**The TICTACUFO Office Scripts Game . A Demo**

**Personal Introduction**

I am Leslie Black a Microsoft Applications Specialist working for Analysis Cloud, the Irish Cloud Company. See [www.analysis.ie](http://www.analysis.ie)

I have experience working with governmental organisations in Europe and the UK building reporting, analysis and performance management systems using Microsoft tools.

I have recently finished working with the European Commission building a large scale, 100%, cloud based Covid Reporting application built on Azure, Power Platform and Office 365. It was on that project that I became familiar with Office Scripts.

**The key messages for this demo are:**

1. Think of Excel Online as a calculation engine available to assist your built online apps.
2. Complex automation tasks can be written using Office Scripts but Power Automate can reduce that complexity by handling the logic instead. It is your choice.
3. Excel Online has the power and capability to be used a front-end user interface for your web-based applications.
4. Because Excel Office Scripts is integrated with Power Automate it opens up a whole world of Azure Services and data sources available for your solutions.

**Links**

* See the demo files at my GitHub Repo:

<https://github.com/MrAnalyticals/OfficeScripts/tree/main/TICTACUFOGame>

* See the video demonstrating the app on my YouTube Channel at : <https://youtu.be/FMOCPm1aV_Q>
* See the associated code-review video here : TO DO.
* Analysis Cloud Limited, The Irish Cloud Company, www.analysis.ie

**The TIC TAC UFO Game**

Worksheet “TICTAC” is the user interface to play the game. Options available are :

* You play first
* Excel plays first
* Clear board and start a new game

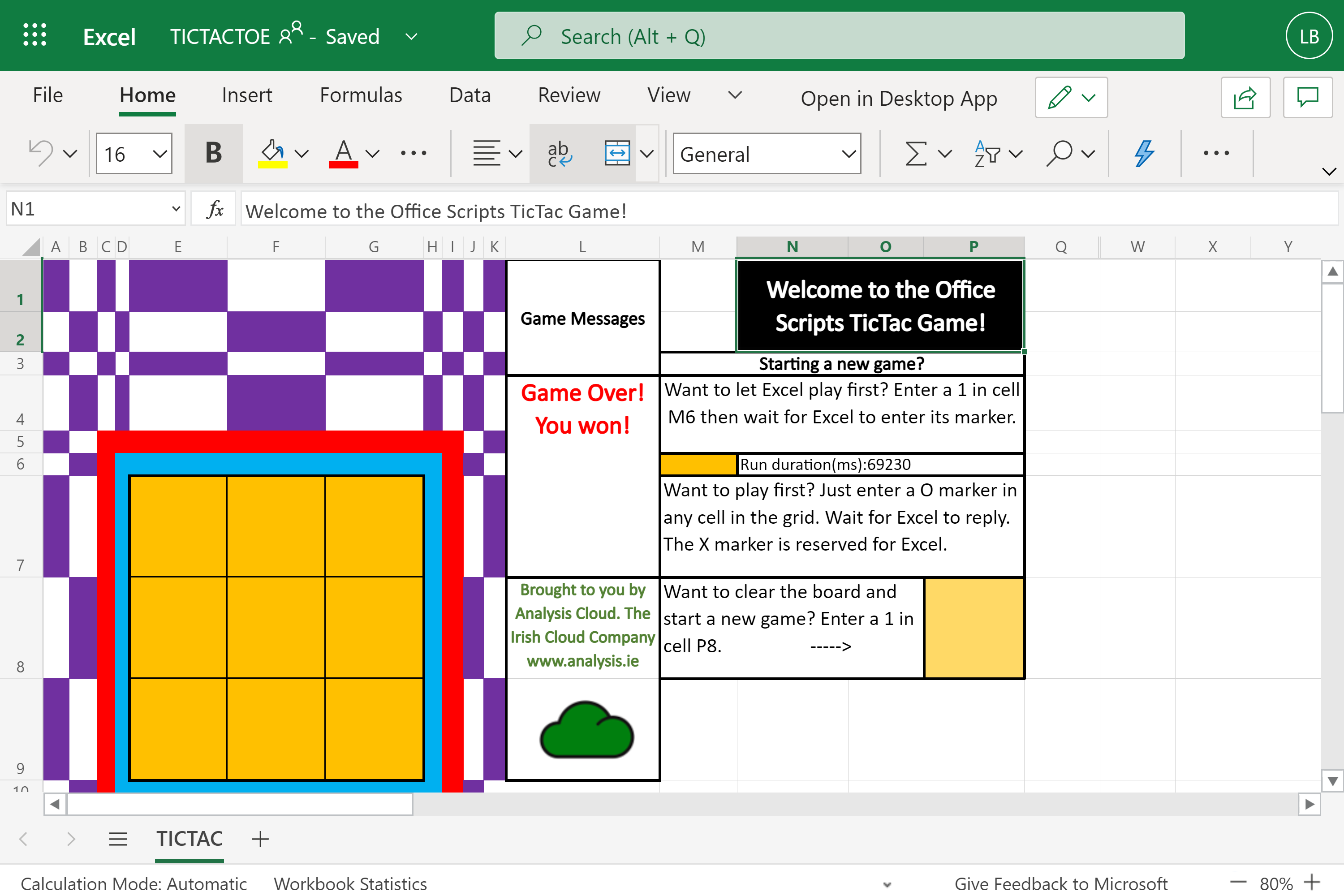
You enter a marker into the grid and Excel responds by putting an ‘X’ marker into the grid.

You cannot enter an ’X’ marker as that is reserved for Excel. You can use any marker that is not an ‘x’ or ‘X’ and the markers can be different from each other.

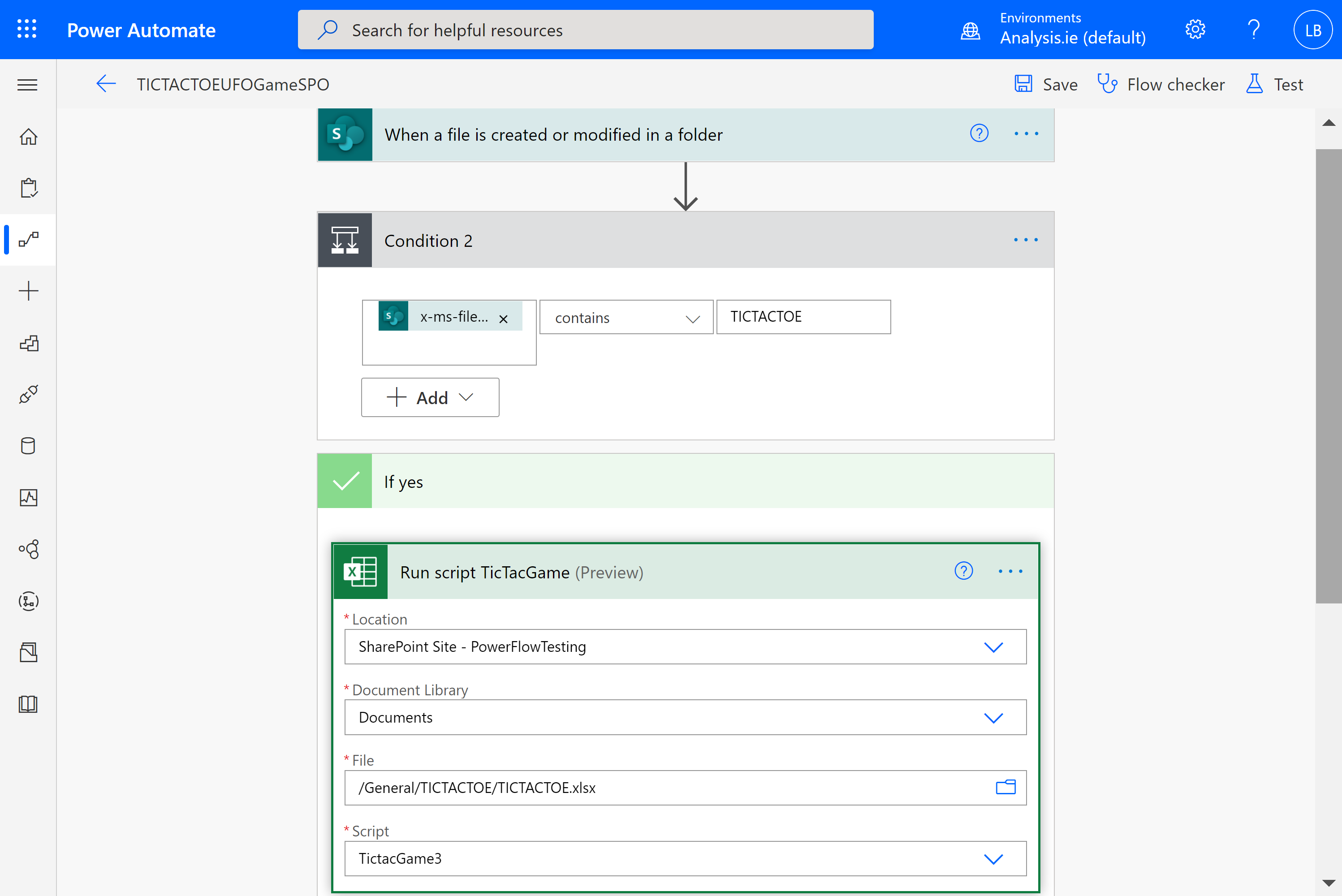
When the user wins, i.e. Excel loses, an animation starts.

The Sharepoint “When a file is created or modified in a folder” connector is used for the trigger event. The time for Power Automate to respond to the entering of a marker is in addition to the time it takes for the script to run. The script takes, on average, around 70 seconds (70,000 miliseconds). The Power Automate trigger ranges from 10 seconds to 110 seconds to run. So, Excel can take up to 3 minutes to reply with its marker. In the demo video I have speeded up the time between marker events.

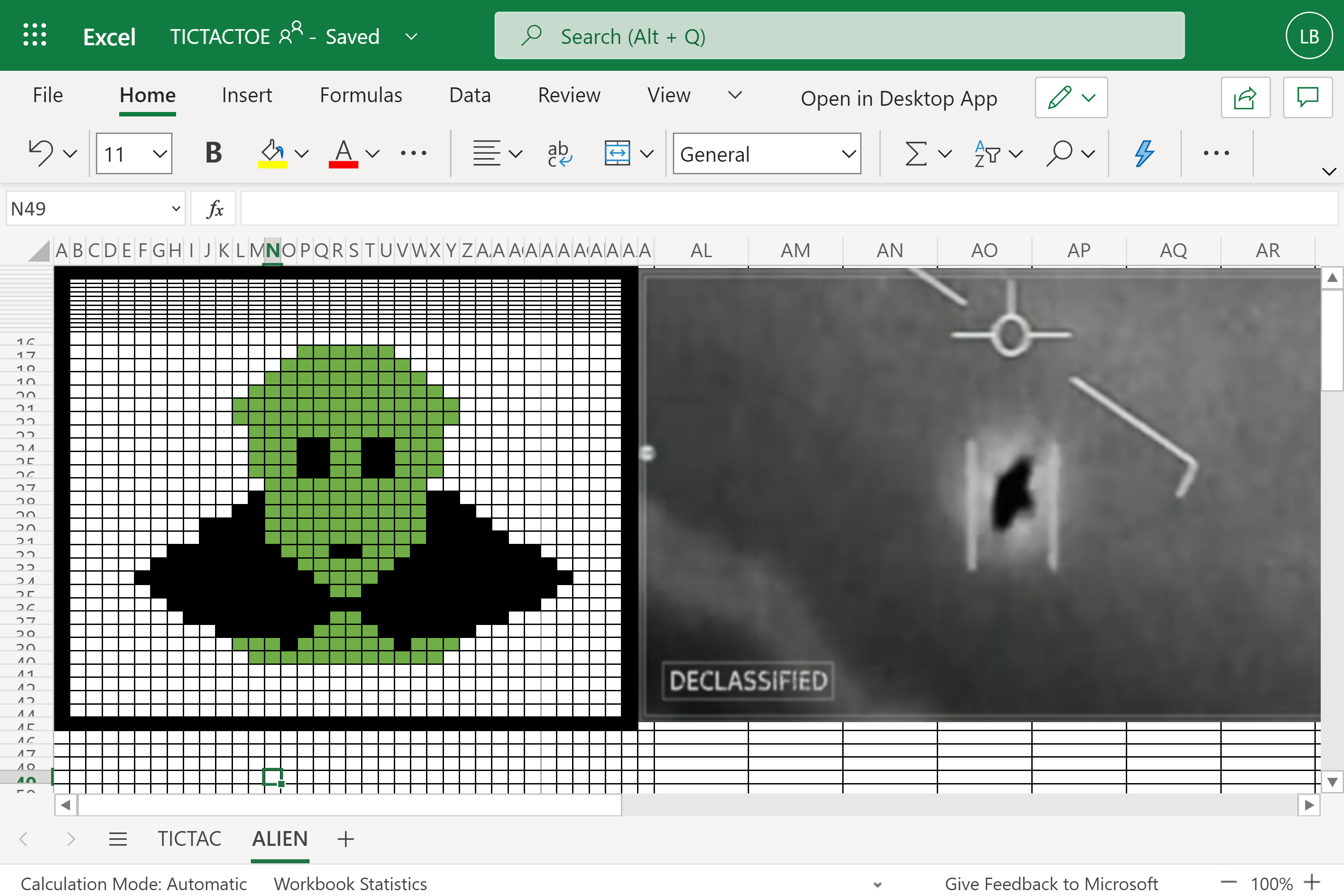
**Fig 1. The Tic Tac UFO Game User Interface.**



The associated Power Automate Flow contains only three connectors, the trigger (SharePoint “When a file is created or modified in a folder”) and two action steps : a condition and an Excel “Run Script”.

**Fig 2. Tic Tac UFO Game Power Automate Flow**

With the Excel macro recorder building solutions is, now, extremely rapid. The animation element of this app was built in less than half a working day.

**Fig 3. Animation code was built using the macro recorder**

**Fig 4. Screenshot of the Tic tac game being built.** Graphical user interface, application

Description automatically generated

**Code Review**

The script is 3,002 lines long. 1,977 lines, of which, are for the animation alone.

The functions, gridLocation and gridLocationCol1 functions are two no-longer needed functions and total of 65 lines.

So, the total number of necessary lines of code in this script (excluding the animation) is 960. This is a surprisingly large number considering all we are trying to do, with this game, is to put an ‘X’ in an empty cell in the grid. But that is not the full story!

**The full list of functions**

Here is the full list of the functions making up the script. Those functions, used for the animation, have been marked with a \*. The animation functions were built using the macro recorder and so, on that basis, they are trivial and will not be reviewed in this document.

\*denotes those functions used for animation only.

function main(workbook: ExcelScript.Workbook)

\*function flashColours(workbook: ExcelScript.Workbook)

\*function Red(workbook: ExcelScript.Workbook)

\*function Gold(workbook: ExcelScript.Workbook)

\*function Purple(workbook: ExcelScript.Workbook)

\*function Yellow(workbook: ExcelScript.Workbook)

\*function Green(workbook: ExcelScript.Workbook)

function Linesofthree(workbook: ExcelScript.Workbook)

function GetRangeCount(workbook: ExcelScript.Workbook)

function EnterMarker(workbook: ExcelScript.Workbook)

function LineofThreeXs(workbook: ExcelScript.Workbook)

\*function DisplayGreenMan(workbook: ExcelScript.Workbook)

\*function BlackCircle(workbook: ExcelScript.Workbook)

\*function GreenMan(workbook: ExcelScript.Workbook)

function AddMarkerLocationToTable1(workbook: ExcelScript.Workbook, location: string)

function findLocation(workbook: ExcelScript.Workbook)

function GetRangeCountNotX(workbook: ExcelScript.Workbook)

function GetRangeCountX(workbook: ExcelScript.Workbook)

(not used) function gridLocationCol1(rowN: number, colN: number)

(not used) function gridLocation(rowN: number, colN: number)

function datetime(workbook: ExcelScript.Workbook)

function setDuration(workbook: ExcelScript.Workbook,startTime:number)

* The functions Linesofthree and LineofThreeXs are used within the Main function to test the existence of a line of markers. LineofThreeXs tests, specifically, for a line of three X’s.
* The EnterMarker function builds an array of the locations of the empty cells then randomly selects one of those locations and places an ‘X’ marker in that cell address.
* The GetRangeCount function counts the total number of markers entered in the grid.
* The findLocation function iterates through the grid finding a cell that is not listed in Table1. Table 1 is used to record the locations of the entered markers and the order in which they were added to the grid. It is hidden from user view.
* The AddMarkerLocationToTable1 function adds the recently added marker location to the next available row in Table1. It uses the cell address value returned by the findLocation function, above.
* The current number of markers added to the grid is compared to the number added to Table 1. Thus, the code can ascertain whether a human or Excel moved last. When Excel enters a marker, Table1 is not updated whereas when a human enters a marker the code does update Table1. So I am, in effect, using the Excel worksheet as a location to store Global variables between script runs. This could have been done using a List in SharePoint but the purpose of the application is to showcase the capabilities of Excel not Lists.
* Because Excel, currently, has only one trigger connector in Power Automate, which is not applicable to this scenario, it leaves, only, the SharePoint “When a file is created or modified in a folder” connector to use as the event trigger. This means any change to the workbook will trigger the Office Script. Even modifying an attribute of the file in SharePoint will trigger the Office Script. That is, it is not only changes to the workbook contents that triggers the Script. This means that there needs to be a test, in the Script, that ascertains the change, if it occurred and where in the worksheet it happened. Capturing the entering of a marker, by a human, has, already, been explained in the notes above.
* In order to allow the human player to choose to play first or to start a new game, the Script tests for a change in cells M6 and P8, as per the screenshot of the main user interface. In the main function the first lines of code tests the values in those two cells and then takes action accordingly. The main function is 369 lines long.
* The duration for each turn is calculated by the datetime and setDuration functions and is displayed in cell N6.