A Virtual Joystick in Excel

by George Lungu

<excelunusual.com>

Introduction:

Focusing on a programming language takes one's eyes off the ball. This website is based on Excel because Excel is very easy to use without much prior knowledge. The main purpose of the blog is not to teach you Excel, VBA, Math, Physics, Geometry or Trigonometry but to challenge you to think. The best teacher is locked up inside your head. You must just learn to trust him and he will help you solve any problem.

Two years back I built a Tetris game in Excel using buttons on the screen as controls. I also tried to use shortcut keys but their functionality seemed to be somewhat unreliable and problematic in the sense that the action of the keys was occasionally delayed.

A little later I made a roller coaster and my next objective was to build a flight simulator game in Excel. For something like that I really needed to simultaneously and finely control two angles, the roll angle and the pitch angle.

The opportunity eventually showed up when I found a VBA code snippet on the internet which retrieved the X, Y position of the mouse on the screen. And that's what I was looking for ...

A plan:

- This tutorial has three parts, the first part contains a short code which prints the absolute mouse coordinates of the cursor on the screen, the second part demonstrates a macro which retrieves relative cursor coordinates from an initial click point, and the third worksheet will contain a complete virtual joystick.
- You don't need to understand 100% of the code, just save the file somewhere else and when you need this function use it. Again, the main purpose of this blog is not to teach you any programming language but to motivate you to focus on building interesting applications on your own.

Retrieving absolute coordinates – *check out "Windows API"* on Wikipedia

The following are declarations:

```
Public Declare Function GetCursorPos _
        Lib "user32" (Some_String As POINTAPI) As Long
```

Declaration of a special API (Application Programming Interface) function which retrieves the cursor position

```
Type POINTAPI
   X  As Long
   Y  As Long
End Type
```

Declaration of a special structure (Point API) used as the output type of the previous API function. It is essentially the pair of coordinates (as long integers) of the cursor on the screen started to be measured from the upper left corner of the screen.

The first macro – absolute mouse coordinates retrieval:

- In the VBA editor insert a module (named Module1). Write both, the previous declarations and the following macro in Module1.
- This macro will print the cursor coordinates as an infinite time loop (about 50-500 times a second) in cell B5 and C5 respectively:

```
Sub GetCoordinate()
Dim Pt As POINTAPI
    Do
    DoEvents
    GetCursorPos Pt
    [B5] = Pt.X
    [C5] = Pt.Y
    Loop
End Sub
```

Declaration of the macro name and return (nothing)

Declaration of the string “Pt” as a Point API type structure

Infinite “Do” loop declaration (start)

Always add this statement if you ever need to stop the loop manually or update a chart while the loop is running

The “Pt” structure is given a value by the API function “GetCursorPos” during each “Do” loop cycle

The cursor position is printed in the range “B5:C5”

End of loop declaration

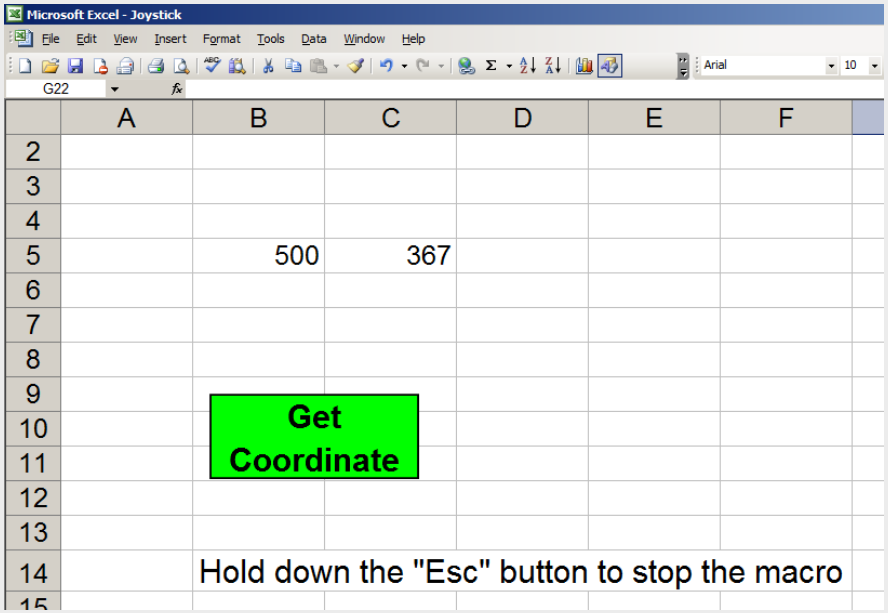
End of macro declaration

Let's create a start button for this macro:

- In the drawing toolbar: choose "Rectangle" -> draw a rectangle
- Change the color of the square and add the text (I used "Get Coordinates")
- Right click the button -> Assign Macro -> choose "Get Coordinate" from the list -> OK

Testing the macro:

- Click the button and observe how the numbers in cells (B5,C5) change to reflect the position of the cursor as we move the mouse
- Observe how fast and precisely the updating is done
- Hold down the "Escape" button to stop the macro



The second macro – relative mouse coordinates retrieval:

- Inserting a new worksheet we'll write the next macro in Module1 under the first macro
- This macro will print the cursor coordinates as an infinite time loop (about 50-500 times a second) in cell B5 and C5 respectively, relative to the point on the screen where the start button was first clicked
- After the macro is created we will create a start button in a second different sheet (named "GetRelativeCoordinates") where we can test the macro

```
Sub GetRelativeCoordinate()
Dim Pt0 As POINTAPI
Dim Pt1 As POINTAPI

GetCursorPos Pt0
    Do
    DoEvents
    GetCursorPos Pt1
    [B5] = Pt1.X - Pt0.X
    [C5] = Pt0.Y – Pt1.Y
    Loop
End Sub
```

Declaration of the macro

Declaration two Point API type structures, one as initial click coordinates and the second as the current (dynamic) cursor coordinates

Assigns "Pt0" the initial click coordinates

Infinite "Do" loop declaration (start)

Always add this statement if you ever need to stop the loop manually or update a chart while the loop is running

Every loop cycle assigns "Pt1" the cursor coordinates

Every loop cycle calculate the relative coordinates and display them in the range "B5:C5" (figure out why I wrote those formulas the way I wrote them)

End of "Do" loop declaration

End of macro declaration

The third macro – the virtual joystick:

- Inserting a new worksheet we write the next macro in Module1 under the previous macro. On top we declare a Boolean variable which has the role of a “switch”, keeping track if the macro is running or is stopped. This will allow the macro to be started or stopped using the same button.

```
Dim RunPause As Boolean
```


- This macro will print the cursor coordinates as an infinite time loop (about 50-500 times a second) in cell B5 and C5 respectively, relative to the point on the screen where the start button was first clicked and it will also move the image of a joystick on a chart.

```
Sub JoyStick()
Dim Pt0 As POINTAPI
Dim Pt1 As POINTAPI
RunPause = Not RunPause

GetCursorPos Pt0
  Do While RunPause = True
  DoEvents
   GetCursorPos Pt1
   [B5] = Pt1.X - Pt0.X
   [C5] = -Pt1.Y + Pt0.Y
   Loop
End Sub
```

Declaration of the macro

Declaration two Point API type structures, one as initial click coordinates and the second as the current (dynamic) cursor coordinates

Boolean “flip”, if the macro is stopped this will start it and vice versa

Assigns variable “Pt0” the initial click coordinates

Conditional “Do” loop declaration (start)

Always add this statement if you ever need to stop the loop manually or update a chart while the loop is running

Every loop cycle assigns “Pt1” the cursor coordinates

Every loop cycle calculate the relative coordinates and display them in the range “B5:C5” (figure out why I wrote those formulas the way I wrote them)

End of “Do” loop declaration

End of macro declaration

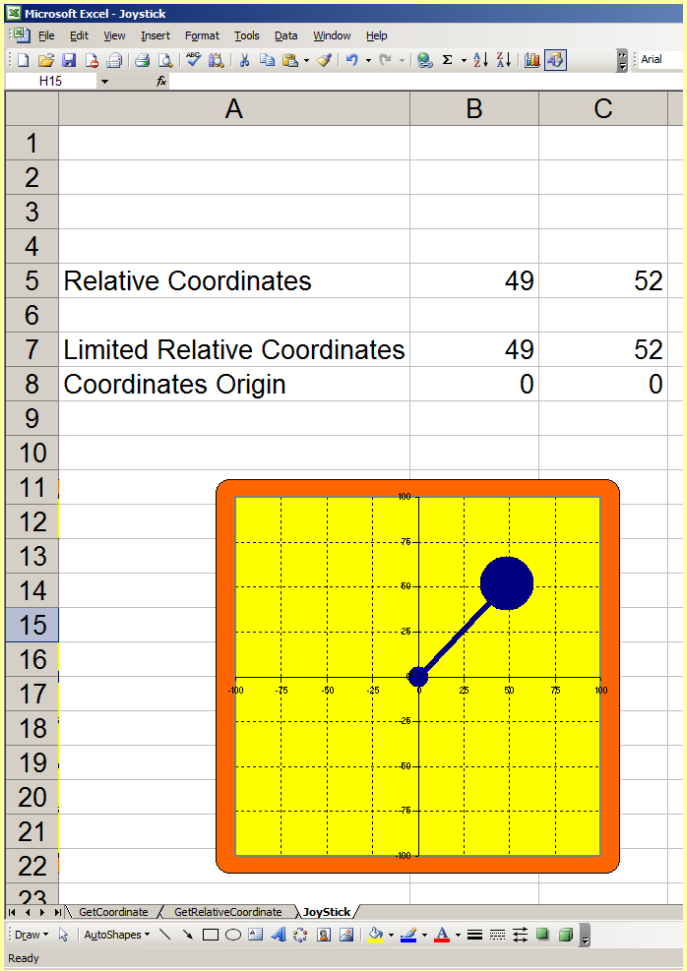
The virtual joystick the spreadsheet:

- While the macro “JoyStick()” is running, it will dynamically update the relative coordinates of the cursor on the screen the cells B5 and C5

- Cells B7 and C7 contain the relative coordinates taken from cells B5, C5 but limited to the interval [-100, 100]

  - B7: “ =IF(B5<0,MAX(-100,B5),MIN(B5,100))”
  - C7: “ =IF(C5<0,MAX(-100,C5),MIN(C5,100))”

- Set cells B8 and C8 to constant zero
- Create a 2D scatter chart with [B7:B8] as the X axis values and [C7:C8] as the Y axis values
- Delete any chart legend or title and change the color of the plot area to your preference
- Size the chart as a square and also adjust the range of both axes to [-100, 100]
- Right click the chart -> “Assign Macro” -> select “JoyStick()”



Usage:

1. If the macro is stopped, click the chart to start it. It is a good idea to position the pointer in the origin of the chart before clicking, this way the cursor will roughly be positioned on the joystick head on the chart. The joystick position is controlled by the mouse movement.
2. To stop the macro click the chart again

Conclusions:

- A virtual smooth joystick macro controlled by mouse movements has been demonstrated
- While in operation, the macro updates the coordinates of the pointer in two spreadsheet cells
- The two coordinates are then incorporated in formulas which limit the range of motion of the joystick. Just like a real device the stick head movement is confined within a square (with a side length of 200 pixels) and so is the numerical output (limited to [-100,100] on both X and Y)
- Immediate applications of this joystick are: a race car game (acceleration, brake and steering), a flight simulator (emulates the cockpit yoke or control stick – Roll+Pitch), a shooting game (azimuth and altitude cannon control), or a combination of the above applications