Review

- Mouse and Keyboard events
- Hue-Saturation-Brightness vs. Red-Green-Blue color models
- Decimal, Hex, Binary numbers and colors
- Variables and Data Types
- Data type conversion
- Other "things," including Strings and Images
- Operators: Mathematical, Relational and Logical
- Expressions and Expression Evaluation (PEMDAS)

Conditionals: if-statements

```
if ( boolean_expression ) {

       //statements;

}
```

```
// What does this do?
void draw() {

    if ( mouseY < 50 ) {
        println("the sky");
    }

}
```

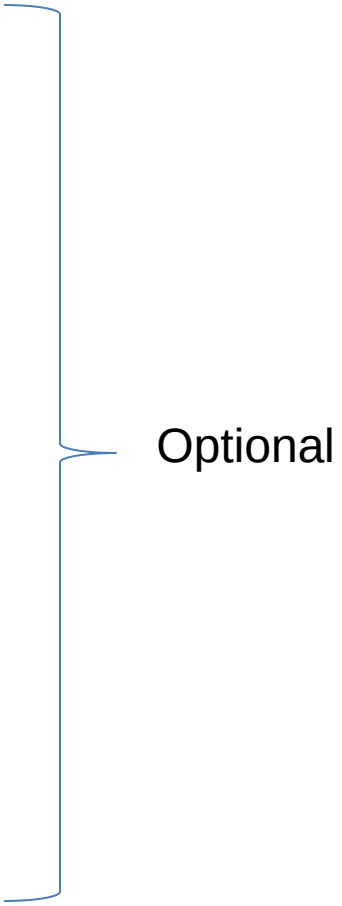
Conditionals: if-else-statement

```
if ( boolean_expression ) {

   //statements executed when boolean_expression is true;

} else {

   //statements executed when boolean_expression is false;

}
```

```
// What does this do?
void draw() {
   if ( mouseY < 50 ) {
       println("the sky");
   } else {
       println("the ground");
   }
}
```

Conditionals: if-statements

```
if ( boolean_expression_1 ) {
        //statements;
} else if ( boolean_expression_2 ) {
        //statements;
} else if ( boolean_expression_3 ) {
        //statements;
} else {
        //statements;
}
```



Conditionals: If-statement examples

```
if (j < i) { … }

if (true) { … }

if (keyCode == 38) { … }

if (mouseX > 250 && mouseY > 250) { … }

if (speed > 100.0 && bMoving == false) { … }

if (speed > 100.0 && !bMoving) { … }

if (x < 10 || x > 20) { … }
```

```
void setup() {
  size(500,500);
  smooth();
  ellipseMode(CENTER);
}

void draw() {

  if ( mouseX < 250 && mouseY < 250 )
  {
    stroke(255, 0, 0);
    fill(0, 255, 0);
  }
  else if ( mouseX < 250 && mouseY >= 250 )
  {
    stroke(255, 0, 0);
    fill(0, 0, 255);
  }
  else if ( mouseX >= 250 && mouseY < 250 )
  {
    stroke(0, 0, 255);
    fill(255, 0, 0);
  }
  else
  {
    stroke(0, 0, 255);
    fill(255);
  }
  ellipse(mouseX, mouseY, 50, 30);
}
```

What will this do?

if1.pde

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

void draw() {

  if ( mouseX > 100 )
  {
    background( 255, 0, 0 );
  }
  else if ( mouseX > 200 )
  {
    background( 0, 0, 255 );
  }

}
```

What does this do?

if2.pde

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

void draw() {

  if ( mouseX > 200 )
  {
    background( 255, 0, 0 );
  }

  if ( mouseX > 100 )
  {
    background( 0, 0, 255 );
  }

}
```

What does this do?

if3.pde

What does this do?

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

void draw() {}

void keyPressed()
{
  if (key =='a' || key == 'A')
  {
      println("Turning left");
  }
  else if (key =='s' || key == 'S')
  {
      println("Turning right");
  }
}
```

switch1.pde

The Walker

```
// The Walker

boolean walkPose = false;   // Current walk pose
float speed = 5.0;          // Max walking speed in any
direction
float cX = 100.0;           // Current walker location
float cY = 100.0;

void setup() {
  size(500, 500);
  smooth();
  frameRate(20);
}
```

Continued …

```
void draw() {

  background(255);
  fill(200);
  stroke(0);

  // Draw the walker
  // Space legs based on current walk step
  line(cX, cY, cX, cY+20);           // body
  ellipse(cX, cY, 10, 10);           // head

  if (walkPose == true)
  {
    line(cX-10, cY+10, cX+10, cY+10);// arms pose 1
    line(cX, cY+20, cX-10, cY+30);   // legs pose 1
    line(cX, cY+20, cX+10, cY+30);
  }
  else
  {
    line(cX-10, cY+5, cX+10, cY+15); // arms pose 2
    line(cX, cY+20, cX-5, cY+30);    // legs pose 2
    line(cX, cY+20, cX+5, cY+30);
  }

}
```

What will this do?

```
void keyPressed() {

  if (keyCode == UP)
  {
    walkPose = !walkPose;
    cY -= speed;
  }
  else if (keyCode == DOWN)
  {
    walkPose = !walkPose;
    cY += speed;
  }
  else if (keyCode == LEFT)
  {
    walkPose = !walkPose;
    cX -= speed;
  }
  else if (keyCode == RIGHT)
  {
    walkPose = !walkPose;
    cX += speed;
  }
}
```

walker.pde

Equations of Motion (Simplified)

s = displacement
t = time
v = velocity
a = acceleration

- Constant acceleration (a)

$$s_{i+1} = s_i + v_i \Delta t$$
$$v_{i+1} = v_i + a \Delta t$$

```
float sx = 0.0;    // x position
float sy = 0.0;    // y position
float vx = 1.0;    // x velocity
float vy = 1.0;    // y velocity
float ay = 0.2;    // y acceleration (gravity)

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);
}

void draw() {
  // Equations of Motion
  sx = sx + vx;
  sy = sy + vy;
  vy = vy + ay;

  // Bounce off walls
  if (sx <= 0.0 || sx >= width) vx = -vx;

  // Bounce off floor and
  // lose some velocity due to friction
  if (sy >= (height-10.0)) vy = -0.9*vy;

  // Draw at current location
  background(255);
  ellipse(sx, sy, 20, 20);
}
```

What does this do?

bounce.pde

Iteration

Repetition of a program block

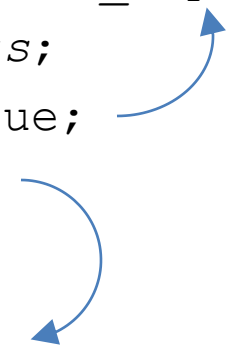
- Iterate when a block of code is to be repeated multiple times.

Options

- The while-loop
- The for-loop

Iteration: while-loop

```
while ( boolean_expression ) {
   statements;
   // continue;
   // break;
}
```



- Statements are repeatedly executed while the boolean expression continues to evaluate to **true**;
- To break out of a while loop, call **break;**
- To stop execution of statements and start again, call **continue;**
- All iterations can be written as while-loops.

```
void setup() {
  size(500, 500);
  smooth();

  float diameter = 500.0;
  while ( diameter > 1.0 ) {
   ellipse( 250, 250, diameter, diameter);
   diameter = diameter * 0.9;
  }
}

void draw() { }
```

What does this do?

while1.pde

```
void setup() {
  size(500, 500);
  smooth();

  float diameter = 500.0;
  while ( true ) {
   ellipse( 250, 250, diameter, diameter);
   diameter = diameter * 0.9;
   if (diameter <= 1.0 ) break;
  }
}

void draw() { }
```

while2.pde

Iteration: for-loop

```
for ( initialization; continuation_test; increment )
{
   statements;
   // continue;
   // break;
}
```

- A kind of iteration construct
- initialization, continuation test and increment commands are part of statement
- To break out of a while loop, call **break;**
- To stop execution of statements in block and start again, call **continue;**

```
void mousePressed() {

  for (int i = 0; i < 10; i++ )
  {
    print( i );
  }
  println();

}

void draw() { }
```

```
void mousePressed() {
  for (int i = 0; i < 10; i++ )
  {
    if ( i % 2 == 1 ) {
        continue;
    }
    print( i );
  }
  println();
}

void draw() { }
```

```
void setup() {
  size(500, 500);
  smooth();

  float diameter = 500.0;
  while ( diameter > 1.0 ) {
   ellipse( 250, 250, diameter, diameter);
   diameter = diameter - 10.0;
  }
}

void draw() { }
```

Initialize (runs only once)

Test to continue

Update

```
void setup() {
  size(500, 500);
  smooth();

  for (float diameter = 500.0; diameter > 1.0; diameter -= 10.0 )
  {
    ellipse( 250, 250, diameter, diameter);
  }
}

void draw() { }
```

for1.pde

Assignment #2 - Hints

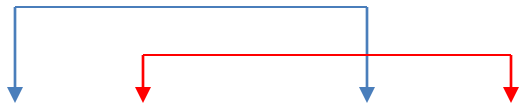
- Decide what to draw based on the relative position of mouse and horizon line.
  - If mouse is above horizon, draw sky-appropriate things
  - If mouse is below horizon, draw ground-appropriate things

- Calculate a scale factor based on the distance of the mouse to horizon and if above or below.
  - Use built-in map() function to convert mouse y-position to a scale factor
  - Use scale factor to size the object being drawn

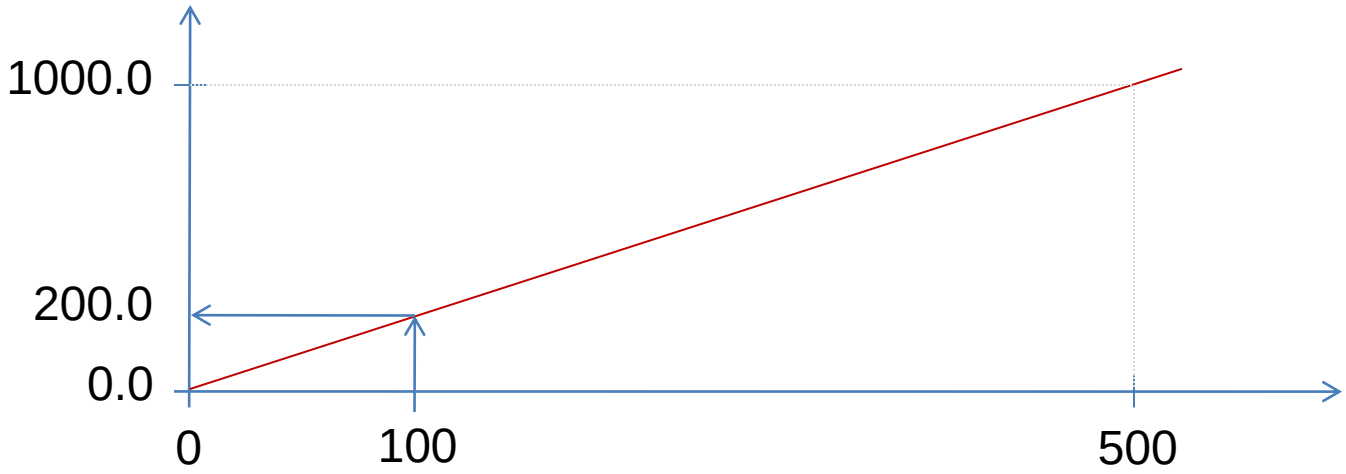
map

- A built-in function that maps some value from one range to another

```
map(value, low1, high1, low2, high2);
```



```
map(100,   0, 500,    0, 1000);   →  200.0
map(250,   0, 500, -250,  250);   →  0.0
```

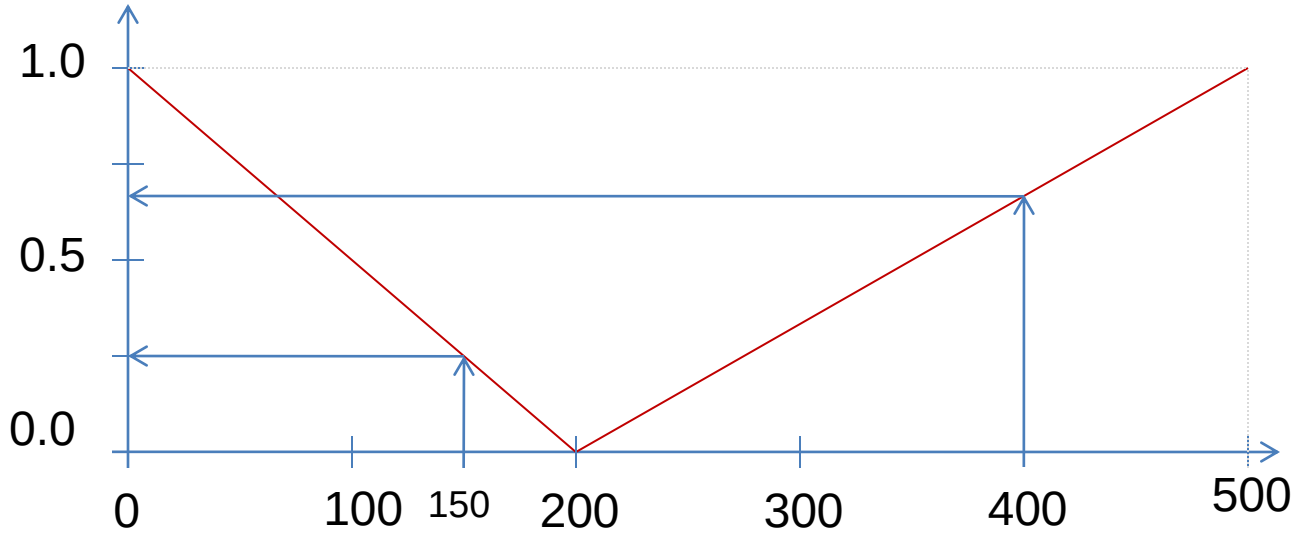


map

- A built-in function that maps some value from one range to another

```
map(value, low1, high1, low2, high2);
```

```
map(400, 200, 500,   0,   1);   → 0.6666667
map(150,   0, 200,   1,   0);   → 0.25
```



Pseudocode

- When the user clicks the mouse…
  - If the mouse's y-position is above the horizon
    - Use **one map function** to compute a scale factor that converts a range from the horizon to the top of the sketch (0.0) to a value between **0.0 and 1.0**
    - Set the object type to a sky-appropriate thing
  - If the mouse's y-position is below the horizon
    - Use **a second map function** to compute a different scale factor that converts a range from the bottom of the sketch (height) to the horizon to a value between **1.0 and 0.0**
    - Set the object type to a ground-appropriate thing
  - Use the mouse position and scale factor to draw appropriate object(s)

```
float delta = 5.0;
float factor = 0.0;

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

void draw() {

  factor+=0.2;
  noStroke();

  for (float r=0.0; r<height; r+=delta) {
    for (float c=0.0; c<width; c+=delta) {

      // Use factor to scale shape
      float x = map(c, 0.0, 500.0, 0.0, 3.0*TWO_PI);
      float y = map(r, 0.0, 500.0, 0.0, 3.0*TWO_PI);
      float shade = map(sin(factor)*sin(x)*sin(y), -1.0, 1.0, 0, 255);

      // Use factor to shift shade
//      float x = map(c, 0.0, 500.0, factor, factor+3.0*TWO_PI);
//      float y = map(r, 0.0, 500.0, factor, factor+3.0*TWO_PI);
//      float shade = map(sin(x)*sin(y), -1.0, 1.0, 0, 255);

      fill( shade );
      rect(r, c, delta, delta);
    }
  }
}
```

What does this do?

for2.pde