How the Facebook algorithm update will change your news feed

“Zuckerberg has said his goal for this year is to fix Facebook, whether by protecting against foreign interference and abuse or by making users feel better about how they spend time on Facebook. So, the company is set to try to have users see fewer posts from publishers, businesses and celebrities, and more from friends and family.

“Facebook says it will start prioritising news sources deemed trustworthy in the US and then internationally. It says it has surveyed a ‘diverse and representative sample’ of US users and next week it will begin testing prioritising the news sources deemed trustworthy.”

• http://metro.co.uk/2018/01/20/facebook-algorithm-update-will-change-news-feed-7245129/
Administrivia

- **Assignments:**
  - Reading Check 3 due tomorrow *before lab* (1/25)
  - *Jumping Monster* due Friday (1/26)

- “Big Idea” this week: The Internet

- **Upcoming:** Creativity Project, Midterm (2/5)
  - Vote on Piazza for Midterm Review Session?
Lecture Outline

- Other Useful Processing Tools
- User Input and Output
  - Mouse (input)
  - Keyboard (input)
  - Text (output)
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:
  - `mouseX`, `mouseY`, `width`, `height`

- We’ll see more today
Drawing and Frames

- Control and track how frequently `draw()` runs
  - Each time `draw()` runs, it is called a new `frame`

- `frameRate()` changes the desired number of frame updates per second
  - Larger argument is faster
  - Default is `frameRate(60)`

- System variable `frameCount` returns the number of frames since the start of the program
  - Starts at 0 in `setup()`
Drawing and Frames

- Control and track how frequently \texttt{draw()} runs
  - Each time \texttt{draw()} runs, it is called a new \textit{frame}

- \texttt{noLoop()} stops \texttt{draw()} from being continuously executed
  - Can restart using \texttt{loop()}

Transparency/Opacity

- You can add a 4\textsuperscript{th} argument to a color!
  - This also applies to the \texttt{fill()} and \texttt{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.

```javascript
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(x, y - h/2);
    vertex(x + w/2, y);
    vertex(x, y + h/2);
    vertex(x - w/2, y);
    vertex(x, y - h/2);
    endShape();
}
```

```cpp
void draw() {
    fill(255,0,0);
    diamond(50,50,100,100);
    fill(0,255,0);
    diamond(250,250,100,100);
    fill(0,0,255);
    diamond(450,450,100,100);
}
```
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.
The Mouse

- **System variables:**
  - `mouseX` – x-coordinate of mouse in current frame
  - `mouseY` – y-coordinate of mouse in current frame
  - `pmouseX` – x-coordinate of mouse in previous frame
  - `pmouseY` – y-coordinate of mouse in previous frame
  - `mousePressed` – is a button currently being pressed?

- **Built-in functions:**
  - `mousePressed()` – called very time a button is pressed
  - `mouseReleased()` – called every time a button is released
Example: Path Drawing

- Last lecture we wrote a *dot*-drawing program
  ```java
  ellipse(mouseX, mouseY, 10, 10);
  ```

- 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)
Hovering Over a Rectangle

if (mouseY >= y)

if (mouseY <= y + h)
Hovering Over a Rectangle

if (mouseX >= x) && (mouseX <= x + w) && (mouseY >= y) && (mouseY <= y + h)
Hovering Over a Rectangle

```java
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';`
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') {
        fill(0, 255, 0); // change fill to green
    }
    if (key == 'y') {
        fill(255, 255, 0); // fill to yellow
    }
    if (key == 'm') {
        fill(255, 0, 255); // fill to magenta
    }
    position = position + 50; // position+=50;
    // this executes no matter which key is pressed.
}
```
Example: Keyboard Dots

```cpp
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

```cpp
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, particularly your portfolio

- `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

```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 (1/30)
  ▪ Actual programs due next Friday (2/2)

❖ Portfolio Update 1 is due Wednesday (1/31)
  ▪ Taijitu, Logo Design, Lego Family, Animal Functions
  ▪ Ask your TAs for assistance if you encounter problems!