# Excel PONG Tutorial #7

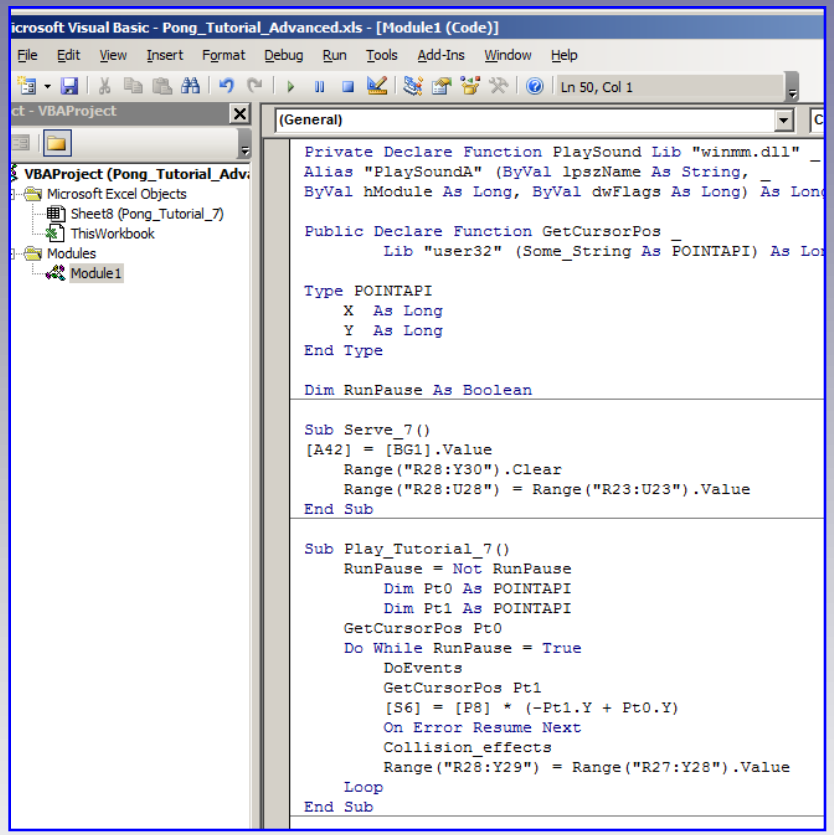
by George Lungu

## – associating sound effects to ball collision events


- The previous tutorial showed how to add adjustable skill levels to the virtual pong opponent. This section (which is a continuation of part#6) adds three different sound effects to following events: ball bouncing of the horizontal walls, ball bouncing off the bats and ball missing one of the bats (when score is supposed to change). In the next tutorial a score display scheme will be introduced.

# 42. Create a new copy of the workbook in a new directory

- Create a new folder called “Pong_Tutorial_Archive”
- In the new folder, save a copy of the Excel Pong file (as Pong_Tutorial_Advanced.xls) plus a number of four sound effects in the .wav format downloaded off the internet.
- In the new workbook delete all the worksheets except the last one (Pong_Tutorial_6).
- Change the name of the worksheet to “Pong_Tutorial_7”.
- Delete all the code in Module1 except for the API function declaration on the top of the page.
- Copy all the code from Module3 into Module1 under the API function declaration, then delete Module2 and Module 3 and now you are left with a single worksheet and a single module.
-Change the macro names to: “Serve_7” and “Play_Tutorial_7”
- Verify the functionality of the game




## Sound macro theory

- Among other .wav files in the same directory with the Excel file there are four important sound files:
  - crowd_applause.wav - *used when the ball is missed by Bat #1 (the player scores)*
  - crowd_laugh.wav - *used when the ball is missed by Bat #2 (the opponent scores)*
  - wall_bounce - *used when the bal bounces off the upper or the lower wall*
  - bat_bounce - *used when the ball bounces off either of the bats*

- In order to create a VBA macro which plays sound from a .wav file you need to start the VBA code with the PlaySound API function declaration, which all has to be typed in one line on the top of the page (an underscore will physically break the line in two but it will keep the 1-line code functionality):

```
Private Declare Function PlaySound Lib "winmm.dll"_
    (ByVal  someName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

Arguments:

Indicates the file name and file path to play

Specifies options for playing the sound using one or more of the flags explained in the next page

Besides the first argument which is the file name and path, this function has flag arguments by which its run options can be controlled. You could declare these as constants as follows (or just use the number):

**SND_SYNC = &H0** - The sound is played synchronously and the function does not return until the sound ends.

**SND_ASYNC = &H1** - The sound is played asynchronously and the function returns immediately after beginning the sound.

**SND_NODEFAULT = &H2** - If the sound cannot be found, the function returns silently without playing the default sound.

**SND_LOOP = &H8** - The sound will continue to play repeatedly until PlaySound is called again. You must also specify the SND_ASYNC flag to loop sounds (use the compound condition: SND_ASYNC or SND_LOOP).

**SND_NOSTOP = &H10** - If a sound is currently playing, the function continue playing the old sound and it will immediately return False without playing the newly requested sound.

*-Out of the above list of flags we will use &H1 since we want the action of the game to be continued while sound effects are played (if we used &H0 the action will freeze until the sound effect ends – I recommend you change &H1 with &H0 and see the effect for yourself).*

*- We could also use &H2 in order to make our code simpler (by giving a file a name that doesn't exist when we need the macro silent). The effect of using this flag could possibly simplify the code by saving an IF statement.*

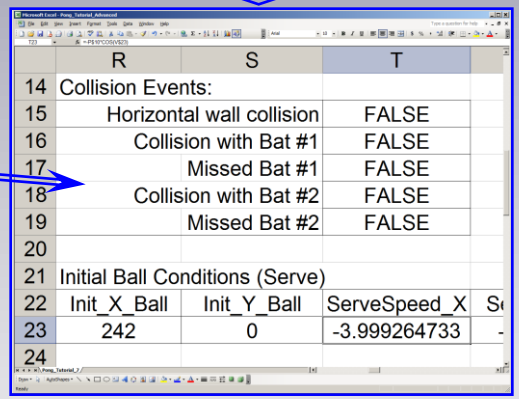
# 43. Create the sound macro

- This is an auxiliary macro which will be used by the main “Play_Tutorial_7” macro during each loop cycle.
- This macro is named “Collision_Effects_7” and it will be upgraded later to include score effects.

```
Private Declare Function PlaySound Lib "winmm.dll" (ByVal someName _
As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long


Sub Collision_Effects_7()
    If [T15] Then Call PlaySound(ThisWorkbook.Path & "\wall_bounce.wav", 0&, &H1)
    If [T16] Or [T18] Then Call PlaySound(ThisWorkbook.Path & "\bat_bounce.wav", 0&, &H1)
    If [T17] Then Call PlaySound(ThisWorkbook.Path & "\crowd_applause.wav", 0&, &H1)
    If [T19] Then Call PlaySound(ThisWorkbook.Path & "\crowd_laugh.wav", 0&, &H1)
    On Error Resume Next
End Sub
```

*While studying the macro make sure you consult the Collision Events table*

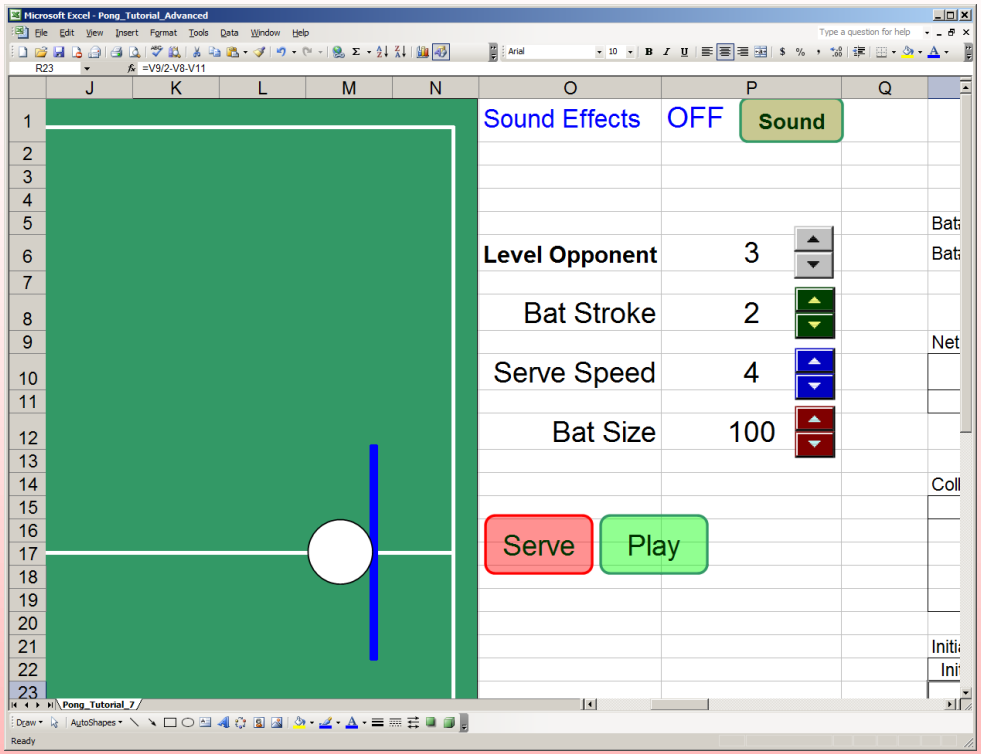


| | R | S | T |
|---|---|---|---|
| 14 | Collision Events: | | |
| 15 | Horizontal wall collision | | FALSE |
| 16 | Collision with Bat #1 | | FALSE |
| 17 | Missed Bat #1 | | FALSE |
| 18 | Collision with Bat #2 | | FALSE |
| 19 | Missed Bat #2 | | FALSE |
| 20 | | | |
| 21 | Initial Ball Conditions (Serve) | | |
| 22 | Init_X_Ball | Init_Y_Ball | ServeSpeed_X |
| 23 | 242 | 0 | -3.999264733 |
| 24 | | | |

- The results in the Collision Events table are used as conditions for selecting which macro plays at that time.
- If all the results in the collision table are “FALSE”, no sound will play, if the horizontal wall collision event is “TRUE” then the “wall_bounce.wav” will play and so on.
- We chose the asynchronous flag in the sound function to make sure the game does not freeze for the duration of the sound effect

# 44. Create an "Enable_Sounds" button for the sound effects

- In order to be able to turn the sounds on and off, a button is created which toggles the value of cell P1 between "ON" and "OFF"




- The P1 cell value will be used in the "Play_Tutorial_7" macro to enable or disable the sound effects as it will be shown in the next page.
- All the macros associated to the current sheet are displayed to the right (Module1 macros are not shown here).
- The last macro ("Enable_Sounds" in red) is the newly added macro.

```
Private Sub Level_Opponent_Change()
Range("P6") = Level_Opponent.Value
End Sub

Sub Bat_Stroke_Change()
Range("P8") = Bat_Stroke.Value
End Sub

Private Sub Serve_Speed_Change()
Range("P10") = Serve_Speed.Value
End Sub

Private Sub Bat_Size_Change()
Dim BArr As Variant
BArr = Array(2, 5, 10, 15, 20, 40)
[P12] = 10 * BArr(Bat_Size.Value)
End Sub

Sub Enable_Sounds()
    If [P1] = "ON" Then
    [P1] = "OFF"
    Else
    [P1] = "ON"
    End If
End Sub
```

# 45. Integrate the sound macro within the Module1

- This is a snapshot of the Module1 code
- On the very top in yellow is the declaration of the sound function
- Below that still in yellow is the declaration of the mouse position API function and the structure PointAPI
- The last yellow line is a declaration of a “latch” variable – ”RunPause”. This variable allows the “Play_Tutorial_7” macro to be started and stopped from the same button.
- The serve, play and collision effects macros are situated below, each with its own color

```
Private Declare Function PlaySound Lib "winmm.dll" (ByVal someName _
As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long

Public Declare Function GetCursorPos _
      Lib "user32" (Some_String As POINTAPI) As Long
Type POINTAPI
   X  As Long
   Y  As Long
End Type
Dim RunPause As Boolean

Sub Serve_7()
[A42] = [BG1].Value
   Range("R28:Y30").Clear
   Range("R28:U28") = Range("R23:U23").Value
End Sub

Sub Play_Tutorial_7()
   RunPause = Not RunPause
      Dim Pt0 As POINTAPI
      Dim Pt1 As POINTAPI
   GetCursorPos Pt0
   Do While RunPause = True
      DoEvents
      GetCursorPos Pt1
      [S6] = [P8] * (-Pt1.Y + Pt0.Y)
      On Error Resume Next
      If [P1]="ON" Then Collision_Effects_7
      Range("R28:Y29") = Range("R27:Y28").Value
   Loop
End Sub

Sub Collision_Effects_7()
If [T15] Then Call PlaySound(ThisWorkbook.Path & "\wall_bounce.wav", 0&, &H1)
If [T16] Or [T18] Then Call PlaySound(ThisWorkbook.Path & "\bat_bounce.wav", 0&, &H1)
If [T17] Then Call PlaySound(ThisWorkbook.Path & "\crowd_applause.wav", 0&, &H1)
If [T19] Then Call PlaySound(ThisWorkbook.Path & "\crowd_laugh.wav", 0&, &H1)
On Error Resume Next
End Sub
```

*Conditional call of the auxiliary macro “Collision_Efects_7” macro within the main “Play_Tutorial_7” macro*

***“On Error Resume Next”*** *Specifies that when a run-time error occurs, control goes to the statement immediately following the statement where the error occurred where execution continues. It is useful while handling multiple sound effects.*