Unit 3 Lesson 2 Use of built-in methods for user interaction

Syntax:

1. **setup()** block runs once. It allows for any **initialization such as graphics window size, background color, stroke and frame rate.**
2. **size()** function sets the global variables **width** and **height.**
3. **mousePressed() code is executed when a mouse key is pressed**
4. **keyPressed()- code placed in this procedure is executed when key is pressed**
5. **width and height – detect and allows you to use window size**
6. **mouseX, mouseY – returns the x and y coordinates when mouse is clicked**
7. **pmouseX,pmouseY – returns the previous mouseX and Y coordinates**
8. **frameRate(int)- sets the number of times per second the draw() is executed. Application can only have one frame rate.**

1.Using the system variables width and height in your code

Circle in the center: *This program draws a circle with radius 50 in the center of the Graphics Window*

Example 1

Center circle using own calculations
for a specific size window
size(400,400);
ellipse(200,200,50,50);
	Always in the middle regardless of window size
	
	ellipse(width/2, height/2, 50, 50);

2. Using mouseX and mouseY to draw

1. Create window size (800,500) and a background FUSCIA , the stroke color white
2. Mouse click draws lines between point1 (350,55) and point (mouseX, mouseY), where mouseX and mouseY are the coordinates of a point where the mouse is clicked.

Example 2

```
//DrawLine w/mouse
void setup() {
        size(800, 500);
        background(192, 0, 255);
        stroke(255);
}
void draw() {
        line(150, 25, mouseX, mouseY);
}
```

1. mouseX and mouseY are Processing system variables which hold the value of the current X and Y locations when the mouse is clicked on the graphics window.

2. The mouse coordinates can be used in variety of applications especially when interaction with the user is required.

3. **Putting it all together. Type the code shown below (do not include line numbers) . Describe the output.**

Example 3

```
1.  float y=200.0;
2.  float x=10.0;
3.  void setup()  {
4.  size(200,200);
5.  background(255);
6.  }
7.  void draw()  {
8.  y=y-1;  //decrement y
9.  x=x+1;  //increment x
10. delay(y);
11. fill(10,30,255);
12. rect(0,0,200,200);
13. fill(255);
14. triangle(30,60,90,100,20,10);
15. ballLeft();
16. ballRight();
17. }
18. void ballLeft() {
19. delay(100);
20. fill(10,30,255);
21. ellipse(100+y,30,20,20);
22. }
23. void ballRight() {
24. fill(10,255,255);
25. ellipse(10+x,60,30,30);
26. }
27. void mousePressed() {
28. stroke(0);
29. fill(175);
30. rectMode(CENTER);
31. rect(mouseX,mouseY,16,16);
32. }
33. void keyPressed()  {
34. background(255);
35. }
```

1.Why are x and y declared outside of all methods at the top of the program?

This way they can be used anywhere in the program.

2. Change delay(y) to delay(200) and explain what changes in the output.

There seems to be a more consistent unevenness

3. Comment out delay(y) and explain what changes in the output

The circle seems to run a little faster since it's not asked to stop as often. It still staggers a bit since there's still delay() in other methods.

4.How does the program change when one of the coordinates is continuously changed?

There will be a change probably in appearance as it moves in various directions.

5.Press a mouse key when the output window is selected. What changes in the output?

Grey squares with a black stroke appear wherever you click as long as it's outside the kind of "square" of the triangle.

6.Press a keyboard key. How does the output change?

The entire screen turns white the time you hold the keyboard key, well, it seems like it.

Use mouse coordinates in various shape commands

1.**Point**

Place the following line inside
void draw() {
point(mouseX, mouseY);
}

Output

Dots follow where my mouse is/was.

2**. pmouseX,pmouseY**
(previous mouseX, previous mouseY)

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

Output

A literal line goes where my mouse goes.

4. Mouse coordinates and mouse press working together

```
void draw() {

// only executes code if mouse button is pressed

 if (mousePressed == true) {
  stroke(255,255,255);
  strokeWeight(10);
 line(mouseX, mouseY, pmouseX, pmouseY);
 }
}
```

This time it's a white line that's 10 pixels thick, and it only draws when I left click.

5. Think of ways any and all of the Processing built-in variables and methods introduced in this lesson can be used in a mini game. Work with a partner and share your idea for a minigame.

Right now the best I can think of is a simple-ish drawing + mario style game. You have the arrow or wasd keys as forward, backwards, jump, and crouch. Sometimes up and down too. To add a twist we can make it so you have limited "material" you can draw with every level that you need to use to go through the level.
Other concepts could be clicking where you want to redirect a moving object and try to direct them to an exit, typing challenges, etc.
Simple drawing and adventure things could work too, like a drawing app or something like the dinosaur game.
Oh, and other keys could also be used to open menus and such.