Review

- Evaluating logical expressions
- switch vs. if-statements
- Iteration: while-loop
- Iteration: for-loop
- map(…) function

text()

- Strings can be drawn on a sketch using the text() function.
- Can set text position, font, size, alignment, …
- Font files are loaded from the data folder.

```
// Set attributes
textSize( sizeInPixels );
textAlign( {LEFT | CENTER | RIGHT}
    [,{TOP, BOTTOM, CENTER, BASELINE}] );
fill( color );

// Render text
text( string, X, Y );
text( string, X, Y, width, height );
```

```
// text
void setup() {
  size(500, 500);
  noLoop();
}

void draw() {
  // bounding box
  stroke(0);
  fill(255);
  rect(50, 50, 400, 400);

  // text options
  fill(0);    // black text
  text("Default", 50, 50, 400, 400);
  textAlign(CENTER);
  text("CENTER", 50, 50, 400, 400);
  textAlign(RIGHT);
  text("RIGHT", 50, 50, 400, 400);
  textAlign(CENTER, CENTER);
  text("CENTER-CENTER", 50, 50, 400, 400);
  textAlign(RIGHT, BOTTOM);
  text("RIGHT-BOTTOM", 50, 50, 400, 400);
  textAlign(LEFT, BOTTOM);
  text("LEFT-BOTTOM", 50, 50, 400, 400);
}
```

text1.pde

Functions

- A function names a block of code, making it reusable.
- Arguments can be “passed in” to function and used in body.
- Arguments are a comma-delimited set of variable declarations.
- Argument values are ***copies*** of passed values, not originals.
- Function must return a value that matches function declaration.

```
return_type function_name( argument_decl_list ) {
    statements;
    return value;
}
```

Function Examples

```
void setup() { … }
void draw() { … }


void line( float x1, float y1, float x2, float y2) { … }
… and other graphic functions


float float( … )
… and other type-conversion functions

… etc.
```

Functions

Modularity

- Functions allow the programmer to break down larger programs into smaller parts.
- Promotes organization and manageability.

Reuse

- Enables the reuse of code blocks from arbitrary locations in a program.

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

void draw() { }

void mousePressed()
{
  float secret = secretFunction( mouseX, mouseY );
  fill(0);
  background(255);
  text(secret, mouseX, mouseY);
}

float secretFunction( float x, float y )
{
  float r, x2, y2;

  x2 = x - (0.5*width);
  y2 = y - (0.5*height);
  r = sqrt(x2*x2 + y2*y2);

  return r;
}
```

Names of passed variables do not have to match names of variables in function. Values are copied.

The single value returned by the function is preceded by the 'return' keyword.

secret.pde

```
// Draw a grid to help position objects
void drawGrid(int gridSize)
{
  // Compute number of rows and columns
  int cols = width / gridSize;
  int rows = height / gridSize;

  // Set drawing parameters
  noFill();
  stroke(200);
  strokeWeight(1);
  textAlign(CENTER);
  textSize(25);

  // Draw grid
  for (int i = 0; i < cols; i++) {
    int x = i*gridSize;
    line(x, 0, x, height);
  }

  for (int j = 0; j < rows; j++) {
    int y = j*gridSize;
    line(0, y, height, y);
  }

  // Show position
  fill(0);
  String msg = "(" + str(mouseX) + ", " + str(mouseY) + ")";
  text( msg, width/2, height/2);
  //println( msg );
}
```

```
// Grid
// Copyright © 2011 Ilena Y. Pegan
//

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

void draw() {
  background(255);
  drawGrid(10);

  // Put your own drawing commands here

}
```

A generic function to draw a happy face.

```
// Draw happy face
void happyFace( float x, float y, float diam )
{
  // Face
  fill(255, 255, 0);
  stroke(0);
  strokeWeight(2);
  ellipseMode(CENTER);
  ellipse(x, y, diam, diam );

  // Smile
  float startAng = 0.1*PI;
  float endAng = 0.9*PI;
  float smileDiam = 0.6*diam;
  arc(x, y, smileDiam, smileDiam, startAng, endAng);

  // Eyes
  float offset = 0.2*diam;
  float eyeDiam = 0.1*diam;
  fill(0);
  ellipse(x-offset, y-offset, eyeDiam, eyeDiam);
  ellipse(x+offset, y-offset, eyeDiam, eyeDiam);
}
```

happyFace.pde

Draw a happy face at mouse position when mouse or 'h' key is pressed.

```
void setup() {
  size(500, 500);
  background(0);
  smooth();
}

void draw() { }

// If mouse pressed, draw large happy face
void mousePressed() {
  float diam = random(30, 60);
  happyFace( mouseX, mouseY, diam );
}

// If h-key pressed, draw small happy face
void keyPressed() {
  if (key == 'h' || key == 'H') {
    float diam = random(10, 30);
    happyFace( mouseX, mouseY, diam );
  }
}
```

happyFace.pde

```
void setup() {
  // Set up the drawing.
  // Draw the sky and the ground
  size(500, 500);
}

void draw() { }

void mousePressed() {

  // Use the mouseY position to decide whether drawing on sky or on ground
  if (mouseY <= 250) {

    println( "above horizon");
    // Draw your sky-appropriate object at the current mouseX, mouseY

  } else {

    println( "below horizon");
    // Draw your ground-appropriate object at the current mouseX, mouseY
  }
}
```