**Matthew`s Games**

**A Website containing several games made using JavaScript, HTML and CSS.**

**By Matthew Cole**

**Project concept**

*See the About Us page for more detail*

This idea was first started in June 2021, when I was experimenting with Microbits and made a very basic game on them. It was a dodging game, similar to not-so-flappy Dot. I started to expand it, but was unable to due to the file size limit on the Microbits.

I came back to it after the October holidays and tried to make the same game on a website. it took some time to get started - changing the colour of a particular table cell proved more difficult than I expected. I have a bad habit with my programming in that I do not stop adding stuff to it, which often leads to there being aspects that are useless, or not needed. Not-so-flappy Dot did get that. There are many things in it that are not required to work. Probably the best addition was the addition of Highscores. This then needed a way of displaying the highscores, so an additional website was made. This site then evolved to become the main site today.

For the other games, I started making them because I wanted to improve my own programming skills. Tetris was for moving a collection of dots together. Minesweeper has the only use of a function that could call itself that I could think of. Dotman had the ghosts, which were meant to be kind of smart to move towards you.

I’m interested in this project because it is programming, but I can display it in a nice way with HTML and CSS. Compared to many of the other programming languages that are used in the real world, JavaScript was accessible for me and had a nice way of outputting what it is doing. Python may have been able to do this, but it was harder to get started.

The games are more about me trying to think around major programming aspects. Minesweeper makes use of recurring function (functions that call themselves), Tetris is about objects moving together (an object exists in more than one table cell, a concept I had not tried before now), and Dot Man (Pac Man) has Ghosts / NPCs / the computer has to think to go towards you. Not-so-flappy-Dot was more about the idea of getting a JavaScript game to work.

I had originally started my STEM project on a different theme, but I lost interest in it and dropped out of STEM. However, whilst making Not-so-Flappy Dot, with it working well, I though that I could re-join STEM with this. To ensure that I remembered what I did, for this write up, I started the About Us page and its History section, where I would add what I’ve done throughout the project. I wasn`t until March that I actually re-joined STEM, but by this point, I knew that I was enthusiastic about my website and I still want to do more to it.

**Project process**

*When it says ‘finish the game’ or make it a game, it includes adding a way to finish the game (ie You win message), a way to lose the game (i.e. you died), a way to start the game (a start button usually). Often also includes scores or lives, and timer to make the game move forward.*

There was not really a ‘plan’ as such to the project. It started off as way to expand a Microbit project and new parts were added when I thought of them.

Create Game 1 (not-so-flappy Dot)

* create table with ids for every cell
* create function that will change the specified ID to a different colour
* get Cells to change on a timer, then in a nice order
* create player
* put it all together and add final functions to make it a game

Create Website

* Nav bars
* Title / header
* Highscores page
* About us page - contains the history / diary of website
* log in aspect.

Create Dot man

* get Table and player working (similar to steps for not-so-flappy Dot)
* get NPCs working
* put all together

Create Minesweeper

* get the cascading uncover function working
* get the numberings working
* get the hiding mines function to be working
* finish the game

Create Tetris

* Get the objects to move as one
* get the objects to rotate - kind of cheated
* move them down each time
* get them to stop on the boundaries (important - could cause error if it is trying to plot an id that does not exists) and other places blocks
* make it a game

The website part, however, is constantly under development. the adjustable nav bar, highscores for the last three games was introduced later on, and the history section was constantly updated..

**21st Century Employability & Future Skills**

Throughout this project, I developed several skills, although I had them already, they got better. One of these my ICT skills, as it was an ICT project. Logical thinking was also critical, and most likely one of the most improved ones as I had to think what the code would do in a logical manner. Adaptability was also improved because although I have had previous programming skills with Visual Basic, Scratch and Python, I have not had much experience with JavaScript. Although I may have covered some of it in S4, it was not until the end of S5 that I started to understand it and this project required me to learn more.

In terms of self-motivation, resilience and perseverance, I found this to be quite easy. I became quite enthusiastic with this project and would happily spend time on it, to the point that I was not doing the revision I should when I was expanding this project. This also led to me having to develop my time management skills in order to allow me to do some revision each day.

On Christmas eve I was out walking the dog when I came up with the first solution for the ghost`s code for Dotman, and I spent time that day and the next trying to get it to work. By New Year’s Day, the orange ghost ‘born’, and the others were made in early 2022. Often when there is an error that I cannot seem to fix, I leave it and go do something else for a bit which may be another aspect of the project.

There was not very much team work or communication skills involved, as most of my peers and teachers had very little knowledge of JavaScript. Andrew Twigg has helped me with a bit of code that reduced the amount of code that I have by a very large amount. He also has found a way for the project to be online.

For my problem-solving skills, there were several problems that I have had to find ways round. Sometimes they are big problems, like how to make a major aspect of the game (such as Tetris’s rotating blocks and moving the dots together) or it could be something small, like a syntax error.

One issue that had been persistent was in Dotman, the orange ghost often would go straight into the walls. this would cause the game to crash originally. I could not find the error and put additional code in the ghost, which enabled the game to continue but a hole would appear where the ghost was. Only fairly recently did I find the error. Turns out I was allowing the ghost to move without checking if there was a wall there. This bit was only for a few scenarios, which rarely came up.

Originally, the ‘timers’ or the way to keep things moving was done by using the setInterval() function. This worked, but to stop them I had to refresh the page, which was OK until I tried to put everything in the same document and you would keep going back to the home screen.

For minesweeper, I experimented with setTimeOut(). This would call the function once after a period of time, rather than repeatable forever. Minesweeper made use of this, defining it in a variable and clearing the timer when it is to stop. After this, I went to my other games and found a different way to get around it.

There is a function, that when called, will run each aspect of the games to make them run. Once it has run one ‘tick’ of the game, it will call itself via a setTimeOut() function. This line however, is within an If statement that checks to see if the Boolean value for storing if the game is still playing is true. If true, wait a period then run again. If false, the function will reach its end and nothing else will start after it, ending the timer.

Initiative and organisational skills were also improved. This was a mostly independent project and I had to make all the decisions about it. However, for the design layout, I asked other people what they thought of it. Organisational skills were also improved because this is a very large project with around 23 thousand lines of code, I had to be organised. The project is divided up into the games and the main website each having a folder of their own (however, for convenience, the HTML page for the website is not in the folders). In each folder is a HTML file, a CSS file and a JavaScript file.

The CSS file is for the styling of the HTML page, and I will be honest and say that none of them are particularly organised. But I think I get away with it as they are all relatively small. The JavaScript files are for the code. Often functions that work together, such as ones for the player or the plotting functions are found together, which is me moving them about to organise them. Realistically, I made these function together and they naturally appeared near each other.

For my communication and literacy skills, this is most likely the largest part of the project that Involves them, unless you count what’s in the About Us page and the instructions. Finally, my research skills were also improved as I often ‘researched’ how each of the games were meant to work. (Most of the time I had to play them) Looking through W3 Schools for the pre-defined functions that I needed was also research.

**Dotman creation**

The creation of Dotman is an example of my logical thinking, problem solving and to some extent my numeracy skills.

To start, I Knew that I had to create a HTML table that originally filled the entire page. To do it manually would take too long, so I made another JavaScript program that does it for me. I can input the height and width of the table, along with a class name for each table cell, and a function for what to do on a right or left click of the mouse. The program then runs a For loop in a For loop and outputs the html code into a text box, which can be copied. I also found a way for it to be downloaded.

Once the HTML and CSS was made, I moved on to making the JavaScript. Firstly, It was the plotting function, the functions that receive a coordinate point on the table to change to their colour. As each table cell has a unique ID in the form ‘btnY\_X’, I can use the x and y points given to create the ID of the table cell to change. I then remove every CSS class from the given cell and replace it with the one that I want to use. There is a total of 8 of these plotting functions. The benefits of using the classList.add and .remove is that I can alter more than just its colour the Circular Dotman and ghosts is done by setting the border radius to half of its height/width.

I then also create the function whichButton(). This is activated when a keyboard key is pressed and takes the value of the key pressed through a Large If statement to see what the program should do. This is how the arrow keys / WASD work, as the player's direction is altered. This function is also useful for testing certain aspect of the program.

The next step was to be able to save the entire background that the game has. This is done in a record (array of arrays) and a number representing what is there is stored in the record. 0 is empty, 1 is a wall. The record itself is stored outside the functions so that is is a global record, being accessed by every function. A function that then goes through this large record and calls the relevant plotting function was also created.

After that, I started working on the player, which required the use of a few more global variables, the players X and Y coordinates and the players direction, as pac man would not stop until you change direction. the function that moves the player, movePlayer() starts by unpotting the current player position, using one of the plotting functions mentioned above. Then it goes through a if statement which basically looks to see which direction the player is going and if it would still be within the boundaries of the game. It then also runs a test with the function testPosition() of where the player would be if the statement was to return true. if all the statements are true, then the pliers X and Y values will change.

The function testPosition() is used a lot later, but it is used to determine if the given coordinates are a wall or if it is a passable place. It looks at the containers in the record that stores what the background of the game is to see if it has the value of 1. if it does it returns false, meaning that a character cannot exist there.

The function collision() is also called whilst moving the player and it looks to see if the player is colliding with a ghost, if it is, then the player dies by calling the DIE() function. Unless of course the

Boolean variable that states whether the player has the power up effect on is active, then the player's score increases by 50, and the ghost that got eaten is sent back to the start.

Collision() also detects whether or not the player is touching a coin, if so increasing the score by 1 at altering the record for where the coin is to be 0, empty. If the player is not touching a coin, then it might be touching a power up. If so, then the Boolean variable that states if the power up is not or not is set to true and a timer is called to turn the variable back to false after a certain amount of time. This is the reason that the power ups cannot be stacked. the function that returns it to false cannot be altered once it has started.

After collision(), the movePLayer() function calls tp(), teleport. this is for when the player is at the edge of the screen, it sends it to the other side. It merely changes the players X and or Y coordinates, as the player is still not showing since the unplot() function was called at the beginning. The player comes back on the next and final line, which re-plots the player at the players X and Y coordinates, which may have altered during the course of the function running.

The ghosts’ properties, such as coordinates, direction and colour are saved in a record, with each category of the record is the ghost, and the array within that part is that ghost's properties. Whenever this record is used, its length is never really in the code, meaning that I can easily add more ghosts, or reduce the number of ghosts.

Each of the functions that moves the ghosts are very similar, but each is slightly different. The code for the orange one, the first to move, is mostly random, hence it will not usually follow you. However, if you get within 6 or so table cells and are directly in a straight line towards you, it will follow you. This is where there are some bugs relating to the orange one being capable of going through walls. For the following bit, M = gradient between the ghost and the player. The red one takes the shortest route towards you. if in a straight line, M = 0 or M = undefined, the ghost will move towards you directly, unless it is at a junction where it cannot go in your direction, then it will pick a random direction perpendicular to you. If the gradient is between -1 and 1, I will move towards you along the X axis. If the gradient is less than -1 or greater than 1, then it will move along the Y axis. This is assuming that is is able to do so. If it is at a corner or a straight line, then it will continue ‘forward’. A dead end is treated like a junction, except it can only go in one direction.

The pink ghost is very similar to the red ghost, except at junctions. Where the option is available, the pink ghost favours going along the Y axis rather than the X axis when the gradient is between -1 and 1, and favours the X axis when the red would go along the Y. These two then create an aspect of the ghosts being able to cut you off. The blue ghost however, is a combination of the red and pink ghost. Each time its functions run, it picks (at random) which function to use - the one for red or the one for pink. This is handy as there where / are areas of the game where the red and pink ghosts would end up getting stuck and going in loops should you stay still. The blue ghosts may take a while, but will get to you eventually, ensuring you can't keep going forever.

The ghosts, particularly the orange one, makes use of several functions that support the decision-making process. They mostly just see if the ghosts can exist at a certain position, or return a list of directions that the ghosts are allowed to go. One of them is IsCorner(), and it returns the number of positions that the ghost could go.

If it is 2, then the ghosts will look further to see if it is a corner or straight line. If it is 1, then it will move backwards as it must be dead end. 3 or 4 are junctions and makes the above decisions to decide which way to go.

When you click the play button, it runs a function that is only executed at the beginning of the game, and it ensures that the game is not already running. (There were glitches for some time that if you clicked the play button too many times, the ghosts would end up going faster.) The main purpose of this start function is to set the default values to the global variables and then start two functions which once they are finished, set a timer to recall themselves. these two are the functions that run the player and the ghosts. When, for example, the player function runs, it calls the move player function. (The move player function does not call itself because it's easier to control. It also means that I can call that function more often without sleeping the game up.) Move player also then calls the function that updates the display, which ensures that the score is up to date.

When it calls itself, it can only do so if the Boolean value that shows weather or not the game is still active is true. Once this value is set to false, the game will end as the functions that keep the game going will end and not restart. This means that you do not have to refresh the page to stop the timers, unlike some of my other games

**Outcome**

One of the aspects of my website, which is good in some ways, but negative in others, is the fact that the ‘Website’ part exists all in the same webpage. I tried to do this because the Navigation bar, The footer, The header, and several other lines of code working in the background would be the same, only the content would need to change. So, I made each of these ‘pages’ actually a different <div> tag, and made them show (and hide the rest) when you click to access that ‘page’. Good aspects about it include the fact it saves time, and any edits to the nav bar don`t have to be copied 10 times, which could lead to inconsistencies and it not working on some pages.

The negative is mainly when you refresh the page, you would always come back to the home screen. In the intended use of the site, it does not cause any real issues, but when testing I may make small changes to one of the other pages and I had to keep changing to this page. I would cause an issue when I tried to add all my games to the same document with the site. As there are timers running, all but one of them make use of refreshing the page to stop these timers. If people are playing not-so-flappy dot and die, then they would have to navigate back to the game.

*Thinking about it, I could have a local storage container for the ‘page’, which would storage what page you are on. Then whenever the page loads, which is does after a refresh, it would check to see which page it should be on, according to the local storage…*

Dotman, I think turned out very good. The code is designed to operate on any map, and there are other ones coded into the game, but only the one that is very similar to the real game is active. It’s this map that I was happy with. No other one I made I liked.

The ghosts have turned out to be very good, at catching you and running away. I had to have you start with 10 lives to give anyone a change to get to level two, although I have done a whole level in 1 life before. The ghosts also have a useful ‘glitch’ in them that I don`t bother to fix. As they have no way of knowing if they are going down a dead end, that you are not there for, they will repeatable go up and down at the beginning of the game. This means you can press play, but it is not until you press and arrow key to move away that they start coming for you.

For Tetris, I feel that I cheated. The objects do not rotate, when you rotate, the original shape is lost and a new shape is generated at the same position, assuming that it does not collide with anything else.

One this that I would say to people to encourage them to make their own games like these is that they;

a) don`t require internet if you have the files, and;

b) They require a lot of ‘*testing’* to ensure that it works correctly