Interactive Animations

Great for games

Using `setup` and `draw`

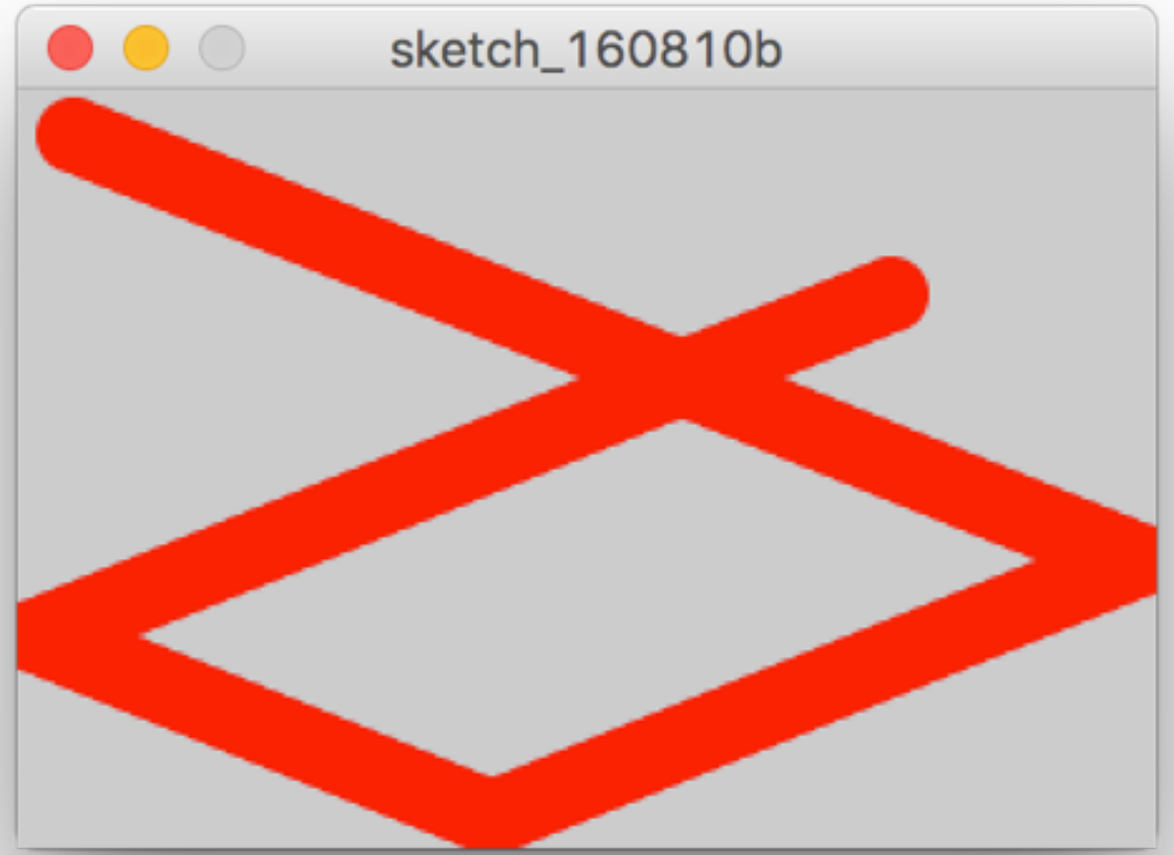
- To do animations, you must use these two methods:
  - `void setup()` is called only once, and does any initialization you need
  - `void draw()` is called 60 times a second
    - This can be changed by calling `frameRate(`*timesPerSecond*`)`
    - `draw` is called automatically; you should never call it yourself
- To animate, `draw` should do something different each time it is called
  - This requires you to use *global variables*--variables declared outside any of your methods

Bouncing ball 1

```
float x = 10; // global variables
float y = 10;
float dx = 5;
float dy = 2;

void setup() { // called once
  size(300, 200);
  fill(255, 0, 0);
  noStroke();
}

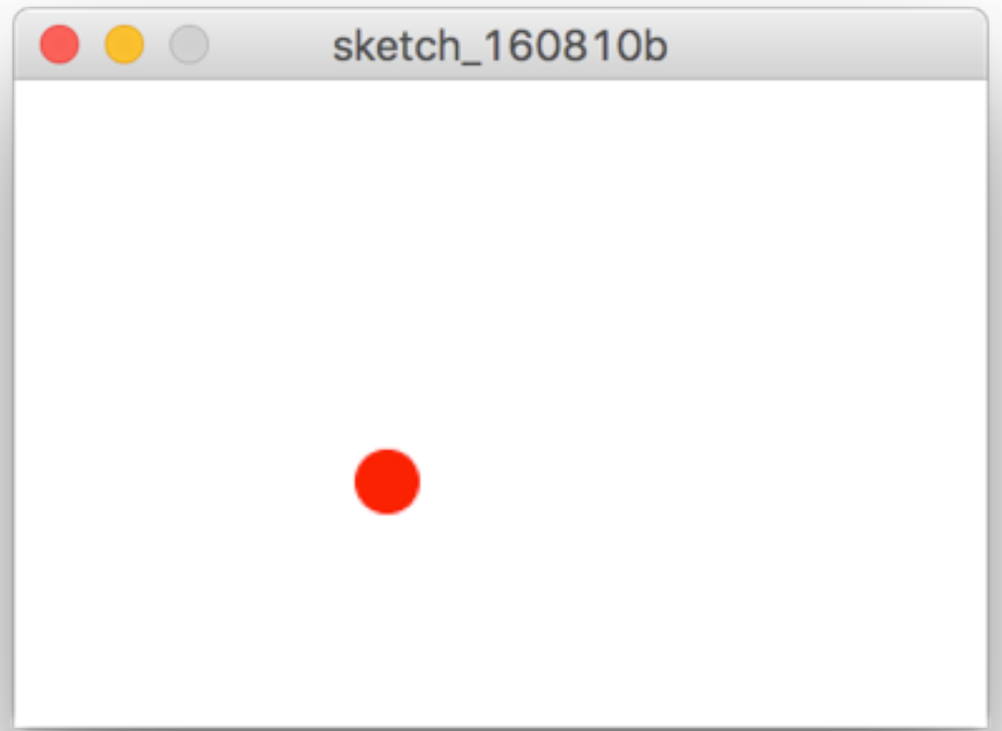
void draw() { // called 60/sec
  x += dx; // changing where the
  y += dy; // ellipse is drawn
  ellipse(x, y, 20, 20);
  if (x < 10 || x > 290) {
    dx = -dx;
  }
  if (y < 10 || y > 190) {
    dy = -dy;
  }
}
```



Bouncing ball 2

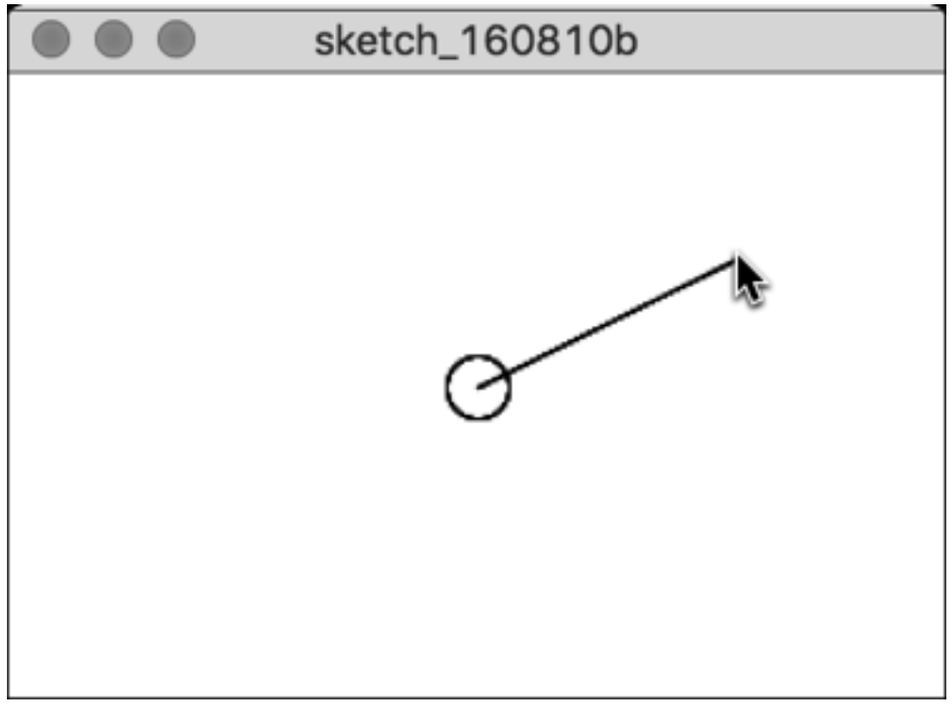
- What happened?
- Drawing the ball in a new place did not erase the previous drawing
- Solution: In `draw`, erase *everything* and draw it all again each time
-

```
void draw() {
   x += dx;
   y += dy;
   background(255);
   ellipse(x, y, 20, 20);
   if (x < 10 || x > 290) {
      dx = -dx;
   }
   if (y < 10 || y > 190) {
      dy = -dy;
   }
}
```



The mouse

- The built-in variables `mouseX` and `mouseY` can be used to find out where the mouse is in the window
- The built-in variable `mousePressed` is `true` if the mouse button is down



-
```
void setup() {
  size(300, 200);
  frameRate(0.2);
}

void draw() {
  background(255);
  if (mousePressed) {
    ellipse(width / 2, height / 2, 20, 20);
    line(width / 2, height / 2, mouseX, mouseY);
  }
}
```

Detecting mouse motion

- Just as the built-in variables `mouseX` and `mouseY` can be used to find out where the mouse is *now*, the variables `pmouseX` and `pmouseY` can be used to find out where the mouse was during the *previous* call to `draw`

- The method `dist(`*x1*`, `*y1*`, `*x2*`, `*y2*`)` computes the distance between two points

- `dist(mouseX, mouseY, pmouseX, pmouseY)` is a simple measure of how fast the mouse is being moved

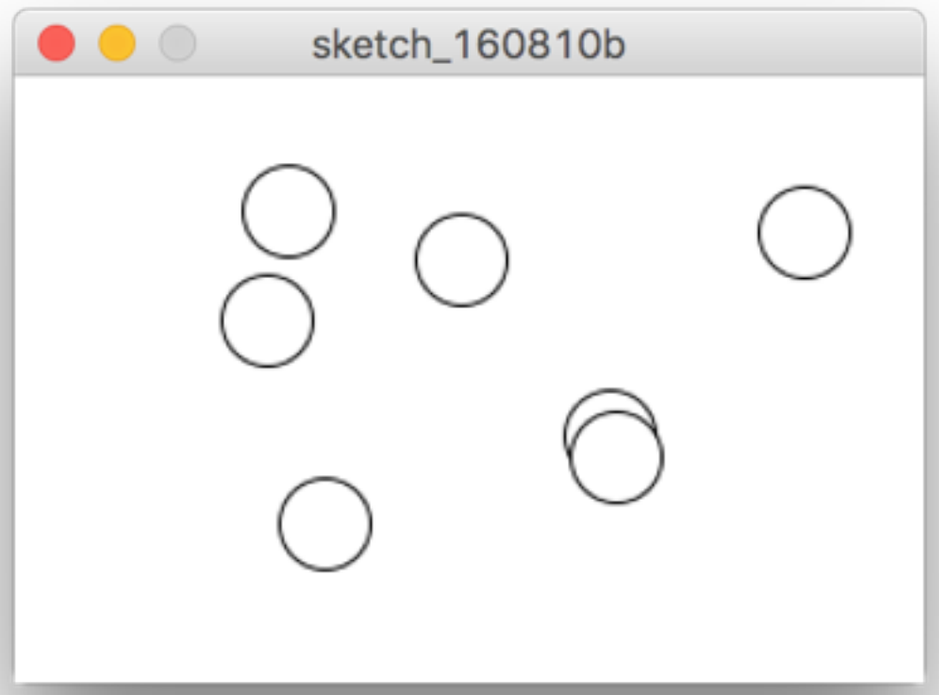
The `mousePressed` method

- In an earlier slide we saw the use of the `mousePressed` built-in *variable*
- There is also a void `mousePressed()` built-in *method*, which will be called each time any mouse button is clicked
- You must also have a `draw()` method for this to work, even if it doesn't do anything

```
void setup() {
  size(300, 200);
  background(255);
}

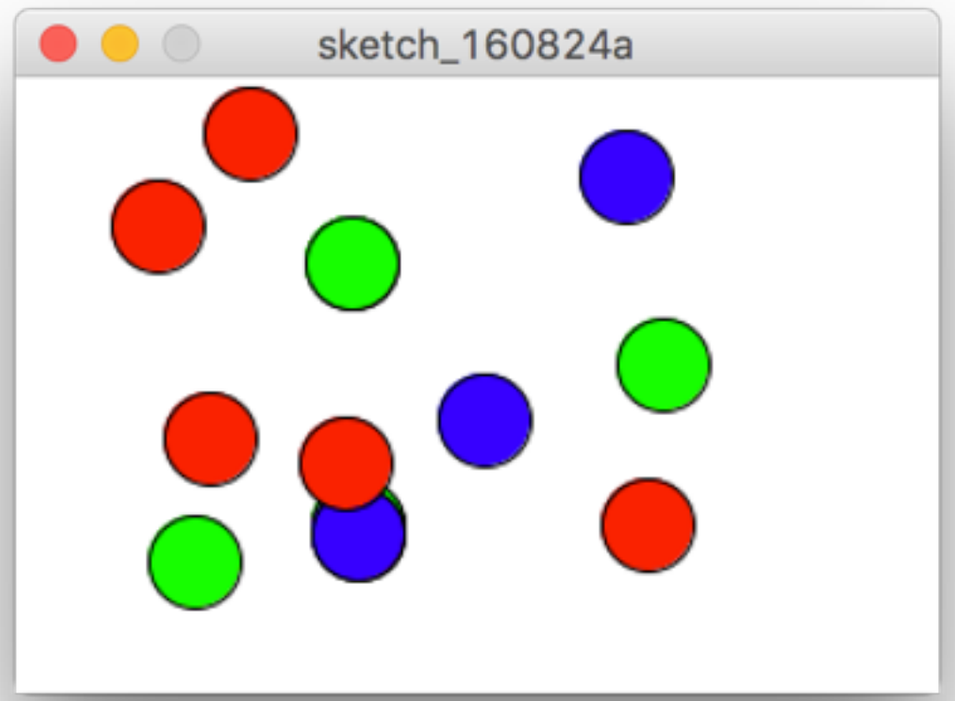
void draw() {
}

void mousePressed() {
  ellipse(mouseX, mouseY, 30, 30);
}
```



The `mouseButton` variable

- The `mousePressed` built-in method and variable tell us *some* mouse button was clicked
- The `mouseButton` variable tells us *which* button it was




```
void setup() {
  size(300, 200);
  background(255);
}

void draw() {
}

void mousePressed() {
  if (mouseButton == LEFT) {
    fill(255, 0, 0);
  } else if (mouseButton == CENTER)
{
    fill(0, 255, 0);
  } else if (mouseButton == RIGHT)
{
    fill(0, 0, 255);
  }
  ellipse(mouseX, mouseY, 30, 30);
}
```

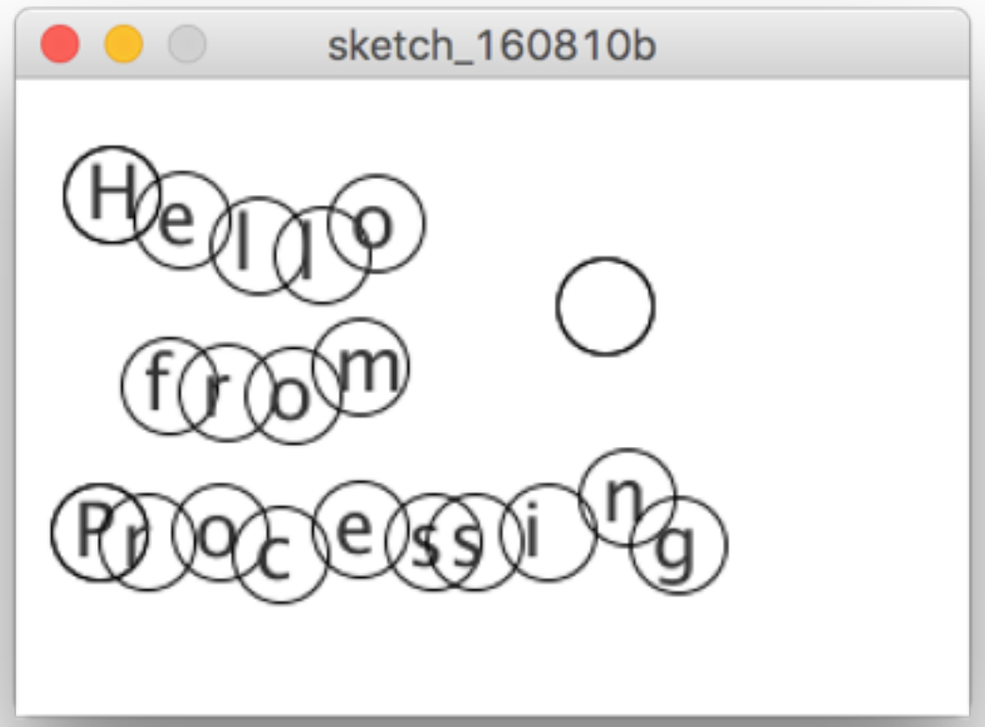
Control of looping

- By default, the `draw` method is called 60 times a second
  - You can change this by calling `frameRate(`*nPerSec*`)`
- Calling `noLoop()` will cause `draw` to be called once only
- Calling `loop()` will resume calling `draw` at the specified frame rate
- You can turn off looping with `noLoop(),` then call `redraw()` each time you want `draw` to be called
  - For example, call `redraw()` when the mouse is clicked
  - Again, never call `draw` yourself!

The `keyPressed` method

- The `keyPressed()` built-in *method* will be called each time a keyboard key is pressed
- The key value will be in the `key` variable as a `char`
  - It will hold this value until some other key is pressed
- The `text` method will accept either a `char` or a `String`
- You must also have a `draw()` method for this to work, even if it's empty

```
void setup() {
  size(300, 200);
  background(255);
  textSize(24);
}
void draw() {
}
void keyPressed() {
  noFill();
  ellipse(mouseX, mouseY, 30, 30);
  fill(50);
  text(key, mouseX - 8, mouseY+ 8);
}
```



Recognizing which key is pressed

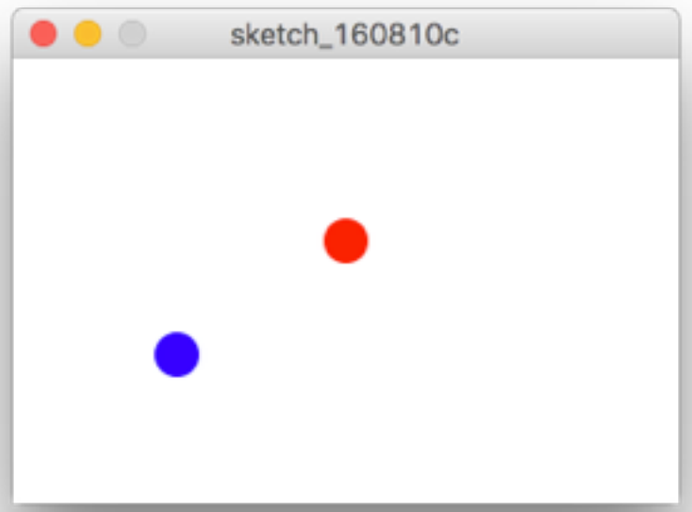
- Most keyboard characters come in as `char` values in the built-in `key` variable, so you can say for example,
  `if (key == 'a') {...}`
  - Remember that `char` literals must be in single quotes, like `'a'`
- The `key` variable can also be compared to any of `BACKSPACE`, `TAB`, `ENTER`, `RETURN`, `ESC`, and `DELETE`
- To check for arrow keys, compare `keyCode` (*not* `key`) to any of `LEFT`, `RIGHT`, `UP`, or `DOWN`
- Also use `keyCode` to recognize `ALT`, `CONTROL`, and `SHIFT`

A trivial game

- Goal: Hit the red ball with the blue ball to stop it from moving

```
float x = 150;
float y = 100;
float dx = 5;
float dy = 2;

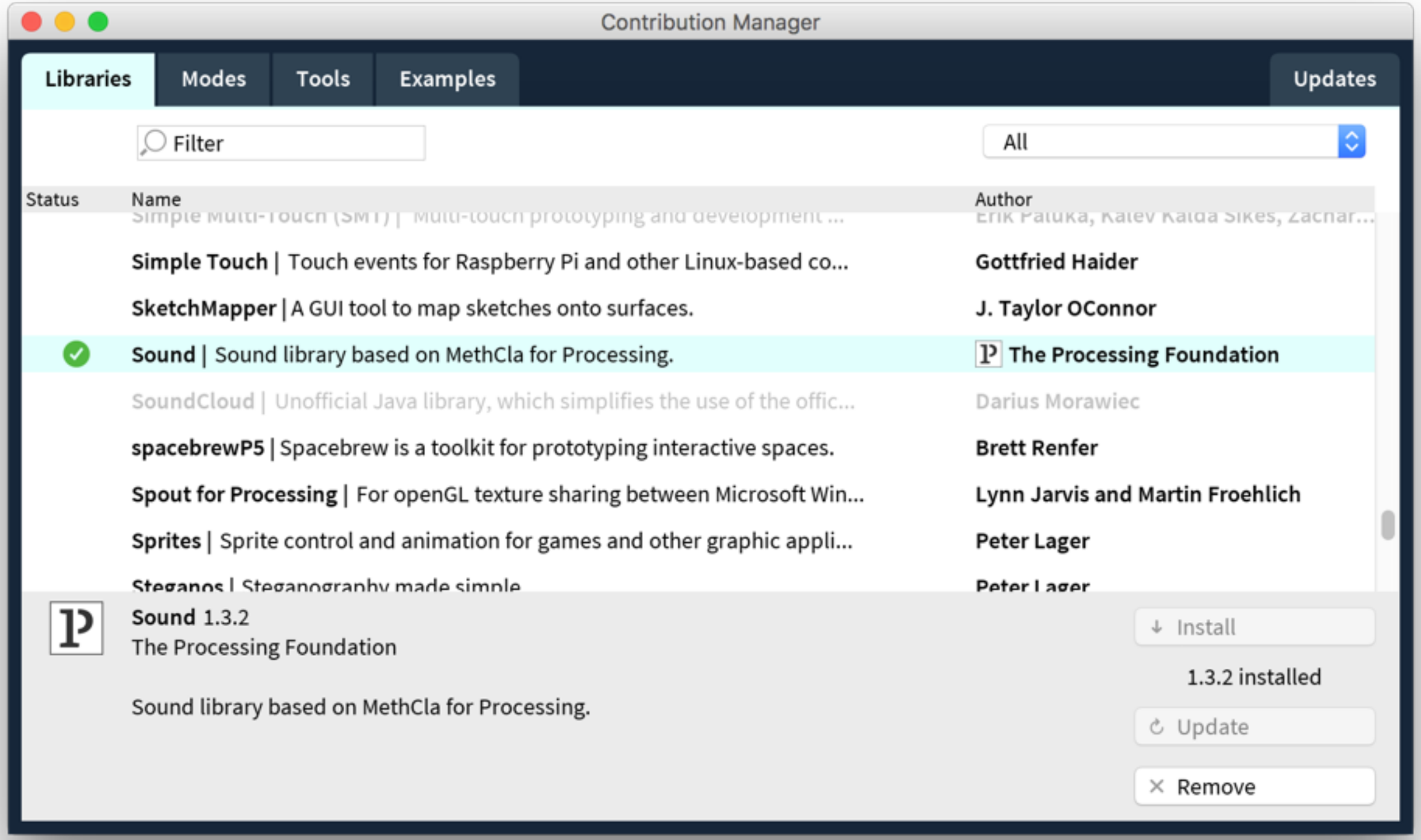
void setup() {
  size(300, 200);
  noStroke();
}
```




```
void draw() {
  // hit the red ball with the
  // blue ball to stop it
  x += dx;
  y += dy;
  background(255);
  // bouncing red ball
  fill(255, 0, 0);
  ellipse(x, y, 20, 20);
  if (x < 10 || x > 290) { dx = -dx; }
  if (y < 10 || y > 190) { dy = -dy; }
  // blue ball
  fill(0, 0, 255);
  ellipse(mouseX, mouseY, 20, 20);
  // detect collision
  if (dist(mouseX, mouseY, x, y) < 20) {
    dx = 0;
    dy = 0;
  }
}
```

Sound I

- To use sound in your program, you have to import a library
- Install from `Sketch -> Import Library... -> Add Library...`



Sound II

- Use `Sketch -> Add File...` to choose the sound file, or manually put it in the `/Data` folder of your sketch
- ```
  import processing.sound.*;
  SoundFile file;

  void setup() {
    file = new SoundFile(this, "wilhelm_scream.mp3");
  }

  void mousePressed() {
    file.play();
  }

  void draw() {
  }
  ```
- Sound is a little tricky to use--you *don't* want to start playing the sound each time `draw` is called!
  ```

It seems I've heard that scream before...

- The Wilhelm scream is a stock sound effect of a man screaming that has been used in more than 225 movies and television episodes, beginning in 1951 for the film Distant Drums.

  ...

  The effect gained new popularity (its use often becoming an in-joke) after it was used in the Star Wars series, the Indiana Jones series, Disney cartoons, and many other blockbuster films, as well as many television programs and video games.

  -- https://en.wikipedia.org/wiki/Wilhelm_scream

The End