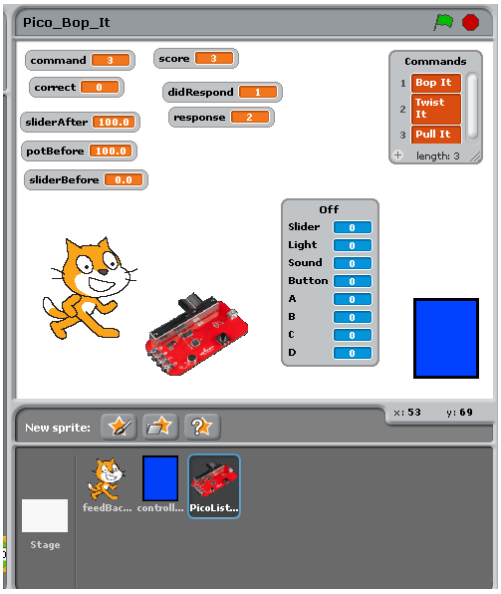
Directions for Bop It with Pico and Scratch



Description:

This Scratch and Pico Board activity simulates the Bop It ® Game created by Hasboro. The game has three actions that the player must perform when they are called by the program:

a. Bop It – Player must push the Button on the Pico Board
b. Slide It – Player must slide the slider on the Pico Board
c. Twist It – Player must twist the Potentiometer Connected to Pico Port A.

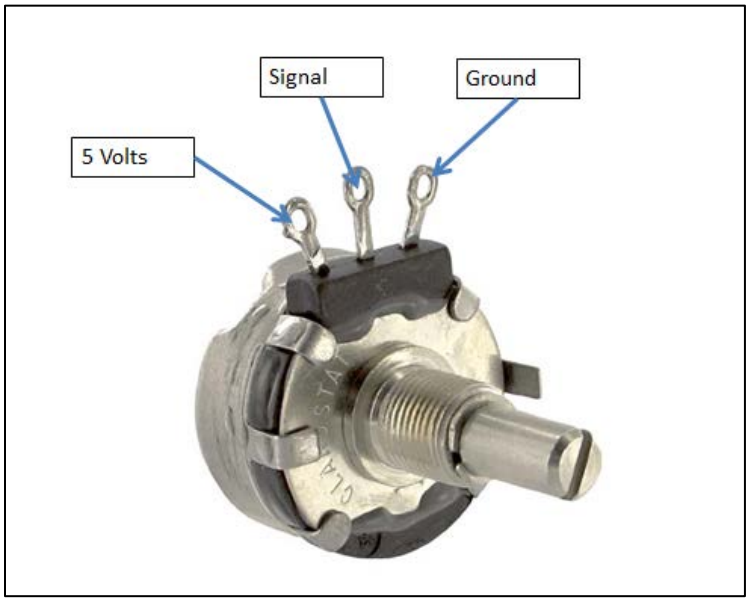
The Scratch Program randomly picks an action and then measures if the user correctly performs the action. The game is over when the user either does not perform any action, or the user performs the wrong action. The game is similar to Simon Says.

These directions will outline the Sprites (Objects) in the program, the variables and logic that drive the program, and the connections between the Pico Board and Scratch.

Process:

Part 1: Setup Pico Board and Potentiometer:

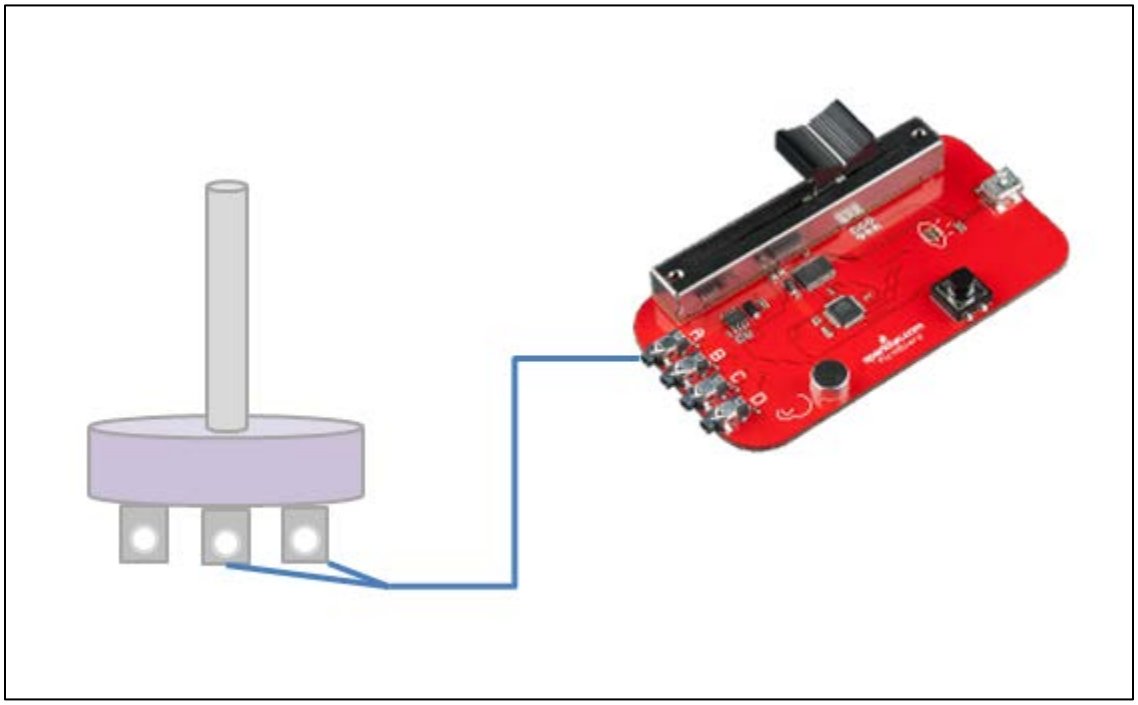
A Potentiometer is a type of variable resistor similar to the slider on the Pico Board. A Potentiometer usually has three connection points:



For use with the PicoBoard – make sure to connect the alligator clips to the middle connection (the signal) and one of the outer connections (5 Volts or ground).

Depending on the ohm rating of the Potentiometer, the reading from the Pico Board should range from 0 to 100.

For this lesson, connect the Potentiometer to Port A of the Pico Board and plug the Pico Board into the computer:

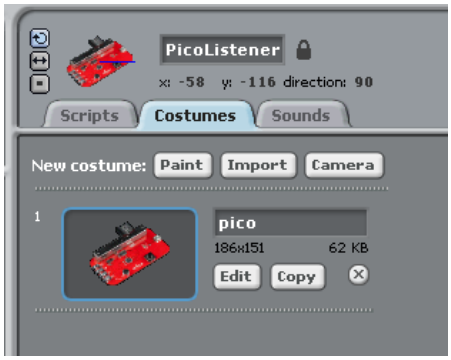


You might also find it helpful to secure the Potentiometer so a solid surface with tape. (The game can get intense).

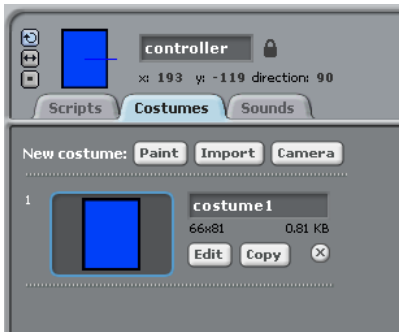
Part 2: Create the Sprites in Scratch

1. Start Scratch and create three Sprites: (The costumes do not change the operation of the Game. You may choose any graphic you wish)
    a. feedBackCat: States directions for Game and gives text commands

    b. picoListener: Contains the three threads that 'listen' for user input from the Pico Board. This object then Broadcasts commands based on User Input



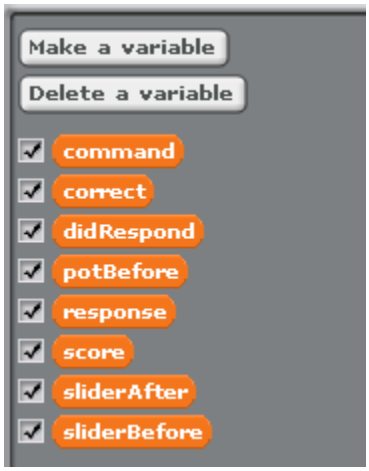
    c. controller: Contains the Scripts with the Game Logic



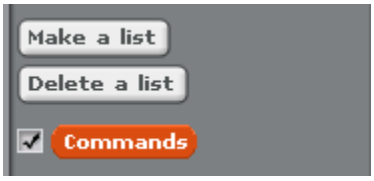
(Note: We will also Put Scripts on the Stage for this Program)

Part 3: Create the Variables for the Program:

1. Create the following Variables: (For All Sprites)
    a. `command`: This represents the Program's direction for the user.
    b. `correct`: Variable storing whether the user has the correct or incorrect response.
    c. `didRespond`: Variable storing if the user responded to the command
    d. `potBefore`: Stores the value of the Potentiometer (Port A) before the user changes the slider.
    e. `response`: Stores the response value from the user.
    f. `score`: How many responses from the user are correct.
    g. `sliderAfter`: Value of the Slider after the User moves the slider.
    h. `sliderBefore`: Value of Slider before Users changes the slider position.



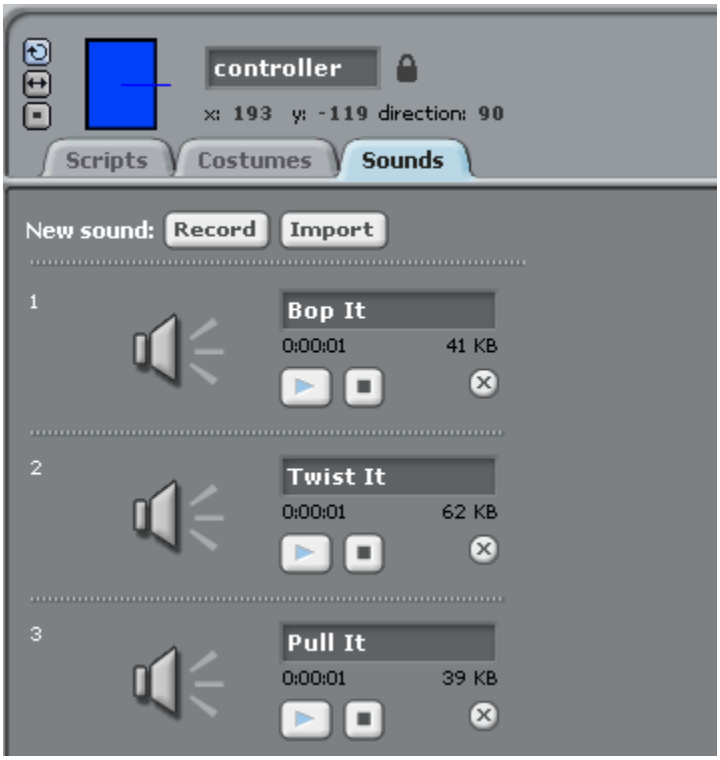
2. Create a list 'Commands' for all Sprites. 'Commands' will store the list of moves the program will choose from.



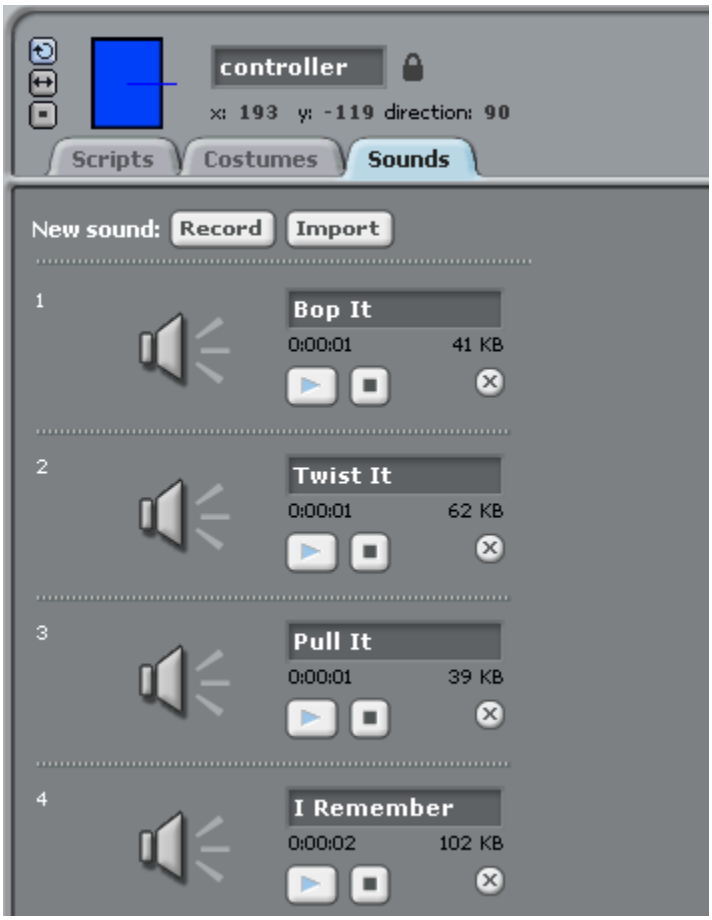
3. In the List, type in the following Data. Be sure to use the exact spelling
    a. Bop It
    b. Twist It
    c. Pull It

Part 4: Import or Record the Sound Media
(Option – download the media from http://nebomusic.net/picoboardlessons/bopitmedia/)

1. Go to the 'controller' Sprite. Click on the Sounds Tab.
2. You will need to record three sounds (With you speaking!)
    a. "Bop It"
    b. "Twist It"
    c. "Pull It"
3. Name the Sounds Exactly as you see them below: (They need to match the names from the Commands List)



4. Record a sound for “losing” the game. I used the phrase “I remember the first time I played Bop It.” (The actual Bop It game selected different phrases for losing). Name the sound ‘I Remember’



5. Import the following Scratch Sounds:
    a. Pop
    b. Rattle
    c. Plunge
    d. Zoop

6. Go the the Stage and Click Sounds. Import a Scratch music sound onto your stage. I selected Xylo1: (This will be the background music)

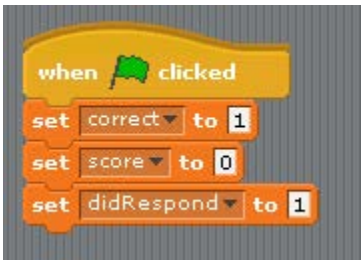
Part 5: Write the Scripts for the Controller:

The controller Object ('Sprite') consists of one Game Loop and several events ('Broadcast' or 'When I Receive"). We will write the Game Loop First and then build the Event Scripts. Here is the script for the Game Loop in pseudo code:

```
whenGreenFlag():
      # Initialize the variables
      correct = 0
      score = 0
      didRespond = 0
      while (true):
            until(correct == 0):
                  wait(2)
                  if (didRespond == 0):
                        correct = 0
                        stopAllSounds()
                        wait(0.2)
                        YouLose()
                  command = randomInteger(1, 3)
                  if (correct == 1):
                        playSound(Commands[command])
                        catSpeak()
                        didRespond  = 0
```

Process:

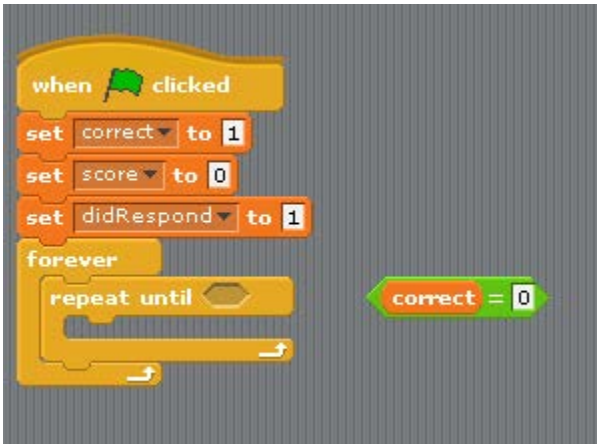
1. Go to the Controller Sprite.
2. The first part of the Game Loop initializes the several variables when the Green Flag is clicked. Assemble the following Scripts:



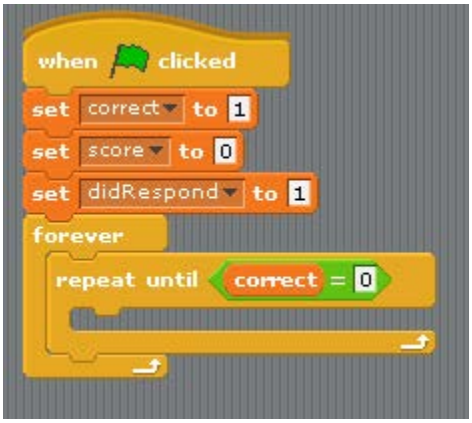
Note that 'correct' and 'didRespond' are set to '1'. In Scratch – we do not have Boolean variables, so we will use '1' for true and '0' for false. We are setting 'correct' and 'didRespond' to 1 so the game will make the first move. Score is set to 0 because the game is starting.

3. Drag a ‘forever’ block and a ‘repeat until’ block onto the Green Flag Stack.

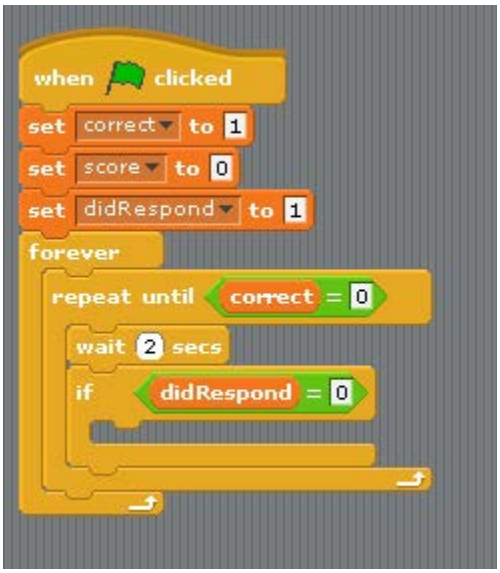
4. Put the correct variable and equals operator next to the stack. Set the equals block to ‘correct’ = 0.



5. Place the ‘correct’ = 0 into the repeat until block.



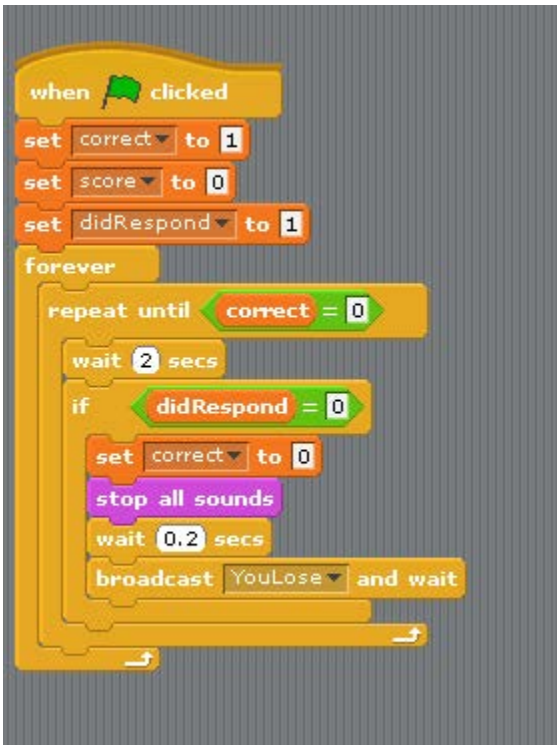
6. We will now do the if statement that measures the scenario that the user did not give a response. Place a wait block, if block, and an expression 'didRespond' = 0 into the repeat until block.



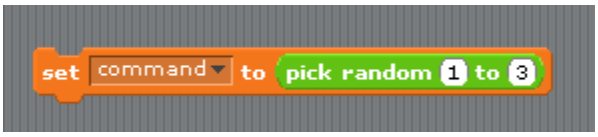
7. If the user did not respond, we want the following four actions to take place:
   a. Set correct to 0 so the repeat until loop will stop
   b. Stop all the sounds
   c. Wait a short duration of time.
   d. Broadcast or 'call' the 'YouLose' procedure.

   Place the following blocks together below the stack:

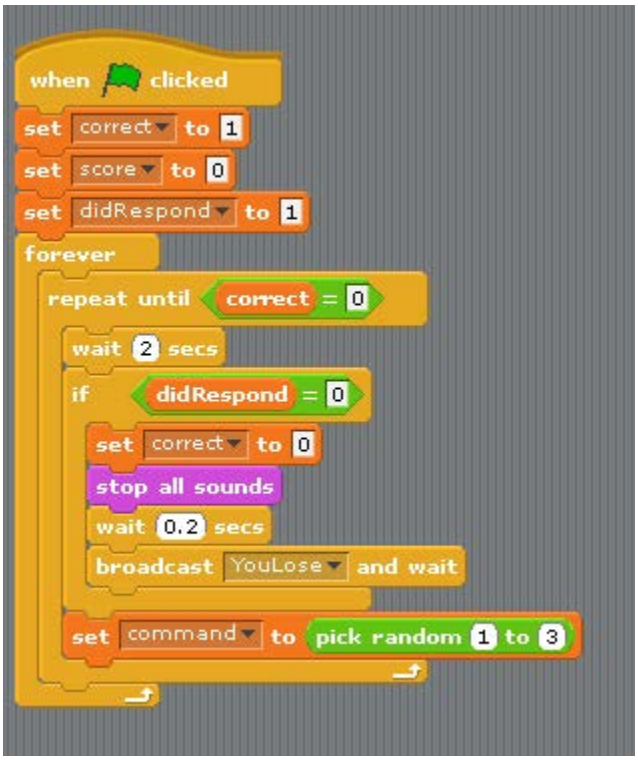
8. Place step 7's blocks inside the if didRespond = 0 statement



9. We now need to have the program select a random item from the List of commands. We will set the 'command' variable to be equal to a random number between 1 and the length of the List. (Lists in Scratch are 1 based).

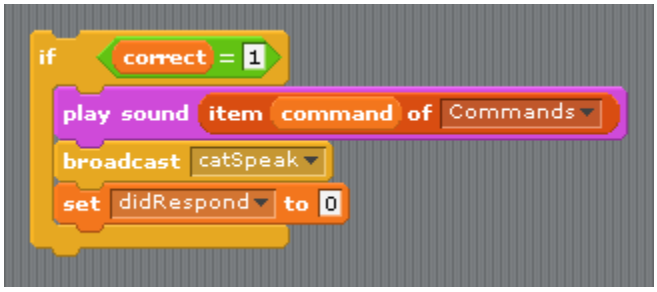


10. Place step 9's set command block below the if statement and inside the repeat until block:

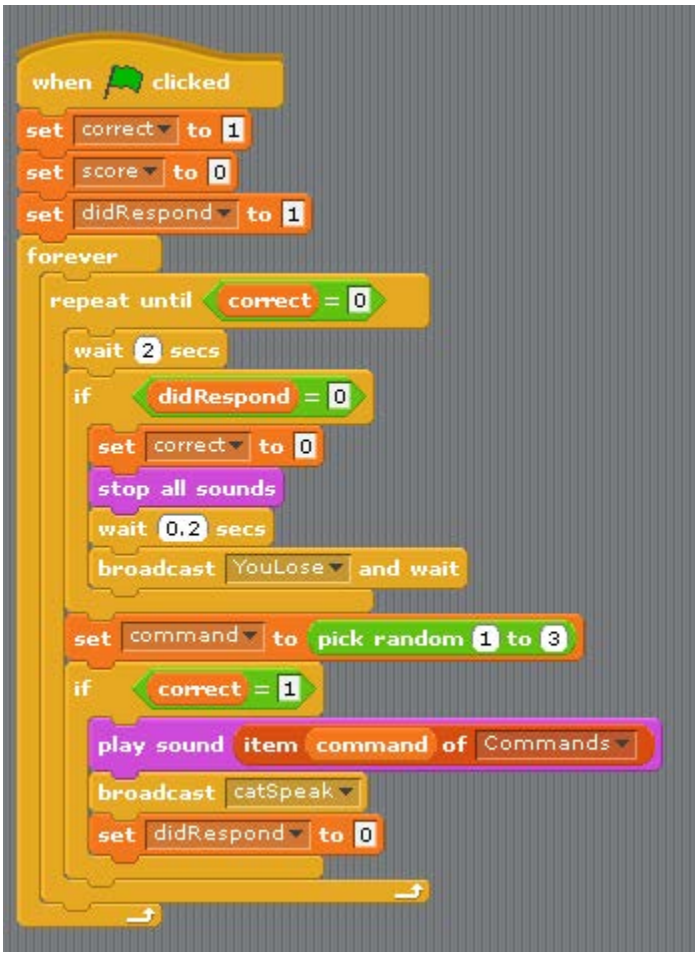


11. Now build the if statement that does the procedure for telling the user what action to perform:
    a. Play sound corresponding to the 'command' variable index from the List
    b. Broadcast the 'catSpeak' message to have the feedBackCat say the action.
    c. Set the 'didRespond' variable to 0 to wait for the user's response

Build these scripts:



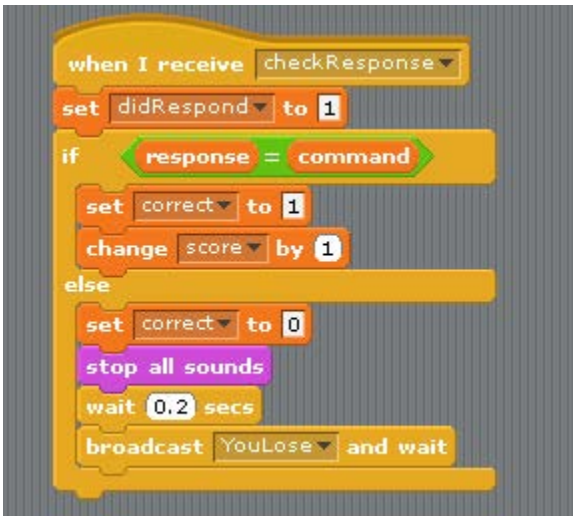
12. Place step 11's scripts into the 'repeat until' loop below the 'set command to' block.



13. Now build the method that check's the user's response. The pseudo code is:

```
def checkResponse():
   didRespond = 0
   if (response == command):
         correct = 1
         score = score + 1
   else:
         correct = 0
         stop all sounds
         wait(0.2)
         YouLose()
```

Build these blocks for the checkResponse message:

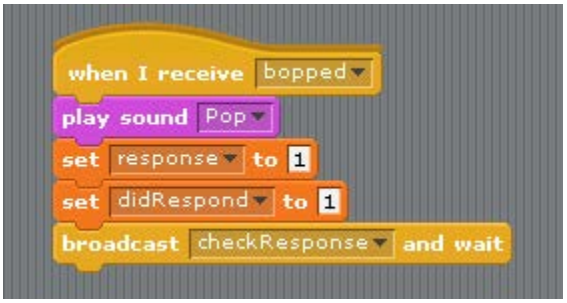


14. Build the response to the YouLose message:

15. The next series of events cover the responses to the actions from the user on the Pico Board. The pseudo code is:

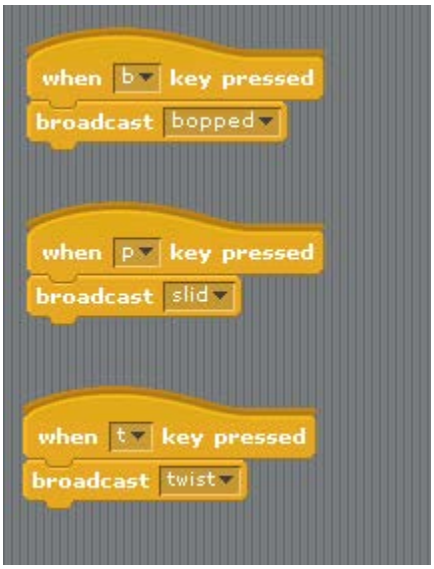
```
def bopped():
   playSound(Pop)
   response = 1  # User selected 1, the button
   didRespond = 1 # The user responded in some way
   checkResponse()  # Call the checkResponse() method
```

Build these blocks for the bopped message:



16. Build the blocks for the twist and slid messages:

17. To Test these scripts without the Pico Board, I built these blocks to react to keyboard events:

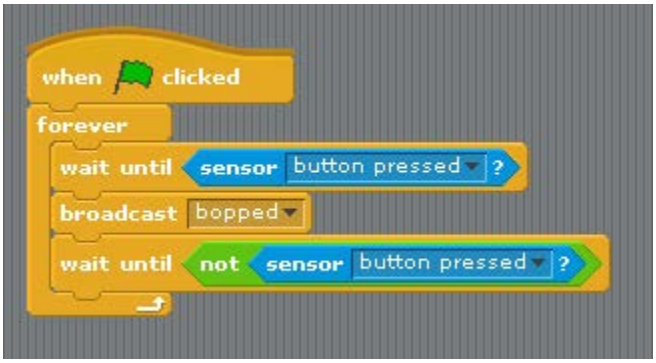


Part 6: Building the Scripts for the PicoListener

The PicoListener will have three threads with loops that respond to the button pressed, slider moved, and potentiometer turned events. For each event, a specific message ('bopped', 'slid', or 'twist') is broadcast. The controller sprite listens for these broadcasts and then checks the answer against the command.

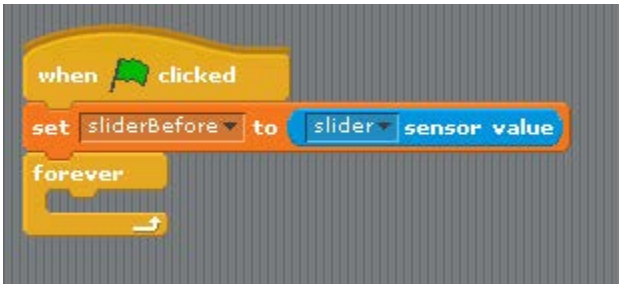
Process:

1. Go to the PicoListener Sprite.
2. Build these scripts for the button pressed event:

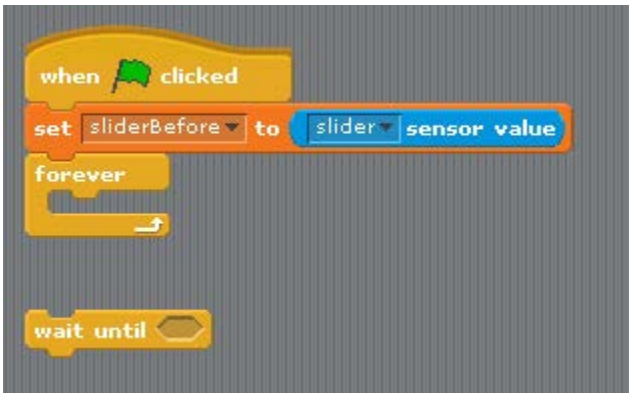


Using the wait until blocks inside a forever allows the event to only register once per press. This keeps the event from registering multiple times if the user holds the button down.

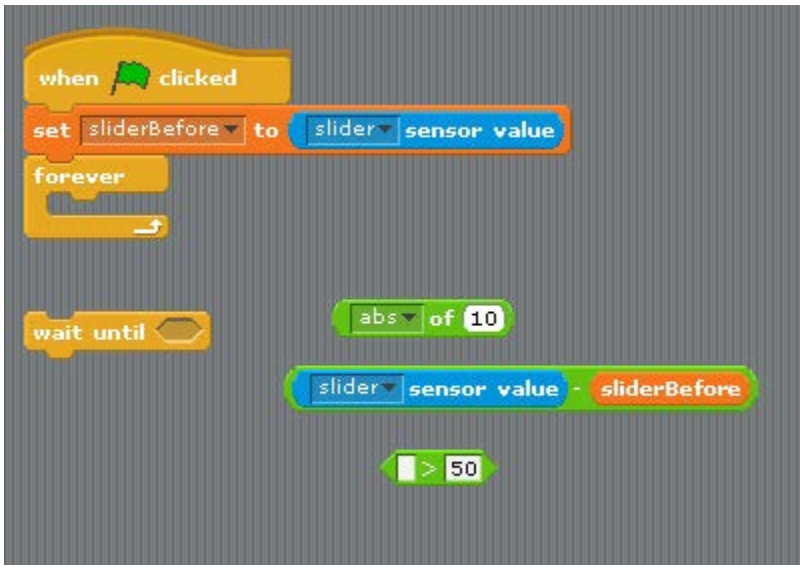
3. For the slider event, build the outline:



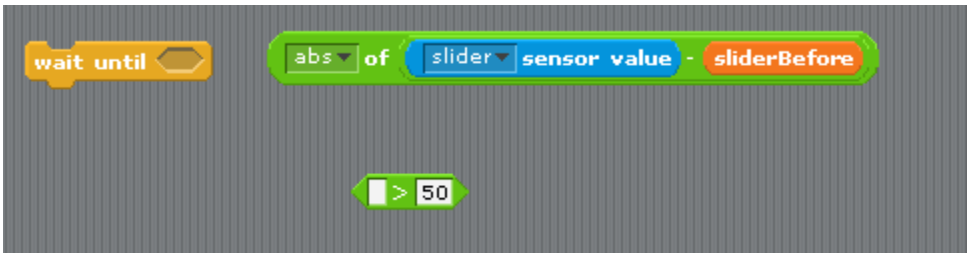
4. Now add a wait until block below the slider event stack:



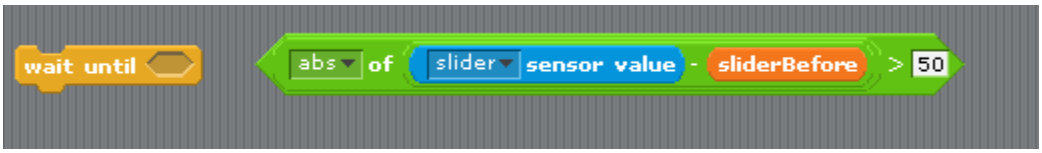
5. We need to compare if the slider is moved. This is done by waiting until the absolute value between the current slider position and the previous slide position is greater than 50. Add the following blocks:
    a. Abs block (sqrt block, then change the parameter to abs)
    b. Minus block
    c. Slider sensor value block
    d. sliderBefore variable oval
    e. greater than block



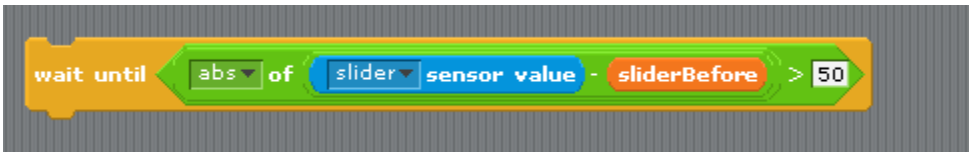
6. Place the (slider sensor value – sliderBefore) block into the 10 of the abs block.



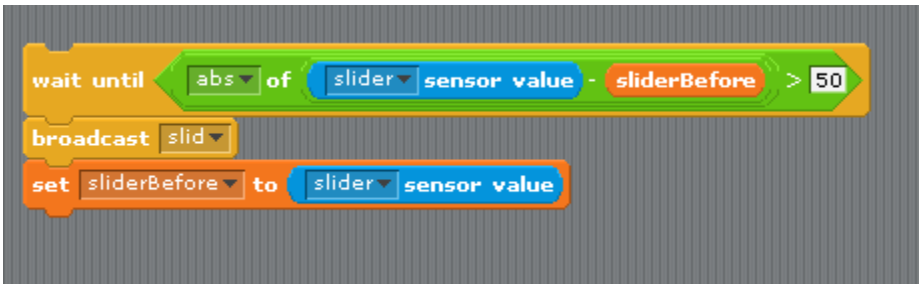
7. Place the abs(slider-sliderBefore) into the > block.



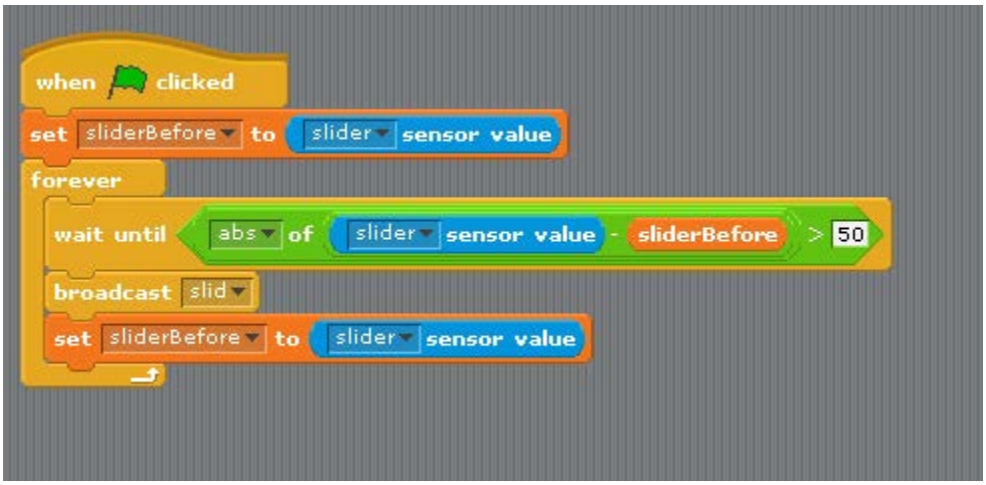
8. Place the > expression into the wait until block.



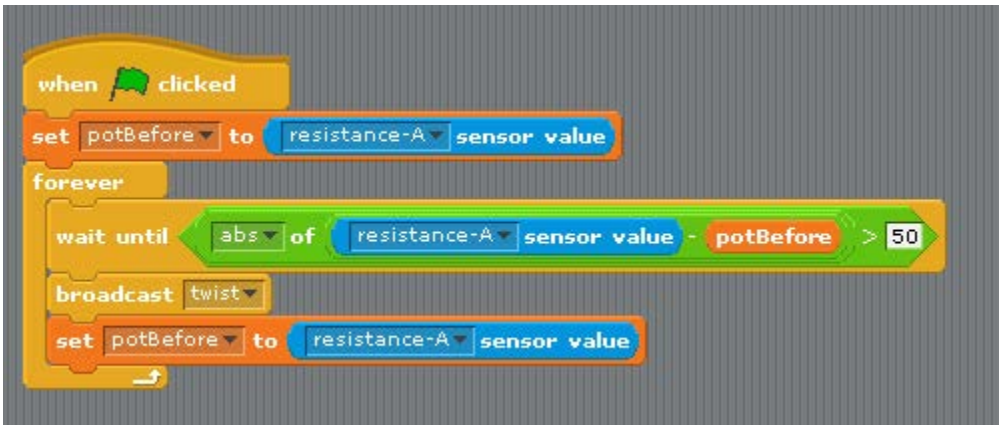
9. Place a 'broadcast slid' and set sliderBefore blocks under the wait until.



10. Place these inside the forever block from step 4.



11. In the same manner, do the scripts for the potentiometer event:

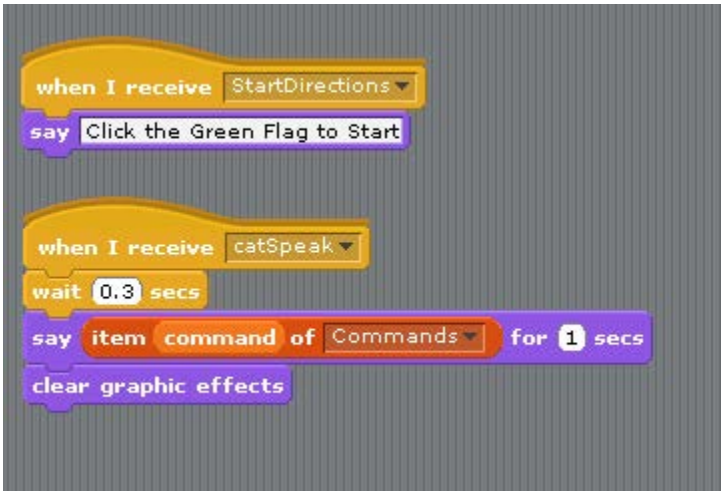


Part 7: Building the Scripts for the feedBackCat

The feedBackCat sprite provides a short statement with directions to start and then verbal commands for the movements.

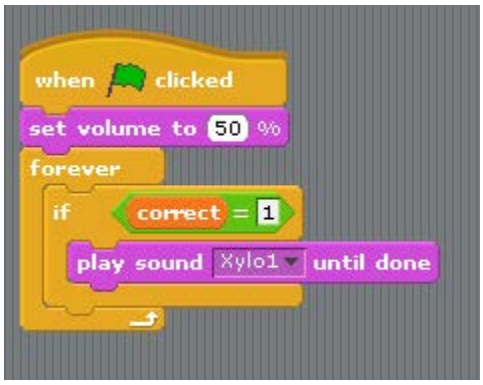
Process.

1. Go to the feedBackCat sprite.
2. Assemble the following scripts:



Part 8: Building the Scripts for the Stage

The stage in this program plays the background music for the game. Assemble the following Scripts on the stage:



At this point – you are done! Test your Pico Board and reaction skills!

Some extra challenges:

1. Keep a "High Scores" List
2. Provide feedback to the user on how many they got correct.
3. Ask the user if they want to play again.
4. Add more commands ("Shout it", "Stomp it" . . .)
5. Add a multi-player mode.
6. Add more statements when the player loses.
7. Refactor the Direction code to be more efficient.