  circleColor                  = color(255);
  circleHighlight              = color(204);
  baseColor                    = color(102);
  currentColor                 = baseColor;
  circleX                      = width/2+circleSize/2+10;
  circleY                      = height/2;
  rectX                        = width/2-rectSize-10;
  rectY                        = height/2-rectSize/2;
  ellipseMode(                   CENTER);
}
void                             draw()
{ update(                        mouseX, mouseY);
  background(                    currentColor);
  if                             (rectOver)   { fill(rectHighlight);}
  else                           { fill(rectColor); }
  stroke(                        255);
  rect(                          rectX, rectY, rectSize, rectSize);
  if                             (circleOver) { fill(circleHighlight) ; }
  else                           { fill(circleColor);  }
  stroke(                        0);
  ellipse(                       circleX, circleY, circleSize, circleSize);
}
void                             update(int x, int y)
{ if                             ( overCircle(circleX, circleY, circleSize) )
{ circleOver                   = true;
  rectOver                     = false;
} else if                        ( overRect(rectX, rectY, rectSize, rectSize) )
{ rectOver                     = true;
  circleOver                   = false;
} else                           {   circleOver = rectOver = false; }
}
void                             mousePressed()
{ if                           (circleOver) {   currentColor = circleColor;  }
  if                           (rectOver)   {   currentColor = rectColor;    }
}
boolean                          overRect(int x, int y, int width, int height)
{ if                             (mouseX >= x && mouseX <= x+width &&   mouseY >= y && mouseY <= y+height)
{ return                         true; }
  else                           { return false; }
}
boolean                          overCircle(int x, int y, int diameter)
{ float disX                   = x - mouseX;
  float disY                   = y - mouseY;
  if                             (sqrt(sq(disX) + sq(disY)) < diameter/2 ) {  return true; }
  else                           { return false; }
}
```

==========pushed-pulled-by-cursor==========

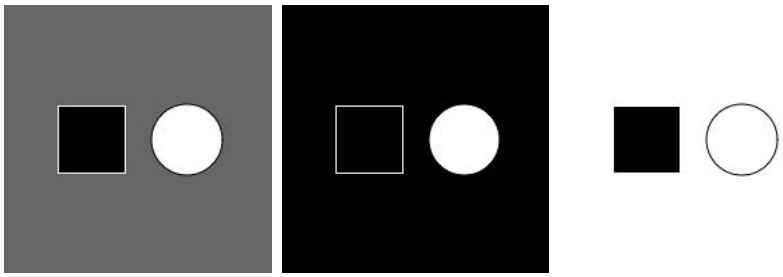
Follow 1. Based on code from Keith Peters (www.bit-101.com).
A line segment is **pushed-pulled-by-cursor**.

```
float x = 100;
float y = 100;
float angle1 = 0.0;
float segLength = 50;

void setup()
{ size(200, 200);
  smooth();
```

```
  strokeWeight(20.0);
  stroke(0, 100);
}

void draw()
{ background(226);
  float dx = mouseX - x;
  float dy = mouseY - y;
  angle1 = atan2(dy, dx);
  x = mouseX - (cos(angle1) * segLength);
  y = mouseY - (sin(angle1) * segLength);
  segment(x, y, angle1);
  ellipse(x, y, 20, 20);
}

void segment(float x, float y, float a)
{ pushMatrix();
  translate(x, y);
  rotate(a);
  line(0, 0, segLength, 0);
  popMatrix();
}
```

==========Rollover-button==========

Roll over the colored squares in the center of the image to change the color of the outside rectangle.



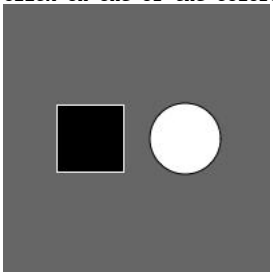
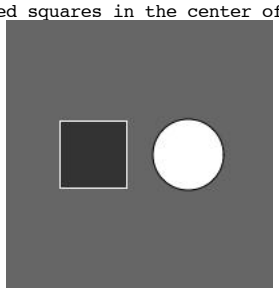
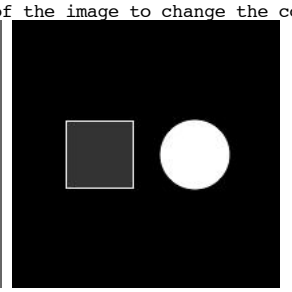
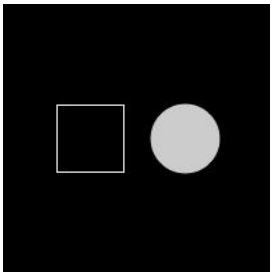
```
int rectX, rectY;      // Position of square button
int circleX, circleY;  // Position of circle button
int rectSize = 50;     // Diameter of rect
int circleSize = 53;   // Diameter of circle

color rectColor;
color circleColor;
color baseColor;

boolean rectOver = false;
boolean circleOver = false;

void setup()
{ size(200, 200);
  smooth();
  rectColor  = color(0);
  circleColor = color(255);
  baseColor   = color(102);
  circleX     = width/2+circleSize/2+10;
  circleY     = height/2;
  rectX       = width/2-rectSize-10;
  rectY       = height/2-rectSize/2;
  ellipseMode(CENTER);
}
void draw()
{ update(mouseX, mouseY);
  noStroke();
  if (rectOver)  { background(rectColor); }
  else if (circleOver) { background(circleColor); }
  else { background(baseColor); }
  stroke(255);
  fill(rectColor);
  rect(rectX, rectY, rectSize, rectSize);
  stroke(0);
  fill(circleColor);
  ellipse(circleX, circleY, circleSize, circleSize);
}
void update(int x, int y)
{ if( overCircle(circleX, circleY, circleSize) )
{ circleOver = true;
  rectOver   = false;
}
  else if ( overRect(rectX, rectY, rectSize, rectSize) )
{ rectOver   = true;
  circleOver = false;
}
  else {   circleOver = rectOver = false;  }
}
boolean overRect(int x, int y, int width, int height)
{ if (mouseX >= x && mouseX <= x+width &&
      mouseY >= y && mouseY <= y+height)   { return true;  }
  else                                     { return false; }
}
boolean overCircle(int x, int y, int diameter)
{ float disX = x - mouseX;
  float disY = y - mouseY;
```

```
  if(sqrt(sq(disX) + sq(disY)) < diameter/2 ) { return true; }
  else                                        { return false; }
}
void keyPressed() { save("pict.jpg"); }
```

==========BUTTON_OVER==========

Button.

Click on one of the colored squares in the center of the image to change the color of the background.

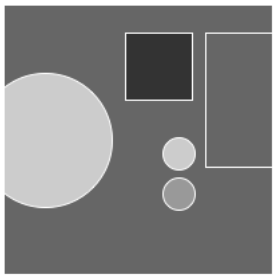





```
int     rectX, rectY;      // Position of square button
int     circleX, circleY;  // Position of circle button
int     rectSize = 50;     // Diameter of rect
int     circleSize = 53;   // Diameter of circle
color   rectColor, circleColor, baseColor;
color   rectHighlight, circleHighlight;
color   currentColor;
boolean rectOver = false;
boolean circleOver = false;

void setup()
{ size(200, 200);
  smooth();
  rectColor = color(0);
  rectHighlight = color(51);
  circleColor = color(255);
  circleHighlight = color(204);
  baseColor = color(102);
  currentColor = baseColor;
  circleX = width/2+circleSize/2+10;
  circleY = height/2;
  rectX = width/2-rectSize-10;
  rectY = height/2-rectSize/2;
  ellipseMode(CENTER);
}
void draw()
{ update(mouseX, mouseY);
  background(currentColor);
  if(rectOver)          { fill(rectHighlight); }
  else                  { fill(rectColor); }
  stroke(255);
  rect(rectX, rectY, rectSize, rectSize);
  if(circleOver)        { fill(circleHighlight); }
  else                  { fill(circleColor); }
  stroke(0);
  ellipse(circleX, circleY, circleSize, circleSize);
}
void update(int x, int y)
{ if( overCircle(circleX, circleY, circleSize) )
{ circleOver = true;
  rectOver = false;
}
  else if ( overRect(rectX, rectY, rectSize, rectSize) )
{ rectOver = true;
  circleOver = false;
} else   { circleOver = rectOver = false; }
}

void mousePressed()
{ if(circleOver) { currentColor = circleColor; }
  if(rectOver)   { currentColor = rectColor; }
}
boolean overRect(int x, int y, int width, int height)
{ if (mouseX >= x && mouseX <= x+width &&
      mouseY >= y && mouseY <= y+height) { return true; }
  else                                   { return false; }
}
```

```
boolean overCircle(int x, int y, int diameter)
{ float disX = x - mouseX;
  float disY = y - mouseY;
  if(sqrt(sq(disX) + sq(disY)) < diameter/2 ) {    return true; }
  else                                        {    return false;  }
}

void keyPressed() { save("pict.jpg"); }
```

==========Buttons==========

Click on one of the shapes to change the background color. This example demonstates a class for buttons.



```
color        currentcolor;
CircleButton circle1, circle2, circle3;
RectButton   rect1, rect2;
boolean      locked = false;

void setup()
{ size(200, 200);
  smooth();
  color baseColor   = color(102);
  currentcolor      = baseColor;
  color buttoncolor = color(204);// Define and create circle button
  color highlight   = color(153);
  ellipseMode(CENTER);
  circle1     = new CircleButton(30, 100, 100, buttoncolor, highlight);
  buttoncolor = color(204);// Define and create circle button
  highlight   = color(153);
  circle2     = new CircleButton(130, 110, 24, buttoncolor, highlight);
  buttoncolor = color(153);// Define and create circle button
  highlight   = color(102);
  circle3     = new CircleButton(130, 140, 24, buttoncolor, highlight);
  buttoncolor = color(102);// Define and create rectangle button
  highlight   = color(51);
  rect1       = new RectButton(150, 20, 100, buttoncolor, highlight);
  buttoncolor = color(51);// Define and create rectangle button
  highlight   = color(0);
  rect2       = new RectButton(90, 20, 50, buttoncolor, highlight);
}
void draw()
{ background(currentcolor);
  stroke(255);
  update(mouseX, mouseY);
  circle1.display();
  circle2.display();
  circle3.display();
  rect1.display();
  rect2.display();
}
void update(int x, int y)
{ if(locked == false)
{ circle1.update();
  circle2.update();
  circle3.update();
  rect1.update();
  rect2.update();
}
  else { locked = false; }
  if(mousePressed)
{ if(circle1.pressed())      { currentcolor = circle1.basecolor; }
  else if(circle2.pressed()) { currentcolor = circle2.basecolor; }
  else if(circle3.pressed()) { currentcolor = circle3.basecolor; }
  else if(rect1.pressed())   { currentcolor = rect1.basecolor;   }
  else if(rect2.pressed())   { currentcolor = rect2.basecolor;   }
}
}
class Button
{ int x, y;
  int size;
  color basecolor, highlightcolor;
  color currentcolor;
  boolean over = false;
  boolean pressed = false;

  void update()
{ if(over()) { currentcolor = highlightcolor; }
  else { currentcolor = basecolor; }
}
  boolean pressed()
{ if(over)
{ locked = true;
  return true;
}
```

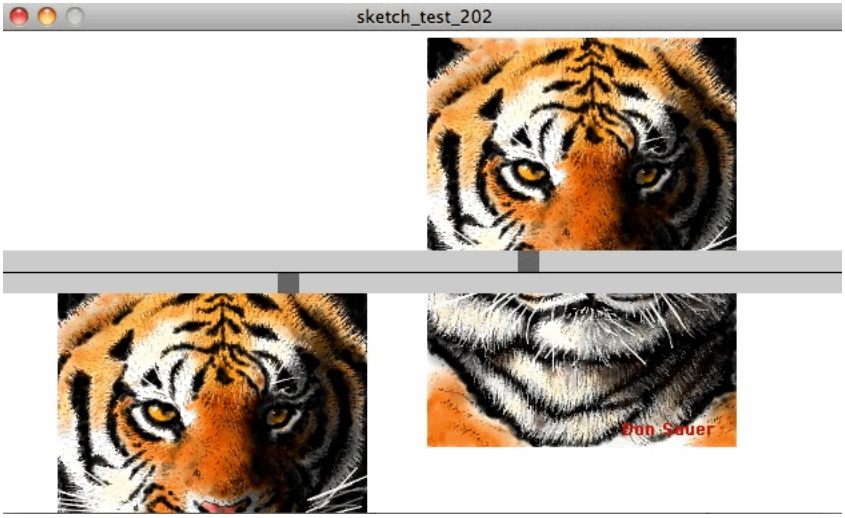
```
  else
{ locked = false;
  return false;
}
}
  boolean over() { return true; }
  boolean overRect(int x, int y, int width, int height)
{ if (mouseX >= x && mouseX <= x+width &&
  mouseY >= y && mouseY <= y+height) { return true;}
  else { return false; }
}
  boolean overCircle(int x, int y, int diameter)
{ float disX = x - mouseX;
  float disY = y - mouseY;
  if(sqrt(sq(disX) + sq(disY)) < diameter/2 ) { return true;}
  else { return false;}
}
}
class CircleButton extends Button
{ CircleButton(int ix, int iy, int isize, color icolor, color ihighlight)
{ x = ix;
  y = iy;
  size = isize;
  basecolor = icolor;
  highlightcolor = ihighlight;
  currentcolor = basecolor;
}
  boolean over()
{ if( overCircle(x, y, size) )
{ over = true;
  return true;
}
  else
{ over = false;
  return false;
}
}
  void display()
{ stroke(255);
  fill(currentcolor);
  ellipse(x, y, size, size);
}
}
class RectButton extends Button
{ RectButton(int ix, int iy, int isize, color icolor, color ihighlight)
{ x = ix;
  y = iy;
  size = isize;
  basecolor = icolor;
  highlightcolor = ihighlight;
  currentcolor = basecolor;
}
  boolean over()
{ if( overRect(x, y, size, size) )
{ over = true;
  return true;
}
  else
{ over = false;
  return false;
}
}
  void display()
{ stroke(255);
  fill(currentcolor);
  rect(x, y, size, size);
}
}
```

==========ReDraw==========

```
float y;

void setup()
{ size(640, 360);  // Size should be the first statement
  stroke(255);     // Set line drawing color to white
  noLoop();
  y = height * 0.5;
}
void draw()
{ background(0);     // Set the background to black
```

```
  y = y - 4;
  if (y < 0) { y = height; }
  line(0, y, width, y);
}

void mousePressed() {  redraw();}
```

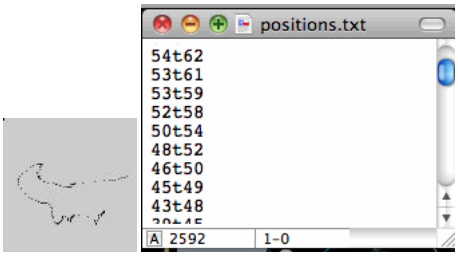
======================Scrollbar==============================




```
HScrollbar hs1, hs2;  // Two scrollbars
PImage img1, img2;    // Two images to load

void setup()
{ size(640, 360);
  noStroke();
  hs1 = new HScrollbar(0, height/2-8, width, 16, 16);
  hs2 = new HScrollbar(0, height/2+8, width, 16, 16);
  img1 = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");// Load images
  img2 = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");
}
void draw()
{ background(255);
  float img1Pos = hs1.getPos()-width/2;    // Get position of img1 scrollbar to display the img1 image
  fill(255);
  image(img1, width/2-img1.width/2 + img1Pos*1.5, 0);
  float img2Pos = hs2.getPos()-width/2;    // Get position of img2 scrollbar to display the img2 image
  fill(255);
  image(img2, width/2-img2.width/2 + img2Pos*1.5, height/2);
  hs1.update();
  hs2.update();
  hs1.display();
  hs2.display();
  stroke(0);
  line(0, height/2, width, height/2);
}
class HScrollbar
{ int swidth, sheight;    // width and height of bar
  float xpos, ypos;       // x and y position of bar
  float spos, newspos;    // x position of slider
  float sposMin, sposMax; // max and min values of slider
  int loose;              // how loose/heavy
  boolean over;           // is the mouse over the slider?
  boolean locked;
  float ratio;
HScrollbar (float xp, float yp, int sw, int sh, int l)
{ swidth = sw;
  sheight = sh;
  int widthtoheight = sw - sh;
  ratio = (float)sw / (float)widthtoheight;
  xpos = xp;
  ypos = yp-sheight/2;
  spos = xpos + swidth/2 - sheight/2;
  newspos = spos;
  sposMin = xpos;
  sposMax = xpos + swidth - sheight;
  loose = l;
}
void update()
{ if (overEvent())
{ over = true;
} else { over = false;}
  if (mousePressed && over)    { locked = true;}
  if (!mousePressed)           { locked = false;}
  if (locked)                  { newspos = constrain(mouseX-sheight/2, sposMin, sposMax);}
  if (abs(newspos - spos) > 1) { spos = spos + (newspos-spos)/loose;}
}
float constrain(float val, float minv, float maxv)
{ return min(max(val, minv), maxv);
}
```

```
boolean overEvent()
{ if (mouseX > xpos && mouseX < xpos+swidth && mouseY > ypos && mouseY < ypos+sheight)
{ return true;
} else
{ return false;
}
}
void display()
{ noStroke();
  fill(204);
  rect(xpos, ypos, swidth, sheight);
  if (over || locked)
{ fill(0, 0, 0);
} else
{ fill(102, 102, 102);
} rect(spos, ypos, sheight, sheight);
}
float getPos()
{ return spos * ratio;// Convert spos to be between 0 and width of scrollbar
}
}
```

==========createWriter==========




```
PrintWriter output;
void                    setup()
{ output                = createWriter("positions.txt");   // Create a new file in the sketch directory
}
void                    draw()
{ point(                mouseX, mouseY);
  output.println(       mouseX + "t" + mouseY);          // Write the coordinate to the file
}
void                    keyPressed()
{ output.flush();       // Writes the remaining data to the file
  output.close();       // Finishes the file
  exit();               // Stops the program
}
void                    mouseClicked()
{ save(                 "pict.jpg");                     // Mouse has been clicked
}
```

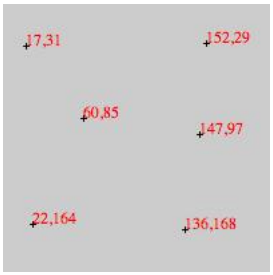
==========MOUSE_MIDDLE==========

```
import                  java.awt.event.*;
void                    setup()
{                        addMouseWheelListener( new MouseWheelListener()
{ public void           mouseWheelMoved(MouseWheelEvent mwe) { mouseWheel(mwe.getWheelRotation()); } }  );
}
void                    draw()
{ if                    (mousePressed == true) { println("mouse button pressed");}      // mouse button pressed
}
void                    mouseWheel(int delta)
{ println(              "mouse wheel has moved by " + delta + " units.");               //mouse wheel has moved by 1 units.
}
void                    mousePressed()
{ if                    (mouseButton == LEFT)  {println("mouse Left button ");   }      // mouse Left button
  if                    (mouseButton == RIGHT) {println("mouse Right button ");   }      // mouse Right button
}
void                    mouseReleased()
{ println(              "MouseX = "+ mouseX + " MouseY= " + mouseY );                   // MouseX = 24 MouseY= 93
}
void                    mouseMoved()
{ println(              "previousMouseX = "+ pmouseX + " previousMouseY= " + pmouseY );  // previousMouseX = 93 previousMouseY= 24

}
void                    mouseClicked()
{ println(              "Mouse has been clicked");                                      // Mouse has been clicked
}
void                    mouseDragged()
{ int dx                = mouseX - pmouseX;
  int dy                = mouseY - pmouseY;
  println(              "deltaMouseX = "+ dx + " deltaMouseY= " + dy );                 // deltaMouseX = 2 deltaMouseY= 4
}
```

==========cursor_types==========


```
void                    draw()
{ if                   (mouseX < 50)
{ cursor(               CROSS);
} else
{ cursor(               HAND);
}
}
void                    keyPressed()
{ save(                "pict.jpg");
}
```

====================MOUSE_LOCATION==================================




```
PFont                   font;
String                  str1,str2;
void                    setup()
{ size(                 200, 200);
  //font              = loadFont("/Users/don_sauer/Downloads/more/sketch_131105a/data/Serif-12.vlw");
  font              = createFont("Arial",9,true);
  stroke(               0);
  fill(                 255, 0, 0);
  textFont(             font,12);
}
void                    draw()
{ if                    (mousePressed == true)
{ println(              "mouse button pressed");}      // mouse button pressed
}
void                    mousePressed()
{ str1              =   String.valueOf(mouseX) +","+String.valueOf(mouseY);
  text(                 str1, mouseX,  mouseY);
  stroke (              0,0,255);
  line(                 mouseX -2, mouseY , mouseX +2, mouseY);
  line(                 mouseX , mouseY -2, mouseX, mouseY +2);
}
void                    keyPressed()
{ save(                "pict.jpg");
}
```

====================saveBytes==============================

```
  byte[]    nums          = { 0, 34, 5, 127, 52};
  String[]  list          = str(nums);
  int[]     num2          = int(nums);
  println(                  nums);
  println(                  list);
  println(                  num2);
  for                       (int i=0; i<nums.length;i++) { print(list[i]+"  ");  }
  println();
  for                       (int i=0; i<nums.length;i++) { list[i] =hex(nums[i]);}
  for                       (int i=0; i<nums.length;i++) { print(list[i]+"  ");  }
  saveBytes(                "numbers.dat", nums);        // now write the bytes to a file
```

```
==================

[0] 0
[1] 34
[2] 5
[3] 127
[4] 52
[0] "0"
[1] "34"
[2] "5"
[3] "127"
[4] "52"
[0] 0
[1] 34
[2] 5
[3] 127
[4] 52
0   34  5  127  52
00  22  05 7F   34
```

==========LoadBytes==========

```
byte b[]                  = loadBytes("numbers.dat");        // open a file and read its binary data
for                         (int i = 0; i < b.length; i++)   // print each value, from 0 to 255
{ if                        ((i % 10) == 0) {  println();  } // every tenth number, start a new line
  int a                   = b[i] & 0xff;                     // bytes are from -128 to 127, this converts to 0 to 255
  print(                    a + " ");                        // 0 34 5 127 52
}
println();                                                   // print a blank line at the end
```

```
0  00 22 05 7F 34
```

==========saveStrings==========

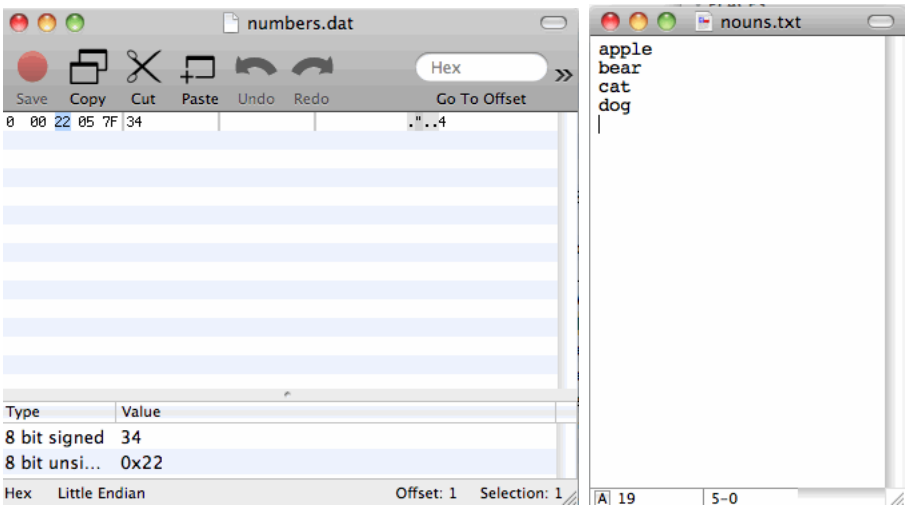
```
String words              = "apple bear cat dog";
String[] list             = split(words, ' ');
saveStrings(                "nouns.txt", list);        // now write the strings to a file, each on a separate line
```

==========loadStrings==========

```
String lines[]            = loadStrings("nouns.txt");
println(                    "there are " + lines.length + " lines");
for                         (int i=0; i < lines.length; i++) { println(lines[i]); }
```

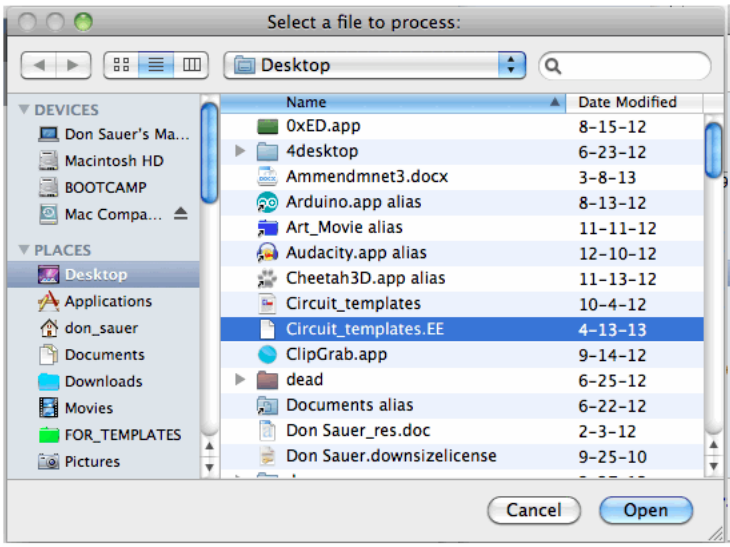
```
====================
there are 4 lines
apple
bear
cat
dog
```

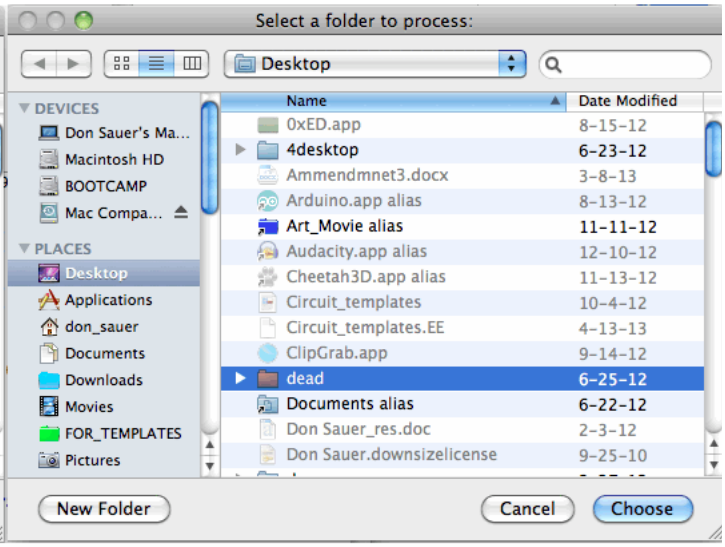
==========FILE_READ/WRITE==========

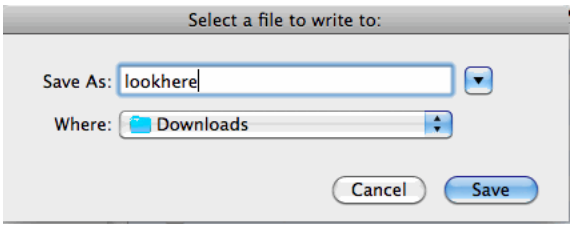



```
  byte[] nums             = { 0, 34, 5, 127, 52};
  saveBytes(                  "numbers.dat", nums);              // Writes the bytes to a file
  byte b[]                = loadBytes("numbers.dat");          // Open a file and read its binary data
  for                         (int i = 0; i < b.length; i++)
{ if                          ((i % 10) == 0) {   println();  }  // Every tenth number, start a new line
  int a                   = b[i] & 0xff;                       // bytes from -128 to 127, converts to 0 to 255
  print(                      a + " ");
} println();                                          // 0 34 5 127 52

  String words            = "apple bear cat dog";
  String[] list           = split(words, ' ');
  saveStrings(                "nouns.txt", list);                                  // Writes strings to file on separate line
  String lines[]          = loadStrings("nouns.txt");
  println(                   "there are " + lines.length + " lines");            // there are 4 lines
  for                         (int i = 0 ; i < lines.length; i++) { println(lines[i]);} // {apple, bear, cat, dog}


  String lines2[]         = loadStrings("http://processing.org/about/index.html");
  println(                   "there are " + lines2.length + " lines2");                  // there are 136 lines2
  for                         (int i = 0 ; i < lines2.length; i++) { println(lines2[i]);}
```

==========FILEFOLDERSSELECT==========

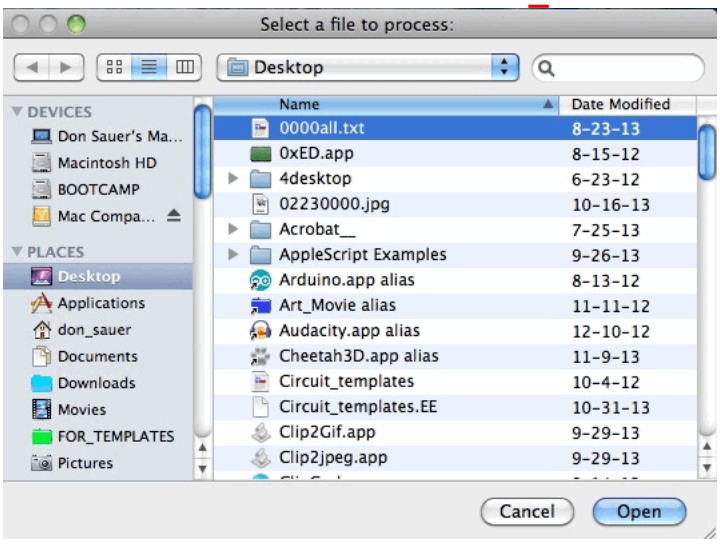







```
void                    setup()
{ selectInput(          "Select a file to process:"  , "fileSelected");
  selectFolder(         "Select a folder to process:", "folderSelected");
  selectOutput(         "Select a file to write to:" , "fileSelected2");
}
void                    fileSelected(File selection)
{ if                    (selection == null) {    println("Window was closed or the user hit cancel."); }
  else                  {println("User selected " + selection.getAbsolutePath());  }          //User selected /Users/don_sauer/
Desktop/Circuit_templates.EE
}
 void                   folderSelected(File selection2)
{ if                    (selection2 == null) {   println("Window was closed or the user hit cancel.");  }
   else                 {println("User selected " + selection2.getAbsolutePath());  }         //User selected /Users/don_sauer/
Desktop/dead
}

void                    fileSelected2(File selection3)
{ if                    (selection3 == null) { println("Window was closed or the user hit cancel."); }
  else                  {println("User selected " + selection3.getAbsolutePath());}          //User selected /Users/don_sauer/
Downloads/lookhere
}
```

==========fileSelection_2.02==========

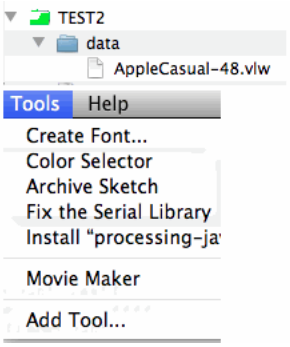


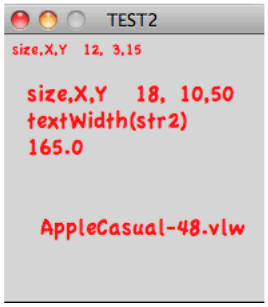

```
 void                   setup()
{ selectInput(      "Select a file to process:", "fileSelected");
}
void                    fileSelected(File selection)
{ if                    (selection == null)
```

```
{ println(          "Window was closed or the user hit cancel.");
} else
{ println(          "User selected " + selection.getAbsolutePath());
}
}
```

```
User selected /Users/don_sauer/Desktop/0000all.txt
```

==========TEXT_CREATE_FONT==========



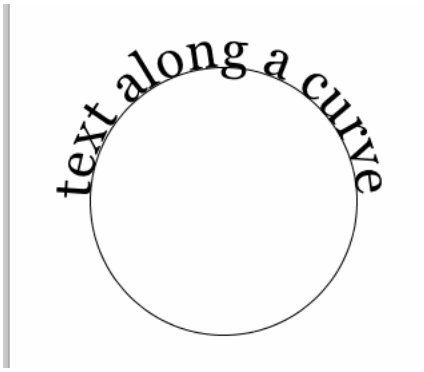



```
PFont                   metaBold;
String                  str1,str2;
void                    setup()
{ size(                   200, 200);
  str1                  = "AppleCasual-48.vlw";
  metaBold              = loadFont(str1);   // need in directory
  textFont(                 metaBold);
  stroke(                   0);
  fill(                     255, 0, 0);
  textSize(                 12);
  str2 =                    " size,X,Y  12, 3,15";
  text(                     str2, 3, 15);
  textSize(                 18);
  str2                  = " size,X,Y  18, 10,50";
  text(                     str2, 10, 50);
  float w               = textWidth(str2);
  text(                     String.valueOf(w),20, 90);
  text(                    "textWidth(str2)",20, 70);
  text(                     str1, 30, 150);
}
```

==========TEXT==========

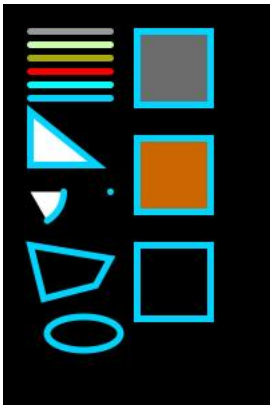

```
PFont                   NewFont;
String                  str2;
void                    setup()
{ size(                   200, 200);
  NewFont               = createFont("Arial",24,true);
  str2                  = "size,X,Y  12, 3,15";
  stroke(                   0);
  fill(                     255, 0, 0);
  text(                     str2, 13, 115);
}
```

==========text along a curve==========




```
// Learning Processing
// Daniel Shiffman
// http://www.learningprocessing.com

// Example 17-8: Characters along a curve

String                  message = "text along a curve";
PFont                   f;
float                   r = 100;          // The radius of a circle

void                    setup()
{ size(                 320,320);
  f = createFont(       "Georgia",40,true);
  textFont(             f);
  textAlign(            CENTER);// The text must be centered!
  smooth();
}
void                    draw()
{ background(           255);
  translate(            width/2, height/2);// Start in the center and draw the circle
  noFill();
  stroke(               0);
  ellipse(              0, 0, r*2, r*2);
  float arclength       = 0;                                  // keep track of position along curve
  for                   (int i = 0; i < message.length(); i ++ )  // For every box
{ char currentChar      = message.charAt(i);                  // The character and its width  ############
  float w               = textWidth(currentChar);             // check the width of each character.
  arclength            += w/2;                                // Each box is centered so we move half the widt
  float theta           = PI + arclength / r;                 // Angle Starting left side of circle by adding PI
  pushMatrix();
  translate(            r*cos(theta), r*sin(theta));          // Polar to Cartesian conversion
  rotate(               theta + PI/2);                        // Rotate box (rotation is offset by 90 degrees)
  fill(0);
  text(                 currentChar,0,0);                     // Display the character
  popMatrix();
  arclength            += w/2;                                // Move halfway again
}
}
```

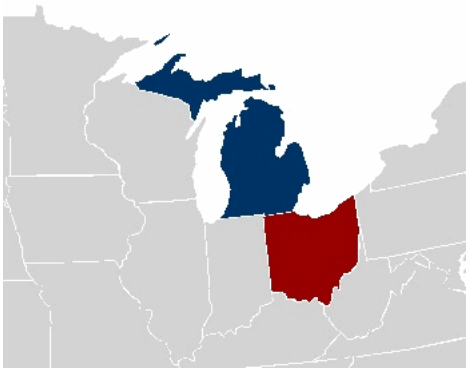
==========STROKE==========



```
 size(200, 300);
 background(0);
 stroke(153);                           // grey
 smooth();
 strokeWeight(5);
 stroke(153);
 line(20, 20, 80, 20);
 stroke(#CCFFAA);                       // RGB
 line(20, 30, 80, 30);
 stroke(0xAAFFFF11);                    // RGBA
 line(20, 40, 80, 40);
 stroke(255, 0, 0);                     // RED
 line(20, 50, 80, 50);
 colorMode(RGB, 1);                     // COLOR 0->1
 stroke(0, 1, 1);                       // CYAN
 line(20, 60, 80, 60);
 colorMode(HSB, 360, 100, 100);         // HSB ANGLE and PERCENT
```

```
 stroke(190, 100, 100);                    // HSB
 line(20, 70, 80, 70);
 triangle(20, 80, 20,120, 70, 120);        // normal fill white with  3 x,y points
 arc(20, 140, 50, 50, 0, PI/3);            // x y xrad yrad start stop
 point(80, 140);                           // X,Y
 noFill();                                 // NO FILL
 quad(20, 180, 80, 190, 69, 210, 30, 220);
 ellipse(60, 246, 55, 25);                 // X,Y xrad yrad
 fill(153);
 rect(100, 20, 55, 55);                    // X, Y, H, W
 colorMode(RGB, 255);
 fill(204, 102, 0);
 rect(100, 100, 55, 55);                   // X, Y, H, W
 noFill();
 rect(100, 180, 55, 55);                   // X, Y, H, W

 save("pict.jpg");
```

==========SVG==========

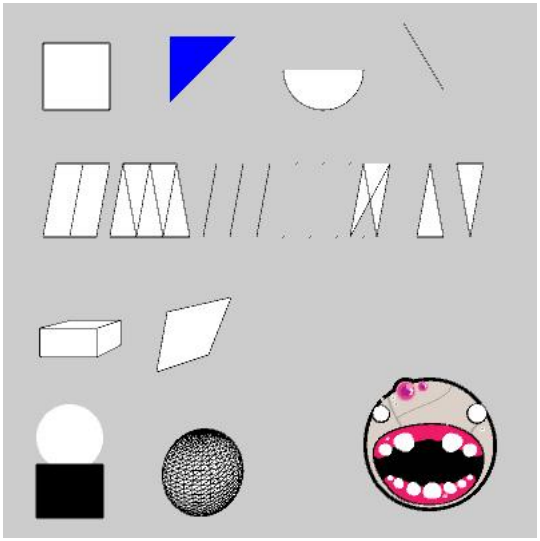


```
PShape usa;
PShape michigan;
PShape ohio;

void setup()
{ size(640, 360);
  usa = loadShape("/Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/data/usa-wikipedia.svg");
  michigan = usa.getChild("MI");
  ohio = usa.getChild("OH");
}

void draw()
{ background(255);
  shape(usa, -600, -180);      // Draw the full map
  michigan.disableStyle();     // Disable the colors found in the SVG file
  fill(0, 51, 102);            // Set our own coloring
  noStroke();
  shape(michigan, -600, -180); // Wolverines! // Draw a single state
  ohio.disableStyle();         // Disable the colors found in the SVG file
  fill(153, 0, 0);             // Set our own coloring
  noStroke();
  shape(ohio, -600, -180);     // Buckeyes!// Draw a single state
}
```

usa-wikipedia.svg

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 12.0.0, SVG Export Plug-In . SVG Version: 6.00 Build 51448)  -->
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" [
        <!ENTITY ns_svg "http://www.w3.org/2000/svg">
        <!ENTITY ns_xlink "http://www.w3.org/1999/xlink">
]>
<svg  version="1.1"
         id="svg2" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:cc="http://web.resource.org/cc/" xmlns:inkscape="http://www.inksc
namespaces/inkscape" xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd" sodipodi:version="0.32" xmlns:svg="http://www
2000/svg" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" sodipodi:docname="Blank US Map.svg" inkscape:version="0.44"
sodipodi:docbase="/home/theshibboleth/Desktop/Wiki/election maps"
         xmlns="&ns_svg;" xmlns:xlink="&ns_xlink;" width="1368" height="936" viewBox="0 0 1368 936"
         overflow="visible" enable-background="new 0 0 1368 936" xml:space="preserve">
<sodipodi:namedview  fill="#d3d3d3" inkscape:window-x="0" inkscape:window-y="25" showguides="true" inkscape:current-layer="svg2" inksc
bbox="true" showgrid="false" inkscape:window-height="796" inkscape:window-width="1430" inkscape:pageshadow="2" inkscape:pageopacity="0
id="base" pagecolor="#ffffff" borderopacity="1.0" bordercolor="#666666" inkscape:cx="374.60758" inkscape:zoom="0.99999999"
inkscape:cy="340.17222">
        </sodipodi:namedview>
<path id="HI" fill="#D3D3D3" d="M521.087,690.311l.939-3.55712.263-0.32310.324,0.8091-2.102,3.071H521.087L521.087,690.31z
         M531.271,686.59116.143,2.58712.103-0.32311.617-3.881-0.647-3.3961-4.203-0.4841-4.042,1.777L531.271,686.591z M561.987,696.613
        13.718,5.49612.426-0.32311.132-0.48311.455,1.29313.718-0.16210.97-1.4541-2.91-1.7781-1.939-3.7191-2.103-3.5581-5.82,2.91
        L561.987,696.613z M582.194,705.50611.293-1.9414.688,0.9710.646-0.48316.144,0.6461-0.323,1.2931-2.587,1.4551-4.365-0.323
        L582.194,705.506z M587.529,710.67911.94,3.8813.071-1.13210.323-1.6161-1.617-2.1031-3.718-0.322V710.679L587.529,710.679z
         M594.48,709.54712.264-2.90914.688,2.42514.365,1.13114.364,2.749v1.9391-3.557,1.7771-4.85,0.971-2.426-1.454L594.48,709.547z
         M611.133,725.06511.616-1.29313.396,1.61617.598,3.55713.396,2.10311.616,2.42511.94,4.36414.042,2.5881-0.323,1.2931-3.88,3.232
        1-4.203,1.4551-1.455-0.6461-3.071,1.7771-2.425,3.2331-2.263,2.9091-1.778-0.1621-3.557-2.5861-0.323-4.52510.646-2.426
        1-1.615-5.6571-2.103-1.7781-0.162-2.58712.265-0.9712.102-3.07210.485-0.9711-1.617-1.777L611.133,725.065z"/>
<path id="AK" fill="#D3D3D3" d="M446.076,615.6751-0.322.85.35611.616,0.97113.071,0.16111.455-1.132h2.58610.162,2.9116.952,6.789
        10.484,2.58713.396-1.93910.646-0.16210.324-3.0711.455-1.61711.132-0.16111.94-1.45513.07,2.10310.647,2.90911.939,1.131
        11.132,2.42713.88,1.77613.395,5.98212.748,3.8812.263,2.74811.455,3.71815.012,1.77815.174,2.10210.97,4.36510.484,3.071
        1-0.97,3.3951-1.777,2.2641-1.617-0.8091-1.455-3.0711-2.748-1.4551-1.778-1.1321-0.809,0.80911.455,2.74810.161,3.721-1.131,0.483
        1-1.939-1.9381-2.103-1.29410.485,1.61611.293,1.7781-0.808,0.809c0,0-0.809-0.324-1.294-0.971
        c-0.484-0.646-2.102-3.396-2.102-3.3961-0.97-2.264c0,0-0.323,1.294-0.971,0.971c-0.646-0.324-1.293-1.455-1.293-1.45511.778-1.939
        l-1.455-1.455v-5.011h-0.808l-0.81,3.3951-1.132,0.4851-0.97-3.721-0.646-3.7181-0.809-0.48510.323-5.658v1.1321-1.455-1.293
```

==========PSHAPES==========



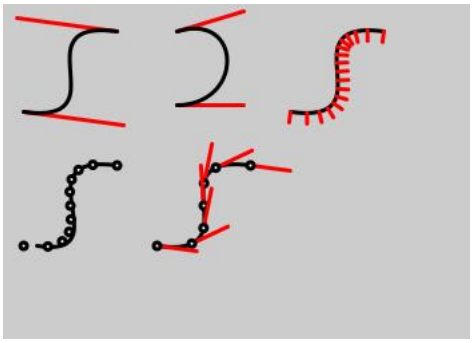
```
PShape square ,tri ,tristrip,  alien, head, body, quadstrip, quad;
PShape arc, box, line,sphere,lines ,quads,points, tris, svg;

void                    setup()
{ size(                 400, 400, P3D);
  square              = createShape(RECT,   0, 0, 50, 50);                    // x y w h
  quad                = createShape(QUAD,   38, 31, 86, 20, 69, 63, 30, 76 ); // quad(x1, y1, x2, y2, x3, y3, x4, y4)E
  arc                 = createShape(ARC,    10, 15, 60, 60, 0, PI );          // cx cy rx ry startangle stopAngle
  box                 = createShape(BOX,    40, 20, 50 );                     // box(w, h, d)
  line                = createShape(LINE,   0,0,30,50 );                      // x1 y1 x2 y2
  sphere              = createShape(SPHERE, 30 );                             // r
  svg                 = loadShape("bot.svg");                                 // in data folder

//=========================================
  tri                 = createShape();                //##########
  tri.beginShape();
  tri.fill(             0, 0, 255);
  tri.noStroke();
  tri.vertex(           0, 0);
  tri.vertex(           0, 50);
  tri.vertex(           50, 0);
  tri.endShape();

//=========================================
  tristrip            = createShape();
```

```
  tristrip.beginShape(TRIANGLE_STRIP);
  tristrip.vertex(        30, 75);
  tristrip.vertex(        40, 20);
  tristrip.vertex(        50, 75);
  tristrip.vertex(        60, 20);
  tristrip.vertex(        70, 75);
  tristrip.vertex(        80, 20);
  tristrip.vertex(        90, 75);
  tristrip.endShape();
//=========================================
  lines                 = createShape();
  lines.beginShape(LINES);
  lines.vertex(           30, 75);
  lines.vertex(           40, 20);
  lines.vertex(           50, 75);
  lines.vertex(           60, 20);
  lines.vertex(           70, 75);
  lines.vertex(           80, 20);
  lines.vertex(           90, 75);
  lines.endShape();
//=========================================
  points                = createShape();
  points.beginShape(POINTS);
  points.vertex(          30, 75);
  points.vertex(          40, 20);
  points.vertex(          50, 75);
  points.vertex(          60, 20);
  points.vertex(          70, 75);
  points.vertex(          80, 20);
  points.vertex(          90, 75);
  points.endShape();
//=========================================
  tris                  = createShape();
  tris.beginShape(TRIANGLES);
  tris.vertex(            30, 75);
  tris.vertex(            40, 20);
  tris.vertex(            50, 75);
  tris.vertex(            60, 20);
  tris.vertex(            70, 75);
  tris.vertex(            80, 20);
  tris.vertex(            90, 75);
  tris.endShape();
//=========================================
  quads                 = createShape();
  quads.beginShape(QUADS);
  quads.vertex(           30, 75);
  quads.vertex(           40, 20);
  quads.vertex(           50, 75);
  quads.vertex(           60, 20);
  quads.vertex(           70, 75);
  quads.vertex(           80, 20);
  quads.vertex(           90, 75);
  quads.endShape();
//=========================================
  quadstrip             = createShape();
  quadstrip.beginShape(QUAD_STRIP);
  quadstrip.vertex(       30, 75);
  quadstrip.vertex(       40, 20);
  quadstrip.vertex(       50, 75);
  quadstrip.vertex(       60, 20);
  quadstrip.vertex(       70, 75);
  quadstrip.vertex(       80, 20);
  quadstrip.vertex(       90, 75);
  quadstrip.endShape();
//=========================================
  alien                 = createShape(GROUP);                   //#############
  head                  = createShape(ELLIPSE, 0, 0, 50, 50);
  head.noStroke();
  body                  = createShape(RECT,    0, 45, 50, 40);
  body.fill(0);
  alien.addChild(         head);
  alien.addChild(         body);

}

void                    draw()
{ shape(                  square,    30,  30);
  shape(                  quad,      85,  200);
  shape(                  arc,       200, 5);
  shape(                  box,       60,  250);
  shape(                  line,      300, 15);
  shape(                  sphere,    150, 350);
  shapeMode(              CORNER);                          //CENTER
  shape(                  svg,       270, 280 , 100, 100); // x,y, h,w
  shape(                  tri,       125, 25);
  shape(                  tristrip,  50,  100);
  shape(                  lines,     120, 100);
  shape(                  quadstrip, 0,   100);
  shape(                  points,    180, 100);
  shape(                  quads,     230, 100);
  shape(                  tris,      280, 100);
  shape(                  alien,     25,  300 );           // Draw the group

}

void                     mouseClicked()
{ save(                  "pict.jpg");                                  // Mouse has been clicked
}
```

==========Bezier_Equation==========

```
  PVector         P1, C1 , C2, P2 , Bt;
  P1            = new PVector(50, 75);
  P2            = new PVector(100, 75);
  C1            = new PVector(25, 25);
  C2            = new PVector(125, 25);
  Bt            = new PVector(50, 85);
  size(150, 150);
  background(255);
  smooth();
  colorMode(RGB, 1);
  ellipse(P1.x, P1.y, 5, 5); // endpoints of curve
  ellipse(P2.x, P2.y, 5, 5);
  fill(255, 0, 0);
  ellipse(C1.x, C1.y, 5, 5);  // control points
  ellipse(C2.x, C2.y, 5, 5);
  noFill();
  stroke(0);
  strokeWeight(2);
  bezier(P1.x, P1.y, C1.x, C1.y, C2.x, C2.y ,P2.x, P2.y);

  stroke(1, 0, 1);
  strokeWeight(2);
  fill(255, 0, 0);
  float   t , nott;
  for                    (int i = 0; i <= 100; i++)
{ t = float(i)/100;
  nott = 1-t;
  Bt.x = nott*nott*nott*P1.x +3*nott*nott*t*C1.x  +3*nott*t*t*C2.x + t*t*t*P2.x;
  Bt.y = nott*nott*nott*P1.y +3*nott*nott*t*C1.y  +3*nott*t*t*C2.y + t*t*t*P2.y;
  point(Bt.x,Bt.y+3);
}

 save(                          "pictr2.jpg");                                    // Mouse has been clicked
```

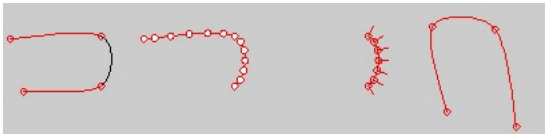
==========CURVES==========



```
size(                   350, 250);
noFill();
strokeWeight(           3);
//=======================================
stroke(                 255, 0, 0);
line(                   85, 20, 10, 10);
line(                   90, 90, 15, 80);
stroke(                 0, 0, 0);
bezier(                 85, 20, 10, 10, 90, 90, 15, 80);    //  bezier(ax1, ay1, az1, cx2, cy2, cz2, cx3, cy3, cz3, ax4, ay4, az4)
//=======================================
stroke(                 255, 0, 0);
line(                   130, 20, 180, 5);
line(                   180, 75, 130, 75);
stroke(                 0, 0, 0);
bezier(                 130, 20,  180, 5,  180, 75,  130, 75);
//=======================================
bezier(                 85, 120, 10, 110, 90, 190, 25, 180);
fill(                   255);
int steps             = 10;
for                     (int i = 0; i <= steps; i++)
{ float t             = i / float(steps);
  float x             = bezierPoint(85,  10,  90,  15,  t);
  float y             = bezierPoint(120, 110, 190, 180, t);
  ellipse(              x, y, 5, 5);
}
//=======================================
noFill();
bezier(                 185, 120, 110, 110, 190, 190, 115, 180);
steps                 = 6;
```

```
fill(                    255);
for                      (int i = 0; i <= steps; i++)
{ float t              = i / float(steps);
  float x              = bezierPoint(  185, 110, 190, 115, t);   // Get the location of the point
  float y              = bezierPoint(  120, 110, 190, 180, t);
  float tx             = bezierTangent(185, 110, 190, 115, t);   // Get the tangent points
  float ty             = bezierTangent(120, 110, 190, 180, t);
  float a              = atan2(ty, tx);                          // Calculate an angle from the tangent points
  a                   += PI;
  stroke(              255, 0, 0);
  line(                x, y, cos(a)*30 + x, sin(a)*30 + y);    // This code make inverse of line(x, y, cos(a)*-30 + x, sin(a)*-30 +
y);
  stroke(              0);
  ellipse(             x, y, 5, 5);
}
//=====================================
noFill();
bezier(                  285, 20, 210, 10, 290, 90, 215, 80);
stroke(                  255, 0, 0);

steps                  = 16;
for                      (int i = 0; i <= steps; i++)
{ float t              = i / float(steps);
  float x              = bezierPoint(  285, 210, 290, 215, t);
  float y              = bezierPoint(  20,  10,  90,  80,  t);
  float tx             = bezierTangent(285, 210, 290, 215, t);
  float ty             = bezierTangent(20,  10,  90,  80,  t);
  float a              = atan2(ty, tx);
  a                   -= HALF_PI;
  line(                x, y, cos(a)*8 + x, sin(a)*8 + y);
}
save(                   "pict.jpg");
```

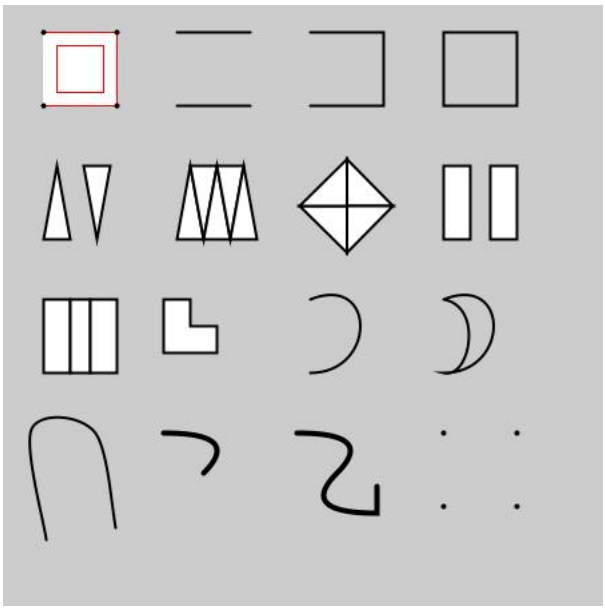
==========CURVE==========



```
size(                    450, 100);
noFill();
stroke(                  255, 0, 0);
curve(                   5, 26, 5, 26, 73, 24, 73, 61);   //curve(cx1, cy1, ax2, ay2, ax3, ay3, cx4, cy4)
stroke(                  0);
curve(                   5, 26, 73, 24, 73, 61, 15, 65);
stroke(                  255, 0, 0);
curve(                   73, 24, 73, 61, 15, 65, 15, 65);

ellipse(                5, 26, 5, 5);
ellipse(                5, 26, 5, 5);
ellipse(                73, 24, 5, 5);
ellipse(                15, 65, 5, 5);

ellipse(                5, 26, 5, 5);
ellipse(                73, 24, 5, 5);
ellipse(                73, 61, 5, 5);
ellipse(                73, 61, 5, 5);


//=====================================
noFill();
curve(                   105, 26, 105, 26, 173, 24, 173, 61);
curve(                   105, 26, 173, 24, 173, 61, 115, 65);
fill(                    255);
ellipseMode(             CENTER);
int steps              = 6;
for                      (int i = 0; i <= steps; i++)
{ float t              = i / float(steps);
  float x              = curvePoint(105, 105, 173, 173, t);
  float y              = curvePoint(26, 26, 24, 61, t);
  ellipse(             x, y, 5, 5);
  x                    = curvePoint( 105, 173, 173, 115, t);
  y                    = curvePoint( 26, 24, 61, 65, t);
  ellipse(             x, y, 5, 5);
}
//=====================================
noFill();
curve(                   205, 26, 273, 24, 273, 61, 215, 65);
steps                  = 6;
for                      (int i = 0; i <= steps; i++)
{ float t              = i / float(steps);
  float x              = curvePoint(205, 273, 273, 215, t);
  float y              = curvePoint(26, 24, 61, 65, t);
  ellipse(             x, y, 5, 5);
  float tx             = curveTangent(205, 273, 273, 215, t);
  float ty             = curveTangent(26, 24, 61, 65, t);
  float a              = atan2(ty, tx);
  a                   -= PI/2.0;
  line(                x, y, cos(a)*8 + x, sin(a)*8 + y);
}
//=====================================
noFill();
beginShape();
curveVertex(             384,  91);
curveVertex(             384,  91);
curveVertex(             368,  19);
```

```
curveVertex(           321,  17);
curveVertex(           332,  80);
curveVertex(           332, 80);
endShape();
save(                  "pict.jpg");

ellipse(              384,  91, 5, 5);
ellipse(              384,  91, 5, 5);
ellipse(              368,  19, 5, 5);
ellipse(              321,  17, 5, 5);
ellipse(              332,  80, 5, 5);
```

==========**BEGINSHAPE**==========



```
size(                 450, 450);
strokeWeight(          1);
//=======================================
stroke(255, 0, 0);
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
beginContour();                          //####################
vertex(40, 30);
vertex(75, 30);
vertex(75, 65);
vertex(40, 65);
endContour();
endShape(CLOSE);
//=======================================
stroke(0, 0, 0);
strokeWeight(          4);
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
//=======================================
stroke(0, 0, 0);
strokeWeight(          2);
beginShape(LINES);
vertex(130, 20);
vertex(185, 20);
vertex(185, 75);
vertex(130, 75);
endShape();
//=======================================
noFill();
beginShape();
vertex(230, 20);
vertex(285, 20);
vertex(285, 75);
vertex(230, 75);
endShape();
//=======================================
noFill();
beginShape();
vertex(330, 20);
vertex(385, 20);
vertex(385, 75);
vertex(330, 75);
endShape(CLOSE);
//=======================================
```

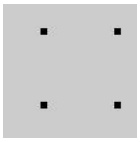
```
fill(255);
beginShape(TRIANGLES);
vertex(30, 175);
vertex(40, 120);
vertex(50, 175);
vertex(60, 120);
vertex(70, 175);
vertex(80, 120);
endShape();
//=====================================
beginShape(TRIANGLE_STRIP);
vertex(130, 175);
vertex(140, 120);
vertex(150, 175);
vertex(160, 120);
vertex(170, 175);
vertex(180, 120);
vertex(190, 175);
endShape();
//=====================================
beginShape(TRIANGLE_FAN);
vertex(257.5, 150);
vertex(257.5, 115);
vertex(292, 150);
vertex(257.5, 185);
vertex(222, 150);
vertex(257.5, 115);
endShape();
//=====================================
beginShape(QUADS);
vertex(330, 120);
vertex(330, 175);
vertex(350, 175);
vertex(350, 120);
vertex(365, 120);
vertex(365, 175);
vertex(385, 175);
vertex(385, 120);
endShape();
//=====================================
beginShape(QUAD_STRIP);
vertex(30, 220);
vertex(30, 275);
vertex(50, 220);
vertex(50, 275);
vertex(65, 220);
vertex(65, 275);
vertex(85, 220);
vertex(85, 275);
endShape();
//=====================================
beginShape();
vertex(120, 220);
vertex(140, 220);
vertex(140, 240);
vertex(160, 240);
vertex(160, 260);
vertex(120, 260);
endShape(CLOSE);
//=====================================
noFill();
beginShape();
vertex(230, 220);
bezierVertex(280, 200, 280, 275, 230, 275);
endShape();
//=====================================
beginShape();
vertex(330, 220);
bezierVertex(380, 200, 380, 275, 330, 275);
bezierVertex(350, 280, 360, 225, 330, 220);
endShape();
//=====================================
noFill();
beginShape();
curveVertex(84,  391);
curveVertex(84,  391);
curveVertex(68,  319);
curveVertex(21,  317);
curveVertex(32, 400);
curveVertex(32, 400);
endShape();
//=====================================
noFill();
strokeWeight(4);
beginShape();
vertex(120, 320);
quadraticVertex(180, 320, 150, 350);
endShape();
//=====================================
noFill();
strokeWeight(4);
beginShape();
vertex(220, 320);
quadraticVertex(280, 320, 250, 350);
quadraticVertex(220, 380, 280, 380);
vertex(280, 360);
endShape();
//=====================================
beginShape(POINTS);
vertex(330, 320);
vertex(385, 320);
vertex(385, 375);
```

```
vertex(330, 375);
endShape();

save(                  "pict.jpg");
```

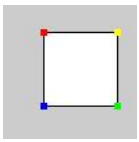
================vertex-types==============================

vertex()

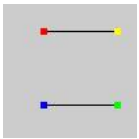
Examples
example pic



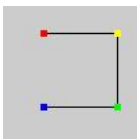
```
void setup()
{ strokeWeight(5);
}
void draw()
{ beginShape(POINTS);
  vertex(30, 20);
  vertex(85, 20);
  vertex(85, 75);
  vertex(30, 75);
  endShape();
}
void mousePressed() { save("pict.jpg"); }
```



```
  stroke(0);
  strokeWeight(1);
  beginShape();
  vertex(30, 20);
  vertex(85, 20);
  vertex(85, 75);
  vertex(30, 75);
  endShape(CLOSE);
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(30, 20);
  stroke(255,255,0); vertex(85, 20);
  stroke(0,255,0);   vertex(85, 75);
  stroke(0,0,255);   vertex(30, 75);
  endShape();
  save("pict.jpg");
```

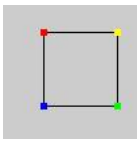


```
  stroke(0);
  strokeWeight(1);
  beginShape(LINES);
  vertex(30, 20);
  vertex(85, 20);
  vertex(85, 75);
  vertex(30, 75);
  endShape();
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(30, 20);
  stroke(255,255,0); vertex(85, 20);
  stroke(0,255,0);   vertex(85, 75);
  stroke(0,0,255);   vertex(30, 75);
  endShape();
  save("pict.jpg");
```



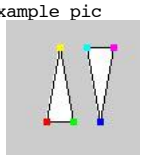
```
  stroke(0);
  strokeWeight(1);
  noFill();
  beginShape();
  vertex(30, 20);
  vertex(85, 20);
```

```
  vertex(85, 75);
  vertex(30, 75);
  endShape();
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(30, 20);
  stroke(255,255,0); vertex(85, 20);
  stroke(0,255,0);   vertex(85, 75);
  stroke(0,0,255);   vertex(30, 75);
  endShape();
  save("pict.jpg");
```

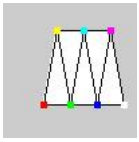


```
  stroke(0);
  strokeWeight(1);
  noFill();
  beginShape();
  vertex(30, 20);
  vertex(85, 20);
  vertex(85, 75);
  vertex(30, 75);
  endShape(CLOSE);
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(30, 20);
  stroke(255,255,0); vertex(85, 20);
  stroke(0,255,0);   vertex(85, 75);
  stroke(0,0,255);   vertex(30, 75);
  endShape();
  save("pict.jpg");
```

example pic

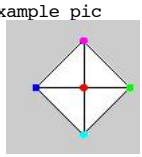


```
  stroke(0);
  strokeWeight(1);
  beginShape(TRIANGLES);
  vertex(30, 75);
  vertex(40, 20);
  vertex(50, 75);
  vertex(60, 20);
  vertex(70, 75);
  vertex(80, 20);
  endShape();
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(30, 75);
  stroke(255,255,0); vertex(40, 20);
  stroke(0,255,0);   vertex(50, 75);
  stroke(0,255,255); vertex(60, 20);
  stroke(0,0,255);   vertex(70, 75);
  stroke(255,0,255); vertex(80, 20);
  endShape();
  save("pict.jpg");
```

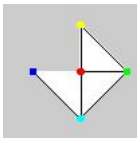


```
  stroke(0);
  strokeWeight(1);
  beginShape(TRIANGLE_STRIP);
  vertex(30, 75);
  vertex(40, 20);
  vertex(50, 75);
  vertex(60, 20);
  vertex(70, 75);
  vertex(80, 20);
  vertex(90, 75);
  endShape();
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);     vertex(30, 75);
  stroke(255,255,0);   vertex(40, 20);
  stroke(0,255,0);     vertex(50, 75);
  stroke(0,255,255);   vertex(60, 20);
  stroke(0,0,255);     vertex(70, 75);
  stroke(255,0,255);   vertex(80, 20);
  stroke(255,255,255); vertex(90, 75);
  endShape();
  save("pict.jpg");
```

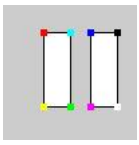
example pic



```
stroke(0);
strokeWeight(1);
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92,   50);
vertex(57.5, 85);
vertex(22,   50);
vertex(57.5, 15);
endShape();
stroke(255,0,0);
strokeWeight(5);
beginShape(POINTS);
stroke(255,0,0);   vertex(57.5, 50);
stroke(255,255,0); vertex(57.5, 15);
stroke(0,255,0);   vertex(92, 50);
stroke(0,255,255); vertex(57.5, 85);
stroke(0,0,255);   vertex(22, 50);
stroke(255,0,255); vertex(57.5, 15);
endShape();
save("pict.jpg");
```

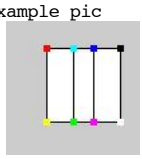


```
stroke(0);
strokeWeight(1);
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92,   50);
vertex(57.5, 85);
vertex(22,   50);
endShape();
stroke(255,0,0);
strokeWeight(5);
beginShape(POINTS);
stroke(255,0,0);   vertex(57.5, 50);
stroke(255,255,0); vertex(57.5, 15);
stroke(0,255,0);   vertex(92, 50);
stroke(0,255,255); vertex(57.5, 85);
stroke(0,0,255);   vertex(22, 50);
endShape();
save("pict.jpg");
```

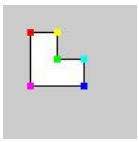


```
stroke(0);
strokeWeight(1);
beginShape(QUADS);
vertex(30, 20);
vertex(30, 75);
vertex(50, 75);
vertex(50, 20);
vertex(65, 20);
vertex(65, 75);
vertex(85, 75);
vertex(85, 20);
endShape();
stroke(255,0,0);
strokeWeight(5);
beginShape(POINTS);
stroke(255,0,0);     vertex(30, 20);
stroke(255,255,0);   vertex(30, 75);
stroke(0,255,0);     vertex(50, 75);
stroke(0,255,255);   vertex(50, 20);
stroke(0,0,255);     vertex(65, 20);
stroke(255,0,255);   vertex(65, 75);
stroke(255,255,255); vertex(85, 75);
stroke(0,0,0);       vertex(85, 20);
endShape();
save("pict.jpg");
```

example pic

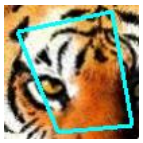


```
  stroke(0);
  strokeWeight(1);
  beginShape(QUAD_STRIP);
  vertex(30, 20);
  vertex(30, 75);
  vertex(50, 20);
  vertex(50, 75);
  vertex(65, 20);
  vertex(65, 75);
  vertex(85, 20);
  vertex(85, 75);
  endShape();
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);     vertex(30, 20);
  stroke(255,255,0);   vertex(30, 75);
  stroke(0,255,0);     vertex(50, 75);
  stroke(0,255,255);   vertex(50, 20);
  stroke(0,0,255);     vertex(65, 20);
  stroke(255,0,255);   vertex(65, 75);
  stroke(255,255,255); vertex(85, 75);
  stroke(0,0,0);       vertex(85, 20);
  endShape();
  save("pict.jpg");
```



```
  stroke(0);
  strokeWeight(1);
  beginShape();
  vertex(20, 20);
  vertex(40, 20);
  vertex(40, 40);
  vertex(60, 40);
  vertex(60, 60);
  vertex(20, 60);
  endShape(CLOSE);
  stroke(255,0,0);
  strokeWeight(5);
  beginShape(POINTS);
  stroke(255,0,0);   vertex(20, 20);
  stroke(255,255,0); vertex(40, 20);
  stroke(0,255,0);   vertex(40, 40);
  stroke(0,255,255); vertex(60, 40);
  stroke(0,0,255);   vertex(60, 60);
  stroke(255,0,255); vertex(20, 60);
  endShape();
  save("pict.jpg");
```

==========Vertex==========



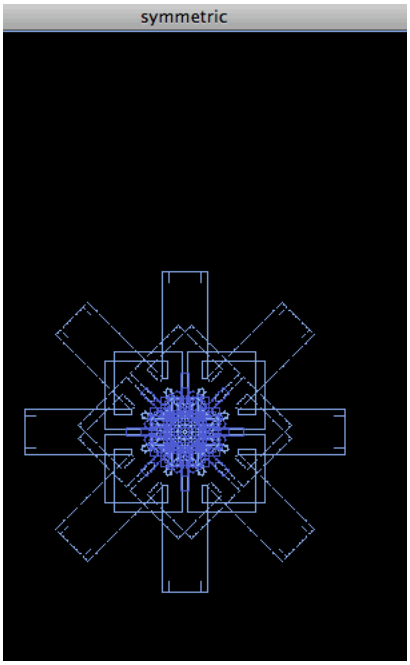
```
PImage img = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");
println (img.width );
  size(100, 100, P3D);
  image(img, 0, 0);                    // x y
  strokeWeight(3);
  noFill();
  stroke(0,255,255);                  // grey

  beginShape(QUADS);
  vertex(10, 20);
  vertex(80, 5);
  vertex(95, 90);
  vertex(40, 95);
  endShape();
  save("pict.jpg");
```

==========Pvector==========

```
PVector                    v1, v2 , v;
float[] vvv              = { 20.0, 30.0, 40.0 };
v1                       = new PVector(20.0, 30.0, 40.0); println(v1);          // [ 20.0, 30.0, 40.0 ]
v2                       = new PVector(0.0, 0.0, 0.0);    println(v2);          // [ 0.0, 0.0, 0.0 ]
v2.set(                    v1);                           println(v2);          // [ 20.0, 30.0, 40.0 ]
v                        = new PVector(0.0, 0.0, 0.0);
v.set(                     vvv);                          println(v);           // [ 20.0, 30.0, 40.0 ]
println (                  "v.x = "+v.x);                                       // v.x = 20.0
v                        = PVector.random2D();            println(v);           // [ -0.24328443, 0.96995497, 0.0 ]
v                        = PVector.random3D();            println(v);           // [ 0.32202893, 0.17134926, 0.9310944 ]
v                        = PVector.fromAngle(0.01);       println(v);           // Prints "[ 0.99995, 0.009999833, 0.0 ]"
v2                       = v1.get();                      println(v2);          // [ 20.0, 30.0, 40.0 ]
```

```
float m                 = v1.mag();                  println(m);            // Prints "53.851646"
m                       = v1.magSq();                println(m);            // Prints "2900.0"
v.add(                    v1);                       println(v);            // [ 20.99995, 30.01, 40.0 ]
v.add(                    25, 50, 0);                println(v);            // [ 45.99995, 80.01, 40.0 ]
v.sub(                    v1);                       println(v);            // [ 25.99995, 50.010002, 0.0 ]
v.mult(                   2);                        println(v);            // [ 51.9999, 100.020004, 0.0 ]
v.div(                    2);                        println(v);            // [ 25.99995, 50.010002, 0.0 ]
float d                 = PVector.dist(v1, v);       println(d);            // Prints "45.126484"
PVector v3              = v.cross(v2);               println(v3);           // [ 2000.4001, -1039.998, -220.20154 ]
v.limit(                  5);                        println(v);            // Prints "[ 2.3063972, 4.4362745, 0.0 ]"
v.normalize();            println(v);                                       // [ 0.46127942, 0.88725495, 0.0 ]
v.setMag(10);             println(v);                                       // Prints " 4.6127944, 8.872549, 0.0 ]
PVector v4              = new PVector(10.0, 20.0);
println(                  v4.heading());                                    // Prints "1.1071488" 1.1071488
v4.rotate(                HALF_PI);                  println(v4);           // Prints "[ -20.0, 9.999999, 0.0 ]
float ang               = PVector.angleBetween(v1, v); println(degrees(ang)); // Prints 48.272224"
float[] f               = v.array();                 println( f);           //[0] 4.6127944 [1] 8.872549 [2] 0.0
```

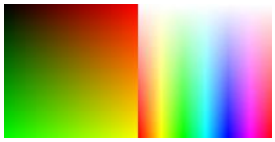
==========Pvector2ImageArray==========




```
int       sides       = 8;                       // symmeyry
int       pnum        = 2;                       // particles to draw
float     p[]         = new float[pnum];         // 0 < p < 1; chance of turning
PVector[] pos         = new PVector[pnum];       // position
PVector[] vel         = new PVector[pnum];       // velocity
color[]   col         = new color[pnum];         // color
boolean[] ref         = new boolean[pnum];       // randonm binary
int       lastf       = 0;                       // last frame

void                    setup()
{ size(                 600,600,P2D);
  frameRate(            30);
  restart();
}
void                    draw()
{ fill(                 0,0.00001);
  rectMode(             CORNERS);
  rect(                 0,0,width,height);
  translate(            width/2,height/2);
  stroke(               255);
  for                   (int i=0;i<pnum;i++)
{ pos[i].add(           vel[i]);
  if                    (random(1) < p[i] || (pos[i].mag() > (width+height)/4 && pos[i].dot(vel[i])>=0))
{ float x             = vel[i].x;                          // swap x and y velocity
  vel[i].x            = vel[i].y;
  vel[i].y            = -x;
}                                                          // if(random(1) < p[i]
  for                   (int j=0;j<sides;j++)
{ rotate(               TWO_PI/sides);
  stroke(               col[i]);
  point(                pos[i].x, pos[i].y);
  if                    (ref[i]) point(-pos[i].x, pos[i].y);
}                                                          // for(int j=0;j<sides;j++)
}                                                          // for (int i=0;i<pnum;i++)
 if                     ( frameCount-lastf >3  )
{ saveFrame(            "t/line-####.png");
  lastf                = frameCount;
}
```

```
}                                                  // void draw()
void                      restart()
{ background(             0);
  for                     (int i=0;i<pnum;i++)
{ p[i]                  = random(0.2);
  pos[i]                = new PVector();
  vel[i]                = new PVector(0,1);                      // x = 0, y = 1
  col[i]                = color(random(255), random(255), random(255));
  ref[i]                = (random(2)<1);                         // random 1 and 0
}
}
void                      mousePressed()  { restart(); }
void                      keyPressed()    { save("pict.jpg"); }
```

==========colorMode==========

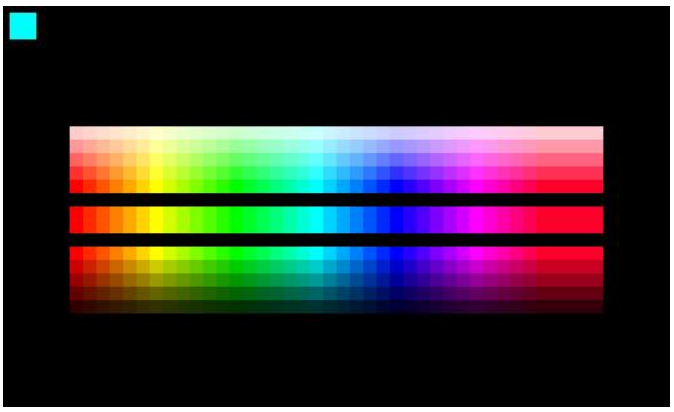


```
  size(                   200, 100);   // need 1.5  ##########
  noStroke();
  colorMode(              RGB, 100);
  for                     (int i = 0; i < 100; i++)
{ for                     (int j = 0; j < 100; j++)
{ stroke(                 i, j, 0);
  point(                  i, j);
}
}
  noStroke();
  colorMode(              HSB, 100);
  for                     (int i = 100; i < 200; i++)
{ for                     (int j = 0; j < 100; j++)
{ stroke(                 i-100, j, 100);
  point(                  i, j);
}
}
  save(                   "pict.jpg");
```

==========ARGB==========

```
PImage img          = createImage(66, 66, ARGB);
img.loadPixels();
for                  (int i = 0; i < img.pixels.length; i++)
{ img.pixels[i]     = color(0, 90, 102, i % img.width * 2);
}
img.updatePixels();
image(                img, 17, 17);
image(                img, 34, 34);
```

==========HSB==========

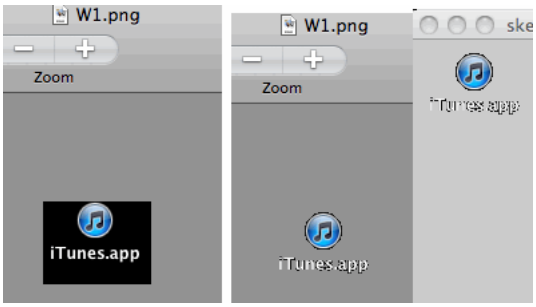


```
float hue1, sat,brite;
color c;
void                  setup()
{ size(               500, 300);
  colorMode(          HSB, 1, 1, 1);
  noStroke(           );
  background(         0);
}
```

```
void                draw()
{ fill(             .5, 1, 1);
  rect(             5,5, 20, 20);
  for               ( int i = 0; i < 36; i++ )
{ hue1              = float(i)/36;
  fill(             hue1, 1, 1);
  rect(             10*i+50,150, 50, 20);
  c                 = get(10*i+55,155);
  println(          i+" "+ red(c)+" "+green(c)+" "+blue(c) +" "+hue(c) +" "+saturation(c) +" "+brightness(c) );
}
  c                 = get(10,10);
  println(          red(c)+" "+green(c)+" "+blue(c) +" "+hue(c) +" "+saturation(c) +" "+brightness(c) );

  for               ( int j = 0;j < 5; j++ )
{ for               ( int i = 0; i < 36; i++ )
{ hue1              = float(i)/36;
  sat               = 1-float(j)/5;
  fill(             hue1, sat, 1);
  rect(             10*i+50,130-10*j, 50,10);
}
}
  for               ( int j = 0;j < 5; j++ )
{ for               ( int i = 0; i < 36; i++ )
{ hue1              = float(i)/36;
  brite             = 1-float(j)/5;
  fill(             hue1, 1, brite);
  rect(             10*i+50,180+10*j, 50,10);
}
}
}

void                mouseClicked()
{ save(             "pict.jpg");              // Mouse has been clicked
}
```

```
=====================================================
   0.0         1.0         1.0         0.5         1.0 1.0
0  1.0         0.0         0.0         0.0         1.0 1.0
1  1.0         0.16470589  0.0         0.02745097  1.0 1.0
2  1.0         0.33333334  0.0         0.05555555  1.0 1.0
3  1.0         0.49803922  0.0         0.08300653  1.0 1.0
4  1.0         0.6666667   0.0         0.11111110  1.0 1.0
5  1.0         0.83137256  0.0         0.1385621   1.0 1.0
6  1.0         1.0         0.0         0.16666667  1.0 1.0
7  0.83137256  1.0         0.0         0.19477125  1.0 1.0
8  0.6627451   1.0         0.0         0.22287583  1.0 1.0
9  0.49803922  1.0         0.0         0.25032678  1.0 1.0
10 0.32941177  1.0         0.0         0.2784314   1.0 1.0
11 0.16470589  1.0         0.0         0.30588236  1.0 1.0
12 0.0         1.0         0.0         0.33333334  1.0 1.0
13 0.0         1.0         0.16470589  0.3607843   1.0 1.0
14 0.0         1.0         0.33333334  0.38888887  1.0 1.0
15 0.0         1.0         0.49803922  0.41633987  1.0 1.0
16 0.0         1.0         0.6666667   0.44444445  1.0 1.0
17 0.0         1.0         0.83137256  0.47189543  1.0 1.0
18 0.0         1.0         1.0         0.5         1.0 1.0
19 0.0         0.83137256  1.0         0.52810454  1.0 1.0
20 0.0         0.6627451   1.0         0.55620915  1.0 1.0
21 0.0         0.49803922  1.0         0.5836601   1.0 1.0
22 0.0         0.33333334  1.0         0.6111111   1.0 1.0
23 0.0         0.16470589  1.0         0.6392157   1.0 1.0
24 0.0         0.0         1.0         0.6666667   1.0 1.0
25 0.16470589  0.0         1.0         0.6941176   1.0 1.0
26 0.32941177  0.0         1.0         0.7215686   1.0 1.0
27 0.49803922  0.0         1.0         0.7496732   1.0 1.0
28 0.6666667   0.0         1.0         0.77777773  1.0 1.0
29 0.83137256  0.0         1.0         0.80522877  1.0 1.0
30 1.0         0.0         1.0         0.8333333   1.0 1.0
31 1.0         0.0         0.83137256  0.8614379   1.0 1.0
32 1.0         0.0         0.6627451   0.88954246  1.0 1.0
33 1.0         0.0         0.49803922  0.9169935   1.0 1.0
34 1.0         0.0         0.33333334  0.9444444   1.0 1.0
35 1.0         0.0         0.16470589  0.972549    1.0 1.0
```

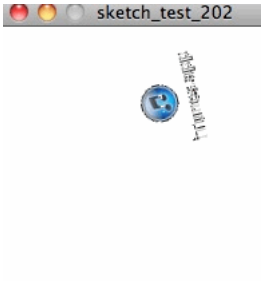
==========Convert_ColorPNG_To_TransparentPNG==========




```
String FILENAME = "/Users/don_sauer/Downloads/W1.png";
PImage img;
PImage newImg;
int x;
int y;
int i;
```

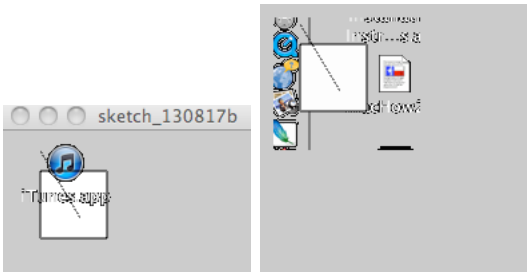
```
void                setup()
{ size(             200, 200);
  img             = loadImage( FILENAME );
  newImg          = createImage( img.width, img.height, ARGB );
  for               ( x = 0; x < img.width; x++ )
{ for               ( y = 0; y < img.height; y++ )
{ i               = ( ( y * img.width ) + x );
  if                ( img.pixels[i] < color( 9, 9, 9 ) ){ newImg.pixels[i] = color( 0, 0, 0, 0 );}
  else
{ newImg.pixels[i] = img.pixels[i];
}                   //else
}                   //for( y = 0; y < img.height; y++ )
}                   //for( x = 0; x < img.width; x++ )
  newImg.save(      FILENAME );
}
void                draw()
{ image(            newImg, 10, 10);
}
```

==========Sprite==========




```
PImage head; // A variable for the image file
float x,y;   // Variables for image location
float rot;   // A variable for image rotation

void setup()
{ size(200,200);
  head = loadImage("//Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/data/W1.png");// Load image, initialize
variables
  x = 0.0;
  y = width/2.0;
  rot = 0.0;
}

void draw()
{ background(255);
  translate(x,y);   // Translate and rotate
  rotate(rot);
  image(head,0,0); // Images can be animated just like regular shapes using variables, translate(), rotate(), and so on.
  x += 1.0;        // Adjust variables for animation
  rot += 0.02;
  if (x > width+head.width) {x = -head.width; }
}
```

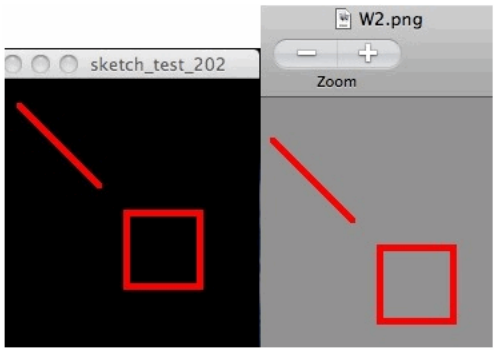
==========Images_together_with_Shapes==========




```
String FILENAME         = "/Users/don_sauer/Downloads/W1.png";
PImage img;
PImage newImg;
PShape line, square ;
int x;
int y;
int i;
void setup()
{ size(200, 200,P3D);
  line                  = createShape(LINE,   0, 0,30,50 );
  square                = createShape(RECT,   0, 0, 50, 50);
  img                   = loadImage( FILENAME );
  newImg                = createImage( img.width, img.height, ARGB );
  for                     ( x = 0; x < img.width; x++ )
{ for                     ( y = 0; y < img.height; y++ )
{ i                     = ( ( y * img.width ) + x );
  if                      ( img.pixels[i] == color( 0, 0, 0 ) )
{ newImg.pixels[i]      = color( 0, 0, 0, 0 );
} //                       if( img.pixels[i] == color( 0, 0, 0 ) )
```

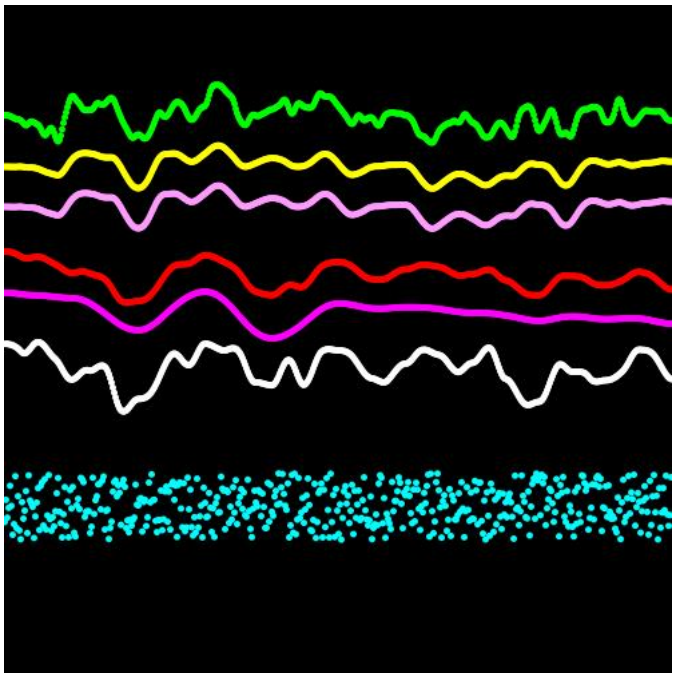
```
  else                    { newImg.pixels[i] = img.pixels[i]; }
} //                      for( y = 0; y < img.height; y++ )
} //                      for( x = 0; x < img.width; x++ )
  newImg.save(            FILENAME );     // x1 y1 x2 y2
} //                      setup()

void                      draw()
{ shape(                  line,      30, 15);
  shape(                  square,    30, 30);
  image(                  newImg,    10, 10);
}
void                      keyPressed()
{  if                     (key == ' ')
{  save(                  "pict.png");
}
}
```

================Pixels_to_Transparent_file==================



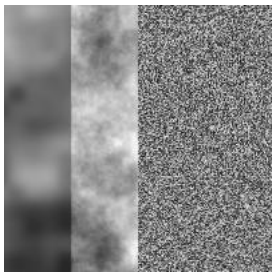

```
String FILENAME         = "/Users/don_sauer/Downloads/W2.png";
PImage img;
PImage newImg;
int x;
int y;
int i;

  size(200, 300);
  background(0);
  stroke(255, 0, 0);                    // HSB
  strokeWeight(5);
  line(20, 20, 80, 80);
  noFill();
  rect(100, 100, 55, 55);                   // X, Y, H, W

  loadPixels();                                         // load display
  newImg                = createImage( 200, 200, ARGB ) ; //
  for                     ( x = 0; x < 200; x++ )
{ for                     ( y = 0; y < 200; y++ )
{ i                     = ( ( y * 200 ) + x );
  if                      (  red(pixels[i]) > 5 )  { newImg.pixels[i] = color( 256, 0, 0, 256 ); }
  else                                             { newImg.pixels[i] = color( 0,   0, 0, 0 ); }
}
}
  newImg.save(            FILENAME );     // x1 y1 x2 y2
```

==================NOISE============================



```
float noiseScale        = 0.02;
float noiseScale2       = 0.05;
void                      setup()
{ size(                   500, 500);
  strokeWeight(           5);
  noLoop();                                       // draw() will not loop
}
void draw()
{ background(0);
  for                     (int x=0; x < width; x++)
{ noiseDetail(            3,0.5);
  float noiseVal1       = noise((x)*noiseScale, height*noiseScale/2);
  stroke(                 255, 0, 0);                          // RED = 0.02 scale 3 decades roll off 0.5
  point(                  x, noiseVal1*80+170);
  float noiseVal2       = noise((x)*noiseScale2, height*noiseScale2/2);
  stroke(                 0,255, 0);                         // GREEN  = 0.05 scale 3 decades roll off 0.5
  point(                  x, noiseVal2*80+50);
  noiseDetail(            6,0.1);
  float noiseVal3       = noise((x)*noiseScale, height*noiseScale/2);
  stroke(                 255, 0, 255);                        // purple = 0.02 scale 6 decades roll off 0.1
  point(                  x, noiseVal3*80+210);
  float noiseVal4       = noise((x)*noiseScale2, height*noiseScale2/2);
   stroke(                255,255, 0);                         // yellow  = 0.05 scale 6 decades roll off 0.1
  point(                  x, noiseVal4*80+100);
  noiseDetail(            3,0.9);
  float noiseVal5       = noise((x)*noiseScale, height*noiseScale/2);
  stroke(                 255, 255, 255);                   // White = 0.02 scale 3 decades roll off 0.9
  point(                  x, noiseVal5*80+220);
  float noiseVal6       = noise((x)*noiseScale2, height*noiseScale2/2);
  stroke(                 255,155, 255);                    // pink  = 0.052 scale 3 decades roll off 0.9
  point(                  x, noiseVal4*80+130);
  stroke(                 0,255, 255);                      // Cyan =  random
  point(                  x, random(50)+350);
}
}

void                       mouseClicked()
{ save(                   "pict.jpg");                                  // Mouse has been clicked
}
```

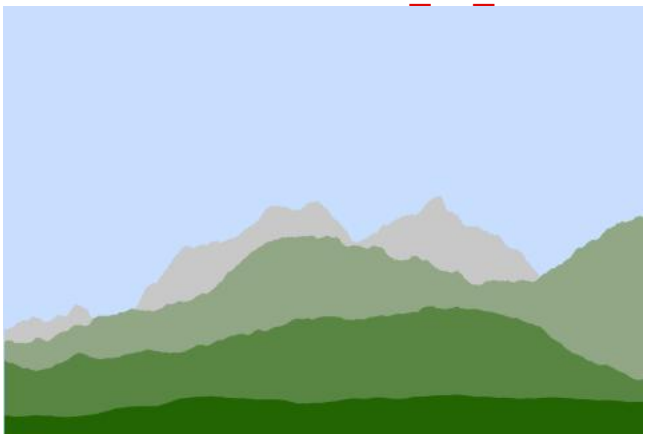
====================Perlin_vs_Random_2D==========================



```
float   noiseVal;
float   noiseScale        = 0.02;
int     pwidth            = 100;
void                        setup()
{ size(                     pwidth*2,200);
}
void                        draw()
{ for                       (int y = 0; y < height;  y++)
{ for                       (int x = 0; x < pwidth/2; x++)
```

```
{ noiseDetail(              3,0.5);                                                 // three decades, decay as 0.5
  noiseVal                = noise((mouseX+x)*noiseScale, (mouseY+y)*noiseScale); // noise(x, y) is 1/f
  stroke(                   noiseVal*255);
  point(                    x,y);                                                 // plot as points
  noiseDetail(              8,0.65);                                              // eight decades, decay as 0.65
  noiseVal                = noise((mouseX +x + pwidth/2)*noiseScale, (mouseY +y)*noiseScale);
  stroke(                   noiseVal * 255);
  point(                    x + pwidth/2, y);
}
}
  for                      (int i = 0;      i < height; i++)         // Draw points
{ for                      (int j = 0;      j < width; j++)
{ stroke(                  int(random(255)));
  point(                   j+ pwidth ,i);
}
}
}
void                        mouseClicked()
{ save(                    "pict.jpg");                                          // Mouse has been clicked
}
```

==========Perlin_To_Mountains==========



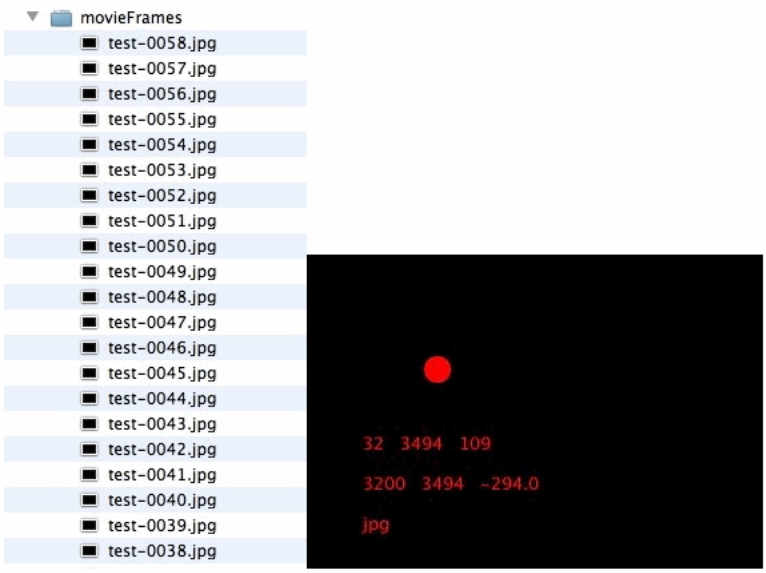
```
int                         i,w=480,h=320,m=w/2;
float                       p,g,x,y,z;
void                        setup()
{ size(                     w,h);
}
void                        draw()
{ background(               200,222,255);
  p                       += .005;
  for                       (x=0;x++<w;)
{ y                        = 0;
  for                       (i=5;i-->0;)
{ noiseDetail(              14-i*2);                        //noiseDetail(ordersMag, falloff) for noise
  g                        = 200-i*33;
  stroke(                   g-i*22, g, g-i*33);            // stroke color
  z                        = h-(noise(33*i+p+x/m)*(h-90*i));  // noise is perlin
  if                        (y>0&&z<y)
{ line(                     x,y,x,z);
}
  y                        = z;
}
}
}
void                        mouseClicked()
{ save(                     "pict.jpg");                                          // Mouse has been clicked
}
```

==========Perlinellipse==========

```
float time                = 0.0;
float increment           = 0.01;
void                        setup()
{ size(                     200,200);
  smooth(                   );
}
void                        draw()
{ background(               255);
  float n                 = noise(time)*width; // Get a noise value at "time" and scale it according to the window's width.
```

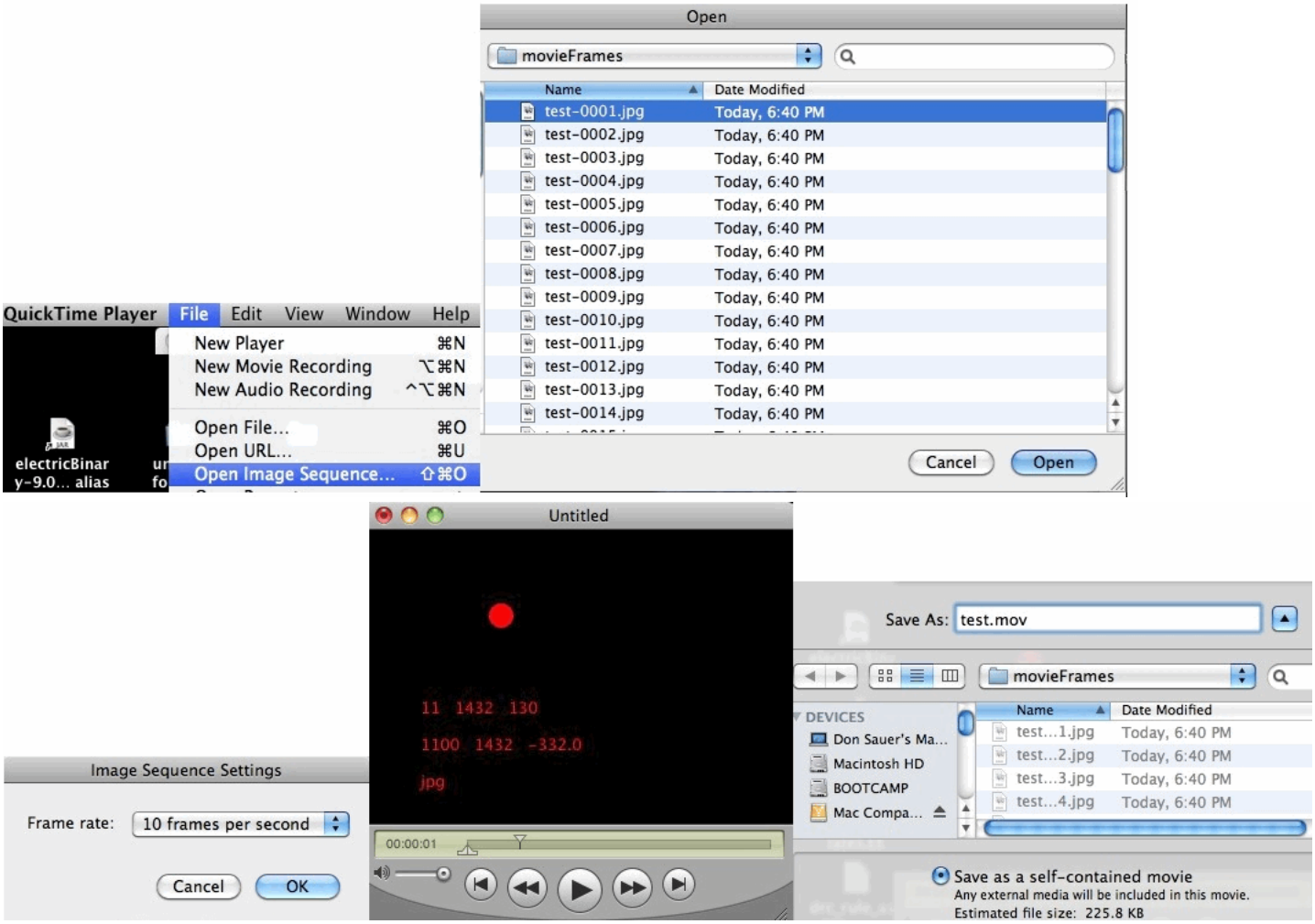
```
  time                  += increment;          // With each cycle, increment the " time "
  fill(                    0);
  ellipse(                 width/2,height/2,n,n);// Draw the with size determined by Perlin noise
}
void                        mouseClicked()
{ save(                    "pict.jpg");                                    // Mouse has been clicked
}
```

==========Create_Image_Sequence==========

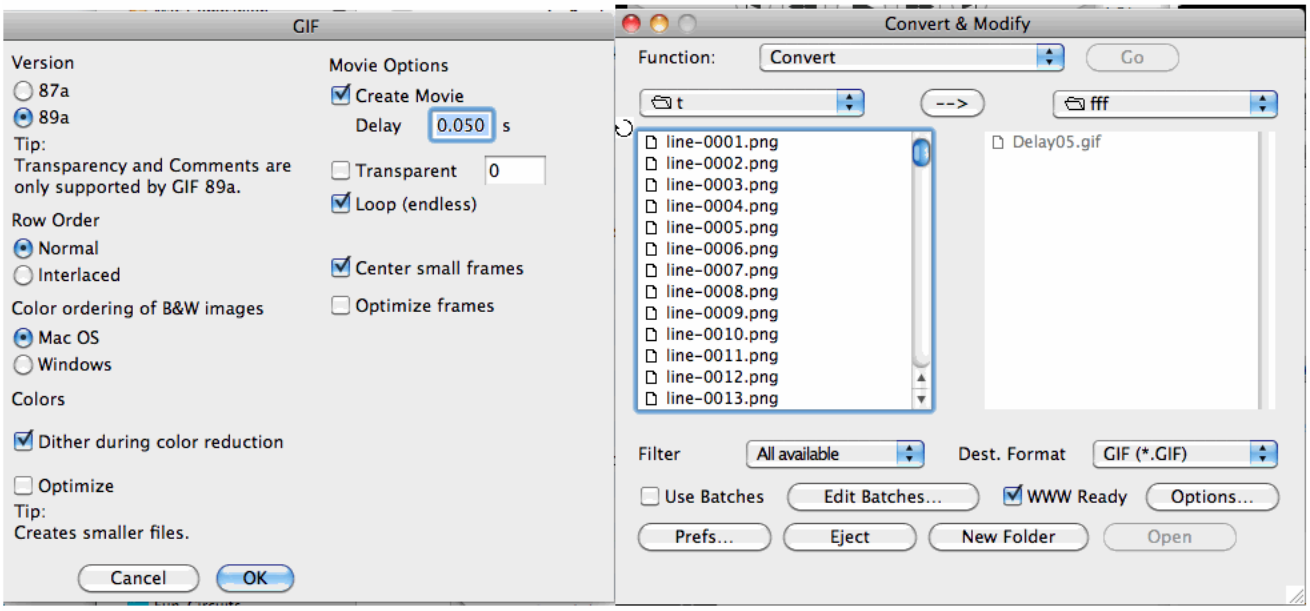



```
int  lastf               = 0;                      // last frame
int  start               = 0;
int  currentTime         = 0;
String str1,str2,str3;
float delay              = 0.0;

void                        setup()
{ size(                    320, 240);
  smooth(                  );
  background(              0);
  frameRate(               10);
  start                   = millis() ;
}
void                        draw()
{ fill(                    255,0,0);
  noStroke(                );
  background(              0);
  if                        ( frameCount == 0 ) start  = millis();
  currentTime              = millis() -start;
  str1                     = frameCount + "   " + currentTime+ "   " + currentTime/frameCount  ;
  delay                    = frameCount*100  -    currentTime;
  str2                     = frameCount*100 + "   " + currentTime+ "   " + str(delay) ;
  str3                     = "jpg";
  text(                      str1, 30, 150);
  text(                      str2, 30, 180);
  text(                      str3, 30, 210);
  ellipse(                   mouseX, mouseY, 20, 20);
  if                        (frameCount == 0  )  start                  = millis() ;
  if                         (frameCount-lastf >0  )
{ saveFrame(                "movieFrames/test-####.jpg"); //".tif", ".tga", ".jpg", or ".png"
  lastf                    = frameCount;
}
  // currentFrame++;
}
```

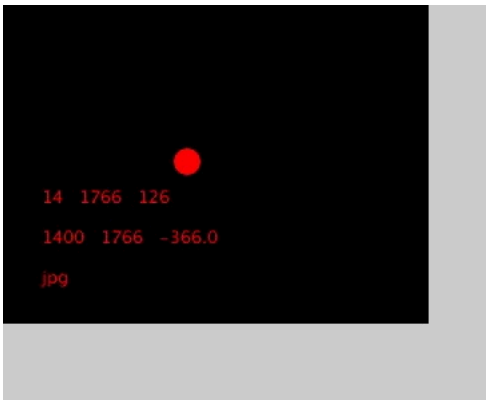
==========Movie_From_Frames_Using_QuickTime==========



==========GraphicConverter_Create_Giff==========

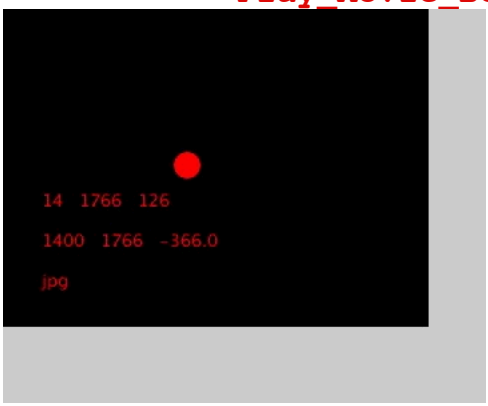


==========Play_Movie==========




```
import processing.video.*;
Movie myMovie;
void setup()
{ size(640, 360);
  myMovie = new Movie(this, "/Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/data/test.mov");
  frameRate(25);
  myMovie.speed(1.0); // sets playback speed
  myMovie.loop();
  println(myMovie.duration());
}
void draw()       // not as good
{ tint(255, 20);
  image(myMovie, 0, 0);
}
void movieEvent(Movie m) { m.read(); }   // Called every time a new frame is available to read

void mousePressed() {  myMovie.noLoop(); }

void keyPressed() {  myMovie.jump(1.0); }
```

==========Play_Movie_Better==========




```
import processing.video.*;   // core.jar kills progrma
Movie myMovie;
void setup()
{ size(640, 360);
  myMovie = new Movie(this, "/Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/data/test.mov");
  frameRate(25);
  myMovie.speed(1.0); // sets playback speed
  myMovie.loop();
  println(myMovie.duration());
}
void draw()   // much better
{ if (myMovie.available()) { myMovie.read();  }
  image(myMovie, 0, 0);
}

void mousePressed()
{ myMovie.pause();
  println(myMovie.time());
}

void mouseReleased() { myMovie.play();}

//void keyPressed() {  myMovie.stop(); }

void keyPressed() {  myMovie.jump(1.0); }
```

==========ReadArduino==========

==========Read/Display_Analog_At_Max_Baud==========

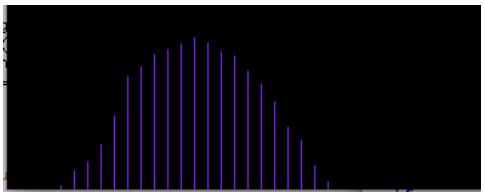
==========Arduino_Code==========

```
void setup()
{ Serial.begin( 115200);                          // initialize to max baud
}
void loop()
{ Serial.println( analogRead(0));                 // send the value of analog input 0:
}
```

==========Processing_Code==========

```
import processing.serial.*;

Serial myPort;                                    // The serial port
int xPos = 1;                                     // horizontal position of the graph

void setup ()
{ size( 400, 300);                                // set the window size:
  println( Serial.list());                        // List all the available serial ports
  myPort = new Serial(this, Serial.list()[0], 115200); // initialize to max baud
  myPort.bufferUntil('\n');                       // no serialEvent() unless newline character:
  background(0);                                  // set inital background:
}                                                 // end setup
void draw () {  }                                 // everything happens in the serialEvent()

void serialEvent (Serial myPort)
{ String inString = myPort.readStringUntil('\n'); // get the ASCII string:
  if (inString != null)
{ inString = trim(inString);                      // trim off any whitespace:
  println( inString);
  float inByte = float(inString);
  inByte = map(inByte, 0, 1023, 0, height);       // convert to int at screen height:
  stroke( 127,34,255);                            // set color used to draw
  line( xPos, height, xPos, height - inByte);     // draw the line:
  if (xPos >= width)
{ xPos = 0;                                       // if edge screen, go back beginning:
  background( 0);                                 // or like background(#FFCC00)
} else
{ xPos= xPos+10;                                  // increment the horizontal position:
}                                                 // end if (xPos >= width)
}                                                 // end if (inString != null)
}                                                 // end serialEvent
```

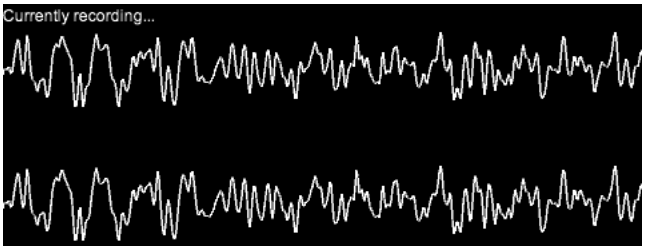
==========Processing_Display==========



The ADC is 10Bit @100us. That is a 10KHz rate. But the data is being shipped serially at 115200 baud. Using 60Hz as a reference, looks like about 20 data points are being display over a 60Hz cycle. That is about a 1200Hz data rate. The overall Sample rate appears to be set by the serial data rate. That is 16.66msec/20 or 833usec per data point. If each data point is 5char "1024," @ 8 bits = 48000 bits/sec

==========Audio==========

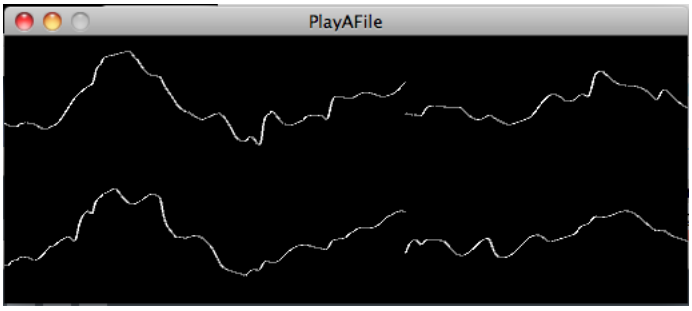
==========Audio_recorder==========




```
import ddf.minim.*;

Minim minim;
AudioInput in;
AudioRecorder recorder;
```

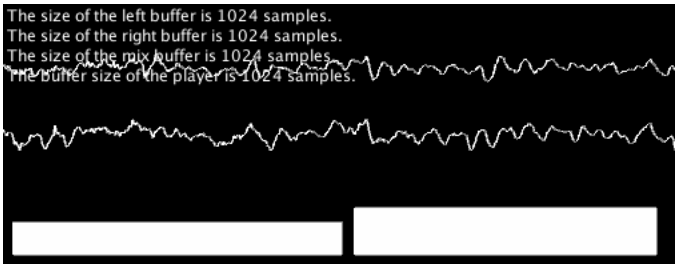
```
AudioPlayer player;

void setup()
{ size(512, 200, P3D);
  minim = new Minim(this);
  in = minim.getLineIn(Minim.STEREO, 2048);
  recorder = minim.createRecorder(in, "myrecording.wav", true);
  textFont(createFont("Arial", 12));
}
void draw()
{ background(0);
  stroke(255);
  for (int i = 0; i < in.bufferSize() - 1; i++)
{ line(i,  50 + in.left.get(i)*50,  i+1,  50 + in.left.get(i+1)*50);
  line(i, 150 + in.right.get(i)*50, i+1, 150 + in.right.get(i+1)*50);
}
 if ( recorder.isRecording() ){text("Currently recording...", 5, 15); }
 else{ text("Not recording.", 5, 15);}
}
void keyReleased()
{ if ( key == 'r' )
{ if ( recorder.isRecording() ) {recorder.endRecord();}
  else {recorder.beginRecord();}
}
 if ( key == 's' )
{ recorder.endRecord();
  recorder.save();
  println("Done saving.");
}
}
void stop()
{ in.close();
  minim.stop();
  super.stop();
  exit();
}
```

==========Audio_player==========



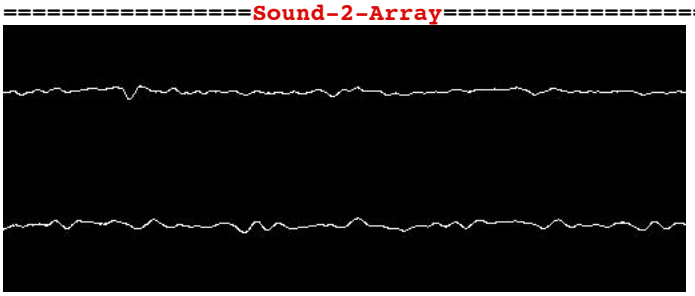

```
import ddf.minim.*;

Minim minim;
AudioPlayer player;

void setup()
{ size(512, 200, P3D);
  minim = new Minim(this);
  player = minim.loadFile("myrecording.wav", 2048);   //  2048 samples long
  player.play();
}

void draw()
{ background(0);
  stroke(255);
  for (int i = 0; i < player.bufferSize() - 1; i++)          // draw the waveforms
{ float x1 = map(  i, 0, player.bufferSize(), 0, width);
  float x2 = map(i+1, 0, player.bufferSize(), 0, width);
  line(x1, 50  + player.left.get(i)*50,  x2, 50  + player.left.get(i+1)*50);
  line(x1, 150 + player.right.get(i)*50, x2, 150 + player.right.get(i+1)*50);
}
}

void keyReleased()
{ player.close();          // always close Minim audio classes when you are done with them
  minim.stop();            // always stop Minim before exiting
  super.stop();
  exit();
}
```

==========Get-Sound==========




```
import      ddf.minim.*;
Minim       minim;
AudioPlayer groove;
AudioOutput out;
void setup()
{ size(512, 200, P3D);
  minim = new Minim(this);
  out = minim.getLineOut();
  groove = minim.loadFile("groove.mp3"); // in data folder
  groove.loop();
  rectMode(CORNERS);
}
void draw()
{ background(0);
  stroke(255);
  text("The size of the left buffer is "   + out.left.size()  + " samples.", 5, 15);
  text("The size of the right buffer is "  + out.right.size() + " samples.", 5, 30);
  text("The size of the mix buffer is "    + out.mix.size()   + " samples.", 5, 45);
  text("The buffer size of the player is " + out.bufferSize() + " samples.", 5, 60);
  rect(10,       height-10, width/2, height - groove.left.level()*500);
  rect(width/2+10, height-10, width-20,   height - groove.right.level()*500);
  for ( int i = 0; i < groove.bufferSize() - 1; i++ )     // multiply values returned by get by 50
{ float x1 = map(i,   0, groove.bufferSize(), 0, width);  // map(value, lowin, highin, lowout, highout)
  float x2 = map(i+1, 0, groove.bufferSize(), 0, width);
  line(x1, height/4   - groove.left.get(i)*50,  x2, height/4   - groove.left.get(i+1)*50);
  line(x1, 2*height/4 - groove.right.get(i)*50, x2, 2*height/4 - groove.right.get(i+1)*50);
}
}
void stop()
{ groove.close();// always close Minim audio classes when you finish with them
  minim.stop();  // always stop Minim before exiting
  super.stop();
}

void keyPressed() { save("pict.jpg"); }
```

==========Sound-2-Array==========



```
import      ddf.minim.*;
Minim       minim;
AudioPlayer player;
void setup()
{ size(512, 200, P3D);
  minim = new Minim(this);
  player = minim.loadFile("groove.mp3");
//player = minim.loadFile("marcus_kellis_theme.mp3", 2048); // load file, give AudioPlayer buffers 2048 samples long
//player = minim.loadFile("http://code.compartmental.net/minim/examples/AudioPlayer/marcus_kellis_theme.mp3", 2048);
//player = minim.loadFile("http://205.188.215.225:8018/", 2048);// load a URL
  player.loop();
}
void draw()
{ background(0);
  stroke(255);
  float[] left  = player.left.toArray();
  float[] right = player.right.toArray();
  for ( int i = 0; i < left.length - 1; i++ )   // loop to left.length - 1 because accessing index i+1 in loop
{ float x1 = map(i,   0, player.bufferSize(), 0, width);
  float x2 = map(i+1, 0, player.bufferSize(), 0, width);
  line(x1, height/4   - left[i]*50,  x2, height/4   - left[i+1]*50);
  line(x1, 3*height/4 - right[i]*50, x2, 3*height/4 - right[i+1]*50);
}
}
void stop()
{ player.close(); // always close Minim audio classes when you finish with them
  minim.stop();// always stop Minim before exiting
  super.stop();
}
```

```
void keyPressed() { save("pict.jpg"); }
```

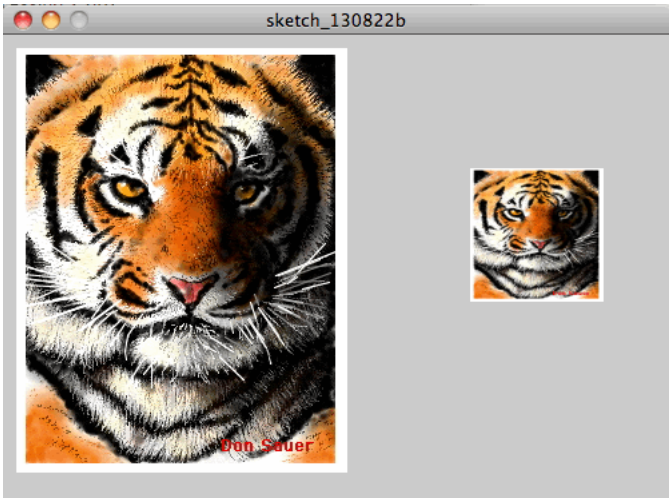
==========scrub-forward-backward==========


```
/**
  * This is a relatively simple file player that lets you scrub-forward-backward in an audio file.<br />
  * It should be noted that it's not *exactly* scrubbing because the playback speed is not changed,
  * it's simply that the position in the song is changed by very small increments when fast-forwarding or rewinding.
  * But the end result is convincing enough.
  * <p>
  * The positioning code is inside of the Play, Rewind, and Forward classes, which are in button.pde.
  */

import      ddf.minim.*;

Minim       minim;
AudioPlayer song;
Play        play;
Rewind      rewind;
Forward     ffwd;

void setup()
{ size(512, 200, P3D);
  minim = new Minim(this);
  song = minim.loadFile("fair1939.wav", 512);  // load a file from the data folder, use a sample buffer of 1024 samples
  play = new Play(width/2 - 50, 130, 20, 10);  // buttons for control
  rewind = new Rewind(width/2, 130, 20, 10);
  ffwd = new Forward(width/2 + 50, 130, 20, 10);
}
void draw()
{ background(0);
  stroke(255);// draw the wave form
  for (int i = 0; i < song.bufferSize() - 1;  i++)
{ line(i, 50 - song.left.get(i)*50, i+1, 50 - song.left.get(i+1)*10);
}
  float x = map(song.position(), 0, song.length(), 0, width);// draw the position in the song in milliseconds,
  stroke(255, 0, 0);
  line(x, 50 - 20, x, 50 + 20);
  play.update();// do the controls
  play.draw();
  rewind.update();
  rewind.draw();
  ffwd.update();
  ffwd.draw();
}
void mousePressed()
{ play.mousePressed();
  rewind.mousePressed();
  ffwd.mousePressed();
}
void mouseReleased()
{ play.mouseReleased();
  rewind.mouseReleased();
  ffwd.mouseReleased();
}
void stop()
{ song.close();// always close Minim audio classes when you are done with them
  minim.stop();
  super.stop();
}
  abstract class Button
{ int x, y, hw, hh;

  Button(int x, int y, int hw, int hh)
{ this.x = x;
  this.y = y;
  this.hw = hw;
  this.hh = hh;
}                                       //
  boolean pressed() { return mouseX > x - hw && mouseX < x + hw && mouseY > y - hh && mouseY < y + hh; }
  abstract void mousePressed();
  abstract void mouseReleased();
  abstract void update();
  abstract void draw();
}                                       //abstract class Button
class Play extends Button
{ boolean play;
  boolean invert;
  Play(int x, int y, int hw, int hh)
{ super(x, y, hw, hh);
```

```
  play = true;
}                                       // Play(int x, int y, int hw, int hh)
  void mousePressed()                   // code to handle playing and pausing the file
{ if ( pressed() )
{ invert = true;
  if ( song.isPlaying() )
{ song.pause();
  play = true;
}                                       // if ( song.isPlaying() )
  else
{ song.loop();
  play = false;
}                                      //  else
}                                      //  if ( pressed() )
}                                       //  void mousePressed()
  void mouseReleased()                 { invert = false;  }

  void update()        // play is a boolean value used to determine what to draw on the button
{ if ( song.isPlaying() ) play = false;
  else play = true;
}                                       //  void update()
  void draw()
{ if ( invert )
{ fill(255);
  stroke(0);
}                                       // if ( invert )
  else
{ noFill();
  stroke(255);
}                                       // else
  rect(x - hw, y - hh, hw*2, hh*2);
  if ( invert )
{ fill(0);
  stroke(255);
}                                       //if ( invert )
  else
{ fill(255);
  noStroke();
}                                       // else
  if ( play ){ triangle(x - hw/3, y - hh/2, x - hw/3, y + hh/2, x + hw/2, y);}
  else
{ rect(x - hw/3, y - hh/2, hw/4, hh);
  rect(x + hw/8, y - hh/2, hw/4, hh);
}                                       // else
}                                       // void draw()
}                                       //void mouseReleased

class Rewind extends Button
{ boolean invert;
  boolean pressed;
  Rewind(int x, int y, int hw, int hh)
{ super(x, y, hw, hh);
  invert = false;
}                                       // Rewind(int x, int y, int hw, int hh)
  void update()                         // code used to scrub backward in the file
{ if (pressed)                          // if the rewind button is currently being pressed
{ int pos = song.position();           // get the current song position
  if ( pos > 200 )                      // if it greater than 200 milliseconds
{ song.skip(-200);                      // rewind the song by 200 milliseconds
}                                       // if ( pos > 200 )
  else
{ song.rewind();                        // if song hasn't played more than 100 milliseconds rewind to beginning
}                                       // else
}                                       // if (pressed)
}                                       // void update()
  void mousePressed()
{ pressed = pressed();
  if ( pressed )
{ invert = true;
  if ( !song.isPlaying() ) song.rewind();       // if the song isn't currently playing, rewind it to the beginning
}                                      //  if ( pressed )
}                                       //  void mousePressed()
  void mouseReleased()
{ pressed = false;
  invert = false;
}                                       //  void mouseReleased()
  void draw()
{ if ( invert )
{ fill(255);
  stroke(0);
}                                       // if ( invert )
  else
{ noFill();
  stroke(255);
}                                       // else
  rect(x - hw, y - hh, hw*2, hh*2);
  if ( invert )
{ fill(0);
  stroke(255);
}                                       //  if ( invert )
  else
{ fill(255);
  noStroke();
}                                       // else
  triangle(x - hw/2, y, x, y - hh/2, x, y + hh/2);
  triangle(x, y, x + hw/2, y - hh/2, x + hw/2, y + hh/2);
```

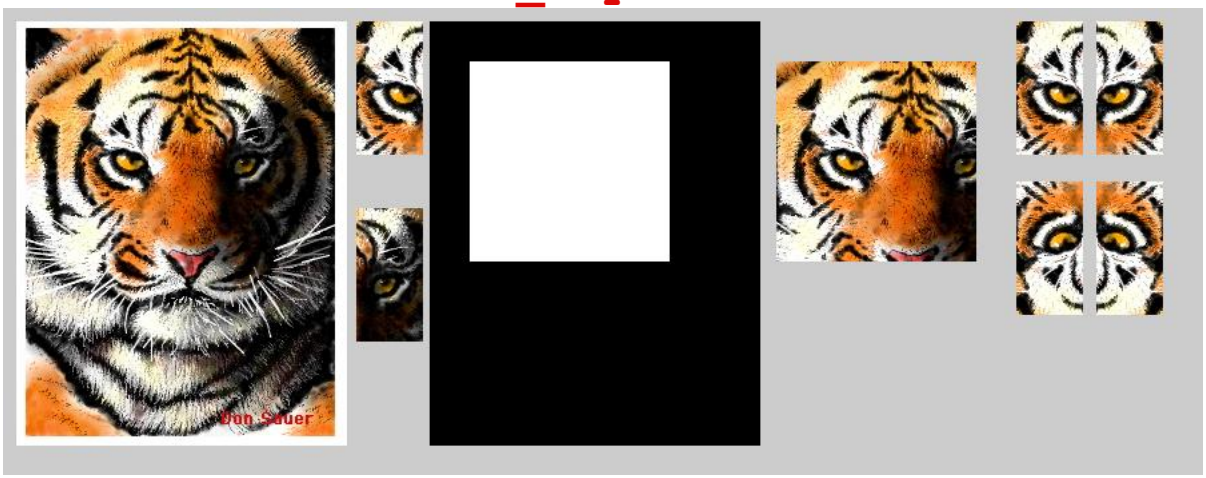
```
}                                       // void mouseReleased()
}                                        //   class Rewind extends Button

  class Forward extends Button
{ boolean invert;
  boolean pressed;
  Forward(int x, int y, int hw, int hh)
{ super(x, y, hw, hh);
  invert = false;
}                                       // Forward(int x, int y, int hw, int hh)
  void update()
{ if (pressed)                          // if the forward button is currently being pressed
{ int pos = song.position();            // get the current position of the song
  if ( pos < song.length() - 40 )       // if the song's position is more than 40 milliseconds from the end of the song
{ song.skip(40);                        // forward the song by 40 milliseconds
}                                       // if ( pos < song.length() - 40 )
  else
{ song.cue( song.length() );           // otherwise, cue the song at the end of the song
}                                      //else
  song.play();                         // start the song playing
}                                       // if (pressed)
}                                       //  void update()
  void mousePressed()
{ pressed = pressed();
  if ( pressed )
{ invert = true;
}                                       // if ( pressed )
}                                        // void mousePressed()
  void mouseReleased()
{ pressed = false;
  invert = false;
}                                        //void mouseReleased()
  void draw()
{ if ( invert )
{ fill(255);
  stroke(0);
}                                       //if ( invert )
  else
{ noFill();
  stroke(255);
}                                       // noFill
  rect(x - hw, y - hh, hw*2, hh*2);
  if ( invert )
{ fill(0);
  stroke(255);
}                                       //if ( invert )
  else
{ fill(255);
  noStroke();
}                                       //else
  triangle(x, y, x - hw/2, y - hh/2, x - hw/2, y + hh/2);
  triangle(x, y - hh/2, x, y + hh/2, x + hw/2, y);
}                                        //  void draw()
}                                        //  class Forward extends Button

void keyPressed() { save("pict.jpg"); }
```

======================Resize_image==================



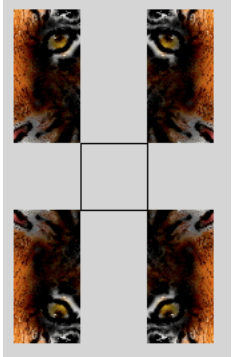

```
  size(                    500, 350);                                    // size window
  PImage img             = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");  // loads image into Pimage Object
  image(                   img, 10, 10);                                 // display image at x=10, y=10
  PImage img2            = img.get();
  img2.resize(             100, 100);  ;                                 // loads image into Pimage Object
  image(                   img2, 350, 100);
```

==========PIMAGE_AlphaMask==========




```
  size(                          900, 350);                                    // size window
  PImage img                   = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");  // loads image into Pimage Object
  image(                         img, 10, 10);                               // display image at x=10, y=10
//=============================================
  PImage img2                  = img.get(50, 50, 50, 100);                   // x y w h  #####
  PImage img3                  = img.get(150, 50, 50, 100);                  // x y w h
  image(                         img2, 265, 10);                             //
  image(                         img3, 265, 150);
//=============================================
  PImage img4                  = createImage(img.width, img.height, RGB);    // Create opaque image same size  ######
  for                            (int y = 30; y < 180; y++)                  // Loop through every pixel in the image.
{ for                            (int x = 30; x < 180; x++)                  // Skip left and right edges
{ color c                    = color(255, 255, 255);                       // need white to show picture
  img4.set(                      x, y, c);
}
}
//=============================================
  img.mask(                      img4);                                      // Masks image ######################
  image(                         img, 550, 10);
//=============================================
  image(                         img4,320, 10);                              // View Mask
//=============================================
  image(                         img2, 760, 10);
  pushMatrix();
  translate(                     870, 10);
  scale(                        -1,1);
  image                          (img2, 0, 0);
  popMatrix();
  pushMatrix();
  translate(                     760, 230);
  scale(                         1,-1);
  image(                         img2, 0, 0);
  popMatrix();
  pushMatrix();
  translate(                     870, 230);
  scale(                         -1,-1);
  image(                         img2, 0, 0);
//=============================================
  save(                         "pict.jpg");
```

==========Crop-Flip==========



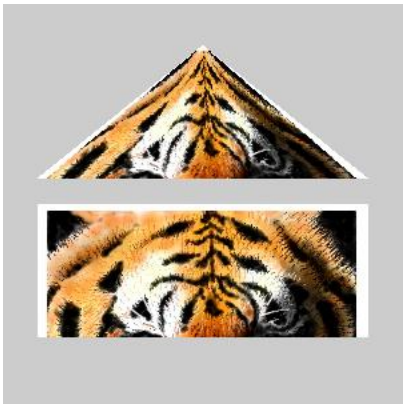
```
PImage img , img2  ;
```

```
 int w, h ;
  void                          setup()
{ size(                         300, 300);
  img                          = loadImage("/Users/don_sauer/Desktop/Tiger.jpg"); // loads image into Pimage Object
  w                            = img.width;
  h                            = img.height;
  println(                      "width = "+img.width + " height = "+img.height );
}
  void                          draw()
{ img2                         = img.get( constrain(mouseX,0,w-50), constrain(mouseY ,0,h-100), 50, 100);   // x y w h
  image(                        img2, 10, 10);                      // x y
  pushMatrix();
  translate(                    160, 10);
  scale(                       -1,1);
  image                         (img2, 0, 0);                      // x y
  popMatrix();
  pushMatrix();
  translate(                    10, 260);
  scale(                        1,-1);
  image(                        img2, 0, 0);                     // x y
  popMatrix();
  pushMatrix();
  translate(                    160, 260);
  scale(                        -1,-1);
  image(                        img2, 0, 0);                     // x y
  popMatrix();
  noFill();
  rect(                         60, 110, 50, 50);
  }
```

==========Map==========

```
 PImage img , img2 , img3 ;
 int w, h ;
 float xscal, xoff;
  void                          setup()
{ size(                         300, 300);
  img                          = loadImage("/Users/don_sauer/Desktop/Tiger.jpg"); // loads image into Pimage Object
  w                            = img.width;
  h                            = img.height;
  img3                         = img.get( 0, 0, img.width, 100);   // x y w h
}
  void                          draw()
{ //background(204);
  for                           (int i=0; i <=100;  i++)
{ img2                         = img.get( w/2 -w*i/200, 1*i, w*i/100, 1);   // x y w h
  xscal =                       w;
  xoff =                        150-xscal/2 ;
  image(                        img2, xoff, 1*i+30, xscal,1);       // x y w h
}
  image(                        img3, width/2-w/2, 150);       // x y w h
  strokeWeight(2);
  stroke(0,255,255);                                  // color
  line( width/2-w/2,250,width/2+w/2,250);
  line( width/2-w/2,250,width/2 ,150);
  line( width/2,150,width/2+w/2,250);
}

void                            mouseClicked()
{ save(                        "pict.jpg");                                    // Mouse has been clicked
}
```

==========Map==========



```
PImage img , img2 , img3 ;
int w, h ;
float xscal, xoff;
 void                          setup()
{ size(                        300, 300);
  img                         = loadImage("/Users/don_sauer/Desktop/Tiger.jpg"); // loads image into Pimage Object
  w                           = img.width;
  h                           = img.height;
  img3                        = img.get( 0, 0, img.width, 100);   // x y w h
}
 void                          draw()
{ //background(204);
  for                          (int i=0; i <=100;  i++)
{ img2                        = img.get( 0, 1*i, img.width, 1);   // x y w h
  xscal =                      float(i*w/100);
  xoff =                       150-xscal/2 ;
  image(                       img2, xoff, 1*i+30, xscal,1);    // x y w h
  image(                       img3, width/2-w/2, 150);         // x y w h
}
}

void                           mouseClicked()
{ save(                       "pict.jpg");                                  // Mouse has been clicked
}
```

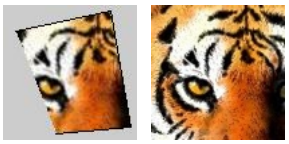
==========Texture==========

==========TEXTURE==========


```
  size(                        800, 350, P3D);
  PImage img                  = loadImage("/Users/don_sauer/Desktop/Tiger.jpg"); // loads image into Pimage Object
  int w                       = img.width;
  int h                       = img.height;
  image(                       img, 10, 10);          // display image at x=10, y=10
//==================================
  strokeWeight(4);
  noFill();
  stroke(0,255,255);                                  // color
  beginShape(QUADS);                                  //
  vertex(                      50, 50);               // top left
  vertex(                      180, 75);              // top right
  vertex(                      195, 190);             // bottom right
  vertex(                      40, 195);              // bottom left
  endShape();                                         // draw shape
//==================================
  pushMatrix();                                       // save current
  translate(                    250,-20);
  noStroke();
```

```
  beginShape();
  texture(                      img);               // vertex( x, y, u, v)
  vertex(                       50,  50,  0, 0);    // top left
  vertex(                       180, 75,  w/2, 0);  // top right
  vertex(                       195, 190, w/2, h/2); // bottom right
  vertex(                       40,  195, 0,   h/2); // bottom left
  endShape();                                       // draw shape
//==================================
  translate(                    160,120);
  beginShape();
  texture(                      img);               // vertex( x, y, u, v)
  vertex(                       50,  50,  0, 0);    // top left
  vertex(                       180, 75,  w, 0);    // top right
  vertex(                       195, 190, w, h);    // bottom right
  vertex(                       40,  195, 0, h);    // bottom left
  endShape();
//==================================
  translate(                    350,-100);          //  if reverse direction of X need this offset
  rotate(                       PI/3);
  beginShape();
  texture(                      img);
  vertex(                       50,  50,  w, 0);    // top left
  vertex(                       180, 75,  0, 0);    // top right
  vertex(                       195, 190, 0, h);    // bottom right
  vertex(                       40,  195, w, h);    // bottom left
  endShape();                                       // draw shape
//==================================
  popMatrix();
  save(                         "pict.jpg");
```

==========Texture-to-clip==========



```
PImage a = loadImage("/Users/donsauer/Downloads/Tiger100.jpg");
  size(100, 100, P3D);
  beginShape();
  texture(a);              // "Tiger100" is 100x100 ,0 -> 100 are used for "u" and "v" to
  vertex(10, 20, 10, 20);
  vertex(80, 5,  80, 5);
  vertex(95, 90, 95, 90);
  vertex(40, 95,40, 95);
  endShape();
  save("pict.jpg");
```

==========Texture-to-clip-Rotate==========

```
size(200, 200, P3D);
PImage a = loadImage("/Users/donsauer/Downloads/Tiger100.jpg");

pushMatrix();
translate(60, 60);
scale(-1,1);

beginShape();
texture(a);              // "Tiger100" is 100x100 ,0 -> 100 are used for "u" and "v" to
vertex(10, 20, 10, 20);
vertex(80, 5,  80, 5);
vertex(95, 90, 95, 90);
vertex(40, 95,40, 95);
endShape();
popMatrix();

save("pict.jpg");
```

==========Texture-Mapping==========

```
size(100, 100, P3D);
```

```
PImage a = loadImage("/Users/donsauer/Downloads/Tiger100.jpg");
beginShape();
texture(a); // "Tiger100" is 100x100 ,0 -> 100 are used for "u" and "v" to
vertex(10, 20, 0,   0);
vertex(80, 5,  100, 0);
vertex(95, 90, 100, 100);
vertex(40, 95, 0,   100);
endShape();
save("pict.jpg");
```

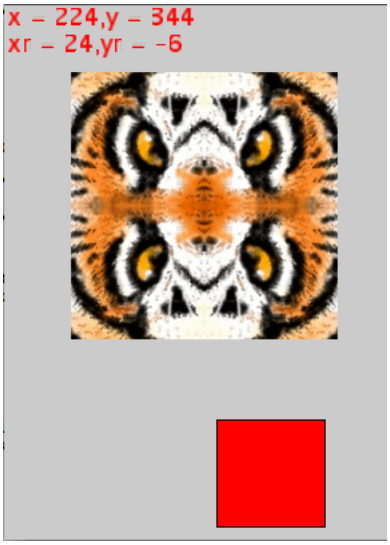
```
size(100, 100, P3D);
PImage a = loadImage("/Users/donsauer/Downloads/Tiger100.jpg");
beginShape();
texture(a); // "Tiger100" is 100x100 ,0 -> 100 are used for "u" and "v" to
vertex(10, 20, 0, 0);
vertex(80, 5,  50, 0);
vertex(95, 90, 50, 50);
vertex(40, 95, 0, 50);
endShape();
save("pict.jpg");

noStroke();
```

ze: 248 × 318 pixels

```
size(100, 100, P3D);
PImage a = loadImage("/Users/donsauer/Downloads/Tiger.jpg");
beginShape();
texture(a); // "Tiger100" is 248x318 ,used for "u" and "v" to
vertex(10, 20, 0, 0);
vertex(80, 5,  248, 0);
vertex(95, 90, 248, 318);
vertex(40, 95, 0, 318);
endShape();
save("pict.jpg");
```

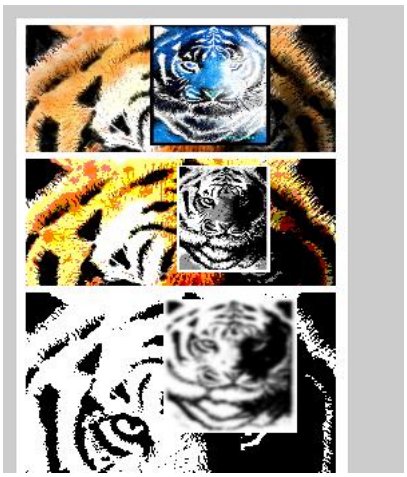
```
size(100, 100, P3D);
noStroke();
PImage a = loadImage("/Users/donsauer/Downloads/Tiger.jpg");
textureMode(NORMALIZED);
beginShape();
texture(a);
vertex(10, 20, 0, 0); //vertex(x, y, u, v);
vertex(80, 5,  1, 0);
vertex(95, 90, 1, 1);
vertex(40, 95, 0, 1);
endShape();
save("pict.jpg");
```

==========Translate_Flip_Rotate_TEXTURE==========



```
import                    java.awt.event.*;

PImage                    img ,img2 ;     // loads image into Pimage Object
String                    str1, str2 ;
int                       x =0, y=0 , xr =0, yr=0, dial=0;
int                       xorg = 200, yorg = 350, wid =80;
float                     R, A, DX, DY, xa, ya;

void                      setup()
{ size(                   400, 400, P3D);
  stroke(                 0);
  img                   = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");    // loads image into Pimage Object
  fill(                   255, 0, 0);                                        // text will use fill
  textSize(               18);                                               //
  addMouseWheelListener( new MouseWheelListener()
{ public void mouseWheelMoved(MouseWheelEvent mwe) { mouseWheel(mwe.getWheelRotation()); } }  ); //####################
}

void                      draw()
{ background(             204);
  stroke(                 0);
  rect(                   xorg-wid/2,yorg-wid/2, wid, wid );                // lines will use stroke// redraw gray background
  str1                  = "x = "+str(x)+ ",y = "+ str(y)  ;
  text(                   str1, 3, 15);                                     // 3 pixel across and 15 down
  str2                  = "xr = "+str(xr)+ ",yr = "+ str(yr)  ;
  text(                   str2, 3, 35);                                     // 3 pixel across and 15 down
  noStroke();
  A                     = radians(45);
  xa                    = (float) dial/10;
  ya                    = (float) yr;
  R                     = 50 ;

  beginShape();
  texture(                img);                                             // "Tiger100" is 100x100 ,0 -> 100 used for "u" and
"v" to
  vertex(                 50,   50, 100-R*cos(A-xa)-xr, 100-R*sin(A-xa)-yr);      // top left  x, y , u , v
  vertex(                 150,  50, 100+R*cos(A+xa)-xr, 100-R*sin(A+xa)-yr);      // top right
  vertex(                 150, 150, 100+R*cos(A-xa)-xr, 100+R*sin(A-xa)-yr);      // bottom right
  vertex(                 50,  150, 100-R*cos(A+xa)-xr, 100+R*sin(A+xa)-yr);      // bottom left
  endShape();

  beginShape();
  texture(                img);                                             // "Tiger100" is 100x100 ,0 -> 100 used for "u"
and "v" to
  vertex(                 150,   50, 100+R*cos(A+xa)-xr, 100-R*sin(A+xa)-yr);      // top left  x, y , u , v
  vertex(                 250,   50, 100-R*cos(A-xa)-xr, 100-R*sin(A-xa)-yr);      // top right
  vertex(                 250,  150, 100-R*cos(A+xa)-xr, 100+R*sin(A+xa)-yr);      // bottom right
  vertex(                 150,  150, 100+R*cos(A-xa)-xr, 100+R*sin(A-xa)-yr);      // bottom left
  endShape();

  beginShape();
  texture(                img);                                             // "Tiger100" is 100x100 ,0 -> 100  used for "u"
and "v" to
  vertex(                 50,   150, 100-R*cos(A+xa)-xr, 100+R*sin(A+xa)-yr);      // top left  x, y , u , v
  vertex(                 150,  150, 100+R*cos(A-xa)-xr, 100+R*sin(A-xa)-yr);      // top right
  vertex(                 150,  250, 100+R*cos(A+xa)-xr, 100-R*sin(A+xa)-yr);      // bottom right
  vertex(                 50,   250, 100-R*cos(A-xa)-xr, 100-R*sin(A-xa)-yr);      // bottom left
  endShape();

  beginShape();
  texture(                img);                                             // "Tiger100" is 100x100 ,0 -> 100 used for "u"
and "v" to
  vertex(                 150,   150, 100+R*cos(A-xa)-xr, 100+R*sin(A-xa)-yr);      // top left  x, y , u , v
  vertex(                 250,   150, 100-R*cos(A+xa)-xr, 100+R*sin(A+xa)-yr);      // top right
  vertex(                 250,   250, 100-R*cos(A-xa)-xr, 100-R*sin(A-xa)-yr);      // bottom right
  vertex(                 150,   250, 100+R*cos(A+xa)-xr, 100-R*sin(A+xa)-yr);      // bottom left
  endShape();
}
void                      mouseMoved()                                      // Click to add a line segment
{ x                     = mouseX; y =  mouseY;
  xr                    = x - xorg ; yr = y - yorg;
  xr                    = constrain(xr, -40, 40);
  yr                    = constrain(yr, -40, 40);
}
void                      mouseDragged()                                    // Click to add a line segment
{ x                     = mouseX; y =  mouseY;
  xr                    = x - xorg ; yr = y - yorg;
  xr                    = constrain(xr, -40, 40);
  yr                    = constrain(yr, -40, 40);
}
void                      mouseWheel(int delta)
{ println(                "mouse has moved by " + delta + " units.");
  dial                  = dial+ delta;
}
```

==========FILTER_IMAGE_AND_CORNERS==========



```
size(                   300,350);
PImage img             = loadImage("/Users/don_sauer/Desktop/Tiger.jpg"); // loads image into Pimage Object
image(                  img, 10, 10);
imageMode(              CORNER);
img.filter(             INVERT);
image(                  img, 110, 15, 90,90);              // x, y w, h
img.filter(             INVERT);
img.filter(             POSTERIZE,3);
image(                  img, 10, 110);
imageMode(              CORNERS);
img.filter(             GRAY);
image(                  img, 130, 120,  200, 200);        // left top X, Y , right bottom X,Y
img.filter(             THRESHOLD, .3);
image(                  img, 10, 210);
imageMode(              CENTER);
img.filter(             BLUR, 3);
image(                  img, 170, 270, 100,100);          // x, y, w, h
save(                   "pict.jpg");
```

==========BLEND==========


```
size(                   300, 400);
PImage img             = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");    // loads image into Pimage Object
println (               "width  = " +img.width      );   // width = 248
println (               "height = " +img.height     );   // height = 318
PImage img2            = createImage(img.width, img.height, RGB);
img2.loadPixels();
for                     (int i = 0; i < img2.pixels.length; i++) { img2.pixels[i] = color(255, 0, 0); }
img2.updatePixels();
img.blend(              img2, 0, 0, 100, 100, 0, 0, 100, 100, MULTIPLY);     // inx iny inw inh outx outy outw outh mode
for                     (int i = 0; i < img2.pixels.length; i++) { img2.pixels[i] = color(0, 255, 0); }
img2.updatePixels();
img.blend(              img2, 0, 100, 100, 100, 0, 100, 100, 100, MULTIPLY);  // inx iny inw inh outx outy outw outh mode
for                     (int i = 0; i < img2.pixels.length; i++) { img2.pixels[i] = color(0, 0,255); }
img2.updatePixels();
img.blend(              img2, 0, 200, 100, 100, 0, 200, 100, 100, MULTIPLY);  // inx iny inw inh outx outy outw outh mode
image(                  img, 10, 10);
save(                   "pict.jpg");
```

==========

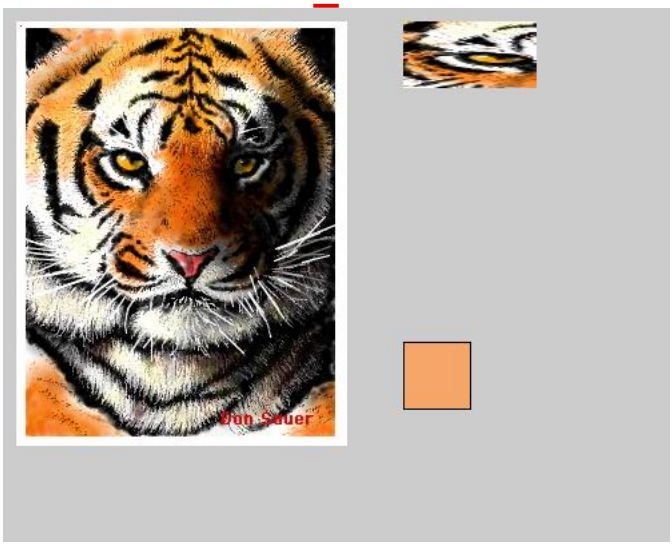
```
BLEND -       linear interpolation of colours: C = A*factor + B
ADD -         additive blending with white clip: C = min(A*factor + B, 255)
SUBTRACT -    subtractive blending with black clip: C = max(B - A*factor, 0)
DARKEST -     only the darkest colour succeeds: C = min(A*factor, B)
LIGHTEST -    only the lightest colour succeeds: C = max(A*factor, B)
DIFFERENCE -  subtract colors from underlying image.
EXCLUSION -   similar to DIFFERENCE, but less extreme.
```

```
MULTIPLY -   Multiply the colors, result will always be darker.
SCREEN -     Opposite multiply, uses inverse values of the colors.
OVERLAY -    A mix of MULTIPLY and SCREEN. Multiplies dark values, and screens light values.
HARD_LIGHT - SCREEN when greater than 50% gray, MULTIPLY when lower.
SOFT_LIGHT - Mix of DARKEST and LIGHTEST. Works like OVERLAY, but not as harsh.
DODGE -      Lightens light tones and increases contrast, ignores darks. Called "Color Dodge" in Illustrator and Photoshop.
BURN -       Darker areas are applied, increasing contrast, ignores lights. Called "Color Burn" in Illustrator and Photoshop.
```

==========MASK==========

```
size(300, 400);
PImage img = loadImage("/Users/donsauer/Downloads/Tiger.jpg");
PImage maskImg = loadImage("/Users/donsauer/Downloads/Tiger.jpg"); // must be same size
img.mask(maskImg);              // only blue channel is used.
image(img, 10, 10);
save("pict.jpg");
```

==========GET_Pixels==========




```
size(500, 400);
PImage img = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");
color red = color(255, 0, 0);
img.set(3, 3, red);
image(img, 10, 10);
println ( "width  = " +img.width      );   // width = 248
println ( "height = " +img.height     );   // height = 318

PImage img2 = img.get(50, 50, 50, 100);   // x y w h
img2.resize(100, 50);                      // w h
image(img2, 300, 10);                      // x y

color c = img.get(60, 10);
fill(c);
rect(300, 250, 50, 50);

save("pict.jpg");
```

==========Graphing_2D_Equations==========

```
void setup()
{ size(640, 360);
}

void draw()
{ loadPixels();
  float n = (mouseX * 10.0) / width;
  float w = 16.0;                                          // 2D space width
  float h = 16.0;                                          // 2D space height
  float dx = w / width;                                    // Increment x this amount per pixel
  float dy = h / height;                                   // Increment y this amount per pixel
  float x = -w/2;                                          // Start x at -1 * width / 2
  for (int i = 0; i < width; i++)
{ float y = -h/2;                                          // Start y at -1 * height / 2
  for (int j = 0; j < height; j++)
{ float r = sqrt((x*x) + (y*y));                           // Convert cartesian to polar
  float theta = atan2(y,x);                                // Convert cartesian to polar
  float val = sin(n*cos(r) + 5 * theta);                   // Results in a value between -1 and 1 // Compute 2D polar coordinate function
  //float val = cos(r);                                    // Another simple function
  //float val = sin(theta);                                // Another simple function
  pixels[i+j*width] = color((val + 1.0) * 255.0/2.0);      // Scale to between 0 and 255// Map resulting vale to grayscale value
  y += dy;                                                 // Increment y
}
  x += dx;                                                 // Increment x
}
  updatePixels();
}
```

==========Process_Pixels==========


```
PImage img;
void setup()
{ img = loadImage("/Users/don_sauer/Desktop/Tiger.jpg");
  size(img.width,img.height);
}
void draw()
{ loadPixels();   // We must also call loadPixels() on the PImage since we are going to read its pixels.
  img.loadPixels();
  for (int y = 0; y < height; y++ )
{ for (int x = 0; x < width; x++ )
{ int loc = x + y*width;
  float r = red(img.pixels [loc]);   // The functions red(), green(), and blue() pull out the three color components from a pixel.
  float g = green(img.pixels[loc]);
  float b = blue(img.pixels[loc]);
  // Image Processing would go here
  // If we were to change the RGB values, we would do it here, before setting the pixel in the display window.
  pixels[loc] = color(b,g,r);// Set the display pixel to the image pixel
}
} updatePixels();
}
```

==========Convolution_Pixels==========

Applies a convolution matrix to a portion of the index.
Move mouse to apply filter to different parts of the image.

```
PImage img;
int    w = 80;
  float[][] matrix = { { -1, -1, -1 },
                       { -1,  9, -1 },
                       { -1, -1, -1 } };
void setup()
{ size(200, 200);
  frameRate(30);
  img = loadImage("/Users/don_sauer/Desktop/Tiger200.jpg");
}
void draw()      // only going to process a portion of the image
{ image(img,0,0);  // so let's set the whole image as the background first
  int xstart = constrain(mouseX-w/2,0,img.width); // Where is the small rectangle we will process
  int ystart = constrain(mouseY-w/2,0,img.height);
  int xend = constrain(mouseX+w/2,0,img.width);
  int yend = constrain(mouseY+w/2,0,img.height);
  int matrixsize = 3;
  loadPixels();
  for (int x = xstart; x < xend; x++) // Begin our loop for every pixel
{ for (int y = ystart; y < yend; y++ )
{ color c = convolution(x,y,matrix,matrixsize,img);
  int loc = x + y*img.width;
  pixels[loc] = c;
}
}
  updatePixels();
}
color convolution(int x, int y, float[][] matrix,int matrixsize, PImage img)
{ float rtotal = 0.0;
  float gtotal = 0.0;
  float btotal = 0.0;
  int offset = matrixsize / 2;
  for (int i = 0; i < matrixsize; i++)
{ for (int j= 0; j < matrixsize; j++)
{ int xloc = x+i-offset;// What pixel are we testing
  int yloc = y+j-offset;
  int loc =  xloc + img.width*yloc;
  loc = constrain(loc,0,img.pixels.length-1);// Make sure on image
  rtotal += (red(img.pixels[loc]  ) * matrix[i][j]);// Calculate the convolution
  gtotal += (green(img.pixels[loc]) * matrix[i][j]);
  btotal += (blue(img.pixels[loc] ) * matrix[i][j]);
}
}
  rtotal = constrain(rtotal,0,255);// Make sure RGB is within range
  gtotal = constrain(gtotal,0,255);
  btotal = constrain(btotal,0,255);
  return color(rtotal,gtotal,btotal);// Return the resulting color
}

void keyPressed() { save("pict.jpg"); }
```

==========Blur==========

Bluring half of an image by processing it through a low-pass filter.


```
float     v = 1.0/9.0;
float[][] kernel = { { v, v, v },
                     { v, v, v },
                     { v, v, v } };
```

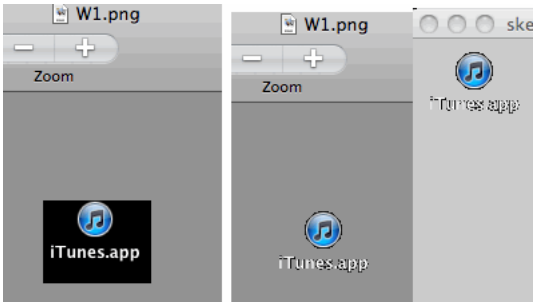
```
size(200, 200);
PImage img = loadImage("/Users/don_sauer/Desktop/Tiger200.jpg"); // Load the original image
image(img, 0, 0);                                               // Displays the image from point (0,0)
img.loadPixels();

PImage edgeImg = createImage(img.width, img.height, RGB);// Create opaque image same size


  for (int y = 1; y < img.height-1; y++)    // Skip top and bottom edges// Loop through every pixel in the image.
{ for (int x = 1; x < img.width-1; x++)     // Skip left and right edges
{ float sum = 0;                            // Kernel sum for this pixel
  for (int ky = -1; ky <= 1; ky++)
{ for (int kx = -1; kx <= 1; kx++)
{ int pos = (y + ky)*img.width + (x + kx);  // Calculate the adjacent pixel for this kernel point
  float val = red(img.pixels[pos]);         // Image is grayscale, red/green/blue are identical
  sum += kernel[ky+1][kx+1] * val;          // Multiply adjacent pixels based on the kernel values
}
}
  edgeImg.pixels[y*img.width + x] = color(sum);  // set gray value sum from kernel
}
}
  edgeImg.updatePixels();     // State that there are changes to edgeImg.pixels[]
  image(edgeImg, 100, 0);     // Draw the new image
```

==========Edge-Detection==========

Edge-Detection.Exposing areas of contrast within an image by processing it through a high-pass filter.

```
float[][] kernel = { { -1, -1, -1 },
                     { -1,  9, -1 },
                     { -1, -1, -1 } };
size(200, 200);
PImage img = loadImage("/Users/don_sauer/Desktop/Tiger200.jpg"); // Load the original image
image(img, 0, 0);     // Displays the image from point (0,0)
img.loadPixels();

PImage edgeImg = createImage(img.width, img.height, RGB); // Create an opaque image of the same size as the original
for (int y = 1; y < img.height-1; y++) // Skip top and bottom edges// Loop through every pixel in the image.
{ for (int x = 1; x < img.width-1; x++) // Skip left and right edges
{ float sum = 0; // Kernel sum for this pixel
  for (int ky = -1; ky <= 1; ky++)
{ for (int kx = -1; kx <= 1; kx++)
{ int pos = (y + ky)*img.width + (x + kx); // Calculate the adjacent pixel for this kernel point
  float val = red(img.pixels[pos]);        // Image is grayscale, red/green/blue are identical
  sum += kernel[ky+1][kx+1] * val;         // Multiply adjacent pixels based on the kernel values
}
}
  edgeImg.pixels[y*img.width + x] = color(sum);   // For this pixel in the new image, set the gray value// based on the sum from the
kernel
}
}
edgeImg.updatePixels();     // State that there are changes to edgeImg.pixels[]
image(edgeImg, 100, 0);     // Draw the new image
```

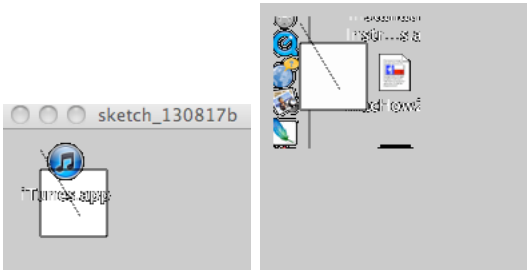
==========Color_To_Transparent==========




```
String FILENAME = "/Users/don_sauer/Downloads/W1.png";
PImage img;
PImage newImg;
int x;
int y;
int i;
```

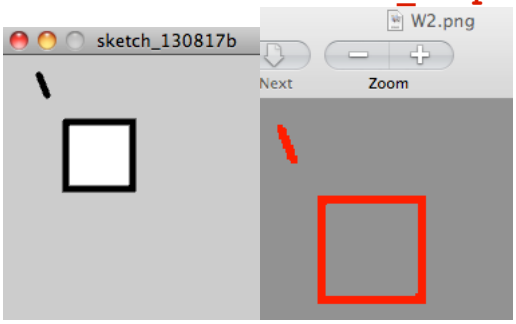
```
void setup()
{ size(200, 200);
  img = loadImage( FILENAME );
  newImg = createImage( img.width, img.height, ARGB );
  for( x = 0; x < img.width; x++ )
{ for( y = 0; y < img.height; y++ )
{ i = ( ( y * img.width ) + x );
  if( img.pixels[i] < color( 5, 5, 5 ) ){ newImg.pixels[i] = color( 0, 0, 0, 0 );}
  else { newImg.pixels[i] = img.pixels[i]; }
} //for( y = 0; y < img.height; y++ )
} //for( x = 0; x < img.width; x++ )
  newImg.save( FILENAME );
}

void draw() { image(newImg, 10, 10); }
```

==========Images_together_with_Shapes==========




```
String FILENAME = "/Users/don_sauer/Downloads/W1.png";
PImage img;
PImage newImg;
PShape line, square ;

int x;
int y;
int i;
void setup()
{ size(200, 200,P3D);
  line                = createShape(LINE,   0, 0,30,50 );
  square              = createShape(RECT,   0, 0, 50, 50);
  img                 = loadImage( FILENAME );
  newImg              = createImage( img.width, img.height, ARGB );
  for                   ( x = 0; x < img.width; x++ )
{ for                   ( y = 0; y < img.height; y++ )
{ i                   = ( ( y * img.width ) + x );
  if                    ( img.pixels[i] == color( 0, 0, 0 ) )
{ newImg.pixels[i]    = color( 0, 0, 0, 0 );
} //                     if( img.pixels[i] == color( 0, 0, 0 ) )
  else                  { newImg.pixels[i] = img.pixels[i]; }
} //                     for( y = 0; y < img.height; y++ )
} //                     for( x = 0; x < img.width; x++ )
  newImg.save(          FILENAME );     // x1 y1 x2 y2
} //                     setup()

void draw()
{ shape(                line,      30, 15);
  shape(                square,    30, 30);
  image(                newImg,    10, 10);
}

void keyPressed()
{  if (key == ' ')
{  save("pict.png");
} // Finish the movie if space bar is pressed!
}
```

==========Extract_Display_Pixels_to_Transparent_file==========



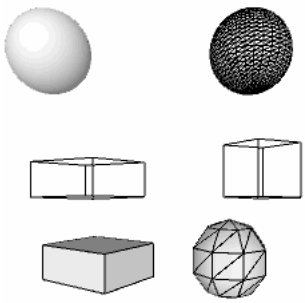

```
String FILENAME = "/Users/don_sauer/Downloads/W2.png";
PImage img;
PImage newImg;
PShape line, square ;
int x;
int y;
int i;
```

```
  size(200, 200,P2D);
  colorMode(RGB, 256);
  stroke(0, 0,0);
  strokeWeight(5);
  line                  = createShape(LINE,   0, 0,6,15 );
  square                = createShape(RECT,   0, 0, 50, 50);
  shape(                  line,      30, 15);
  shape(                  square,    50, 50);
  loadPixels();                                        // load display
  newImg                = createImage( 200, 200, ARGB ) ; //
  for                     ( x = 0; x < 200; x++ )
{ for                     ( y = 0; y < 200; y++ )
{ i                   = ( ( y * 200 ) + x );
  if                      (  red(pixels[i]) < 5 )  { newImg.pixels[i] = color( 256, 0, 0, 256 ); }
  else                                             { newImg.pixels[i] = color( 0,   0, 0, 0 ); }
}
}
  newImg.save(            FILENAME );     // x1 y1 x2 y2
```

==========MouseSnake==========

```
int[] xpos              = new int[50];                          // Declare two arrays with 50 elements.
int[] ypos              = new int[50];
void                      setup()
{ size(                   200,200);
  smooth(                 );
  for                     (int i = 0; i < xpos.length; i ++ )   // Initialize all elements of each array to zero.
{ xpos[i]               = 0;
  ypos[i]               = 0;
}}
void                      draw()
{ background(           255);
  for                     (int i = 0; i < xpos.length-1; i ++ )  // Shift array values
{ xpos[i]               = xpos[i+1];                            // Shift all elements down one spot.
  ypos[i]               = ypos[i+1];                            // Stop at second to last element.
}
  xpos[xpos.length-1]   = mouseX;                               // Update last spot in array with mouse location.
  ypos[ypos.length-1]   = mouseY;                               // New location
  for                     (int i = 0; i < xpos.length; i ++ )   // Draw everything
{ noStroke(               );
  fill(                   255-i*5);                             // Color and size are tied to the loop's counter: i.
  ellipse(                xpos[i],ypos[i],i,i);                 // Draw an ellipse for each element in the arrays.
}
}
void                      keyPressed()
{ save(                   "pict.jpg");
}
```

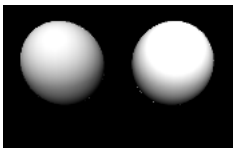
==========3D_Planar==========



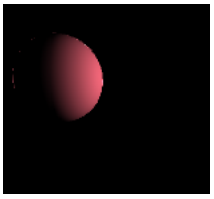
```
size(         300, 300, P3D);
background(   255);
noStroke(     );
lights(       );// default ambient ambientLight(128, 128, 128)and directionalLight(128, 128, 128, 0, 0, -1)
translate(    50, 50, 0);
sphere(       30);
translate(    150, 0, 0);
stroke(       0);
sphere(       30);
translate(    8, 88, 0);
rotateY(      0.5);
noFill(       );
box(          40);
translate(   -128, 8, 0);
rotateY(      0.5);
noFill(       );
box(          40, 20, 50);
```

```
translate(    8, 48, 0);
fill(         255);
box(          40, 20, 50);
translate(    180, 30, 0);
sphereDetail( 8, 5 ); //numbLongseg, numblatseg
sphere(       40);
```

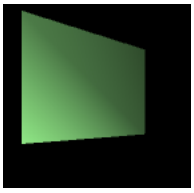
```
size(              200, 200, P3D);
background(        0);
noStroke(          );
directionalLight(  36, 156,36, 0, 0, -1); //
ambientLight(      102, 32, 102); // r,g,b
rotateY(          -PI/8);
translate(         32, 50, -50);
box(               40);
translate(         80, 0, 0);
sphere(            30);
```



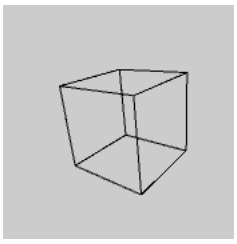
```
size(                 200, 200, P3D);
background(           0);
noStroke(             );
directionalLight(     102, 102, 102, 0, 0, -1);  //r,g,b, xdir,ydir,zdir
lightSpecular(        204, 204, 204);            //r,g,b
directionalLight(     102, 102, 102, 0, 1, -1);
lightSpecular(        102, 102, 102);
translate(            50, 50, 0);
specular(             51, 51, 51);
sphere(               30);//
translate(            80, 0, 0);
specular(             102, 102, 102);
sphere(               30);
```



```
void                  setup()
{ size(               200, 200, P3D);
}
void                  draw()
{ background(         0);
  noStroke();
  spotLight(          251, 102, 126, 80+mouseX, 20+ mouseY, 40, -1, 0, 0, PI/2, 2);
  translate(          50, 60, 10);
  sphere(             30);
  println(            mouseX +"  "+  mouseY);
}
```

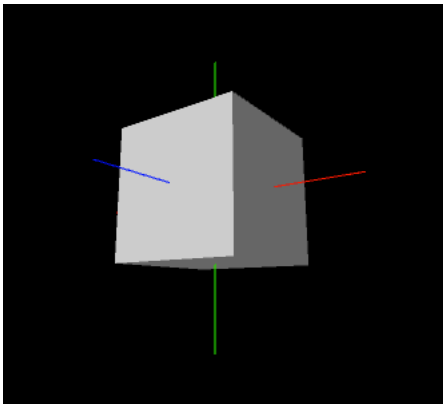


```
void                  setup()
{ size(               200, 200, P3D);
}
void                  draw()
{ noStroke(           );
  background(         0);
  lightFalloff(       1.0*mouseX/100, 0.001*mouseY/100, 0.0);  //lightFalloff(constant, linear, quadratic)
  pointLight(         150, 250, 150, 50 +29, 50 + 92, 50);     //r,g,b , xloc,yloc,zloc, xdir,ydir,zdie, concentrtion
  translate(          20, 20, 0);
  beginShape(         );
  vertex(             0, 0, 0);
  vertex(             100, 0, -100);
  vertex(             100, 100, -100);
  vertex(             0, 100, 0);
  endShape(           CLOSE);
  println(            mouseX +"  "+  mouseY);
}
```



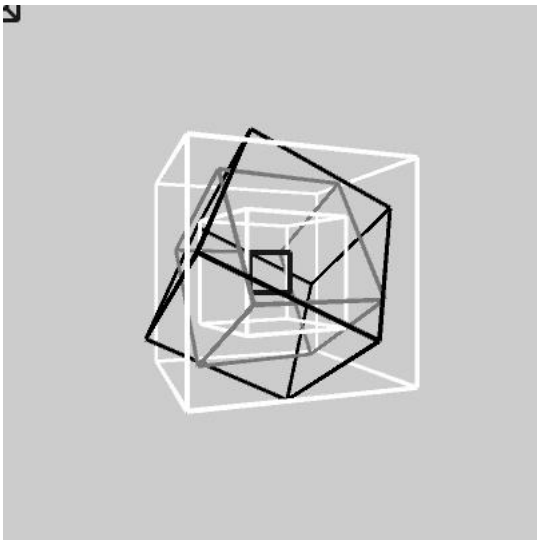
```
void                    setup()
{ size(                 200, 200, P3D);
}
void                    draw()
{ noFill(               );
  background(           204);
  camera(               70.0 +mouseX,35.0+mouseY,120.0,50.0,50.0,0.0,0.0,1.0,0.0);//camera(eyeX,eyeY,eyeZ,ordX,ordY,ordZ,upX,upY,upZ)
  translate(            50, 50, 0);
  rotateX(              -PI/6);
  rotateY(              PI/3);
  box(                  45);
  println(              mouseX +"  "+  mouseY);
}
```

==========camera-controlled-by-mouse==========



```
void                    setup()
{ size(                 640, 360, P3D);
  fill(                 204);
}
void                    draw()
{ lights();
  background(           0);
  camera(               mouseX, mouseY, 220.0,      // eyeX, eyeY, eyeZ
                        0.0, 0.0, 0.0,              // centerX, centerY, centerZ
                        0.0, 1.0, 0.0);             // upX, upY, upZ
  noStroke(             );
  box(                  90);
  stroke(               255,0,0);
  line(                -100, 0, 0, 100, 0, 0);      // +/- X
  stroke(               0,255,0);
  line(                 0, -100, 0, 0, 100, 0);      //  +/- y
  stroke(               0,0,255);
  line(                 0, 0, -100, 0, 0, 100);      // +/- Z
  println(              mouseX +"  "+  mouseY);

}


void                    keyPressed() { save("pict.jpg")}
```

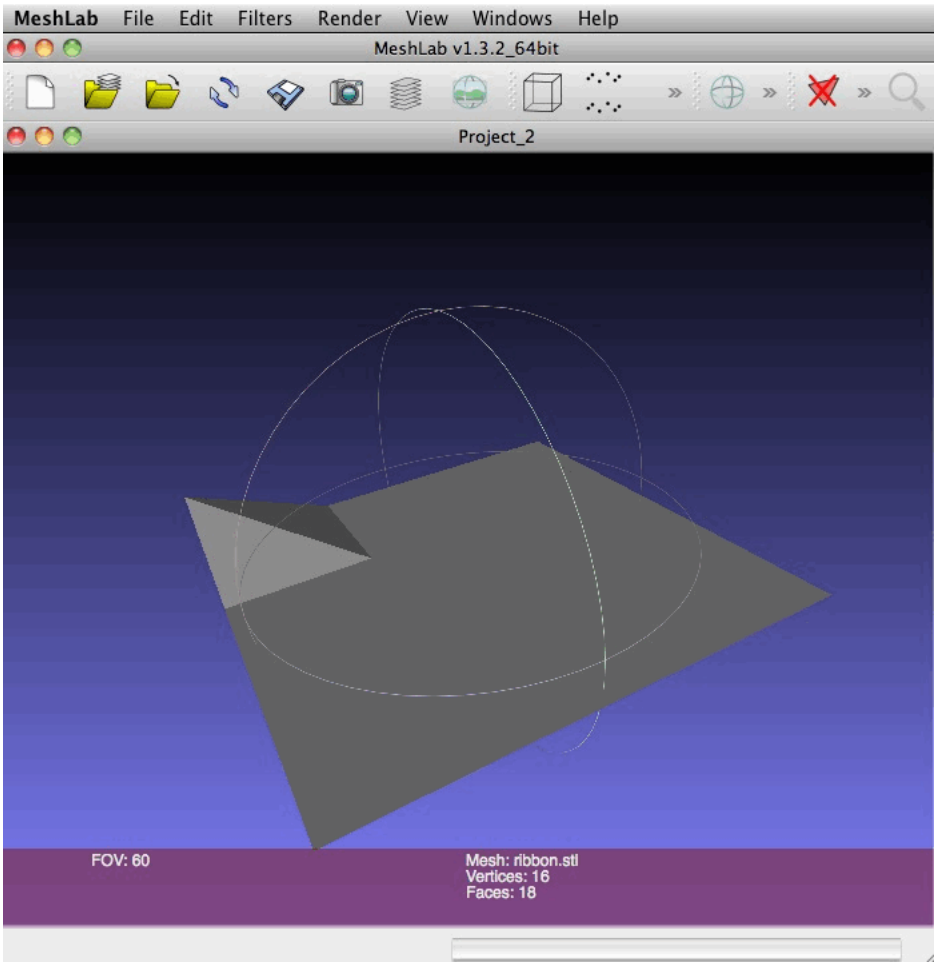
==========applyMatrix()==========

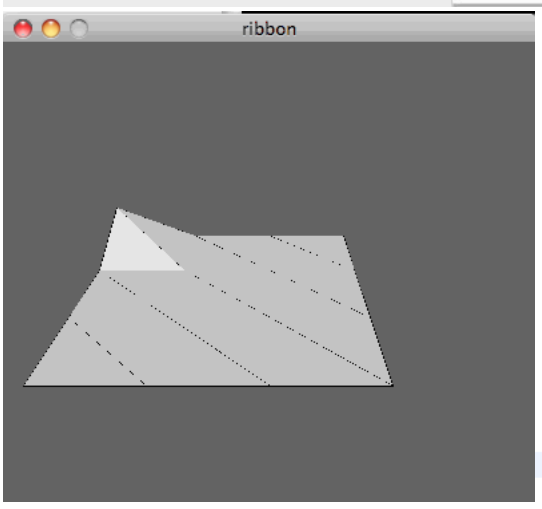


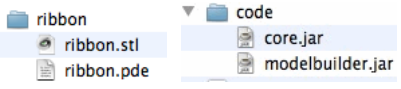
```
size(                   400, 400, P3D);
pushMatrix();
println(                "Start maxtrix");
printMatrix();
// Start maxtrix
// 001.0000  000.0000  000.0000 -200.0000
// 000.0000  001.0000  000.0000 -200.0000
// 000.0000  000.0000  001.0000 -346.4102
// 000.0000  000.0000  000.0000  001.0000
noFill();
stroke(                 25);
strokeWeight(           5);
box(                    15);
//=======================================
translate(              200, 200, 0);
println(                "translate maxtrix");
printMatrix();
// translate maxtrix
// 001.0000  000.0000  000.0000  000.0000
// 000.0000  001.0000  000.0000  000.0000
// 000.0000  000.0000  001.0000 -346.4102
// 000.0000  000.0000  000.0000  001.0000
box(                    30);
//=======================================
rotateY(PI/6);
println(                "rotateY maxtrix");
printMatrix();
// rotateY maxtrix
// 000.8660  000.0000  000.5000  000.0000
// 000.0000  001.0000  000.0000  000.0000
//-000.5000  000.0000  000.8660 -346.4102
// 000.0000  000.0000  000.0000  001.0000
stroke(                 255);
box(                    80);
//=======================================
rotateX(                PI/6);
printMatrix();
println(                "rotateX maxtrix");
// rotateX maxtrix
// 000.8660  000.2500  000.4330  000.0000
// 000.0000  000.8660 -000.5000  000.0000
//-000.5000  000.4330  000.7500 -346.4102
// 000.0000  000.0000  000.0000  001.0000
stroke(                 120);
box(                    100);
//=======================================
rotateZ(                PI/6);
println(                "rotateZ maxtrix");
printMatrix();
// rotateZ maxtrix
// 000.8750 -000.2165  000.4330  000.0000
// 000.4330  000.7500 -000.5000  000.0000
//-000.2165  000.6250  000.7500 -346.4102
// 000.0000  000.0000  000.0000  001.0000
stroke(                  0);
box(                    120);
//=======================================
popMatrix();
float ct              = cos(PI/9.0); // Set rotation angles
float st              = sin(PI/9.0);
float tx              = 200;         // Set Translate
float ty              = 200;
applyMatrix(  ct, 0.0,  st,  tx,
              0.0, 1.0, 0.0,  ty,
              -st, 0.0,  ct,  0.0,
              0.0, 0.0, 0.0,  1.0);
println(                "applied maxtrix");
printMatrix();
// applied maxtrix
// 000.9397  000.0000  000.3420  000.0000
// 000.0000  001.0000  000.0000  000.0000
//-000.3420  000.0000  000.9397 -346.4102
```

```
// 000.0000  000.0000  000.0000  001.0000
stroke(                 255);
box(                    150);
//=======================================
save(                   "pict.jpg");
```

==========CREATE_STL==========



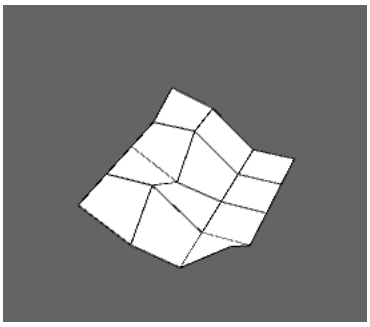





```
import unlekker.util.*;
import unlekker.modelbuilder.*;                 // need 1.50

MouseNav3D  nav;
UGeometry   model;
int[][]     myArray = {  {4, 3, 3, 3},
                         {3, 3, 3, 3},
                         {3, 3, 3, 3},
```

```
                        {3, 3, 3, 3}  };
void                      setup()
{ size(                   400,400, P3D);
  nav                    = new MouseNav3D(this);      // add MouseNav3D navigation
  nav.trans.set(          width/2,height/2,0);
  buildModel();
  nav.rot.x              = 1;
  println(                nav.rot);
  nav.trans.x            = 100;
  println(                nav.trans);
}
void                      draw()
{ background(             100);
  lights();
  nav.doTransforms();                   // call MouseNav3D transforms
  fill(                   255);
  model.draw(             this);
}

public void               mouseDragged()  { nav.mouseDragged(); }

public void               keyPressed()
{ nav.keyPressed();
  if                      (key=='s')
{ model.writeSTL(         this, "ribbon.stl");
}
}
void                      buildModel()
{ float                   x,y,z,x2, z2;
    //  1--3--5--7        QUAD_STRIP
    //  |  |  |  |
    //  0--2--4--6
    //  u[0][0] u[1][0] u[2][0]   //  0--2--4--6
    //  u[0][1] u[1][1] u[2][1]   //  1--3--5--7

  model                  = new UGeometry();
  for                     (int j=0; j<3; j++)
{ model.beginShape(       QUAD_STRIP);
  for                    (int i=0; i<4; i++)
{ model.vertex(           50*i,       50*j   ,        20* myArray[j][i]);    //myArray[vert][horz]
  model.vertex(           50*i   ,    50*j+50,        20* myArray[j+1][i]);
} model.endShape();
}
}
```

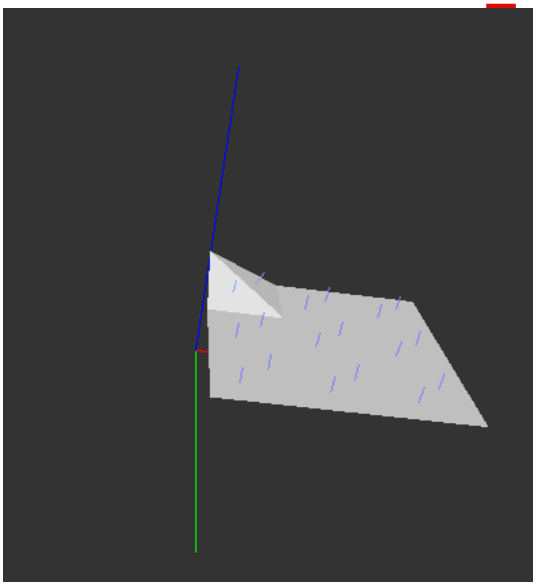
==========MOUSE_VIEW_3D_Plane==========

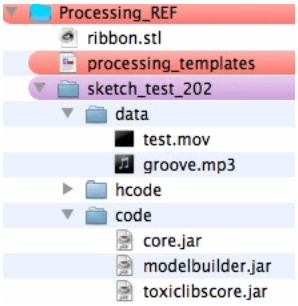


```
float cx                = 10;
float cy                = 10;
float w                 = 30;
float h                 = 30;
int[][] zd              = {  {40, 30, 3, 3},
                             {30, 30, 3, 3},
                             {30, 10, 3, 3},
                             {30, 30, 3, 3},
                             {30, 10, 3, 30}  };
void                      setup()
{ size(                   500, 500, P3D);
  println (               "zd.length = "+  zd.length+"  zd[0].length = "+  zd[0].length);   //  zd.length = 5  zd[0].length = 4
}
void                      draw()
{ background(             100);
  pushMatrix();
  translate(              200, 200, 0);
  rotateX(                PI-mouseY/150.0);
  rotateZ(                PI/2-mouseX/150.0);
  for                     ( int i=0; i <zd[0].length-1; i++ )
{ for                     ( int j=0; j <zd.length-1; j++ )      // x step
{ drawPlate(              cx+i*w,cy+j*h,zd[j][i],cx+w*(i+1),cy+j*h,zd[j][i+1],cx+w*(i+1),cy+h*(j+1),zd[j+1][i+1],cx+i*w,cy+h*(j+1),zd[j+
1][i]);
}
}
  popMatrix();
}
void drawPlate(     float x1,float y1,float z1,float x2,float y2,float z2,float x3,float y3,float z3,float x4,float y4,float z4 )
{ stroke(               0);
  strokeWeight(         1);
  beginShape(           QUADS);
```

```
  vertex(             x1, y1, z1);
  vertex(             x2, y2, z2);
  vertex(             x3, y3, z3);
  vertex(             x4, y4, z4);
  endShape();
}
```

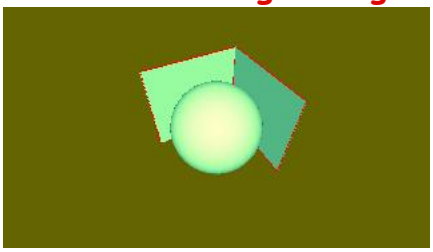
========================READ_STL==






```
import toxi.geom.*;
import toxi.geom.mesh.*;      // need 1.50

import toxi.processing.*;

TriangleMesh     mesh;
ToxiclibsSupport gfx;

void setup()
{ size(600,600,P3D);
  mesh                = (TriangleMesh)new STLReader().loadBinary(sketchPath("ribbon.stl"),STLReader.TRIANGLEMESH);
  //mesh=(TriangleMesh)new STLReader().loadBinary(sketchPath("mesh-flipped.stl"),STLReader.TRIANGLEMESH).flipYAxis();
  gfx                 = new ToxiclibsSupport(this);
}

void draw() {
  background(51);
  lights();
  translate(width/2,height/2,0);
  rotateX(mouseY*0.01);
  rotateY(mouseX*0.01);
  gfx.origin(new Vec3D(),200);    //origin(toxi.geom.ReadonlyVec3D o, float len)  Draws the major axes from the given point.
  noStroke();
  gfx.mesh(mesh,false,10);        //mesh(toxi.geom.mesh.Mesh3D mesh, boolean smooth)  Draws a mesh instance.
}


print(sketchPath("ribbon.stl"));   /Users/don_sauer/Downloads/moreProcessing/Processing_REF/ribbon.stl
```

==============Lighting-to-3D==================================

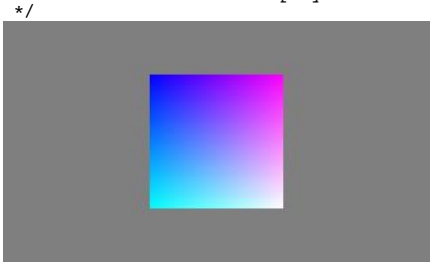


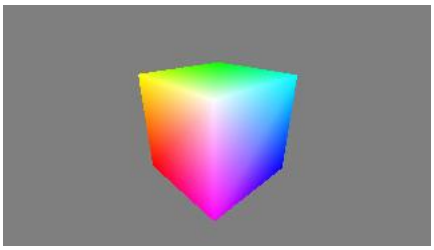
```
float xmag, ymag = 0;       //rotate angles
float newXmag, newYmag = 0; //next rotate angle

void setup()
{ size(320, 180, P3D);
  colorMode(RGB, 255);
}
void draw()
{ background(100,100,0);
  // lights();
  directionalLight(255,255,100, 0,0, -1);
```

```
  //spotLight(100, 100, 100, 8, 2, 4, -1, 0, -1, PI/2, 2);
  ambientLight(0, 100, 100);
  pushMatrix();
  translate(width/2, height/2, -30);         //center
  newXmag = mouseX/float(width) * TWO_PI;    // new rotate
  newYmag = mouseY/float(height) * TWO_PI;    // new rotate
  float diff = xmag-newXmag;
  if (abs(diff) >  0.01) { xmag -= diff/4.0; }
  diff = ymag-newYmag;
  if (abs(diff) >  0.01) { ymag -= diff/4.0; }
  rotateX(-ymag);                            // cube follow mouse
  rotateY(-xmag);                            // cube follow mouse
  scale(45);

  noStroke();
  sphere(.9);
  stroke(255,0,0);

  beginShape(QUADS);
  vertex(-1,  1,  1);
  vertex( 1,  1,  1);
  vertex( 1, -1,  1);
  vertex(-1, -1,  1);

  vertex( 1,  1,  1);
  vertex( 1,  1, -1);
  vertex( 1, -1, -1);
  vertex( 1, -1,  1);
  endShape();
  popMatrix();
}
void keyPressed() { save("pict.jpg"); }
```

==========RGB-Cube==========

```
/**
 * RGB Cube.
 *
 * The three primary colors of the additive color model are red, green, and blue.
 * This RGB color cube displays smooth transitions between these colors.
 */
```

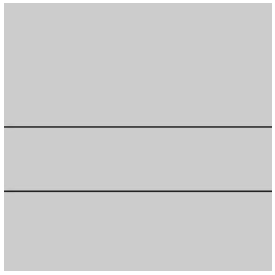




```
float xmag, ymag = 0;       //rotate angles
float newXmag, newYmag = 0; //next rotate angle

void setup()
{ size(320, 180, P3D);
  noStroke();
  colorMode(RGB, 1);
}
void draw()
{ background(0.5);
  pushMatrix();
  translate(width/2, height/2, -30);         //center
  newXmag = mouseX/float(width) * TWO_PI;    // new rotate
  newYmag = mouseY/float(height) * TWO_PI;    // new rotate
  float diff = xmag-newXmag;
  if (abs(diff) >  0.01) { xmag -= diff/4.0; }
  diff = ymag-newYmag;
  if (abs(diff) >  0.01) { ymag -= diff/4.0; }
  rotateX(-ymag);                            // cube follow mouse
  rotateY(-xmag);                            // cube follow mouse
  scale(45);

  beginShape(QUADS);
  fill(0, 1, 1); vertex(-1,  1,  1);         //cyan
  fill(1, 1, 1); vertex( 1,  1,  1);         //white
  fill(1, 0, 1); vertex( 1, -1,  1);         //magne
  fill(0, 0, 1); vertex(-1, -1,  1);         //blue

  fill(1, 1, 1); vertex( 1,  1,  1);
  fill(1, 1, 0); vertex( 1,  1, -1);
  fill(1, 0, 0); vertex( 1, -1, -1);
  fill(1, 0, 1); vertex( 1, -1,  1);

  fill(1, 1, 0); vertex( 1,  1, -1);
  fill(0, 1, 0); vertex(-1,  1, -1);
  fill(0, 0, 0); vertex(-1, -1, -1);
  fill(1, 0, 0); vertex( 1, -1, -1);

  fill(0, 1, 0); vertex(-1,  1, -1);
  fill(0, 1, 1); vertex(-1,  1,  1);
  fill(0, 0, 1); vertex(-1, -1,  1);
  fill(0, 0, 0); vertex(-1, -1, -1);

  fill(0, 1, 0); vertex(-1,  1, -1);
```

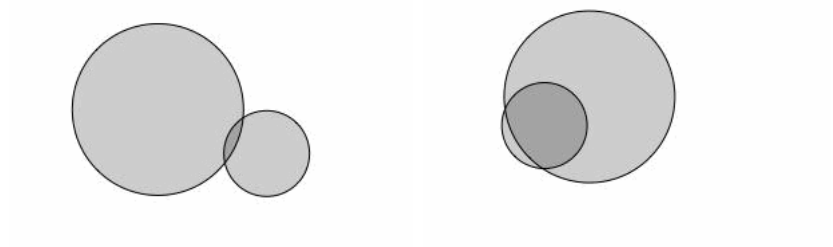
```
  fill(1, 1, 0); vertex( 1,  1, -1);
  fill(1, 1, 1); vertex( 1,  1,  1);
  fill(0, 1, 1); vertex(-1,  1,  1);

  fill(0, 0, 0); vertex(-1, -1, -1);
  fill(1, 0, 0); vertex( 1, -1, -1);
  fill(1, 0, 1); vertex( 1, -1,  1);
  fill(0, 0, 1); vertex(-1, -1,  1);

  endShape();
  popMatrix();
}


void mousePressed() { save("pict.jpg"); }
```

==========CLASS==========



```
HLine h1                = new HLine(20, 2.0);       // Declare and construct two objects (h1, h2) from class HLine
HLine h2                = new HLine(50, 2.5);
void                      setup()
{ size(                   200, 200);
  frameRate(              30);
  noLoop();                                          // ################################
}
void                      draw()
{ background(             204);
  h1.update();
  h2.update();
}
class                     HLine
{ float                   ypos, speed;
  HLine                   (float y, float s)
{ ypos                  = y;
  speed                 = s;
}
  void                    update()
{ ypos                 += speed;
  if                      (ypos > width)  { ypos = 0; }
  line(                   0, ypos, width, ypos);
}
}
void                      mousePressed()
{ loop();                                            // Holding down the mouse activates looping ##########
  println(               "frameRate = "  + frameRate);  // 4.099067 an average value starting at 10 #########
  println(               "frameCount = " + frameCount); // 37 ##############
}
void                      mouseReleased()
{ noLoop();                                          // Releasing the mouse stops looping draw() ############
  save(                   "pict.jpg");
}
```

==========ClassBouncingball==========

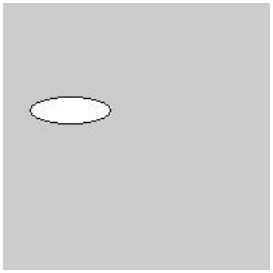


```
Ball ball1;               // Two ball variables
Ball ball2;
void                      setup()
{ size(                   400,400);
  smooth(                 );
  ball1                 = new Ball(64); // Initialize balls
  ball2                 = new Ball(32);
}
void                      draw()
{ background(             255);
  ball1.move(             );            // Move and display balls
  ball2.move(             );
```

```
  ball1.display(           );
  ball2.display(           );
}
class                      Ball
{ float r;                 // radius
  float x,y;               // location
  float xspeed,yspeed;     // speed
Ball(                      float tempR)   // Constructor
{ r                       = tempR;
  x                       = random(width);
  y                       = random(height);
  xspeed                  = random( - 5,5);
  yspeed                  = random( - 5,5);
}
void                       move()
{ x                      += xspeed; // Increment x
  y                      += yspeed; // Increment y
  if                        (x > width || x < 0)  { xspeed *= - 1;}            // Check horizontal edges
  if                        (y > height || y < 0) { yspeed *= - 1;} //Check vertical edges
}
void                       display() // Draw the ball
{ stroke(                  0);
  fill(                    0,50);
  ellipse(                 x,y,r*2,r*2);
}
}
void                       mousePressed()
{ save(                    "pict.jpg");
}
```

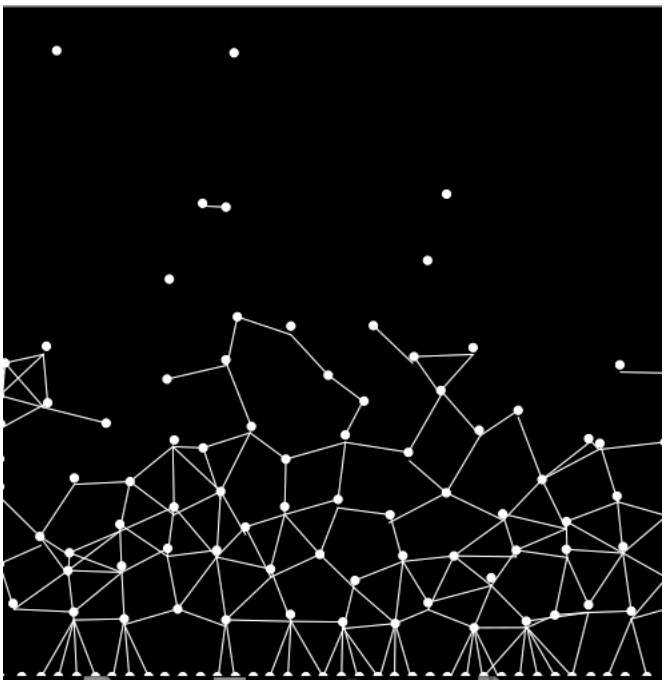
==========CLASS_extends==========

Allows a new class to inherit the methods and fields (data members) from an existing class.



```
DrawDot dd1             = new DrawDot(50, 80);
void                      setup()
{ size                   (200, 200);

}
void                      draw()
{ dd1.display();
}
class                     Dot
{ int                     xpos, ypos;
}
class                      DrawDot extends Dot
{ DrawDot(               int x, int y)
{ xpos                  = x;
  ypos                  = y;
}
void                      display()
{ ellipse                 (xpos, ypos, 60, 20);
}
}
void mousePressed()
{ save                   ("pict.jpg"); }
```

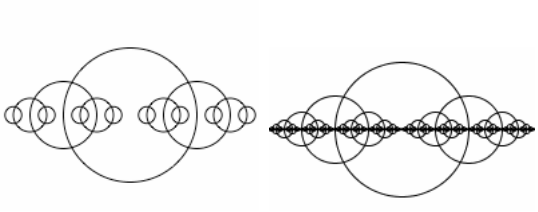
==========CLASS_METHODS==========



```
int     numPoints          = 150;
float   g                  = 5;
float   spring             = 0.01;
float   resistance         = 0.02;
Point[] group1             = new Point[numPoints];
void                        setup()
{ size(                     500, 500);
  stroke(                   255);
  smooth();
  for                       (int i = 0; i < numPoints; i++)
{ group1[i]                = new Point(random(width), random(height),  i, group1);
}
}
void                        draw()
{ background(               0);
  for                       (int i = 0; i < numPoints; i++)
{ group1[                   i].move();
  group1[                   i].display();
  group1[                   i].collide();
  group1[                   i].gravity();
  if                        (mousePressed)
{ group1[i].x              = random(width);
  group1[i].y              = random(height);
}
}
}
class                       Point
{ float   x, y;
  float   vx               = random(-1, 1);
  float   vy               = random(-1, 1);
  int     id;
  Point[] others;                                  //#######   has access to its own array
Point(                      float xin, float yin, int idin, Point[] oin)
{ x                        = xin;
  y                        = yin;
  id                       = idin;
  others                   = oin;
}
void                        collide()
{ for                       (int i = id + 1; i < numPoints; i++)
{ float dx                 = others[i].x - x;                    // find distance dx to another object for all others
  float dy                 = others[i].y - y;                    // find distance dy to another object
  float distance           = sqrt(dx*dx + dy*dy);                // find distance    to another object
  float minDist            = 50;
  if                        (distance < minDist)                 // if within 50 of another object
{ float angle              = atan2(dy, dx);                      // find angle of approuch
  float targetX            = x + cos(angle) * minDist;           // ax =( x - others[i].x +  cos(angle) * minDist ) * spring
  float targetY            = y + sin(angle) * minDist;
  float ax                 = (targetX - others[i].x) * spring;   // create force based on spring collishion
  float ay                 = (targetY - others[i].y) * spring;
  vx                      -= ax;                                 // adjust velocity this object
  vy                      -= ay;
  others[i].vx            += ax;                                 // change other object as well
  others[i].vy            += ay;
  line(                       x, y, others[i].x, others[i].y);   // line shows force between objects
}}}
void                        move()
{ x                       += vx;
  y                       += vy;
  if                          (vx > 0)     { vx -= resistance; } else { vx += resistance; } // slow down velocity vx
  if                          (vy > 0)     { vy -= resistance; } else { vy += resistance; } // slow down velocity vy
  if                          (x > width)  { x = width;   vx *= -1;   }                     // bounce of wall
```

```
  else if                 (x < 0)       { x = 0; vx *= -1; }                          // bounce of wall
  if                      (y > height) { y = height; vy *= -1; }                      // bottom of display is height
}
void                      display()  { ellipse(x, y, 6, 6);  }                      // draw each ball
void                      gravity() {   y += g;  }                                  // add gravity to each object
}
```

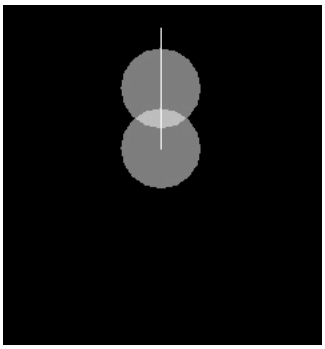
==========NEST_LEVELS==========



```
float off1, off2;
int level      = 0;

void                      setup()
{ size(                   200,200);
  smooth();
}
void                      draw()
{ background(             255);
  stroke(0);
  noFill();
  drawCircle(             width/2,height/2,100);
  noLoop();
}
void                      drawCircle(float x, float y, float radius)
{ off1                  = x - 100;
  println(                "draw  " + off1 +" "  + radius );
  ellipse(                x, y, radius, radius);
  if                      (radius > 20)
{ off2                  = x - 100 + radius/2;
  level                 = level+1;
  println(                "+"+level +"    "  + off2 +" "  + radius/2  +" "  + radius);
  drawCircle(             x + radius/2, y, radius/2);   // drawCircle() calls itself twice, a branching effect.
  off2                  = x - 100 - radius/2;
  println(                "-"+level +"    "  + off2 +" "  + radius/2  +" "  + radius);
  drawCircle(             x - radius/2, y, radius/2);   // For every circle, smaller to   left and right.
  level                 = level-1;
}
}
```

```
level  x     r     r/2
draw  0.0   100.0
+1    50.0  50.0 100.0
draw  50.0  50.0
+2    75.0  25.0 50.0
draw  75.0  25.0
+3    87.5  12.5 25.0
draw  87.5  12.5
-3    62.5  12.5 25.0
draw  62.5  12.5
-2    25.0  25.0 50.0
draw  25.0  25.0
+3    37.5  12.5 25.0
draw  37.5  12.5
-3    12.5  12.5 25.0
draw  12.5  12.5
-1    -50.0 50.0 100.0
draw  -50.0 50.0
+2    -25.0 25.0 50.0
draw  -25.0 25.0
+3    -12.5 12.5 25.0
draw  -12.5 12.5
-3    -37.5 12.5 25.0
draw  -37.5 12.5
-2    -75.0 25.0 50.0
draw  -75.0 25.0
+3    -62.5 12.5 25.0
draw  -62.5 12.5
-3    -87.5 12.5 25.0
draw  -87.5 12.5
```

==========Chain==========



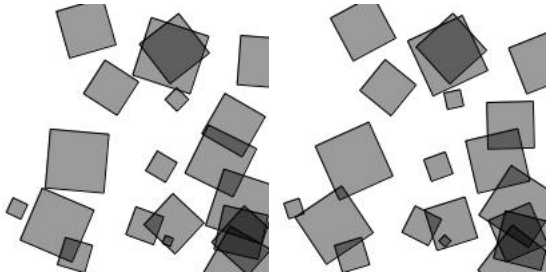
```
Spring2D s1, s2;
float gravity = 9.0;
float mass = 2.0;

void setup()
{ size(640, 360);
  fill(255, 126);
  s1 = new Spring2D(0.0, width/2, mass, gravity);// Inputs: x, y, mass, gravity
  s2 = new Spring2D(0.0, width/2, mass, gravity);
}
void draw()
{ background(0);
  s1.update(mouseX, mouseY);
  s1.display(mouseX, mouseY);
  s2.update(s1.x, s1.y);
  s2.display(s1.x, s1.y);
}
class Spring2D
{ float vx, vy;  // The x- and y-axis velocities
  float x, y;    // The x- and y-coordinates
  float gravity;
  float mass;
  float radius = 30;
  float stiffness = 0.2;
  float damping = 0.7;
Spring2D(float xpos, float ypos, float m, float g)
{ x = xpos;
  y = ypos;
  mass = m;
  gravity = g;
}
void update(float targetX, float targetY)
{ float forceX = (targetX - x) * stiffness;
  float ax = forceX / mass;
  vx = damping * (vx + ax);
  x += vx;
  float forceY = (targetY - y) * stiffness;
  forceY += gravity;
  float ay = forceY / mass;
  vy = damping * (vy + ay);
  y += vy;
}
void display(float nx, float ny)
{ noStroke();
  ellipse(x, y, radius*2, radius*2);
  stroke(255);
  line(x, y, nx, ny);
}
}
```

==========SimpleSolarSystem==========

```
// Angle of rotation around sun and planets
float theta = 0;
void setup()
{ size(200,200);
  smooth();
}
void draw()
{ background(255);
  stroke(0);
  translate(width/2,height/2);   // Translate to center of window to draw the sun.
  fill(255,200,50);
```

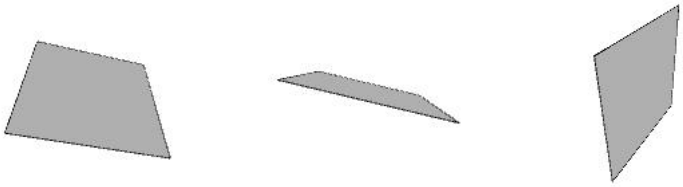
```
  ellipse(0,0,20,20);
  pushMatrix();                 // The earth rotates around the sun
  rotate(theta);
  translate(50,0);
  fill(0,0,255);
  ellipse(0,0,10,10);
  pushMatrix();                 // Moon #1 rotates around the earth
  rotate(-theta*4);             // pushMatrix() is called to save the transformation state before drawing moon #1.
  translate(15,0);              // This way we can pop and return to earth before drawing moon #2.
  fill(5,250,5);                // Both moons rotate around the earth (which itself is rotating around the sun).
  ellipse(0,0,6,6);
  popMatrix();
  pushMatrix();                 // Moon #2 also rotates around the earth
  rotate(theta*2);
  translate(25,0);
  fill(250,05,2);
  ellipse(0,0,6,6);
  popMatrix();
  popMatrix();
  theta += 0.01;
}
void                          mousePressed()
{ save(                      "pict.jpg");
}
```

======================RotatingManyObjects==================================



```
Rotater[]               rotaters;       // An array of Rotater objects
void                    setup()
{ size(                 200,200);
  rotaters              = new Rotater[20];
  for                   (int i = 0; i < rotaters.length; i++ )        // Rotaters are made randomly
{ rotaters[i]           = new Rotater(random(width),random(height),random(-0.1,0.1),random(48));
}
}
void                    draw()
{ background(           255);
  for                   (int i = 0; i < rotaters.length; i++ ) // All Rotaters spin and are displayed
{ rotaters[i].spin();
  rotaters[i].display();
}
}
class                   Rotater
{ float x,y;            // x,y location
  float w;              // size of rectangle
  float theta;          // angle of rotation
  float speed;          // speed of rotation
Rotater(                float tempX, float tempY, float tempSpeed, float tempW)
{ x                     = tempX;
  y                     = tempY;
  theta                 = 0;                  // Angle is always initialized to 0
  speed                 = tempSpeed;
  w                     = tempW;
}
void                    spin()                // Increment angle
{ theta                += speed;
}
void                    display()             // Display rectangle
{ rectMode(             CENTER);
  stroke(               0);
  fill(                 0,100);
  pushMatrix(           );              // pushMatrix() and popMatrix() are called inside the class' display() method.
  translate(            x,y);            // This way, every Rotater object is rendered with its own independent translation and
rotation!
  rotate(               theta);
  rect(                 0,0,w,w);
  popMatrix(            );
}
}
void                    mousePressed()
{ save(                 "pict.jpg");
}
```

================RotateXaxisYaxis=======================================



```
void                    setup()
{ size(                 200,200,P3D);
}
void                    draw()
{ background(           255);
  stroke(               0);
  fill(                 175);
  translate(            width/2,height/2);
  rotateX(              PI*mouseY/height);
  rotateY(              PI*mouseX/width);
  rectMode(             CENTER);
  rect(                 0,0,100,100);
}
void                    mousePressed()
{ save(                "pict.jpg");
}
================================================
```

==========VIEWHSB==========


```
PFont                   f;
PImage                  img;
int                     r,g,b, hue1, sat,brite;
color                   c;
int                     i = 0 ;
String                  str1,str2;
void                    setup()
{ img                 = loadImage("/Users/don_sauer/Desktop/Flower.jpg");
  size(                 img.width, img.height);
  colorMode(            RGB, 100);
  strokeWeight(         5);
  fill(                 255, 0, 0);
  image(                img, 0, 0);
  f                   = createFont("Arial",9,true);
}
void                    draw()
{ if                    (mousePressed )
{ c                   = get( mouseX,mouseY);
  r                   = int(red(c));
  g                   = int(green(c));
  b                   = int(blue(c));
  hue1                = int(hue(c) );
  sat                 = int(saturation(c));
  brite               = int(brightness(c) );
  println(              i+") "+mouseX+"  "+ mouseY+" rgb=  "+ r+" "+g+" "+b +" hsb= "+hue1 +" "+sat +" "+brite );
  i                   = i+1;
  stroke(               100-r,100-g,100-b);
  point(                mouseX, mouseY);
  str1                = String.valueOf(hue1) +","+String.valueOf(sat) +","+String.valueOf(brite);
  fill(                 100-r,100-g,100-b);
  text(                 str1, mouseX, mouseY);
  delay (               150);
}}
void                    mousePressed()
{ save(                "pict.jpg");
}
```

```
================================================
0) 310   239 rgb=  1   29 65 hsb= 59 97  65
1) 400   135 rgb=  1   23 61 hsb= 60 97  61
2) 298   159 rgb=  77  59 0  hsb= 12 100 77
```

```
3) 247   196 rgb=  100 98 4  hsb= 16 95  100
4) 196   209 rgb=  98  90 0  hsb= 15 100 98
5) 182   86  rgb=  58  36 0  hsb= 10 100 58
6) 142   89  rgb=  60  41 1  hsb= 11 98  60
7) 113   116 rgb=  77  50 2  hsb= 10 96  77
8) 134   191 rgb=  100 87 3  hsb= 14 96  100
9) 161   222 rgb=  79  62 0  hsb= 13 100 79
10) 272  137 rgb=  99  87 4  hsb= 14 95  99
11) 353  186 rgb=  1   27 64 hsb= 59 96  64
12) 333  186 rgb=  0   28 64 hsb= 59 99  64
13) 315  104 rgb=  0   22 57 hsb= 60 100 57
14) 287  94  rgb=  1   23 61 hsb= 60 97  61
15) 241  56  rgb=  0   20 56 hsb= 60 99  56
16) 192  271 rgb=  0   7  0  hsb= 33 100 7
17) 185  250 rgb=  2   14 0  hsb= 31 94  14
18) 221  283 rgb=  0   31 63 hsb= 58 100 63
19) 199  283 rgb=  3   12 3  hsb= 34 75  12
```

==========TableMaker==========

```
 String[]                  words ;
 String                    str1 = "";
 int[] maxlen             = new int[30];

  for                      (int k = 0 ; k < 30; k++) { maxlen[k]=0; }
  String lines[]           = loadStrings("/Users/don_sauer/Downloads/moreProcessing/Processing_REF/sketch_test_202/tableRaw.txt");

  for                      (int i = 0 ; i < lines.length; i++)
{ println(                 lines[i]);
  words                    = split(lines[i], ' ');
  for                      (int j = 0 ; j < words.length; j++)
{ str1                     = words[j];
  if                       ( str1.length() > maxlen[j]) {  maxlen[j] = str1.length();   }
}
}
 println("================================================ ");
  for                      (int i = 0 ; i < lines.length; i++)
{ words                    = split(lines[i], ' ');
  for                      (int j = 0 ; j < words.length; j++)
{ while                    ( words[j].length() < maxlen[j] ) {  words[j] =  words[j] + " ";    }
  print(                    words[j]+"  ");
} println(                 " ");
}
```

==========tableRaw.txt==========

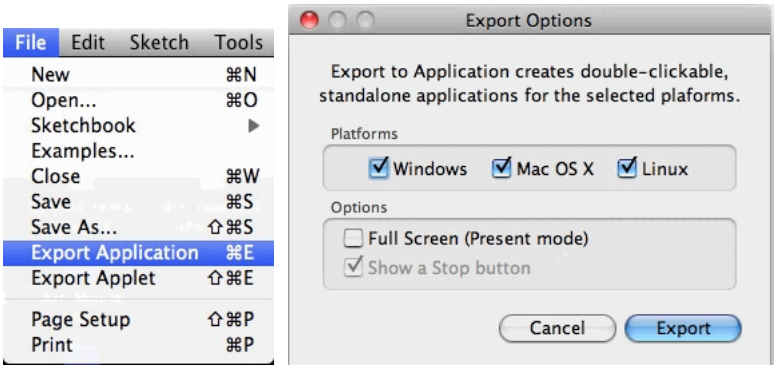
```
0) 365  76 rgb=  0 20 56 hsb= 60 99 56
1) 357  198 rgb=  0 27 67 hsb= 59 99 67
2) 264  146 rgb=  100 85 13 hsb= 13 86 100
3) 254  195 rgb=  100 94 0 hsb= 15 99 100
4) 214  211 rgb=  100 90 3 hsb= 15 96 100
5) 151  230 rgb=  91 66 8 hsb= 11 91 91
6) 199  282 rgb=  3 12 3 hsb= 34 75 12
7) 104  54 rgb=  100 79 6 hsb= 13 93 100
8) 148  91 rgb=  57 35 0 hsb= 10 100 57
9) 178  149 rgb=  30 18 7 hsb= 8 76 30
10) 257  87 rgb=  90 72 1 hsb= 13 98 90
11) 46  145 rgb=  0 16 53 hsb= 61 100 53
12) 116  169 rgb=  5 20 2 hsb= 30 88 20
13) 141  209 rgb=  17 28 7 hsb= 25 73 28
14) 305  270 rgb=  1 29 65 hsb= 59 97 65
15) 254  26 rgb=  0 20 56 hsb= 60 99 56
16) 256  118 rgb=  76 50 0 hsb= 11 100 76
================================================
0)   365    76   rgb=    0    20   56  hsb=  60  99   56
1)   357    198  rgb=    0    27   67  hsb=  59  99   67
2)   264    146  rgb=    100  85   13  hsb=  13  86   100
3)   254    195  rgb=    100  94   0   hsb=  15  99   100
4)   214    211  rgb=    100  90   3   hsb=  15  96   100
5)   151    230  rgb=    91   66   8   hsb=  11  91   91
6)   199    282  rgb=    3    12   3   hsb=  34  75   12
7)   104    54   rgb=    100  79   6   hsb=  13  93   100
8)   148    91   rgb=    57   35   0   hsb=  10  100  57
9)   178    149  rgb=    30   18   7   hsb=  8   76   30
10)  257    87   rgb=    90   72   1   hsb=  13  98   90
11)  46     145  rgb=    0    16   53  hsb=  61  100  53
12)  116    169  rgb=    5    20   2   hsb=  30  88   20
13)  141    209  rgb=    17   28   7   hsb=  25  73   28
14)  305    270  rgb=    1    29   65  hsb=  59  97   65
15)  254    26   rgb=    0    20   56  hsb=  60  99   56
16)  256    118  rgb=    76   50   0   hsb=  11  100  76
```

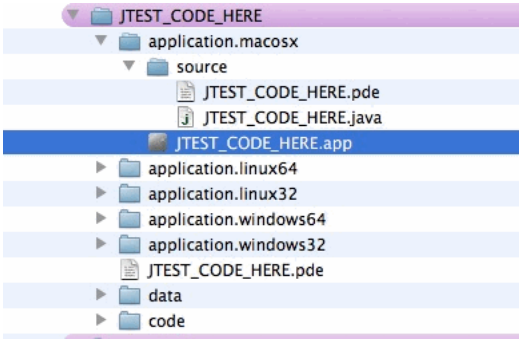
==========Pad_Text==========

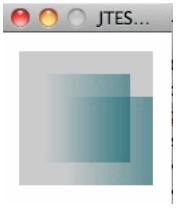
```
String p            = "rrr ";
char data[]         = {'C', 'C', 'C', 'P'};
String str2         = new String(data, 2, 2);
println               ("// test"+ str2);
while                 ( p.length() < 8 ) { p = " "+p;}
println               ("// test"+p);

// testCP
// test    rrr
```

==========EXPORT_AS_APPLICATION==========

```
import processing.core.*;

import java.applet.*;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.event.MouseEvent;
import java.awt.event.KeyEvent;
import java.awt.event.FocusEvent;
import java.awt.Image;
import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;
import java.util.zip.*;
import java.util.regex.*;

public class JTEST_CODE_HERE extends PApplet {
  public void setup() {
PImage img          = createImage(66, 66, ARGB);
img.loadPixels();
for                  (int i = 0; i < img.pixels.length; i++)
{ img.pixels[i]     = color(0, 90, 102, i % img.width * 2);
}
img.updatePixels();
image(                img, 17, 17);
image(                img, 34, 34);
  noLoop();
}
  static public void main(String args[]) {
    PApplet.main(new String[] { "--bgcolor=#FFFFFF", "JTEST_CODE_HERE" });
  }
}
```

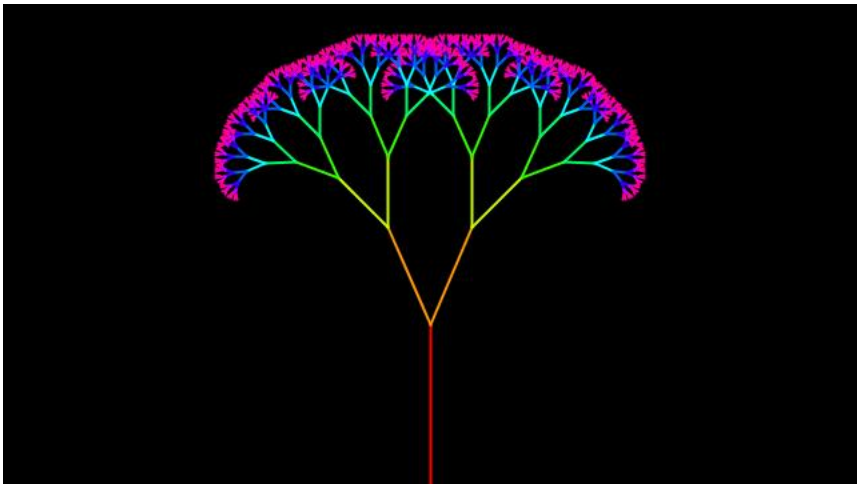






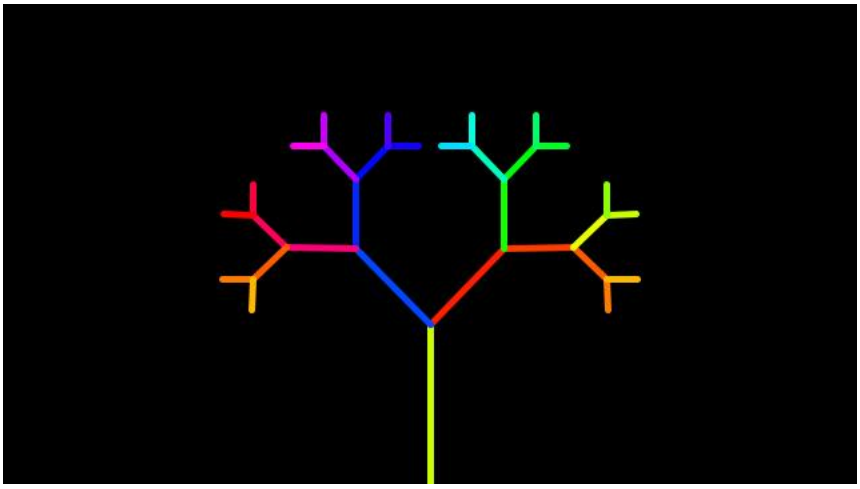
==========Recursive-Tree_View_Level_As_Hue==========

```
Renders a simple tree-like structure via recursion.
The branching angle is calculated as a function of the horizontal mouse location.
Move the mouse left and right to change the angle.
```



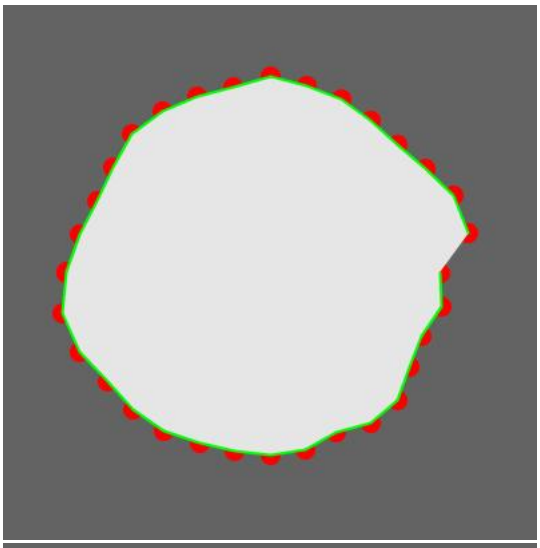
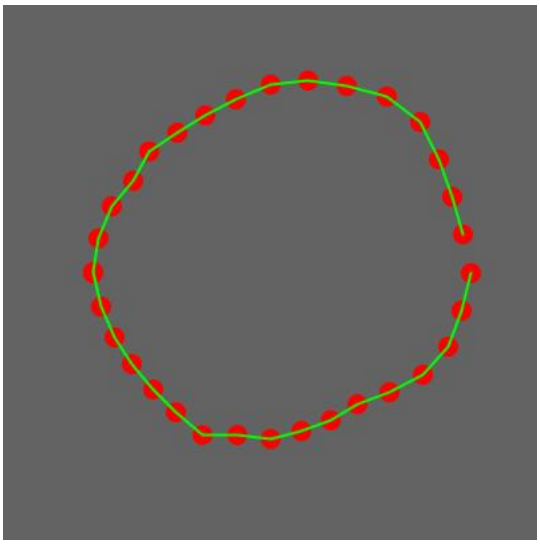
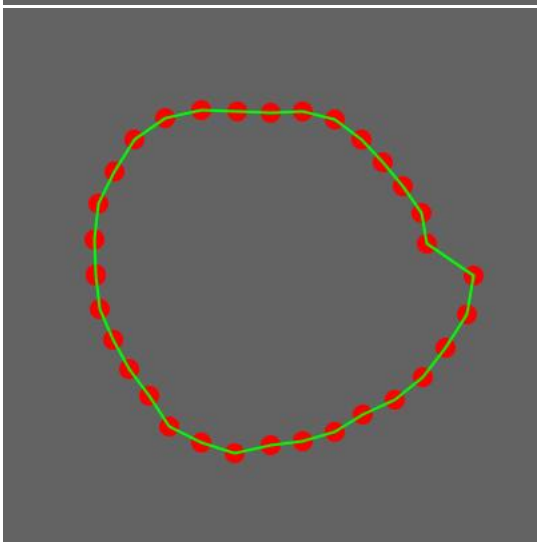
```
float                   theta;
float   hue           = 0;
void                    setup()
{ size                  (640, 360);
  smooth                ();
  colorMode(            HSB, 100);
  strokeWeight          (2);
}
void                    draw()
{ background            (0);
  frameRate             (30);
  stroke(               hue, 100, 100);
  float a               = (mouseX / (float) width) * 90f;          // pick an angle 0 to 90 degrees based on the mouse position
  theta                 = radians(a);                              // Convert it to radians
  translate             (width/2,height);                          // Start the tree from the bottom of the screen
  line                  (0,0,0,-120);                              // Draw a line 120 pixels
  translate             (0,-120);                                  // Move to the end of that line
  branch                (120);                                     // Start the recursive branching!
}
void                    branch(float h)
{ h                   *= 0.66;                                     // Each branch will be 2/3rds the size of the previous one
  if                    (h > 2)                                    // recursive exit condition!!length of the branch is 2 pixels or
less
{ pushMatrix            ();                                        // Save the current state of transformation (i.e. where are we
now)
  hue                   = hue + 10;
  stroke(               hue, 100, 100);
  rotate                (theta);                                   // Rotate by theta
  line                  (0, 0, 0, -h);                             // Draw the branch
  translate             (0, -h);                                   // Move to the end of the drawn branch
  branch                (h);                                       // Ok, now call myself to draw two new branches!!
  popMatrix             ();                                        // Whenever we get back here, we "pop" in order to restore the
previous matrix state
  hue                   = hue - 10;
  stroke(               hue, 100, 100);
  pushMatrix            ();                                        // Repeat the same thing, only branch off to the "left" this time!
  hue                   = hue + 10;
  stroke(               hue, 100, 100);
  rotate                (-theta);                                  // Rotate by theta
  line                  (0, 0, 0, -h);
  translate             (0, -h);
  branch                (h);                                       // Ok, now call myself to draw two new branches!!
  popMatrix             ();
  hue                   = hue - 10;
  stroke(               hue, 100, 100);
}
}
void keyPressed() { save("pict.jpg"); }
```

==========VIEW_RECURVIVE_AS_HUE==========



```
float                   theta;
float   hue             = 0;
float dhue              = 2;
void                    setup()
{ size                  (640, 360);
  smooth                ();
  colorMode(            HSB, 100);
  strokeWeight          (5);

}
void                    draw()
{ background            (0);
  frameRate             (30);
  stroke(               hue, 100, 100);
  float a               = (mouseX / (float) width) * 90f;          // pick an angle 0 to 90 degrees based on the mouse position
  theta                 = radians(a);                              // Convert it to radians
  translate             (width/2,height);                          // Start the tree from the bottom of the screen
  line                  (0,0,0,-120);                              // Draw a line 120 pixels
  translate             (0,-120);                                  // Move to the end of that line
  hue                   = 0;                                       // set color to red
  branch                (120);                                     // Start the recursive branching!
}
void                    branch(float h)
{ h                   *= 0.66;                                     // Each branch will be 2/3rds the size of the previous one
  if                    (h > 20)                                    // recursive exit condition!!length of the branch is 2 pixels or
less
{ pushMatrix            ();                                        // Save the current state of transformation (i.e. where are we
now)
  hue                   = (hue + dhue) % 100 ;
  stroke(               hue, 100, 100);
  rotate                (theta);                                   // Rotate by theta
  line                  (0, 0, 0, -h);                             // Draw the branch
  translate             (0, -h);                                   // Move to the end of the drawn branch
  branch                (h);                                       // Ok, now call myself to draw two new branches!!
  popMatrix             ();                                        // Whenever we get back here, we "pop" in order to restore the
previous matrix state
  hue                   = (hue + dhue) % 100 ;
  stroke(               hue, 100, 100);
  pushMatrix            ();                                        // Repeat the same thing, only branch off to the "left" this time!
  hue                   = (hue + dhue) % 100 ;
  stroke(               hue, 100, 100);
  rotate                (-theta);                                  // Rotate by theta
  line                  (0, 0, 0, -h);
  translate             (0, -h);
  branch                (h);                                       // Ok, now call myself to draw two new branches!!
  popMatrix             ();
  hue                   = (hue + dhue) % 100 ;
  stroke(               hue, 100, 100);
}
}
void                    keyPressed() { save("pict.jpg"); }
```

==========CREATE_ANY_SHAPE==========

```
float g = 0,j,i, x,y;
void                    setup()
{ size                  (400, 400);   // size sceen
  background            (102);  //gray
}
void                    draw()
{ background            (99);
  smooth                ();
  fill                  (230);
  noFill();
  beginShape            ();
  for                   (i=0; i<2*PI; i+=PI/16)
{ j                   = (i>6.2)?0:i;
  x                   = 200+(99+80*noise(j,g))*cos(j);
  y                   = 200+(99+80*noise(j,g))*sin(j);
  stroke                (255,0,0);                      // stroke is red
  strokeWeight          (15);
  point                 (x, y);                         // can do a point within a shape ##########
  stroke                (0,255,0);
  strokeWeight          (2);
  vertex                (x,y);                          // vertext to shape
} endShape              ();                             // endShape(CLOSE);  ##########
  g                   += .01;
  println               (g);
}

void mousePressed() { save("pict.jpg"); }
```

==========COMMANDLIST==========

```
  !  (logical NOT)                  //
  != (inequality)                   //
  %  (modulo) 20%100;               // Calculates remainder one number divided by another.
  && (logical AND)                  //
  <  (less than)                    //
  <= (less than or equal to)        //
  == (equality)                     //
  >  (greater than)                 //
  >= (greater than or equal to)     //
  ?                                 // s = (i < 50) ? 0 : 255;
  ArrayList ArrayList_obj;          // create an array of objects or list.
  ArrayList_obj = new ArrayList();  // create an array of objects.
```

```
ArrayList_obj.add(new Object( arg)) // element is added to list
ArrayList_obj.get(i)                // returns element at specified position in list.
ArrayList_obj.remove(i);            // deleted element at i
ArrayList_obj.size();               // length of the ArrayList
append(sa1, "MA ");                 // String[] sa2 = append(sa1, "MA ");  // Prints OH, NY, CA, MA
append(X, mouseX);                  // appends new point to X[]
arc(20, 140, 50, 50, 0, PI/3);      // arc( x y xrad yrad startangle stopangle )
BLEND ()                            // DARKEST,DIFFERENCE,EXCLUSION,MULTIPLY,SCREEN,OVERLAY,HARD_LIGHT,SOFT_LIGHT,DODGE,BURN
BLEND(img,x,y,w,h,ox,oy,ow,o,w,mod) // BLEND,ADD,SUBTRACT,LIGHTEST,    // BLEND(imag,inx iny inw inh outx outy outw outh mode)
background(204);                    // gray background(204, 0 ,0);
beginShape();                       // POINTS, LINES, TRIANGLES, TRIANGLE_FAN, TRIANGLE_STRIP, QUADS, and QUAD_STRIP.
bezierVertex(cx1,cy1,cx2,cy2,x,y)   // bezierVertex(cx1,cy1, cz1, cx2, cy2, cz2, x, y, z)
binary(c)                           // println(binary(-13312)); //Prints 11111111111111111100110000000000
binary(c, 16)                       // println(binary(-13312, 16)); // Prints 1100110000000000
byte a;                             // Declare variable "a" of type byte  127 to -128.
byte(c);                            // println("E"+" : "+ byte("E"));  // Prints "E : 69"
ceil(8.22);                         // Set "a" to 9
char m;                             // "m" of type char two bytes (16 bits) Unicode   m = 'A';  use single quotes
char(b);                            // b = 65
color c1 = color(204, 153, 0);      // colorMode(RGB, 255);
color c3 = get(10, 50);             // get pixel   color cp = get(x, y);
colorMode(HSB, 360, 100, 100);      // 360 degree, 100%, 100%
colorMode(RGB, 1);                  // red = color(1, 0, 0);
colorMode(RGB, 255);                // red = color(255, 0, 0);
constrain(val, min, max);           // Constrain Clip all points
continue                            // if (i == 70) {continue;} // If i is 70 skip to next iteration,
cos(radians(angle))                 // angle in degrees
createImage(w,h,mode);              // PImage edgeImg = createImage(w, h, RGB);// Create opaque
cursor(CROSS);                      // ARROW, CROSS, HAND, MOVE, TEXT, WAIT
dist(x1, y1, x2, y2);               // distance dist(x1, y1, z1, x2, y2, z2)
ellipse(60, 246, 55, 25);           // x y xrad yrad
endShape();                         // draws the shape
exit();                             // quit program
expand                              // y = expand(y,i);          // ###########  add to array y[]
fill(204, 102, 0);                   // fill(r1, g1, b1); fill(gray, alpha) fill(rgb, alpha)
filter(BLUR,0.6);                   // THRESHOLD,GRAY,INVERT,POSTERIZE,BLUR,OPAQUE,ERODE,DILATE
floor(2.88);                        // Set "a" to 2
for (int i=0; i<max;i++)            //
frame = (frame+1) % numFrames;      // Use % to cycle through frames
frameCount                          //
frameRate                           // println(frameRate); // 10.0   averaging in draw starting at 10
frameRate(30);                      //
get(x, y)                           // get pixel   color cp = get(30, 20);
get(x, y, w, h)                     // get pixels in region
height                              //
hex(c)                              // println(hex( -13312));     // Prints FFFFCC00
hex(c, 6)                           // println(hex( -13312, 6));  // Prints FFCC00
if (currentTime > lastTime+30)      //
if (mousePressed)                   //  if (mousePressed == true)
int a;                              // Declare variable "a" of type int 32 bits  +/-2,147,483,647
image(img, x, y, w, h)
join(list, ", ");                   // Joins a array or list of text using ,
key                                 // if(key      == 'w')
keyCode                             // if(keyCode == UP)
keyPressed                          // if(keyPressed)
keyPressed()                        //
keyTyped()                          //
keyReleased()                       //
line(mouseX, 20, pmouseX, 80);      // line(mouseX, mouseY, pmouseX, pmouseY);
loadBytes("N55W004.hgt");           // byte b[] = loadBytes("N55W004.hgt");           full binary file
loadFont                            // PFont metaBold  = loadFont("CourierNew36.vlw");
loadImage(name);                    // PImage img = loadImage("/Users/donsauer/Desktop/Tiger200.jpg")
loadPixels();                       // Loads image into pixels[] array. img2.loadPixels(); img2.pixels[i]
loadStrings("nouns.txt");           // String lines[] = loadStrings("nouns.txt");
long a;                             // int long 64 bits +/-9,223,372,036,854,775,807
loop()                              // restart looping draw()
map(value, 0, 255, 0, 1)            // map(value, lowin, highin, lowout, highout)
mask(maskImg);                      // img.mask(maskImg); visible where white in maskImg
mouseButton                         // if(mouseButton> 0)
mouseClicked()                      //
mouseDragged()                      //
mouseMoved()                        //
mousePressed                        // if(mousePressed)
mousePressed()                      //
mouseReleased()                     //
mouseX                              // line(mouseX, mouseY, pmouseX, pmouseY);
mouseY                              // line(mouseX, mouseY, pmouseX, pmouseY);
noFill();                           // transparent fill
noise(min,max);                     // float noiseVal = noise((mouseX+x)*noiseScale, mouseY*noiseScale);
noLoop();                           // stop draw() from looping
norm(value, low, high)              // Identical to map(value, low, high, 0, 1);
noStroke();                         //
noise(x, y)                         // Returns the Perlin noise    noise(x, y, z)      noise(x)
PImage                              // PImag_imgs = createImage(img.width, img.height, RGB);// Create opaque image same size
PImage                              // PImag_imgs = loadImage("/Users/donsauer/Downloads/Tiger100.jpg");
PImage                              // PImag_imgs = new PImage(msk.width,msk.height);
PImage                              // PImag_img2 = img.get(50, 50, 50, 100);   // x y w h
PImage[]                            // PImag_imgs = new PImage[numFrames];
PImag_imgs.filter(INVERT);          // invert colors
PImag_imgs.blend( ...)              // blend( img2,0,0,10,10,0,0,10,10, MULTIPLY);    // inx iny inw inh outx outy outw outh mode
PImag_imgs.filter( BLUR, 3);        //
PImag_imgs.filter( GRAY);           //
PImag_imgs.filter( INVERT);         //
PImag_imgs.filter( POSTERIZE,3);    //
PImag_imgs.filter( THRESHOLD, .3);  //
PImag_imgs.height                   // Image height
PImag_imgs.length                   // Prints "32"
PImag_imgs.loadPixels();            // Loads the pixel data for the image into its pixels[] array
PImag_imgs.mask(msk);               // Masks part of the image from displaying
PImag_imgs.pixels.length            // image zize
PImag_imgs.pixels[i]                // PImag_imgs.pixels[y*img.width + x] = color(sum);  // set gray value sum from kernel
PImag_imgs.resize(100, 50);         // w h   Changes the size of an image to a new width and height
```

```
  PImag_imgs.set(3, 3, red);          // x ,y, color
  PImag_imgs.get(3, 3);               // color c = PImag_imgs.get(x, y)
  PImag_imgs.updatePixels();          // Updates the image with the data in its pixels[] array
  PImag_imgs.width                    // Image width
  PImag_imgs.copy                     // img.copy(sx, sy, sw, sh, dx, dy, dw, dh) // copy()Copies the entire image
  PVector PVector_v1;                 // stores two or three variables
  PVector_v1 =                        // PVector_v1 = new PVector(x,y,z);
  PVector_v1.set()                    // Sets the x, y, z of vector           v1.set(40, 20,0);
  PVector_v1.get()                    // v2 = v1.get();;  // println(v2.x);  // Prints "20.0"
  PVector_v1.random2D()               // return random direction
  PVector_v1.random3D()               // return random direction
  PVector_v1.fromAngle(0.01);         // fromAngle(angle)  fromAngle(angle, target)  v = PVector.fromAngle(0.01)"[ 0.99995,
0.009999833, 0.0 ]"
  PVector_v1.mag()                    // Calculate magnitude  vector          val = v1.mag();
  PVector_v1.magSq()                  // Calculate magnitude  vector          val = v1.mag();
  PVector_v1.add(v2)                  // Adds vectors                         v1.add(v2);   println(v1);
  PVector_v1.sub(v2)                  // Subtracts one vector from another    v1.sub(v2);  println(v1);
  PVector_v1.mult(val)                // Multiplies the vector by a scalar    v1.mult(2);
  PVector_v1.div(val)                 // Divides the vector by a scalar       v1.div(2);
  PVector_v1.dist(v2)                 // distance between two points          val = v1.dist(v2);
  PVector_v1.dot(v2)                  // Calculates the dot product           val = v1.dot(v2);
  PVector_v1.cross(v2)                // Calculates the cross product         v3 = v1.cross(v2);
  PVector_v1.normalize()              // Normalizes the vector                v1.normalize(); println(v1);
  PVector_v1.limit(mag)               // Limits magnitude of vector           v1.limit(2) ;   println(v1);
  PVector_v1.setMag()                 // Sets the x, y, z of vector           v1.set(40, 20,0);
  PVector_v1.heading2D()              //
  PVector.angleBetween(v1,v2)         // angle between two vectors        val = PVector.angleBetween(v1, v2);
  PVector_v1.array()                  // vector to array                      float[] vals =  v1.array();
  PVector_v1.get()                    // Gets the x, y, z of vector           v3 = v1.get();
  PVector_v1.x                        // x component of vector                val  = v1.x
  PVector_v1.y                        // y component of vector
  PVector_v1.z                        // z component of vector
  pmouseX                             //
  pmouseY                             //
  point(x, y);                        //
  popMatrix();                        // "pop"  to restore the previous matrix state
  println(p[0]);                      // Prints "a"
  printMatrix();                      //
  pushMatrix();                       // Save the current state of transformation (i.e. where are we now)
  quad(20,180,80,190,69,210,30,220)   //
  radians(angle)                      //
  random(-range, range);              //
  rect(100, 20, 55, 55);              //
  rectMode(CENTER)  ;                 // center point and width and height.
  rectMode(CORNER) ;                  // upper left corner and width and height.
  rectMode(CORNERS) ;                 // one corner and opposite corner.
  rectMode(CORNERS);                  // one corner and opposite corner.
  rectMode(RADIUS) ;                  // center point half of the image's width and height.
  rotate(theta);                      // Rotate by theta
  round(9,2);                         // Sets rx to 9 closest to the value parameter
  String str1;                        // str1 =str(mouseX)+ ", "+ str(mouseY) ;
  String words                        // String words = "apple bear cat dog";
  String[] sa1                        // String[] sa1 = { "OH ", "NY ", "CA "};
  save("pict.jpg");                   // TIFF, TARGA, PNG, or JPEG file
  saveBytes("numb.dat", nums);        // byte[] nums = { 0, 34, 5, 127, 52};
  saveFrame("line-####.tif");         // "tif", "tga", "jpg", "png"
  saveFrame("t/line-####.png");       // Write frames for gif every third frame
  saveStrings( fname, data);          // saveStrings("lines.txt", "apple bear cat dog");
  scale(0.7);                         //
  selectInput                         // loadPath = selectInput( "The Prompt");  // Opens file chooser
  set(x, y, c);                       // set pixel
  size(200, 200);                     // size(100, 100, P3D);
  smooth();                           //
  split(words, ' ');                  //  split(lines[index], '\t');
  str(b);                             // converts to string
  stroke(0xAAFFFF11);                 // Alpha,rgbstroke(#CCFFAA);stroke(153);// grey
  strokeWeight(5);                    //
  subset(str,start,stop);             // String[] sa3 = subset(sa1, 2, 3);
  switch(letter )                     // {case 'A':println("Alpha"); break; }
  text("word", 15, 60);               // text(stringdata, x, y )
  text(s, 15, 20, 90, 170);           // text(stringdata, x, y, width, height)
  textFont(font);                     //
  textFont(metaBold);                 //
  textSize(14);                       //
  tint(value1, value2, value3, alpha) //tint(value1, value2, value3)   tint(hex, alpha) tint(gray, alpha)tint(color)
  translate(width/2,height);          // move the matrix
  triangle(20, 80, 20,120, 70, 120);  //
  unbinary(s1)                        // println(unbinary("00010000"));  // Prints 16
  updatePixels();                     // update pixel array to screen
  vertex(x, y);                       // vertex(x, y, z);
  vertex(x, y, u, v);                 // x, y, u = hoz texture,v = vert texture vertex(x, y, z, u, v);// x, y, z, u
  while((iy>255)||(iy<0)) {}
  width
  || (logical OR)
  array.length
  alpha(c)                            // float value = alpha(c);       //color c = color(0, 126, 255, 102);Sets "value" to "102"
  saturation(c);                      // float value = saturation(c);  //color c = color(0, 126, 255); Sets "value" to "126"
  red(c)                              // float value = red(c);         //color c = color(0, 126, 255, 102);Sets "value" to "102"
  brightness(c);                      // float value = brightness(c);  //color c = color(0, 126, 255); Sets "value" to "126"
```