**Power Point Notes Document**

**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)

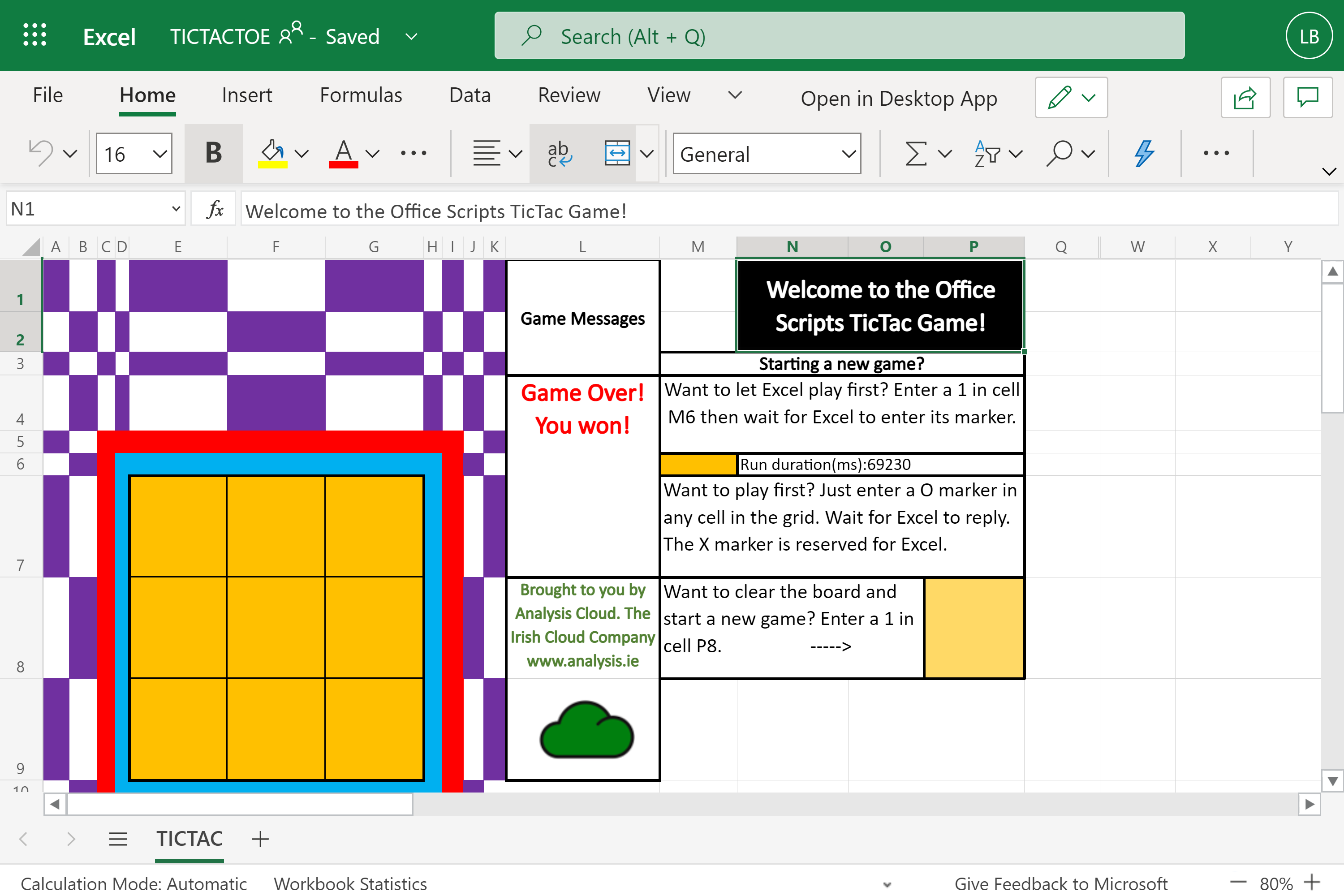
I have experience working with governmental organisations in Europe and the UK building reporting, analysis and performance management systems using Microsoft tools.

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 as 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.

**The TIC TAC UFO Game User Interface**



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

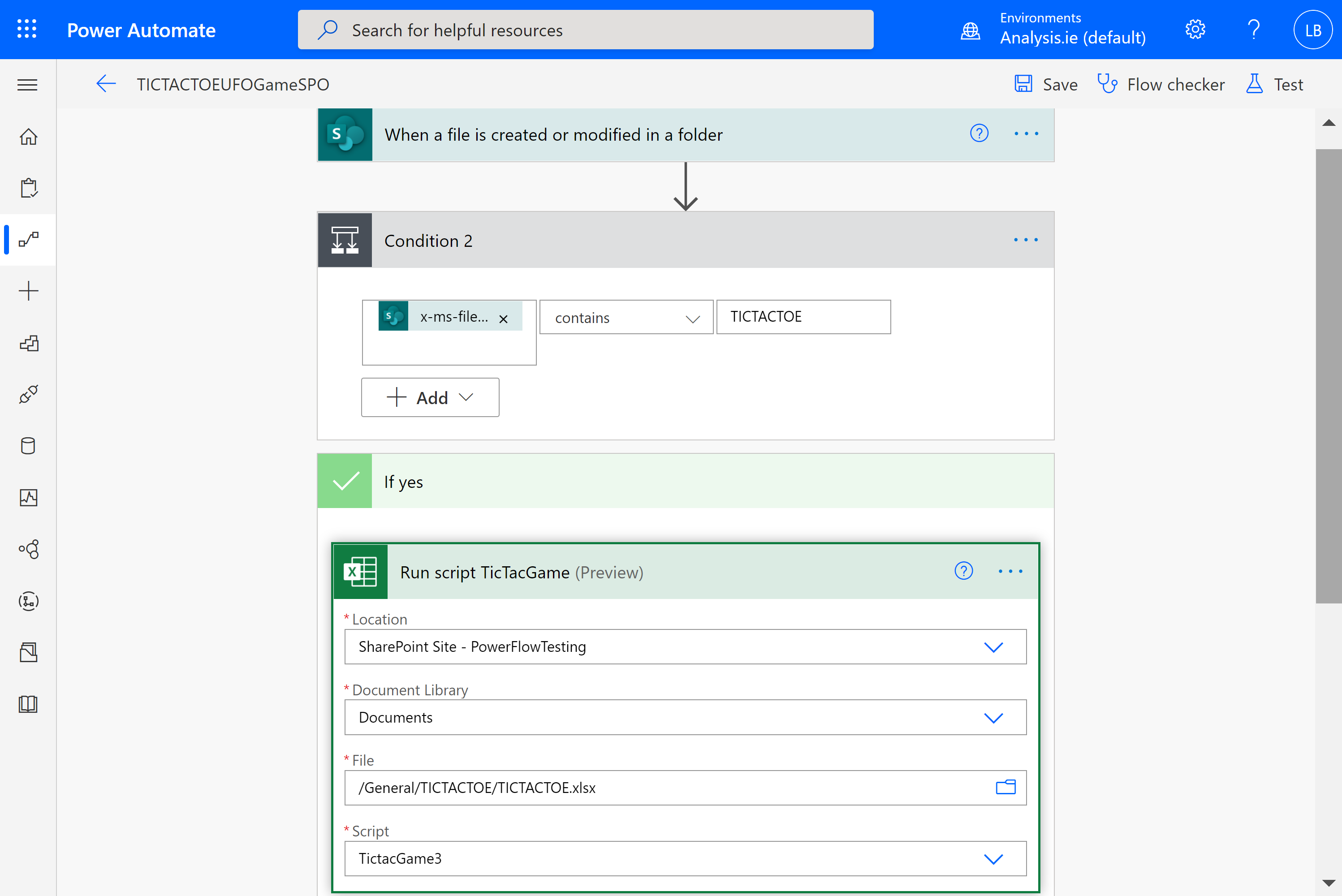
* You play first (in the Grid)
* Excel plays first (cell M6)
* Clear board and start a new game (cell P8)

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 time taken for Excel to make his turn is displayed in cell N6.

**Tic Tac UFO Game Power Automate Flow**

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”.

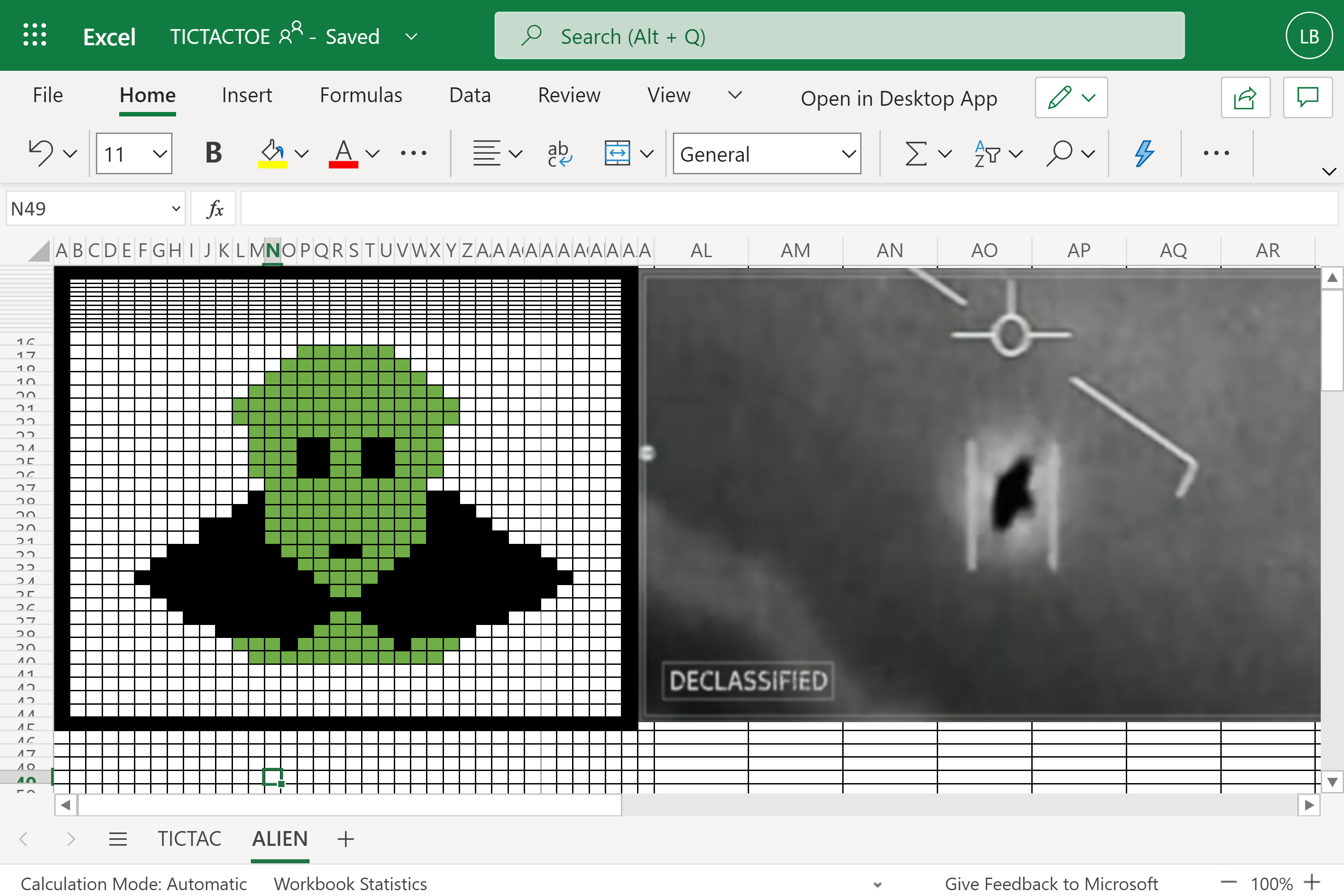
Because Excel, currently, has only one trigger connector in Power Automate, which is not applicable to this scenario, it leaves, only, the SharePoint file trigger to us as the application’s event trigger. This means any change to the workbook file will trigger the Office Script. Even modifying an attribute 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.

When a marker or any other change occurs in the worksheet the Script needs to ascertain where that change occurred and what type of change occurred. This is done by testing the values within the grid and cells M6 and P8.

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 this up.

Let’s, now, have a look at the application and get Excel to take a turn.

[SHOW EXCEL ONLINE TICTAC.XLSX FILE. ENTER YOUR MARKER AND THEN CLICK RUN. So while that is running lets continue with the presentation slides.]

**The Animation code was built using the macro recorder**

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

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. That is, 1977 lines of TypeScript code was written in less than 4 hours.

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

Description automatically generated

In the single script file there is a total of 20 functions including the Main function. The Main function is 369 lines long. Excluding the lines of code that control the animation, the total number of lines of code 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!

[Let’s go back to the application and see what it has done.]

**The full list of functions**

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)

function datetime(workbook: ExcelScript.Workbook)

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

\*denotes those functions used for animation only.

Here is the full list of the functions. Those functions, used for the animation, have been marked with a \*.

* 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.

**Thank you for attending this demo. Here are the Links**

* See the demo files at my GitHub Repo:

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

* App demo video https://youtu.be/FMOCPm1aV\_Q (Please Subscribe)
* See the associated code-review video here : TO DO.
* Analysis Cloud Limited, The Irish Cloud Company, www.analysis.ie

A picture containing clipart

Description automatically generated