User Control

Programs are more meaningful for users if they are allowed to affect the flow of events. These notes provide a guideline, through examples, of how we can create programs that allow the user to control its elements. These will include techniques in Mouse and Keyboard control.

When the user interacts through clicking a mouse button or keyboard key, the machine would know about it at Operating System level. However, unless we dictate the program to do something about it, the flow of the program will remain unaltered.

In other programming languages, these events need to be handled up to a certain level of detail. However, in Processing, there is a very neat and easy way of making the best out of user interaction. It is nonetheless assumed that before reading these notes, the 'Logical Statements' notes were understood.

Mouse Control

Mouse control consists of the user doing the following main actions:

- Moving the mouse pointer, where at every movement the mouse point has a specific x and y coordinate.
- Click a button
    - Which can be Right button
    - Or the Left button.

The handling of mouse control will in fact follow the same logic.

Mouse movements

For the detection and tracking of movement, Processing provides 2 very important functions that return the coordinates of the mouse pointer:

- `mouseX`
    - Whenever called, this returns the x-value of the current mouse pointer coordinate.
- `mouseY`
    - Whenever called, this returns the y-value of the current mouse cursor coordinate.

These 2 values can be passed as parameters in all functions that accept a number as parameter. It is also important that the said functions are called within the `draw()` method so that it would provide an ongoing experience for the user. Below follow some examples:

- `ellipse(mouseX, mouseY, 5,5);`
    - This function would allow the user to move the cursor around the screen and leaving a trail of circles behind.
- `ellipse(300, 300, mouseX,mouseY);`
    - This function results in a special effect of having an ellipse drawn at the centre of a 600x600px window and the ellipse's width and height would change with the movement of the mouse cursor.

Mouse Clicking

The second aspect of mouse control is clicking of buttons. Whenever a mouse button is clicked, irrespective of which one, Processing is able to know about it. This is possible through the `mousePressed` flag. This flag of type Boolean can either contain a True or False value. When a mouse button is pressed, it is set to True and then back to False when released.

This flag can be used as condition of an if-statement and thus enabling the user to only perform a function when a button is clicked. The example below shows

how the program can allow the user to leave a trail of circles if a mouse button is clicked.

```
void draw(){
      if(mousePressed){
            ellipse(mouseX, mouseY, 5,5);
      }
}
```

A convenient additional feature is making the program recognise which mouse button was clicked. This can be determined through another condition, this time polling the `mouseButton` system variable that stores which button was clicked. We can check whether the left, right or centre button was clicked by checking whether `mouseButton` is equal to any of them by using the == comparison operator. Note that when using the comparison operator, the enumerations LEFT, RIGHT or CENTER need to be written in uppercase for Processing to recognise them as such.

The example below builds on the previous example. In addition, it draws an ellipse when left button is clicked and rectangles when the the right button is clicked:

```
if(mouseButton == LEFT){
       ellipse(mouseX, mouseY, 5,5);
     }else
       if(mouseButton == RIGHT){
         rect(mouseX, mouseY, 5,5);
       }
```

Keyboard Control

When using keyboard control, a very similar logic to mouse control follows. First, we will need the program to know that a keyboard button is pressed. This

can be done by checking the value of the `keyPressed` Boolean flag in a similar way to the way we checked `mousePressed`. The second step is to check which key was pressed. Processing stores the `key` being pressed in the key system variable.

The example below allows the user to press either the 'a' or 'b' key. When the 'a' button is pressed, a circle is drawn in the top left corner of the screen. When the 'b' button is pressed, a square is drawn in the bottom right corner of the screen. It is important to note that the value of the key in question needs to be written in single quotes when being polled against the `key` variable as shown below:

```
void draw(){
  if(keyPressed){
    if(key == 'a'){
      ellipse(150,150, 50,50);
    }else
      if(key == 'b'){
        rect(450, 450, 50,50);
      }
  }
}
```