CISC 1600 Lecture 8
Interactivity in Processing

Topics:

Processing syntax

Read processing demos

Interactivity

Animation

Processing's `if` statements are the same as JavaScript's

- Processing implements the imperative paradigm
- The syntax of its `if` statement is identical to JavaScript's

```
if (mouseX < 150 && mouseY >= 150) {
    // mouseX is less than 150
    // AND mouseY is greater than or equal to 150
} else if (mouseX < 150) {
    // What can we say about mouseY here?
} else {
    // What can we say about mouseX here?
}
```

Processing's `for` loops are the same as JavaScript's

- For loops are good when the number of iterations is known in advance

```
for (int i=0; i<5; i++) {
    rect(0, 60*i, 40, 40);
}
```

- The three statements in the parentheses are
  - Initialization: run once before loop
  - Continuation: loop continues until this is not true
  - Increment: update counter, run before each iteration

Processing's `while` loops are the same as JavaScript's

- While loops are good when you don't know the number of iterations in advance
- Repeat a set of statements while a condition holds

```
while (y < height - 60) {
    rect(0, y, 40, 40);
    y = y + 60;
}
```

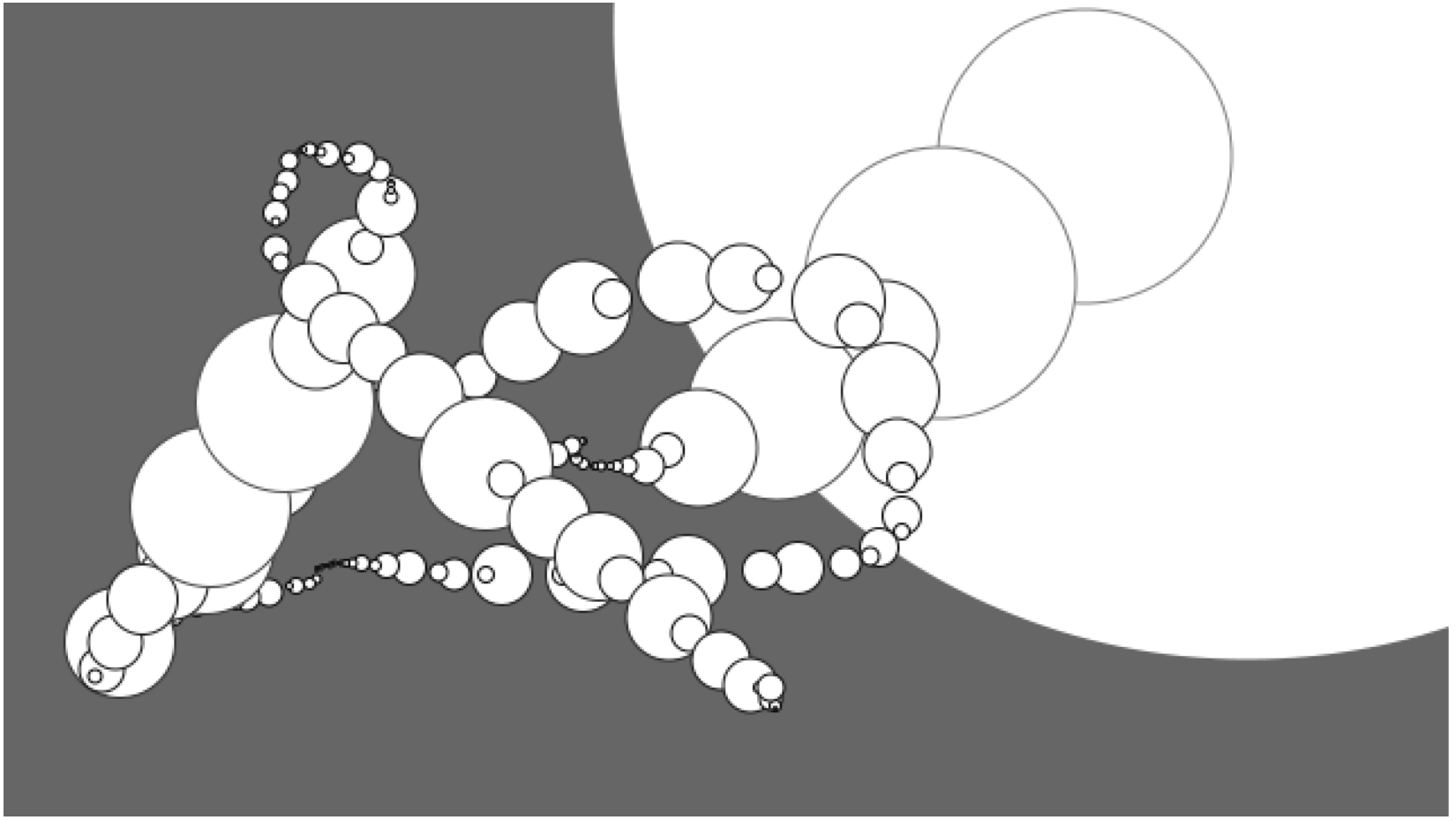
Processing uses typed variables, unlike JavaScript

- JavaScript uses untyped variables
  - All variables are declared using the `var` keyword
  - The same variable can hold a string and then a number
- Processing (like Java) uses typed variables
  - Variables must be declared to be a specific type
  - `int, float, double, boolean, String, etc.`
  - You cannot assign a value of one type to a variable of another type
  - Create your own types using the `class` keyword

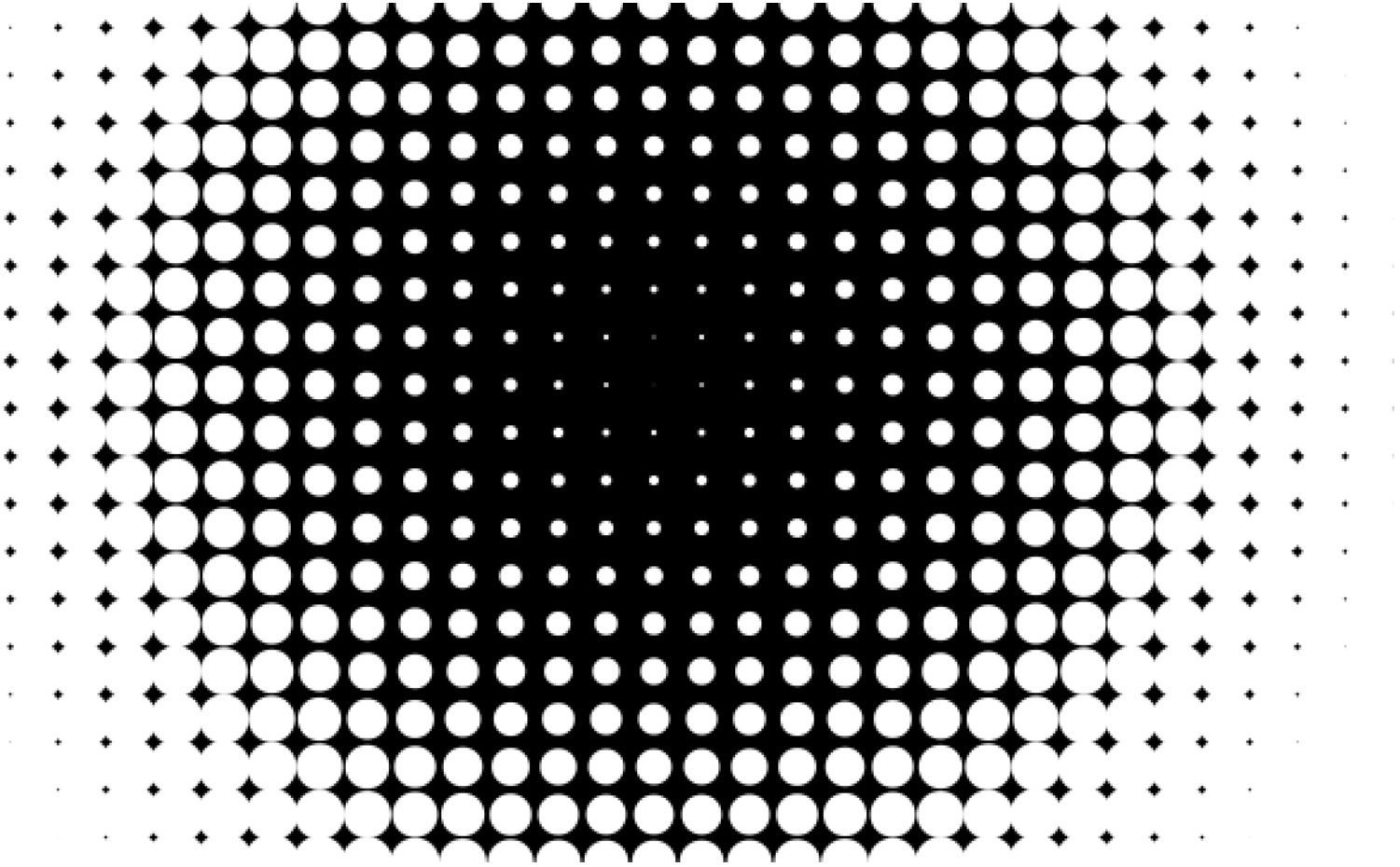
Outline of a Processing sketch

```
/* Comment describing sketch */
// Global variables
int x, y;
// Setup function is called once before everything
void setup() {
    // Do things that need to be done once
}
// Draw function is called once for every frame
void draw() {
    // Draw to canvas
}
// Event handler functions
// ...
// User-defined helper functions
// ...
```

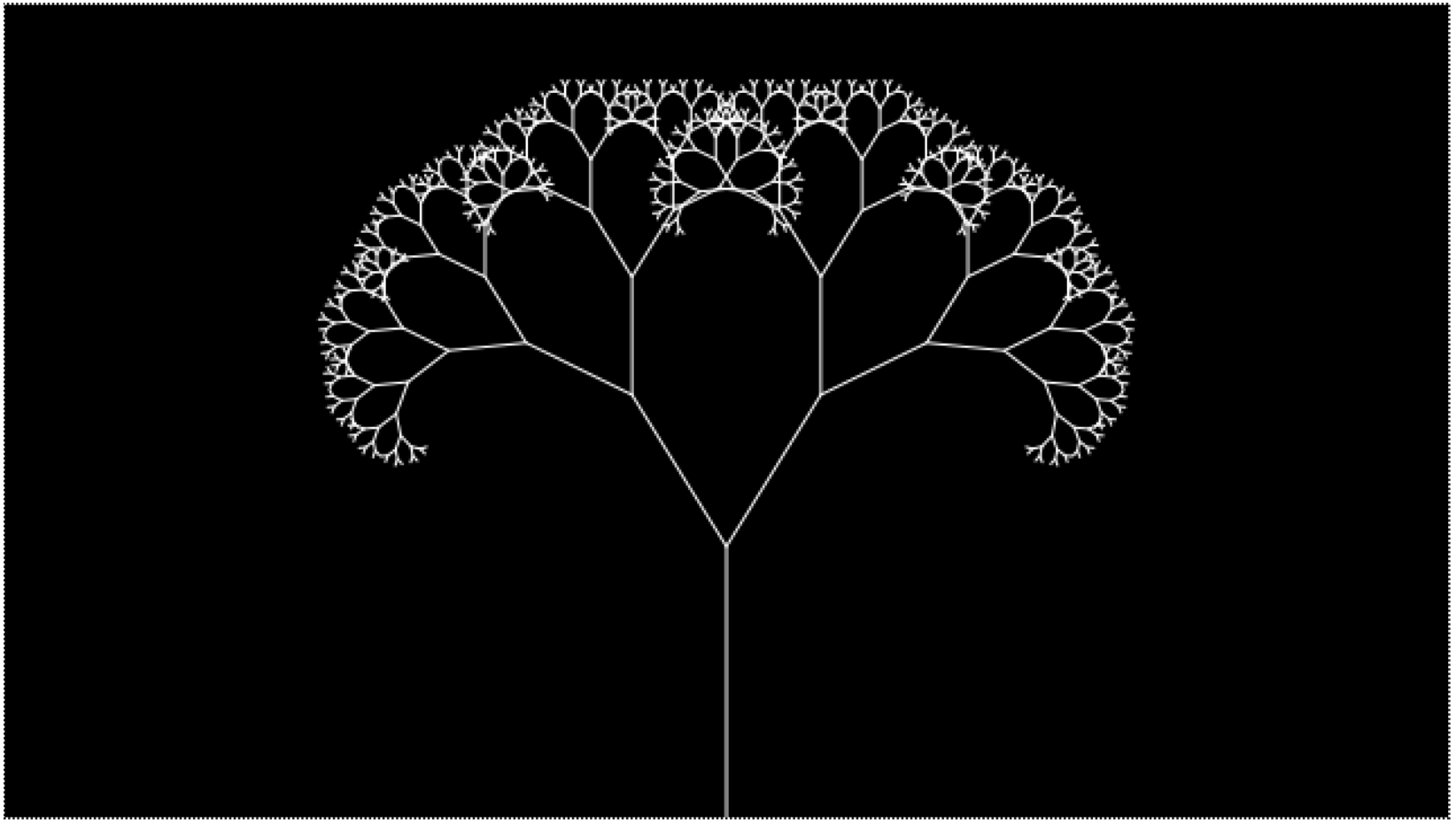
Read the code of demo sketches



Read the code of demo sketches

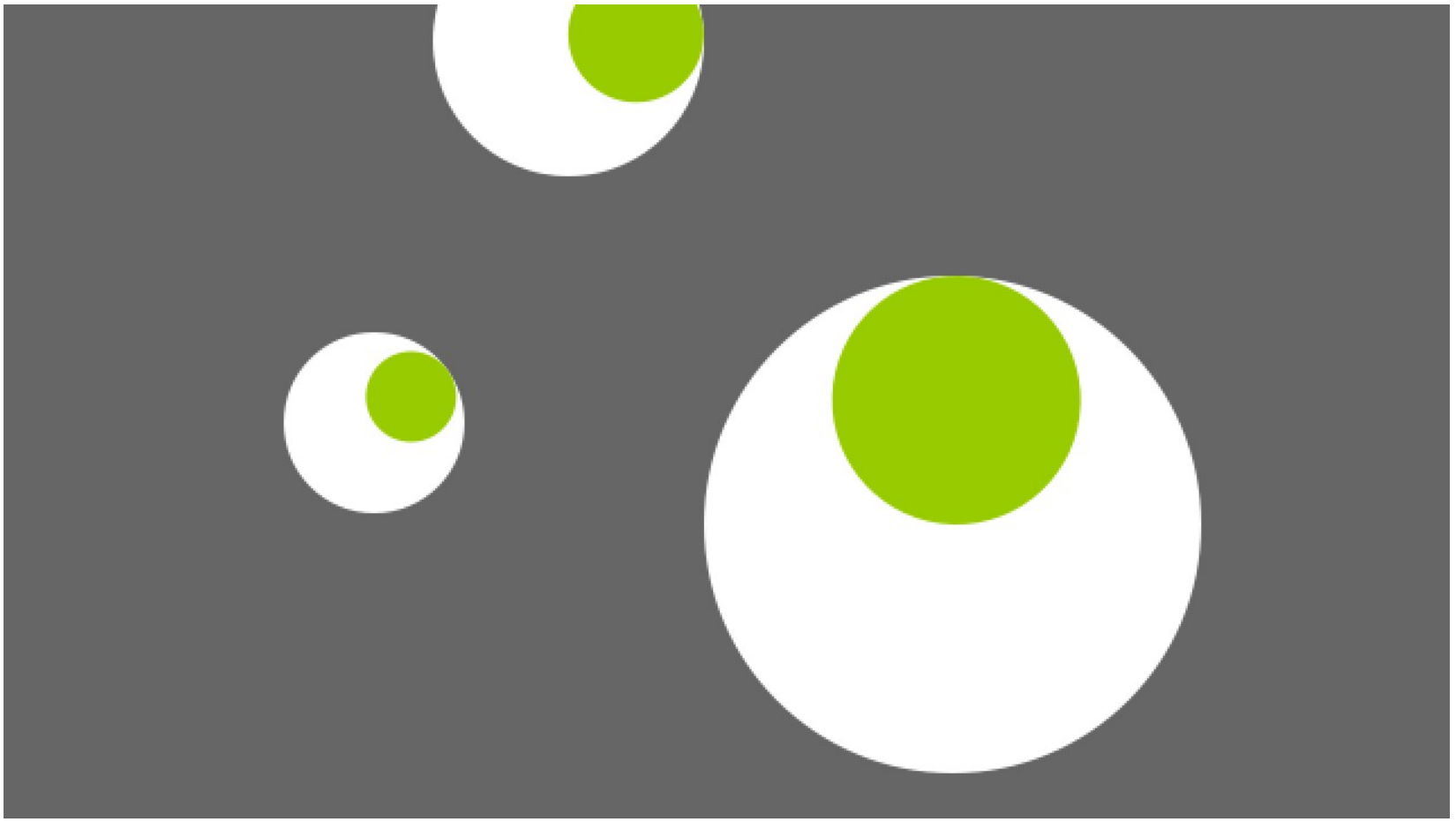


Read the code of demo sketches



Official Processing 2D transformations tutorial

Read the code of demo sketches



Official Processing object tutorial

Interactivity: mouse variables

- Processing populates some variables automatically describing the current state of the mouse
- `float mouseX`: current x coordinate of mouse
- `float mouseY`: current y coordinate of mouse
- `boolean mousePressed`: true if a mouse button is currently pressed
- `int mouseButton`: the identity of the mouse button that is pressed (`LEFT, RIGHT, CENTER`)

Interactivity: keyboard variables

- Processing populates some variables automatically describing the current state of the keyboard
- `boolean keyPressed`: true if any key is currently pressed
- `int key`: number specifying which key is pressed
- `int keyCode`: number specifying which special key is pressed (if `key == CODED`)

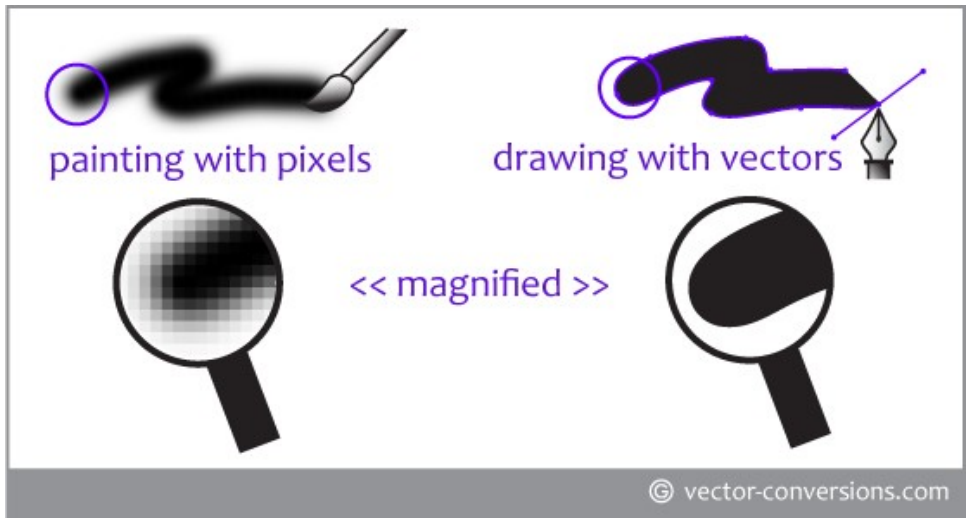
Interactivity: mouse events

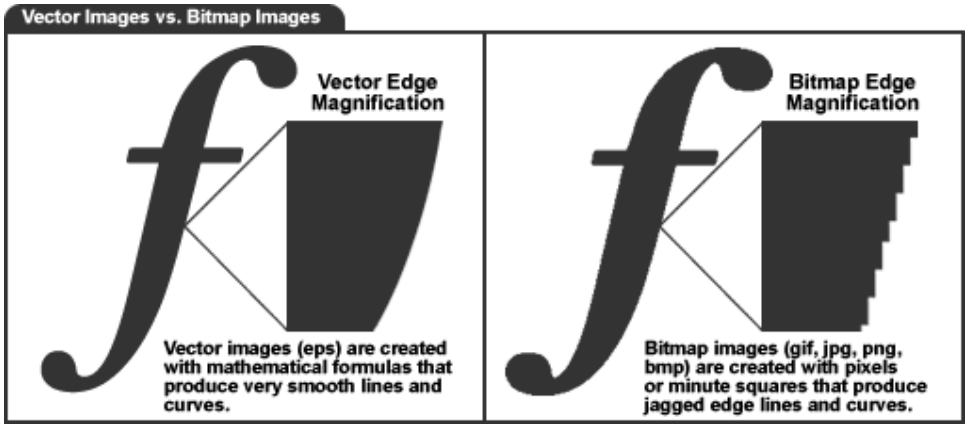
- Processing will call certain functions with special names when particular events happen
- `mousePressed()`: a mouse button is pressed
- `mouseReleased()`: a mouse button is released
- `mouseMoved()`: the mouse is moved without a button being pressed
- `mouseDragged()`: the mouse is moved while a button is pressed

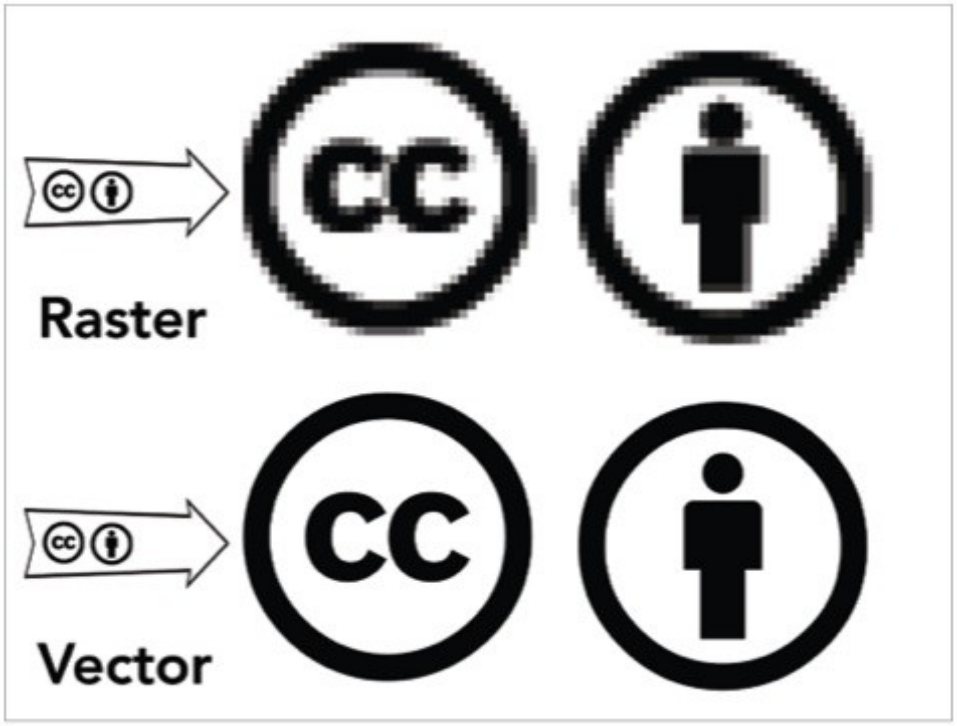
Interactivity: keyboard event listeners

- Processing treats the keyboard as a set of buttons
- Use these functions in combination with the keyboard variables we just learned about
- `keyPressed()`: any key (including special keys) is pressed down
- `keyReleased()`: any key is released
- `keyTyped()`: any key is first pressed then released
- Listener functions are called less often than `draw()`
  - It might be more efficient to update certain things when they change as opposed to recalculating the same value every frame

Graphics: vector vs bitmap/raster

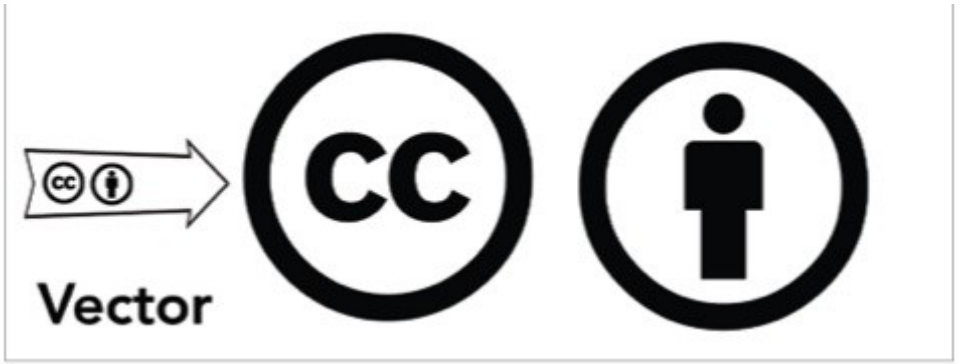






Animation with vector graphics

- Provide set of instructions for drawing scene
- Complex scenes take longer to draw
- Scenes still look sharp after being transformed
  - Zooming, rotating, stretching
- Better for drawings, non-realistic scenes
- For animation, move individual lines / shapes



Animation with bitmap graphics

- Provide the color at each pint on a grid
- All scenes take the same amount of time to draw
- Artifacts are visible after large transformations
  - Zooming, rotating, stretching
- Better for realistic scenes, photographs
- For animation, switch photos quickly