Variables & Scope

Global Variables
- Either predefined
- Or defined at top
- Are visible everywhere in the program

Local Variables
- Either parameters
- Or defined inside blocks
- Are visible ONLY in the block after they are defined

Processing: Math Functions

• Math functions return values:
  Example:
  ```
  void square(float x, float y, float side) {
    // example of function
  }
  ```
- Use:
  ```
  square(50, 50, 100); // draws a 100x100 square at 50, 50
  ```
• Processing has several pre-defined Math functions for calculation, trigonometry, and random number generation

Math Functions: Examples

• Calculation
  ```
  float x, y;
  y = 42;
  x = sqrt(y);
  ```
• Trigonometry
  ```
  float rad = radians(180);
  float deg = degrees(PI/4);
  ```
• Random
  ```
  float x = random(10); // returns a random number [0.0, 10.0)
  ```
**Example: Using random()**

```java
void setup() { // Create and set canvas
  size(300, 300);
  smooth();
  background(255);
} // setup()

void draw() {
  stroke(0);
  rect(random(width),
       random(height),
       random(5, 20),
       random(5, 20));
} // draw();
```

---

**2D Transformations: Translate**

```java
rect(20, 20, 40, 40);
rect(20+60, 20+80, 40, 40);
```

---

**Preserving Context**

- `translate()` will change the coordinate system for the entire duration of the draw() cycle. It resets at each cycle.

- Use pushMatrix() and popMatrix() to preserve context during a draw() cycle. i.e.

```java
pushMatrix();
translate(<x>, <y>);
<Do something in the new coordinate context>
popMatrix();
```

---

**Example: House() again!**

```java
// Draw a simple house
void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
  int roofHeight = houseHeight/2;
  int doorHeight = houseHeight/4;
  int doorWidth = houseWidth/4;
  pushMatrix();
  translate(houseX, houseY);
  // wall
  fill(206, 224, 14);
  rect(0, -wallHeight, houseWidth, wallHeight);
  // Draw Door
  fill(72, 26, 2);
  rect(houseWidth/2 - doorWidth/2, -doorHeight, doorWidth, doorHeight);
  // Draw roof
  fill(224, 14, 14);
  triangle(0, -wallHeight, houseWidth/2, -houseHeight, houseWidth, -wallHeight);
  popMatrix();
}
```
Key Computing Ideas

- The computer follows a program’s instructions. There are four modes:
  - **Sequencing**
    - All statements are executed in sequence
  - **Function Application**
    - Control transfers to the function when invoked
      - Control returns to the statement following upon return
  - **Repetition**
    - Enables repetitive execution of statement blocks
  - **Selection**
    - Enables choice among a block of statements

- All computer algorithms/programs utilize these modes.

Sequencing

- Refers to sequential execution of a program’s statements

```plaintext
void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
  house(20, 100, 50, 50);
  house(230, 100, 50, 75);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  // Draw a house at <houseX, houseY> (bottom left corner)
  // with width houseWidth and height houseHeight
  ...
} // house()
```

Function Application

- Control transfers to the function when invoked
- Control returns to the statement following upon return

```plaintext
void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  do this;
  then do this;
  and then do this;
  etc.
  size(200, 200);
  background(255);
  stroke(128);
  rect(20, 20, 40, 40);
} // draw()
```

Repetition

- Enables repetitive execution of statement blocks

```plaintext
void draw() {
  // Do-While Loop
  do { stuff to repeat } while (<condition>);

  // While Loop
  while (<condition>) {
    stuff to repeat
  }

  // For Loop
  for (<init>, <condition>, <update>) {
    stuff to repeat
  }
} // draw()
```

Loops: Controlled Repetition
Loops: Controlled Repetition

- **While Loop**
  ```
  while (condition) {
    stuff to repeat
  }
  ```

- **Do-While Loop**
  ```
  do stuff to repeat
  while (condition)
  ```

- **For Loop**
  ```
  for (init; condition; update) {
    stuff to repeat
  }
  ```
  All of these repeat
  the stuff in the block
  The block {
  } is called
  the Loop's Body

Conditions

- Conditions are **boolean** expressions.
- Their value is either true or false
e.g. POTUS is a woman
  5 is greater than 3
  5 is less than 3

Writing Conditions in Processing

- Boolean expressions can be written using boolean operators.
  Here are some simple expressions...

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>5 &lt; 3</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than/equal to</td>
<td>x &lt;= y</td>
</tr>
<tr>
<td>==</td>
<td>equal to</td>
<td>x == (y+j)</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
<td>x != y</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>x &gt; y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than/equal to</td>
<td>x &gt;= y</td>
</tr>
</tbody>
</table>

Logical Operations

- Combine two or more simple boolean expressions using logical operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>and</td>
<td>(x &lt; y) &amp; (y &lt; z)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td>!</td>
<td>not</td>
<td>! (x &lt; y)</td>
</tr>
</tbody>
</table>

  | A | B | A & B | A || B | A != B |
  |---|---|------|------|-------|
  | F | F |  F   |  F   |  F    |
  | F | T |  F   |  T   |  T    |
  | T | F |  F   |  T   |  T    |
  | T | T |  T   |  T   |  F    |
Conditions in While Loops

while (<condition>) {
  stuff to repeat
}

Loops: Controlled Repetition

• While Loop
  while (<condition>) {
    stuff to repeat
  }

• Do-While Loop
  do {
    stuff to repeat
  } while (<condition>)

• For Loop
  for (<init>; <condition>; <update>) {
    stuff to repeat
  }

Loops: Critical Components

• Loop initialization
  Things to do to set up the repetition

• Loop Termination Condition
  When to terminate the loop

• Loop Body
  The stuff to be repeated

• Loop update
  For the next repetition/iteration
Loops: Critical Components

• Loop initialization
  Things to do to set up the repetition

• Loop Termination Condition
  When to terminate the loop

• Loop Body
  The stuff to be repeated

• Loop update
  For the next repetition/iteration

Key Computing Ideas

• The computer follows a program’s instructions. There are four modes:
  – Sequencing
    All statements are executed in sequence
  – Function Application
    Control transfers to the function when invoked
    Control returns to the statement following upon return
  – Repetition
    Enables repetitive execution of statement blocks
  – Selection
    Enables choice among a block of statements

• All computer algorithms/programs utilize these modes.
**Selection**

- Enables choice among a block of statements

  Should I...
  
  { study }
  { sleep }
  { watch a movie }
  { veg out }
  { etc. }

- If-statements are one way of doing this

**Selection: If Statement**

```
if (<condition>) {
  do this
}
else {
  do that
}
else if (<condition>) {
  do that
}
else if (...) {
  ...
}
else {
  whatever it is you wanna do
}
```

At most ONE block is selected and executed.

Examples with if...