Facebook to integrate WhatsApp, Instagram and Messenger

“Until now, WhatsApp, Instagram and Messenger have been run as separate and competing products. Integrating the messaging parts might simplify Facebook's work. It wouldn't need to develop competing versions of new features, such as Stories, which all three apps have added with inconsistent results.

“Cross-platform messaging may also lead the way for businesses on one platform to message potential customers on another. And it might make it easier for Facebook to share data across the three platforms, to help its targeted advertising efforts.”

Administrivia

- **Assignments:**
  - Reading Check 4 due tomorrow @ 3:30 pm (1/31)
  - Jumping Monster due Friday (2/1)

- “Big Idea” this week: Digital Distribution

- **Upcoming:** Creativity Project, Midterm (2/11)
Jumping Monster

- Using *expressions* and *conditionals* in conjunction with *variables* and *user input* (today) to control what is drawn as well as motion:
Lecture Outline

- **Other Useful Processing Tools**
- **User Input and Output**
  - Mouse (input)
  - Keyboard (input)
  - Text (output)
Transparency/Opacity

- You can add a 4th argument to a color!
  - This also applies to the `fill()` and `stroke()` functions
- This argument also takes an integer between 0–255
  - 0 is fully transparent (invisible)
  - 255 is fully opaque (the default)
Custom Shapes

- Define vertices between `beginShape()` and `endShape()`
  - If planning to reuse, best to create in a separate function
    - `so can use like rect(), ellipse(), etc`

```plaintext
size(480, 240);
fill(153, 176, 180);
beginShape();
  vertex(100, 240);
  vertex(200, 180);
  vertex(220, 120);
  vertex(160, 40);
  vertex(420, 120);
  vertex(320, 160);
  vertex(400, 180);
  vertex(280, 200);
  vertex(260, 240);
endShape();
fill(0);
ellipse(310, 120, 16, 16);
```
Functions Practice: Diamond

- Fill in the code to produce:

```cpp
void diamond( float x, float y, float w, float h ) {
    beginShape();
    vertex( __________, __________ );
    vertex( __________, __________ );
    vertex( __________, __________ );
    vertex( __________, __________ );
    vertex( __________, __________ );
    endShape();
}
```
Lecture Outline

- Other Useful Processing Tricks
- **User Input and Output** *
  - Mouse
  - Keyboard
  - Text

* We will look at a subset of the available Processing commands. For a full list, see the Processing Reference.
Reminder: System Variables

- Special variables that hold values related to the state of the program, often related to user input
  - You don’t need to declare these variables
  - These variables will update automatically as the program runs
  - Colored **pink/magenta-ish** in the Processing environment

- We’ve used some of these already:
  - `width`, `height`, `frameCount`

- We’ll see a lot more today
The Mouse

- **System variables:**
  - \( \text{mouseX} \) – x-coordinate of mouse in current frame
  - \( \text{mouseY} \) – y-coordinate of mouse in current frame
  - \( \text{pmouseX} \) – x-coordinate of mouse in previous frame
  - \( \text{pmouseY} \) – y-coordinate of mouse in previous frame
  - \( \text{mousePressed} \) – is a button currently being pressed?

- **Built-in functions:**
  - **\( \text{mousePressed}() \)** – called very time a button is pressed
  - **\( \text{mouseReleased}() \)** – called every time a button is released
Example: Drawing Dots

```java
void draw() {
    if (mousePressed) {
        fill(0, 0, 255);  // blue if mouse is pressed
    } else {
        fill(255, 0, 0);  // red otherwise
    }
    ellipse(mouseX, mouseY, 5, 5);  // draw circle
}
```
Example: Path Drawing

- We just wrote a *dot*-drawing program
- We can additionally use `pmouseX` and `pmouseY` to create a *path*-drawing program

```java
void setup() {
    size(500,500);  // set drawing canvas size
    strokeWeight(8);  // thicker lines
    stroke(0,0,0, 120);  // black line with some transparency
    frameRate(30);  // slow down the frame rate
}

void draw() {
    line(mouseX, mouseY, pmouseX, pmouseY);  // drawing the path your mouse takes
}
```
Hovering Over a Rectangle

if (mouseX >= x) 

if (mouseX <= x + w)
Hovering Over a Rectangle

\[ \text{if (mouseY } \geq y) \]
\[ \text{if (mouseY } \leq y + h) \]
Hovering Over a Rectangle

\[
\text{if (}( \text{mouseX} \geq x) \quad \&\& \\
\text{mouseX} \leq x + w) \quad \&\& \\
\text{mouseY} \geq y) \quad \&\& \\
\text{mouseY} \leq y + h))
\]
Hovering Over a Rectangle

```cpp
int x = 100;    // x-position of upper-left corner
int y = 160;    // y-position of upper-left corner
int w = 200;    // width of rectangle
int h = 160;    // height of rectangle

void setup() {
    size(500,500);  // set drawing canvas size
    noStroke();     // no shape outlines
}

void draw() {
    background(204); // clear the canvas

    if ((mouseX >= x) && (mouseX <= x+w) && (mouseY >= y) && (mouseY <= y+h)) {
        fill(0);        // black is mouse is hovering over
    } else {
        fill(255);      // white otherwise
    }

    rect(x, y, w, h); // draw the rectangle
}```
The Keyboard

- **System variables:**
  - `key` – stores the ASCII value of the last key press
  - `keyCode` – stores codes for non-ASCII keys (e.g. UP, LEFT)
  - `keyPressed` – is any key currently being pressed?

- **Built-in functions:**
  - `keyPressed()` – called every time a key is pressed

- **New datatype:** `char`
  - Stores a single character (really just a number)
  - Should be surrounded by *single* quotes
  - *e.g.* `char letter = 'a';`
  - *actually the ASCII value for ‘a’*
Example: What does this code do?

```cpp
int position = 0;

void setup() {
  size(400, 100);
  noStroke();
  background(0);
  fill(0);
}

void draw() {
  ellipse(position, 40, 40, 40);
} // draws a circle at (position, 40) every frame

void keyPressed() {
  if(key == 'g') { // change fill to green
    fill(0, 255, 0);
  }

  if(key == 'y') { // fill to yellow
    fill(255, 255, 0);
  }

  if(key == 'm') { // fill to magenta
    fill(255, 0, 255);
  }

  position = position + 50; // position+=50;
}
```

- runs anytime a key is pressed
- these are chars
- draw next circle at new position
- this executes no matter which key is pressed.
Example: Keyboard Dots

```java
int position = 0;

void setup() {
    size(400, 100);
    noStroke();
    background(0);
    fill(0);
}

void draw() {
    ellipse(position, 40, 40, 40);
}

void keyPressed() {
    if(key == 'g') {
        fill(0, 255, 0);
    }

    if(key == 'y') {
        fill(255, 255, 0);
    }

    if(key == 'm') {
        fill(255, 0, 255);
    }

    position = position + 50; // position+=50;
}
Example: Moving a Rectangle

- **Note:** non-character keys, such as the arrow keys (UP, DOWN, LEFT, RIGHT) are *coded* keys.

```java
if(keyPressed) {
    if(key == CODED) {
        if(keyCode == LEFT) {
            x = x - 1;
        }
    }
}
```
Example: Moving a Rectangle

```java
int x = 215;

void setup() {
    size(480, 120);
}

void draw() {
    background(0);
    rect(x, 45, 50, 50);

    if(keyPressed) {
        if(key == CODED) {
            if(keyCode == LEFT) {
                x = x - 1;
            }
        }
    }

    if(keyCode == RIGHT) {
        x = x + 1;
    }
}
```
Text Output

- `println(yourText);`
  - Prints `yourText` to the *console*, which is the black area below your Processing code
  - Useful for debugging

- `text(yourText, x, y);`
  - Prints `yourText` on the drawing canvas, starting with the *bottom-left* corner at coordinate `(x, y)`
  - Change the size of your text using `textSize(size);`

- `yourText` should be between *double* quotes
  - We will talk more about the datatype *String* later
Example: Displaying Typed Keys

![Displaying Letter Z](image)

```cpp
void setup() {
  size(120, 120);
  textSize(64);
  textAlign(CENTER);
}

void draw() {
  background(0);
  text(key, 60, 80);
}
```
Looking Forward

- Next week is the Creativity Assignment
  - In pairs, you will be asked to brainstorm TWO Processing projects of your choice
  - You will implement and submit ONE of your two projects
  - The point is to use the tools available to you to make something fun and creative!
  - Planning document due Tuesday (2/5)
  - Actual programs due next Friday (2/8)

- Portfolio Update 1 is due Wednesday (2/6)
  - Taijitu, Logo Design, Lego Family, Animal Functions
  - Ask your TAs for assistance if you encounter problems!