Basic Input and Output

CSE 120 Spring 2017

Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

How airlines like United choose who to kick off a flight

On Sunday law-enforcement officers in Chicago forcibly removed a passenger from a United Airlines flight… [T]he airline needed to bump four passengers from the flight to make room for pilots and crew it needed to transport down to Louisville to operate flights later that evening.

Based on United Airlines' **contract of carriage**, passengers with disabilities and unaccompanied minors are the least likely to bumped from a flight. For everyone else, the contract says that the airline's decision is based on a passenger's frequent-flyer status, the layout of his or her itinerary (whether the passenger has a connecting flight), the fare class of the ticket, and the time he or she checked in to the flight.

- http://www.businessinsider.com/how-airline-choose-who-kick-off-flight-united-american-delta-2017-4

Administrivia

- Assignments:
  - Animal Functions due today (4/12)
  - Reading Check 3 due tomorrow *before lab* (4/13)
  - Jumping Monster! due Saturday (4/15)
    - ↳ harder assignment, will take time!
- "Big Idea" this week: Algorithms

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

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)

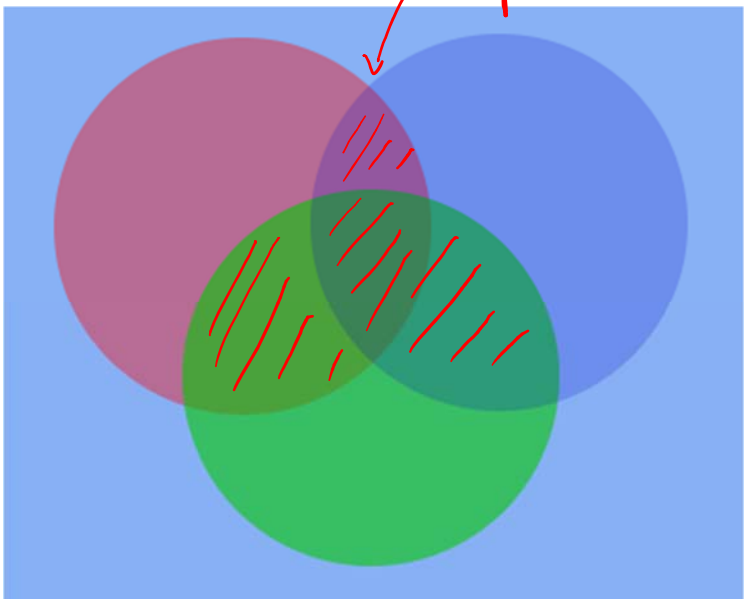
```
size(400, 320);
noStroke();
background(136, 177, 245);

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);
```

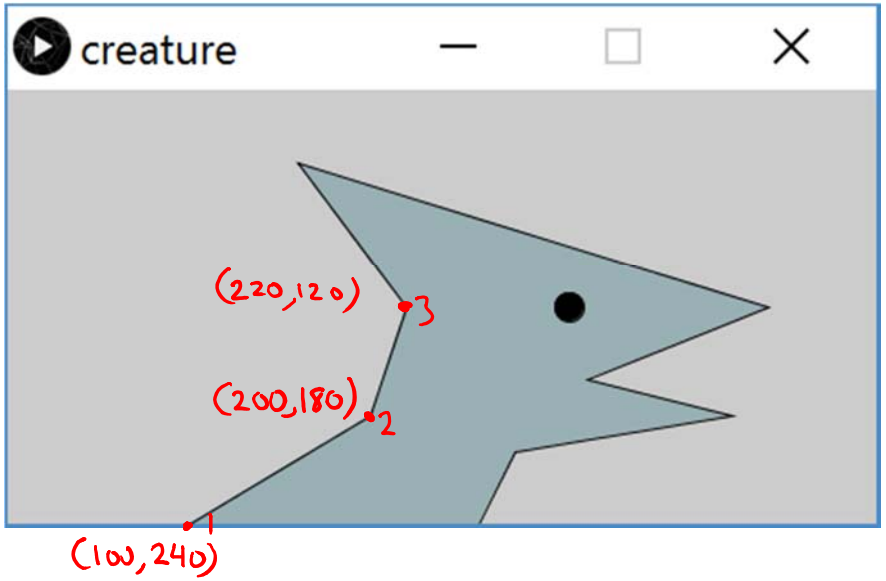
slightly transparent



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.




```
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);
```

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 there are *per second*
  - So 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

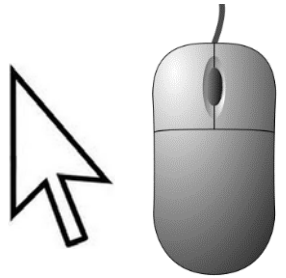
- Control and track how frequently `draw()` runs
  - Each time `draw()` runs, it is called a new *frame*
- `noLoop()` stops `draw()` from being continuously executed
  - Can restart using `loop()`

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

```
if (mousePressed) {
  // statements
}
```

vs.

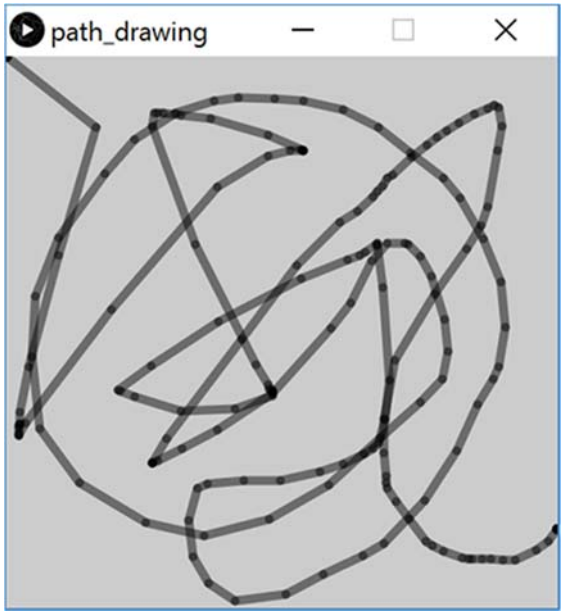
```
void mousePressed() {
  // statements
}
```

probably should only use one or the other

- 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
- 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 }
```

Example: Frame Dots

- Slow down to 1 frame per second and have a new dot appear on each frame
  - Number of dots on screen will equal current frame count
  - Reminder: `frameRate()`, `frameCount`
  - Calculate position using division and modulus

0/3 = 0 remainder 0 , 3/3 = 1 remainder 0
1/3 = 0 remainder 1 , 4/3 = 1 remainder 1
2/3 = 0 remainder 2 , 5/3 = 1 remainder 2

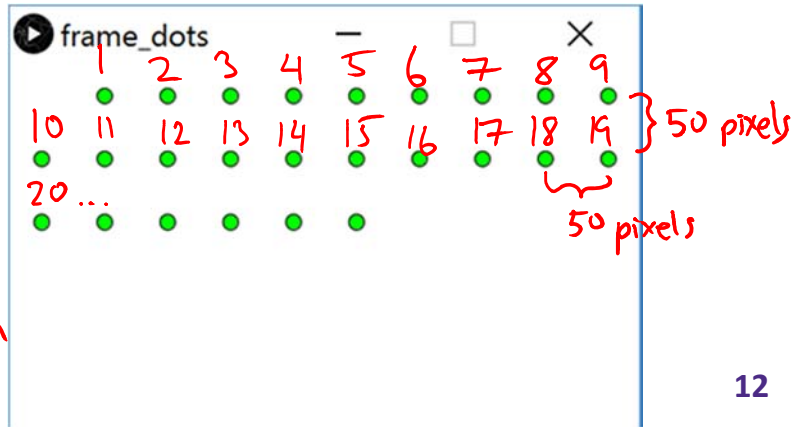
(div = 1, mod = 0, 1, 2)

frameCount	x	y
1	50	50
2	100	50
⋮	⋮	⋮
9	450	50
10	0	100
⋮	⋮	⋮
19	450	100
20	0	150

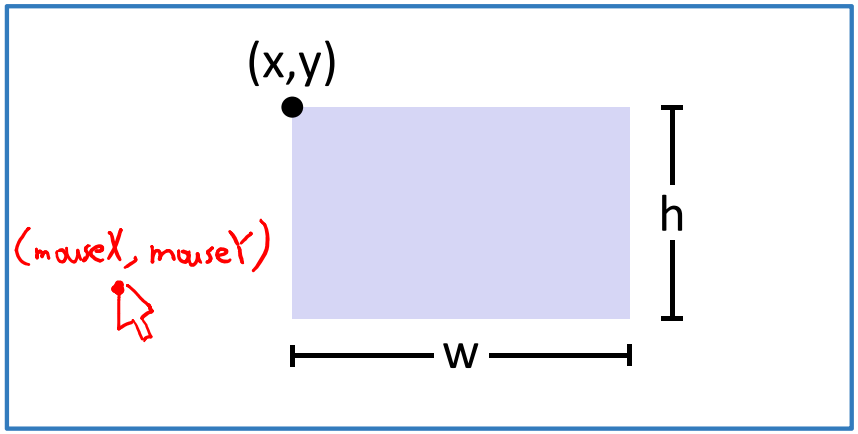
- Control the animation with the mouse
  - Pause while mouse is pressed
  - Reminder: `loop()`, `noLoop()`

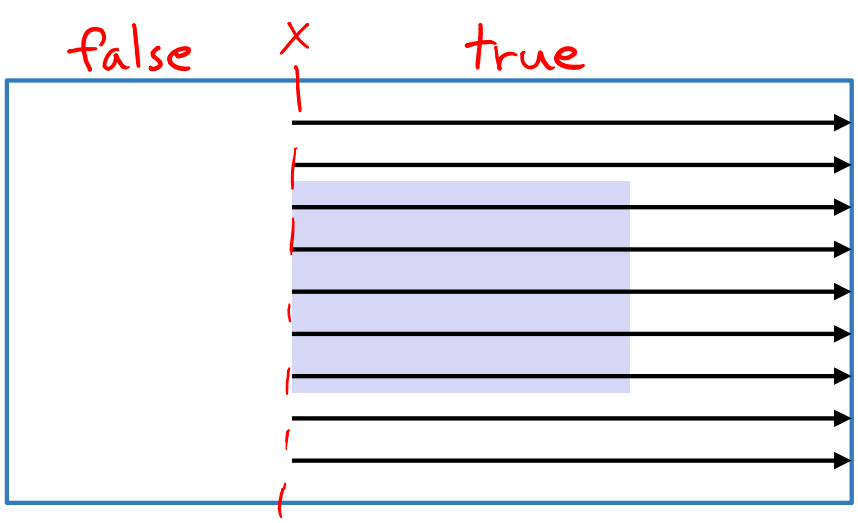
```
x = (50 * frameCount) % 500;
y = 50 * (1 + (50 * frameCount / 500));
```

need to round down

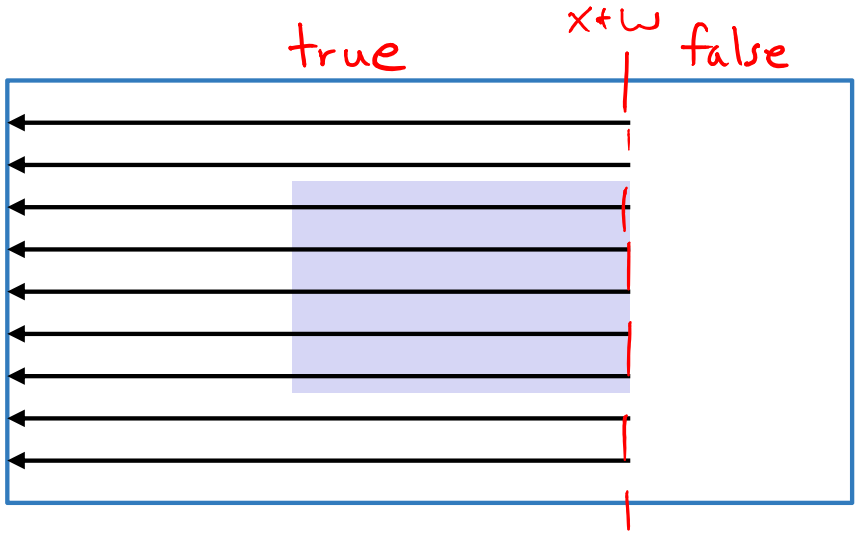


Hovering Over a Rectangle



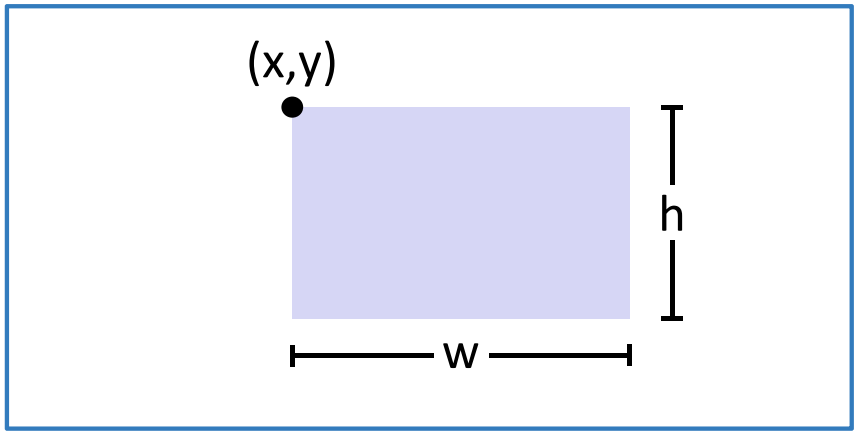


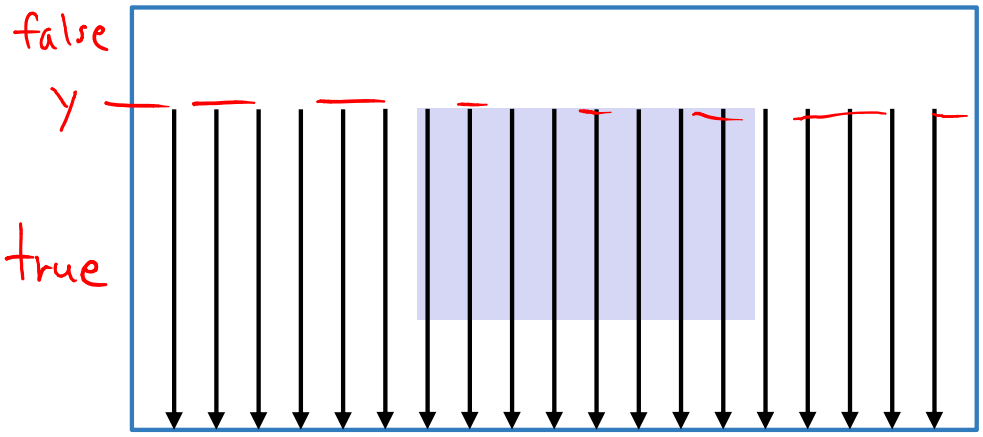

```
if(mouseX >= x)
```



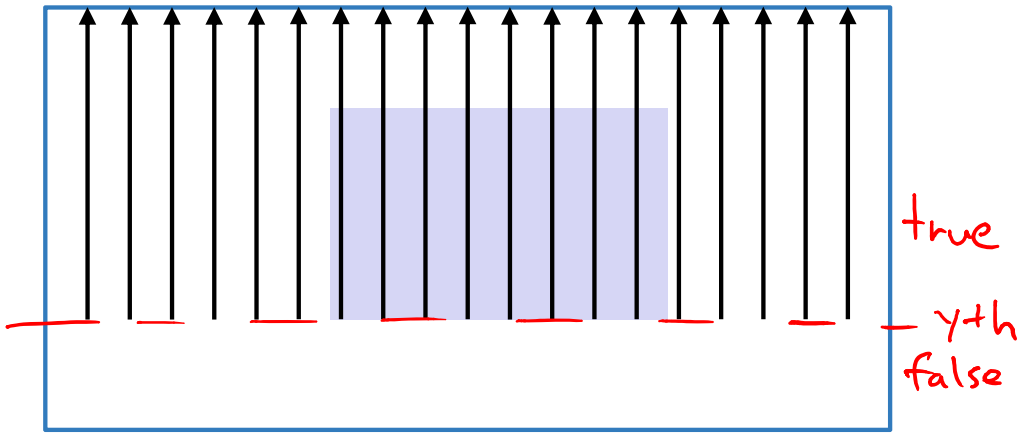

```
if(mouseX <= x + w)
```

Hovering Over a Rectangle



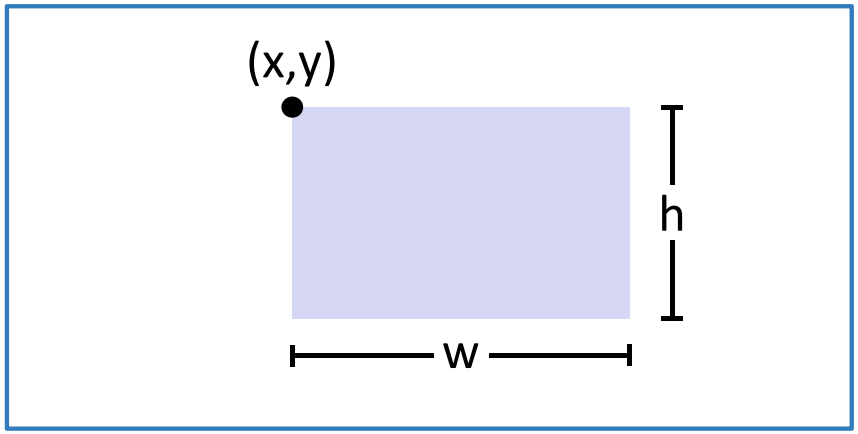


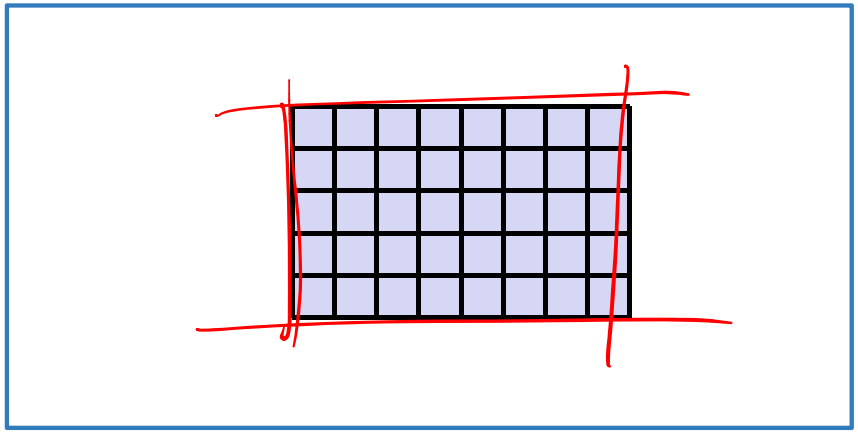

```
if(mouseY >= y)
```




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

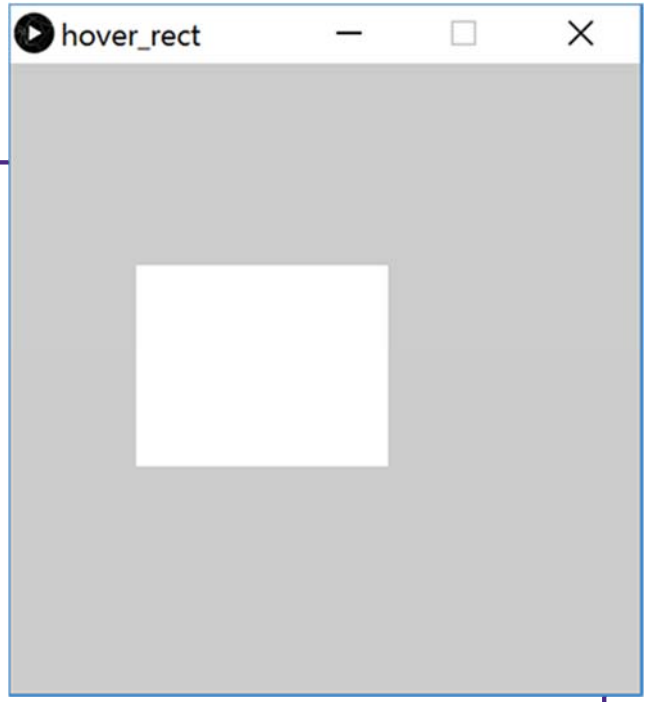
Hovering Over a Rectangle





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

Hovering Over a Rectangle




```
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

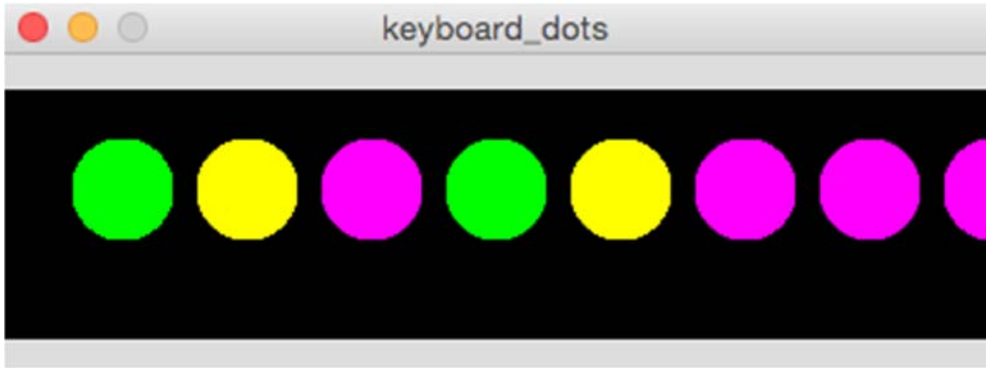
- System variables:
  - **key** – stores the ASCII value of the last key press
  - **keyCode** – stores codes for non-ASCII keys (*e.g.* UP, LEFT)
  - **keyPressed** – is any key currently being pressed?

again, recommended to only use one or the other for each program

- 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: 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;
}
```

char

draw next circle at new position

Text Output

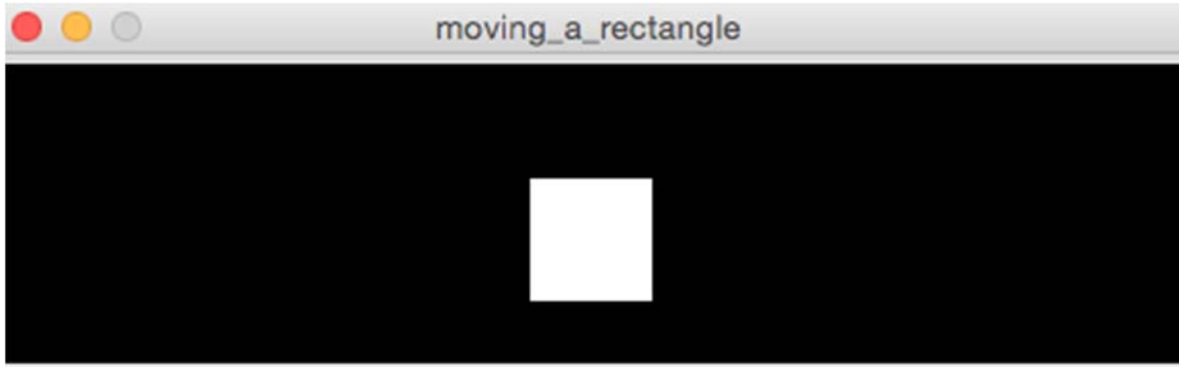
- `println(yourText);`
  - Prints `yourText` to the *console*, which is the black area below your Processing code
  - Useful for debugging, particularly your portfolio
- `text(yourText, x, y);`
  - Prints `yourText` on the drawing canvas, starting with the *bottom-left* corner at coordinate `(x,y)`
  - Change the size of your text using `textSize(size);`
- `yourText` should be between *double* quotes
  - We will talk 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 }
```

Example: Moving a Rectangle



- **Reminder:** 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

```
1  int x = 215;
2
3  void setup() {
4    size(480, 120);
5  }
6
7  void draw() {
8    background(0);
9    rect(x, 45, 50, 50);
10
11   if(keyPressed) {
12     if(key == CODED) {
13       if(keyCode == LEFT) {
14         x = x - 1;
15       }
16
17       if(keyCode == RIGHT) {
18         x = x + 1;
19       }
20     }
21   }
22 }
```

Looking Forward

- Next week is the Creativity Assignment
  - In pairs, you will be asked to create and submit TWO Processing projects *of your choice*
  - The point is to use the tools available to you to make something fun and creative!
  - Planning document due Tuesday (4/18)
  - Actual programs due next, next Monday (4/24)

- Portfolio Update 1 is due Tuesday (4/18)
  - Building a Robot, Logo Design, Lego Family, Animal Functions, Jumping Monster
  - Ask your TAs for assistance!