Processing Boot Camp
Control Structures

Creative Coding & Generative Art in Processing 2
Ira Greenberg, Dianna Xu, Deepak Kumar

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

```java
do this; size(200, 200);
then do this; background(255);
and then do this; stroke(128);
etc. rect(20, 20, 40, 40);
```

Function Application

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

```java
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
  // // draw()
  // house()
}```
Function Application

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

```java
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()
```

Parameter Transfer

Repetition

• Enables repetitive execution of statement blocks

```java
void draw() {
    do this;
    then this;
    and then this;
    etc.
} // draw()
```

Default `frameRate = 60`

Repeat `frameRate` times/second
Loops: Controlled Repetition

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

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

- **For Loop**
  
  ```java
  for (<init>; <condition>; <update>) {
    stuff to repeat
  }
  ```
While Loops

```java
void setup() {
  size(500, 500);
  smooth();
  background(164, 250, 238);
} // setup()

void draw() {
  fill(232, 63, 134, 127);
  stroke(0);
  int i = 0;
  while (i < width) {
    ellipse(i, height/2, 50, 50);
    i = i + 55;
  }
} // draw()
```

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
Conditions

• Conditions are **boolean** expressions.
• Their value is either **true** or **false**
  e.g.

  POTUS is a woman  \( \text{false} \)
  
  5 is greater than 3  \( \text{true} \)
  
  5 is less than 3  \( \text{false} \)

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 \leq 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 \neq 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 \geq y )</td>
</tr>
</tbody>
</table>
Logical Operations

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

\begin{align*}
& 
\text{&&} & \text{and} & (x < y) \ & \ (y < z) \\
\text{||} & \text{or} & (x < y) \ & \ (x < z) \\
\text{!} & \text{not} & (x < y) \\
\end{align*}

|   |   | A && B | A || B | !A |
|---|---|--------|--------|----|
| false | false | false | false | true |
| false | true  | false | true  | true |
| true  | false | false | true  | false |
| true  | true  | true  | true  | false |

Conditions in While Loops

```java
int i = 0;
while (i < width) {
    ellipse(i, height/2, 50, 50);
    i = i + 50;
}
```
10,000 circles!

```java
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
} // setup()

void draw() {
  fill(232, 63, 134, 127);
  stroke(0);
  int i = 0;
  while (i < 10000) {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
  }
} // draw()
```

Loops: Controlled Repetition

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

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

- **For Loop**
  ```java
  for (<init>; <condition>; <update>) {
    stuff to repeat
  }
  ```
Do-While Loops

```java
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
} // setup()

void draw() {
  fill(232, 63, 134, 127);
  stroke(0);
  int i = 0;
  do {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
  } while (i < 10000);
} // draw()
```

For Loops

```java
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
} // setup()

void draw() {
  fill(232, 63, 134, 127);
  stroke(0);
  for (int i = 0; i < 10000; i++) {
    ellipse(random(width),
            random(height),
            25, 25);
  }
} // draw()
```
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

```java
for (int i = 0; i < 10000; i++) {
  ellipse(random(width), random(height), 25, 25);
}

int i = 0;
while (i < 10000) {
  ellipse(random(width), random(height), 25, 25);
  i = i + 1;
}

int i = 0;
do {
  ellipse(random(width), random(height), 25, 25);
  i = i + 1;
} while (i < 10000);
```
Loops: Critical Components

```java
for (int i = 0; i < 10000; i++) {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
}
```

Termination Condition

```java
int i = 0;
do {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
} while (i < 10000);
```

Loop Update

```java
int i = 0;
while (i < 10000) {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
}
```

```
int i = 0;
while (i < 10000) {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
}
```
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

What happens when any one of these is missing or incorrectly encoded??
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
}

if ( <condition> ) {
  do this
} 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...