

NOISY PET LIFE

NOTE

This project utilizes **Processing 3.0** and the **Speech to Text** library by Florian Schulz.

While images are provided, it is recommended that you make your own.

Google Chrome is a must for the above library, so you need to have that.

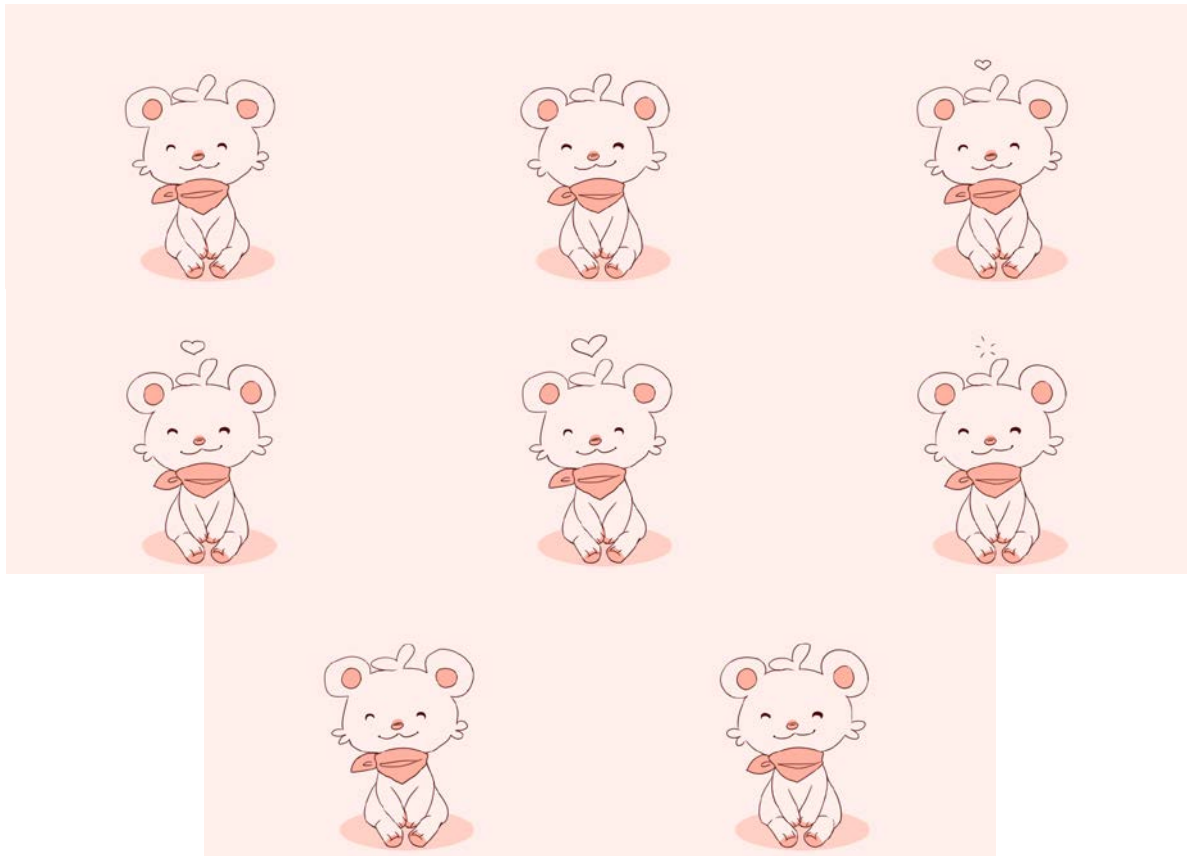
PREPARATIONS

WHAT YOU NEED

- **Processing 3** (*not 2*)
<https://processing.org/download/?processing>
Note, to run this project, you will need something that can run a server that your project can operate on, since the socket requires one. Something like WAMP/XAMPP/MAMP, etc.
- **WebSocketP5**
<http://muthesius.github.io/WebSocketP5/>
You can install this from within Processing.
Sketch > Import Library > Add Library > Search for WebSocketP5
If that doesn't work, simply go to the above link, download it, and put the **websocketP5** folder in the **libraries** folder for Processing 3.
- **Microphone**
A built-in laptop or earbud microphone works, but is not great. Ideally, you should acquire a USB or plug-in microphone.
- **Speech to Text Library** *is what you're using, there's nothing to actually download here...*
<http://stt.getflourish.com/>
As mentioned on the site above, you must have Google Chrome.
- **Silence**
If there is background noise, the app will probably not work too well, as it is very sensitive to all sound. The quieter it is, the more accurate it will be.
- **Super Auto Refresh** (or anything like it) - **OPTIONAL**
A chrome extension which will periodically refresh your page. This is due to a bug with the STT library where the connection will drop anywhere between 1 min to 10-20 min. Refreshing the page fixes that, so it's a good idea to have one of these refresh the socket page on its own so you don't have to do it.
- **Art Program**
For making your animations in!

ARTWORK & ANIMATION CLASS









P3 has provided a class on their site to do animations- which I tweaked for my project. I made all my animations frame by frame, and they are centered on the canvas. It does take a good number of frames to make a proper animation. Here is the shortest one I have:



The image files should be loaded into the project with a certain format to make things easier for Processing to load them.

The proper format for the code I am providing is as shown on the right:

- One shared file name for the first part (Teddy_Have_)
- Numbers written in two digits.
 - o If you have more animations, you may also write more digits (ie. 001, 002, etc.). The important part is that all the file names MUST have the same number of digits.

 Teddy_Have_00.png
 Teddy_Have_01.png
 Teddy_Have_02.png
 Teddy_Have_03.png
 Teddy_Have_04.png
 Teddy_Have_05.png
 Teddy_Have_06.png
 Teddy_Have_07.png

FROM: <https://processing.org/examples/animatedsprite.html>

```
//animation class to load the bear images
class Animation {
  PImage[] images; //array of images loaded
  int imageCount; // how many images are being loaded
  //the frames- minus one from the total as arrays count from 0
  int frame = imageCount - 1;

  //the constructor has the prefix and also the total number of images
  Animation(String imagePrefix, int count) {
    imageCount = count;

    //make an array the same size as the total number of images
    images = new PImage[imageCount];

    //populate this array with the files- load them!
    for (int i = 0; i < imageCount; i++) {
      // nf specifies there are two digits- so nf(i,3) would be 001.png, etc
      String filename = imagePrefix + nf(i, 2) + ".png";
      images[i] = loadImage(filename);
    }
  }

  // there are three display classes for this particular app
  // displayL (L stands for loop)- this will allow the frames to loop upon completion
  void displayL(float xpos, float ypos) {
    frame = (frame+1) % imageCount;
    image(images[frame], xpos, ypos);
  }

  // displayN (N stands for no loop)- a one time animation from a word trigger-
  // it goes back to the looping animation after
  void displayN(float xpos, float ypos) {
    frame = (frame+1) % imageCount;
    image(images[frame], xpos, ypos);
    if (frame == (imageCount - 1)){
      state = 0;
      counter = 0;
      displayMsg = "";
      println("animation complete");
    }
  }

  // displayB (B stands for bye) - for the goodbye animation
  // after it is done, the program closes
  void displayB(float xpos, float ypos) {
    frame = (frame+1) % imageCount;
    image(images[frame], xpos, ypos);
    if (frame == (imageCount - 1)){
      exit();
    }
  }
}
```

And to use them:

```
Animation baseBear, foodBear, deadBear, whatBear,
byeBear, dozeBear, wakeBear, giveBear, haveBear, callBear;
PFont font;

void setup() {
  fullscreen();
  background(255, 238, 235);
  frameRate(5);
  smooth();
  font = createFont("Please write me a song", 48, true);

  baseBear = new Animation("Teddy_Base_", 16);
  foodBear = new Animation("Teddy_Eat_", 19);
  deadBear = new Animation("Teddy_Dead_", 40);
  whatBear = new Animation("Teddy_What_", 11);
  byeBear = new Animation("Teddy_Bye_", 12);
  dozeBear = new Animation("Teddy_Doze_", 10);
  wakeBear = new Animation("Teddy_Wake_", 11);
  giveBear = new Animation("Teddy_Give_", 2);
  haveBear = new Animation("Teddy_Have_", 8);
  callBear = new Animation("Teddy_Call_", 13);
  socket = new WebSocketP5(this, 8080);
}
```

Simple make a new **Animation object** with the file name (without number) and the total number of frames there are. Then, using the Animation class, everything is setup for you!

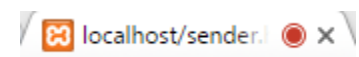
SOCKET CODE

FROM: <http://stt.getflourish.com/>

sender.html is identical to what is linked above.

```
<!DOCTYPE HTML>
<html>
<head>
  <script type="text/javascript">
    // We need to check if the browser supports WebSockets
    if ("WebSocket" in window) {
      // Before we can connect to the WebSocket, we need to start it in Processing.
      // Example using WebSocketP5
      // http://github.com/muthesius/WebSocketP5
      var ws = new WebSocket("ws://localhost:8080/p5websocket");
      console.log(ws)
    } else {
      // The browser doesn't support WebSocket
      alert("WebSocket NOT supported by your Browser!");
    }
    // Now we can start the speech recognition
    // Supported only in Chrome
    // Once started, you need to allow Chrome to use the microphone
    var recognition = new webkitSpeechRecognition();
    console.log(recognition);
    // By default, Chrome will only return a single result.
    // By enabling "continuous", Chrome will keep the microphone active.
    recognition.continuous = true;
    recognition.onresult = function(event) {
      console.log("im sending")
      // Get the current result from the results object
      var transcript = event.results[event.results.length-1][0].transcript;
      // Send the result string via WebSocket to the running Processing Sketch
      ws.send(transcript);
    }
    // Start the recognition
    recognition.start();
  </script>
</head>
<body>
</body>
</html>
```

Run it with your server app, and if it's working, you should see a red circle. (You may need to give chrome microphone access for the site). This page is also where you need to turn on the auto refresher for.



Meanwhile, in Processing:

```
import muthesius.net.*;
import org.webbitserver.*;

WebSocketP5 socket;

socket = new WebSocketP5(this,8080);
```

Import the appropriate libraries and make the web socket.

```
void websocketOnOpen(WebSocketConnection con){
  println("A client joined");
}

void websocketOnClosed(WebSocketConnection con){
  println("A client left");
}
```

These two functions connect the Processing app to the sender.html connected to google (and the speech recognition library).

```
void websocketOnMessage(WebSocketConnection con, String msg){
  receiveMsg = msg;
  displayMsg = receiveMsg;
  if (state==0){
    if (receiveMsg.contains("food")){
      println("state change");
      state = 1;
    }
  }

  //communicating with chrome for google speech library
  void stop(){
    socket.stop();
  }
}
```

msg is the **transcript** from sender.html. It's the word that is received. With this, it's easy to build a variety of reactions and recognized words for the pet!

Close the connection with stop() (otherwise the program will crash on exit).

MY CODE

See the demo folder in the github for all applicable files and full code.

```
WebSocketP5 socket;  
int state = 0;  
int counter = 0;  
String receiveMsg;  
String name = "noneset";  
String displayMsg = "noisy pet life";  
  
Animation baseBear, foodBear, deadBear, whatBear,  
byeBear, dozeBear, wakeBear, giveBear, haveBear, callBear;  
PFont font;
```

Global variables.

State is to decide what animation is currently playing (state changes are by word input), counter is used to put the bear to state after a certain amount of time. receiveMsg stores the word from user input. Name can be set by the user- the name of the pet. displayMsg is the current text on the screen, which begins as the title.

```
void setup() {  
  fullscreen();  
  background(255, 238, 235);  
  frameRate(5);  
  smooth();  
  font = createFont("Please write me a song", 48, true);
```

Set up the app- I've made mine fullscreen and set the colour (a light pink). The frame rate should be chosen to best fit the frame by frame animations. Smooth simply smooths everything that is drawn. I chose a custom font for my app, which is what the font part is.

```
//all the display states for the bear  
if (state==0){ // neutral  
  counter++;  
  baseBear.displayL(displayWidth/2, (displayHeight/2)+50);  
}  
else if (state==1){ // being fed  
  foodBear.displayN(displayWidth/2, (displayHeight/2)+50);  
}  
else if (state==2){ // playing dead  
  deadBear.displayN(displayWidth/2, (displayHeight/2)+50);  
}  
else if (state==3){ // asking its name  
  haveBear.displayN(displayWidth/2, (displayHeight/2)+50);  
}
```

After everything is placed and drawn, states change the animation playing. Depending on which state the app is in, display a different animation.


```

if (receiveMsg.contains("food")){
    println("state change");
    state = 1;
}
else if (receiveMsg.contains("dead")){
    println("state change");
    state = 2;
}
else if (receiveMsg.contains("name") && name != "noneseet"){
    state = 3;
    displayMsg = "My name is " + name + "!";
}

```

receiveMsg is the input from the user. If a keyword is found, change the state... and the previous function will display the proper animation.

With this knowledge, this app is highly expandable! More states, more animations- there's so much to be done with it! Hope this was helpful, and have fun making your noisy pet!