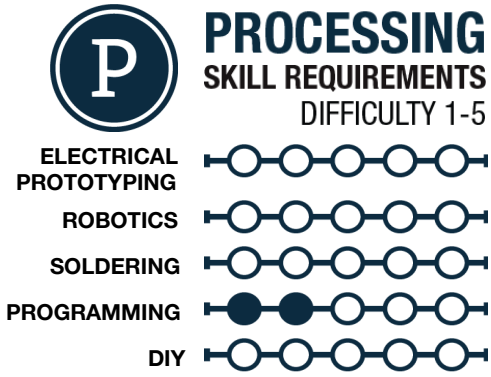
Processing the Mouse

We are now on our way to really breaking through in Processing. We are now going to focus on inputs for Processing. The most simple one you have been using this whole time is the mouse. We are going to now do a little more with the mouse than just read its X and Y coordinates!



MATERIALS LIST

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

STEP 1: Mouse Variables

We have been using a few mouse based variables for quite a while. But, that is only a small portion of what is available to you. Here are a few other mouse based variables and what they return. A couple of them require an `if()` statement to be really useful to you but we will give examples for those later on in this sheet.

`mouseX`- The mouse X coordinate between 0 and width.
`mouseY`- The mouse Y coordinate between 0 and height.
`pmouseX`- The mouseX coordinate the last time the sketch cycled through
`pmouseY`- The mouseY coordinate the last time the sketch cycled through
`mousePressed`- A boolean (True or False) value if the mouse button is being pressed
`mouseButton`- Returns either a RIGHT or LEFT value (is used with an if() statement to do different things if a different mouse button is pressed)

STEP 2: Mouse Functions

There are also a number of mouse event functions as well! We use these in a little bit of a different way than other functions. An event function is called out as a void outside of the draw() code and is constantly waiting for a specific event to trigger it. See the example of creating a simple drawing tool in Processing in Step 3. A couple of mouse event functions are:

`mouseClicked()` - The event is triggered when the mouse button is clicked
`mouseDragged()` - The event is triggered when the mouse button is held down and the mouse moved
`mouseMoved()` - The event is triggered when the mouse is moved
`mousePressed()` - The event is triggered when the mouse button is pressed
`mouseReleased()` – The event is triggered when the mouse button is released

```
void setup()
{
  //setup code
}

void draw()
{
   //draw code
}

void mouseDragged()
{
  //mouseDragged event code
}
```

Remember that the mouse functions are event functions which have their own set of curly brackets outside of the `draw()` and `setup()` code.

STEP 3: Using an Event function

Event functions are pretty cool! They only happen when the specific event happens, so you can have a bunch of code running and then when that specific event is triggered the code jumps to that event.

Our first event is going to be `mouseDragged()`. As you can see in the example code that there is nothing in the `draw()` code and everything is either in the `setup()` or the `mouseDragged()`, that's OK! So, at first glance nothing is going to happen until we click and drag! The key to remember for event functions is that they are their own void and what happens in that function is dictated by the code you put inside the curly brackets.

```
void setup()
{
  size(600, 600);
  background(150);
}

void draw()
{

}

void mouseDragged()
{
 line(pmouseX, pmouseY,mouseX,mouseY);

}
```

STEP 4: Using Mouse Variables within functions

We can also use a couple of mouse variables as well. Within the `mouseDragged()` event function there is a `line()` function with four mouse variables. pmouse variables are the coordinate of the mouse at the previous frame. So this line is from the previous mouse X and Y to the current mouse X and Y. Try this code and then we will add on to it!

STEP 5: Stacking event functions

There is no rule against using multiple event functions. For example, add a `mouseClicked()` event function and place `background(mouseX);` inside of it. You can then `mouseDragged()` to draw a line and then `mouseClicked()` to redraw the background, essentially erasing what you just drew. Pretty awesome!

```
void setup()
{
 size(400,400);
 background(150);
}

void draw()
{

}

void mouseDragged()
{
 line(pmouseX,pmouseY,mouseX,mouseY);
}

void mouseClicked()
{
  if(mouseButton==RIGHT)
  {
    background(150);
  }
}
```

STEP 6: Using Mouse Variables in an if()

To allow for more control we can also add an `if()` statement to the `mouseClicked()` event. This `if()` statement is going to check which mouse button is clicked and if it is the left button recall the `background()` function.

This gives you further control over what button is pushed and when, as well as what happens if a specific event is triggered! If we had to recreate this code within just the `draw()` it would be an elaborate string of `if()` statements nest within one another, this allows for simple and clean code! Enjoy!

STEP 7: 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!