Basic Input and Output

CSE 120 Winter 2020

Instructor: Sam Wolfson

Teaching Assistants: Yae Kubota, Erika Wolfe, Eunia Lee

Avast's Free Antivirus Harvests All Your Clicks, Sells Them to Third-Parties

"As we've learned time and time again, "free" things on the internet are almost never truly free. If you're not paying with money, you're probably paying with your data. That's the case with the free antivirus products from Avast, which harvest browsing history for sale to major corporations. Despite claims that its data is fully anonymized, an investigation by our sister site PCMag and Motherboard shows how easy it is to unmask individual users.

"Avast, which offers antivirus products under its own brand as well as AVG, has traditionally gotten high marks for its malware blocking prowess. When setting up the company's free AV suite, users are asked to opt into data collection. Many do so after being assured all the data is anonymized and aggregated to protect their identities. However, Avast is collecting much more granular data than anyone expected, and that puts your privacy at risk."

- https://www.extremetech.com/internet/305344-avasts-free-antivirus-harvests-all-your-clicks-sells-them-to-third-parties

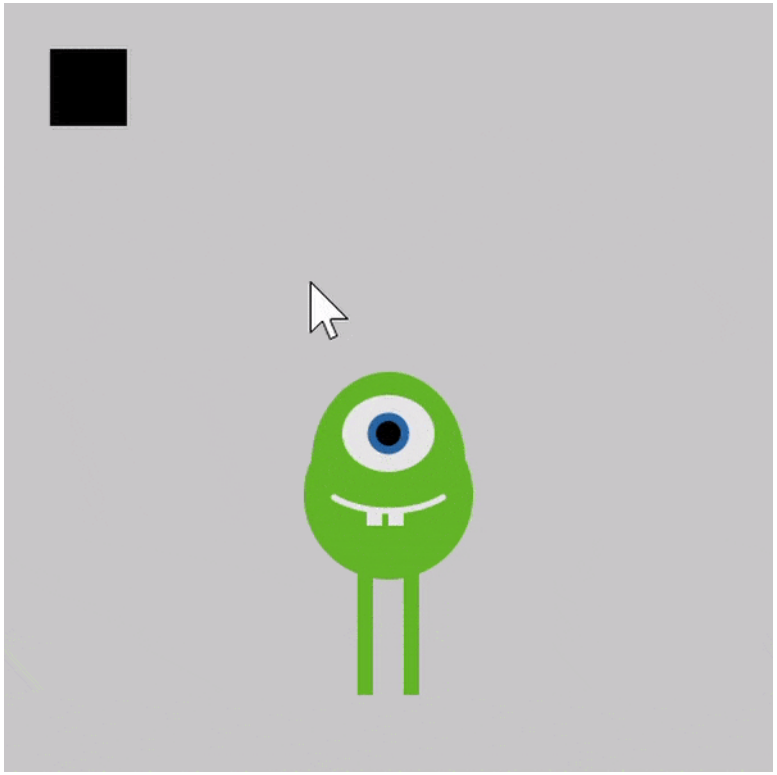
Administrivia

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

- "Big Idea" this week: Digital Distribution
- Quiz 2 this Friday
  - Topics posted on course website
  - **New:** memorization of *short* code snippets
- Upcoming: Creativity Project

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

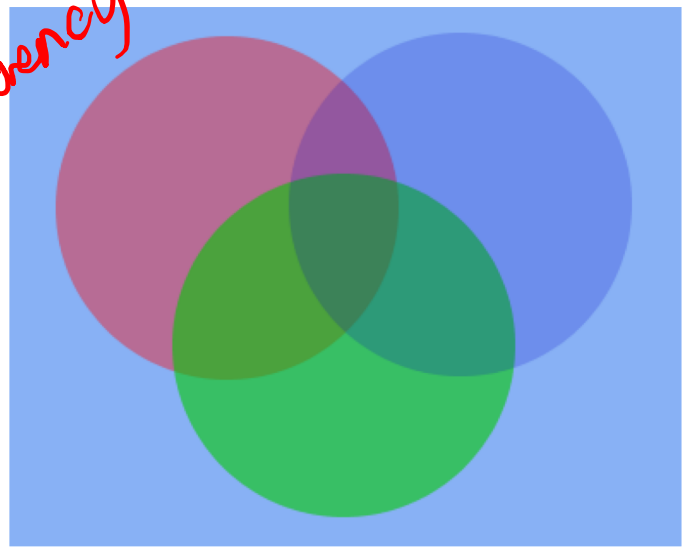
- **Other Useful Processing Tools**
- User Input and Output
  - Mouse (input)
  - Keyboard (input)
  - Text (output)

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)

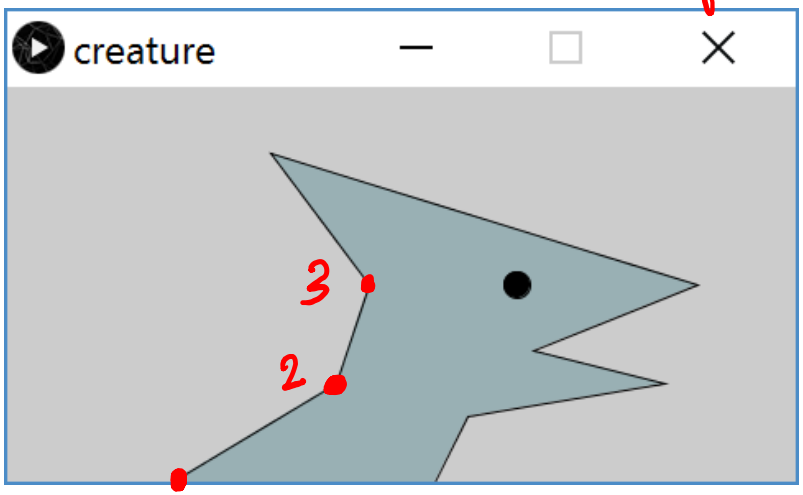
```
size(400, 320);
noStroke();
background(136, 177, 245);

fill(255, 0, 0, 100);
ellipse(132, 120, 200, 200);

fill(0, 200, 0, 150);
ellipse(200, 200, 200, 200);

fill(0, 0, 200, 50);
ellipse(268, 118, 200, 200);
```

transparency



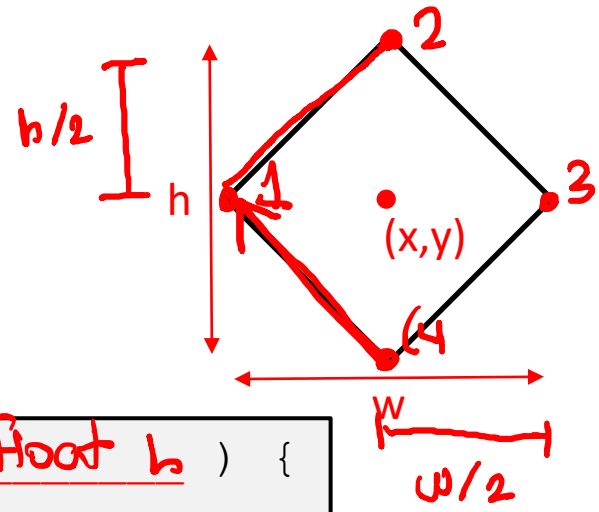
Custom Shapes

- Define vertices between `beginShape()` and `endShape()`
  - If planning to reuse, best to create in a separate function




```
creature
1  size(480,240);
2
3  fill(153, 176, 180);
4  beginShape();
5    vertex(100, 240);
6    vertex(200, 180);
7    vertex(220, 120);
8    vertex(160,  40);
9    vertex(420, 120);
10   vertex(320, 160);
11   vertex(400, 180);
12   vertex(280, 200);
13   vertex(260, 240);
14 endShape();
15
16 fill(0);
17 ellipse(310, 120, 16, 16);
```

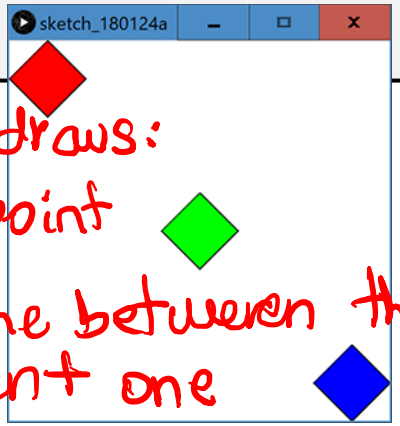
Functions Practice: Diamond



- Fill in the code to produce:

```
void diamond( float x, float y, float w, float h ) {
  beginShape();
    1) vertex( x-w/2, y );
    2) vertex( x, y-h/2 );
    3) vertex( x+w/2, y );
    4) vertex( x, y+h/2 );
    5) vertex( x-w/2, y );
  endShape();
}
```

```
void draw() {

}
```



vertex draws:
- a point
- a line between the prev vertex and the current one

Lecture Outline

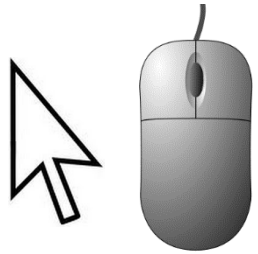
- Other Useful Processing Tricks
- **User Input and Output ***
  - Mouse
  - Keyboard
  - Text

* 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:
  - `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?

current frame

prev frame

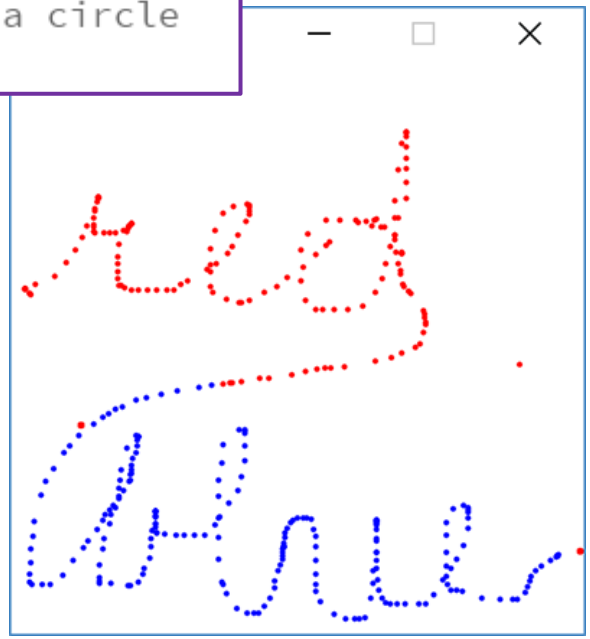
→ true / false (boolean)

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

don't use both

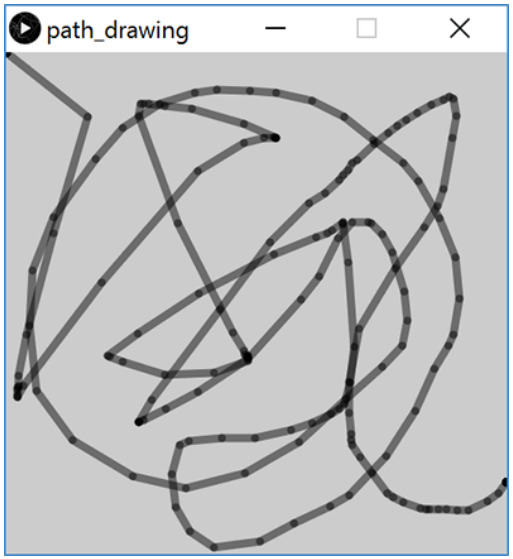
Example: Drawing Dots

```
7  void draw() {
8    if (mousePressed) {
9      fill(0, 0, 255);  // blue if mouse is pressed
10   }
11   if (!mousePressed) {
12     fill(255, 0, 0);  // red if mouse is not pressed
13   }
14   ellipse(mouseX, mouseY, 5, 5);  // draw a circle
15 }
```



Example: Path Drawing

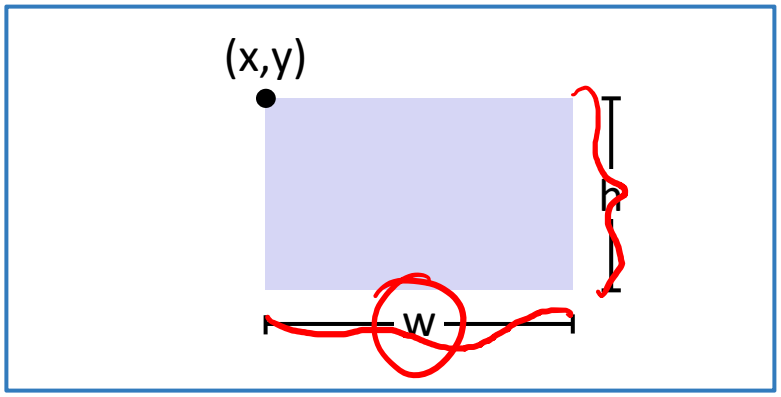
- We just wrote a *dot*-drawing program
- We can additionally use `pmouseX` and `pmouseY` to create a *path*-drawing program

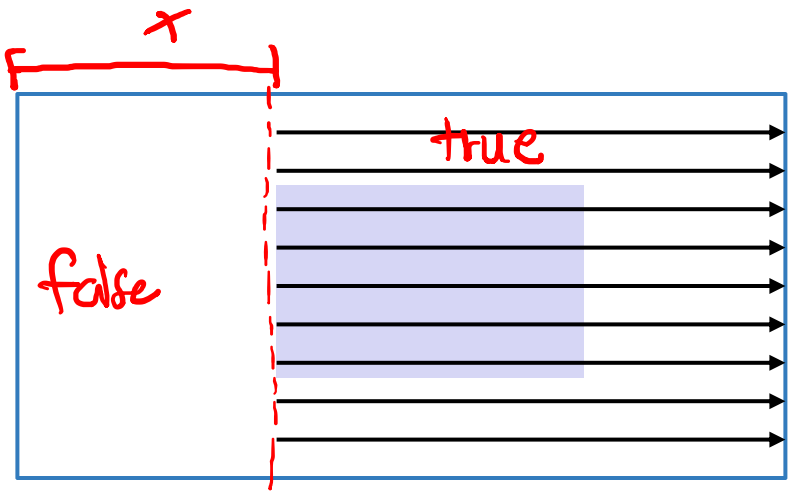



```
7  void setup() {
8    size(500,500);          // set drawing canvas size
9    strokeWeight(8);        // thicker lines
10   stroke(0,0,0, 120);     // black line with some transparency
11   frameRate(30);          // slow down the frame rate
12 }
13
14 void draw() {
15   line(mouseX, mouseY, pmouseX, pmouseY);
16 }
```

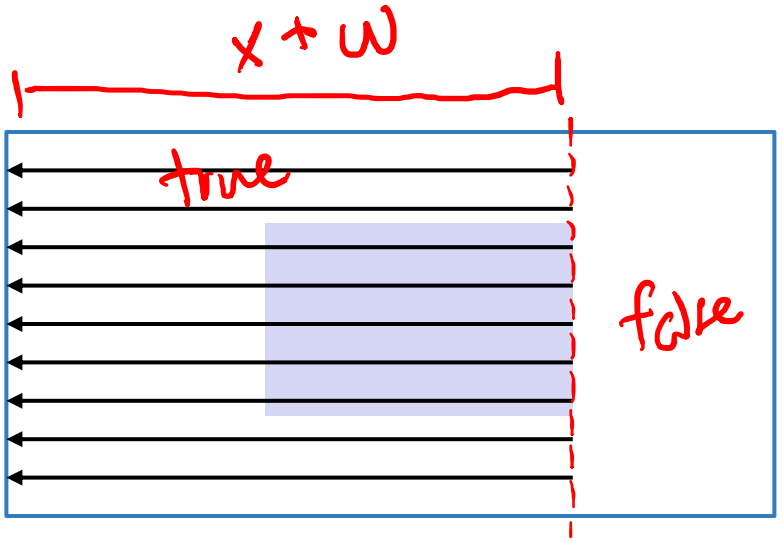
← ☆ drawing the path your mouse takes

Hovering Over a Rectangle





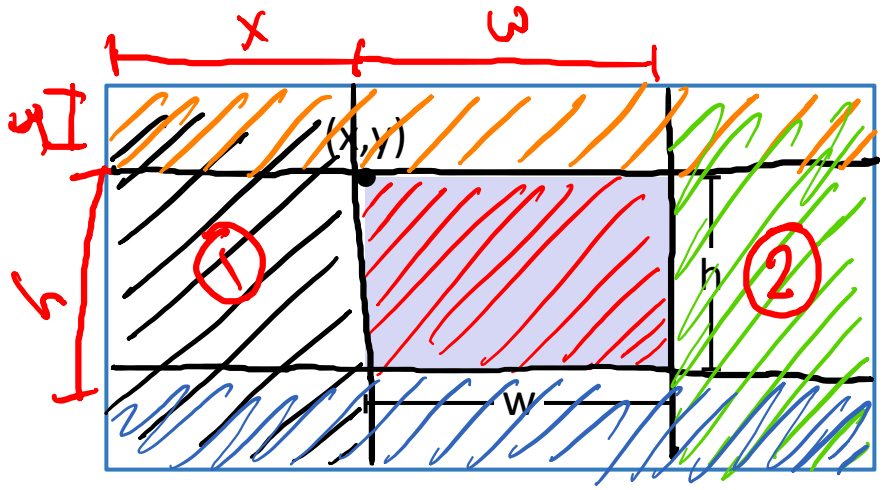

```
if(mouseX >= x)
```



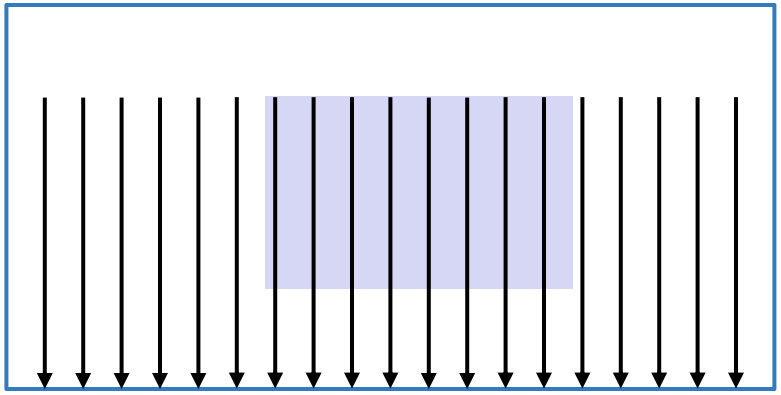

```
if(mouseX <= x + w)
```

Hovering Over a Rectangle

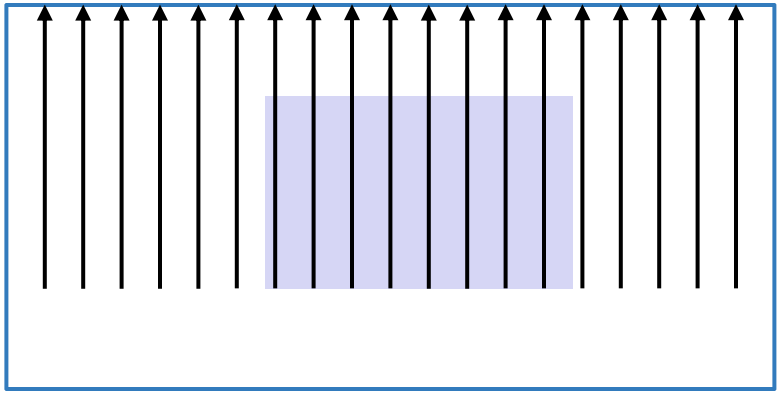
"&&" : "AND"



1) mouseX >= x &&
mousex <= x+w &&
mouseY >= Y &&
mouseY <= y+h

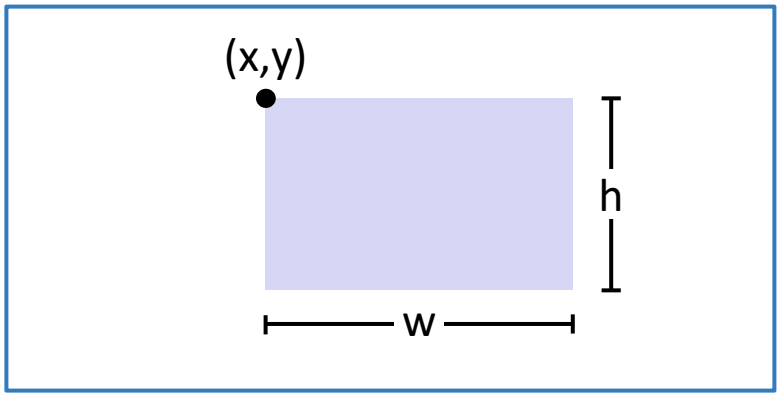


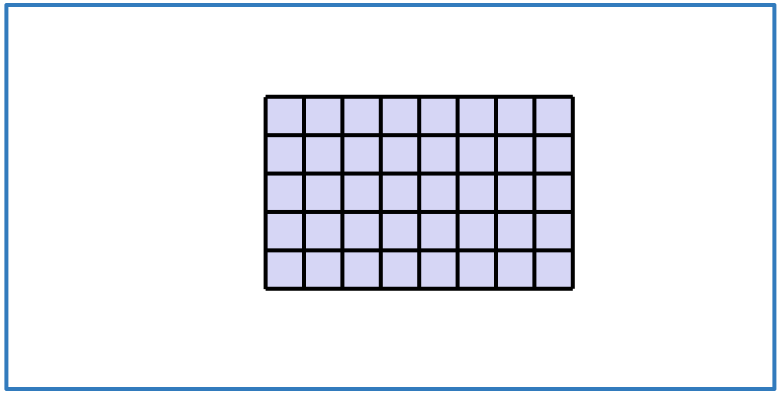
```
if(mouseY >= y)
```



```
if(mouseY <= y + h)
```

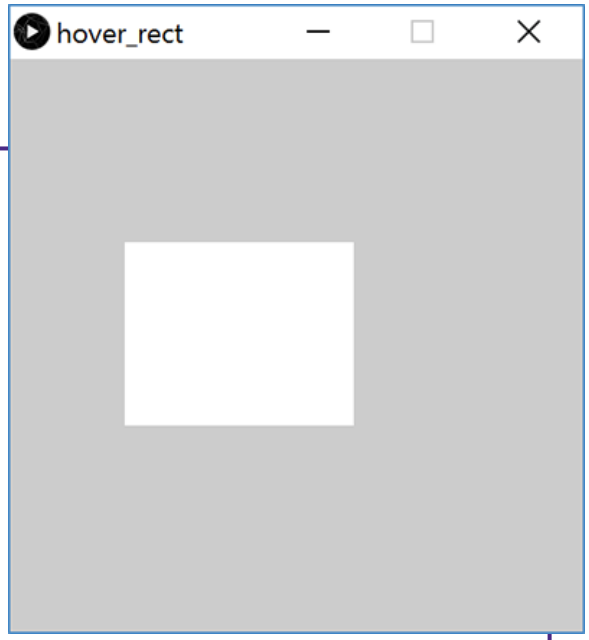
Hovering Over a Rectangle





```
if(  (mouseX >= x)       &&
     (mouseX <= x + w)   &&
     (mouseY >= y)       &&
     (mouseY <= y + h)   )
```

Hovering Over a Rectangle




```
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

  fill(255);  // white by default

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

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

The Keyboard

single quotes

if (key == 'a')

- System variables:
  - `key` – stores the ASCII value of the last key press — char
  - `keyCode` – stores codes for non-ASCII keys (*e.g.* UP, LEFT)
  - `keyPressed` – is any key currently being pressed? — boolean
- Built-in functions:
  - `keyPressed()` – called every time a key is pressed

use one or the other!

- 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?

```
1  int position = 0;
2
3  void setup() {
4    size(400, 100);
5    noStroke();
6    background(0);
7    fill(0);
8  }
9
10 void draw() {
11   ellipse(position, 40, 40, 40);
12 }
13
14 void keyPressed() {
15   if(key == 'g'){
16     fill(0, 255, 0);
17   }
18
19   if(key == 'y') {
20     fill(255, 255, 0);
21   }
22
23   if(key == 'm') {
24     fill(255, 0, 255);
25   }
26
27   position = position + 50; // position+=50;
28 }
```

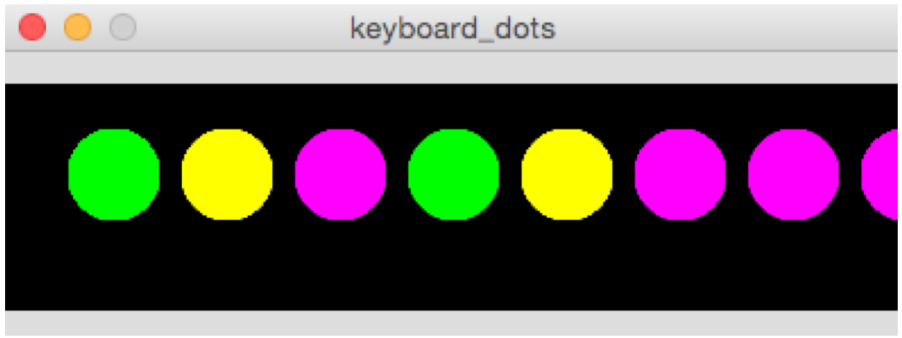
initially black

ellipse drawn @ (pos, 40)

any key pressed

always executed

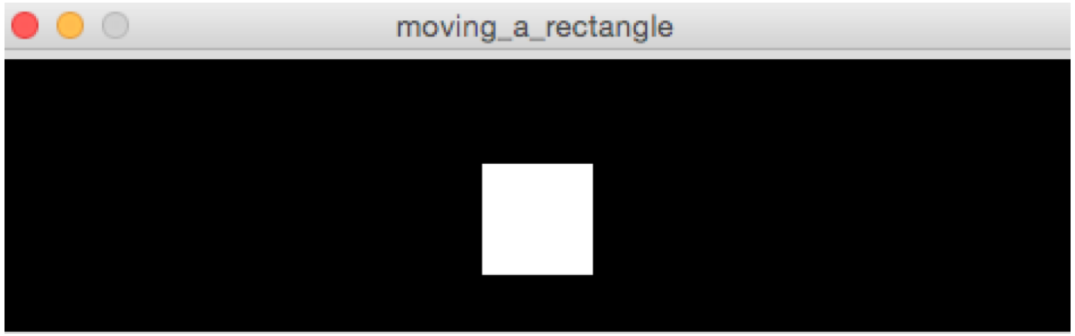
Example: Keyboard Dots



keyboard_dots

```
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

```
11  if(keyPressed) {
12    if(key == CODED) {
13      if(keyCode == LEFT) {
14        x = x - 1;
15      }
```

Example: Moving a Rectangle

moving_a_rectangle

```
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

String: "hello"
char: 'h'

- `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

display_letters

```
1  void setup() {
2    size(120, 120);
3    textSize(64);
4    textAlign(CENTER);
5  }
6
7  void draw() {
8    background(0);
9    text(key, 60, 80);
10 }
```

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 (2/4)
  - Actual programs due next Friday (2/7)

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