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

```plaintext
do this;
then do this;
and then do this;
etc.
```

```plaintext
size(200, 200);
background(255);
```

```plaintext
stroke(128);
rect(20, 20, 40, 40);
```

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
    house(50, 250, 200, 200);
    house(20, 100, 50, 50);
    house(230, 100, 50, 75);
} // draw()
```

```plaintext
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

```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
    ...
} // house()
```

Parameter Transfer

Repetition

- Enables repetitive execution of statement blocks

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

Default frameRate = 60

Repeat frameRate times/second
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
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 false
  5 is greater than 3 true
  5 is less than 3 false

Writing Conditions in Processing

• Boolean expressions can be written using boolean operators.
  Here are some simple expressions...
  
<table>
<thead>
<tr>
<th>Symbol</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:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A &amp;&amp; B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

Conditions in While Loops

```
int i = 0;
while (i < width) {
    ellipse(i, height/2, 50, 50);
    i = i + 55;
}
```
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

  while (condition) {
    stuff to repeat
  }

• Do-While Loop

  do {
    stuff to repeat
  } while (condition)

• For Loop

  for (init; condition; update) {
    stuff to repeat
  }
Do-While Loops

```
do{
  stuff to repeat
} while (<condition>);
```

```cpp
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

```
for(<init>; <condition>; <update>) {
  stuff to repeat
}
```

```cpp
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
Loops: Critical Components

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

Termination Condition

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

Loop Update

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

else if ( <condition> ) {
  do that
}

else if (…) {
...
}

else {
  whatever it is you wanna do
}

At most ONE block is selected and executed.