Using Methods

Methods that Handle Events

Lecturer: Caio Fonseca

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Caveat

- The term function is used in Processing e.g. line(), fill(), etc.
- The term method is used in Java.
- As this course is primarily about learning the Java language, we are going to use the word method instead of function from here on in.

Topics List

- Introduction to Methods

- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List

- Using methods to handle Mouse Events.

Recap: Methods in Processing

- Processing comes with several pre-written methods that we can use.
- A methods comprises a set of instructions that performs some task.
- When we invoke (call) the method, it performs the task.
- Some methods we have used are: rect, ellipse, stroke, line, fill, etc.

Recap: Methods in Processing

- We have also **written** two methods to animate our drawings:
  - void setup()
    this method is automatically called once when the program starts and should not be called again. It typically sets up your display window e.g. screen size, background colour.
  - void draw()
    this method is automatically called straight after the setup() call. It continuously executes the code contained inside it.

Introduction to Methods

- Normally, a method will perform a single well-defined task.
- Methods can be run many times within your program.
- This makes code easier to read, update and reduces the chance of error.

Introduction to Methods

- When we get a method to perform its task we say that we are **calling** the method (invoke the method).

- When we call a method, what we are actually doing is telling the program to:

  1. Jump to a new place (where the method instructions are stored).
  2. Carry out the set of instructions that it finds there.
  3. When it has finished, return and carry on where it left off.

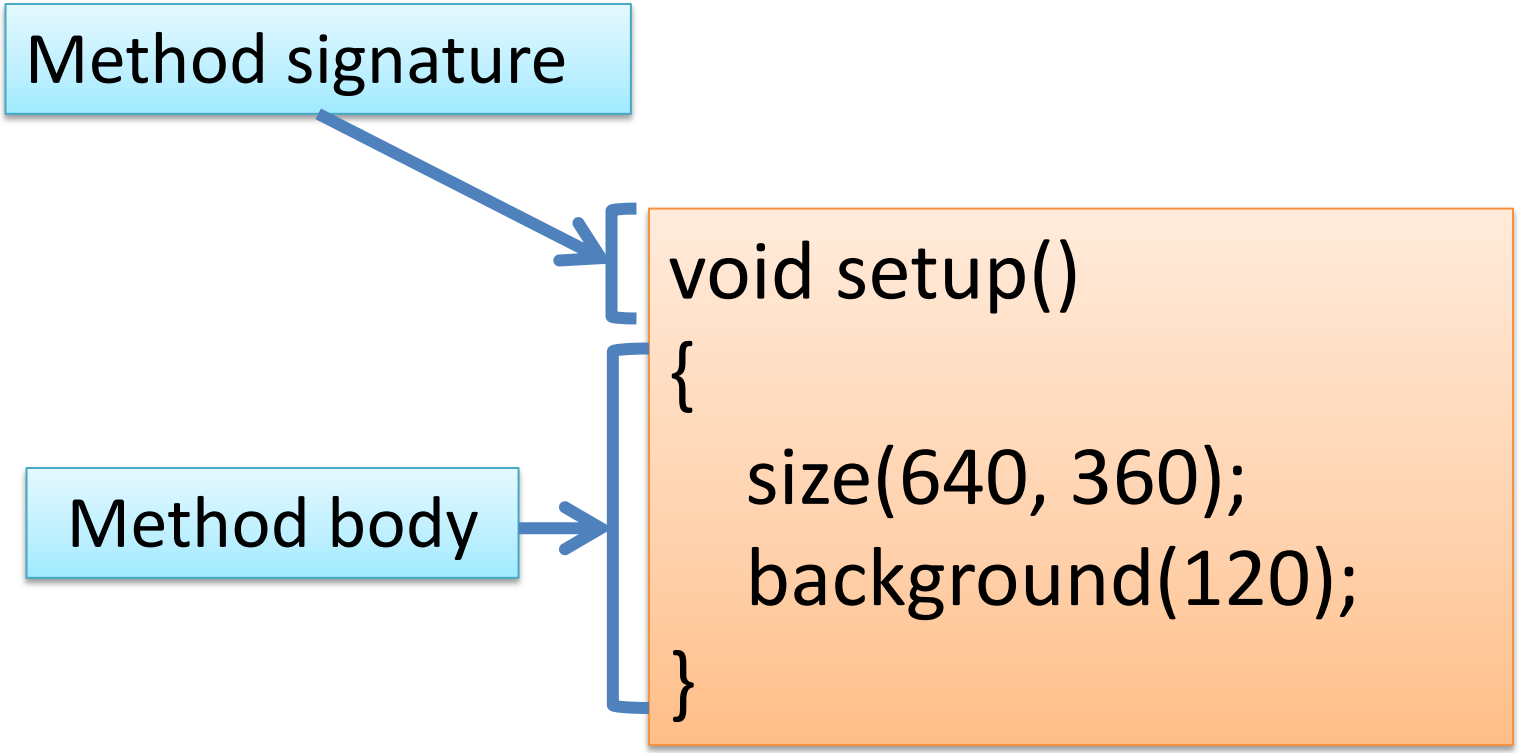
Topics List

- Introduction to Methods

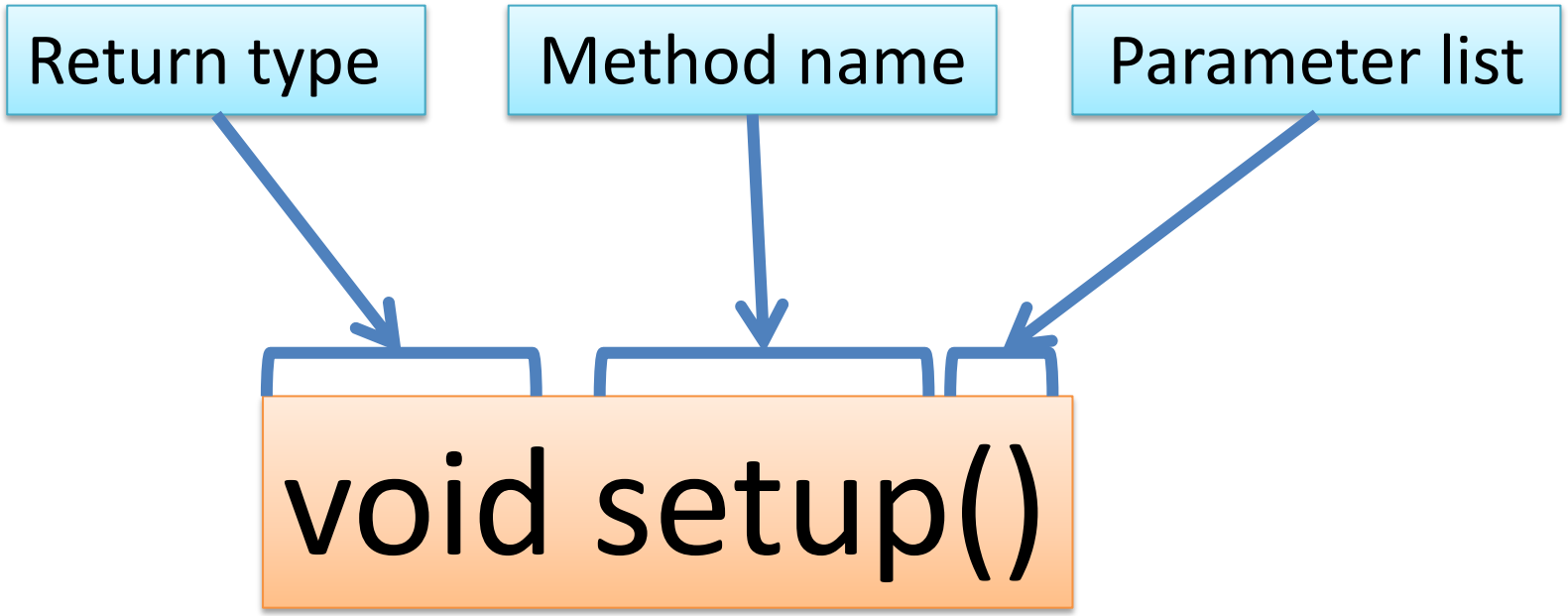
- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List

- Using methods to handle Mouse Events.

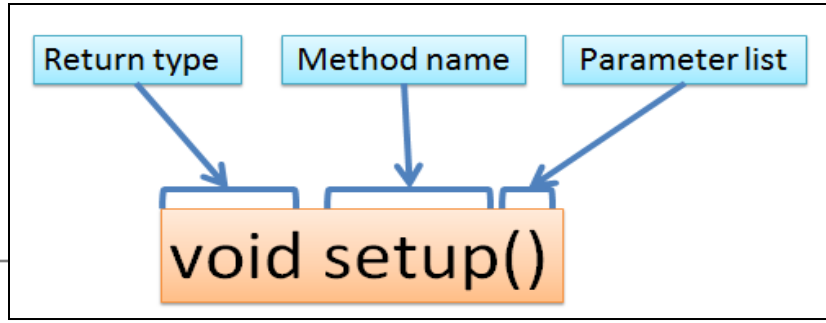
Method Terminology



Method Signature



Return Type: void



- Methods can return information.
- The void keyword just before the method name means that nothing is returned from the method.
- void is a return type and must be included in the method signature if your method returns no information.

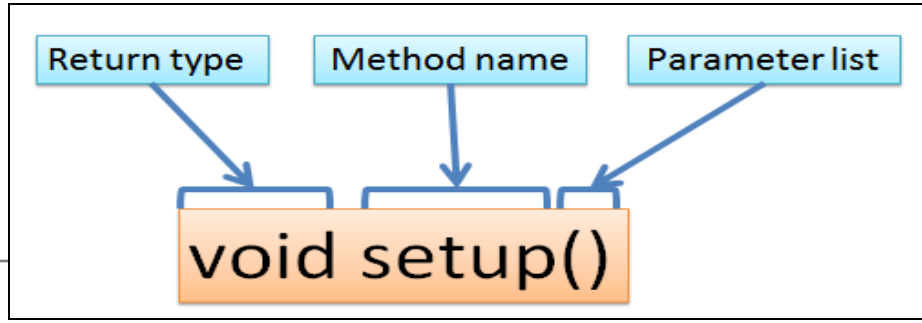
Topics List

- Introduction to Methods

- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List

- Using methods to handle Mouse Events.

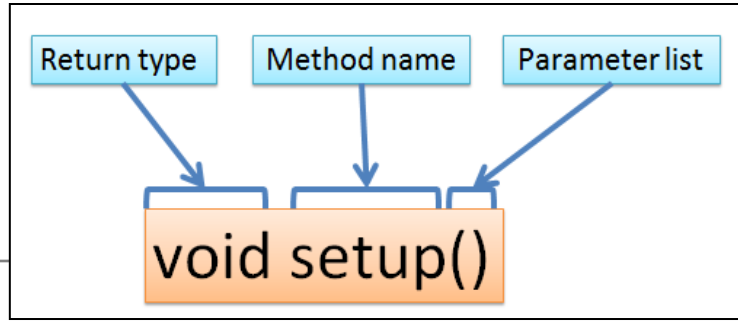
Return Type: int



- When a data type (e.g. int) appears before the method name, this means that something is returned from the method.

- Within the body of the method, you use the return statement to return the value.

Return Type: int



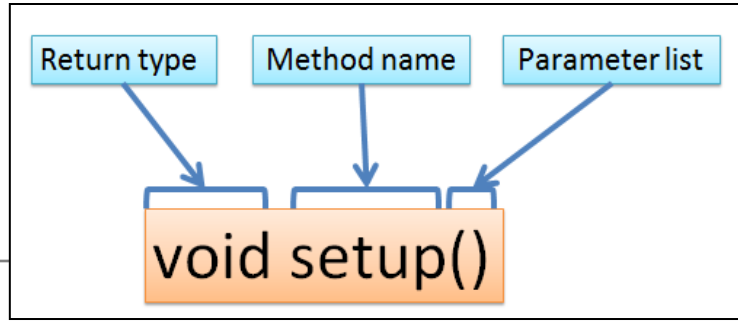

```
int val = 30;


void draw()
{
    int result = timestwo(val);
    println(result);
}
```

```
int timestwo(int number)
{
    number = number * 2;
    return number;
}
```

// The red **int** in the function declaration
// specifies the type of data to be returned.

https://processing.org/reference/return.html

Return Types

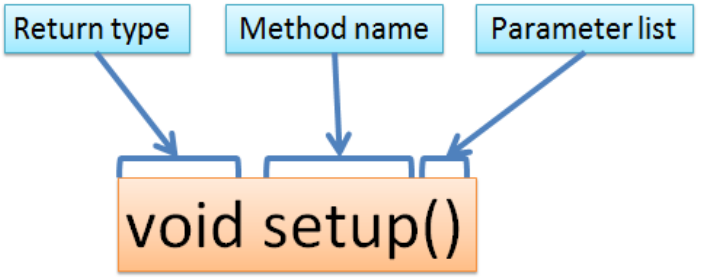


- Methods can return any type of data e.g.
  - boolean
  - byte
  - char
  - int
  - float
  - String
  - etc.
- You can only have one return type per method.

Topics List

- Introduction to Methods
- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List
- Using methods to handle Mouse Events.

Method Name

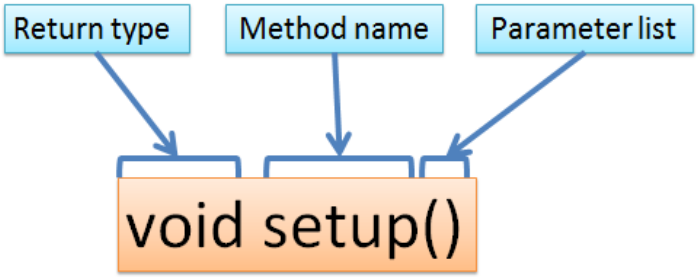


- Method names should:
  - Use verbs (i.e. actions) to describe what the method does e.g.
    - calculateTax
    - printResults
  - Be mixed case with the first letter lowercase and the first letter of each internal word capitalised.

Topics List

- Introduction to Methods
- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List
- Using methods to handle Mouse Events.

Parameter List



- Methods take in data via their parameters.
- Methods do not have to pass parameters e.g. setup() has no parameters.

Methods with NO Parameters

```
void noStroke()
void setup()
void noCursor()
```

- Methods do not have to pass parameters.
- These methods have no parameters; note how no variable is passed in the parenthesis i.e. ().
- These methods don’t need any additional information to do its tasks.

Methods WITH Parameters

```
void strokeWeight(float
weight)

void size(int width, int
height)
```

- If a method needs additional information to execute, we provide a parameter so that the information can be passed into it.
- The methods above have one parameter.
- A method can have any number of parameters.
- A parameter is a variable – it has a type (e.g. int) and a name (e.g. width).

Topics List

- Introduction to Methods

- Method Terminology:
  - Return Type
  - Method Names
  - Parameter List

- Using methods to handle Mouse Events.

Mouse Actions and their Methods

Action	Description	Method
Clicked	Mouse button is pressed and then released	mouseClicked()
Pressed	Mouse button is pressed and held down	mousePressed()
Released	Mouse button was pressed, but now released	mouseReleased()
Moved	Mouse is moved without any buttons being pressed	mouseMoved()
Dragged	Mouse is moved with a button pressed	mouseDragged()

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

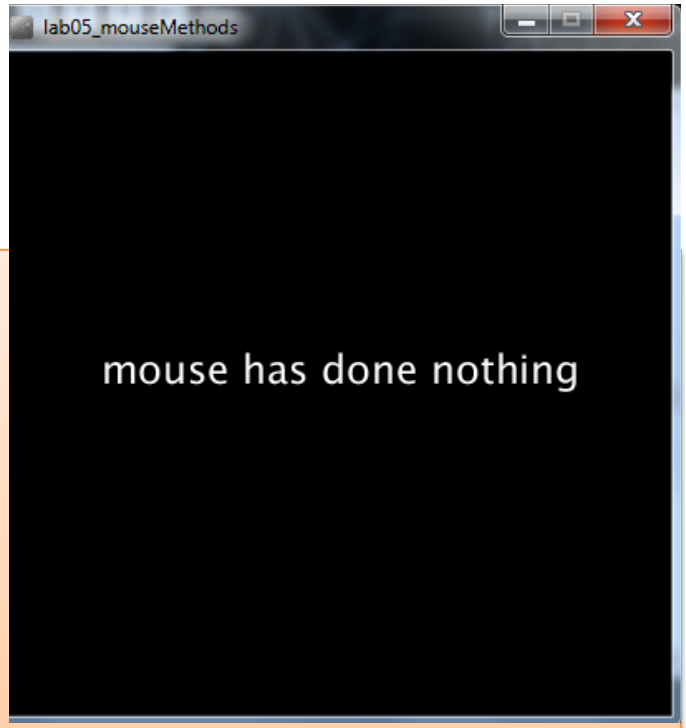
Mouse Methods

- Mouse and keyboard events only work when a program has draw().

- Without draw(), the code is only run once and then stops listening for events.

https://processing.org/reference/mousePressed_.html

Method Example 2 – setup()

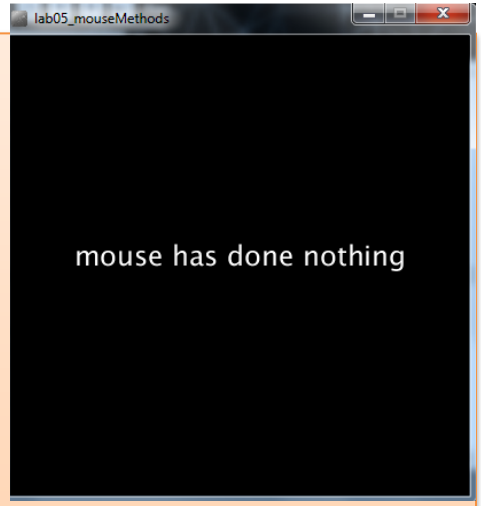



```
void setup() {
    size(400, 400);
    background(0);
    textAlign(CENTER);
    textSize(24);
    fill(255);
    text("mouse has done nothing", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Method Example 2 – draw()

```
void setup() {
    size(400, 400);
    background(0);
    textAlign(CENTER);
    textSize(24);
    fill(255);
    text("mouse has done nothing", width/2, height/2);
}

void draw(){
}
```

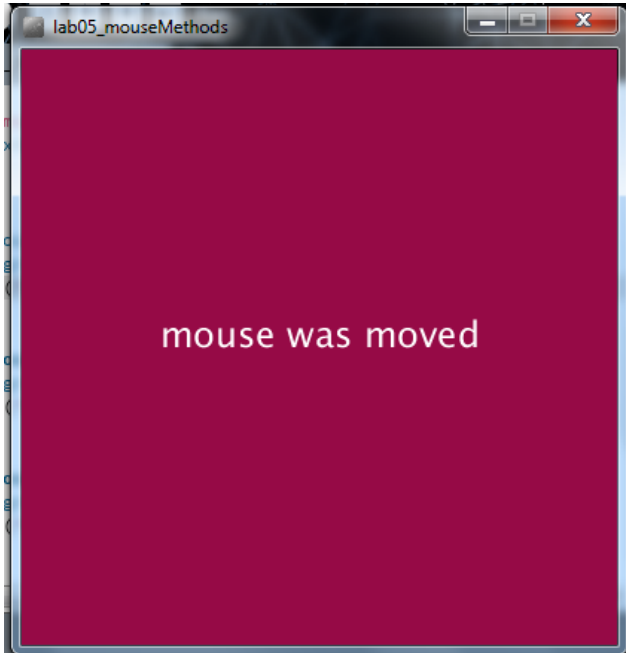


draw() is required because the mouse events only work when a program has it.

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Method Example 2 – mouseMoved()

```
void setup() {
   size(400, 400);
   background(0);
   textAlign(CENTER);
   textSize(24);
   fill(255);
   text("mouse has done nothing", width/2, height/2);
}

void draw(){
}
```

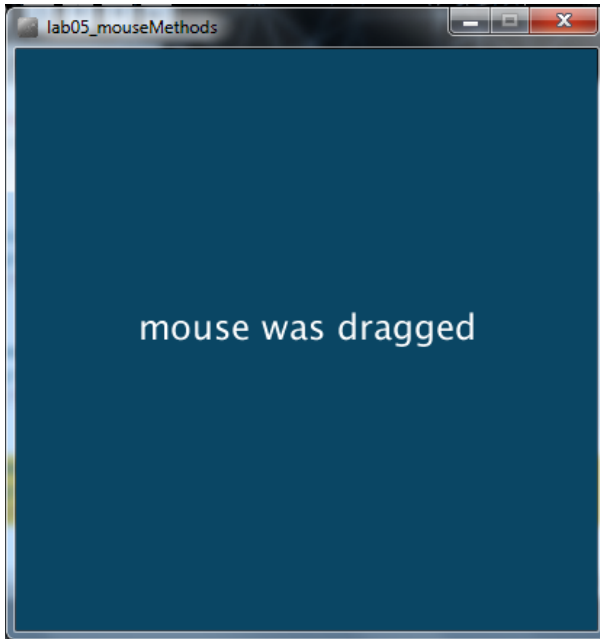



```
void mouseMoved() {
    background(150, 10, 70);
    text("mouse was moved", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Method Example 2 – mouseDragged()

```
void setup() {
   size(400, 400);
   background(0);
   textAlign(CENTER);
   textSize(24);
   fill(255);
   text("mouse has done nothing", width/2, height/2);
}

void draw(){
}
```

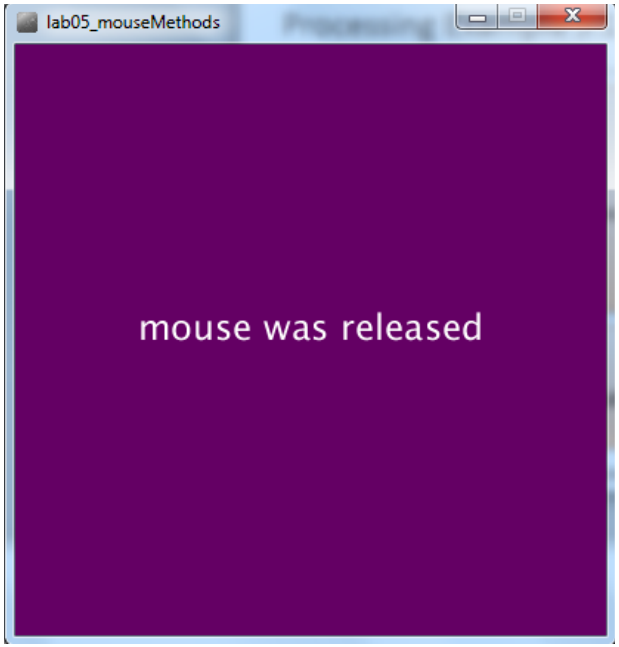



```
void mouseDragged() {
    background(10, 70, 100);
    text("mouse was dragged", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Method Example 2 – mouseReleased()

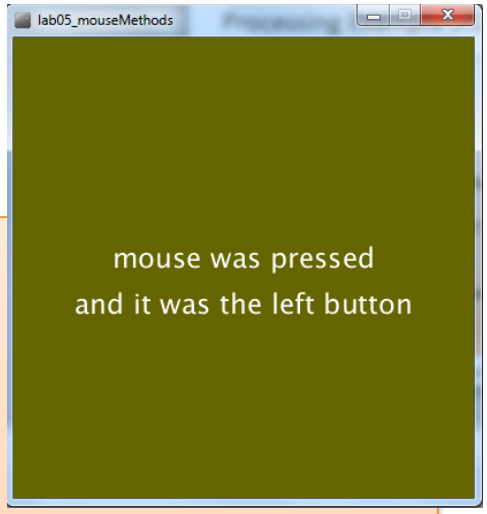
```
void setup() {
   size(400, 400);
   background(0);
   textAlign(CENTER);
   textSize(24);
   fill(255);
   text("mouse has done nothing", width/2, height/2);
}

void draw(){
}
```




```
void mouseReleased() {
    background(100, 0, 100);
    text("mouse was released", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Method Example 2 – mousePressed ()




```
void mousePressed() {
   background(100, 100, 0);
   text("mouse was pressed", width/2, height/2);
   if ( mouseButton == LEFT) {
     text("and it was the left button", width/2, height/2 + 40);
   }
   if (mouseButton == RIGHT) {
     text("and it was the right button", width/2, height/2 + 40);
   }
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

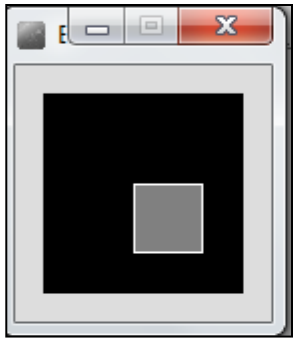
Some Previous Exercises

- We will now re-work the following examples that we covered previously:
  - Mouse Pressed Example 1
  - Example 3.6
  - Example 3.7
  - Example 3.8

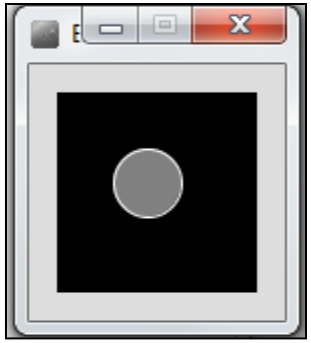
- Each of these exercises tested the mousePressed variable. Now we want them to use the mousePressed() method instead.

Recap: Processing Example 3.5

- Functionality:
  - If the mouse is pressed, draw a gray square with a white outline.



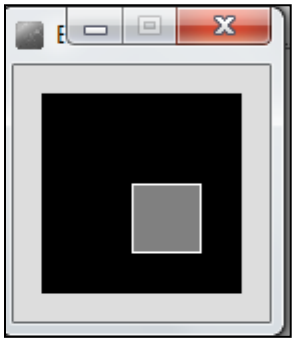
  - Otherwise draw a gray circle with a white outline.

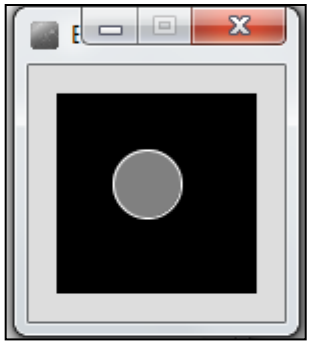


Recap: Processing Example 3.5

```
void setup() {
   size(100,100);
   stroke(255);
   fill(128);
}
```

```
void draw() {
   background(0);
   if (mousePressed == true)
   {
       rect(45,45,34,34);
   }
   else{
       ellipse(45,45,34,34);
   }
}
```





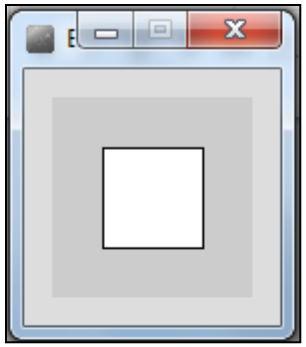
Using Mouse Methods…

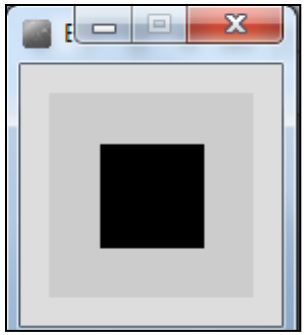
```
void setup()
{
  size(100,100);
  stroke(255);
  fill(150);
  background(0);
  ellipse(45,45,34,34);
}

void draw(){
}
```

```
void mousePressed(){
  background(0);
  rect(45,45,34,34);
}

void mouseReleased(){
  background(0);
  ellipse(45,45,34,34);
}
```

Recap: Processing Example 3.6

- Functionality:
  - If the mouse is pressed, set the fill to white and draw a square.
  - Otherwise set the fill to black and draw a square.

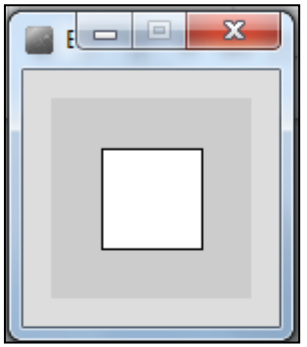


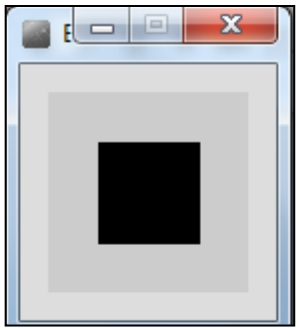


Recap: Processing Example 3.6

```
void setup() {
  size(100,100);
}
```

```
void draw() {
  background(204);
  if (mousePressed == true)
  {
       fill(255); // white
   } else {
       fill(0);     // black
   }
   rect(25, 25, 50, 50);
}
```





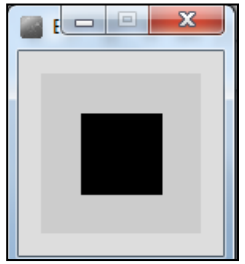
Using Mouse Methods…

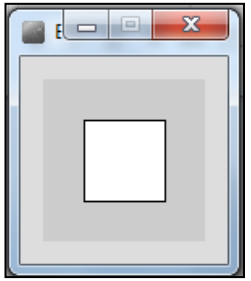
```
void setup()
{
  size(100,100);
  background(204);
  fill(0);
}

void draw()
{
  rect(25, 25, 50, 50);
}
```

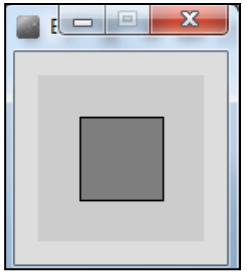
```
void mousePressed()
{
  fill(255);
}

void mouseReleased()
{
  fill(0);
}
```

Recap: Processing Example 3.7

- Functionality:
  - If the LEFT button on the mouse is pressed, set the fill to black and draw a square. As soon as the LEFT button is released, gray fill the square.
  - If the RIGHT button on the mouse is pressed, set the fill to white and draw a square. As soon as the RIGHT button is released, gray fill the square.
  - If no mouse button is pressed, set the fill to gray and draw a square.



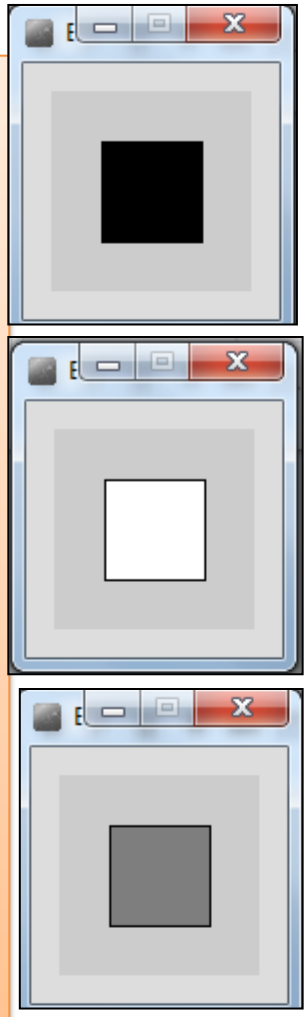




Recap: Processing Example 3.7

```
void setup() {
  size(100,100);
}
```

```
void draw() {
   if (mousePressed == true)
   {
       if (mouseButton == LEFT)
        {
            fill(0);        // black
        }
       else if (mouseButton == RIGHT)
        {
            fill(255);    // white
        }
   }
   else
   {
       fill(126);        // gray
    }
    rect(25, 25, 50, 50);
}
```



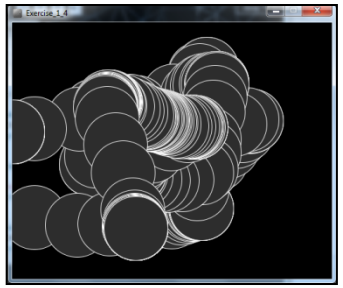
Using mouse methods…

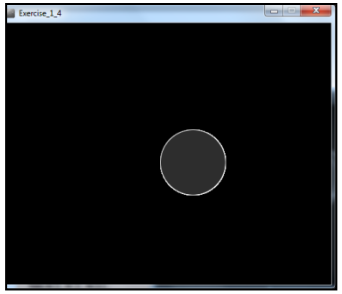
```
void setup()
{
  size(100,100);
  background(204);
  fill(126);
}

void draw(){
  rect(25, 25, 50, 50);
}
```

```
void mousePressed()
{
  if (mouseButton == LEFT)
  {
    fill(0); // black
  }
  else if (mouseButton == RIGHT)
  {
    fill(255);    // white
  }
void mouseReleased()
{
  fill(126);
}
```

Recap: Processing Example 3.8

- Functionality:
  - Draw a circle on the mouse (x,y) coordinates.
  - Each time you move the mouse, draw a new circle.
  - All the circles remain in the sketch until you press a mouse button.
  - When you press a mouse button, the sketch is cleared and a single circle is drawn at the mouse (x,y) coordinates.

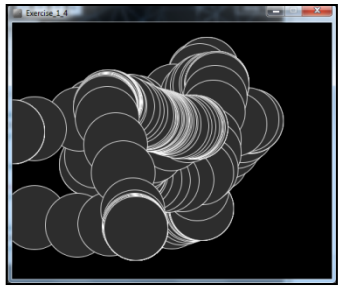


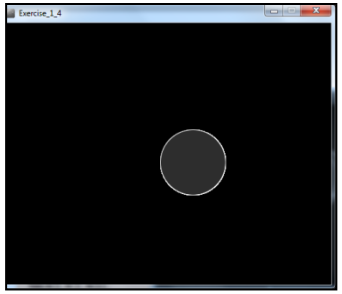


Recap: Processing Example 3.8

```
void setup() {
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
}
```

```
void draw()
{
  if (mousePressed == true)
  {
    background(0);
  }
  ellipse(mouseX, mouseY, 100, 100);
}
```





https://processing.org/tutorials/interactivity/

Using mouse methods…version 1

```
void setup()
{
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
}

void draw()
{
   ellipse(mouseX, mouseY, 100, 100);
}
```

```
void mousePressed()
{
   background(0);
}
```

Using Mouse Methods…Version 2

```
void setup()
{
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
}

void draw()
{
}
```

```
void mouseMoved()
{
   ellipse(mouseX, mouseY, 100, 100);
}

void mouseClicked()
{
  background(0);
  ellipse(mouseX, mouseY, 100, 100);
}
```

Questions?

Produced by:

Dr. Siobhán Drohan
Mairead Meagher
Sinéad Walsh

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/