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

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

Conditionals: if-else-statement

```java
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

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

Optional

Conditionals: If-statement examples

```java
if (j < i) { … }  // if
if (true) { … }   // if
if (keyCode == 38) { … }  // if
if (mouseX > 250 && mouseY > 250) { … }  // if
if (speed > 100.0 && !bMoving == false) { … }  // if
if (speed > 100.0 && !bMoving) { … }  // if
if (x < 10 || x > 20) { … }  // if
```
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?

switch1.pde

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

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);
  }

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;
  }
}

what will this do?

walker.pde

Equations of Motion (Simplified)

s = displacement
v = velocity
t = time
\( a = \text{acceleration} \)

- Constant acceleration (a)
  \( s_{n+1} = s_n + v_n \Delta t \)
  \( v_{n+1} = v_n + a \Delta t \)
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.

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 setup() {
  size(500, 500);
  smooth();
  float diameter = 500.0;
  while (diameter > 1.0) {
    ellipse(250, 250, diameter, diameter);
    diameter = diameter - 10.0;
  }
}

void draw() {
}

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

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)