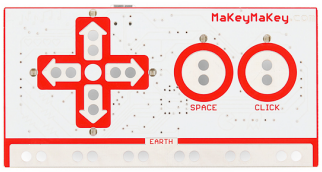
Processing with the MaKey MaKey

You have now built your skills up to the point that we can start adding other hardware as inputs into Processing. The most simple is the MaKey MaKey. This board emulates your keyboard and mouse, which you already have experience with!



PROCESSING
SKILL REQUIREMENTS
DIFFICULTY 1-5

ELECTRICAL PROTOTYPING	○—○—○—○—○
ROBOTICS	○—○—○—○—○
SOLDERING	○—○—○—○—○
PROGRAMMING	●—●—○—○—○
DIY	○—○—○—○—○

MATERIALS LIST

- Computer
- Processing
- Graph Paper
- Colored Pencils
- Super fine-tip sharpie

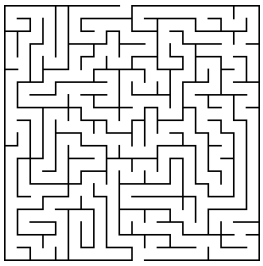
STEP 1: Intro to the MaKey MaKey

The MaKey MaKey is a piece of hardware that turns an item that conducts electricity into a button from your keyboard or mouse. All you have to do is attach yourself to Ground (Earth on the Board) and then clip an object to the button you want to use and then touch that object. That touch cause the board to emulate the key on the keyboard being pressed. More on the Makey Makey at www.makeymakey.com

STEP 2: Using the Makey Makey with pre-existing projects!

Open up Processing the Mouse projects. You can now use the Makey Makey in conjunction with everyday objects to create a creative interface for your previous Processing projects. Some good objects to use: fruits, vegetables, play-dough, copper foil, graphite on paper, try anything!

STEP 3: Maze Game with the Makey Makey



Lets develop a project that is designed to use the Makey Makey. With the arrow keys as the main focus of the Makey Makey we are going to create a simple maze game in Processing. This also takes us into some other subjects having to do with color/ edge detection and editing an image.

First we need a maze, we are actually going to be just one generated from the web. Go to www.mazegenerator.net and generate a maze using the standard settings. Do not download the maze, right click on the image and save it as a .png file. You can now open Processing and get ready to code!

STEP 4: The Setup

We first need to add the image of our maze using the **add file…** option in the Sketch dropdown menu. We can then create a `PImage` object we will call `maze` and then use the setup code to the right to get us up and running. After playing around with the size I have found that the standard size of these mazes is 325 pixels square. You can also load the maze image in `setup()` while you are at it.

```
int x= 170;
int y= 315;

void setup()
{
  size(325,325);
  maze= loadImage("maze.png");
}

void draw()
{
  image(maze,0,0);

  pushMatrix();
  translate(x,y);
  fill(255,0,0);
  ellipse(0,0,8,8);
  popMatrix();
}
```

STEP 5: Draw Code

The draw code for this sketch is going to be relatively simple. We are going to load the maze as an image and then create a red dot that we will control. Place the dot inside of a matrix so we can move it separate from the maze. The first thing is loading the image using the `image( )` function, placing it at 0,0. We then create a new matrix using a `pushMatrix();` and a `popMatrix();` Since we will be moving the ellipse, we need to create an x and y global variables. We are going to initialize the x and y variables as the starting point for our dot, we are using x= 170 and y= 315. Since we want to move the ellipse separately from the maze image we place the ellipse inside of the matrix and we will translate the whole matrix by x and y. We are now ready to increment and decrement x and y using the `keyPressed()` event.

STEP 6: KeyPressed

We can now increment and decrement x and y using the keyPressed() event function. We create the `void keyPressed()` and place four `if()` statements inside of it. These are going to check to see if the key is coded and if the `keyCode` is UP, DOWN, LEFT, or RIGHT. If it is `UP` we will decrement y, if `DOWN` we will increment y. If the coded key is `RIGHT` we will increment x and `LEFT` decrements x.

```
void keyPressed()
{
  if(key==CODED&& keyCode==UP)
  {
    y--;
  }
  if(key==CODED&& keyCode==DOWN)
  {
    y++;
  }
  if(key==CODED&& keyCode==RIGHT)
  {
    x++;
  }
  if(key==CODED&& keyCode==LEFT)
  {
    x--;
  }
}
```

Once you get these `if()` statements hammered out, hit play. Make sure that you can move your red dot with the arrow keys on your keyboard, or if you already have the Makey Makey plugged in touch Earth and the arrow pads of your Makey Makey and make sure your code works. What happens when you run into a maze wall? Nothing? We need to check to see if it is touching black or white!

STEP 7: Collision!

The last piece of code we have to create is to check for collisions. We are going to use the `get()` function. This function returns the color value of a specific x and y coordinate. In this case we want to get the color value of the center of our dot, or x, y. If that value shows black, or 0 then we want x and y to reset to the starting point. We first of all are going to create a global color variable call wall and we don't initialize it. Within our `draw()` loop we then make wall to `blue(get(x,y))`; The reason we use a color value is that white is all colors set at 255, black is all at 0. so if blue of the `get()` is 0, we are touching black. We then check this value using an `if()` statement. See the `if()` statement we added to our `draw()` code.

```
void draw()
{
  background(maze);

  hit= red(get(x,y));

  if(hit==0)
  {
    x=168;
    y=318;
  }

  pushMatrix();
  translate(x,y);
  fill(255,0,0);
  ellipse(0,0,5,5);
  popMatrix();

 if(x>148&&x<158)
 {
   if(y<2)
   {
   fill(0);
   text("You WIN!",50,150,200,50);
   }
 }
}
```

We also added another `if()` statement for detecting if the player gets to the end of the maze. Processing will print the string “You WIN” if the red dot enters a specific area that is within the end of the maze

STEP 8: Share your work!

If you would like to share your work...which you do! You can sign up at openprocessing.org. Open Processing allows you to share your code, view the drawing online, as well as allows others to fork or borrow your code to manipulate and change it on their own while still keeping your sketch intact for others to check out.

OpenProcessing

TAKING IT FURTHER

- Check out some examples of Processing at www.processing.org/exhibition/
- What other geometry could we draw using Processing...Hint: Reference!
- Draw a second line going from the upper right corner to the lower left corner
- change the size of your window!