Dungeon master

Jordan Campbell 3055 33649

Contents

[Analysis 4](#_Toc511820005)

[Description of problem – make it understandable 4](#_Toc511820006)

[End user – who’s going to help what do they want 5](#_Toc511820007)

[Outline of research – what are you trying to find out? 6](#_Toc511820008)

[Research 6](#_Toc511820009)

[Tile engine 6](#_Toc511820010)

[Random maze generation 7](#_Toc511820011)

[Implementation of rooms 9](#_Toc511820012)

[Pokémon AI 10](#_Toc511820013)

[Results of research 12](#_Toc511820014)

[Tile engine 12](#_Toc511820015)

[Random maze generation 12](#_Toc511820016)

[Implementation of rooms 13](#_Toc511820017)

[Pokémon AI 13](#_Toc511820018)

[End user research 14](#_Toc511820019)

[Informal interview with Sze Sze. 14](#_Toc511820020)

[Informal interview with James. 15](#_Toc511820021)

[Results 15](#_Toc511820022)

[Scope of problem 16](#_Toc511820023)

[Objectives – what should be in it 17](#_Toc511820024)

[Critical path 19](#_Toc511820025)

[Data modelling – UML diagram 20](#_Toc511820026)

[Class Diagram v1 20](#_Toc511820027)

[Class Diagram v2 22](#_Toc511820028)

[Prototyping 23](#_Toc511820029)

[Movement around map and in game action 23](#_Toc511820030)

[Room spawning 25](#_Toc511820031)

[Battle system with AI 27](#_Toc511820032)

[Tile engine 28](#_Toc511820033)

[Documented design 30](#_Toc511820034)

[Frameworks 30](#_Toc511820035)

[Structure design 30](#_Toc511820036)

[Key algorithm definitions 30](#_Toc511820037)

[Algorithms 31](#_Toc511820038)

[Display main menu 31](#_Toc511820039)

[Get menu input 32](#_Toc511820040)

[Load game 33](#_Toc511820041)

[Save game 35](#_Toc511820042)

[Random maze generation 37](#_Toc511820043)

[Update game 40](#_Toc511820044)

[Battle AI 44](#_Toc511820045)

[Player battle moves 48](#_Toc511820046)

[Movement and Collision 51](#_Toc511820047)

[Inventory management 52](#_Toc511820048)

[Data structures 54](#_Toc511820049)

[Game 54](#_Toc511820050)

[Map 57](#_Toc511820051)

[AiSwitchChoice 59](#_Toc511820052)

[Battle 60](#_Toc511820053)

[Cell 62](#_Toc511820054)

[Chest 63](#_Toc511820055)

[Item 64](#_Toc511820056)

[MessageBox 65](#_Toc511820057)

[Move 66](#_Toc511820058)

[NPC 67](#_Toc511820059)

[Player 68](#_Toc511820060)

[Pokémon 69](#_Toc511820061)

[Sign 70](#_Toc511820062)

[File structure and organisation 71](#_Toc511820063)

[Pokemon file 71](#_Toc511820064)

[Move file 72](#_Toc511820065)

[Move effectiveness file 73](#_Toc511820066)

[Item file 74](#_Toc511820067)

[Game save file 75](#_Toc511820068)

[Effects file 78](#_Toc511820069)

[HCI – human computer interface 79](#_Toc511820070)

[In game look 79](#_Toc511820071)

[Main menu 81](#_Toc511820072)

[In battle look 83](#_Toc511820073)

[Pokemon selection menu 85](#_Toc511820074)

[Item selection menu 87](#_Toc511820075)

[Move selection menu 89](#_Toc511820076)

[Chest GUI – final 91](#_Toc511820077)

[Sign GUI – final 92](#_Toc511820078)

[Testing 93](#_Toc511820079)

[Choice of test data 93](#_Toc511820080)

[Testing plan 93](#_Toc511820081)

[Scenario testing. 93](#_Toc511820082)

[Unit testing 95](#_Toc511820083)

[Testing evidence 96](#_Toc511820084)

[Evaluation 98](#_Toc511820085)

[Meeting objectives 98](#_Toc511820086)

[Overview of the solution 104](#_Toc511820087)

[Possible improvements 105](#_Toc511820088)

[End user feedback 106](#_Toc511820089)

# Analysis

## Description of problem – make it understandable

Many people play games for fun and to pass time. Playing a game takes you to another world which may at times be more exciting and interesting than real life, for example Pokémon takes place in a very similar world to our own, with a twist, the existence of Pokémon. I aim to create a game that emulates the positive aspects of Pokémon.

Pokémon is a very open game with very few instructions on how to play, leaving it to the user to explore, I will try to emulate this as my game will have no end goal allowing the user to play as they please as well as giving the user no instructions on how to play, similar to the Pokémon GO game.

I will try to take a lot of inspiration from Pokémon and other dungeon exploring games to emulate most of the positive aspects of Pokémon, however one aspect that I will not try to emulate is Pokémon’s vast in-depth features such as its breeding system as well as its online features as this would be too time consuming. This is interesting because Pokémon appeals to a wide audience range such as casual fans as well as hard-core fans, however my games target audience will be casual players as the game is very basic.

My plan is to create a dungeon exploring game. The game will have an underground theme, where a player will explore a randomly generated maze, with multiple rooms.

The dungeon will be randomly generated meaning that it will be different every time a new dungeon is created, allowing the game to be relayed multiple times. You will be able to customise the dungeon by deciding on the size as well as the amount of rooms.

While the player is exploring the maze, they will be able to capture pokemons which they will be able to use to battle against other wild pokemons or battle against other pokemon capturers and their pokemons.

At the start of the game the player will be given six pokemon that they can use to begin battling other pokemons. These NPCs will be able to present a real challenge to the player as they will use AI to choose strategic moves, depending on the typing of the moves and the typing of the enemy pokemon to best defeat the player as well being able to select when to withdraw the pokemons and being able to apply item affects.



The image on the left is a map I will try to emulate and the image on the right is what my game will look like.

## End user – who’s going to help what do they want

My end user will be Sze Sze whom is in year 13 and is a casual gamer, who likes to play Pokémon on her phone. In her childhood, she has grown up with the Pokémon games and TV series, so is very familiar with the concepts and understands what the best aspects of Pokémon are. Using her will help me to focus my project and allow it to appeal to 2 audiences, Pokémon fans and casual gamers. From an informal interview, she expects this game to be “Fun, easy to pick up and have hours for gameplay.”

My second end user will be James who is also a fellow year 13. James is a more hard-core gamer, playing games like FIFA, call of duty and Pokémon. James is a big Pokémon fan whom, watches the TV series, collects the cards and has played the games. James had previously started to code his own Pokémon game, so will be more of a coders perspective when it comes to end user feedback. From an informal interview, he expects the game to be “better than is version.”

## Outline of research – what are you trying to find out?

If I were to abstract my game into its main components it would be into 2 parts, the first being the random generation of the game map, which again breaks down to the random direction and production of the maze as well as how to implement rooms into the maze, while allowing the maze to function still as a maze. The other main part to my project is the battle AI for the player’s opponents, I will need to find out which decision it needs to make and how it makes them. I will also need to know how to create a way to store and display my map.

From this research I will

* Determine which algorithm I will use to generate the maze.
* How to implement the chosen algorithm.
* I will find out how to put rooms in a randomly generated maze
* I will determine how my AI will make decisions and which decisions.
* I will find a way to store my map
* I will find a way to display my map

## Research

### Tile engine

In the past of gaming systems had very limited memory resources, so they couldn’t store all the pixels on the screen in memory as it would be too much, so a solution for this problem would be to use a tile engine. It would instead store a “tile” - a bitmap image, instead in memory and only draw it when needed. This avoids the memory problem as many tiles are already repeated so there is no need to store each tile.

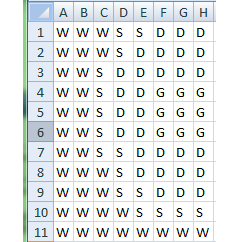
A tile engine works by storing a “cells” –a representation of a tile in a “tile map” –a 2d map of cells. Each cell has a unique identifier to determine the type of tile. So when displaying the map it will go through the array one tile at a time looking at its identifier then displaying the tile at the cell position.

Example

http://www.xnaresources.com/images/tutorialimages/multitile/part1_tileset.png

These would be 4 tiles, water, sand, grass and dirt with a unique identifier of w, s, g and d respectively.

Maps could be stored in a file format such as below, which could be loaded in to the array then used to display the tile map.



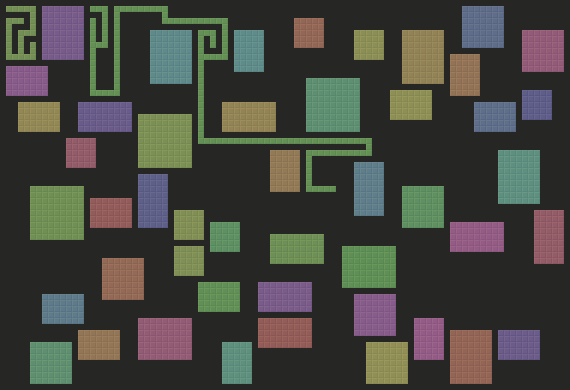
### Random maze generation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Image | Process | Pros | Cons |
| Kruskal's Algorithm |  | Group all edges together  Select a random edge if it connects 2 disjoint trees join them together  Repeat until no edges are left | Based on using a graph already so would be easy to implement path finding within it. | Lots of short dead ends – not very aesthetically pleasing.  Can be inefficient depending on how sets are joined due to iteration. |
| Aldous-Broder algorithm |  | Start at a point, select a random neighbour cell, if it is unvisited then carve a path between them both, else do nothing, repeat until all cells are visited. | The algorithm is very simple so can be very easy to implement. | The algorithm is very inefficient as it can visit cells its already visited due to it being random selection. |
| Recursive Backtracking |  | Select a starting cell, randomly select a neighbouring cell that hasn’t been visited, and remove the wall between the cells. The next cell becomes the current cell above process is repeated, if there are no unvisited neighbours then pop current cell from the stack. The process ends when you backtrack to the starting cell. | It is fast and easy to implement  Quick for small mazes | For large mazes, it is inefficient. |

<https://stackoverflow.com/questions/38502/whats-a-good-algorithm-to-generate-a-maze>

### Implementation of rooms

My initial thoughts for doing this was to create the maze then selecting areas and creating rooms, however problems would arise from this such overlapping as there wouldn’t be a way to differentiate between rooms and corridors, as well as it makes the maze more open which was undesirable as I couldn’t control how many connections there were to a room. After completing research, the solution was clear and very simple to just rearrange the order that I do things. So, I would place a bunch of random non-overlapping rooms making sure they are within the maze bounds. Then fill in the remaining region with maze.



<http://journal.stuffwithstuff.com/2014/12/21/rooms-and-mazes/>

### Pokémon AI

#### Traditional approach

##### General overview

1. First the AI will decide whether to use an item, how it decides is item dependent, e.g. it might use an item to remove a status effect if it has one won’t use it if it doesn’t have a status effect.
2. It will then decide to withdraw the Pokémon or not, depending if it can use any moves if the enemy is more effective against it or it’s about to die of a status effect.
3. If the Pokémon can mega evolve then by default it will.
4. It will then choose a move, which is dependent on effectiveness, current effects. E.g. if a Pokémon has an effect then it won’t use a move to inflict another effect, move effects, weather and held items. Giving each move a score then then selecting the highest scoring move.
5. If there are no good moves to choose from then there is another opportunity to switch out.

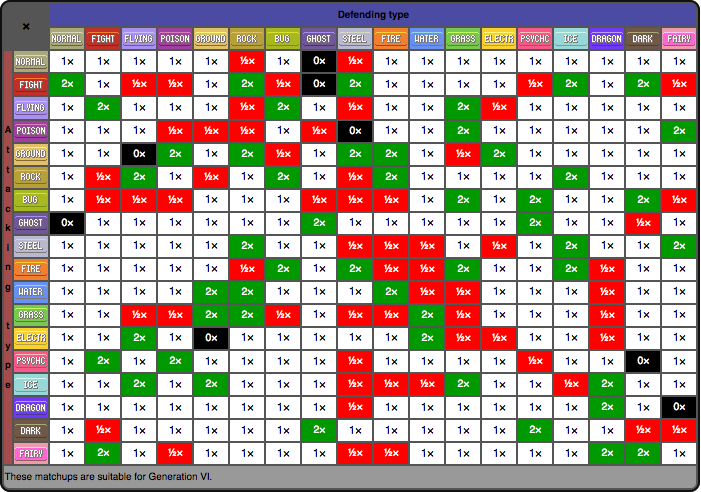
##### Move selection

1. Don’t use a move that only statuses if the Pokémon already has a status.
2. Use a status move.
3. Use the most effective type move with most damage.

##### Damage calculation.

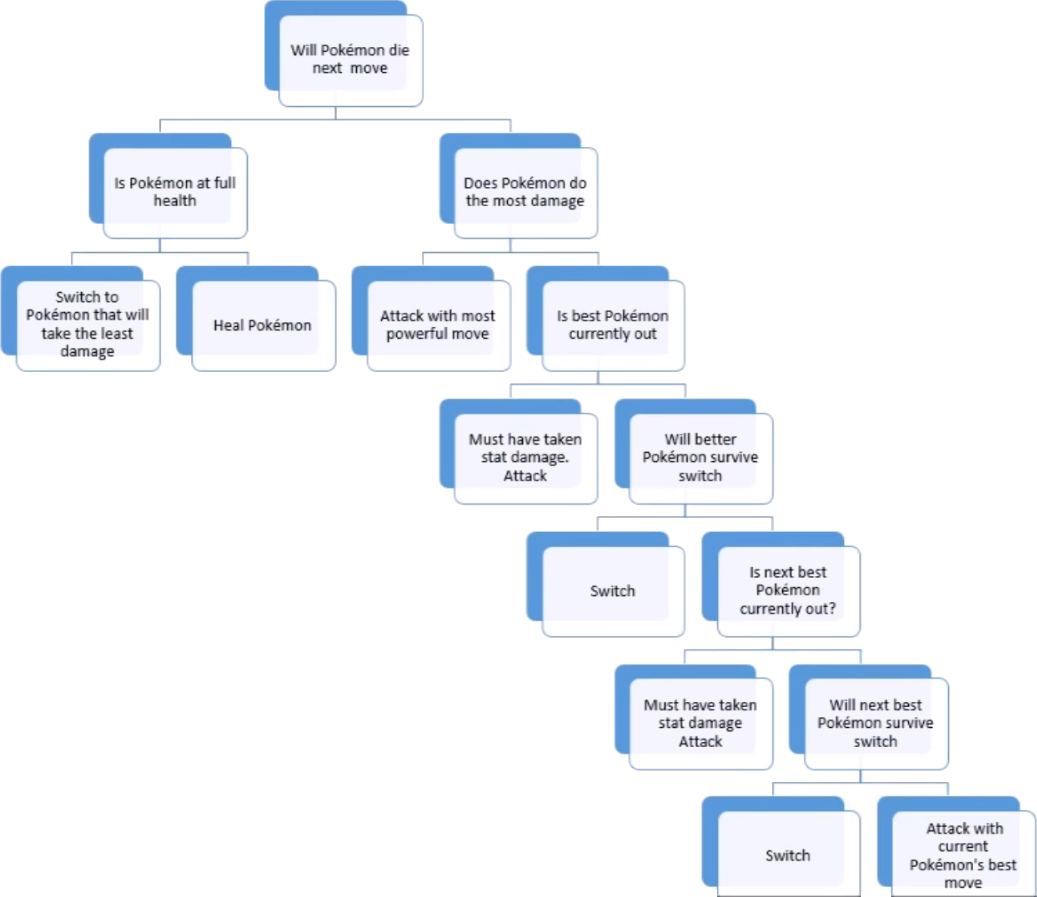
File:DamageCalc.png

##### Damage table



This is a table showing how effective a move type is against a Pokémon type. The Pokémon AI will use this to help it decide what move to use.

#### Drac5290 approach



This is the decision tree he used for his AI with yes to the left and no to the right.

#### willmer’s approach

Each component of the AI will be modular allowing it to be easy to expanded and changed. In the game, there would be different AI characters so each enemy would play different. They would be given personalities such as aggressive or defensive, where the former would be more likely to choose damaging moves and the latter choosing more sustaining moves to keep their Pokémon alive. They could also be given a skill rating where the more skilled they are the more likely they are to choose the best option.

Another point suggested was to assign a value for the effectiveness of each option, add the total and choose a random number between the values and select the option in that range. This allows for the best option to be most likely chosen and sometimes it will choose an unexpected option.

When selecting the moves types of moves can be given preferences, such as status moves when HP is higher or damaging moves when HP is lower.

<http://wiki.pokemonspeedruns.com/index.php/Pok%C3%A9mon_Red/Blue/Yellow_Trainer_AI>

<http://pokemonessentials.wikia.com/wiki/Battle_AI>

https://gamedev.stackexchange.com/questions/28750/designing-ai-in-a-1-on-1-battle-similar-to-pokemon

<https://gamedev.stackexchange.com/questions/137368/pokemon-artificial-intelligence>

## Results of research

### Tile engine

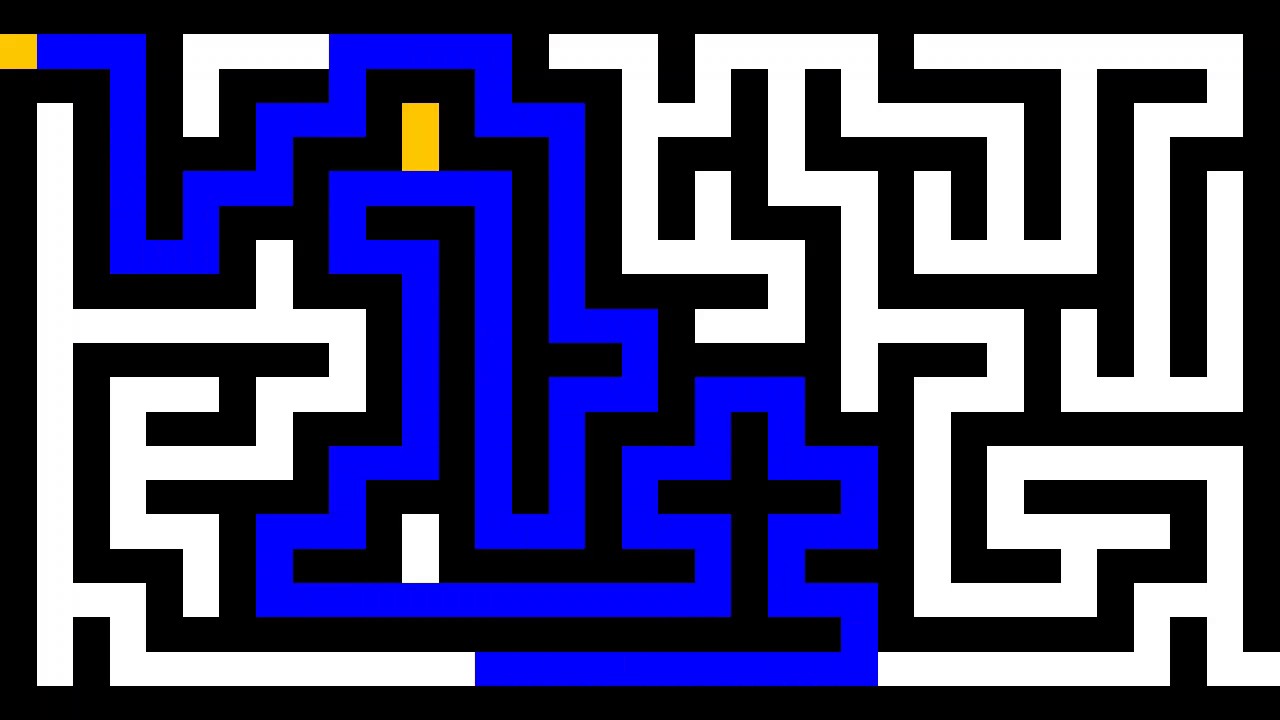
From my research I have decided to store my map in a 2d array which will be suitable as my game is a 2d birds eye view game, I will use a tile engine to store and display my map. The map will be the array and in the array, it will store cells. Cells will have a identifier, coordinates and Booleans to store if it has been visited and if the walls are active.

### Random maze generation

From my research, I have decided to use depth first search recursive back tracking because the mazes it produces are aesthetically pleasing for my underground theme as it has long windy corridors. It also seemed like the easiest to implement. However, a limiting factor of this algorithm is that it is inefficient on larger mazes. From my research I will use this pseudo code

1. Make the initial cell the current cell and mark it as visited
2. While there are unvisited cells
   1. If the current cell has any neighbours which have not been visited
      1. Choose randomly one of the unvisited neighbours
      2. Push the current cell to the stack
      3. Remove the wall between the current cell and the chosen cell
      4. Make the chosen cell the current cell and mark it as visited
   2. Else if stack is not empty
      1. Pop a cell from the stack
      2. Make it the current cell

to make a subroutine to generate a maze. I will create a 2d array to store the maze which will contain the tiles of the maze. I will then select a starting point in the maze then use the algorithm.



### Implementation of rooms

For the implementation of the rooms I will have default amount of room spawning attempts, which will be configurable, the size of the rooms will be random in between a set bound and will be square shaped. In my program before the maze is generated it will first check if the location specified is a valid spawning location, if not it will move onto a new spawning location, else it will remove all walls in the specified location from the tiles and then it will select 2 random tiles around the edges and remove the connecting wall to the room allowing access to the room. In the rooms will be placed tall grass, the maze will then be generated around the rooms.

### Pokémon AI

From my research I have decided I will mostly follow willmer’s approach with the traditional model as base structure for decision making as well as using the effectiveness chart to help aid in the AI decision making for moves. As well as implementing my own methods. I have disregarded Drac5290 approach as his approach seems to be very hard coded as each decision is based upon the previous so it would be difficult to change the AI at a later stage.

From willmer’s approach I will use

* The modular aspect as it allows to make changes a lot easier but also to develop different AI “characters” which could have different play styles.
* For my own approach I will generate possible options e.g. use move, switch and use item. Each option will go through a scoring process with the best option having a higher score so will be more likely to be chosen.
* From willmer’s approach I will also implement the random chance variable, where the AI will choose an option by selecting a random number and the best option have the highest chance but other options not. This will make the AI less predictable and fallible so the player can have a better chance of winning.

## End user research

### Informal interview with Sze Sze.

**What do you know about Pokémon?**

“It is a Japanese franchise. It is a game and anime which everyone loves”

**Specifically, about the Pokémon game, what is your favourite aspect?**

“I like how you are given no instructions and you have to figure things out for yourself. And you must use your common sense to figure out patterns and puzzles. It forces you to talk to other people to collaborate to figure out what to do. It allows you to create a group of friends with similar interests.”

**What are your thoughts and opinions on my concept?**

“I like the randomness of the generation as it means it can be played multiple times, it won’t get boring you could memorise one map, then just make a new one and it will always be different, it makes the game very replay able. The AI is a good idea as it will challenge players. I like how there is random chances for encounters as it is a surprise, however maybe encounters could also occur in not tall grass but have a lower chance.”

**What are your opinions on my AI concept?**

“The idea of chance based selection based on logical decisions makes it good as the AI won’t always be perfect allowing for the player to win, maybe there could be difficulties in the AI based on player level.”

**What do you think of the concept of no end game?**

“I think that this is a bad idea, it may work in games like Tetris but at least they have a payoff, that being the high score, so you can try and beat it. Endless games get boring as the is no satisfaction as there is no progress and this could discourage players from playing”

**What could be an end game scenario?**

“You could have a lose game scenario where it is game over when all your Pokémon have 0HP and you get a score based on how many NPCs you have beaten. A win case scenario would be a boss level where a door could randomly spawn, and you fight really hard Pokémon.”

**What features should be in the game?**

“My favourite Pokémon is Togepi, Togepi should be in the game. A cool idea would be the ability to customise you character sprite to make it more personal to the player. Another feature could be mini-games with in the game to give the player a break from the main game, so it doesn’t get boring.”

### Informal interview with James.

**What are your opinions on my AI concept?**

“The AI structure seems very thought out, I like the modular aspect of the AI because it allows for you to easily change the AI, or for another programmer to do so, and it makes the code more robust.”

**What do you think of the concept of no end game?**

“I do not like this idea, the game has no payoff it will make players leave very quickly after playing, if you were to implement a win case scenario, I would suggest that players have to train their Pokémon to level 100 and then they win.”

**What features should be in the game?**

“I think the game should be very customisable, you should be able to select the different types of caves, like you could have an underwater cave where it’s all blue. And you should be able to change your character, for example their gender, hairstyle and clothes. It would also be cool if you implemented multiplayer as it would present players with new types of challenges. Also, rayquaza and regigigas should be in the game.”

### Results

From my interview with Sze Sze I have learnt that the game shouldn’t become stale, so should be more customisable so the game can be more replay able and different such as changing how the character looks and maze size but also to have mini-games. It also should have an end game scenario to make the game more fulfilling.

From my interview with James I have learnt that the code should be as robust as possible to allow it to be reused to make another game by myself or another programmer. I have also learnt that the game should have a larger pay off to make it worth more playing.

So, I will implement more customisation, the scalability of the AI and an end game scenario. However, I will not implement the mini-games and multiplayer as it will be time consuming as well as hard to code for the multiplayer side.

## Scope of problem

For the AI section of my game I could of use recurrent machine learning where a basic AI would repeat the battle situation in make cam repeatedly until I came up with a strategy to almost always succeed. However, I will not be doing this as it would be too complicated for me to program and would be tie consuming as it would mean a person playing against the AI until it becomes sophisticated.

I will also no be replicating most of the features within Pokémon such as breeding or the vast amount of Pokémon as this will again be too time consuming as there are so many to code. I will limit the number of Pokémon to 10.

In terms of hardware I will be limiting my game to keyboard input only to emulate the traditional input method of buttons and it being playable on windows PCs only. This will reduce the workload, so I can stay within the time limit. I also won’t be doing this due to my lack of knowledge of how to port games.

I will limit my game to single player only to save time as if multiplayer were implemented then I would effectively double the coding time length as there would be a client and server program, also the development and testing would be difficult as I would have to set up my own server.

For textures I will not be making my own textures but use copyright free textures, with slight modifications to make textures more suitable to my solution.

Because not all of my game is turned based a suitable approach will be the XNA framework as it updates the game constantly which allows for live input as well as unprompted updates to the stage.

## Objectives – what should be in it

1. The game will have a start screen, which will display the menu which has 3 options “Start new game”, “Load game” and “Exit”, the start screen will also display the game title “Dungeon Master” and will have a background image of a cave system.
   1. You will be able to use the number keys 1,2 and 3 to select a menu option.
2. If the option “Exit” is chosen, then the program will exit.
3. If “Start new game” is chosen, then
   1. It will then generate the maze in a 2D array of Cells, using a recursive back tracking algorithm.
      1. The room shape will be square with the length and It will fill the 2D array with new Cell.
      2. With the tiles in the 2D array it will apply the recursive back tracking algorithm, starting at Cell (0,0)
      3. It will give a symbol value to each cell depending on which walls are true and false
   2. Then populate the maze with rooms and interactable tiles such as tall grass and chests.
   3. It will the read from a file a create a list of Pokémon
      1. The Pokémon will have specific moves to them
   4. The player will then be given 6 random Pokémon as well as 6 random items
4. If “Load game” is chosen, then
   1. A file explorer will open allowing the user the select the save file in the form of a text file.
   2. The file will be then loaded in to the game and parsed to create a new game.
5. When the game begins.
   1. The player can used the WASD keys to move around the map up,left,down,right
   2. The map will be scrolling screen, with no momentum
   3. They can press E to open their bag
   4. And they can press E to display the menu
6. If the player walks through tall grass, there is a 0.25 chance of a random Pokémon encounter.
   1. In this encounter they can battle the Pokémon, use an item e.g. pokeball switch or run.
7. The player can also press T if standing next to a chest and will see an interactable GUI where they can manage their inventory, chest inventory will be randomly generated.
8. If a player is next to an NPC
   1. They can press T and then the player will be presented with a random phrase then a battle will occur.
   2. The players can’t walk through the NPC, if the player beats the NPC then it will passable allowing the player to progress.
   3. The player has the same option as random encounter but can’t run or catch Pokémon
9. In a battle the player will battle the AI taking it in turns to make a move, this will happen until either all the players or NPCs Pokémon have fainted then it will redisplay the map.
   1. The AI will generate options and score them, higher the score the better the choice, it will then select a random option dependent on the option scores.

Extension objectives.

1. The player will be able to find a key in a random chest
   1. The key will be able to open the door on the map perimeter
   2. The player will be able to open the door by pressing T if they have the key
   3. If the player walks through the door they will be taken to a new map
   4. In the final map the player will be able to walk to the final boss and press T to battle the final boss
2. Exclusive region Pokémon
3. Pokémon can level up and learn new moves.
4. Background music dependent on scenario
5. Customization of character sprite and cave

## Critical path

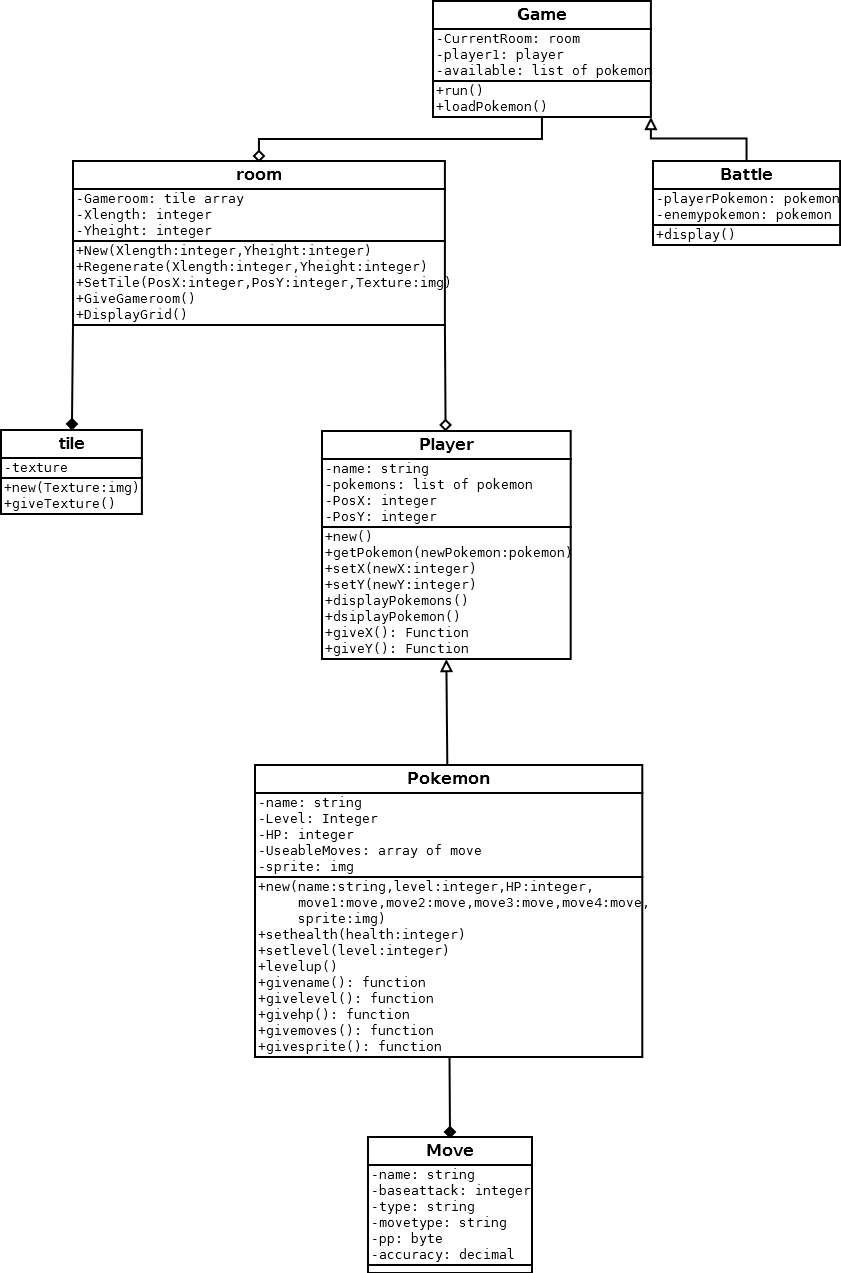
Critical path is the most time efficient way of completing a task leading to the shortest time taken. It is used to decide which way is the best way to flow through a process. Due my game having two separate sections it allows me to have to starting points to choose from, I have decided to start with the map portion as it will be easier and quicker to complete.

1. Define classes
2. Generate maze
3. Load Pokémon
4. Display player
5. Player movement
6. Populate maze with non-player persons
7. Player collision
8. Non-player person interactions
9. Create AI
10. Implement AI into battling
11. Create menus/ GUIs
12. Save level
13. Load level
14. Execute the extension objectives.

## Data modelling – UML diagram

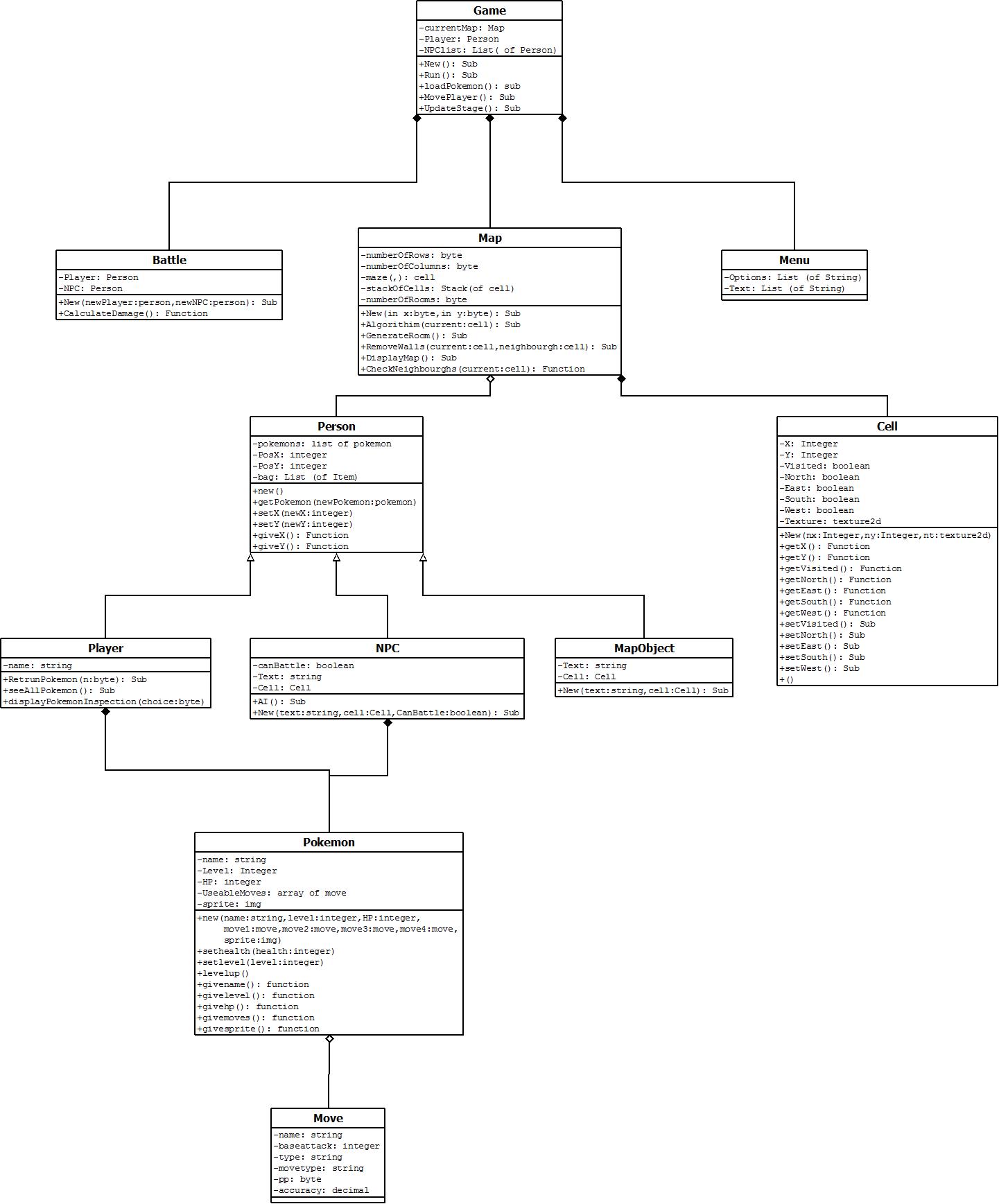
### Class Diagram v1

For my initial idea I decided that I would need 7 classes in my program. That being the Game, room, Battle, tile, player Pokémon and move. Game would be the most important class as it encapsulates all the other classes. With room being a composition of game as the room can be changed as there could be multiple rooms. As well as player being a composition of room as the player could change potential in a multiplayer scenario or a player can select a new map. Pokemon would also be a composition of player as the pokemon can still exist without the player/NPC as they can still appear in random encounters. However, tile is an aggregation of room as the tiles are specific to the map, as well as move to pokemon as moves are specific to the pokemon. Battle has no dependency on Game as they are unique within themselves.



### Class Diagram v2

After a revision I had decided I would need to include 4 new classes as well as adding new variables to existing class. Player has changed to person and has three children class which inherit from it, Player, NPC and MapObject. The final new class is menu, which will be used to create all in game menus. Which will be and aggregation of game.



## Prototyping

### Movement around map and in game action

Sub displaygrid(ByVal x As Integer, ByVal y As Integer) 'displays the array as a room

Console.Clear()

For j = 0 To Sy

For i = 0 To Sx

Console.ForegroundColor = gameroom(i, j).givefront 'displays "tile"

Console.BackgroundColor = gameroom(i, j).giveback

Console.Write(gameroom(i, j).givesymbol)

Next

Console.WriteLine()

Next

Console.BackgroundColor = gameroom(x, y).giveback 'at the player position it will display the character symbol with the same background as the tile which the player is on.

Console.SetCursorPosition(x, y)

Console.Write(Chr(182))

Console.SetCursorPosition(0, 0) 'resets cursor position

End Sub

Sub MovePlayer() 'this sub allows for the user to input one movement action for their character

Dim input As ConsoleKeyInfo 'stores the user key input

input = Console.ReadKey(True) 'takes the users input

Select Case input.Key

Case ConsoleKey.W 'WASD keys used for movement

If player1.y > 0 Then 'checks if the user is at the edge, only allows movement if not at the edge

If currentRoom.GiveGameroom(player1.givex, player1.givey - 1).givesymbol = "^" Or currentRoom.GiveGameroom(player1.givex, player1.givey - 1).givesymbol = "~" Then 'this doesnt allow movement through sea or rocks

player1.sety(player1.givey) 'player stays in the same position

Else

player1.sety(player1.givey - 1) 'otherwise if not moving into rock/sea allows movement

End If

……........

End Select

Select Case player1.givey 'if player is at edge when player makes a move over the edge it deafaults it to the same position of player

Case Is > size - 1

player1.sety(size - 1)

…………..

End Select

End Sub

Sub UpdateStage()

If currentRoom.GiveGameroom(player1.givex, player1.givey).givesymbol = "#" Then 'checks if player is in tall grass

Randomize()

Dim randomNumber As Decimal = Rnd() 'produces random decimal

Select Case randomNumber

Case Is <= 0.25 'if decimal is less than equal to 0.25 a battle will occur, 1/4 chance of battle happening

Console.Clear()

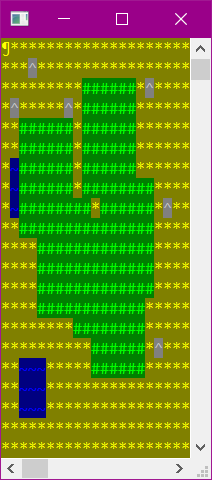
Dim newBattle As New battle(player1.givePokemon(0), availible(Math.Floor(Rnd() \* availible.Count))) 'creates a new battle

End Select

End If

End Sub

#### Result



My prototype is a basic tile engine which users ascii characters instead of actual tiles, the display grid subroutine displays the full map whereas in the final version it will only show 12x12 tiles to reflect the original Pokémon version from a technical standpoint it will be less resource intensive as it renders less tiles allowing for a quicker refresh time. From a user standpoint, it hides more of the map so you feel more immersed as it will make you feel more closed in and will be more likely to get lost in the maze.

In the prototype, it doesn’t allow for the user to move through rocks water or the edges however in the final version the user will only not be able to move through walls, with edges also being replaced with walls. If there are less checks in the final version, then the game should feel more responsive as there will be a quicker update time.

In the update stage, it will check of player is in tall grass and has a random chance of starting a battle. In the final version, it will check proximity to npcs of the player to allow for npc interaction.

### Room spawning

Module Module1

Class point

Public x As Integer

Public y As Integer

Sub New(ByVal i As Integer, ByVal j As Integer)

x = i

y = j

End Sub

End Class

Class tile

Public place As point

Public symbole As Char

Public taken As Boolean

End Class

Sub Main()

Randomize()

Console.Clear()

Dim rooms As Integer = (Rnd() \* 20) + 9 'random number of room spawn attempts

Dim grid(30, 30) As tile 'grid stores tiles of the map

Dim rnd1 As Integer 'random integer for room size

Dim possible As Boolean = True 'variable to store if room spawn is possible

Dim rndpos As New point((Rnd() \* 27) + 1, (Rnd() \* 27) + 1) 'random postion in room where room spawning begins

For y = 0 To 30 'fills grid with tiles

For x = 0 To 30

grid(x, y) = New tile

grid(x, y).place = New point(x, y)

grid(x, y).symbole = "#"

grid(x, y).taken = False

Next

Next

For i = 0 To rooms 'room spawing algorithim

rndpos = New point((Rnd() \* 22) + 1, (Rnd() \* 22) + 1) 'decides spawnpoint

Randomize()

rnd1 = (Rnd() \* 5) + 1 'decides room size

possible = True 'defaults spawing possibilty to true

For y = 1 To rnd1

For x = 1 To rnd1

If grid(x + rndpos.y, y + rndpos.y).taken = True Then 'if part of the room is there then it will exit the loop, to avoid overlap

possible = False

Exit For

End If

grid(x + rndpos.x, y + rndpos.y).taken = True 'else will take current point as taken, replaces symbol and repeats loop until room is completed or until overlap

grid(x + rndpos.x, y + rndpos.y).symbole = " "

Next

If possible = False Then

Exit For

End If

Next

Next

For y = 0 To 30 'displays room

For x = 0 To 30

Console.Write(grid(x, y).symbole)

Next

Console.WriteLine()

Next

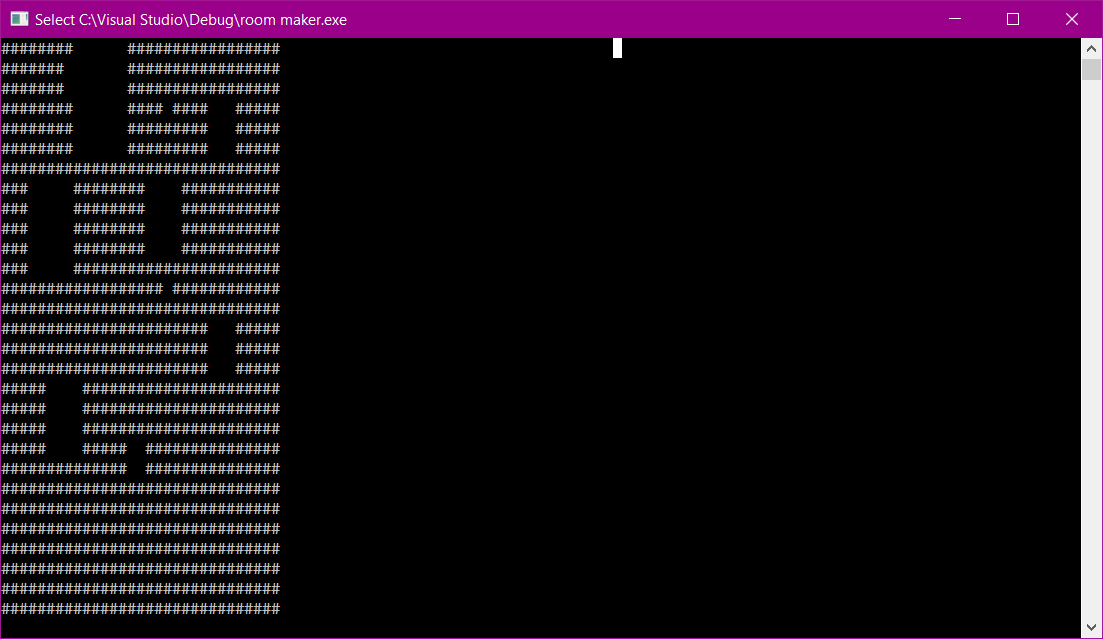
Console.ReadKey()

Main()

End Sub

End Module

#### Result



This method is quite simple so will be very easy to implement into a final version however due to its simplicity it allows some problems to arise such as rooms being smaller than they should be, rooms next to each other so they appear as one room and rooms right on the edge of the map.

### Battle system with AI

Public Class battle

Private playerPokemon As pokemon 'stores the players pokemon

Private enemyPokemon As pokemon 'stores the enemies pokemon

Sub New(ByRef PP As pokemon, ByVal EP As pokemon) 'this sub creates the battle enviroment which is sperate from the game room

playerPokemon = PP 'player pokemon gets stored

enemyPokemon = EP 'enemy pokemon gets stored

Console.WriteLine("BATTLE !!!") 'visual to show battle is occuring - in final build mostly likely will be a grpahical introduction

Do

Console.WriteLine("player hp: {0} enemy hp: {1}", playerPokemon.giveHP, enemyPokemon.giveHP) 'displays both pokemons current health = HP

enemyPokemon.takeDamage((playerPokemon.chooseMove).GiveBaseAttack) 'the player chooses a move which its pokemon does to attack the enmy pokemon

Console.WriteLine("Enemy hp: {0}", enemyPokemon.giveHP) 'shows enemies hp to show damage done

playerPokemon.takeDamage((enemyPokemon.randomMove).GiveBaseAttack) 'the enemy randomly "chooses" a move to use to attack the player - in the final build most likely ai will be built into descion making

Console.WriteLine("Player hp: {0}", playerPokemon.giveHP) 'shows players hp to show damage done

Console.ReadKey()

Loop Until playerPokemon.giveHP <= 0 Or enemyPokemon.giveHP <= 0 'loops until one of the pokemon "faints" = has 0 HP

End Sub

End Class

#### Results

The prototype displays a basic text UI however in the final version it will be graphical. Currently battles only allow for 1 v 1 in the final version it can allow for up to 6 v 6 but will have the same win condition. There is not AI in this prototype which has been substituted for random moves as it was easier to implement in the prototype as it is a very complex feature but will be implemented into the final version.

### Tile engine

Sub Algorithim(ByVal current As Cell)

Dim neighbourghCell As Cell = CheckNeighbourghs(maze(current.getX, current.getY))

If neighbourghCell.getX = 0 And neighbourghCell.getY = 0 Then

If stackOfCells.Count > 0 Then

Algorithim(stackOfCells.Pop())

End If

Else

maze(neighbourghCell.getX, neighbourghCell.getY).setVisited(True)

stackOfCells.Push(current)

RemoveWalls(maze(current.getX, current.getY), maze(neighbourghCell.getX, neighbourghCell.getY))

Algorithim(neighbourghCell)

End If

End Sub

Sub DisplayMaze(ByVal tiles As SpriteBatch)

Dim sourceRectange, destinationRectangle As Rectangle

For y = 0 To numberOfColumns - 1

For x = 0 To numberOfRows - 1

sourceRectange = New Rectangle((maze(x, y).getSymbol Mod \_tileMapTextureColumns) \* \_tileSize, (maze(x, y).getSymbol \ \_tileMapTextureColumns) \* \_tileSize, \_tileSize, \_tileSize)

destinationRectangle = New Rectangle(x \* \_tileSize, y \* \_tileSize, \_tileSize, \_tileSize)

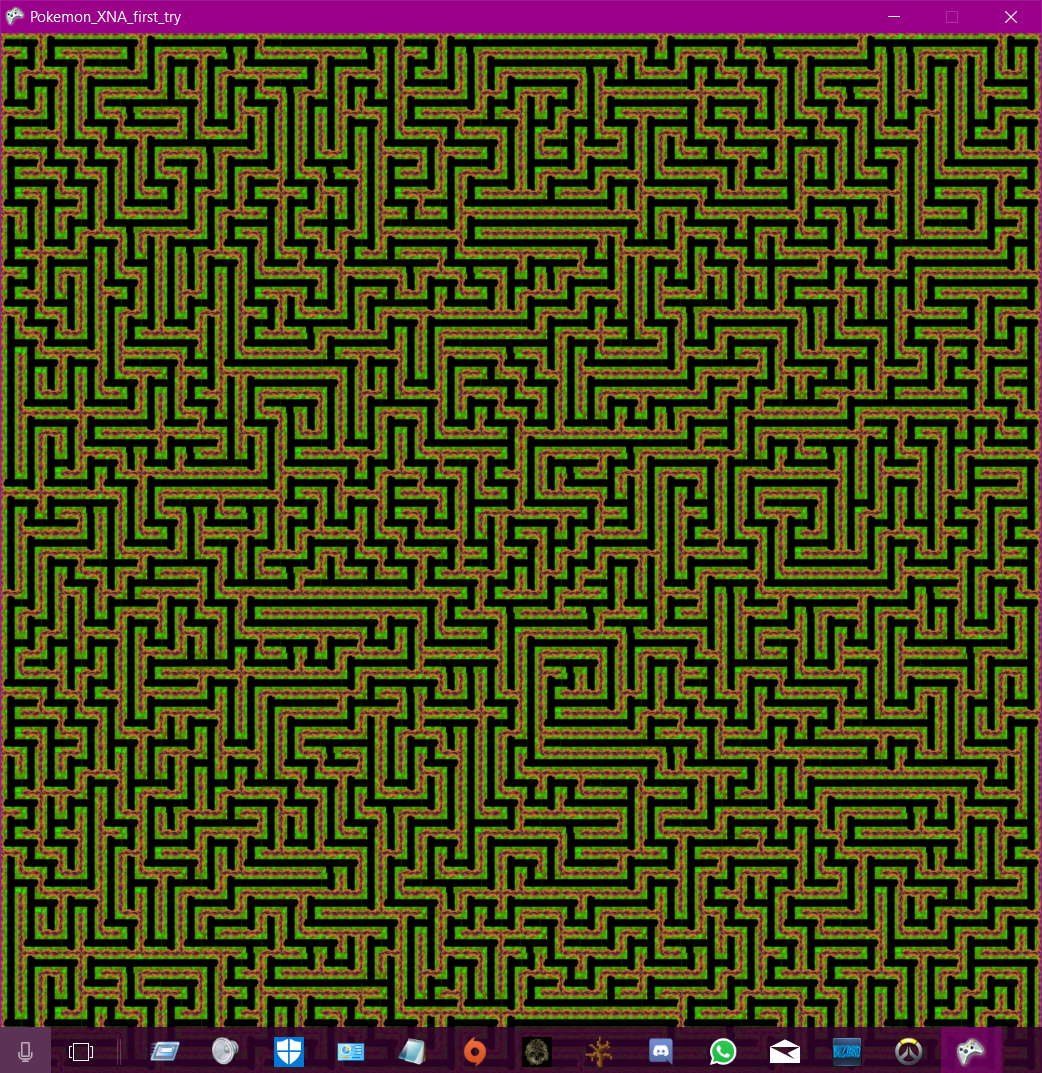
tiles.Draw(\_tileMapTexture, destinationRectangle, sourceRectange, Color.White)

Next

Next

End Sub

End Class



#### Results

From this I have learnt how to implement the recursive back tracking algorithm with a tile engine to create a visual 2d maze using XNA. This prototype will most likely be the final version in my project, with possibly a few slight alterations such as tile size and texture.

# Documented design

## Frameworks

I have used the XNA framework as I want to graphically display my program with textures. And it uses a language I am very familiar with.

## Structure design

I have chosen to use Object orientated programming, as it is suitable in modelling my game as may game many objects. It is also suitable as this is a large project and forces you to plan extensively before coding, this ensures that there will be minimal design flaws in the final project.

## Key algorithm definitions

|  |  |  |
| --- | --- | --- |
| **Algorithm name** | **Parameters** | **description** |
| Display menu | Options as list of string | Displays a menu |
| Get menu input | n/a | Gets menu option selection |
| Load game | n/a | Will load a saved game file |
| Save game | n/a | Will save the current game state |
| Generate new maze | n/a | Will create randomly generated maze and implement rooms, spawn NPCs chests and signs |
| Update game | n/a | Updates game logic |
| Battle AI | n/a | Will select the most viable option for the AI to use |
| Ai move | n/a | Will select which move to use |
| Ai switch | n/a | Will select which Pokémon to switch to |
| Ai item | n/a | Will select which item to use |
| Player move | n/a | Will allow user to select a battle move |
| Player switch | n/a | Allows user to select a Pokémon to switch to |
| Player item | n/a | Allow user to select which item to use |

## Algorithms

### Display main menu

Sub MainMenuDraw()

Dim \_mainMenuTexture As Texture2D

Dim destinationRectangle, sourceRectangle As Rectangle

'The main munu size

Const sizeInPixels As Integer = 512

Try

'Loads the main menu texture

\_mainMenuTexture = Game1.ContentLoader.Load(Of Texture2D)("start menu")

destinationRectangle = New Rectangle(0, 0, sizeInPixels, sizeInPixels)

sourceRectangle = New Rectangle(0, 0, sizeInPixels, sizeInPixels)

TileTextures.Begin()

'Draws main menu texture

TileTextures.Draw(\_mainMenuTexture, sourceRectangle, destinationRectangle, Color.White)

TileTextures.End()

Catch ex As Exception

'Displays the menu options if an error occurs

MsgBox("Press 1 for Exit, Press 2 for Load game, Press 3 to start a new game")

End Try

End Sub

This sub tries to load the menu texture and draw it. destinationRectangle stores where the main menu texture will be stored, sourceRectangle stores which part of the texture will be loaded. It will then catch and display an appropriate error message if there are any crashes.

### Get menu input

Sub MainMenuUpdate()

'Gets keyboard state

kbState = Keyboard.GetState

'If number 1 is pressed the exit game

If kbState.IsKeyDown(Keys.D1) Then

'Denotes a song isn't playing

playingSong = False

'Stops playing the main menu song

MediaPlayer.Stop()

'Sets gameState to exit

gameState = "Exit"

'If number 2 is pressed a game will be loaded

ElseIf kbState.IsKeyDown(Keys.D2) Then

'Denotes a song isn't playing

playingSong = False

'Stops playing the main menu song

MediaPlayer.Stop()

'Sets gameState to Load game

gameState = "LoadGame"

'If number 2 is pressed a game will be loaded

ElseIf kbState.IsKeyDown(Keys.D3) Then

'Denotes a song isn't playing

playingSong = False

'Stops playing the main menu song

MediaPlayer.Stop()

'Plays selection sound effect

gameState = "NewGame"

End If

End Sub

* If 1 Is pressed the current song will stop playing and the game will exit
* If 2 is pressed the current song will stop and a game will be loaded
* If 3 is pressed the current song will be stopped and a new game will be created

### Load game

Sub loadGame()

'Creates a new file dialog

Dim openFileDialog1 As OpenFileDialog = New OpenFileDialog()

'Stores the file dialog result

Dim result As DialogResult

'Instructions for user as file dialog title

openFileDialog1.Title = "Please select a .txt load file location"

'Default file extension

openFileDialog1.DefaultExt = "txt"

'Allows user to select only one file

openFileDialog1.Multiselect = False

'Only displays .txt files

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt"

'Loops until the file is a .txt file

Do

'Opens the file dialog and gets the file dialog result

result = openFileDialog1.ShowDialog()

Loop Until System.IO.Path.GetExtension(openFileDialog1.FileName.ToString) = ".txt"

'If the file dialog has been closed properly then the next code will execute

If result = DialogResult.OK Then

Try

'Loads the save file

Dim saveFile As New System.IO.StreamReader(openFileDialog1.FileName.ToString)

'Gets the map settings, width height and tile texture file location

Dim mapSettings() As String = (Convert.ToString(saveFile.ReadLine)).Split(CChar(";"))

'closes save file as it is no longer required

saveFile.Close()

'Creates a new map

currentMap = New Map(CByte(mapSettings(0)), CByte(mapSettings(1)), True, openFileDialog1.FileName.ToString)

'Sets the game state to "Playing" so the user can play the game

gameState = "Playing"

Catch ex As Exception

'Displays error message

MsgBox("Error loading file, new game started instead " + ex.Message)

'Takes user back to main menu if there was an error loading the save file

gameState = "MainMenu"

End Try

End If

End Sub

* It first creates a new file dialog.
* And a result for the file dialog
* It then sets rules for the file dialog
* It then opens the file dialog and loops until it gets a file with .txt extension
* If the result comes back with no errors then it will read the file, it then gets the first line of the file where it gets the size of the maze
* It then closes the file
* Then creates a new map
* If there are any crashes it will catch them and display an error message

### Save game

Sub saveGame()

'Creates a new file dialog

Dim openFileDialog1 As OpenFileDialog = New OpenFileDialog()

'Stores the file dialog result

Dim result As DialogResult

'Instructions for user as file dialog title

openFileDialog1.Title = "Please select a .txt save file location"

'Default file extension

openFileDialog1.DefaultExt = "txt"

'Allows user to select only one file

openFileDialog1.Multiselect = False

'Only displays .txt files

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt"

'Loops until the file is a .txt file

Do

'Opens the file dialog and gets the file dialog result

result = openFileDialog1.ShowDialog

Loop Until System.IO.Path.GetExtension(openFileDialog1.FileName.ToString) = ".txt"

'If the file dialog has been closed properly then the next code will execute

If result = DialogResult.OK Then

Try

'Opens save file location

Dim saveFile As New System.IO.StreamWriter(openFileDialog1.FileName.ToString, False)

'Writes the map width height and texture location for the tiles

saveFile.WriteLine(currentMap.getSaveConstAsString)

'Double nested for loop, loops through the maze writing the current cells symbol, semi colon deliminated

For c = 1 To currentMap.getSaveMazeList.Count

If c Mod currentMap.getWidth = 0 And c > 0 Then

'When it gets to a new row in the maze it starts a new line in the file

saveFile.WriteLine(currentMap.getSaveMazeList(c - 1))

Else

saveFile.Write(currentMap.getSaveMazeList(c - 1))

End If

Next

'Writes a blank line to the file to seperate blocks

saveFile.WriteLine()

'Double nested for loop, loops through the entitmap writing to file Entity (NPC/CHEST/SIGN) save date, semi colon deliminated

For e = 1 To currentMap.getEntityMapList.Count

If e Mod currentMap.getWidth = 0 And e > 0 Then

'When it gets to a new row in the entitymap it starts a new line in the file

saveFile.WriteLine(currentMap.getEntityMapList(e - 1))

Else

saveFile.Write(currentMap.getEntityMapList(e - 1))

End If

Next

'Writes a blank line to the file to seperate blocks

saveFile.WriteLine()

'Writes the player's save data

saveFile.WriteLine(player.getSaveString)

'Closes the file as it is no longer required

saveFile.Close()

Catch ex As Exception

MsgBox("Error saving file. " + ex.Message)

End Try

'Sets the gamestate back to playing so the user can continue to play

gameState = "Playing"

End If

End Sub

* It first creates a new file dialog.
* And a result for the file dialog
* It then sets rules for the file dialog
* It then opens the file dialog and loops until it gets a file with .txt extension
* It then writes the map width and height and tile texture location on the first line
* In a grid it writes the tile symbols
* In a grid it writes the entity save data
* Then it writes the players save data
* It then catches any crashes and displays an error message

### Random maze generation

For my randomly generated maze I will be using a depth first search recursive back tracking algorithm. This generates a maze in array of cells. This is broken down into 3 main components; the back-tracking part, gathering neighbour cells and removing walls between the neighbours. For this algorithm I will be using a user-defined data structure called cell. Before the algorithm the maze will need to be populated.

#### Maze setup

Const numberOfRows As Integer = 56 'amount of tiles across

Const numberOfcolumns As Integer = 56 'amount of tiles down

Dim maze(numberOfcolumns, numberOfRows) As Cell 'an array to store all the tiles

Sub SetUpMaze()

For y = 0 To numberOfRows - 1 'these for loops loop through the maze array filling it with default cells

For x = 0 To numberOfcolumns - 1

maze(x, y) = New Cell(x, y)

Next

Next

maze(0, 0).visited = True 'sets starting cell to visited

Algorithim(maze(0, 0)) 'sends the algortihim the starting cell

End Sub

In the maze setup there will be two constants, numberOfRows and numberOfcolumns, that will determine the width and height of the maze. The next variable maze will store a 2d array of tiles. In the SetUpMaze algorithm will loop through the 2d array, going left to right then down, filling the array with default tiles.

#### Back tracking part

Dim stackOfCells As New Stack(Of Cell) 'a stack to store the path of the tile progression within the algorithim

Sub Algorithim(ByVal current As Cell) 'the algorithim will end when there are no more cells on the stack and the neighbourgh cell is 0,0

Dim neighbourghCell As Cell = CheckNeighbourghs(maze(current.x, current.y)) ' stores a cell which is a neighbourgh to the current cell

If neighbourghCell.x = 0 And neighbourghCell.y = 0 Then

If stackOfCells.Count > 0 Then

Algorithim(stackOfCells.Pop()) 'if the alogrithim reaches a dea end then it will pop back in the cell till it finds a cell with a neighbourogh

End If

Else

maze(neighbourghCell.x, neighbourghCell.y).visited = True ' this will set the neighbourgh to be true

stackOfCells.Push(current) 'pushes the current cell to the stack to store the travel path

RemoveWalls(maze(current.x, current.y), maze(neighbourghCell.x, neighbourghCell.y)) ' removes the walls between the current and neighbourogh cell

Algorithim(neighbourghCell) ' calls the alogrithim again but with the neighbourgh cell

End If

End Sub

The variable StackOfCells is stack which stores cells, its purpose is to store the progression of the path of tiles with in the algorithm. Due to the design of the stack, first in first out, it allows the algorithm to back track to the previous cells.

The Algorithm subroutine works by being sent a cell, it will then select a random cell next to the current cell, the neighbour cell and if the neighbour cell isn’t the original starting cell, which I hardcoded to be (0,0), it will mark the neighbour cell as visited, push the cell to the StackOfCells, then calls the subroutine Algorithm with the parameters neighbourCell. Heading back to the first if statement if the neighbour cell coordinates are (0,0), therefore signifying no more neighbour cells, and there are still cells on the stack then it will pop a cell of the StackOfcells, i.e. the previous cell and call the subroutine Algorithm again and will search for more neighbours of the previous cell which isn’t the current cell. If there are no more cells on the stack then the subroutine Algorithm will end, ending the recursion.

#### Check neighbour cells

Function CheckNeighbourghs(ByVal current As Cell)

Dim neighbours As New List(Of Cell)

If current.x + 1 < numberOfcolumns Then 'checks if the neighbourgh cell is in the array bounds

If maze(current.x + 1, current.y).visited = False Then ' checks if the neighbourgh cell is visited or not

neighbours.Add(maze(current.x + 1, current.y)) 'if not adds it to a list

End If

End If

Randomize()

If neighbours.Count > 0 Then

Return neighbours(Math.Floor(Rnd() \* neighbours.Count)) ' it will the return a random cell from the list

Else

Return New Cell(0, 0) 'if there are no neighbours ie at a dead end then it will return the srtaing cell to indicate no neighbours

End If

End Function

This is the algorithm that gives a randomly selected neighbour cell to the subroutine Algorithm. It uses a list called neighbours to store all the valid neighbour cells. A valid cell is if it is not visited. A list was used as it can be easily randomized. The algorithm has 4 blocks of if statements for each possible neighbour cell, up down left and right. First, it will check if the cell is within array boundaries to prevent crashes then it will check if the cell is visited if not it will then add it to the list. After it has done that for each possible neighbour cell it will return a random neighbour cell if there are any in the list otherwise it will return the start cell to indicate there are no neighbour cells.

#### Remove walls

Sub RemoveWalls(ByRef current As Cell, ByRef neighbour As Cell)

Dim xResult As Integer = current.x - neighbour.x 'calculates the relative position of the current and neighbourgh cell

Dim yResult As Integer = current.y - neighbour.y

If xResult = 0 Then

If yResult = -1 Then 'dependent on relative position removes walls between them.

current.South = False

neighbour.North = False

Else

current.North = False

neighbour.South = False

End If

ElseIf xResult = -1 Then

current.East = False

neighbour.West = False

Else

current.West = False

neighbour.East = False

End If

End Sub

This is the sub that will remove the walls between the current and neighbour cell, firstly it calculates the differences between the x coordinates and the y coordinates. It will then check the x difference, if it is 0 then it will check the y values depending if its -1 or 1 it will remove the appropriate walls, the same with the x values.

### Update game

'Sets game to refresh at 60Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 60.0F)

'If a battle or encounter has started play battle music

If (gameState = "Battle" Or gameState = "Encounter") And playingSong = False Then

'Denotes a song is playing

playingSong = True

'Stop current song

MediaPlayer.Stop()

Try

'Loads battle song

currentSong = Content.Load(Of Song)(battleSongLocation)

'Plays battle song

MediaPlayer.Play(currentSong)

'Sets battle song to loop

MediaPlayer.IsRepeating = True

Catch ex As Exception

'Displays error message

MsgBox("Error loading battleSong. " + ex.Message)

End Try

End If

'If the player is in the map then play the idle song

If Not (gameState = "Battle" Or gameState = "Encounter" Or gameState = "MainMenu" Or gameState = "LoadGame" Or gameState = "NewGame") And playingSong = False Then

'Denotes a song is playing

playingSong = True

'Stops current song

MediaPlayer.Stop()

Try

'Loads idle song

currentSong = Content.Load(Of Song)(idleSongLocation)

'Plays idle song

MediaPlayer.Play(currentSong)

'Sets idle song to loop

MediaPlayer.IsRepeating = True

Catch ex As Exception

'Displays error message

MsgBox("Error loading idleSong. " + ex.Message)

End Try

End If

'If in main menu then play main menu song

If gameState = "MainMenu" And playingSong = False Then

'Denotes a song is playing

playingSong = True

'Stops current song

MediaPlayer.Stop()

Try

'Loads main menu song

currentSong = Content.Load(Of Song)(mainMenuSongLocation)

'Plays main menu song

MediaPlayer.Play(currentSong)

'Sets main menu song to loop

MediaPlayer.IsRepeating = True

Catch ex As Exception

'Displays error message

MsgBox("Error loading Main menu song. " + ex.Message)

End Try

End If

Select Case gameState

Case "MainMenu"

'Updates main menu logic

MainMenuUpdate()

Case "Playing"

'Updates player logic

player.update()

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

Case "Sign"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "Chest"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "Menu"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "DisplayPokemon"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "PokemonInfo"

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "DisplayBag"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates map logic

currentMap.Update(player.givex, player.givey, gameTime)

'Updates messageBox logic

messageBox.update()

Case "Battle"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates battle logic

gameBattle.Update(player, currentMap.getEntity(player.givex, player.givey), TileTextures)

Case "Encounter"

'Sets game to refresh at 10Hz

Me.TargetElapsedTime = TimeSpan.FromSeconds(1.0F / 10.0F)

'Updates battle logic

gameBattle.Update(player, encounterPokemon, TileTextures)

Case "Save"

'Saves game

saveGame()

Case "Exit"

'Exits game

Me.Exit()

Case "NewGame"

'Creates new game

newGame()

Case "LoadGame"

'Loads a game

loadGame()

End Select

MyBase.Update(gameTime)

End Sub 'Updates game logic

* Sets the games update refresh rate to 60Hz
* If a battle has started, then it will play the battle music
* If the user has gone back to the maze, then it will play the idle song
  + For both if there is a crash it will catch it and display an error message
* It then checks the current game state and runs the appropriate update subroutines

### Battle AI

The battle AI algorithms will decide what option the AI will do in battle, either switch to a different pokemon and which one, or select a move for the current pokemon. The AI is split into 3 algorithms, battleAI, AIswitch and AImove.

#### battleAI

Sub battleAI()

Randomize()

Dim moveChance, switchChance As Integer

If (battleenemy.giveAllPokemon(CurrentEnemyPokemonInt).giveHP / battleenemy.giveAllPokemon(CurrentEnemyPokemonInt).giveMaxHP) <= 0.25 Then

switchChance = 0.75

moveChance = 0.25

Else

switchChance = 0.2

moveChance = 0.8

End If

Dim randomChance As Decimal = Rnd()

Select Case randomChance

Case 0 To moveChance

AImove()

Case Else

If Not battleenemy.giveAllPokemon.count - enemyFainted = 1 Then

AIswitch()

Else

AImove()

End If

End Select

End Sub

In this algorithm it will randomize the current randomize seed. I have created 2 variables called moveChance and switchChance will store the respective decimal chance for selecting a move or switching pokemon. I have hardcoded the if statement to determine these values as I believe it will lead to an AI play style that will emulate my source. So, if the current AI pokemon is at 25% hp or less it will be more likely to switch else it will be more likely to choose a move. It will then generate a random number between 0 and 0.99… so, if the random move is between 0 and the move chance then a move will be selected else it will switch if there is a pokemon to switch to otherwise it will select a move with the current pokemon.

#### AImove

Sub AImove()

Randomize()

Dim moveChance(3) As Decimal

Dim totalPoints As Decimal = 0

Dim selectedmove As move

Dim availableMoves() As move = battleenemy.givePokemon(CurrentEnemyPokemonInt).giveMoves

For i = 0 To 3

moveChance(i) = damageCalculation(battleenemy.givePokemon(CurrentEnemyPokemonInt).giveLevel, availableMoves(i).GiveBaseAttack, Battleplayer.givePokemon(CurrentPlayerPokemonInt).giveType, availableMoves(i).Givetype)

totalPoints = totalPoints + moveChance(i)

Next

For i = 0 To 3

moveChance(i) = moveChance(i) / totalPoints

Next

Dim randomChance As Decimal = Rnd()

If randomChance >= 0 And randomChance <= moveChance(0) Then

selectedmove = availableMoves(0)

ElseIf randomChance > moveChance(0) And randomChance <= (moveChance(0) + moveChance(1)) Then

selectedmove = availableMoves(1)

ElseIf randomChance >= (moveChance(0) + moveChance(1)) And randomChance <= (moveChance(0) + moveChance(1) + moveChance(2)) Then

selectedmove = availableMoves(2)

Else : selectedmove = availableMoves(3)

End If

Console.WriteLine("Enemy used {0}", selectedmove.GiveName)

Battleplayer.givePokemon(CurrentPlayerPokemonInt).TakeDamage(damageCalculation(battleenemy.givePokemon(CurrentEnemyPokemonInt).giveLevel, selectedmove.GiveBaseAttack, Battleplayer.givePokemon(CurrentPlayerPokemonInt).giveType, selectedmove.Givetype))

Console.WriteLine("it did {0} points of damage, player has {1} HP", damageCalculation(battleenemy.givePokemon(CurrentEnemyPokemonInt).giveLevel, selectedmove.GiveBaseAttack, Battleplayer.givePokemon(CurrentPlayerPokemonInt).giveType, selectedmove.Givetype), Battleplayer.givePokemon(CurrentPlayerPokemonInt).giveHP)

End Sub

This is the algorithm that enemy trainer/pokemon will use to decide which move to choose. The algorithm has 4 variables, movechance(3) is an array of decimal that stores the decimal percentage chance for each of the available moves. Totalpoints is a decimal that stores the total of all the points generated from the total damage of all the moves. Selected move is a user defined data structure, move, it stores the move that the pokemon will use. Availablemoves is an array of moves which stores all possible moves from the selected pokemon. A for loop loops 4 times, once for each move and assigns move chance a value depending on how much damage it does. Another for loop converts these values to relativistic percentages, so the higher the damage the more likely it is to be chosen. it then produces a random value, and searches for the value in-between the percentage chances, selecting the corresponding move. It then displays which move is used and calculates damage and outputs the damage done.

#### AIswitch

Sub AIswitch()

Randomize()

Dim pokemonChance(battleenemy.giveAllPokemon.count - 1) As Decimal

Dim totalPoints As Decimal = 0

Dim possiblePokemon As New List(Of AI\_Switch\_Choice)

For i = 0 To battleenemy.giveAllPokemon.count - 1

pokemonChance(i) = effectiveness(Battleplayer.givePokemon(CurrentPlayerPokemonInt).givetype + battleenemy.givePokemon(i).givetype) \* battleenemy.giveAllPokemon(i).giveHP

totalPoints = totalPoints + pokemonChance(i)

Next

For i = 0 To battleenemy.giveAllPokemon.count - 1

If totalPoints <> 0 Then

pokemonChance(i) = pokemonChance(i) / totalPoints

End If

Next

pokemonChance(CurrentEnemyPokemonInt) = 0

For i = 0 To battleenemy.giveAllPokemon.count - 1

If not pokemonChance(i) = 0Then

possiblePokemon.Add(New AI\_Switch\_Choice(pokemonChance(i), i))

End If

Next

Dim randomChance As Decimal = Rnd()

totalPoints = 0

For i = 0 To possiblePokemon.Count - 1

If randomChance > totalPoints And randomChance <= totalPoints + possiblePokemon(i).value Then

CurrentEnemyPokemonInt = possiblePokemon(i).position

End If

totalPoints = totalPoints + possiblePokemon(i).value

Next

If randomChance = 0 Then

CurrentEnemyPokemonInt = Math.Floor(randomChance \* possiblePokemon.Count)

End If

Console.WriteLine("Enemy sent out {0}", battleenemy.givePokemon(CurrentEnemyPokemonInt).givename)

End Sub

This is the algorithm that decides the pokemon the enemy will switch to. It has 3 variables pokemonchance which is an array of decimals that’s stores the decimal chance that a pokemon will be selected. Totalpoints a decimal, which stores the total points generated and possiblePokemon which is a list of a user defined data structure ai\_swicth\_choice which stores the pokemon position and its decimal chance. The algorithm has a for loop which loops through the pokemon and gives points by multiplying effectiveness by HP, as well as totalling total. The next for loop coverts the points to decimal percenatges. It will then add the chance and the position to possiblePokemon if the chance isn’t 0, it will then generate a random number and search for this within the pokemon chances, however if the random chance is 0 then a random pokemon is chosen. it then displays which pokmemon is sent out.

#### AIitem

Sub itemAI(ByRef player1 As Player, ByRef enemy As Player)

'Generates new random seed

Randomize()

'Stores the item chances as decimals

Dim ItemChances As New List(Of Decimal)

'Will store all points allocated to items

Dim totalItemPoints As Decimal = 0

'Stores the random chance for selecting an item

Dim randomItemChance As Decimal = CDec(Rnd())

'Stores the selected items position

Dim selectedItemPosition As Integer = 0

'Stores the selected item

Dim selectedItem As Item

'Loops through all items in the enemy's bag allocating points dependent on their stat multplier

For i = 0 To enemy.getBag.Count - 1

'Allocates itme points

ItemChances.Add(enemy.getBag(i).getValue)

'Adds current item points to total items points

totalItemPoints += ItemChances(i)

Next

'If there is a viable item ie. not all items are pokeballs as enemy can't use pokeballs, then it willexecute the next code

If totalItemPoints > 0 Then

'Loops through all items in emeny bag converting points to percentage chance of total points

For j = 0 To enemy.getBag.Count - 1

'converts points to percentage chance

ItemChances(j) = ItemChances(j) / totalItemPoints

Next

'Sets total item points to 0 so it can be used for storing decimal item chances

totalItemPoints = 0

For k = 0 To enemy.getBag.Count - 1

'Checks if the random chance is between the lower bound and current item decimal chance range

If randomItemChance > totalItemPoints And randomItemChance <= totalItemPoints + ItemChances(k) Then

'If it is then it will set selected itmes position to that items position

selectedItemPosition = k

End If

'Adds the current items decimal chance to total points

totalItemPoints += ItemChances(k)

Next

End If

'Selected item is set to the item in the enemy's bag that is the selected item position

selectedItem = enemy.getBag(selectedItemPosition)

'Removes that item so that it can't be used again

enemy.removeItem(selectedItemPosition)

End Sub

There is a list which will store the decimal chances of each item. It will then loop through all the items in the Enemy’s bag adding the value of the item to the list as well as adding the value to the total value variable, totalItemPoints. It will then loop through the list storing the items values it will then convert the decimal value to a decimal percentage of the total, by diving the decimal value by the total number of points. However, it will only do this if the totalPoints is greater than 0 to avoid diving by zero. Then using the randomItemChance it will loop through all of the items in the bag checking if the randomItemChance is within the items percentage decimal range. If so it will return the corresponding item. if it doesn’t select an item e.g. If the totalPoints is zero, the it will just use the first item. any item used is removed from the bag, so it can’t be used again.

### Player battle moves

#### Player move

If player1.getPokemonList(CurrentPlayerPokemonInt).giveMoves(CInt(pointerPosition.Y)).givePP > 0 Then

‘Decrease the players current pokemon selected moves PP so it can use the move one less time

player1.getPokemonList(CurrentPlayerPokemonInt).giveMoves(Cint(pointerPosition.Y)).decreasePP()

Dim accuracyRandom As Integer

‘Generates a random number

accuracyRandom = Cint(Rnd())

‘If the random number is less than the move accuracy then it will execute the next code, effectively giving the move the chance to succeed as the same as its accuracy

If accuracyRandom <= selectedMove.GiveAccuracy Then

‘String to store the effect text

Dim effectText As String = “”

‘If the effect isn’t “No effect” then it will execute the next code

If Not selectedMove.giveEffectID = 0 Then

‘Stores the effect information from the ID given by the move

Dim moveEffect() As String = effects(selectedMove.giveEffectID).Split(Cchar(“,”))

‘Effect text is set to the last part of the effect information

effectText = moveEffect(5)

‘The effect will then get processed

processEffect(player1, enemy, “P” + moveEffect(2), moveEffect(3), Cdec(moveEffect(4)))

‘Else the effectText will be nothing

Else

effectText = “”

End If

‘The damage of the move will be calculated

Dim damageTaken As Integer = Cint(damageCalculation(player1.getPokemonList(CurrentPlayerPokemonInt).giveLevel, player1.getPokemonList(CurrentPlayerPokemonInt).giveAttack, enemy.getPokemonList(CurrentEnemyPokemonInt).giveDefence, selectedMove.Givetype, player1.getPokemonList(CurrentPlayerPokemonInt).giveType, enemy.getPokemonList(CurrentEnemyPokemonInt).giveType, selectedMove.GiveBaseAttack))

‘The enemy pokemon will take the damage, decreasing its HP

enemy.getPokemonList(CurrentEnemyPokemonInt).takeDamage(damageTaken)

End If

It will check if the selected move use is greater than 0, if so it will allow the user to use the move. it will then decrease the selected move uses by 1. It then generates a random number between 0 and 1. It then checks if the random number is less than or equal to the moves accuracy, if so it will perform the move else it will not. It will then apply an effect if the move has one and applies the damage to the enemy Pokémon which is calculated in the damageCalculation function.

#### Player item

selectedItem = player1.getBag(CInt(pointerPosition.Y))

'Removes the item so it can't be re used

player1.removeItem(CInt(pointerPosition.Y))

battleState = "useItem"

'Executes the next code if the selected itme isn't a pokeball

Case False

'Applies the item effect to the Player Pokemon

processEffect(player1, enemy, "PS", selectedItem.getStat, selectedItem.getValue)

All items, except from pokeball, will apply an effect to the players Pokémon. Items can only be used in battles. It will store a copy of the item, remove the item from the player and then apply the effect to the users Pokémon.

#### Player switch

If player1.getPokemonList(CInt(pointerPosition.Y)).giveHP > 0 And pointerPosition.Y <> CurrentPlayerPokemonInt Then

'Set to false as the switch is finished

switchingFaintedPokemon = False

CurrentPlayerPokemonInt = CInt(pointerPosition.Y)

'Adds which pokemon the player sent out

logText = logText + " " + player1.getName + " sent out " + player1.getPokemonList(CurrentPlayerPokemonInt).giveName + " "

battleState = "displayLog"

End If

It will get the selected pokemons position and check if it is the current pokemons position the players pokemon list also it will check if the selected pokemon has no health points left if wither of these are true then it will not allow the player to make a switch. Else it will set the current PlayerPokemonInt to the value of the selected pokemons positon in the players pokemon list.

### Movement and Collision

#### Initial thoughts

My initial thoughts for collision between players and walls in the maze would be to check the cell where the player is and the cell where the player is trying to move to then check the corresponding walls in the cells if they are both false then the player would be able to move.

Example:

If currentState.IsKeyDown(Keys.W) Then

If currentMap.tile(x, y).getNorth = False And currentMap.tile(x, y - 1).getSouth = False Then

y -= 1

End If

...

So, if the player wanted to move up in the maze by pressing W, it would check if the corresponding wall in the current cell, in this case North, and the wall in the cell the player wanted to move to, in this case South, are both false (not there), allowing the player to decrease there Y position.

However, this didn’t work when the player was on edges of the maze and wanted to move out of bounds it would try to check for a tile which didn’t exist causing the game to crash.

#### Revised solution

After rethinking my original concept from scratch, I have come up with this solution

If Game1.kbState.IsKeyDown(Keys.W) Then

If Not py = 0 Then

If Not py Mod 16 = 0 Then

py -= 1

Else

If Game1.currentMap.tile(x, y).getNorth = False Or Game1.currentMap.tile(x, y - 1).getSouth = False Then

py -= 1

y = CInt(Math.Floor(py / 16))

End If

End If

End If

This doesn’t allow users to walk through walls as well as preventing crashes when the user reaches the edges of the map. It does this by first checking if the user is in an extreme cell – a cell at an edge, if not then it will perform the same checks as before, else it will check if it isn’t at an extreme pixel, else it will allow movement within the extreme cell.

### Inventory management

#### Capture pokemon

Case "useItem"

'If the selected item is a \*ball eg. a pokeball then the next code will execute

Select Case selectedItem.getName.Contains("ball")

Case True

'The player can only use pokeballs in encounters

If isEncounter = True Then

'Stores a random number

Dim randomSuccessChance As Decimal = CDec(Rnd())

'If the random chance is less than 0.75 then the player will catch the pokemon, effectively giving the player a 3/4 to catch the pokemon

If randomSuccessChance <= 0.75 And player1.getPokemonList.Count < 6 Then

'Gives the player the pokemon

player1.getpokemon(New Pokemon(enemy.getPokemonList(CurrentEnemyPokemonInt).giveName, enemy.getPokemonList(CurrentEnemyPokemonInt).giveLevel, enemy.getPokemonList(CurrentEnemyPokemonInt).giveMaxHP, enemy.getPokemonList(CurrentEnemyPokemonInt).giveMoves(0), enemy.getPokemonList(CurrentEnemyPokemonInt).giveMoves(1), enemy.getPokemonList(CurrentEnemyPokemonInt).giveMoves(2), enemy.getPokemonList(CurrentEnemyPokemonInt).giveMoves(3), enemy.getPokemonList(CurrentEnemyPokemonInt).giveType, enemy.getPokemonList(CurrentEnemyPokemonInt).giveAttack, enemy.getPokemonList(CurrentEnemyPokemonInt).giveDefence, enemy.getPokemonList(CurrentEnemyPokemonInt).giveSpeed, enemy.getPokemonList(CurrentEnemyPokemonInt).giveTextureAsString, enemy.getPokemonList(CurrentEnemyPokemonInt).giveID))

'Removes the pokemon from the enemy

enemy.removePokemon(CurrentEnemyPokemonInt)

In an encounter battle (A battle commenced in the tall grass, where it is possible to capture the enemy Pokémon) this checks if the used item is a pokeball, it will create a random decimal number, if it is less than 0.75 and the player has room in there Pokémon list (less than 6 current Pokémon) then the Pokémon will be caught, by giving the player the Pokémon and removing it from the enemy

#### Remove Pokémon

When in the players Pokémon inventory

ElseIf Game1.kbState.IsKeyDown(Keys.R) And Game1.player.getPokemonList.Count > 1 Then

Try

'Removes pokemon at pointer position

Game1.player.removePokemon(getPokemonPosition)

Catch ex As Exception

End Try

When a player wants to get rid of Pokémon, due to it having no health left or wanting to free up inventory space they can open their Pokémon inventory from the menu and press R on the selected Pokémon to remove it, the Pokémon is then removed from their Pokémon list.

#### Chest Inventory

'Gets the pointers X position - left is players inventory, right is chest inventory

Select Case pointerPosition.X

Case 0

Try

'If the chest doesn't have a full inventory

If Game1.currentMap.getEntity(Game1.player.givex, Game1.player.givey).getBag.Count < 6 Then

'Gives the chest the item

Game1.currentMap.getEntity(Game1.player.givex, Game1.player.givey).getitem(Game1.player.getBag(CInt(pointerPosition.Y)))

'Removes the item from the player

Game1.player.removeItem(CInt(pointerPosition.Y))

End If

Catch ex As Exception

End Try

When the player interacts with a chest a GUI will be displayed with the players inventory on the left and the chest inventory on the right, if the cursor is on the left then the X position is 0 if it is on the right then the X position is 1. If the x position is 0 then its in the players inventory thus when the enter key is pressed the item is removed from the players bag and is added to the chest bag, and visa versa if the x position is 1.

## Data structures

### Game

Imports System.Windows.Forms

Public Class Game1

Inherits Microsoft.Xna.Framework.Game

'Handles the configuration and management of the graphics device

Public Shared WithEvents graphics As GraphicsDeviceManager

'Stores sprites to be rendered

Private WithEvents TileTextures As SpriteBatch

'Stores current game state, controls which game logic/ drawing

Public Shared gameState As String

'Stores they keyboars inputs

Public Shared kbState As KeyboardState

Public Shared ContentLoader As ContentManager

'The font which is used for in game text

Public Shared font As SpriteFont

'Stores the games battle/ encounters

Public Shared gameBattle As Battle

'Stores the games map

Public Shared currentMap As Map

'The player in the game

Public Shared player As Player

'A pop textbox/ menus/ GUIS

Public Shared messageBox As MessageBox

'Stores all pokemons in the game

Public Shared pokedex As List(Of Pokemon)

'Stores all the moves a pokemon can havein the game

Public Shared movedex As List(Of Move)

'Stores all the items in the game

Public Shared itemdex As List(Of Item)

'Boolean to determine if a song is playing or not

Public Shared playingSong As Boolean

'Sound effect for when player selects a main menu option or interacts with a NPC/CHEST/SIGN

Public Shared pressA As SoundEffect

'Sound effect for when the player gets an item

Public Shared recieveItem As SoundEffect

'An enemy specifically for encounters in tall grass, has one encounter pokemon

Public Shared encounterPokemon As Player

'Map height

Const MapHeight As Integer = 32

'Map width

Const MapWidth As Integer = 32

'The current song playing idle/Main menu/Battle

Private currentSong As Song

'File location for data for game1.pokedex

Const pokedexFileLocation As String = "content/pokemon.csv"

'File location for data for movedex

Const movedexFileLocation As String = "content/Move index.csv"

'File location for data for itemdex

Const itemdexFileLocation As String = "Content/items.csv"

'File location for sprite font

Const fontFileLocation As String = "font"

'File location for idle song

Const idleSongLocation As String = "idleSong"

'File location for battle song

Const battleSongLocation As String = "battleSong"

'File location for main menu song

Const mainMenuSongLocation As String = "mainMenuSong"

'File location for recieve item sound effect

Const recieveItemEffectLocation As String = "recieveItem"

'File location for selection sound effect

Const pressAEffectLocation As String = "pressA"

'Creates new game

Public Sub New()

'Loads item file data into itemdex

Sub loadItems(ByVal fileLocation As String)

'Loads move file data into movedex

Sub LoadMoves(ByVal fileLocation As String)

'Loads pokemon file data into pokedex

Sub LoadPokemon(ByVal fileLocation As String)

'Updates game logic

Protected Overrides Sub Update(ByVal gameTime As GameTime)

'Saves a game

Sub saveGame()

'Loads game

Sub loadGame()

'Creates a new game, by setting the game1 attributes to new values

Sub newGame()

'Draws the main menu

Sub MainMenuDraw()

'Updates MainMenu logic

Sub MainMenuUpdate()

'Converts a string to a list for the log

Public Shared Function convertToList(ByVal newText As String) As List(Of String)

### Map

'Map class stores the Interactle parts of the map,NPCs as well as where the player can traverse

Public Class Map

'The map height in number of cells

Private numberOfRows As Integer

'The map width in number of cells

Private numberOfColumns As Integer

'The maze which the player walks in

Private maze(,) As Cell

'The NPCs the player can interact with

Private entityMap(,) As Player

'Used to store cells in the depth withs search with recursive backtracking alogrithim

Private stackOfCells As Stack(Of Cell)

'Stores the file loaction for the tiles used when drawing the game

Private tileFileLocation As String

'Stores how much time has passed in the map

Private timer As New TimeSpan

'Creates a new map, this is the contstructor class, which will create a new map when one is intantiated.

Sub New(ByVal x As Byte, ByVal y As Byte, ByVal exisiting As Boolean, ByVal saveLocation As String)

'Generates the maze

Sub Algorithim(ByRef current As Cell)

'Returns a random neighbourgh next to a given cell

Function CheckNeighbourghs(ByRef current As Cell) As Cell

'Removes the walls between two given cells

Sub RemoveWalls(ByRef current As Cell, ByRef neighbour As Cell)

'Creates empty spaces within the maze for tall grass

Sub generateRoom()

'Adds Entities to the map - NPCs, SIGNS and CHESTs

Sub populateMapWithNPC(ByRef pokedex As List(Of Pokemon), ByRef itemdex As List(Of Item))

'Returns a specific tile from within the maze

Function tile(ByVal x As Integer, ByVal y As Integer) As Cell

'Returns the height of the map in cells

Function getHeight() As Integer

'Returns the width of the map in cells

Function getWidth() As Integer

'Returns at entity as specific coordinates of the entity map

Function getEntity(ByVal xPos As Integer, ByVal yPos As Integer) As Player

'Updates the maps logic

Sub Update(ByVal playerXpos As Integer, ByVal playerYpos As Integer, ByVal gameTime As GameTime)

'Loads the in game menu options and returns them as a list of strings

Function loadMenuOptions() As List(Of String)

'Draws the map

Sub drawAll(ByRef tiles As SpriteBatch, ByVal mapHeight As Integer, ByVal mapWidth As Integer, ByVal playerXpos As Integer, ByVal playerYpos As Integer)

'Returns the map save data

Function getSaveConstAsString() As String

'Returns the maze save data

Function getSaveMazeList() As List(Of String)

'Returns the entity map save data

Function getEntityMapList() As List(Of String)

### AiSwitchChoice

'A class to store the percentage chance of the SwitchAi choosing a specific pokemon and the location within the enemy's pokemon list

Public Class AI\_Switch\_Choice

'Stores the percentage chance as a decimal

Public value As Decimal

'Stores the position in the enemy's pokemon list, byte was chosen as maxmimum pokemon list size is 6

Public position As Byte

'Creates a new switch choice

Sub New(ByVal newValue As Decimal, ByVal newPosition As Byte)

### Battle

'A class to store battles

Public Class Battle

'Keep count of enemy pokemon which have fainted (hp = 0)

Private enemyFainted As Integer

'Keeps count of player pokemon whcih have fainted

Private playerFainted As Integer

'Boolean value to dtermine if the battle is an encounter or not, if true then player can catch pokemon else they can't

Private isEncounter As Boolean

'Stores the type effectivness as a dictionary stored as key defender then attacker and the definition being the effectiveness multiplier

Private effectiveness As New Dictionary(Of String, Decimal)

'Stores a list of effects that moves can do

Private effects As New Dictionary(Of Integer, String)

'Stores the current player pokemon, which is in the battlefield, position in the player pokemon list

Private CurrentPlayerPokemonInt As Integer

'Stores the current enemy pokemon, which is in the battlefield, position in the enemy pokemon list

Private CurrentEnemyPokemonInt As Integer

'Stores the current state of the battle

Private battleState As String

'Stores the text which will be in the pop up text box

Private logText As String

'Stores the pointer position in the battle menus

Private pointerPosition As Vector2

'Boolean to decide if player/enemy is switching pokemon because pokemon fainted

Private switchingFaintedPokemon As Boolean

'Creates a new battle

Sub New(ByRef player1 As Player, ByRef enemy As Player, ByVal isEncounterBoolean As Boolean)

'Loads the effectivness file into the dicitonary

Sub loadDictionary(ByVal filePath As String)

'Loads the effect file into the effect dictionary

Sub loadEffects(ByVal filePath As String)

'Updates the battle logic

Sub Update(ByRef player1 As Player, ByRef enemy As Player, ByRef Tiles As SpriteBatch)

'Draws the battle

Sub Draw(ByRef battleTexture As SpriteBatch, ByRef player1 As Player, ByRef enemy As Player)

'The logic to decide if the player move will succeed or not aswell as the effect

Sub move(ByRef player1 As Player, ByRef enemy As Player, ByRef selectedMove As Move)

'This will determine if the AI will chose a move, item or switch pokemon

Sub battleAI(ByRef player1 As Player, ByRef enemy As Player)

'This will determine which Pokemon the AI will switch to

Sub switchAI(ByRef player1 As Player, ByRef enemy As Player)

'This will determine which move the AI will select,AI has infinite move usage to always produce optimal move selection

Sub moveAI(ByRef player1 As Player, ByRef enemy As Player)

'This will determine which item the AI will use

Sub itemAI(ByRef player1 As Player, ByRef enemy As Player)

'Processes a given effect, determines who the effect will target which stat and by how much

Sub processEffect(ByRef player1 As Player, ByRef enemy As Player, ByVal target As String, ByVal attribute As String, ByVal percentage As Decimal)

'Applies an effect to a targets stat by a pecentage, multiplys a stat by a percenatge

Sub applyEffect(ByRef target As Player, ByVal pokemonInt As Integer, ByVal attribute As String, ByVal percentage As Decimal)

'Sets the Y position of the pointer, aswell as a max limit

Sub SetPointerY(ByVal Ymax As Integer)

'Damage calculator

Function damageCalculation(ByRef attackerLevel As Integer, ByRef attackerAttack As Integer, ByRef defenderDefence As Integer, ByRef moveType As String, ByRef attackerType As String, ByRef defenderType As String, ByRef movePower As Integer) As Decimal 'returns a damage value dependent on power, attack/defence and effectiveness

### Cell

'A class to cells which make up the map

Public Class Cell

'X position in the map

Private xPosition As Integer

'Y position in the map

Private yPosition As Integer

'Boolean for the depth first search to store if it is visited or not

Private visited As Boolean

'Store if the north wall exists

Private North As Boolean

'Stores if the east wall exists

Private East As Boolean

'Stores if the south wall exists

Private South As Boolean

'Stores if the west wall exists

Private West As Boolean

'Stores the cells symbol, dependent on which walls exist

Private symbol As Integer

'Creates new cell, sets X and Y position, other attributes as defaulted

Sub New(ByVal nx As Integer, ByVal ny As Integer)

### Chest

'Class which stores the Chest entity

Public Class Chest

'A chest is like a player, which only has a bag

Inherits Player

'Creates a new chest

Sub New(ByVal nx As Integer, ByVal ny As Integer, ByRef itemdex As List(Of Item), ByRef existing As Boolean)

'The chest interaction when the player is on top of it and presses T

Public Overrides Sub interact()

'Returns all the chest attributes as a colon deliminated string

Public Overrides Function getSaveString() As String

### Item

'An item class

Public Class Item

'Item ID unique

Private ID As Integer

'Item name

Private name As String

'Stat item effect

Private stat As String

'decimal percenatge of how much it effects the stat

Private value As Decimal

'Item texture location

Private spriteLocation As String

'Item texture width

Private spriteWidth As Integer

'Item texture height

Private spriteHeight As Integer

'Creates new item

Sub New(ByVal c() As String)

### MessageBox

'Class to store and display in game GUIs

Public Class MessageBox

'Stores message box text

Private text As List(Of String)

'Constant for the message box texture height in pixels

Private textBoxHeight As Integer = 44

'Constant for the message box texture width in pixels

Private textBoxWidth As Integer = 250

'Constant for the pointer texture height in pixels

Private pointerHeight As Integer = 15

'Constant for the pointer texture width in pixels

Private pointerWidth As Integer = 19

'Constant for textbox size multiplier - for signs

Private textBoxMultiplier As Decimal = CDec(((Game1.graphics.GraphicsDevice.PresentationParameters.BackBufferWidth + Game1.graphics.GraphicsDevice.PresentationParameters.BackBufferHeight) / 2) \* (1.5 / 512))

'How far in pxiels is the text offset X direction

Private textOffsetWidth As Integer = 10

'How far in pxiels is the text offset Y direction

Private textOffsetHeight As Integer = 4

'Textbox texture

Private \_textBoxTexture As Texture2D

'Pointer texture

Private \_pointerTexture As Texture2D

'Boolean if the text in the message box is displayed

Private displayTextBoolean As Boolean

'Boolean if the pointer in the message box is displayed

Private displayPointerBoolean As Boolean

'Pointer position

Private pointerPosition As Vector2

'Boolean if the message box is displayed

Private displayMessageBox As Boolean

'Stores pokemon selection from - pokemon list/ pokemon info menu

Private selectedPokemon As Pokemon

'Creates a new message box

Sub New(ByVal listOfNewText As List(Of String), ByVal displayBooleanText As Boolean, ByVal newDisplayPointerBoolean As Boolean)

'Updates message box logic

Sub update()

'Draws the message box

Sub draw(ByRef Tiles As SpriteBatch)

### Move

'Move Class

Public Class Move

'Move ID - uniquw

Private ID As Integer

'Move name

Private name As String

'Move type

Private type As String

'Move base power

Private power As Integer

'Move accuarcy as decimal

Private accuracy As Decimal

'Number of times the move can used

Private PP As Integer

'Move priority, 0 or 1, 1 being most prioritised

Private priority As Byte

'Damage type, physical(Ph)/Special(SP/Status(St)

Private damageType As String

'Effect ID - unique

Private effectID As Integer

'Effect chance as decimal

Private effectChance As Decimal

'Creates new move

Sub New(ByVal c() As String)

'Decreases moves uses left by 1

Sub decreasePP()

'Returns the moves save data as a string

Function getSaveString() As String

### NPC

'NPC class

Public Class NPC

'A NPC is similar to a player, excepted it can be controlled, use pokeballs or move

Inherits Player

'Creates a new NPC

Sub New(ByVal nx As Integer, ByVal ny As Integer, ByRef pokedex As List(Of Pokemon), ByRef itemdex As List(Of Item), ByRef existing As Boolean)

'The NPC interaction when the player is on top of it and presses T

Public Overrides Sub interact()

'Returns all the chest attributes as a colon deliminated string

Public Overrides Function getSaveString() As String

### Player

'Player class - used for the user/ npc's/ chests/ signs and encounter pokemons

Public Class Player

'Stores players name

Protected name As String

'Stores players pokemon - max 6

Protected pokemons As New List(Of Pokemon)

'Stores players items - max 6

Protected bag As New List(Of Item)

'Stores players X position in the map

Protected playerXCoordinate As Integer

'Stores players Y position in the map

Protected playerYCoordinate As Integer

'Stores the players X position on the screen

Protected playerPixelXCoordinate As Integer

'Stores the players Y position on the screen

Protected playerPixelYCoordinate As Integer

'Creates a new player

Sub New()

'Updates the players logic - movement and collision

Public Sub update()

'Draws the players sprite

Sub drawAll(ByRef playerSprite As SpriteBatch)

'Get player's save data as a string

Overridable Function getSaveString() As String

### Pokémon

'Pokemon class

Public Class Pokemon

'Stores the pokemons uique id in then pokedex

Private ID As Integer

'Pokemons name

Private name As String

'Pokemons leavel

Private level As Integer

'How much health the pokemon has currently

Private HP As Integer

'How much health it begins with

Private MaxHP As Integer

'The 4 moves a pokemon has - max 4

Private useableMoves(3) As Move

'Pokemon type

Private type As String

'Pokemon attack stat - how good it is at damaging

Private attack As Integer

'Pokemon defence stat - how good it is at not taking damage

Private defence As Integer

'How fast the pokemon is

Private speed As Integer

'The pokemons sprite texture file location

Private spriteLocation As String

'Sprite width in pixels

Private spriteWidth As Integer

'Sprite height in pixels

Private spriteHeight As Integer

'Creates a new pokemon

Sub New(ByVal newName As String, ByVal newLevel As Integer, ByVal newHP As Integer, ByVal move1 As Move, ByVal move2 As Move, ByVal move3 As Move, ByVal move4 As Move, ByVal newType As String, ByVal newAttack As Integer, ByVal newDefence As Integer, ByVal newSpeed As Integer, ByVal newTexture As String, ByVal newID As Integer)

'Decreases the pokemons health my a given amount

Sub takeDamage(ByVal damage As Integer)

'Gets the save data for the pokemon - all the attributes

Function getSaveString() As String

### Sign

'A sign class interactable tile entity

Public Class Sign

'Its like a player as it has a name

Inherits Player

'Stores the signs text

Private text As List(Of String)

'Booelean to represent if the text is displayed or not

Private displayText As Boolean

'Creates a new sign

Sub New(ByVal nx As Integer, ByVal ny As Integer, ByVal newText As String)

'The signs interaction when the player is on top of it and presses T

Public Overrides Sub interact()

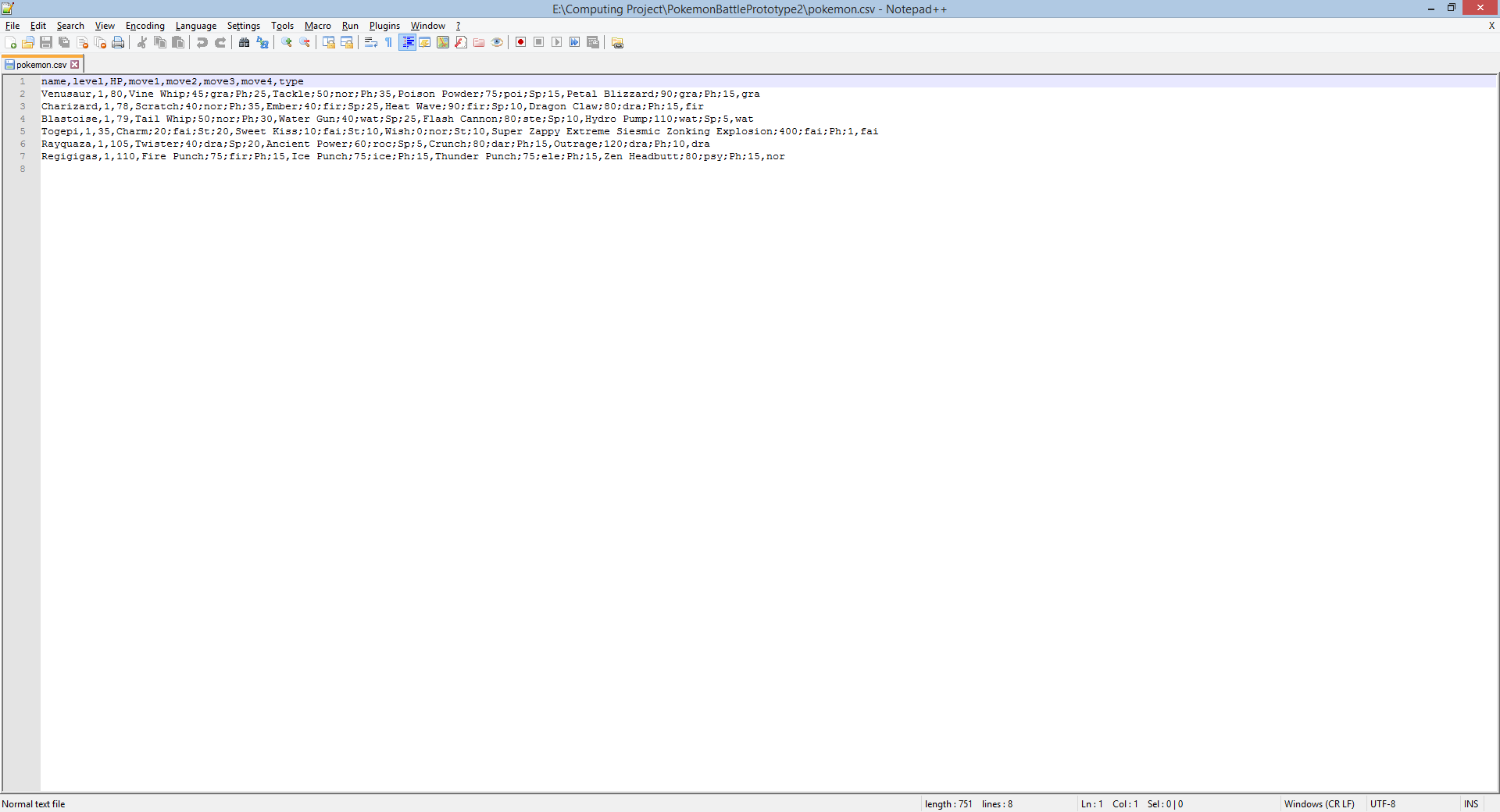
'Returns the save data of the singn as a string

Public Overrides Function getSaveString() As String

## File structure and organisation

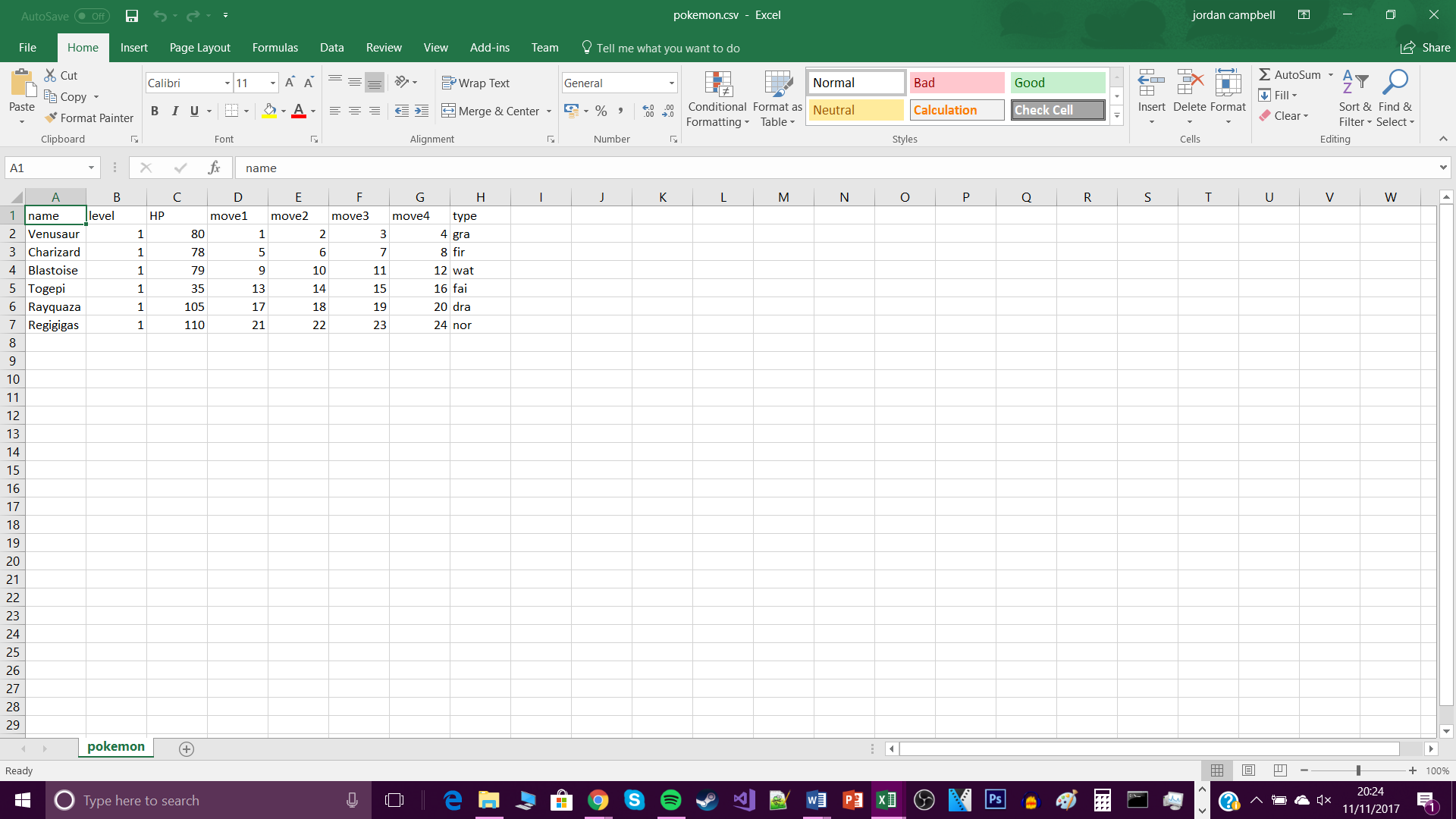
For my game I will need 3 files which will contain constant data, the Pokémon in the game, the items in the game and the effectiveness table. Each are in a CSV format as it gives a structure to the data making it easier to read especially when using an editor such as Microsoft Excel, making it easier to change as well as adding new data to the table. On the first line are headings which are the property titles of the objects, these allow someone who didn’t program this to easily understand the file as they will know what each component is. These files will only be read from and be written to.

### Pokemon file



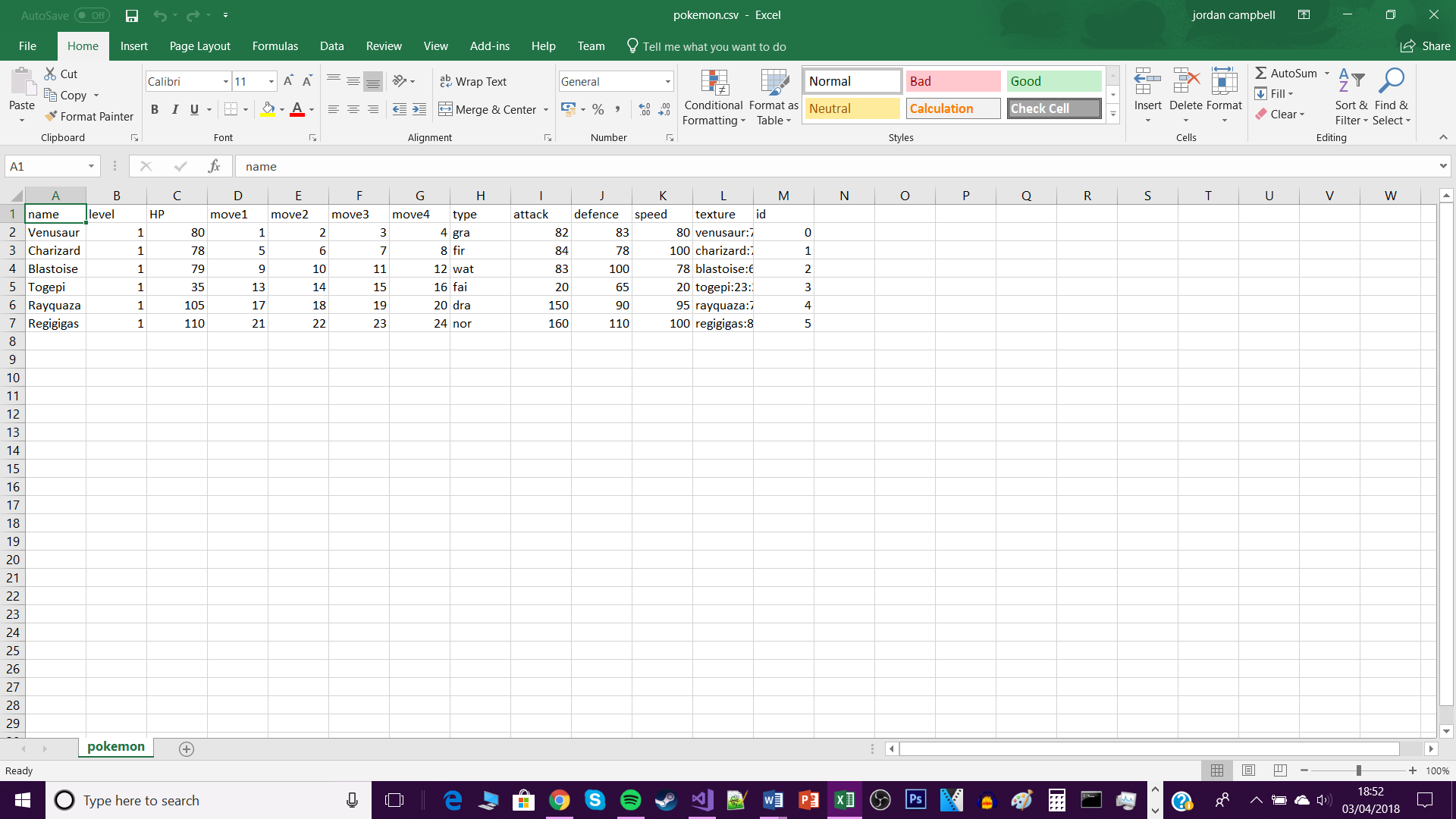
This file stores all the **pokemon** in the game. Each line, excluding the first line, will store the basic standard poroperties of a pokemon, storing the headings as shown about the pokemon. Move1 for example will store move name, the base attack, the typing, the move type and uses.

#### Updated file



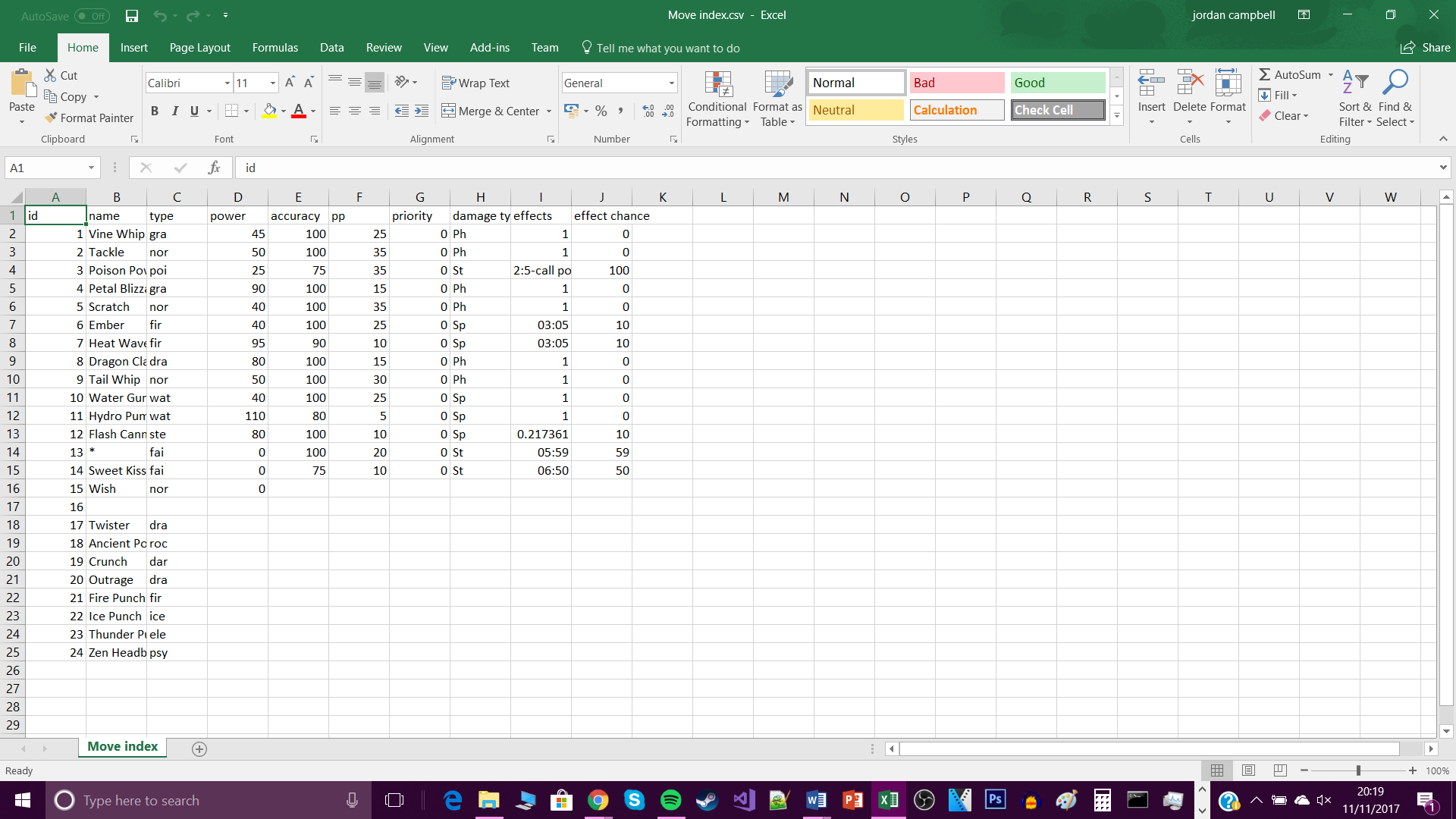
After a revision I altered the file format slightly, so that instead of each move being written in full for each pokemon instead each move will have an id, which will refer to a specific move in the move index file.

#### Updated file v2



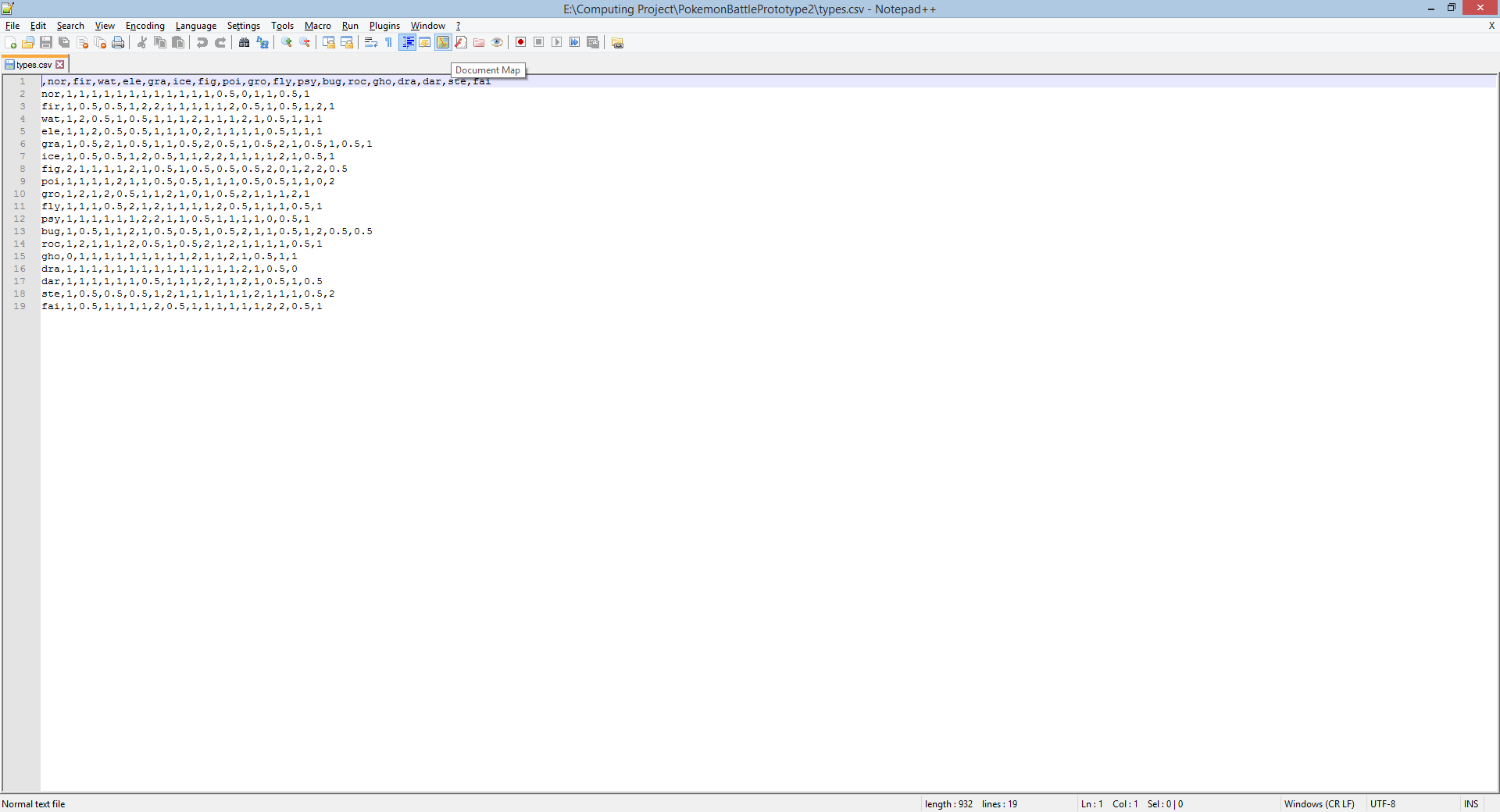
I have now included more statistics about the Pokémon, such as attack, defence and speed allowing for more in-depth battle calculations, I have also stored information about the Pokémon’s sprite, such as its location and its width and height in pixels. I have given each Pokémon a unique Pokémon id.

### Move file



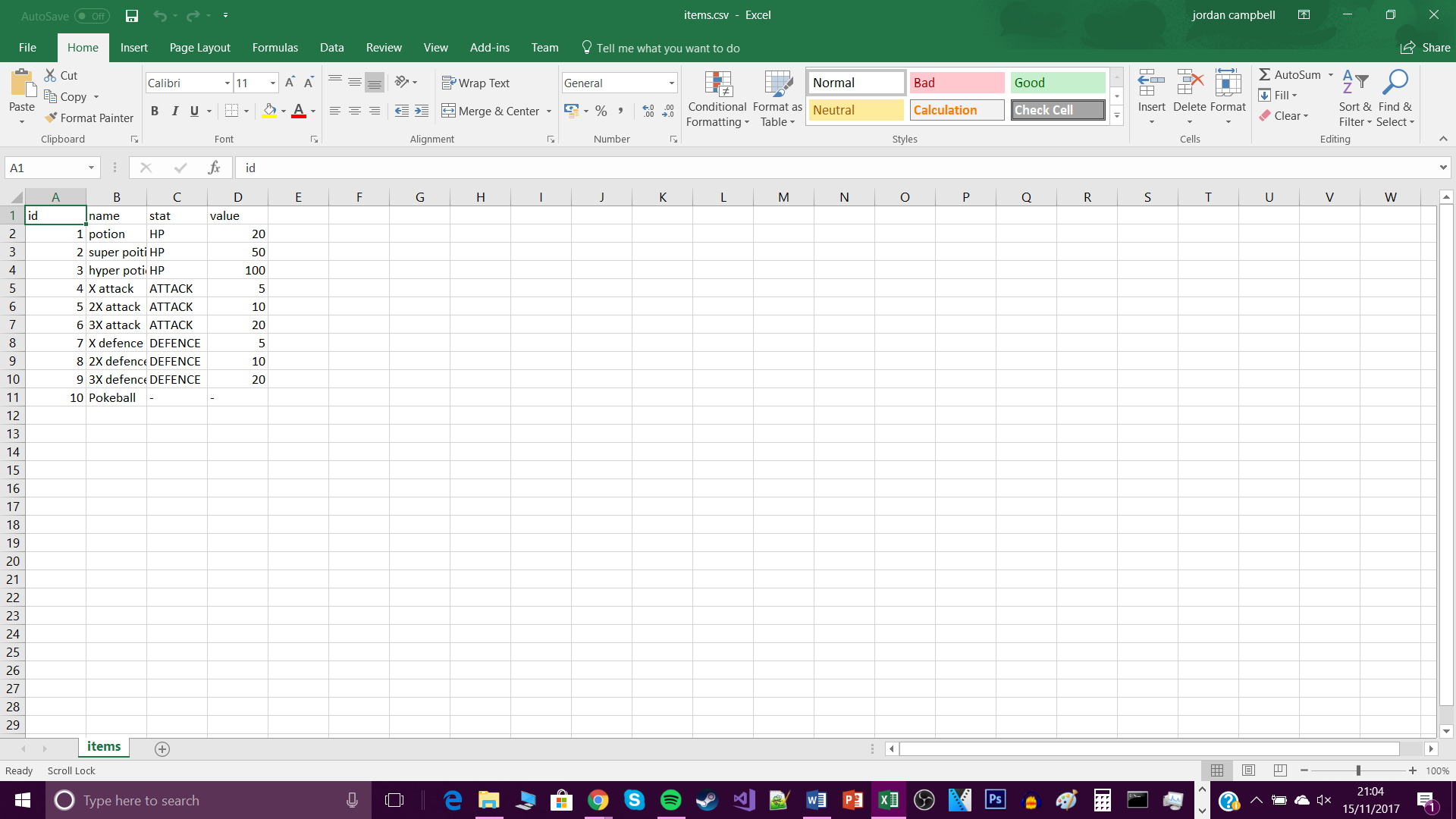
The move index file will store all **moves** which are within the game. This file contains all the components for the move such as id, name, type, power and so forth. It also makes refernce to specific effects which have separate IDs as well, which will be hard coded in the game.

### Move effectiveness file



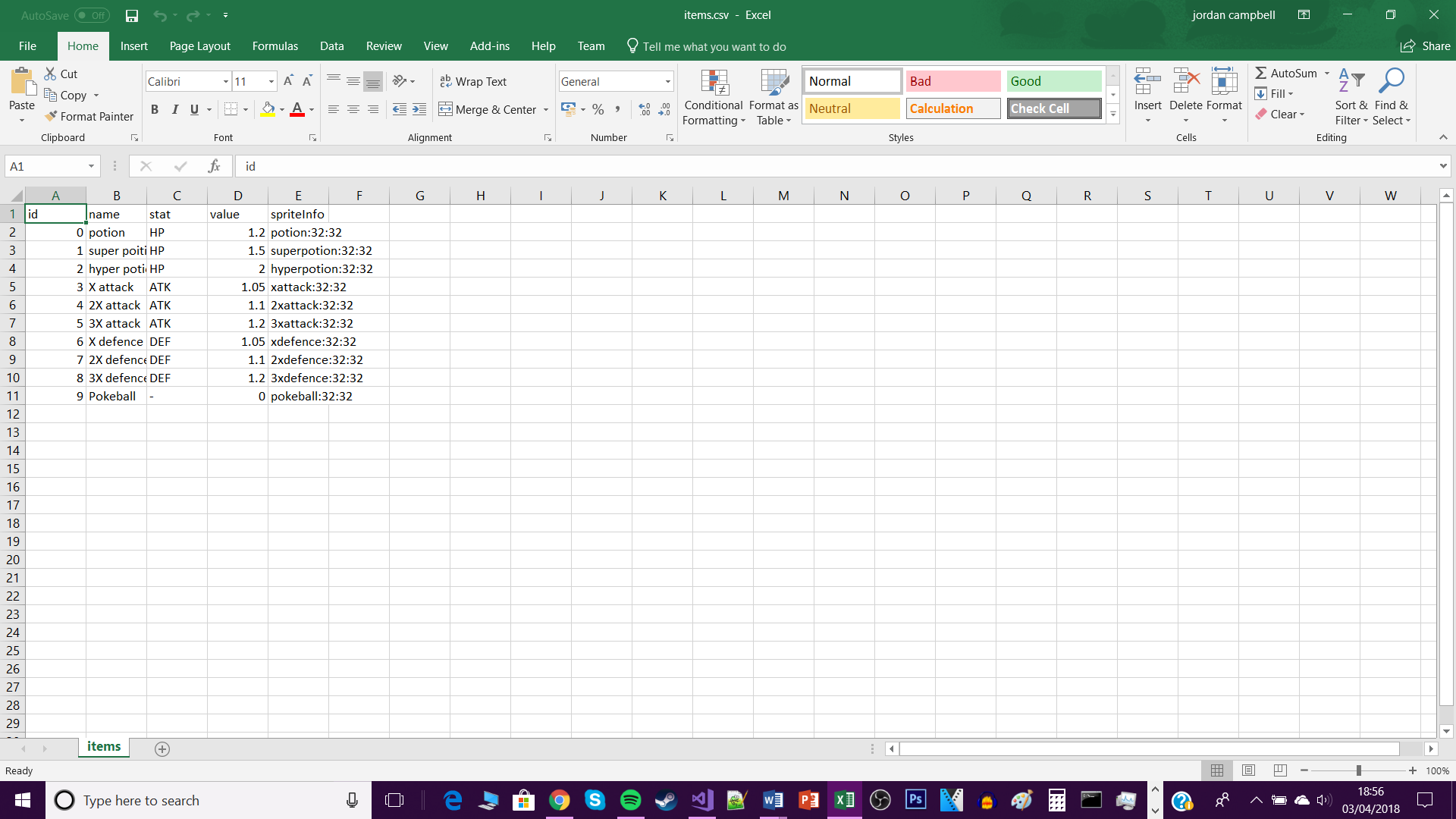
This file will store the **move effectiveness** table. The acronmys in the first column are move types so the attacking type and the acromnyms in the first row are pokemon type so the defending type. And then the corresponding number between he two if the attack multplier. For example if a water move was used against a fire pokemon then the multiplier would be 2.

### Item file



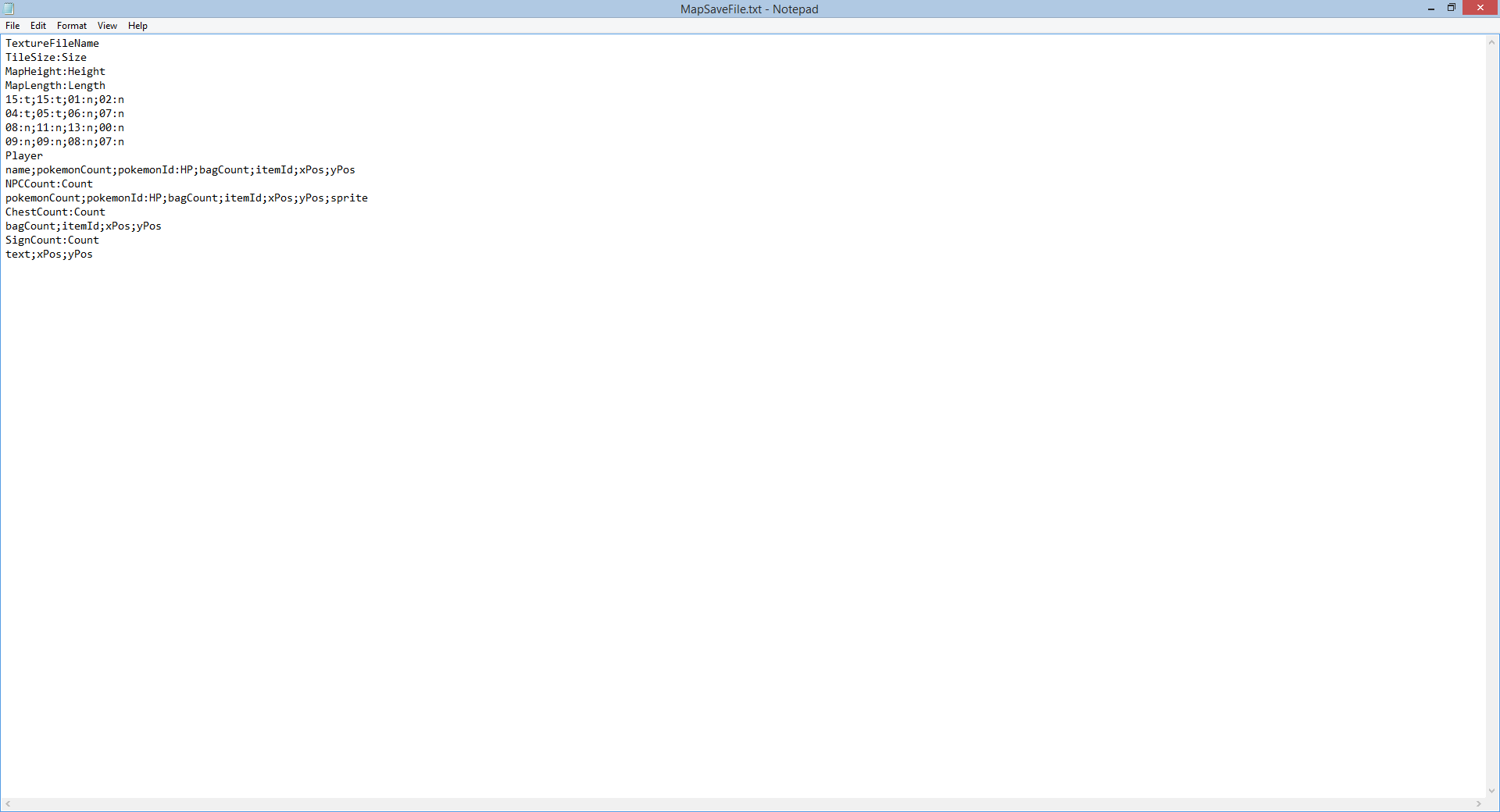
This file will store all the **items** in the game, this file will be similar to the file to store pokemon. Each line, excluding the first line, will store the basic standard poroperties of an item, storing the headings as shown about the item, name stat and value.

#### Updated file



In this updated file I have re index the items, to follow the VB listing structure, the stat names now have been abreviated for ease of use in code, stat values have now become multipliers instead of adding a base value, due to a change in design choice of how the game will work, sprite information has also been included, the sprite location as well as the width and height in pixels.

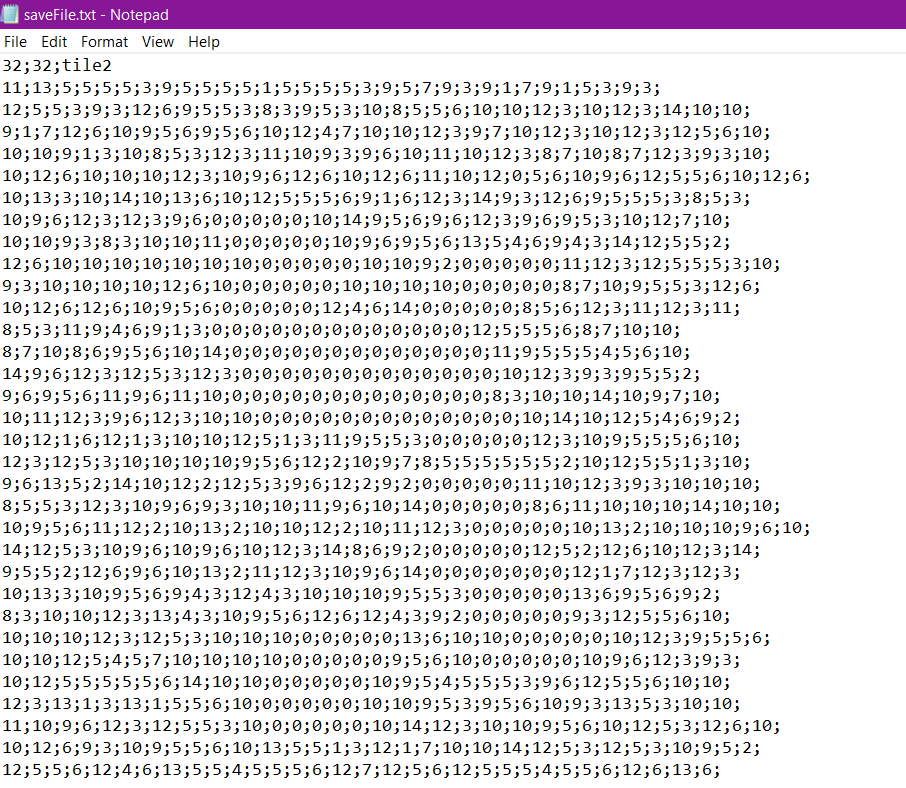
### Game save file

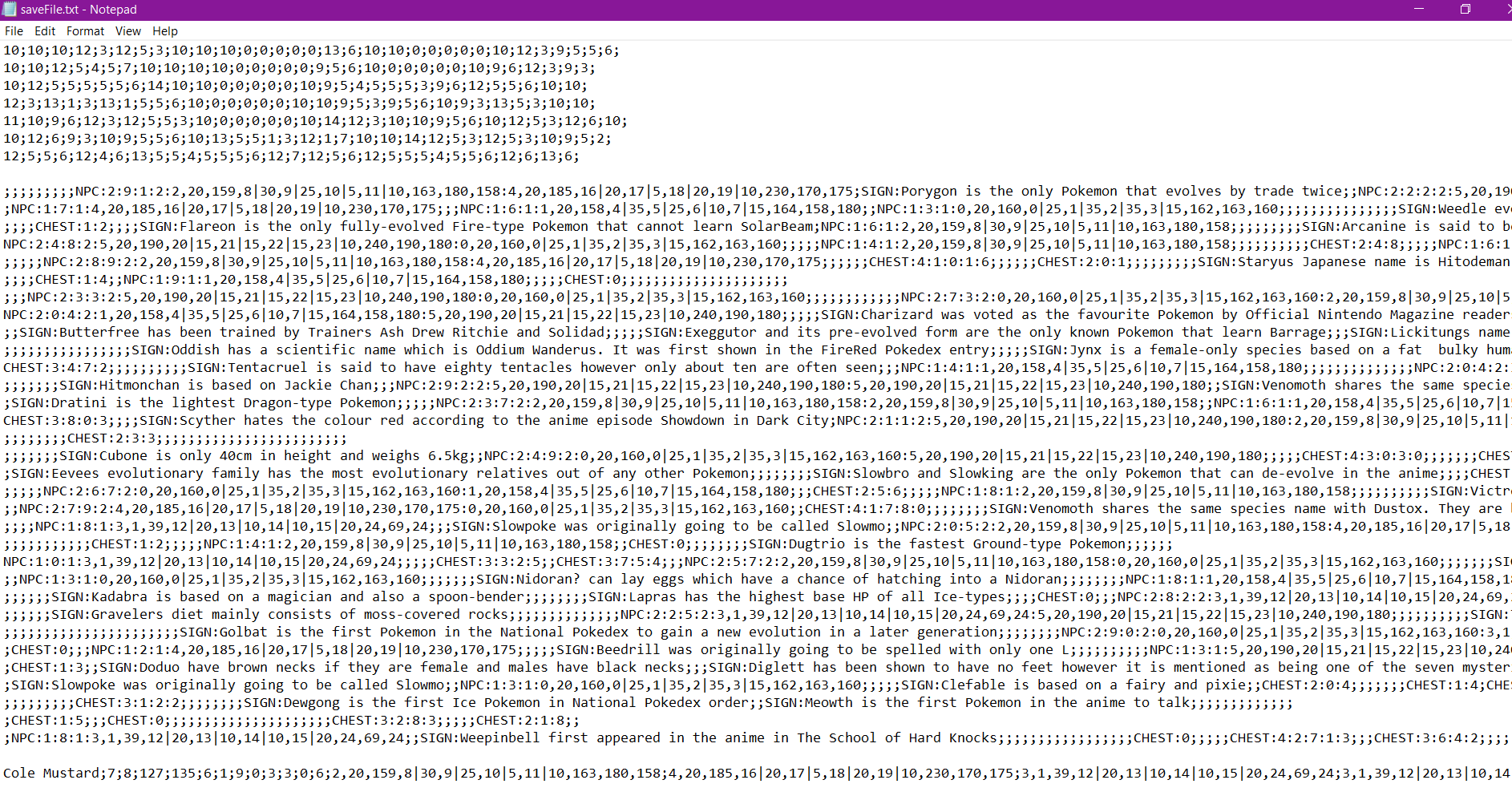


This is the structure I will use to save my **game file**

1. The first line will be the texture name so it can load specific terxtures depednog onmn the map type
2. The next line is the texture size, one number is needed as it will be a square
3. The next two lines dictace the actual map length and width which can be customiosed by the user before the game first player
4. The next n number of lines will be the tile map the length and width is dpependent on the two resious lines, each tile is seperated by semicolons and with in each semicoln is the tile texture number and a boolean to say if the tile will have tall grass or not
5. The next two lines are about the player. The players attritbutes are comma seperated.
6. The next section I will store NPC data similarly to player data
7. The section after that will styore data about chests
8. And the last section will store data about signs in the map.

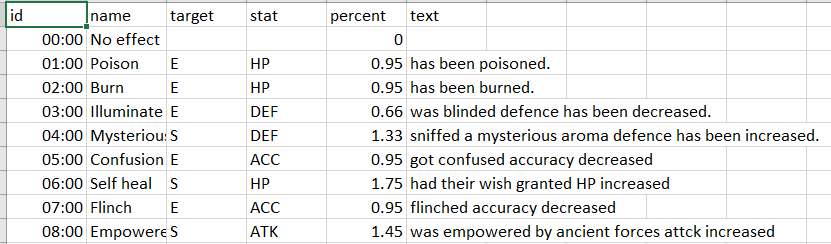
#### Updated file





This is an actual example of a save file used, the first line will have the map width and height in cells and the texture used for the tiles. Next will be a 2d array of the tiles. Then beneath that is a 2d array of the entity map, which includes the save data for npcs, chests and signs. Stored with their name first, then for npcs – bag count, item ids, pokemon count, pokemon save data – id plus current stat values, for chests it stores the bag count and the item ids, for signs it just stores their text. Then for the player it is the same as the npc but their x and y coordinates for in map and on screen included after their name but before the bag count.

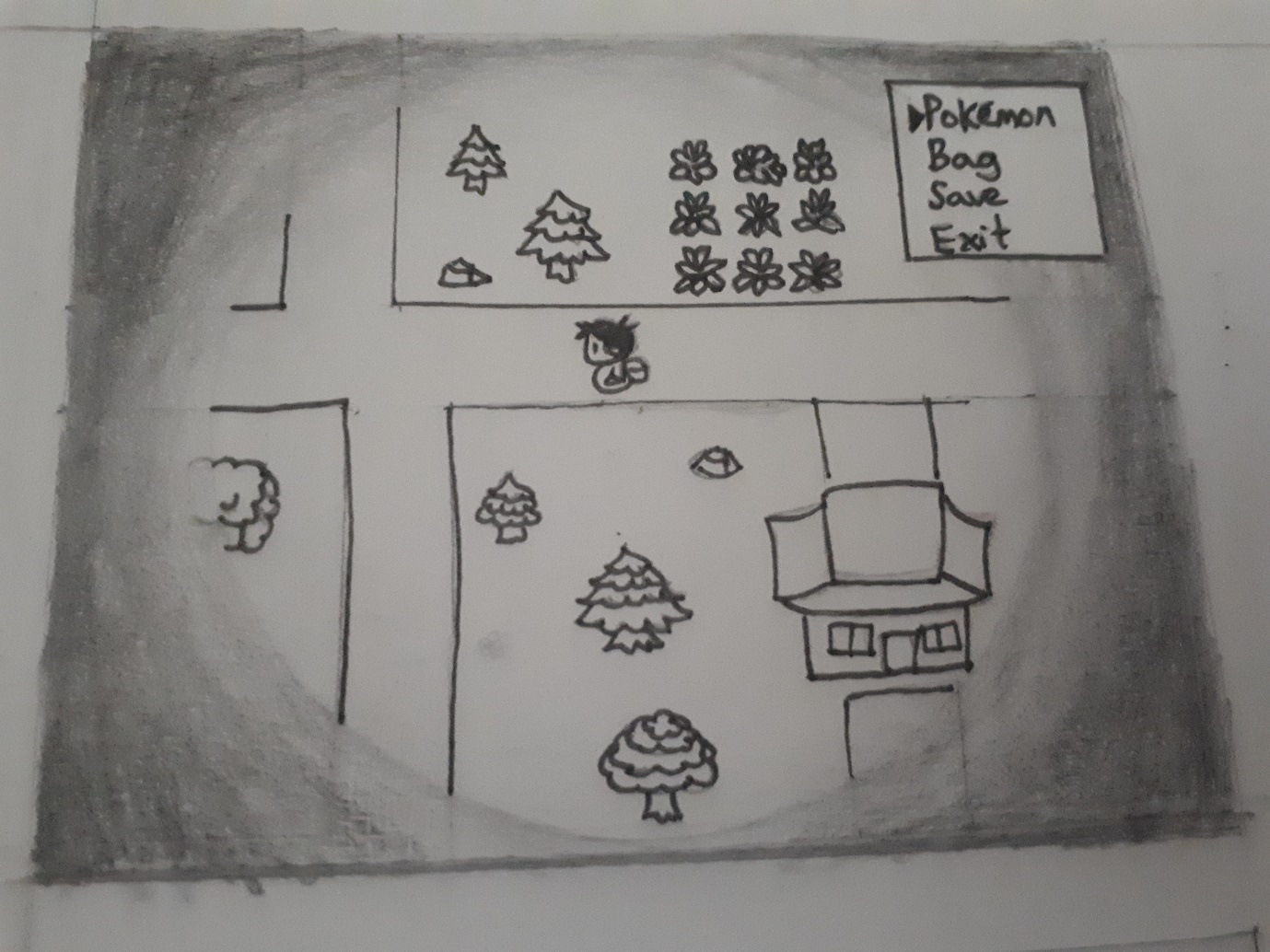
### Effects file



This is a csv file which stores the effect data, it stores the effect id, the name of the effect, the target it affects, E for enemy and S for self, then the stats, HP is hp, DEF is defence, ACC is accuracy and ATK is attack. Next is the decimal chance it will happen and finally is the text displayed associated with the effect.

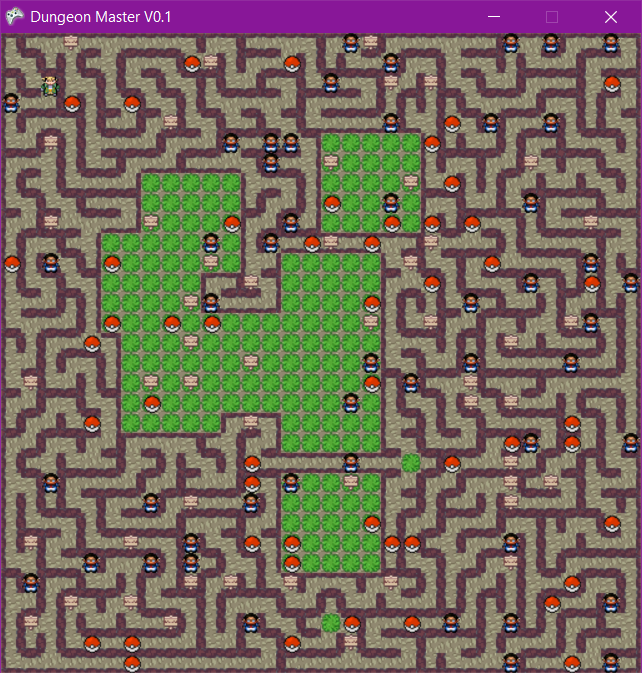
## HCI – human computer interface

### In game look



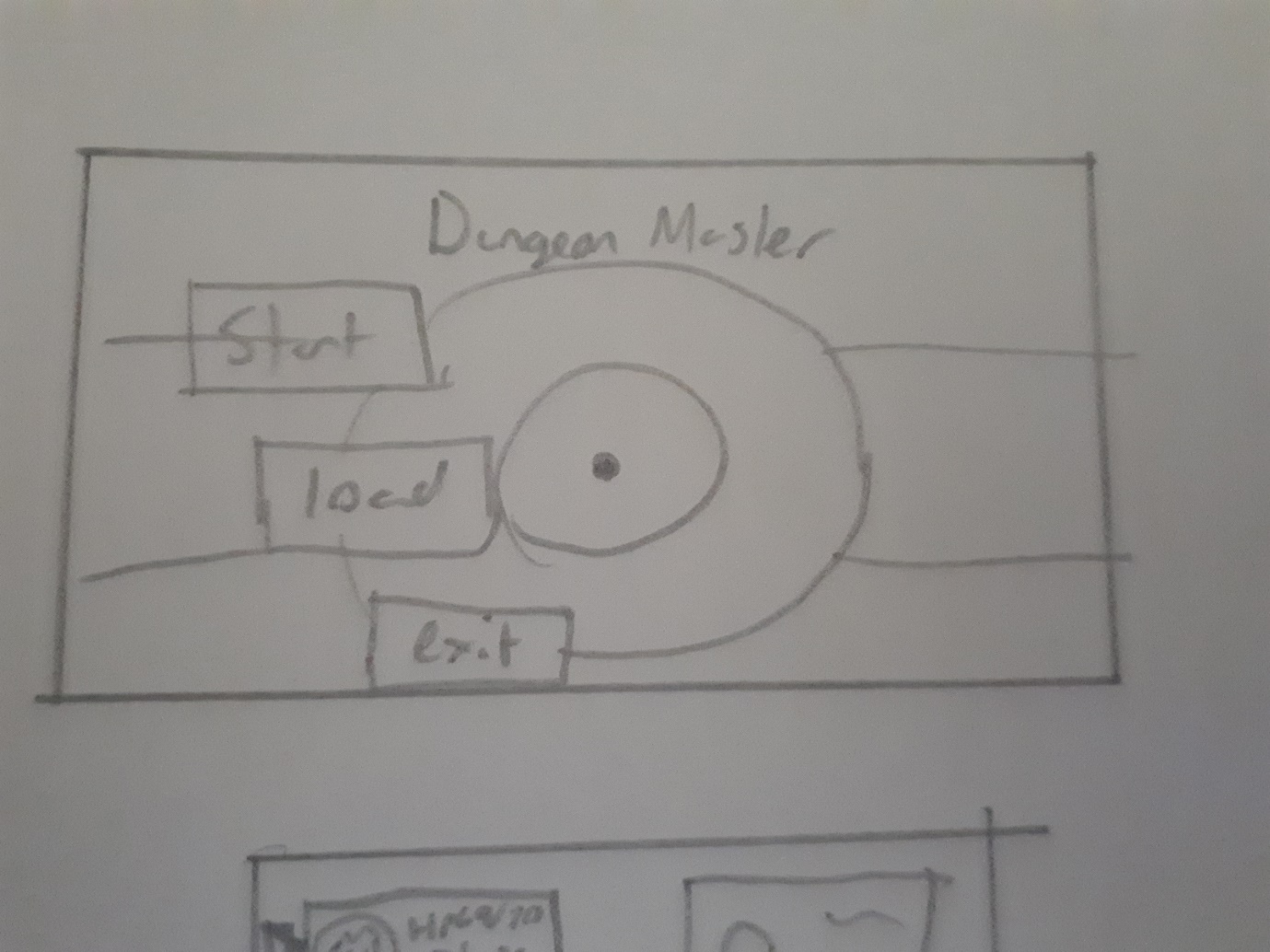
This will be how the game will look like, the player will be able to use the WASD to move around and when they are within one cell of a map object they can press T to interact with it. The player will be able to press M and the menu will be displayed, they can use the arrow keys to move the cursor and press enter to confirm their selection.

#### Updated Look



Due to end user feedback during development the design has been changed so that the full map is shown, and the torch overlay is removed. This made it so the user was less confused when traversing around the maze and the torch overlay was removed just to make the game generally clearer.

### Main menu

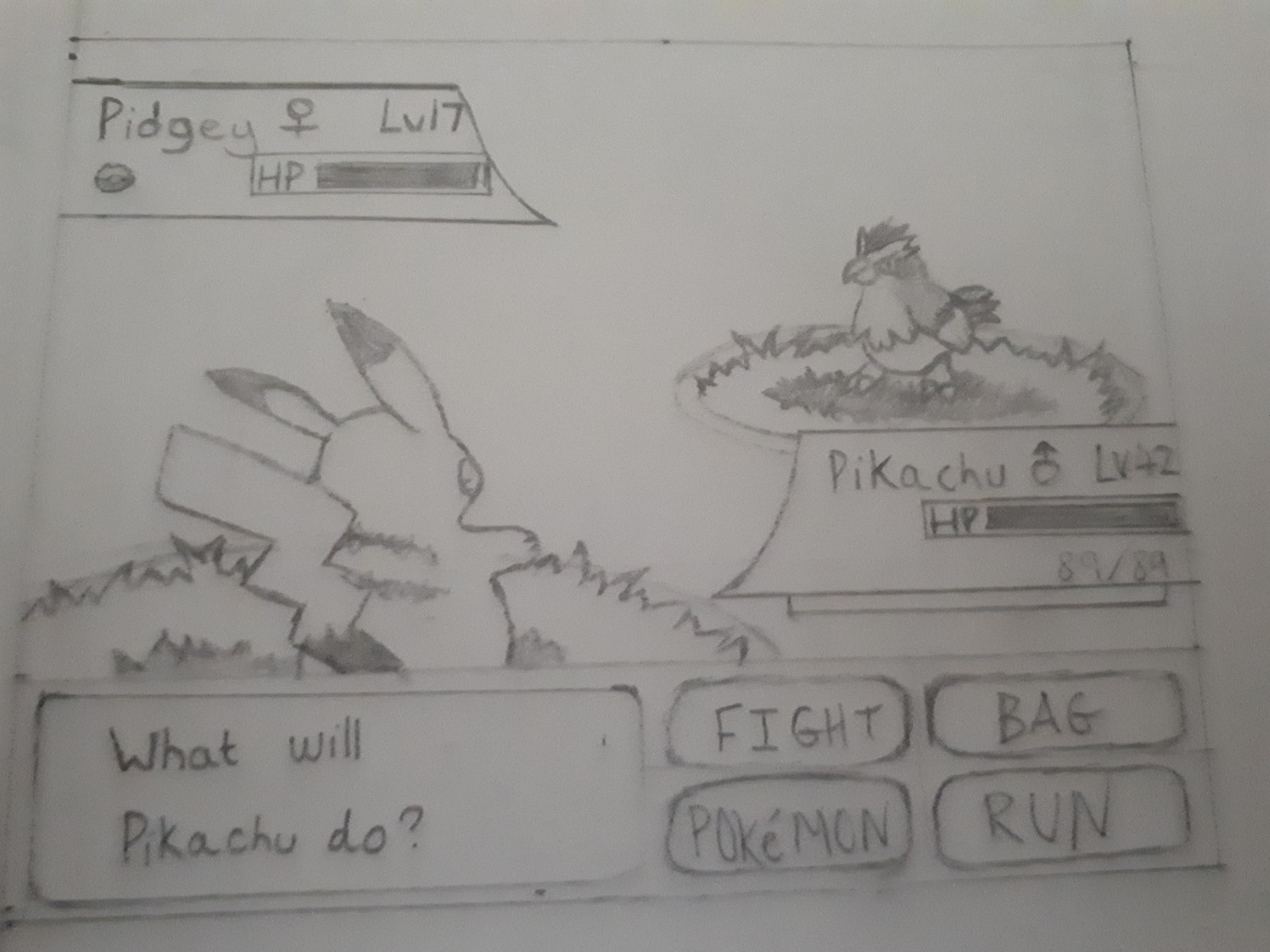


This will be the main game menu at the start of the game. It shows three options, play game which starts a new game, load which loads a saved game and exit which exits the program. The user will be able to use the arrow keys to select the option then the enter key to confirm the option.

#### Actual look



### In battle look



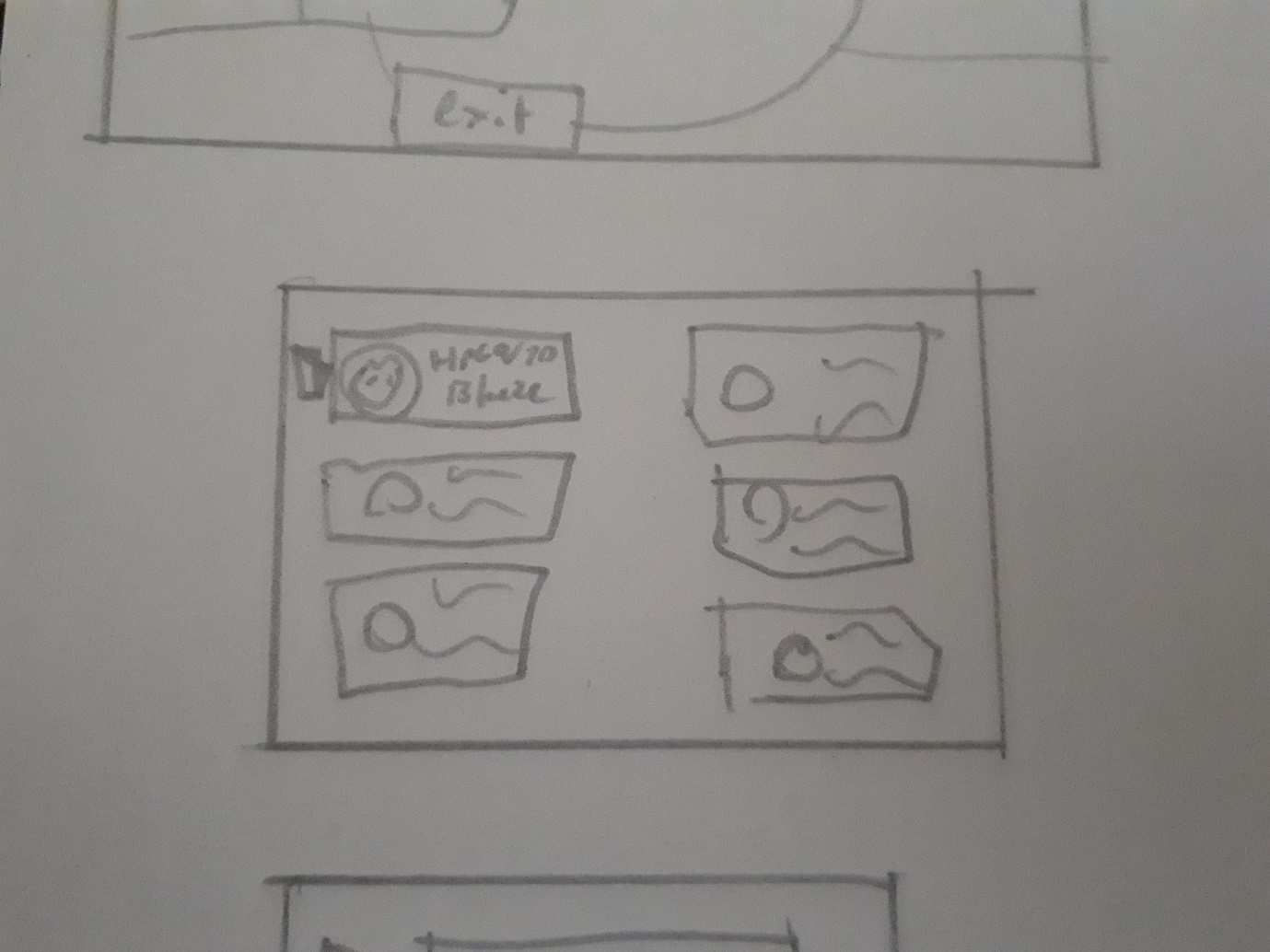
This will be the battle menu; the user will be able to use the arrow keys to select the option then the enter key to confirm the option. It shows 4 options which the user can select, fight, run, Pokémon and bag.

#### Updated look



The new design has been chosen to simplify the layout, so it can be more easily expanded to add more health bars for multiple Pokémon battles or more menu options.

### Pokemon selection menu

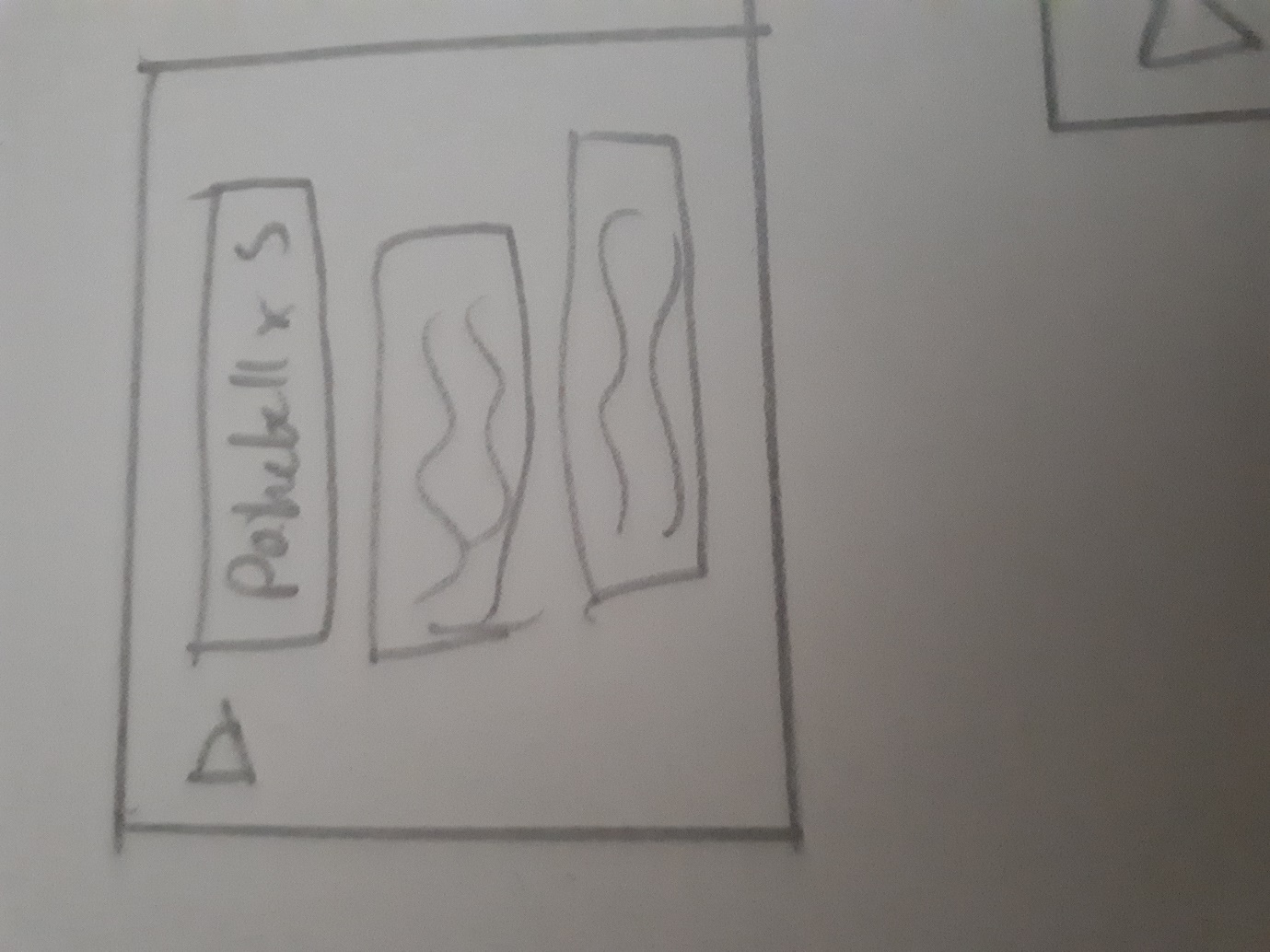


This will be the Pokémon menu; the user will be able to use the arrow keys to select the Pokémon they will want to send out then the enter key to confirm the option. Each option displays the pokemon name and its HP.

#### Actual look



### Item selection menu

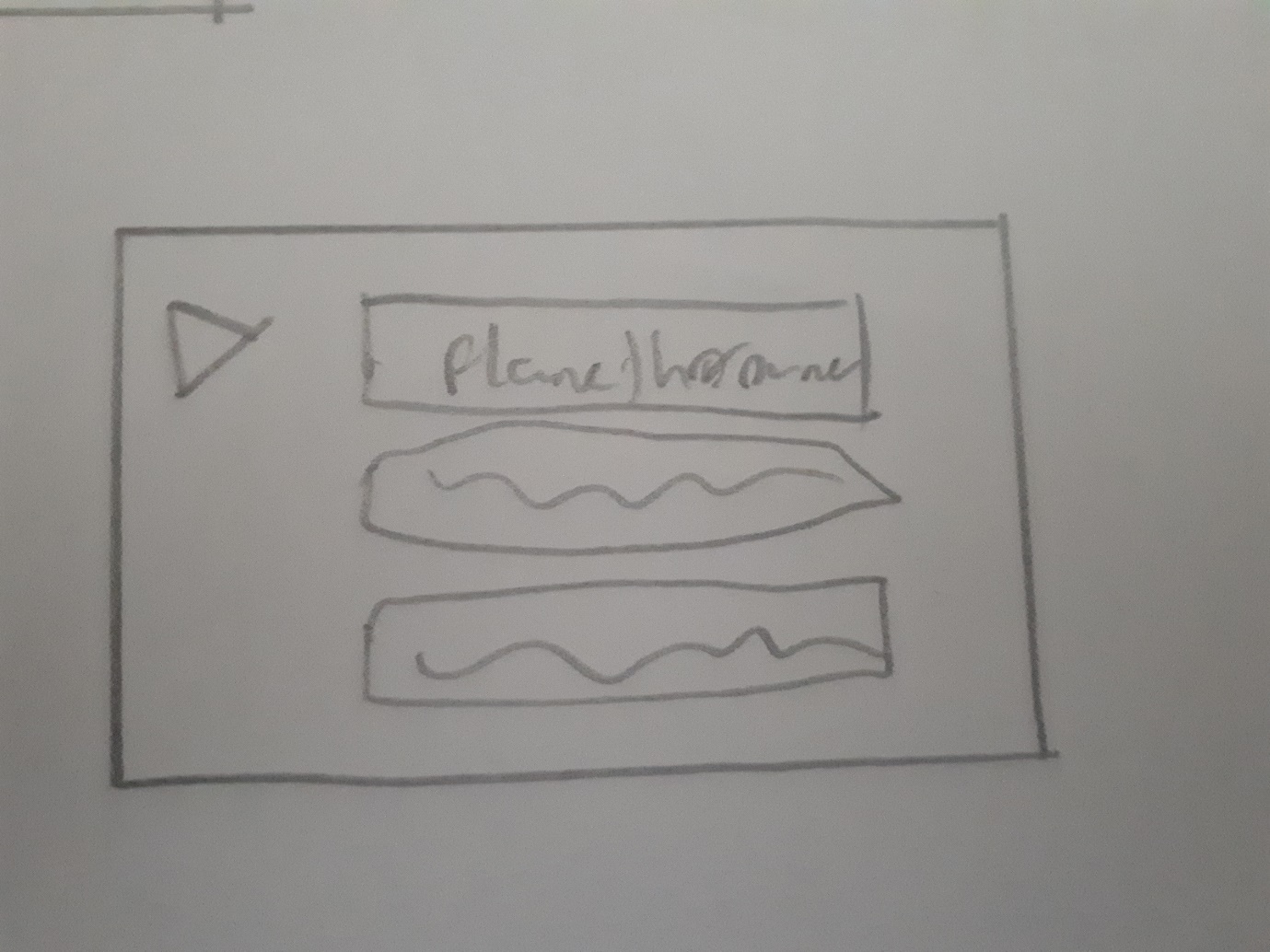


This will be the item menu; the user will be able to use the arrow keys to select the option for the item they want to use then the enter key to confirm the option. Each option will show the item name and the quantity of the item which you have.

#### Actual look



### Move selection menu



This will be the move selection menu; the user will be able to use the arrow keys to select the move option of the current Pokémon then the enter key to confirm the option. It displays the moves name.

#### Actual look



### Chest GUI – final



### Sign GUI – final



# Testing

## Choice of test data

My test scenarios will be using the menus and GUIs, traversing around the maze, such as collision with walls and maze boundaries, and interacting with the map, as well as battling against AI and catching pokemon. For boundary data I will be testing my collision between players and walls as well as the maze bounds. There is only one place for entering erroneous data which is loading a file such as using the wrong file format e.g. .png or the syntax of the .txt being wrong. The game input which is controlled using keys is defined by the program so inputting any erroneous data is impossible. I have limited my scope, so I will not expect my program to still function correctly if files are deleted.

## Testing plan

### Scenario testing.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Objective # | Description  Of objective | Situation/input | Ideal outcome | Actual outcome | comment |
| 1 | 1, 1a, 2 | Shows menu, can exit program | Press 1 | Program exits | Ideal outcome | Pass |
| 2 | 3a, 3b | Generates random maze, gives random items. The maze sections should include dead ends with no loops as it is depth first search algorithm. | Starts a new game | The maze is random each time it is newly generated and giving the player 6random items and pokemon. The mazes will have no loops and will feature dead ends. | Ideal outcome | Pass |
| 3 | 5, 7, 8 | Wasd movement, open display menu – menu function, open signs, open chest GUI, open NPC | Input commands for movement: W up, A left, S down, D right, I will try to walk through walls and out of the maze’s bounnds. E to open/close menus, Up and down arrow for menu movement, Enter key for selection. | WASD for movement, can’t walk through walls/ exit map. Pressing E opens menu, E to exit option. Open bag GUI, opens pokemon GUI, opens pokemon information GUI. Pressing T on top of, sign opens sign GUI, chests open chests GUI, NPC creates new battle. | Ideal outcome | User can move past last options in GUIs, not a bug but a lacking feature, arrow could stop at last option. |
| 4 | 6, 6a | Random encounter 0.25 chance, capture pokemon | Walking through tall grass | On average every 1 in for tall grass walked through creates a new encounter | Random encounters do occur; however, they took a lot longer than expected. | Partial pass |
| 5 | 9 | Battle AI, try to win/ try to lose | For trying to win, I will try to maintain type advantage and use the most effectives moves.  For trying to lose I will try not to have type advantage and use the least optimal moves. | The AI should be able to beat the player and the player should be able to beat the AI | The player can beat the AI. The AI when the player isn’t playing their best is able to beat the user.  Bug – States the Player wins even though AI still had pokemon remaining. | Due to the partial random selection the AI doesn’t always use the most optimal uses, also the percentage chances chosen made need tweaking |
| 6 | 4a, 4b | Open file explorer loads game | Loading an old game | Loads the game with the same NPCs and inventories for users and the same maze. | Ideal outcome. | Pass |
| 7 | Ext. 4 | Plays music depending on scenario |  | Plays menu music in menu, plays idle music in game, plays battle music in battle | Ideal outcome | Pass |

### Unit testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test number | Description | Situation/input | Ideal outcome | Actual outcome | comment |
| 8 | I will test that the file dialog closes if the user accidentally tries to load a file | I will start the game and from the menu I will press the number 2 to load a game, the file dialog will open I will try to close the file dialog by click the close cross in the top right corner. | The file dialog will close | The file dialog did not cancel when the close cross was pressed or cancel | Failed |
| 9 | I will change the texture for a save game. | I will create a new game and then save it. I will then exit the game. I will then open the save file and change the texture location I will then reload the game. | The 1st textures will be different to the 2nd textures. | The textures were different. | Passed |
| 10 | I will test that potions in battle increase pokemon health | I will load a game save file where I have 1 type of each potion, I will start a battle with a NPC and then I will use a potion once for each of my turns. | Potions, super potions and hyper potions increase health by a factor 1.2, 1.5 and 2 respectively. | Potion did raise HP by factor of 1.2 (190 / 158 = 1.2), Super potion raised HP by factor of 1.5 (285 / 190 = 1.5) and hyper potion did raise HP by factor of 2 (556 + 14 / 285 = 2) | Passed  A bug was show, the players bag showed only potions in the players bag, where as in battle it showed it had 6 potions and a X defence – this is probably an error in loading the save file. |

## Testing evidence

Test evidence given in video format.

|  |  |
| --- | --- |
| Test | Description |
| 1 | **00:06 – 00:40** - It first shows that any invalid inputs, those which are not 1, 2 or 3, are ignored. It then shows that the menu options take the users to the right parts of the game. It first shows the user starting a new game by pressing 3. It then shows the user loading a save file after pressing 2 using a file explorer, it also shows the user trying to load an erroneous file, in this case a .png file, the file explorer closes then re opens not lading the .png file and allowing the user to select an appropriate save file. It finally shows the user exiting the program by pressing 1. |
| 2 | **01:20 – 01:56** – this test is a demonstration of the random maze generation and the assignment of random items and pokemon. The user from the main menu presses 3 to start a new game displaying the maze, the user opens the in-game menu displaying their pokemon and their items, this is then repeated 2 more times, with each maze, set of pokemon and items being different each time. |
| 3 | **01:57 – 02:35** – Shows the user moving around the maze using the WASD keys, it shows that the player cannot move outside the bounds of the maze as well as not being able to move through the walls. Due to the type of collision the players centre is what collides with walls and bounds, so it looks like the player is out of bounds and through walls.  **02:36 – 03:22** - It shows the player interacting with a sign first by walking on top of it then pressing T it then displays the sign GUI, the user then presses T again to exit.  The user then walks over to a chest and presses T on top of it to open the chest GUI the user uses the arrow keys to move the cursor, it the user tries to move the above/below the first/last item position respectively the arrow cursor remains in the same position. The user uses the Enter key to swap which inventory the selected item is in. However, if the user has less than 6 items the player can still move the cursor past the last item. This system is used in all GUIs and uses of navigation.  **03:24 – 03:44** - The user then opens the in-game menu by pressing E and then navigates the menu using the arrow keys and the Enter key to select an option. The first option selected is “Pokemon” which displays the user’s pokemon, the user uses the arrow keys to navigate and the Enter key to select an option. The first selected pokemon is blastoise, it then displays this pokemons current stats. The user presses E to exit the menu. The user then Presses R to remove pokemons.  **03:45 – 04:03** – the user then selects option ”Bag” to view the users items within their bag, the user then presses E to return to the in-game menu the user then selects option “Save” and saves the game in a .txt file, the user then using the in game menu selects option “Exit” to return to the main menu.  **04:06 – 04:30** – The user interacts with an NPC by pressing T while on top of it, this commences a battle. By pressing E the user opens the in-battle menu and press R to return to the maze. |
| 4 | **04:41 – 05:14** - The user opens the in-game menu by pressing E then selects option “Pokemon” using the Enter key, to show what pokemon the user currently has, which is currently less than 6. The user then walks around the grass until a random encounter occurs. The player uses 1 pokeball, it announces the player wins. The player then shows which pokemon it now has, the previous one in addition to the one it caught in the encounter. |
| 5 | **05:33 – 06:41** - The user enters a battle by pressing T on top of an NPC. In this scenario the user is trying to win, firstly it swaps their pokemon to have type advantage. The NPC uses its highest attacking move which does 24 damage which is in range of 18 – 33 using the damage calculation, where the modifiers are random which ranges from 0.85 to 1, STAB (Which it had, so was 1.5 as the move type was the same as the pokemon type) and critical chance which has a 5% chance of being 1.5 else 1 and effectiveness which ranges from 0.25 to 4. In this case the effectiveness was 1.  DamageCalc.png  Level = 20 power = 95 A = 84 D = 110 Modifier = 1.5 \* 1 \* (0.85 to 1) \* 1  The NPC used a hyper potion to increase its HP from 158 to 316, it then took damage, so HP was reduced to 250.  The NPC was at type disadvantage and only had one pokemon so couldn’t switch pokemon to gain type advantage.  The battle continues with the user using the most effective moves and the player wins.  **06:56 - 08:46** - In this battle scenario the player is trying to lose, so only has one pokemon which is normal type which is most neutral so most of the type the effectiveness modifier will be 1. The AI doesn’t always choose the most optimal moves or actions such as repeatedly swapping between 2 of the same pokemon. This is due to the random variable in choosing options in the AI. However, it does make a sensible decision to swap out its first pokemon to a pokemon with higher HP when the first pokemon is low on HP. The AI does win however it does display that the player wins – bug. |
| 6 | **09:35 – 10:08** - The user creates a new game, moves to a new location and then opens the in-game menu to display which pokemon the user has and which items they have. The user then proceeds to save the game and then exits to the main menu. The player then using the file explorer opens the same save file. The game then loads the user is in their same previous location with the same items, pokemon and map layout. |
| 7 | **00:02 – 00:11** – plays main menu music  **01:21 – 04:09** – plays in game idle music  **04:10 – 04:27** – plays battle music |
| 8 | **10:13 – 10:25 -** From the main menu the user presses to, the file dialog opens. The user then proceeds to close the file dialog 3 times by clicking the cross in the top right corner each time the file dialog closing then re-opening. The user then tries to click cancel to close the file dialog again with the same results. |
| 9 | **11:46 – 12:28 -** The user starts a new game, then saves this game, the user then exits to main menu. The user then opens the save file changing the tile location from “tile2” to “tile”. The user then proceeds to press 2 on the main menu to open the file dialog. The user then opens the saver file. The game loads and the new textures are loaded. |
| 10 | **10:26 – 11:32 -** The user navigates their way to an NPC, the player then shows their inventory showing the potions, the user then shows their first pokemon’s current HP - 158, interacts with the NPC by pressing T and then entering a battle. The user firsts uses a potion which increase the user’s pokemon’s HP from 158 to 190, then uses a super potion raises HP from 190 to 285, the player then uses a hyper potion but takes 14 points of damage, so HP rises from 285 to 556. |

# Evaluation

## Meeting objectives

*1. The game will have a start screen, which will display the menu which has 3 options “Start new game”, “Load game” and “Exit”, the start screen will also display the game title “Dungeon Master” and will have a background image of a cave system.*

This objective was fully met. However due to its implementation it makes it difficult to have more options as it uses a static image for the menu, so to change menu options I would have to change the static image. However, the background image of a cave system was changed in the final product due to end user feedback during development to make it look more appealing.

*a. You will be able to use the number keys 1, 2, 3 to select a menu option.*

This objective was fully met and was very easy to implement, as it just involved a select case for which keys were pressed. This could be easily expanded if more options were added as it just means adding one extra case per option.

1. *If the option “Exit” is chosen, then the program will exit.*

This objective was fully met and was easily implemented as it was one line of code Me.Exit().

*3. If “Start new game” is chosen, then*

*a. It will then generate the maze in a 2D array of Cells, using a recursive back tracking algorithm.*

*i. The room shape will be square with the length and it will fill the 2D array with new Cell.*

*ii. With the tiles in the 2D array it will apply the recursive back tracking algorithm, starting at Cell (0, 0)*

*iii. It will give a symbol value to each cell depending on which walls are true and false*

These objectives were fully met. The recursive backtracking was easy to implement as it already was prototyped multiple times, so was well tested to work. The cell 0, 0 was chosen as that is where the player will spawn so the player is already given a route to move. For assigning a symbol each cell was given a number based on the walls existence, converting it from a binary number to a denary number. In the order north east south west. True being a 1 and false 0.

*b. Then populate the maze with rooms and interactable tiles such as tall grass and chests.*

This objective was mostly met as the maze is populated with NPCs chests and signs, a new addition, with each one given a new random unoccupied coordinate in a 2d entity array. However tall grass was not implemented as an interactable tile/ entity instead tall grass is a specific tile within the maze, that being tiles with no walls.

*c. It will the read from a file a create a list of Pokémon*

*i. The Pokémon will have specific moves to them*

This objective was fully met, to store this data I used a csv file which was easy to write and read as each section is clearly defined visually. It was also easy to read from in the program as it just meant splitting a string by commas. The file is written so that the order of the data is the same as the order of the parameters in the pokemon constructor sub.

*d. The player will then be given 6 random Pokémon as well as 6 random items*

This objective was met simply as it just involved generating a random number and then returning an item/pokemon from a list of all pokemon/items.

*4. If “Load game” is chosen, then*

*a. A file explorer will open allowing the user the select the save file in the form of a text file.*

*b. The file will be then loaded in to the game and parsed to create a new game.*

This was fully implemented as all aspects of the game was saved and loaded. The saving of the file was easy as the file format was already predetermined. However loading from the file was a bit tricky as no two save files would be the same so it was difficult to implement a general solution.

*5. When the game begins.*

*a. The player can used the WASD keys to move around the map up,left,down,right*

*b. The map will be scrolling screen, with no momentum*

This was not implemented and was instead replaced with showing the whole map, due to end user feedback during development as it makes the whole game clearer. However this would be easy to implement as it would involve drawing a specific section of the map e.g. a 10 by 6 area of cells. This would mean moving around the area when the player moves and making sure it is in the maze bounds. It would be easy to implement as it would be very similar code to the player movement. This would allow me to make the tiles larger, thus adding to the clarity of the game. It also let me add the original torch overlay to add more ambiance to the game.

*c. They can press E to open their bag*

*d. And they can press E to display the menu*

This was mostly implemented, movement was just a select case for keys pressed and the menu system was changed as e opens the menu, arrow keys to navigate, E to return and enter to select an option. The control scheme for menu interaction was slightly changed due to end user feedback to make it easier to use as well as making it more intuitive.

*6. If the player walks through tall grass, there is a 0.25 chance of a random Pokémon encounter.*

*a. In this encounter they can battle the Pokémon, use an item e.g. pokeball switch or run.*

This was partially implemented as players can catch encountered pokemon however there isn’t a 0.25 chance of an encounter due to it be implemented by incrementing a timer and then calculating a random chance. To implement this I could have it so each time a player enters a new cell which is a tall grass cell it then does a random chance. I would need to store the player’s current position and previous position to check that the player has moved into a new piece of tall grass.

*7. The player can also press T if standing next to a chest and will see an interactable GUI where they can manage their inventory, chest inventory will be randomly generated.*

This has been fully implemented, the GUI drawing for it was hard to implement at first but then a came up with a solution to make a GUI class to draw all GUI which made it a lot easier. Assigning the chest inventory was easy as it was the same as the player, so I could use the same code.

*8. If a player is next to an NPC*

*a. they can press T and then the player will be presented with a random phrase then a battle will occur.*

*b. The players can’t walk through the NPC, if the player beats the NPC then it will passable allowing the player to progress.*

*c. The player has the same option as random encounter but can’t run or catch Pokémon*

This was mostly implemented, expect for the player being not able to move through NPCs, which was removed due to end user feedback during development as they found it annoying and limiting. However, to implement this would be straight forward I could when populating the maze if there is an NPC set that cell’s walls to all be true, once the NPCs is defeated I could remove the NPC entity from the entity map and set the cell’s walls to false. Or I could have a Boolean, default false, in the NPC class that stores if it has been defeated or not, when a NPC is defeated in battle then it will be set to true. Then in the player collision in the player update I could check if the cell the player is trying to walk in to is occupied by an undefeated NPC if so the player won’t be allowed to move into that cell.

*9. In a battle the player will battle the AI taking it in turns to make a move, this will happen until either all the players or NPCs Pokémon have fainted then it will redisplay the map.*

*a. The AI will generate options and score them, higher the score the better the choice, it will then select a random option dependent on the option scores.*

This was fully met and was quite straight forward to implement. For moves scores were based on damage done, for items was the multiplier effect and for switching pokemons scores were based on type effectiveness. An option was chosen by generating a random number between 0 and 1 then see where it lay in the range of options e.g. The first item had 0 to 0.2 and the second item had 0.2 to 1

Extension objectives.

Most extension objectives were unmet due to time limitation, however most could be implemented given time.

1. The player will be able to find a key in a random chest

a. The key will be able to open the door on the map perimeter

b. The player will be able to open the door by pressing space if they have the key

c. If the player walks through the door they will be taken to a new map

d. In the final map the player will be able to walk to the final boss and press T to battle the final boss

These objectives were not met. However it would be quite simple to implement, it would involve another if statement in the player update routine to check if the player is in specific spot e.g. The middle of the maze and if they press a key e.g. Space and have the 4 keys then currentmap in game would load a map from a file which would be a place for a final battle. When populating a maze I could select 4 random chests and place keys in their inventories. Using the item constructor with hardcoded attributes. The only difficult bit would be improving the AI more so that it presents a greater challenge to the player as the current AI currently has difficulties presenting a challenge to the player. To improve the AI could take out the random chances and the AI only uses the most optimal moves/ items.

2. Exclusive region Pokémon

This objective was not met. This would a be a bit tricky to implement as it would involve adding a new class called tall grass, which would inherit from player/ or an entity class and would have a list of integers would store a range of numbers which correspond to a pokemon ID number. These numbers would be assigned randomly or specifically when the rooms are generated. And then when a random encounter occurs instead of selecting a random pokemon it would select a pokemon in the tall grass’ list of possible pokemon and use that.

3. Pokémon can level up and learn new moves.

This objective was not met. This could be implemented by giving the pokemon class a list of integers to store the ids of the moves a pokemon could learn. And then when a specific trigger condition is met, e.g. The player wins 5 battles, a move runs out a PP or the pokemon has gained a specific amount of XP a new move randomly chosen or in a specific order could replace a chosen move/ a random move or a move with 0 PP.

4. Background music dependent on scenario

This objective was fully met with music for the main menu, maze and in battle. Also implemented were sound effects for menu option selection and receiving items form a chest.

5. Customization of character sprite and cave

This objective was partially met. It is possible to change the map textures by changing the location in the save file. Customization of the player sprite would be easy to implement I could add a string variable in the player class which stores the sprite location, and then that texture could be used when drawing the player. This could be changed in the save file similar to the map textures or through an in game settings menu similar to the loading of file. However having pre-set maze shapes would be trickier as it would involve have a scale able size algorithm to decide which cells are blocked to make different shapes, which would be trickier to figure out how to do.

## Overview of the solution

Overall, I am very happy with my final solution as it looked and played pretty much exactly as I wanted. With the movement around the map I am pretty pleased with it however I felt it could be improved with animation and better collision.

I was most disappointed in the AI part of my game as it didn’t work as well as I hoped it would have. When playing against it, it has no clear strategy but most of the time does choose a sensible option but not most of the time the best option, which would be due to the randomness aspect of my AI.

I believe this is due to the lack of in-depth planning when it came to the AI. I had a general idea of how I wanted it to work, but never fully broke it down in to each individual part testing probabilities on deciding exactly beforehand how it would make decisions.

The AI was harder than I thought to implement as each part of it was quite interlinked, even though I was trying to not make it this way to make it more modular to add parts to it in the future. It can beat a very bad player would it would be better if it were more competent, this could be done by changing percentage values or taking that out completely or a whole different AI solution.

If I were to re do this project I would consider using Drac5290’s approach as there is a clear defined structure to the AI most likely allowing for a clear AI strategy. However, one benefit of my original design is that it wold be easier to expand than Drac5290’s approach.

The most pleasing part of my game was the generation of the maze and the navigation and interaction with the maze. The depth first search algorithm was very pleasing as it gave the exact maze look I wanted without having to adapt the algorithm too much.

The look of the maze, GUIs and movement was very pleasing as it looks very professional. And was very easy to implement due to the large amount of prototyping, so it was a lot of drag and drop into the final project for implementation.

However, I wasn’t too pleased with the collision it did work as intended and was simple to create and implement however the look of it made the movement of the player unclear. So, if I were to re do the project I would have change the collision so that it looked better as well as adding animation to make movement clearer.

When it came to coding at first it was difficult to get to grips with drawing the images but after a while it got easier as it was quite repetitive. I could improve it by have a function to draw rectangles to save code. As well as deciding what should be a public shared variable, most of the time I would implement them into a specific class then realise that it would need to be in the game class.

Object orientation of my programming was very useful as it forced me to thoroughly design the program beforehand making my project have less design flaws. Also, through inheritance and polymorphism it allowed me to write less code which saved me time allowing me to complete more objectives. So overall OOP was a very suitable choice for my program.

The XNA framework was very suitable for my program as it uses a language that i am already very familiar with, so I could very easily implement the basics, and the new code used was very straight forward. XNA allowed me to graphically display my solution.

## Possible improvements

I could improve my AI, this could be by it always choosing the most optimal option instead of the randomness involved. Or I could try to implement a different type of AI such as a large decision tree or with machine learning. This wold present a greater challenge to the player, and more faithfully following my initial concept.

Another improvement is adding more content such as variance in NPC types by creating different NPC classes which inherit the parent NPC class. I could add more pokemons, moves, effects and items by adding more of them to the files. This would allow the user to experience more things in game allowing them to play the game for longer.

I could have settings to rebind keys, look, texture scale, and maze shape/size. This could be implemented by having a settings file which is loaded when loading a new game and stores the changes to settings. Also, there needs to be an in-game settings menu, so the user can change the settings.

I could also improve the look of the game by adding animation to walking and battles. This would make the game UI a lot clearer as it gives more indications to the user what is happening. To do this it would involve multiple new texture for movement and displaying them in a specific sequence when drawing.

Another thing I could improve is the file format for my saving of the game, currently I save everything in one file, and when loading it is very longwinded to parse and isn’t very easily expandable. It also presented a few errors in loading. To improve it I would have a separate file for each part of saving of the maze, entity map and the player save data. This would make loading a lot easier as each part isn’t reliant on one another.

Another improvement I could make is having less code for drawing GUIs. Currently I have multiple ways a of drawing GUIs, to improve this I could have one general sub which expects parameters for the description of the GUI and have that draw them all for me. This would save time coding as well as making the game more expandable for more GUIs. Or I could have so the GUIs are added from an external file. Allowing for external modification if the software.

## End user feedback

**What do you like about the game?**

**Sze Sze**

“I really like how the game looks, walking around the maze and the textures used make it look like an actual pokemon game. The background music and sounds effects are a cool addition as it gives the game just that extra bit of polish to make it a well-rounded game. I also like how easy it is to save the game, so I can carry on my progress at a later date.”

**James**

“The battling looks really dope man! it actually looks the real game, at first I thought it was an emulator. The controls for were very easy to pick up and felt very natural, after being told how to play the game”

I am very happy with this feedback as it shows that the interface design has worked well to emulate pokemon games. This shows that I have faithfully replicated my initial concept of trying to emulate a pokemon game. However, I should reconsider by initial idea of not having instructions, as I thought it would be more fun trying to find out how to play, however it seems its more frustrating.

**What, if any, changes would you make to the UI?**

**Sze Sze**

“I wouldn’t make any changes to the UI it seems very clear and easy to understand. However, the signs could be formatted a bit better. Some words are split in half”

**James**

“The menus and stuff are pretty good, but the game is just too small I can’t see where I am or where I am going even when I’m wearing my glasses. You should have made it a lot larger otherwise it’s just unplayable.”

I don’t disagree with James comment as it can sometimes be hard to see where you are, however I believe that the small tile sizes are an aseptically pleasing look, however to make it clearer I could use a more distinct player sprite as well as implementing animation to denote direction. For the signs formatting if I had more time I could have focused on improving the algorithm which converts a string to a list of strings so that whole words stay together as one. This wasn’t a priority for me as it still functioned as intended.

**Do you have any suggestions for additional features?**

**Sze Sze**

“I think you should have added more pokemon its annoying getting multiple of the same pokemon I want more variety, so I can improve my battle strategies. Talking about battling I think you should have added animations to the battle the text log is fine and all, but the animations would have made it a lot clear to what exactly is going on In the battle. It would also make them more exciting to play. And finally, I think you should have a final boss the end point would make you have a reason to return back to the game”

**James**

“The obvious thing you could add is just more content like pokemon, items and moves. However, I do acknowledge this would take time and be tedious to do. Apart from that just more customisation of the game in general there should a settings menu to control sound and textures. And also, when creating the game have different ways to generate the maze such as the shape.”

Both of their first two points would be very easy to implement due to my file structure as it would just involve writing more data to the pokemon and move files.

Adding a finally boss would be harder to implement as it would mean coming up with a better AI system than the NPCs to give a player a challenge as a final boss. I would also have to come up with a system on how the activate to the final boss, would it be a matter of collecting certain items or traveling to a specific place in the map. So, this may involve making another map, which would mean having a way to swap the current maps.

I have sort of already added look customisation of the map however it has not been added to be able to change it in game, rather editing the save file. So, I would need to create a in game settings menu.

Different shape maps would be interesting as you can only have 2d arrays in the shape of squares, however I could come up with a way of selecting certain perimeter cells to be set as visited and all walls true and they could determine the shape of the maze.

**What did you think of my code from a coders POV?**

**James**

“The memory management seems to be done quietly well, with using external resources, you can open them when needed and closes them when they are not in use which is quite efficient. Your code is very well commented so I could pretty much understand everything that was going on. I also like the use of you importing windows forms for you error messages. Which probably save you time coding as you didn’t have to code the GUI. However, the way drew everything seems a bit of a mess, especially your messageBox class. I feel like you should have just had one draw sub with parameters on how to daw a specific message box. There seems to be already partially implemented with Booleans for displaying text/ pointers. I also would have change the inheritance structure of you Player, NPC, Chest and Sign class. The way you have done it is the latter three inherit from player. However, have very little similar functionality of player. I would have had a more basic Parent class called entity or something, where NPC, Chest and sign inherit from, and have Player inherit from NPC as they are very closely linked but Player has more functionality. This way in your entity Map in the Map class you could save memory as less data will need to be stored.”

I am glad James picked up on my use on comments, it shows that if I were to give if the code he could get the gist of it add make any changes and additions he wanted. I do defiantly agree with his criticism of the drawing. This is the first time I’ve tried to graphically display any program with textures, so it was a process trying to get used to it, I so I defiantly agree I probably didn’t implement it in the most optimal way. I also agree with his class restructure, the way I didn’t wasn’t wrong as it did work, but James’ way is probably a better way to do it making my code more memory efficient. And it probably wouldn’t take too long to do.