CS 101
Response and Events in Processing

Lecture 12
Interacting with a user

- It's very common for a program to “interact” with a human being
- How do humans interact with programs?
- Input devices!
  - What are some of the input devices we’ve talked about?
- In this lecture (and ch 5) we learn how to make our programs more interactive
Types of Interaction

- Mouse movement (review)
- Mouse click down
- Mouse click up
- Key press
setup and draw

A bit of review:

- The code inside of `setup` is run once, at the very beginning
- The code in `draw` is run repeatedly, until you exit the program
  - This is called the “draw loop”

```cpp
void setup() {
    // code ...
}

void draw() {
    // repeating code ...
}
```
setup and draw

- The changes made within draw do not update the canvas until draw finishes.
- The canvas draws a new “frame” each time the draw function executes once.
- By default, processing executes `draw` 60-times per second, thus 60fps.
  - The frame rate can be changed.

```java
void setup() {
    // code ...
}

void draw() {
    // repeating code ...
}
```
setup and draw

- Use the `frameRate()` function to adjust the number of times per-second a frame is drawn
- Adjusting this can make things more/less smooth
- Can also affect the speed at which an animation runs
  - For example, try modifying the frame-rate of `car_race`

```java
void setup() {
    frameRate(30);
    // code ...
}

void draw() {
    // repeating code ...
}
```
The animated fence

- Recall the animated fence example
- Code is available in “lectures” section of class site
  - Download code
  - Open
  - Try setting the `frameRate` to: 10, 30, 60, 120
  - (The code is on the next slide.)
The animated fence

```java
void setup() {
    size(300, 200);
}
int offset = 0;
void draw() {
    background(200, 230, 255);
    strokeWeight(0);
    fill(100, 255, 100);
    rect(0, 140, 300, 100);
    strokeWeight(4);
    for(int i = 0; i < 1001; i += 25) {
        line(offset + i, 50, offset + i, 150);
    }
    line(0, 75, 300, 75);
    line(0, 125, 300, 125);
    offset = offset - 1;
}
```
Following the mouse

- We’ve already used the `mouseX` and `mouseY` variables to have shapes “follow” the mouse
  - Follower assignment, other in-class activities
Following the mouse

Write a program that looks and behaves like this using an ellipse and mouseX / mouseY
Following the mouse

```java
void setup() {
  size(500, 500);
  frameRate(30);
  noStroke();
}
void draw() {
  fill(100, 200, 250);
  ellipse(mouseX, mouseY, 20, 20);
}
```
Following the mouse

`pmouseX` and `pmouseY` are the `mouseX` and `mouseY` position from previous frame.
Following the mouse

`pmouseX` and `pmouseY` are the `mouseX` and `mouseY` position from previous frame
Following the mouse

`pmouseX` and `pmouseY` are the `mouseX` and `mouseY` position from previous frame
Following the mouse

`pmouseX` and `pmouseY` are the mouseX and mouseY position from previous frame.
Following the mouse

`pmouseX` and `pmouseY` are the `mouseX` and `mouseY` position from previous frame.
Following the mouse

`pmouseX` and `pmouseY` are the mouseX and mouseY position from previous frame.
Following the mouse

`pmouseX` and `pmouseY` are the mouseX and mouseY position from previous frame.
Following the mouse

This code is similar to the program we just looked at.

What is different?

What will this code do when executed?

```cpp
void setup()
{
  size(500, 500);
  frameRate(30);
  strokeWeight(20);
}
void draw()
{
  stroke(100, 200, 250);
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```
void setup() {
  size(500, 500);
  frameRate(30);
  strokeWeight(20);
}
void draw() {
  stroke(100, 200, 250);
  line(pmouseX, pmouseY, mouseX, mouseY);
}
Following the mouse

`pmouseX` and `pmouseY` are the mouseX and mouseY position from previous frame
Following the mouse

the \texttt{dist()} function computes the distance from two points in terms of pixels

\[ \text{dist(pmouseX, pmouseY, mouseX, mouseY)} \]
Following the mouse

What will this code do?

The `dist()` function computes the distance in pixels between two points.

```java
void setup() {
  size(500, 500);
  frameRate(60);
}
void draw() {
  stroke(100, 200, 250, 150);
  float d = dist(pmouseX, pmouseY, mouseX, mouseY);
  strokeWeight(d/7);
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```
void setup() {
  size(500, 500);
  frameRate(60);
}
void draw() {
  stroke(100, 200, 250, 150);
  float d = dist(pmouseX, pmouseY, mouseX, mouseY);
  strokeWeight(d/7);
  line(pmouseX, pmouseY, mouseX, mouseY);
}
Following the mouse

The lines become thicker the faster you move
Built-in variables

- We’ve seen variables that keep track of the location of the mouse:
  - mouseX, mouseY
  - pmouseX, pmouseY
- Processing also keeps track of whether the mouse button is pressed or not:
  - mousePressed
Boolean variables

- `mousePressed` is a variable that determines whether or not the mouse is currently pressed-down
  - Automatically declared and updated, like `mouseX`, `mouseY`, etc…
- However, this variable is not an `int` or `float`!
- New type: `boolean`
  - Can only have two values: `true` or `false`
if statements

- We would like to express the flow of control just as we would in English, i.e.,
  
  if *it is raining*, then
  bring an umbrella

- Notice that the condition *it is raining* is either true or false

- We would like to say,
  
  if *the mouse is pressed*, then
  execute this code

- Notice that the condition *the mouse is pressed* is either true or false
if (test) {
    statements
}

for (init; test; update) {
    statements
}
Mouse Clicks

- The below code uses mousePressed to determine the mouse clicks - how does it work? What does it do?

```java
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}
void draw() {
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```
The below code uses mouse clicks - how does it work? What does it do?

```java
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}
void draw() {
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```

Key “new” features:
- if-statement
- the `mousePressed` variable
- `mousePressed == true`
if (mousePressed == true) {
    // lines of code
}

if is a Processing keyword. It tells process that we are about to specify an if-statement

Define a stop condition (test), just like the condition of for-loop

```
if (mousePressed == true) {
    // lines of code
}
```

The lines of Processing code to run, only if the condition (test) is true

Open and close curly-bracket
if-statement vs for-loop

if (mousePressed == true) {
    // lines of code
}

for (int i = 0 ; i < 10 ; i += 1) {
    // lines to repeat
}

An if-statement is like a non-repeating for-loop
Now does this code make a little more sense?

```java
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}

void draw() {
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```
Mouse Clicks

What does this code do? (don’t try it, just discuss in your groups)

```java
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}
void draw() {
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```
What does this code do? (don’t try it, just discuss in your groups)

```java
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}
void draw() {
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```
Mouse Clicks

When using a boolean variable, can just write the variable without ==

```cpp
void setup() {
  size(500, 500);
  strokeWeight(30);
  background(100);
  stroke(250, 200, 100, 150);
}
void draw() {
  if (mousePressed) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```
Following the mouse

Write a program that has behavior similar to this canvas

When mouse is pressed, a line follows mouse from corner of screen

When not pressed nothing follows mouse
Following the mouse

```java
void setup()
{
    size(500, 500);
    stroke(0, 200, 255);
    strokeWeight(7);
}
void draw()
{
    background(255, 255, 0);
    if (mousePressed) {
        line(0, 0, mouseX, mouseY);
    }
}
```
if-else

if (test) {
    statements 1
} else {
    statements 2
}

if-else-if

if (test 1) {
    statements 1
} else if (test 2) {
    statements 2
}
setup and draw

- If, else-if, and else are useful for many things in computer programming
- In the context of clicks, great for determining things like:
  - a left click vs right-click
  - the area of canvas clicked
- Let’s try it out
Mouse Clicks

What does this code do?

Discuss in your groups!

Use whiteboards!

```java
void setup() {
  size(300, 300);
  frameRate(60);
}

void draw() {
  if (mousePressed == true) {
    if (mouseX < 100) {
      fill(50, 200, 255);
    } else if (mouseX < 200) {
      fill(100, 255, 100);
    } else {
      fill(150, 0, 200);
    }
    ellipse(mouseX, mouseY, 50, 50);
  }
}
```
void setup() {
  size(300, 300);
  frameRate(60);
}

void draw() {
  if (mousePressed == true) {
    if (mouseX < 100) {
      fill(50, 200, 255);
    } else if (mouseX < 200) {
      fill(100, 255, 100);
    } else {
      fill(150, 0, 200);
    }
    ellipse(mouseX, mouseY, 50, 50);
  }
}
Mouse Buttons

- Another pre-declared variable!
- `mouseButton` stores which mouse Button is being pressed (assuming `mousePressed` is true)
  - Can be `LEFT` or `RIGHT`
- You can use this variable inside of an if-statement that checks if the mouse is being pressed
Mouse Buttons

The code on the right uses mouse clicks and coordinates to change what is drawn.

Again, can you tell me what it will do, without running it?

Discuss!

void draw() {
  if (mousePressed == true) {
    if (mouseButton == LEFT) {
      fill(255, 100, 100);
    } else if (mouseButton == RIGHT) {
      fill(50, 50, 255);
    }
  }
  ellipse(mouseX, mouseY, 50, 50);
}
void draw() {
  if (mousePressed == true) {
    if (mouseButton == LEFT) {
      fill(255, 100, 100);
    } else if (mouseButton == RIGHT) {
      fill(50, 50, 255);
    }
    ellipse(mouseX, mouseY, 50, 50);
  }
}
Key Press

- Similar to `mousePressed` the `keyPressed` variable is **true** if a keyboard key is “currently” being pressed
- **false** if not

```java
void setup()
{
  size(400, 400);
}
void draw()
{
  background(100, 100, 100);
  fill(255, 0, 0);
  if (keyPressed)
  {
    ellipse(200, 200, 300, 300);
  }
}
```
Key Press

- Another special variable: `key`
- If a key is being pressed, `key` will be the letter that corresponds to that key
- Use this to differentiate between key-presses

```cpp
void setup() {
  size(300, 300);
  textSize(200);
}

void draw() {
  background(100, 100, 100);
  fill(255, 0, 0);
  if (keyPressed) {
    text(key, 100, 180);
  }
}
```
Key Press and Character

- The variable `key` is analogous to `mouseButton`.
- When a button is pressed, has the corresponding letter stored inside of it.
- The `key` variable is not an `int`, `float`, or `boolean`.
- New type: `char`
  - Short for “character”
Key Press and Character

- A **char** variable can “store” a single character on the keyboard
- Char values are surrounded by single-quotes
- For example, to declare a new char variable:

  ```
  char letter = 'A';
  char period = '.';
  char first_letter = 'x';
  char number = '4';
  ```

- Chars are case-sensitive
Key Press and Character

• To compare a char variable:

  if (key == 'w') { ...  
  if (letter == 'i') { ...  
  if (key == letter) { ...  

• These types of comparisons are needed to check which key on the keyboard is being pressed
Key Press

• Here’s an example of using the `key` variable to differentiate between keys.

```java
void draw() {
  background(100, 100, 100);
  if (keyPressed) {
    if (key == 'r') {
      fill(255, 0, 0);
    }
    else if (key == 'g') {
      fill(0, 255, 0);
    }
    else if (key == 'b') {
      fill(0, 0, 255);
    }
  }
  rect(50, 50, 200, 200);
}
```
Key Press

What does this code do?

Discuss!
void draw() {
    background(100, 100, 100);
    if (keyPressed) {
        if (key == 'r') {
            fill(255, 0, 0);
        }
        else if (key == 'g') {
            fill(0, 255, 0);
        }
        else if (key == 'b') {
            fill(0, 0, 255);
        }
    }
    rect(50, 50, 200, 200);
}
Type review

- So far, we’ve learned about 4 different variable/value types
  - int
  - float
  - boolean
  - char

- Future coursework may require you to use any (or multiple) of these types, so make sure you understand the differences between them!
Materials

- Required Materials
  - GSWP chapter 5