**Computer Science NEA**

**Retro Game Compilation**

**Candidate Name: Hassan Mirza**

**Candidate Number:**

**Centre Name: St Joseph's College**

**Centre Number: 30555**

**Table of Content:**

**Analysis**…………………………………………………………………...……**pg3**

**Problem Identification**……………………………………………….**pg4**

**Background Research**………………………………………………...**pg6**

**User Identification**……………………………………………………**pg12**

**Objectives**………………………………………………….................….**pg13**

**Modelling**………………………………………………………………….**pg20**

**Acceptable Limitations**………………………………………...…..**pg22**

**Design**…………………………………………………………………….…**pg23**

**Overview**……………………………………………………………….….**pg23**

**Planned Design**…………………………………………………...……**pg31**

**The System**……………………………………………………………….**pg35**

**Modular Design**…………………………………………….................**pg36**

**Data Dictionary**………………………………………………...………**pg80**

**Test Strategy**…………………………………………………………….**pg89**

**Technical Solution**……………………………………………………...**pg90**

**Testing**……………………………………………………………………**pg103**

**Test Table (Black Box)**……………………………………………..**pg103**

**White Box Testing**…………………………………………………….**pg112**

**Evaluation**………………………………………………………………...**pg115**

**Conclusion**……...………………………………………………………**pg117**

**Appendix**…………………………………………..………………………**pg118**

**Analysis**

In my project I plan to create a compilation of 4 popular retro game remakes from different periods alongside a game of my own making which follows the same style as the other games. The games I plan to remake are;

* Pac-Man (Originally Released on the PC)
* Space Invader (Released in 1978 on Arcade Machines)
* Pong (Originally Released on the Atari 2600)
* Flappy Bird (Released May 2013 on Mobile OS Android and IOS)
* One of My own Idea of a Retro Type game

I plan to change things up and include new features to enhance replay value and to add my own twist to these games. These features include: - In Pong I plan to include different modes for instance easy mode which will be a remake of the Pong game and a hard mode where the ball will have the ability to change direction, speed etc. In Pac-Man I plan to include different maps where the player can choose which one they want to play to increase the replay value of the game as well as provide user choice this also makes it so the four different ghost AI’s fit the new maps; In Space Invader I plan to add a different mode where there are a set amount of enemy’s whilst also adding a limit to the bullets the player can fire, both these options will be customizable by the player and will allow the player to control the difficulty they want to play at; In Flappy Bird I am looking to add random events which will affect the player either helping them or working against them, my current ideas include the player becoming invisible or invincible and either speeding up the pace of the game or slowing it down. These are only a few ideas which I currently have and will be used to add replay value to the game and to make it different and fun for the player by keeping them focused in case there is a random event; The game (My own Custom game) I am planning to make will include their being a controllable player which can move around and interact with their environment whilst an enemy AI chases them whilst also increasing in number as the player reaches different milestones in their score. All of my games will include a scoring system which will save the usernames and scores of the players playing the game. All of these score will be compiled into a textfile and sorted with a sorting algorithm called ‘Merge Sort’ whilst also allowing the users to search for their scores through the use of Binary Search. I plan to have the games feature a retro 80’s & 90’s arcade style and feel, so older players can experience a feeling of déjà vu and new players can see what the retro genre is composed off. Its purpose is to allow older players to relive classic games which they might have played when they were younger, but also to show young gamers how far gaming has come and advanced, alongside showing them where gaming originated and revive a style of gaming which is slowly becoming obsolete and being replaced with modern day “classics”. I have chosen to create this project due to its complexity and challenge, alongside my Background Research and Problem Identification.

**Problem Identification**

I have chosen to do a retro arcade game compilation, as there are not that many available to gamers around the world and many of which have been forgotten about in the flow of time, as there are very few retro arcade games on the computing market. Currently, there are many games on the computing market, however the majority of it is overrun by modern games for instance: - FPS (First-Person Shooters); - Open World Games etcetera. The retro arcade games are left in the dust and are only developed by indie game developers. Many popular games are of the FPS and Action genre alongside the Open World games, a few games which have been an international hit are “Battlefield 1”, “Grand Theft Auto”. To play retro and arcade genre games many people have been pushed to play these games on mobiles, which isn’t a big platform of gaming compared to console and pc gaming which make up the majority of the gaming industry, this can be seen in figure 1 below.

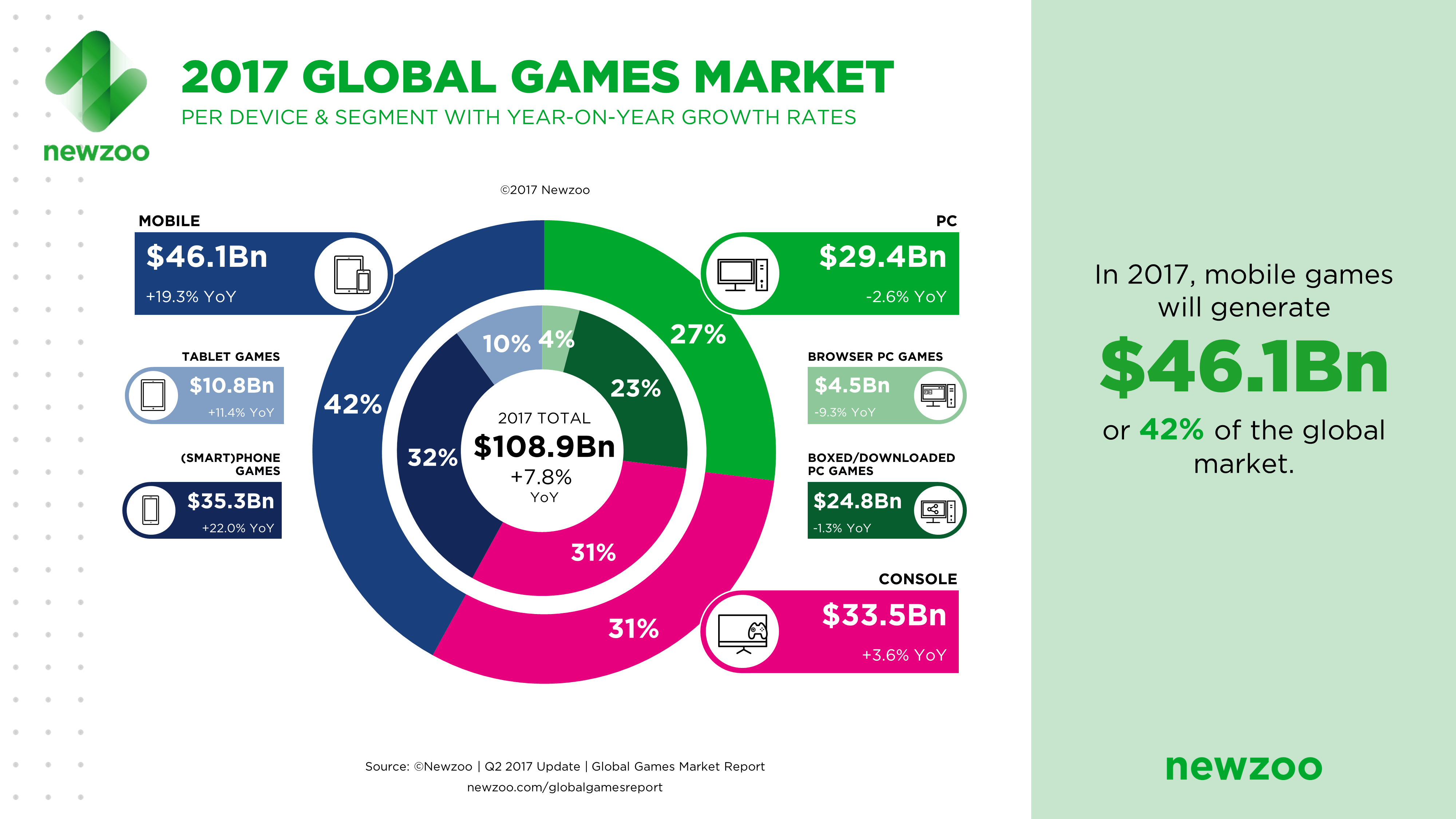
**

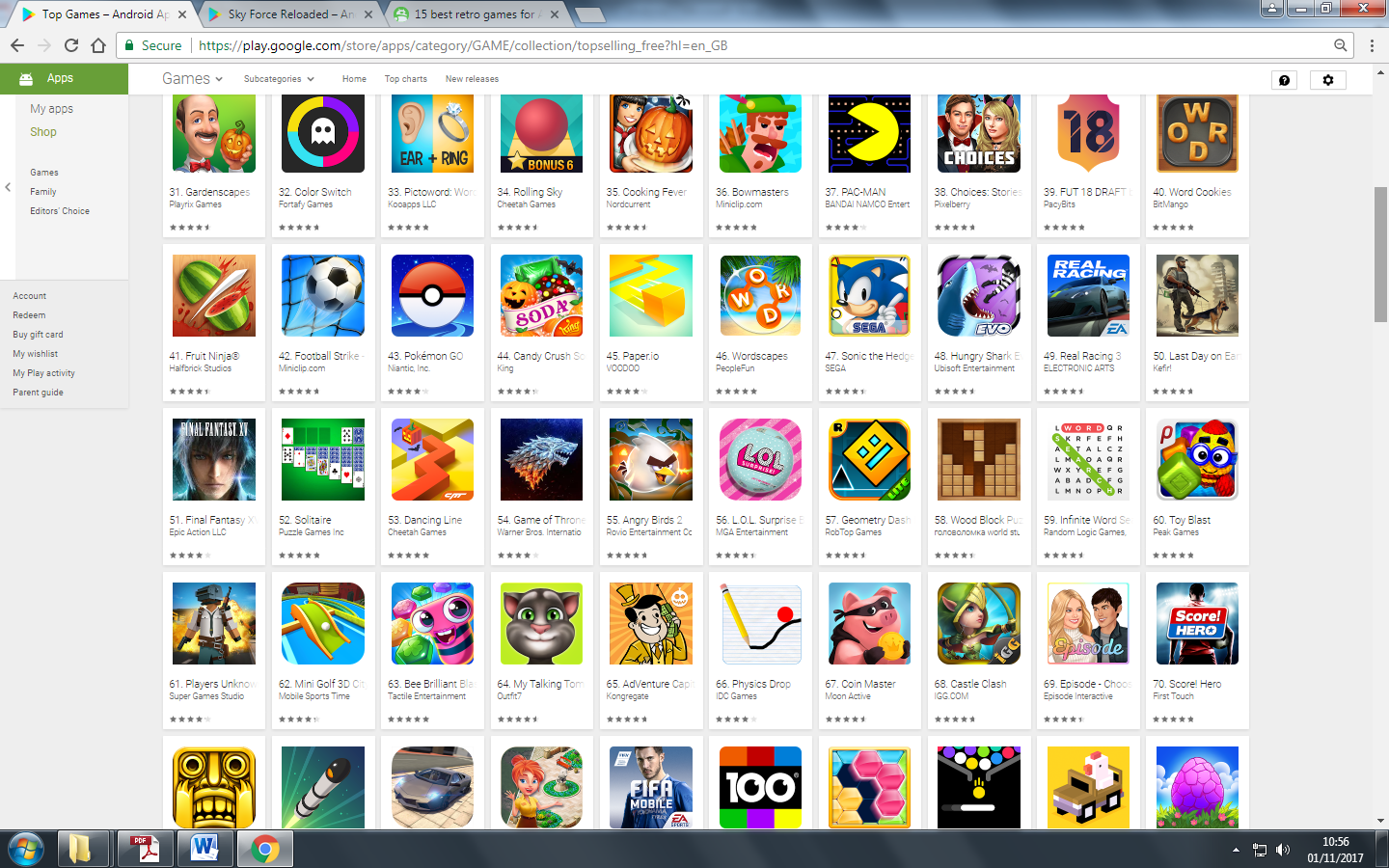
Figure 1: Showing a breakdown of revenue from gaming

<https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/>

In the current day and age the majority of the market consists of consoles and pc which make up 58% of the global games market. They are very popular due to them not requiring to be changed and updated (until a very long time has passed) to stay with the development of new games, unlike the PC which can be made more powerful but costs more to buy and produce, also if your specifications are not up to scratch you won’t be able to keep up with the new releases. As a consequence of this console gaming is preferred more, compared to PC due to console gaming relative lack of expense once you have bought the console. I believe that the old retro arcade style of gaming is better as not everybody has the money or time to buy or build these powerful machines to play the latest for example a very popular game called the Witcher 3: Wild Hunt has very high recommended requirements if not bought on a console which can also be quite expensive to some people. The recommended requirements of the Witcher 3: Wild Hunt are:

* CPU: Intel CPU Core i7 3770 3.4 GHz / AMD CPU AMD FX-8350 4 GHz
* CPU SPEED: Info
* RAM: 8 GB
* OS: 64-bit Windows 7 or 64-bit Windows 8 (8.1)
* VIDEO CARD: Nvidia GPU GeForce GTX 770 / AMD GPU Radeon R9 290
* FREE DISK SPACE: 40 GB

This hardware is quite expensive and not everybody can afford it, or buy the parts and build the PC as they don’t have enough time.

Mobile gaming takes place on any portable devices (unless it’s a handheld console) such as mobile phones and tablets; it can be said and considered that it’s gaming on the go. Many of the top games on the Google Play Store are retro arcade game reskins, as can be seen in figure 2. A very popular game on the Play Store is ‘Crossy Road’ which was released all the way back in November 2014 and is still among the top 100 games on the Play Store. However ‘Crossy Road’ is a reskin of the popular game ‘Frogger’ which was released back in the 1980’s; this further backs my point that retro games are being forgotten slowly but can still be revived due to them being quite popular with the public, therefore I plan to remake a compilation of popular retro arcade games in an attempt to revive the genre and produce entertainment for the public who evidently enjoy remakes and reskins of retro games.

<https://play.google.com/store/apps/category/GAME/collection/topselling_free?hl=en_GB>

Figure 2: Top Apps on the Google Play Store, showing ‘Pacman’ (37th Place) and ‘Crossy Road’ (79th Place) which was released back in 2014 and is still among the top 100 games

I believe retro and arcade games like ‘Crossy Road’ are popular because of the light-hearted nature they possess and the simple theme of the game which is understandable by many. Remakes of retro and arcade games also attract modern gamers as they can relate to the social concept of the game as they are based on the modern themes for instance ‘Crossy Road’ takes the joke of “Why Did the Chicken Cross the Road?” and produces a game of a chicken crossing the road. These games are also popular as they are easy to play and anyone can come and pick it up and play without any difficulty unlike many modern games where you have to learn how to play it. As a consequence of this I believe the player gets immersed into the game and many can get addicted due to it being easy to play and the challenge of the game. The addictive nature of these games is also an aspect of their success as you don’t always lose ‘you almost win’ and you *have* to take risks to beat your high score.

**Background Research**

In my background research I will be going over the history of gaming (arcade) and the development of the gaming industry alongside the development of different gaming systems.

It is a belief of many people that arcade gaming as a whole began in the 1970’s/80’s, however it actually began all the way back in the 1930’s when the first coin-operated pinball machines emerged. However some do not consider these because of them not matching the first pure arcade games and consoles in entertainment and original/classical releases. These early amusement machines differed from there, later on, modern electric countertypes as they were made of wood. As a result the first pure arcade games are considered to have surfaced in the 1970’s as this was when the first gaming console released called the Magnavox Odyssey which was released in the early 1970’s in 1972.

(Source:<https://www.google.co.uk/search?q=what+was+the+first+video+game+console&rlz=1C1GGRV_enGB769GB769&oq=what+was+the+first+video&aqs=chrome.4.69i57j0l5.4735j0j4&sourceid=chrome&ie=UTF-8>)

The first Atari console also released in the 1970’s, the Atari 2600 also known as the Atari VCS was released in 1977 this console was very popular at its time. Also in 1977, most pinball machines in production switched to using a solid state electronics both for operation uses and scoring. As a consequence of this many people believe that arcade gaming emerged and actually began in the 1970’s/80’s even though the history of arcade gaming goes all the way back to the 1920’s/30’s.

The first arcade game (known) was developed and produced by Atari which was formed by Nolan Bushnell and Ted Dabney. Atari essentially created the coin operated gaming industry with their game of Pong, the first successful electronic ping-pong game.

However the Golden Age of arcade video games began with the release of Tailto’s Space Invaders, in 1978, this proved to be the first blockbuster arcade video game and it was its success spawned the Golden Age of arcade video games. By 1981 the arcade video game industry was worth $8 Billion. The gaming industry is expected to reach $108.9 Billion in 2017.

(Source:https://www.google.co.uk/search?q=what+was+the+first+video+game+console&rlz=1C1GGRV\_enGB769GB769&oq=what+was+the+first+video&aqs=chrome.4.69i57j0l5.4735j0j4&sourceid=chrome&ie=UTF-8)

By the late 1980’s the arcade video game craze began to fade away due to advancement in home video game consoles technology.

The period known as the Renaissance of arcade video games began in the early 1990’s, due to the release of Capcom’s Street Fighter II, which made competitive fighting games popular and revived the arcade industry to the level not seen since the days of the release of Pac-Man. This set of the Renaissance of arcade video games. Around the mid 1990’s the success of arcade video games began to decline again, due to the fifth-generation home consoles like the PlayStation and Nintendo 64 etcetera.. This was because fifth generation home consoles were offering true 3D graphics. The decline of arcade video games was further accelerated as personal computers followed the fifth-generation home consoles by having 3D accelerator cards.

The 2nd generation consoles (specifically the popular ones like the Atari 2600, the Magnavox Odyssey 2 and the Atari 5200) differed from the 3rd generation (specifically the popular Nintendo Entertainment system and the Sega Master System) consoles a lot. One difference is that the 3rd generation preferred to use a D-pad Controller opposed to the joystick used by many 2nd generation console. 3rd gen consoles also had improved graphics alongside being able to have more sprites on the screen at once. These differences clearly distinguished the 3rd generation consoles to the old 2nd generation consoles. The 3rd generation of consoles ended in 2003 as the 8-bit console became obsolete compared to the new 16-bit console.

2000’s/2010’s arcade video games settled into the niche market by providing controllers for specific experiences, as a result the ‘Arcade’ as a place is a more socially-oriented hangout, which possesses games with specific performance rather than content, for instance rhythm games such as ‘Dance Dance Revolution’.

On an international scale the revenue from arcade games increased from $1.8 Billion in 1998 to $3.2 Billion in 2002 rivalling the PC game sales which also was $3.2 Billion in 2002. The majority of this increase has come from the eastern side of the world in countries like China and Japan where arcade games are still vastly popular with big AAA game producing companies still making arcade games for instance the company ‘Konami’. The Western side of the world has only seen a slight increase. Arcade games are thriving in China where there are arcades spread throughout the entire country. The US market has also seen a form of resurgence with the number of video game arcades increasing from 2500 to 3500 in 2008, though this is significantly less than the 10,000 arcades in the early 1980’s. As of 2009 it is recorded that a successful arcade game sells about 4000 to 6000 copies worldwide.

Due to the reduced popularity of arcade video games in the West, I plan to produce a compilation of very popular games, however changed so it fits into the modern era of gaming and to produce more of a challenge as I believe the average gamer has also become more skilled. I also want to make a remake of some of the most popular arcade games is that when I was very young I enjoyed playing arcade games in arcades alongside playing on my 2nd and 3rd generation consoles which had remakes and adapted version of classic video games. As a result of this I have chosen my project to help new gamers and young gamers experience what gaming was and how it reached the point it is now by remaking classical games in a modern way so that they fit into the present day and age by having new features.

To conclude, the reason I want to make remakes these classic arcade retro games is so younger gamers can see where games originated and what games were hits “back in the day” and formed the gaming industry as I believe that many people have forgot about this popular genre and are not reaping the benefits of this popular genre when the game is ‘done right’. I believe that I can solve this problem by creating a compilation of remakes of some of these popular games which as stated before are going to be:

* Pac-Man
* Space Invader
* Flappy Bird
* Pong
* Game of my making which is based on the themes of retro games

These games will have a retro arcade style and feel to them, as I postulate that they will show many upcoming and young gamers the origins of the gaming industry and the feel and simple-hearted fun of these games. These games will also be recognisable for older gamers who most presumably played some of these games, thus they might allow older gamers to relive some of their memories from the past and see the development of the gaming industry and technology as a whole. Due to genre of the game being retro and arcade these games will not be resource heavy, hence being able to provide entertainment for many people as many do not possess computers with an abundance of processing power, however the games will still possess the same fast pace, theme and nature of the classical retro arcade games. The project I plan to make is going to be a compilation of the most popular classic retro style games:

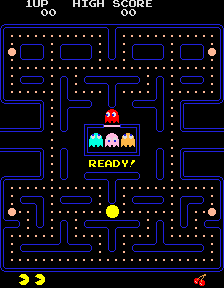
* Pac-Man
* Space Invader
* Pong
* Flappy Bird
* A game of my own design

These games will be enhanced to increase playability by adding new features which are

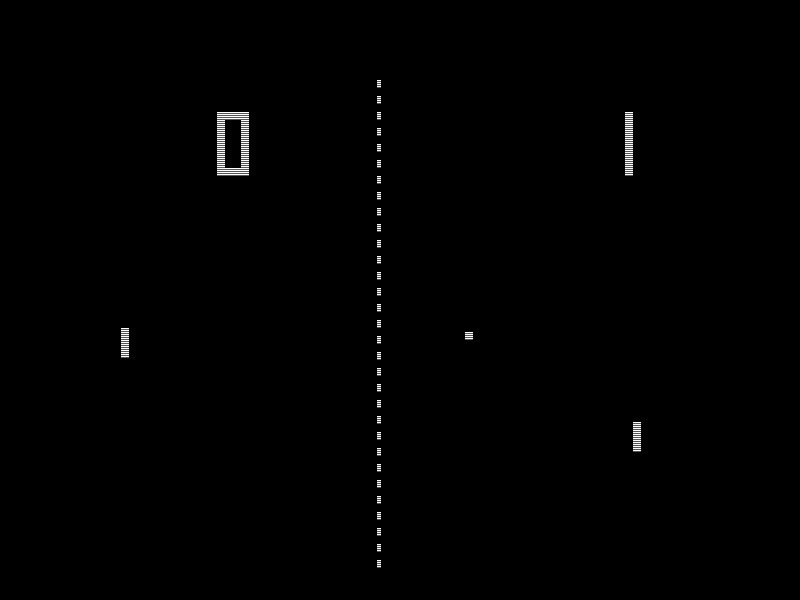
* In Pong I plan to include different modes (for example an easy mode which is just a remake of the original pong and a hard mode where the ball has the ability to change paths near the player which will attempt to confuse the player).
* In Pac-Man I plan to include different maps where the player can choose which one they want to play to increase player personalisation and user choice, this will also increase the replay value of the game making players want to play and complete each map to achieve a 100% completion.
* In Space Invader I plan to add a form of customisation where the player can choose to change the game so essentially they can play the game like they want to play it
* In Flappy Bird I plan to include Random Events which will stir the game up and keep the player alert at all times, one example of a random event is that the player becomes either invincible or invisible

(Source:

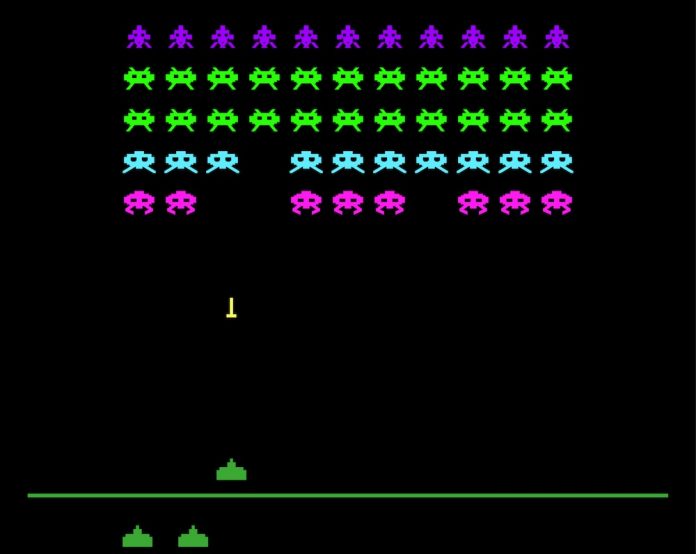
1. <https://en.wikipedia.org/wiki/Arcade_game>
2. <https://en.wikipedia.org/wiki/Video_game>
3. <https://en.wikipedia.org/wiki/Video_game_console>
4. <https://en.wikipedia.org/wiki/Second_generation_of_video_game_consoles> )



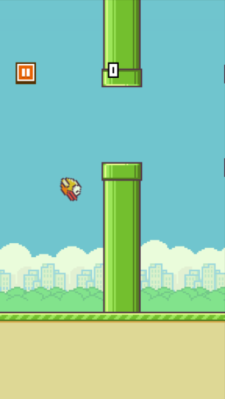
(Picture shows the game ‘Pac-Man’: <https://upload.wikimedia.org/wikipedia/en/5/59/Pac-man.png>)



(Picture shows the game ‘Pong’: <https://upload.wikimedia.org/wikipedia/commons/f/f8/Pong.png>

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj-_Ny9w-veAhUpgHMKHd2lAs0QjRx6BAgBEAU&url=https://www.casino-review.co/scientific-games-space-invaders-slot/&psig=AOvVaw1EOasLutFEiRUceHZYk_0o&ust=1543097278849132)

(Picture shows the game ‘Space Invader’: <https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj20d35w-veAhUozYUKHaV2B5oQjRx6BAgBEAU&url=https%3A%2F%2Fwww.casino-review.co%2Fscientific-games-space-invaders-slot%2F&psig=AOvVaw1EOasLutFEiRUceHZYk_0o&ust=1543097278849132>

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi3j-zaw-veAhWK4IUKHRefA7oQjRx6BAgBEAU&url=http://time.com/4034/no-you-dont-have-to-play-flappy-bird/&psig=AOvVaw2r9CRX9n4HngH-vFhdlD7t&ust=1543097324895035)

(Picture shows the game ‘Flappy Bird’: <https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwi3j-zaw-veAhWK4IUKHRefA7oQjRx6BAgBEAU&url=http%3A%2F%2Ftime.com%2F4034%2Fno-you-dont-have-to-play-flappy-bird%2F&psig=AOvVaw2r9CRX9n4HngH-vFhdlD7t&ust=1543097324895035>

In my research I have learned that it is possible to make all of my games in VB.NET and C#. An A.I would increase the games replay value encouraging the player to play it and try to outsmart it every time. I am going to attempt to form compilation of all the games in VB.NET and then use the .NET Framework to compile the games into a menu/interface using C# via the .NET Framework. This would allow the player to choose the available games and choose which game they would like to play. Through this menu (in C#) the player would be able to access the high score screen where they can view or search for their score through the Binary Search which will be implemented, the data will be sorted with Merge Sort, in a text file in each game, the individual games will also allow the players to access their scores and see their position. The scores from each game will be summed and stored in another text file where Merge Sort will be ran, this will then allow the user to use Binary Search as Binary Search requires the data to be sorted. The summed data will be used to rank and select the top 5 players. This would also add to the complexity of the program through the use of group A as it will include Merge Sort and Binary Search as well as user generated complex algorithms like the pathfinding for the Ghost AI in Pacman which I have seen is doable using A\* pathfinding and Dijkstra shortest path algorithm however it isn’t necessary, as you can do it without using the named method. Also all the Ghosts have their own personal algorithms which they follow.

My first question was is this project fun for people to play therefore I composed a survey where I asked 70 people in different age groups about genre which they played the most if they do play games. I also asked questions on the origins of gaming, my results can be seen below:

Table showing the amount of people who like each genre labelled in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Arcade/Adventure | Racing | FPS (First Person Shooters) | Arcade |
| 23 | 10 | 30 | 7 |

As it can be seen in my survey, majority of people are moving towards and generally prefer plays FPS games and Arcade/Adventure games compared to Arcade games, this just shows that my initial belief and research of the arcade genre going downhill is true and the genre is ‘slowly dying out’ as less people prefer to play it compared to more modern genre’s like FPS.

Questionnaire asking 70 people who game some basic questions on the history of the gaming industry:

|  |  |  |
| --- | --- | --- |
| Question | Answer | Num of People Correct to Wrong R/W |
| Which game is the most popular game/franchise of all time? | Super Mario Bro’s (Arcade Game (stared as Donkey Kong which is also an arcade game)) | 33/70 |
| Which game started of the Mario Franchise? | Donkey Kong/Super Mario Bro’s | 47/70 |
| Which game started of the gaming industry? | Pong on the Atari | 52/70 |
| What was the best time for the gaming industry? | Golden Age | 11/70 |
| Name an example of a third generation console? | NES (Nintendo Entertainment System), Sega Mark III | 25/70 |
| Name any 3 Retro Games? | PacMan/Pong/Space Invaders/River Raid… | 67/70 |

Sources:

<http://uk.businessinsider.com/top-50-video-games-all-time-ranked-2016-12/#1-super-mario-bros-50>

<https://en.wikipedia.org/wiki/Video_game_industry#1940s.E2.80.931960s>

<https://en.wikipedia.org/wiki/Mario>

As it can be seen in my results both of my hypotheses are confirmed as they show that the arcade genre is a dying platform (but with recent games it has a chance of being revived for people to enjoy). The questionnaire also shows that people who game do not know much about the origins of the gaming industry and genre and where it originated and how it was made into the successful industry it is now.

**User Identifications**

The primary targets for my project are teenagers as they are in the age group which play the most video games (13-19 year olds). Furthermore teenage brains learn new facts and information quickly compared to other age groups. Furthermore as shown by my background research it seems most teenagers do not know about the origins of gaming so this experience is perfect for them. However the game has to be easy to play so, anyone (at any age) can play the game even old gamers who might be able to relive old memories and hopefully experience some déjà vu playing this compilation. In addition to that many gamers, young and old, simply cannot afford the expensive new consoles and powerful gaming computers. This means that most people’s computers will have low processing power compared to the gaming PC’s/laptops and consoles, thus my program and the arcade genre is perfect for them as they are made to require a low amount of processing power and as a result it/they can be ran on many PC’s including budget computers. This is also a user need of my game; that my project is playable on any PC and should not require much processing power. This also makes the genre and my project laptop friendly as majority of laptops have basic specifications (i.e. low processing power) due to their integrated components limiting their performance. Finally my game must be entertaining/fun and allow the user to experience the ‘Golden Age’ of arcade/retro gaming and not its ‘Decline’.

In summary my user needs/requirements are:

1. The game has to be approachable by many different age groups, but the primary target of the game are teenagers due to the reason of the compilation being made.
2. The game should be a good example of the ‘Golden Age’ of retro and arcade gaming so it should allow users to learn and experience the history of the gaming industry.
3. The project should be easy to play and should be approachable by many people even older people.
4. The game should have a low specification requirement, so majority of PC’s and Laptops can run it.
5. The game must be fun and entertaining to the user.

These are my user specifications/need/requirements

**Objectives**

1. Space Invader
   1. Player
      1. The player will control a little ship, which will be placed at the bottom of the map; the ship will have the ability to move left and right as well as be able to fire bullets
      2. There should be a limit in the normal mode which will only allow the player to fire one bullet at a time
      3. The player should have some sort of cover from the enemy’s which will also be firing down, however the cover should only last for a certain amount of hits. This should not be available to the player in the custom game section to retain some difficulty for the player
   2. Enemy’s
      1. There should be three rows of enemy’s where one will fire randomly after a set time interval
      2. The Enemy’s should also slowly progress downwards whilst strafing left and right. For instance, in the original Space Invaders the Enemy’s would continue to move right/left until they reached the boundary, when they reached the boundary they would move down
      3. There should be three different types of Enemy’s where each row of enemy provides a different amount of points to the score
      4. The Enemy’s should also be arranged in a hierarchal manner
   3. Player Score
      1. The Players score should be saved in a text file with the users name and score.
   4. Menu
      1. The menu will have buttons, {Start Game }, {Custom Game} and {Exit} respectively
      2. The {Start Game} button will start the normal Space Invaders Game
      3. The {Custom Game} will allow the Player to Change the Settings of the game so they can play the game how they like it, however the game will not be the exact same game as the original Space Invader in custom mode, this will be done to make it so the Player cannot cheat the database by getting the highest score on the easiest mode
         1. In the {Custom Mode} the game will have a set difficulty only allowing the player to control a finite amount of settings to their desire
      4. The {Exit} button will exit to the players desktop
   5. Players Death
      1. The player will have 3 lives and will only die after they lose all three lives
      2. When the Player Dies a Message box should appear telling the player (Game Over) as well as displaying their score
      3. After the Player Dies the player can enter their Username
      4. Once the user has saved their username their score should be saved
2. PAC-MAN
   1. Ghosts
      1. The ghosts should follow the same algorithms as they did in the original PAC-MAN game to provide a sense of nostalgia to the player which is amongst the reason for this compilation being made.
         1. The Red Ghost should follow PacMan
         2. The Pink Ghost will try to get in front of PacMan by taking into account PacMans direction
         3. The Blue Ghost will look at the Red Ghosts Position and two tiles in front of Pacman, so it takes the two tiles and doubles the distance it is away from that tile and tries to get in front of PacMan
         4. The Orange Ghost will try to get close to PacMan but will go back to his corner when he gets too close to PacMan.
   2. The Players Score
      1. The players score will go up as they collect more and more pellets which are the yellow dots on the map.
      2. The players score will also increase as they collect berries put on each of the maps
   3. Menu
      1. The menu will feature a play button which will start the game on the map they have selected to play on
      2. The menu will also feature a button which will allow the user to select which map they want to play on, I hope to create three maps on which the player can play on
      3. The menu will also feature the high score button on menu screen, this button will take the player to the high score screen. The high score for the player will be saved in a database
      4. On the menu an exit button will also be present which will close the program when pressed
   4. Map Selection
      1. On the menu the player will have the option to pick a map out of the options presented to them.
   5. PacMan
      1. The player will play as PacMan who will be a yellow sprite which I hope to make two sprites for which alternate between the two when PacMan is loaded up. One sprite will have a closed mouth the other will have an open mouth.
      2. PacMan should also collide with the Environment by stopping when it hits any of the objects placed around the Map
      3. PacMan should come out from the other side when he goes through a portal in either side of the walls.
   6. Players Death
      1. When one of the ghosts catches PacMan and PacMan runs out of lives the players score and username should be saved
3. Pong
   1. Ball
      1. The ball will be contained by two walls, one at the top of the screen and one at the bottom. The ball will collide and bounce of these walls.
      2. The paddles will try to hit the balls so it doesn’t pass them, the player or computer who let the ball pass them will allow the other to gain a point.
   2. Paddle
      1. There will be two paddles; one paddle on each side of the screen one of these paddles will be controlled by the player and the other by the A.I which will try to hit the ball back at the player.
   3. The Modes
      1. There will be two different modes for my game one mode will be Easy Mode which is the normal remake of the classic game Pong.
      2. The other mode will be Mad Mode which will attempt to trick the player and will keep the player on their toes at all times.
   4. Mad Mode
      1. In this mode the ball will have the ability to change course on its own acting through an A.I. which will try to learn from the player strategies and act accordingly
         1. The A.I. will learn from the player by seeing when the player is winning by looking at their score to see when they are continuously gaining points and judging what will throw them off.
         2. The A.I. will continuously change how its playing, to keep the player on their toes at all times for example for a few plays the ball might keep coming as a slow ball then it speeds up. The ball might also change its direction in an attempt to trick the player, for instance so if the player does not react fast enough they will lose and the computer will gain a point. These are only a few things the AI will be capable to do.
         3. The ball will be able to gain speed to try to confuse the player. Alongside gaining speed the ball will also have the ability to slow down.
         4. The ball will also have the ability to change paths in a sudden
   5. Detection
      1. It is vital for this to work as all the different parts of Pong have to be working together to allow the game to work as intended. Every time the ball collides with the paddle the player the ball should detect that and bounce of appropriately.
      2. Every time the ball goes past a paddle the game should detect that and give the other player a point and reset the game with the score intact by placing the ball in the centre.
      3. Every time the ball hits a wall the ball should detect that and bounce of appropriately.
   6. Ball Rebound
      1. The ball should be bouncing off at an accurate angle and accurate speed this can be done by using Physics Mechanics formula’s i.e. Trigonometry. It will be assumed that there is no friction for maximum efficiency so the player can enjoy the game otherwise the ball would slow down due to dampning which would make the game boring to the player.
      2. When the ball collides with a paddle it should bounce of at a realistic angle, this can be done by changing the horizontal speed of the ball after the collision. This will help provide a realistic rebound for the ball which will hopefully help immerse the player.
   7. Objective for the Player
      1. The main objective of this game will be to beat the A.I. and winning for the player in both Easy Mode and Mad Mode. As a result there won’t be an in-game high score ranking.
      2. Also the game will reset every time it loads up to provide a whole new experience each time the player wins. The first to 10 will win the game/match in both modes. The points for each player will be located in the top right corner for the player and the top left for the A.I.
      3. The player will be given the option to carry on playing after reaching 10.
4. Flappy Bird
   1. Player
      1. The Player will control the bird which will be affected by gravity which will cause it to fall down
      2. The longer the bird falls the quicker it will get as the physics in the game cause the bird to accelerate as it falls down
      3. The Player will be able to press one key which will cause the bird to hop up, using this the player will have to manoeuvre themselves through pipes
   2. Pipes
      1. There should only be two pipes at the screen as a maximum, as a result the number of pipes on the form cannot exceed two pipes
      2. The pipes should also be unique from each other with each one having the gap at a different place. In essence each pipe needs to be different from the one before it, one way of doing this would be to use a random number generator multiplied by a constant to determine the height of the gap
      3. There should be one pipe from below and one from above, both these pipes together will form the gap
   3. Score
      1. The Players score will increase every time they go through a pipe and should increment by one each time.
      2. When the Player crashes their score is saved and the game ends
   4. Random Events
      1. Using a Timer and a Random Number Generator I can implement a random event chance where there is chance of their being a random number
      2. As the score increases the chance of a random event increases
      3. Random Events can include events like the ones mentioned before so invincibility and invisibility, by doing this each event can either help the player or hinder their progression. Other Random Events, I have thought of is, their being a change in pace with the speed of the game either increasing or decreasing and their being a chance for double score or no score. Finally, I have also though of their being additional lives which the player can gain but with an equal chance of losing instantly.
   5. Player Death
      1. When the Player dies the game will save the users score to the database
      2. The player will then be taken to a high score screen where they can use Binary Search to search for their score to see their position within that game.
   6. Menu
      1. The Menu for Flappy Bird will follow a similar pattern as the other games, with their being a start game button which will lead to the game starting. There will also be an option that will start random mode.
      2. The Menu will also have an exit button which will lead to the player leaving the program to their desktop
5. My Retro Inspired Game
   1. Player
      1. The player will control a sprite in a restricted area inside the form. The sprite should be a pixel image following the same artwork as the original retro games with rough edges and squares. This is to provide the retro feel to the player when they play the game to provide an authentic feeling of a retro game
      2. The player goal will be to collect gems to increase their score as well as avoid the enemy’s
      3. The player will travel a constant speed in all four directions (up, down, left, right)
      4. The player will have the ability to gain a power for instance invincibility after they reach a certain milestone in their score. This power up shouldn’t last long as most retro games were fairly quick paced with not much room for the player to gain a breather between gameplay. There will be a set number of these.
      5. The Player should possess three lives
   2. Enemy’s
      1. The Enemy’s will also possess the same retro artwork in the same fashion as other retro games and should match up with the player sprite so they don’t seem foreign from each other and blend in to allow the player to get immersed in the game
      2. After a certain number of points the number of enemies should increase to constantly increase the difficulty of the game as well as providing a fresh play to the player. There should be a limit to the number of enemy’s that can be on the screen at a given time and in total.
      3. The Enemy’s should have an AI which will cause it to chase the player and not to overlap each other
      4. Every time the Enemy and Player character intersect the player loses one life and point the the game resets the players position
   3. Gem’s/Score
      1. As the player collects a gem they gain one point to the score and the gem moves to a random place on the form
      2. When the player collides with an enemy they lose one point from their score as well as a life
      3. Once the player has lost all their lives they will be asked for their Username which will be used to save their score to the text file
      4. The player with the highest score will have their name displayed on the game screen with their score
      5. All of the scores that are saved should be saved to a text file
   4. Menu
      1. The Menu should contain instructions on how to play the game
      2. The Menu should also contain a start game button which will start the game
      3. The Menu should also contain a button which will take the player to a score screen where they can use Merge Sort and Binary Search to search for their score.
6. The Menu
   1. How
      1. The menu will be made from C# in Visual Studio and by using the .NET Framework I hope to link all the games and the menu together. This is because it makes the game run more efficiently as the game will load up when it is desired otherwise the code will not run. This is because it won’t be directly incorporated into the game but will be linked by a button which loads up that code when needed
   2. Look and Work
      1. The menu will be made with buttons which will run the games code causing it to run and open. The menu will have the screenshot of the game and its title. They will also have a button next to them which when pressed will allow the user to play that game. It will also include a high score button which will take the user to a top 5 list where they can search for their scores, through Binary Search. The scores will be sorted through Merge Sort.
7. Data Storage
   * 1. Text Files will store the Players Username and Score in all five games, all of these files will have Merge Sort ran on them which will allow the user to search for their score using Binary Search as Binary Search requires the data to be sorted.
     2. The scores from all of the text files from each game will be added together to form an overall score, this overall score will be saved to another text file in the High Score screen which will display the top 5 Players.
        1. All the scores in the text file (with the summed scored) will be sorted with Merge sort to provide a leader board. Binary Search will also be implemented to provide a search function for a player to search them self if they’re not on the leader board. The combination of Merge Sort works well as they both are efficient algorithms as Merge Sort as an efficiency/complexity of O(nlog(n)) which means it is efficient and a good algorithm whilst Binary Search has a efficiency/complexity of O(log(n)), this means that Binary Search is a very good algorithm. As Binary Search requires the data to be sorted Merge Sort provides a good sorting algorithm to sort the data to allow Binary Search to be used. As I have used Big O notation, both of these values are considering that the algorithm is going through a worst case scenario.

**Modelling**

[See Appendix For Data Flow Diagram]

My Game will use a flat file database of sorts via textfiles; this model (data flow diagram) shows how the database and game work together. The diagram briefly illustrates the flow of data around my project.

|  |  |  |  |
| --- | --- | --- | --- |
| Name  ***Data Dictionary*** | Purpose | Type | Example |
| Score | The Purpose of the score is too save and contain all of the players score in a text file in all of the games. | Integer | 100 |
| Map | The Purpose of this is to save the players choice of map. This will only be used in PacMan | String | “Hello” |
| Speed | The Purpose of this is to show how fast the Player moves, it’s a universal variable which will be used in every game with a different variable to either control the Player Speed or Enemy Speed | Integer/Single/Float | 52 |
| Lives | The Purpose of this is to show the player how many lives they have remaining. They will lose a life every time they die. This will be used in | Integer | 3 |
| Points | The Purpose of this is to show how much the Score Variable increases by after a certain event takes place | Integer | 25 |
| Mode | The Purpose of this is to allow the user to choose which mode they want to play in. | Integer | 75 |
| High Score | The Purpose of this is to store the player high score from the text file after it updates and display it on the main menu screen. | Integer | 111111111 |
| Number of Enemy’s | This Variable controls how many numbers are on the game at a given time. This will be used to either increase the number of enemy’s, decrease them or reset them | Integer | 2 |
| Player Direction | This Variable will be used to control the direction the Player moves in. This will most likely be done via calculation using the vector directions the player faces with the speed, the value produced through the calculation will be saved in the variable | Integer | 5 |
| Gravity | So far I think this variable will only be used in Pong and Flappy Bird to make sure the ball goes down | Integer | 7 |
| Random Event | This Variable will be used to control when I want the player | Boolean | True/False |

This Data Dictionary shows what variables and data I intend on using so far, this could change when I start to develop my Program

**Acceptable Limitations**

One Acceptable Limitation is that some of the games will most likely not have any sound which is due to the limited amount of time available. Another Acceptable Limitations is that there may be no animations for many if not all of the games due to the size of the project and the small amount of time available. This is as I will be aiming to complete my scoring system as it’s the most important part of my project as it is what most of the games are aiming to get to, by storing there scores. However this is not to be mistaken with the games being low quality nor “bad” as I will try my best to take as much inspiration from the originals as I can.

**Design**

**Overview**

In this Project I will be creating 5 different retro arcade games; however my remakes will have enhanced playability and as a result will have more replay value compared to their original counterparts. These modification include, a ‘Mad Mode’ in the game Pong, which allows the ball to change its speed and direction in an attempt to trick the player, 3 maps in Pac-Man, a Random Event mode in Flappy Bird, a customisable experience in Space Invader and finally I will make a game of my own idea which will follow the same feel of a retro arcade game. These games will look similar to their classic counterparts to provide younger players with a similar feel, so new gamers can realise how far gaming has come and advanced whereas older players will be able to experience a feeling of déjà vu. My main enhancement will be that all the scores from each game will be compiled together and saved in a flat file database where they will be sorted and ordered to form a top 5 via Merge Sort. The user will be able to search through the scores to find theirs using Binary Search once the data has been sorted. By adding these changes to the game, it should add a new level of difficulty to the games for gamers who are looking for a challenge, whereas also retaining the classical mode for gamers who are looking for a more laid back, classical and easier feel to the game.

The introduction of ‘Mad Mode’ in Pong makes it difficult for the player to anticipate where the ball is going to next, making it so the player has to stay on their toes at all times. The ability to make choices in Pac-Man allows the player to go out of their comfort zone, as they will most likely be used to the classical Pac-Man map. This added challenge will increase the replayability of the games. The Random Events in Flappy Bird will provide a fresh experience to the user as the player can no longer follow a simple tactic to get through all the pipes as now they can have events such as the invisible event which will make the player invisible so they have to pay attention to the birds location at all times just in case the bird becomes invisible. The Customisation options in Space Invaders will allow the user to change how they want to play the game so they can improve and steadily increase the difficulty as well as play the game according to their mood. This allows for a fun experience which the user can control. In the game that I am making which I have planned to name ‘Bling Bling Boy’ the game will contain an AI which becomes better as the players score increases, the AI will chase the Player around the Map, however the player will also be able to become invincible which will allow the Player to become unbeatable for a few seconds a certain number of times.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage  Design Figure 1:  IPSO Table | Outputs |
| Keyboard :  ‘W’ ‘A’ ‘S’ ‘D’ ‘Space’ | Movement and Collision Detection | In game private and public Variables | Display/Monitor |
|  | Computer AI Movement | In game private and public Variables |  |
|  | Data Storage i.e. Scores | Text Files i.e flat file database |  |
|  | Shooting (Space Invaders only) | Arrays and Variables |  |
|  | Customising Settings | Public Variables |  |
|  | Sound | Variables and/or file location | Speakers/Headphones |
| Mouse | Movement and Collision Detection | In game private and public Variables | Display/Monitor |
|  | Data Storage i.e. Scores | Text Files |  |
|  | Random Events i.e. Ball Changing Direction | Random Number Generator and Variables |  |

The main form of input will be the keys ‘W’ ‘A’ ‘S’ and ‘D’ (for PacMan, Space Invaders, Bling Bling Boy and Flappy Bird) and mouse movement (Up and Down for Pong). The keyboard will also be used to record the players name when they save their score in the database. The mouse will also be used in pause menus and general interaction with the User Interfaces like menus, but not limited to just that.

The main processes will be Collision Detection, as it is going to be applied in all the games I am going to be attempting to make and it will be the fundamental process that causes all the other operations. In PacMan, collision will be detected by using a function which will check if the player (PacMan) or the Computer AI’s (Ghosts) are moving towards a dot which the player will be able to pick up to increase their score.

Pseudocode for PACMAN gaining points

While gameIsRunning

If PacMan CollidesWith Pill Then

plyrScore += 10

End If

End While

If the Computer AI goes over a dot, no interaction should take place. My Collision Detection will work by looking at points where there are multiple directions for PacMan to choose from, at these points I will have pills which will allow PacMan to choose which way he wants to go. As a result of this I will not require hit boxes around each tile in the game as it won’t be possible for PacMan to hit the walls as he would only move in between these points. The same will apply for the Ghosts. Between these points there will be normal pills which PacMan can collect to increase his score. The Ghosts should have no reaction with these pills.

For Pong the Collision Detection function will always check around the ball to see if it collides with the top or bottom walls. If it collides with the top wall it should come off at the same speed as in the game momentum will always be conserved. The same applies when it collides with either the player or the computer paddle. If the ball collides with the walls behind the paddles a point will be added to the paddle opposite to the wall, where the collision took place.

Ball:

Checking for the Top Wall:

If Ball CollidesWith TopWall Then

BallDirection 🡪 -BallDirection

VerticalSpeed 🡪 -VerticalSpeed

End If

Checking the Bottom Wall:

If Ball CollidesWith BottomWall Then

BallDirection 🡪 -BallDirection

VerticalSpeed 🡪 -VerticalSpeed

End If

Checking for Player Paddle:

If Ball CollidesWith Player Then

BallDirection 🡪 -BallDirection

HorizontalSpeed 🡪 -HorizontalSpeed

End If

Checking for Player Wall:

If Ball CollidesWith PlayerWall Then

CompScore += 1

Update ScoreLabels

Reset BallLocation

End If

Checking for Computer Paddle:

If Ball CollidesWith Computer Then

BallDirection 🡪 -BallDirection

HorizontalSpeed 🡪 -HorizontalSpeed

End If

Checking for Computer Wall:

If Ball CollidesWith CompWall Then

plyrScore += 1

Update ScoreLabels

Reset BallLocation

End If

In Pong the Computer Paddle will try to follow the ball to see where it goes and try to hit it. However as the original Pong had the AI sometimes go in the wrong direction or stop my Pong AI will do the same, currently I plan to implement this with a Random number generator (through the .NET framework) which provides there being a chance which decreases as the players score increases.

In PacMan, each ghost will have fixed response to carry out, in an attempt to trap and catch PacMan.

Blinky (Red Ghost): (Blinky always follows PacMan, unless it’s in Scatter Mode or Frightened Mode)

If Mode 🡪 Chase Then

Blinky MoveTo PacMan

BlinkySpeed 🡪 5

End If

Pinky (Pink Ghost): (Pinky tries to look at where PacMan is going and tries to go 4 tiles infront)

If Mode 🡪 Chase Then

x 🡪 PacManOrientation

End If

(Dependent on x)

Pinky MoveTo (PacMan.x + 4)

End If

Inky (Blue Ghost): (Inky Look at Blinkys Position and two tiles infront of Pacman determined by Pacmans orientation, so it takes the two tiles and doubles the distance Blinky is away from that tile)

x 🡪 PacManOrientation

Check x against (Up, Down, Left, Right)

(Dependant on x)

DistanceX 🡪 (PacMan.x + (PacMan.x + 2)2)) Or DistanceX 🡪 (PacMan.x + (PacMan.x + (PacMan.X - 2)2))

DistanceY 🡪 (Blink.Y + PacMan.Y)2 Or DistanceY 🡪 (Blinky.Y + (PacMan.Y + 2)2)

InkyMoveTo 🡪 (DistanceX + DistanceY)1/2

Clyde (Yellow Ghost): (Clyde follows PacMan until it gets within 8 tiles of PacMan then Clyde returns to his Home Corner which is the Left Corner)

If ClydeLocation <= 8 tiles to PacMan

ClydeMode 🡪 ScatterMode

Else

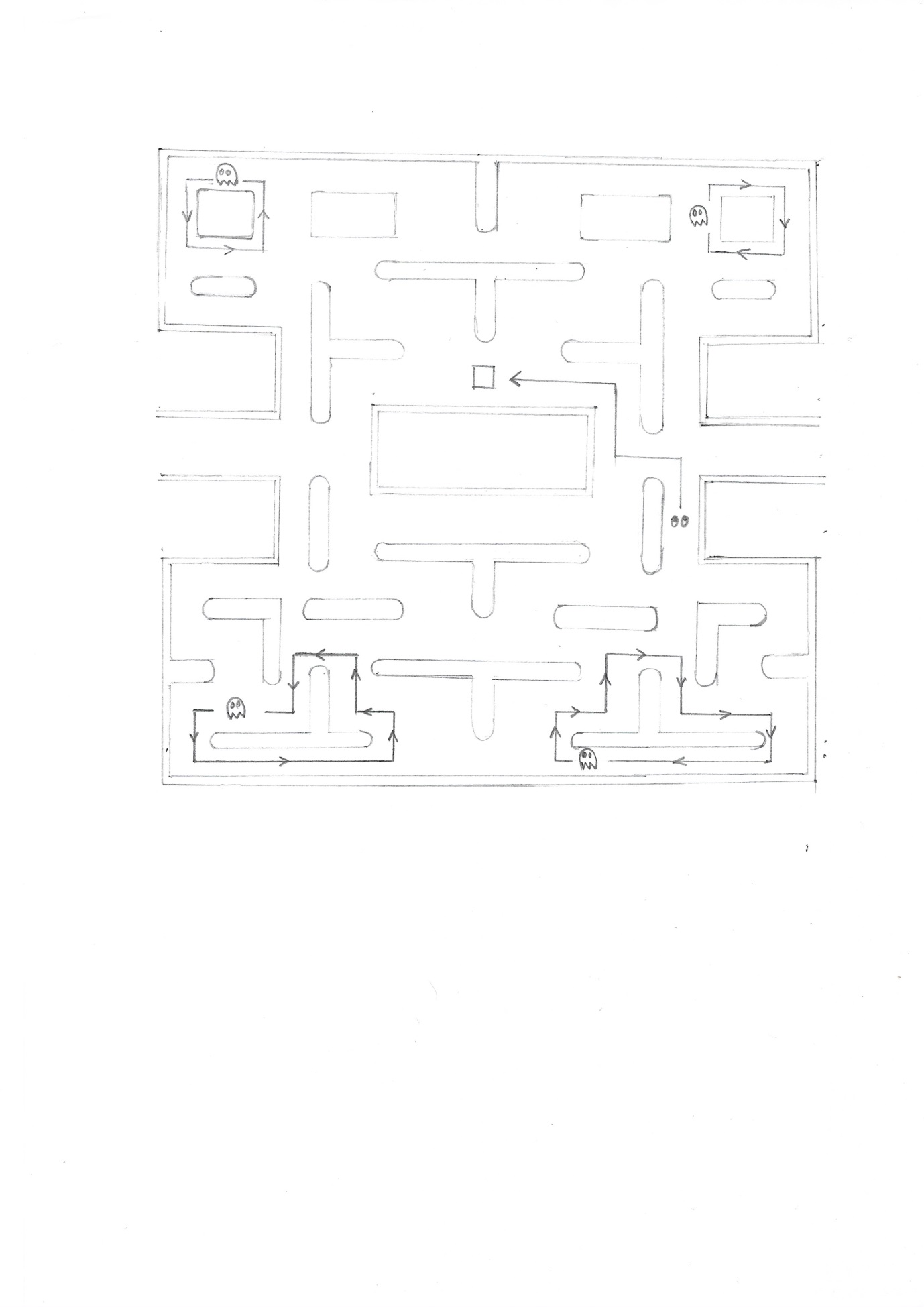
ClydeMode 🡪 ChaseMode

End If

If ClydeMode 🡪 ChaseMode Then

ClydeMoveTo 🡪 (PacMan.x, PacMan.y)

End If

The Sketch below shows how scatter mode will function for each Ghost in PacMan:

Pink Ghost

Red Ghost

When in Consumed Mode

Orange Ghost

Blue Ghost

The Sketch above shows how the Ghosts will move when they are in Scatter Mode and Consumed Mode. Frightened Mode isn’t shown as in Frightened Mode the Ghosts just move at random but at a slower pace. When the Ghost is in Frightened Mode and is Consumed, the Ghost enters Consumed Mode, in this mode the Ghost returns to the Ghost Home which is located in the centre of the Map, as shown in the sketch above. When the Ghosts are in Scatter Mode the Ghost goes round their Home Node as shown in the Sketch above. Each colour Ghost should return to its Home Node and move in the pattern as shown in the Sketch. If PacMan collides with any Ghost whilst they are in Scatter Mode PacMan dies and the game is reset unless the player has any lives remaining. The Pseudo code for Scatter Mode for all the Ghosts can be seen below:

Scatter Mode Pseudo Code:

If ghostMode 🡪 ScatterMode Then

ghostMoveTo 🡪 HomeNode

End If

The Pseudo Code above shows that if the Ghosts mode is Scatter Mode then they move to the Home Nodes otherwise they follow the algorithm they were following previously for the mode the Ghosts were in.

In Flappy Bird there will be no AI, as the player will be playing by themselves as Pipes Generate randomly across maintaining a constant gap between as they generate. Even in the Random Event Mode, there won’t be an AI but will work through a random number generator which will result in random events taking place.

In Space Invaders the aliens won’t have a AI but will strafe all the way to the left boundary of the from then move down then strafe to the right boundary and move down, the aliens will continue this motion until one of them reaches the bottom of the form. Using the .NET Framework random number generator the program will select aliens which will shoot down at the player

In the Game I am making (i.e Bling Bling Boy) the AI of the Bugs will work in the following way where the Bugs will track the players position and move towards him however as there will be multiple bugs using the same AI one problem this would cause is that eventually all the bugs will go on top of one another thus making it seem as though one bug is chasing the player. One way I have thought of overcoming this problem is by making it so the bugs bounce of each other

Bug Chasing Player: (The Bugs will track players location, as this would have to be updated every single time the best way to do it would be to use a Timer and using the TimerTick event to track the Location of the Player)

bugMoveTo 🡪 (player.Location.X, player.Location.Y)

Bug Bouncing of Each Other: (When the Bugs begin to overlap each other they will start bouncing off in the opposite direction, this will be done by either increasing or decreasing their distance from each other horizontally)

If Bug1 CollidesWith OtherBugs Then

Bug1 🡪 MoveAway

ElseIf Bug2 CollidesWith OtherBugs Then

Bug2 🡪 MoveAway

ElseIf Bug3 CollidesWith OtherBugs Then

Bug3 🡪 MoveAway

ElseIf Bug4 CollidesWith OtherBugs Then

Bug4 🡪 MoveAway

End If

Even though the Pseudocode above allows for the bugs to overlap vertically, it won’t let them align as they will start to move away from each other.

The data storage will save the players score from each game in different text files in that game folder, then in my final high score screen the program will take all of the scores of all the users and add them together, then run Merge Sort through the scores to display the top 5 players. The computer will check the username of each score from each game, that way the computer will know which scores to add. On this high score screen the player will be able to see the top players as well as search for their position through Binary Search. One way I can think of to do this is to read the names and scores to a list, dictionary or an array. Then on one of these data structures, I can run the Merge Sort and Binary Search algorithms.

In Pongs Mad Mode Random Events of sorts will occur in the form of the ball changing direction and speed randomly, this will be done through a function/subroutine and a random number generator where each generated number will lead to a event related to the generated number, for instance Number 1 might mean the ball moves up with an increased speed, whereas number 3 might mean the ball moves down with an increased speed. Number 2 might mean the ball continues as it was.

In PacMan there will be berries which will spawn on the map this will be achieved through a timer and a score/level checker. The berries will provide more points allowing the player to increase their high score. These berries would also need to disappear after a certain amount of time.

The main form of storage will be text files, each game will have a separate text file which will act as an external memory system, these files will store the player’s username and score which will be displayed in the following format:

Test1, score1

Test2, score2

The data will be read before the Merge Sort algorithm, this is done so every time new data is added or the player stores a new score to the text file, Merge Sort is ran on that data as well. Once the data is sorted the player will be able to search through the data as Binary Search requires the data to be sorted. Once the player has played the games they want to and they go on the high score screen which displays the top 5 players. When that Form loads all of the data should be summed and added to another text file with the username in the same format as the one show above. Then the text file will be read, this is done to take into account, previous users scores. Once the data is read from the text file the Merge Sort algorithm will be ran which will sort the data, once the data is sorted the Top 5 Players will need to have their names displayed and the Binary Search function should also become usable as the data will be sorted.

The map for PacMan will be designed in a map maker where the map can be previewed here the map can be screenshotted or saved as a image and used to make the map in the game to provide premade maps to the game which will be different from the original PacMan map. Each point in the map maker will be a coordinate of a graph of a fixed size. At each coordinate a tile will be placed, each tile will have a different number allocated to it, making the map maker act as a jigsaw for comparison. I will make a map maker which will allow me to make the maps more easily. The user can also use the map maker to make their own maps as well for PacMan, however this isn’t compulsory as it isn’t one of my prime objectives.

The only form of output is the Monitor or any other form of display where the program can be ran. This is to allow the player to see the game so they can play it and to interact with any menus as the Monitor will present the game environments alongside any menus and any other User Interface. In general the Monitor (display) will show any user interactions.

Design Figure 2:

Menu Design for game selection

**Planned Design**

In line with my retro and simple theme, the game selection menu won’t be anything complicated and will be a simple user interface which will allow the player to select which game to play, the game will then run. If High Scores is selected the Top 5 scores will be seen alongside a search which will find your score. The search will be done through Binary Search. I don’t plan to add any colour to it as it is quite basic and isn’t completely essential to my project.

Pong

Pac-Man

Menu

Space Invaders

Bling Bling Boy

High Scores

Flappy Bird

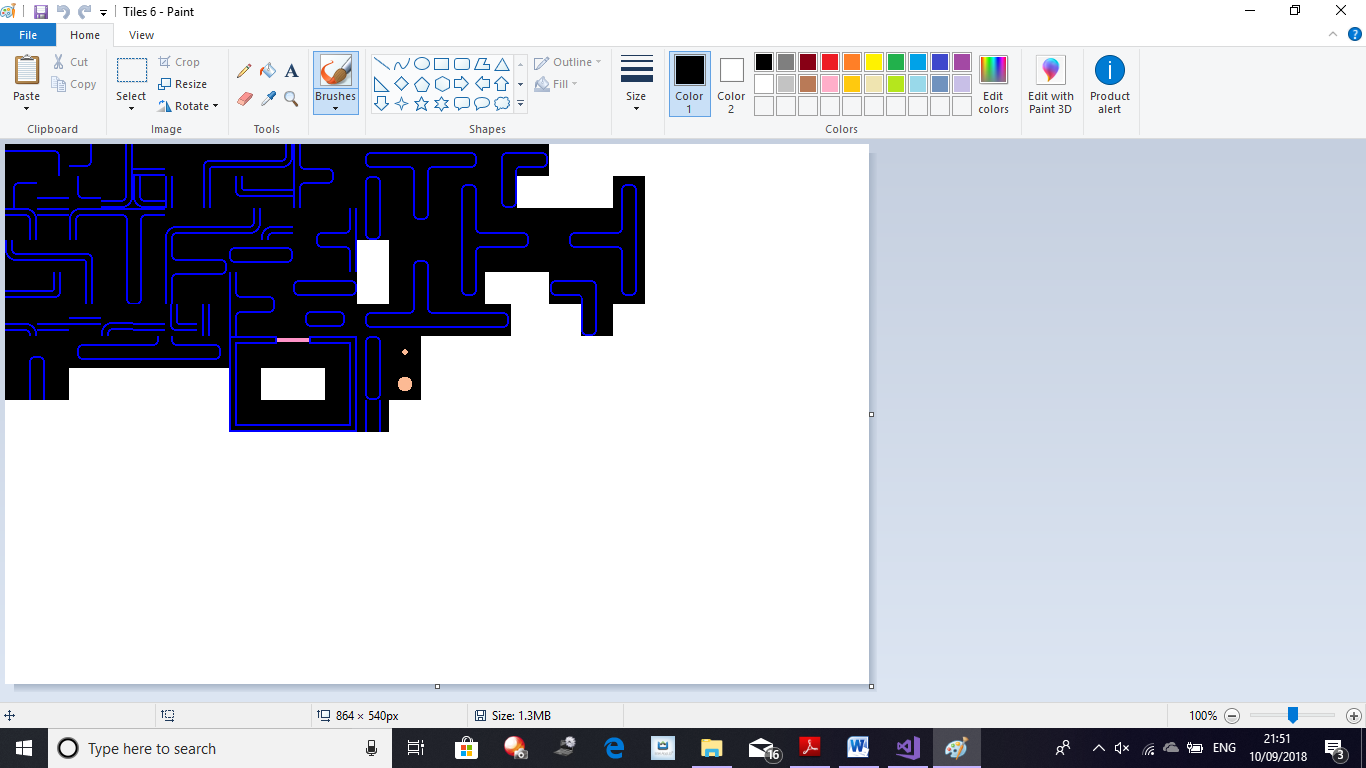
Design Figure 3: Pong in Game Display

0

0

Figure 3 shows my design for Pong, as you can see it is a simple design which does try to mimic its original counterpart. Both Paddles can be seen on the screen clearly, allowing the player to clearly see the location of their paddle. I plan to make the right paddle the player paddle. The ball can be seen on the screen as well with two boxes with the number 0 in them. The number 0 will go up once either the player or the computer gain a point. The planned pseudocode has already been stated above in Pseudocode.

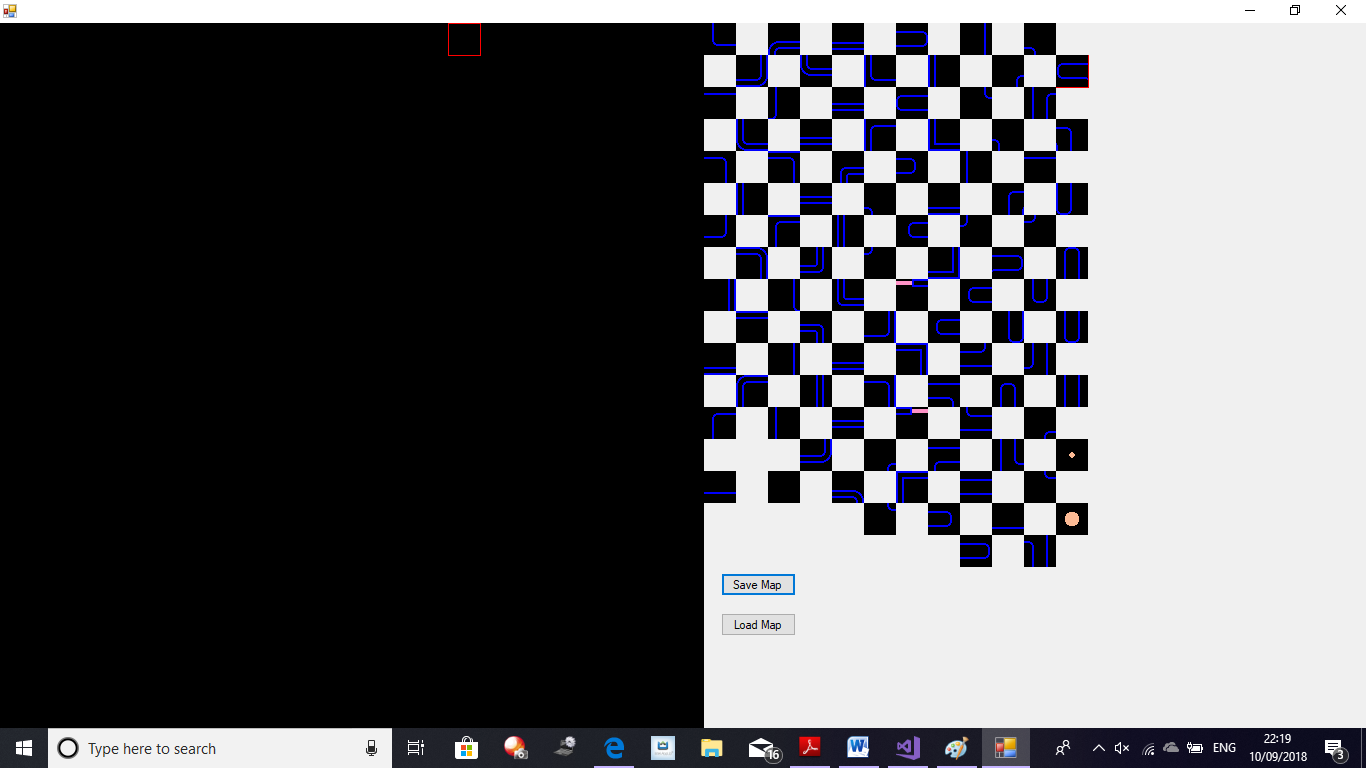
For PacMan I have made a map maker which follows the idea of a graph being present as the game area. Then by making the game map and breaking it down to tiles of size of 32\*32(pix), you can insert the tiles and connect them together to make different game maps.



Design Figure 4: PacMan Tiles

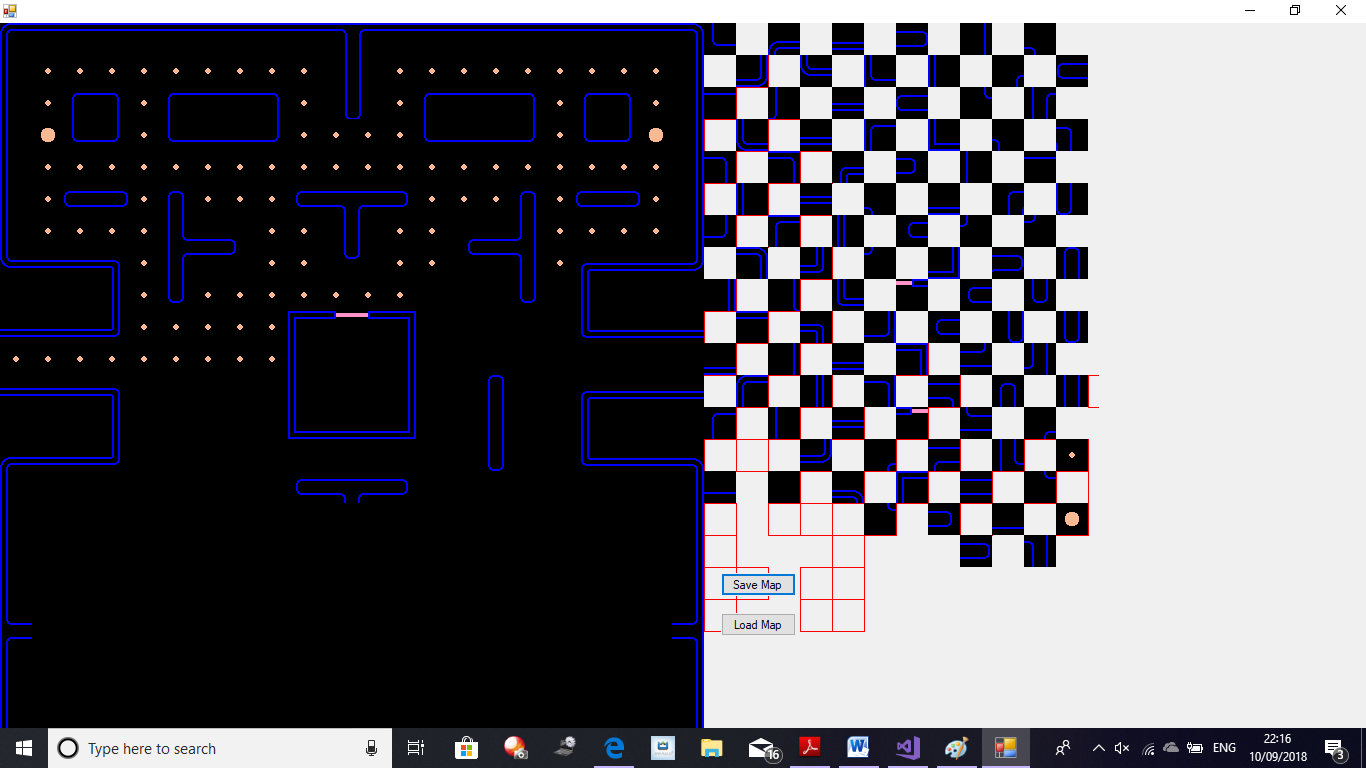
Figure 4 shows the tiles I cut up from the PacMan Map and put them all on a BMP file together. With these tiles I can create different variations of the PacMan map just by importing their tiles cut up. Using the program I have made to allow me to make the different game maps more easily.

The Program uses the BMP file and assigns coordinates to the whole file, starting with (0, 0) and increasing by 32 in either the x or y axis as each tile has the size 32\*32(pix).

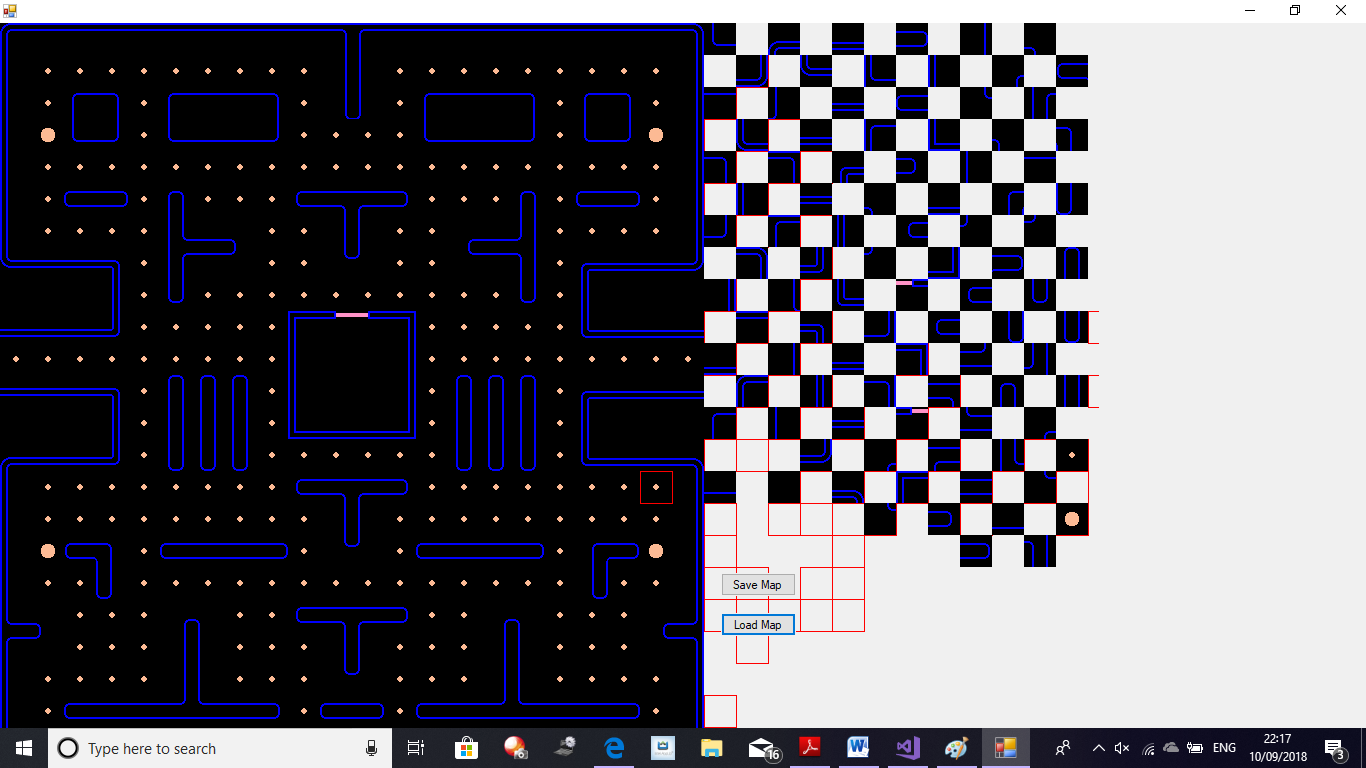


Design Figure 5 Map Maker display on start up

As it can be seen in figure 5 the Map Maker possesses a simple design with the game area starting of a black screen, with the tiles on the left. This makes it easier for me to make the maps. The map acts as a 2D array, storing the tile number (The number associated with each tile) and the tiles position. When the Map is loaded the program loads up the tile number which is associated with an image causing the map to be made.

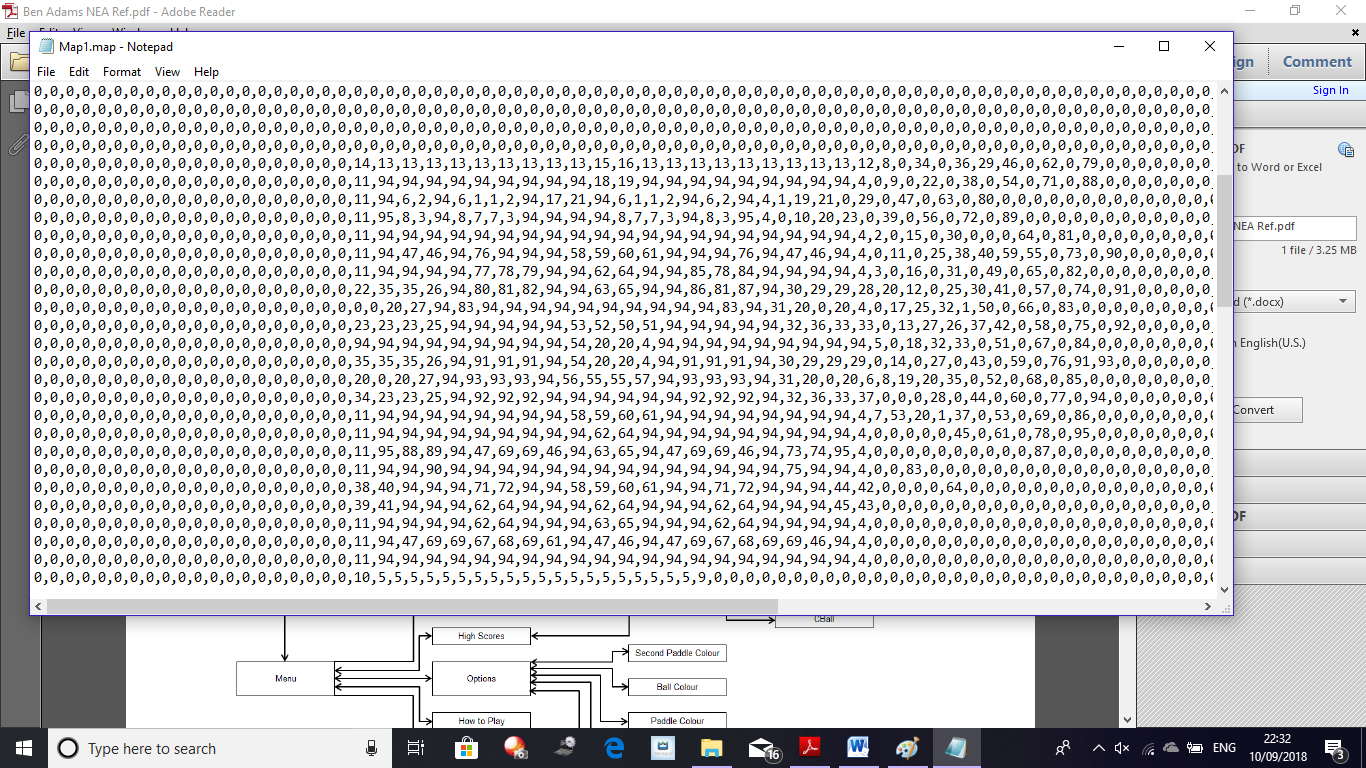


The Code for the Map Maker will be available in the Appendix



Design Figure 6: Example Map Made by Map Maker

In Figure 6, the Map Maker can be seen working to form a variation of the PACMAN Map. The Red outline when on the map maker area indicates the location of the mouse, whilst also allowing me to see which tile I’m targeting. The red outline always rounds down coordinate wise to allow more accurate selection when the mouse is on the edge of a tile. The Red Outline can also be used to see the 32\*32(pix) boxes more clearly. The Save Map button converts the entire map to a text file which can then be imported into the PacMan game, allowing the whole map to be loaded up so it can be worked on later if needed, as long as the BMP file associated with it is also imported.



Design Figure 7:

The Text File Created when the PACMAN Map is saved.

Using the technique of file reading and writing with I can produce many different maps. Figure 7 is produced when the save button is pressed. When the load button is pressed this file is loaded up and is used to make the map as long as all the tiles are also imported to the program. This allows for me to retain my progress on the map if, for instance my computer crashes.

**The System**

Design Figure 8: Diagram which shows the links between the User, the Front End and the Back End

User

External Files

Front End

Back End

Figure 8 shows a diagram similar to a Client-Server model. In a Client-Server model, the Client is considered the Front End and the Server is considered the Back End (presentation work is done on the Client Server (Front End)). In the diagram shown in Figure 8 the Front End is what the User sees and interacts with so things like User Interfaces and the game (program) in general. The Back End is what the User doesn’t see like their scores being converted to strings and being stored in text files then being shared to other text files to sum up the score and being sorted by merge sort to form a top 5 list. It links to the data flow diagram seen in the Analysis (See Appendix).

In my program and due to the software development tool I’m using (Visual Studio), my project will contain Windows Forms alongside, Classes; Modules and Functions to state a few. These Windows Forms and Classes will be the Front End and they will work in coalition with the text files (flat file database) to form the Back End. The Windows Forms will be what the user interacts with as they will contain the Menu and the games as well as any other User Interface, whereas the Classes and the Modules and Functions will be what the User doesn’t see creating the Back End. Some Functions, which will create the back end, will be the functions and Modules which write to and read from files, this will create the External Files. External Files will include the Map file shown prior as well as the function for sorting the scores (Merge Sort), the scores being summed, stored and sorted is also an example of the Back End. The Front End will consist of process like the Computer AI as well as the code for Collision Detection just to state a few as they will be seen acting by the User. All of the Processes in the Front End will be used to make a fully working project which should function correctly. This layout will be used in all of the games as the foundation to creating them.

**Modular Design**

Design Figure 10: System Flow Diagram See Appendix

Design Figure 9: Hierarchy Chart See Appendix

The Hierarchy Chart and the Data Flow Diagram show the planned layout and how the programs will communicate with each other between the Windows Forms Systems and through different Modules that I am using. Movement in the Menus will consist of the user clicking button, which will lead to that program being launched.

My System Flow Diagram shows that when the system starts, the global modules will run once and then open the “Menu” (which will be coded in C#). The Menu will present 7 options to the User. Clicking on one of these options will correspond with the appropriate action taking place, for instance if “Quit” is pressed the program will be closed or if “PacMan Menu” is pressed it will load up the PacMan Menu project. Movement between “Start Game”, “Container” and one of the options afterwards (“Maps” or “Mad Mode” or “Random Events”, “Options”), will automatically take place throughout the program when they are called upon by the user by selecting that mode. “Globals” in the System Flow Diagram sets up all the main global variables and objects which will be needed by all the games and menus. Setting up “Globals” allows me to call and transfer variables like score between different forms. This means that the process of storing the scores becomes quicker as I don’t have to write and read using streamreader/streamwriter from other files to transfer data between forms. “Home Menu” is the screen the User will initially see and it will allow the user to select which game they want to play and it will load up that program. It will also allow the User to see the Top 5 players and their total scores through the “High Score” program. “Start Game” loads up the original copy of the games (original counterparts without my implementations). This will load up the game program which will allow the user to play the normal version of the game. However if the user clicks on “Maps” or “Mad Mode” or “Start Random” or “Custom Game” this will open up the version of the game with my implementation, for example in Pong it will load up “Mad Mode” which is my twist on the original Pong game (more detail on my implementations can be seen in my Objectives and in the Design Overview). All of the Options from each game Menu are linked to “Container” which in essence is a literal container which will contain all of that games sub routines and functions, as a result “Container” controls the game. “Maps” and “Mad Mode” and “Random Events” and “Options” act as sub routines/classes which will be called, this will cause that implementation to run in that game instead of the original game being ran. All of the “Container” which are linked to the original game are linked to the High Score as I have decided to only allow the user to save their score from the original game instead of also adding to their total score through the implementations as it will be fair to the user, for instance a player using the “Options” from the custom game of Space Invaders will be able to gain points more easily by setting all of the options to really easy values, for instance setting it so they are only against one enemy, this means that a player playing it with all the values maxed out will be doing more work for the same amount of points. As a result I have decided to only store the scores from the original remakes of each game as they are the same set difficulty for everyone, meaning that it is fairer on all users. “High Score” from the “Home Menu” will lead the User to the top 5 player list, which will sum all the scores of every single user, then run Merge Sort which will allow me to easily select the top users as the data will already be sorted (more detail can be seen in my Objectives and in the Design Overview). “Quit Game” allows for the user to leave the program, this will not be the only method to quit as the user will still be able to access minimize, maximize and close buttons.

Each games menu should follow a pattern similar to the figure seen below:

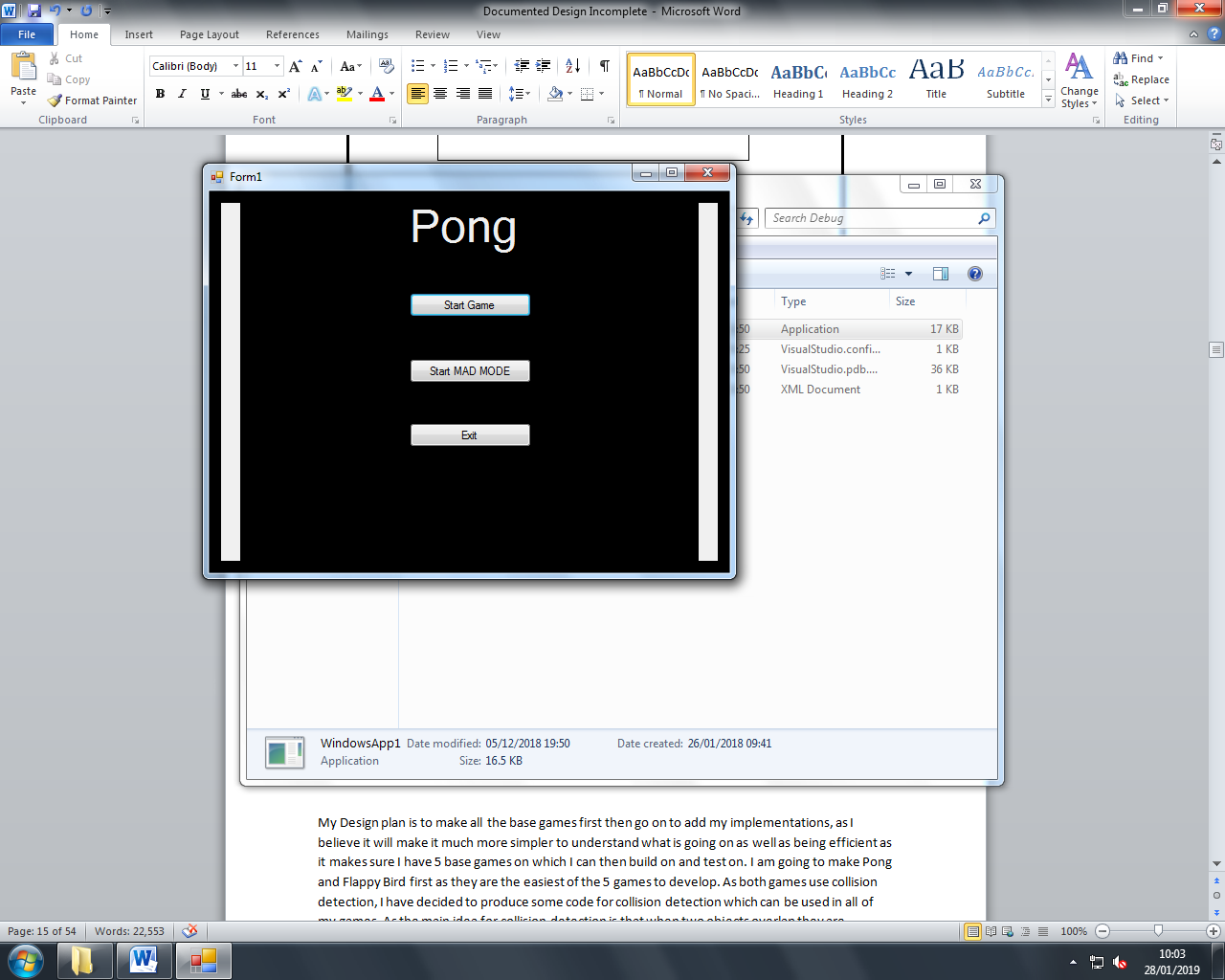
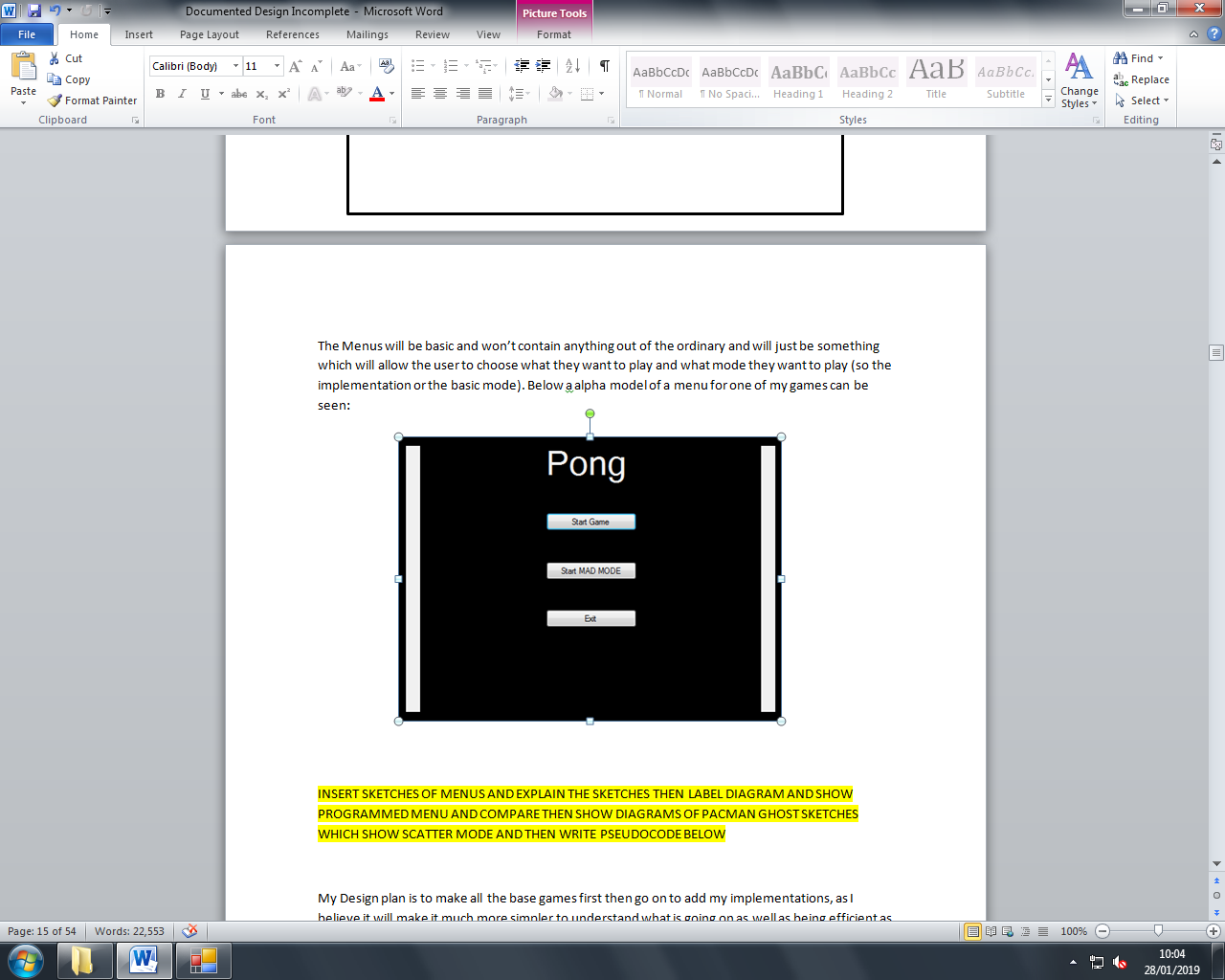
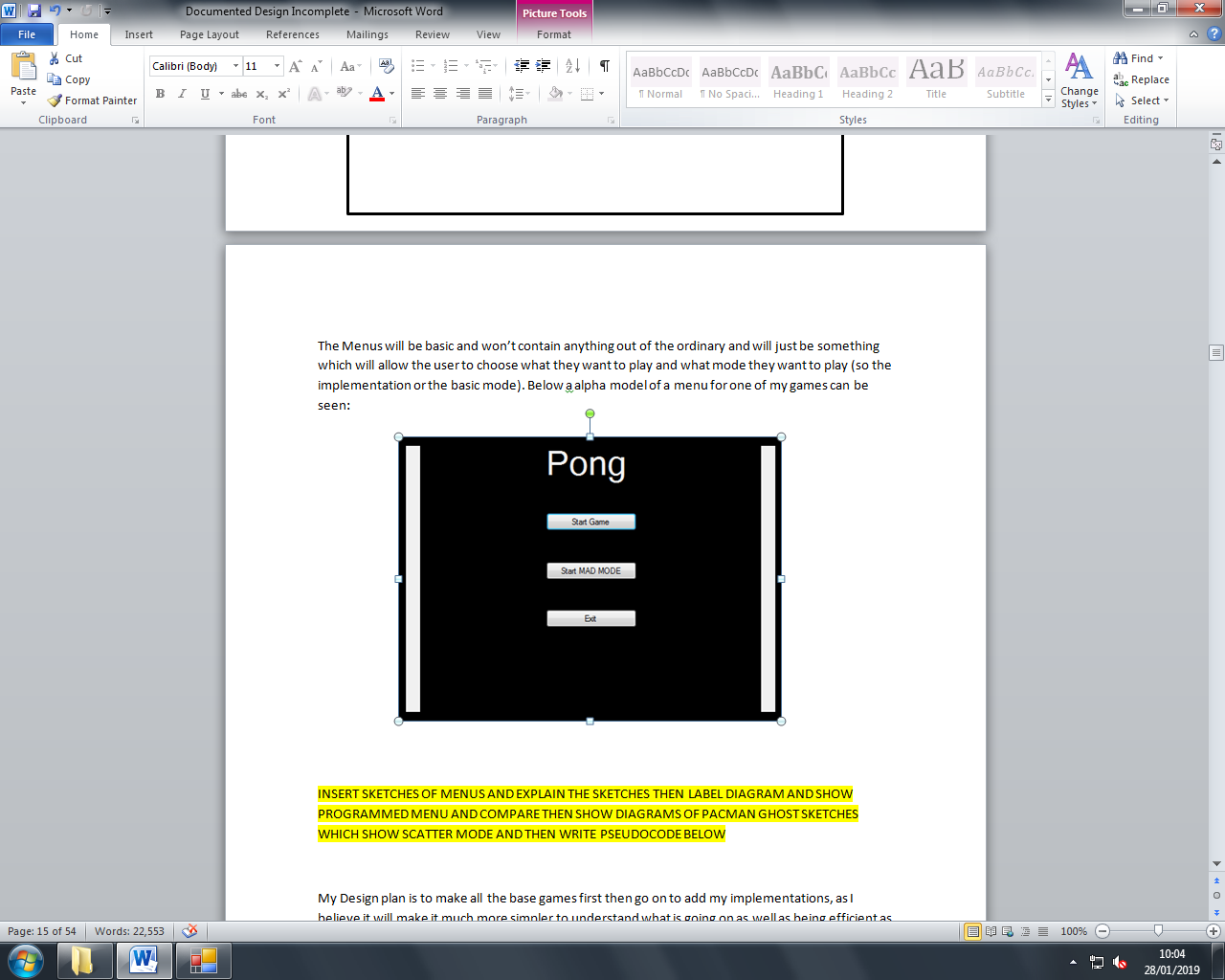
Game Title

Start Game

Start Implementation

Quit

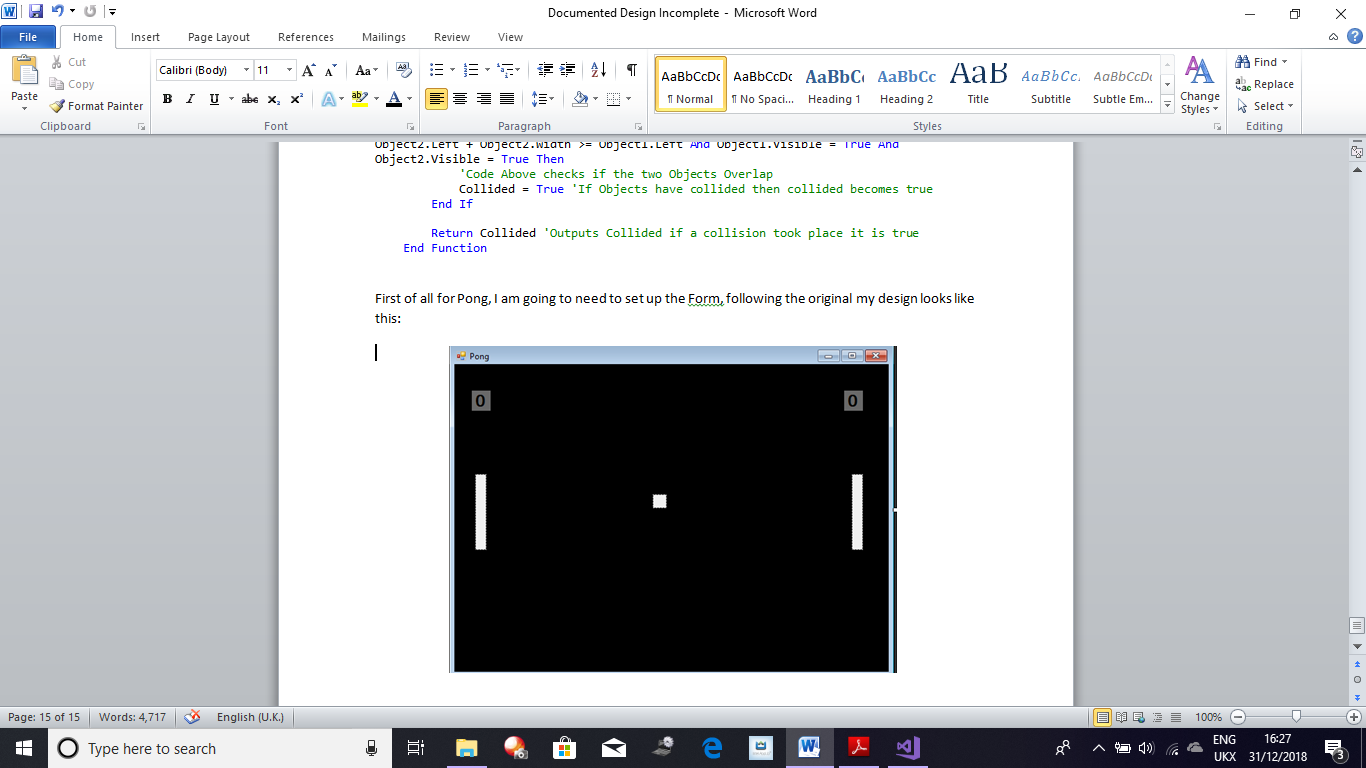
The Menus will be basic and won’t contain anything out of the ordinary and will just be something which will allow the user to choose what they want to play and what mode they want to play (so the implementation or the basic mode). Below a alpha model of a menu for one of my games can be seen:



This design is simple and is easily understood as result I don’t believe any users will find it difficult to navigate. The code for all my menus should be available in the Appendix.

My Design plan is to make all the base games first then go on to add my implementations, as I believe it will make it much more simpler to understand what is going on as well as being efficient as it makes sure I have 5 base games on which I can then build on and test on. I am going to make Pong and Flappy Bird first as they are the easiest of the 5 games to develop

**Making Pong**

First of all for Pong, I am going to need to set up the Form; following the original game my design looks like this:

Design Figure 11: Pong Game Layout

Score Labels (Left: Computer) (Right; Player)

Ball

Player Paddle

Computer Paddle

***Making the Pong Ball and Paddles***

The Player will control the paddle with their mouse and the computer paddle will track the balls position and move to a position where it will hit the ball, however at the same time I need a sub routine which turns off the computer paddle AI so the player can score points because if I didn’t do this the computer would always hit the ball as it would always go the position where the ball is heading. I also need the ball to move when the player is ready and the ball also needs to collide with the top and bottom of the screen, it needs to rebound of the top and bottom of the screen in an accurate manner. To do this I am going to add two timers which will constantly update the game with new information by checking what is happening. As this is the base mode of Pong I am currently not taking into account Mad Mode. Initially I am going to allow the player to move their paddle using the mouse. I am going to do this by using the .NET framework and its built in sub routine of MouseMove. My second concern is to allow the ball to move. As the ball will be moving consistently I am going to need to code it inside a timer so its position is always taken into account to check for any collision. As I need the ball to move when the space bar is pressed, I have decided to make a variable called BallShot which will be a Boolean variable and will store weather the ball has been told to start moving or not:

Dim BallShot As Boolean = False

(Pseduocode Detailing How Ball Shot will be Utilised)

If BallShot 🡪 True

Ball 🡪 StartsMoving

End If

If Ball CollidesWith HorizontalWall Then

BallShot = False

BallLocation 🡪 ResetLocation

End If

Initially this variable will be set to false as I don’t want the ball to move until the space bar is pressed. Then by using the .NET framework again and this time using the KeyDown subroutine I am going to change the value of this variable. When the KeyDown Sub routine is ran, so when the space bar is pressed this variable will become true. This variable is also going to be directly linked to a timer. So when BallShot is True and the ball is allowed to move, I now need to make it so the Ball can move. By using some basic Physics, the ball is going to need to need two variables for velocity. I am going to name these variables xVel and yVel as they are to represent the horizontal and vertical speed respectively. These variables aren’t going to be constant and will be different each time, I have done this by declaring them as singles (number to 1 decimal place) with rnd:

Dim xVel As Single = Math.Cos(rndInst.Next(5, 10)) \* Speed 'Velcoity of the Ball Horizontally (on the x-axis)

Dim yVel As Single = Math.Sin(rndInst.Next(5, 10)) \* Speed 'Velocity of the Ball Vertically (on the y-axis)

The xVel will use the cosine angle with and random integer between 5 and 10, this makes it so the ball moves at a random angle. This angle will be multiplied by speed which is a variable with a constant value:

Dim Speed As Single = 15 'Ball's Speed

This calculation provides the horizontal velocity. The same is done for the vertical velocity, only Sine is used instead of Cosine. Sine and Cosine are used as:

By using Angle θ, we get x by using trig: Cos θ = Adj/Hyp as x is Adj we get

Speed

θθ

x

Cos θ \* Speed = x , θ is going to be value which is randomly chosen between 5 and 9. Thus we declare xVel a shown above.

Same Principle applies to y only its opposite to θ and as we still have to have the hyp involved we use Sine as Sin θ = Opp/Hyp as y is Opp we can rearrange this to Sin θ \* Hyp = Opp which is Sin θ \* Speed = y where θ is again a random value between 5 and 9. Thus we declare yVel as shown above.

y

To make the ball move on the screen we are going to have to update its location each time, this is going to be done by giving the ball picturebox a new point:

If BallShot 🡪 True Then

BallLocation 🡪 New Point(Ball.X + HorizontalSpeed, Ball.Y + VerticalSpeed)

End If

This code would make the ball move by adding the xVel to the balls x location and adding the yVel to the balls y location. Now we need to check for the top and bottom walls and making the ball rebound off them. This will be done by checking for the top and bottom wall, by comparing the location of the ball to the y axis value on the form. If the ball is at the top or bottom of the form we inverse the yvel, so the ball starts moving in the opposite direction. Now I need to make it so, if the ball hits either the left or right wall the correct person gets a point, so if the ball goes past the left wall the player gains a point and if it goes past the right wall the computer gains a point. We also have to return the ball to its original location and turn BallShot to False as well as display the updated scores to the user.

If Ball CollidesWith plyrWall Then

compScore += 1

BallShot 🡪 False

BallLocation 🡪 StartingLocation

End If

If Ball CollidesWith compWall Then

plyrScore += 1

BallShot 🡪 False

BallLocation 🡪 StartingLocation

End If

I am now going to work on the ball colliding with the player and computer paddle. By using the .NET Framework this becomes quite easy as I am using pictureboxes so to check for collision all I will have to do is put a .IntersectsWith() in the timer and insert either the computer or user paddle and declare the effects it has. If the ball collides with a paddle its xVel is inversed so the ball starts to move in the opposite direction. I am now going to check for the scores of the player and see if they have reached the 10 point milestone. If either the player or the computer reach 10 points the user is informed via message box, however if the player gets 10 points first they are congratulated through the message box. Before the message box appears we need to stop all of the timers so that everything stops moving and so we don’t get multiple message boxes appearing. If the computer wins after the user closes the message box the program should close.

***Making the Computer Paddle AI***

Now I am going to start coding the Computer paddle and its movement. To control weather the computer paddle is active or not I have declared another Boolean called AIActive. For a cooldown and reset I have also declared two variables AIChecker and AICoolDown:

Dim AIActive As Boolean = True 'Holds if the AI is on or not

Dim AIChecker As Integer = 400 'Counts down the time until the AI becomes active

Dim AICoolDown As Integer = 100 'AI Cooldown period

Now in a separate time which just controls the computer paddle, I am going to make it count down AIChecker by using:

AIChecker = AIChecker - 100

Then we need to randomly either enable the AI or turn it off. This will require a random number generator as a result I am going to use the in built one. I will call it by stating Randomize() in the Form Load Event which will call up the .NET Framework Randomize() function which can then be used to generate random numbers.

The random number will be used to produce a chance for the AI to either turn on or off. This will be related to the users score so if the user score is below 5 there is less chance for the AI to be on, but as the score increases so does the chance of the AI turning on. There wll also be a cooldown period after the AI has turned on. The cooldown period will start to decrease when the AI is off. If the AICoolDown goes below 0 we set its value to 0, we have to this for a check that will occur later on which will reset AIChecker. This check will only occur while AIActive is False and AICoolDown is 0, if the conditions are met the AI is enabled and the cool down period is reset. The AICoolDown period decreases in number when the AI is disabled and goes down by 50:

If AIActive 🡪 False Then

AICoolDownPeriod -= 50

End If

If AICoolDownPeriod < 0 Then

AICoolDownPeriod 🡪 0

End If

I am also going to add a pause menu to the game. I am going to this by adding two buttons which are going to be invisible until the correct key is pressed. Using the .NET Framework again and the KeyDown event we check if the user has pressed the key which leads to the pause menu, in my case I set this key to be Q. When Q is pressed all of the objects on the form are hidden using .Hide() and the buttons are made visible using Button.Visible = True. To resume the game the user presses W in my case which in essence undoes everything that Q did. (Full Code for Pong will be available in the Appendix).

**Making Flappy Bird**

Before I start developing Flappy Bird, I need to outline what I am going to do and how I am going to it, I did this by referring to my objectives and jotting notes down with ideas of how I am going to do this. First of all, I am going to add Flappy Bird and allow it to move.

***Making Bird***

To do this I added a picturebox to my form in which I then added an image of Flappy Bird. Then I have declared a variable called Yspeed which will hold the speed of Flappy Bird without regards to it moving up or down. I have also declared a variable called gravity which will hold the gravity speed. This will be added to YSpeed as it will increment it by 2 every time the timer ticks as a result the longer the bird is allowed to fall the faster it will fall. The variables declared so far are:

Dim Yspeed As Integer = 0 'Vertical Speed of pbBird

Dim Gravity As Integer = 2 'This Value Determines how quick pbBird Falls, the bigger it is gets the quicker it falls

The code to allow the bird to move down has to be placed inside a timer so the game is updated with this data:

Yspeed += Gravity 'Everytime Timer Ticks Yspeed increases by 2 so from 2 to 4 to 6 following a geometric progrssion

pbBird.Top += Yspeed 'Increase speed varaible by gravity and Top position of pbBird increases by whatever the speed variable is, starting at 0

So far the Bird is only allowed to fall down, to counter this player in the original is allowed to make the ball jump up. We are going to this every time the space key is pressed so by using the .NET Framework we use the KeyDown Event. Every time the Space Key is pressed the Yspeed is decremented by -15:

If e.KeyCode = Keys.Space Then 'When Space is pressed the following code takes place

Yspeed = -15 'If pbBird is falling down speed will be positive, if pbBird is going up speed has to be negative for pbBird to go up

End If

As the codes comment states if the bird is falling down Yspeed will be positive and if it’s negative the bird will go up. Now we have a bird that can move up and down, in my version of Flappy Bird if the bird touched the top of the form or the bottom of the form the player lost.

***Collision With Form Boundary***

To do this I am going to add two pictureboxes at the top and the bottom of the form and using the .NET Framwork with the bounds and IntersectsWith to make this collision. When this collision occurs the game ends and the player loses.

If Bird CollidesWith TopWall Then

Timer 🡪 False

MessageBox(“Game Over, Your Score is (plyrScore)”)

ElseIf Bird CollidesWith BottomWall Then

Timer 🡪 False

MessageBox(“Game Over, Your Score is (plyrScore)”)

End If

As the pseudocode shows, when the collision is detected the timer is stopped and a message box appears which shows the user their score. Now we have a game where the bird moves up and down and it dies if the bird touches the top or the bottom of the form.

***Making Pipes***

I am now going to create the Pipes the Bird has to pass through Now instead of creating pictureboxes on the form, I am going to create them dynamically. To start of I am going to create an Array to limit the number of pipes to a static amount so only a certain amount of pipes are ever made:

Dim Pipe(2) As PictureBox 'Acts like an Array with the number of Pipes inside the Parameter

Dim TopPipe(2) As PictureBox 'Acts Like an Array with the Number of Pipes inside the Parameter

Also as the Pipes have a constant sized gap between them I declared another variable to hold this value:

Dim GapBetweenPipes As Integer = 475

Now to dynamically create the Pipes I have created two sub routines CreatePipes and CreateTopPipes with the only difference being the location of the pipe as one is at the top and one is at the bottom. When creating the pipes dynamically I declare a variable as a new PictureBox which I then add to the Form using Me.Add(). For my Pipes, I gave them the dimension 50 x 350 and made them green to resemble the pipes in the actual game. For my Bottom Pipe, I gave it a value for the .Top value that would be randomly chosen between the range 70 - 350. For the Top Pipe what I did was find the pipe being generated between with that pipe at the bottom and took away the value for GapBetweenPipes. To find the Bottom Pipe being generated alongside the Top Pipe what I did was use the Array I created previously where I added the Pipe to the array once I made generated it. Now I have a bird that is moving up and down and collisions with the top and bottom wall, as well as pipes being generated.

Now the only thing I have left is to check for collisions with the pipes as well as making the pipes move towards the bird. To do this I made a variable called Pipe Speed which will contain the speed the pipes will move at:

Dim PipeSpeed As Single = 3.5 'Variable controls how fast the Pipes Move

Now to make all the pipes move unanimously I used a FOR loop to loop through all the pipes in the array at that index. The pipes need to be moving left, this is done by decrementing the .Left of the pipe at that positon by the Pipe speed:

For i = 0 To 1 'This For Loop allows the Pipes to move to the Left Hand side of the Screen

Pipe(i).Left -= PipeSpeed

TopPipe(i).Left -= PipeSpeed

Now to allow the user to gain a point every time they move past a pipe, what I did was produce a check for a single pipe instead of both pipes as both pipes should be moving together. So what I did was check if the Pipe had gone of the form, if it had then the Bird must have gone through the Pipes, this results in the user gaining a point to their score. Then I have to move the generated pipes back to the other side of the form this will be done by incrementing there .Left by a really large amount. (See Flappy Bird Pipe Flowchart in Appendix).

Finally to make it so a random sized pipe is given each time with a constant gap I implemented the following code inside the check to see if the pipes have passed the bird or not:

Pipe(i).Top = r.Next(125, 350) 'Makes sure a Random Size is given each time

TopPipe(i).Top = Pipe(i).Top - GapBetweenPipes 'This makes sure their is a gap of a constant size between the top pipe and bottom pipe

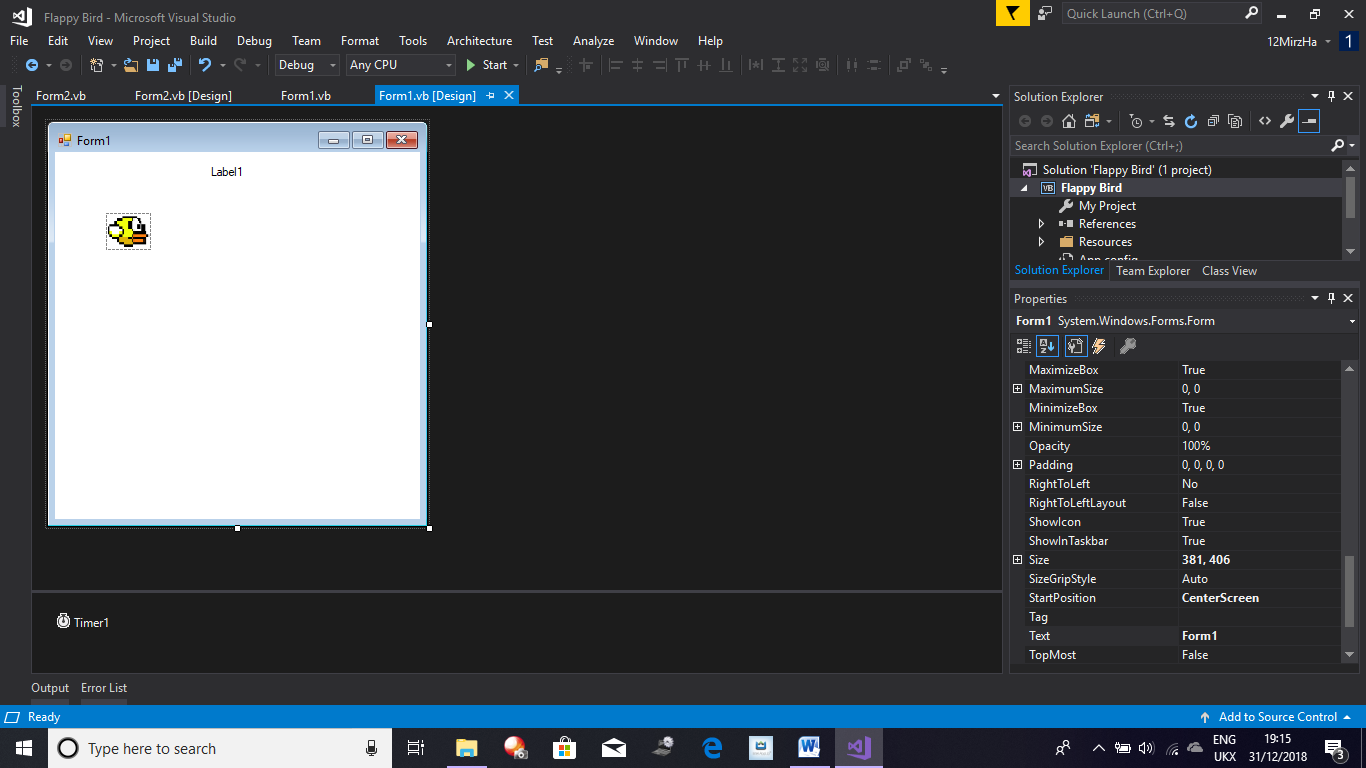
The code above shows that the Pipe(i) is given a random number for its distance from the top of the form this results in the TopPipe at i also being given a random size as the TopPipe always keeps the constant Gap between itself and the Pipe at the bottom. Now I have pipes moving to the left and a bird which cannot collide with the top or bottom of the form, but can move up and down. The only thing left to do is to make the collision detection with the Bird and the Pipes. This will be done using the .NET Framework and a FOR Loop I can check for a collision with every pipe using .Bounds and .IntersectsWith(). If the Bird does collide with a pipe (so if they overlap each other even a bit) the code above will detect this and cause the timer to be stopped so all the links to the timer stop and it will also cause a message box to appear which will display the user score to the user.

If Bird CollidesWith Pipe Then

Timer -> False

MessageBox(“Game Over Your Score Is”)

End If

Now I have a fully functioning flappy bird game, all I need to add is a label to display the score to the user when the game is running as this will provide user accessibility. To do this I added a label to my form so my form now looks like this:

To display the score of the User to the User all I need to do is transfer it to the label. This will be done inside the timer by Converting the score to a string and adding it to the label with

Label1.text = (score).

Now I have a fully functioning flappy bird game, all I need to do now is to add its random mode which I will add once I have made all my other base games.

Design Figure 12: Flappy Bird Form Layout

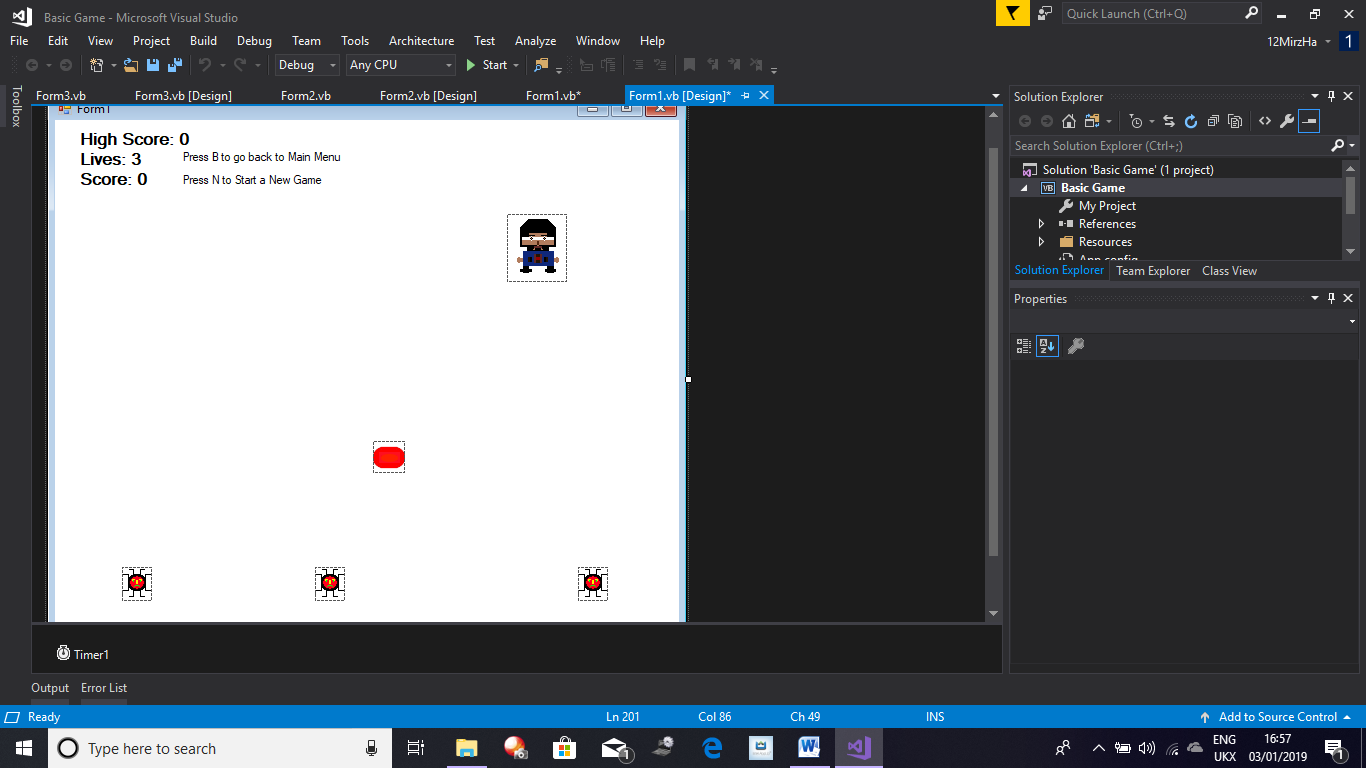
After making Flappy Bird and Pong I decided to make Bling Bling Boy (My Game). (see next page)

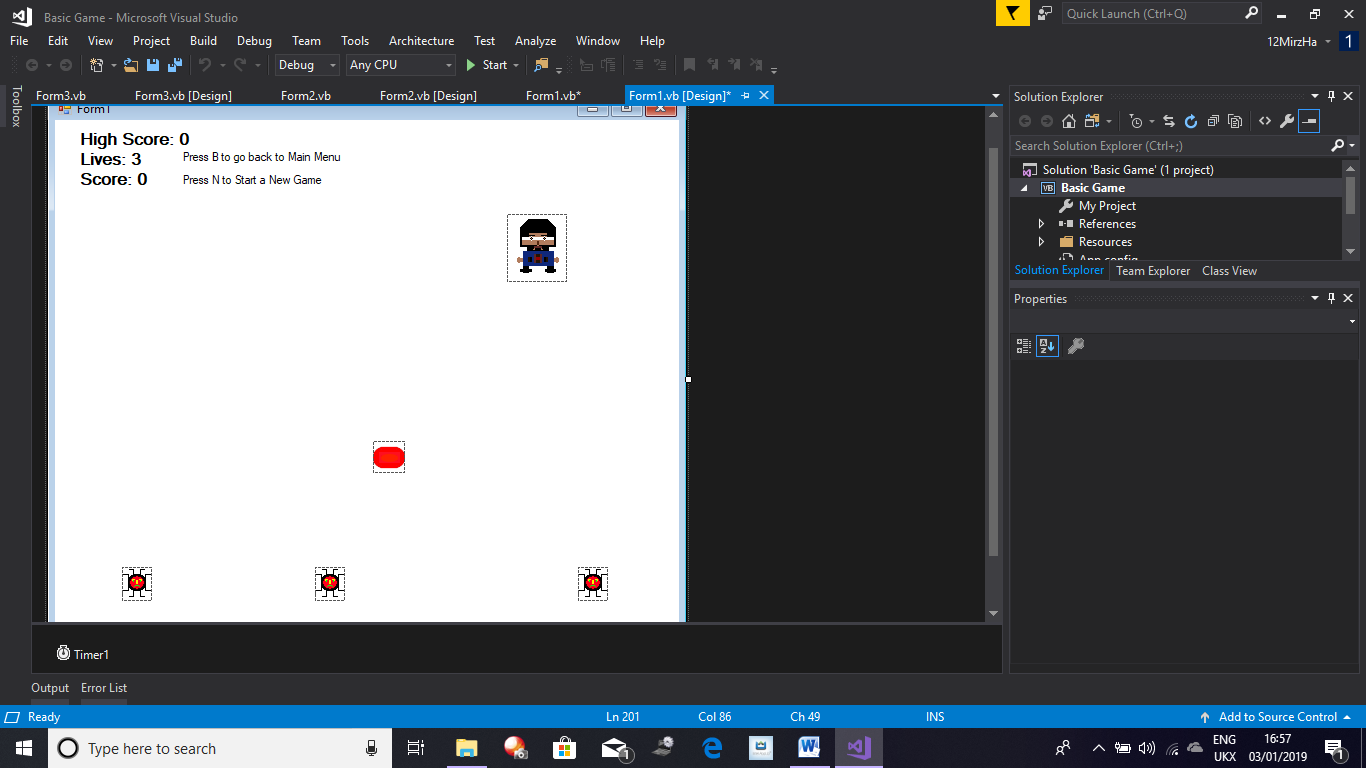
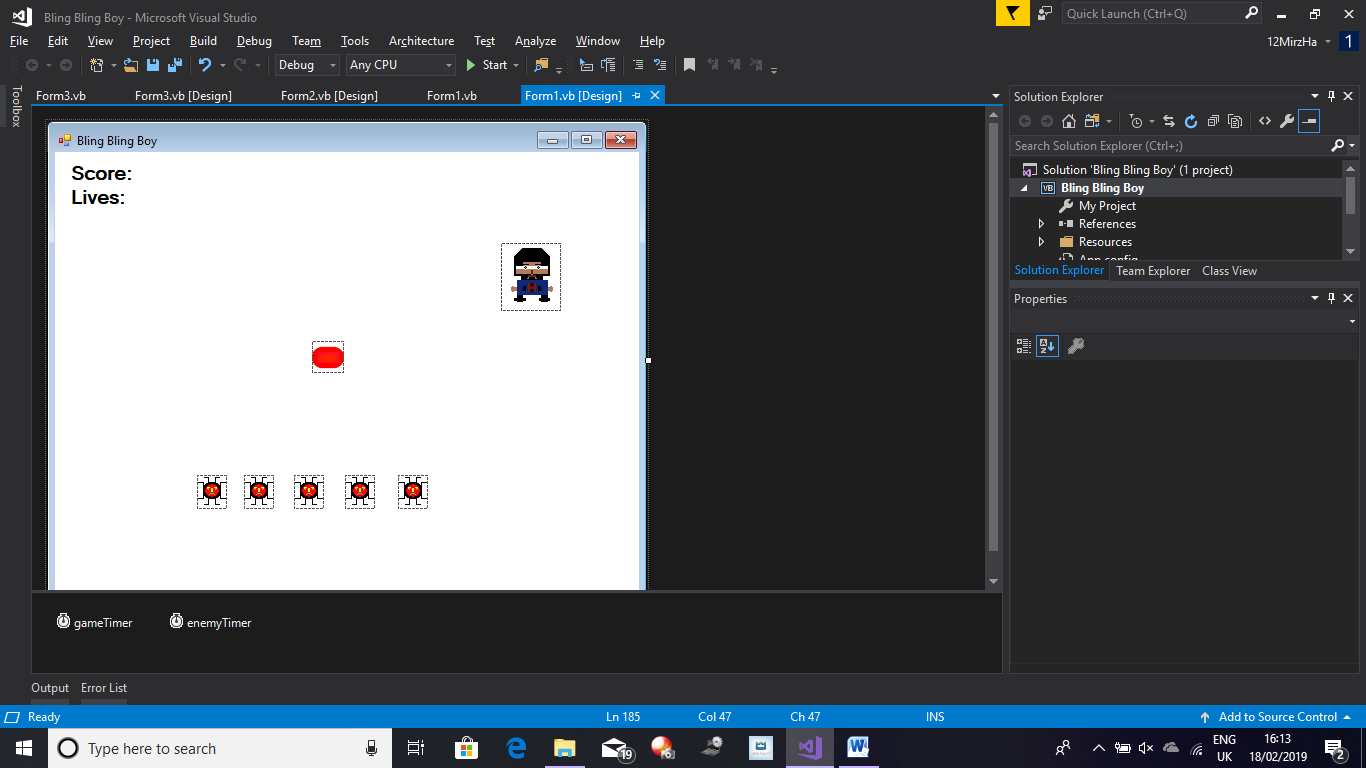
**Making Bling Bling Boy**

First of all I made all the sprites that I wanted, the game only needs two sprites for the player, one sprite for the gem and one sprite for multiple enemies in total I made 3 sprites. I used Paint to make my sprites which can be seen below:

E:\College Subjects\Computer Science\Computer Science NEA\My Games\Basic Game - DataBase\Gem.pngE:\College Subjects\Computer Science\Computer Science NEA\My Games\Basic Game - DataBase\Enemy Sprite.png

Design Figure 13: Bling Bling Boy Sprites

First of all I am going to set up my form with 7 pictureboxes, one will hold the players sprite, and one will hold the gem which will be moved around randomly when the player interacts with it. The other pictureboxes will hold the enemy sprites which will chase the player. I am now going to add 2 labels. These labels will display the Score and number of lives to the player. I am also going to add 2 Timers as I need to program some events which will need to be updated constantly for instance enemy movement.



Design Figure 14: Bling Bling Boy Form Layout

***Making the Player***

Now that my form is set up, I am now going to make my Player Sprite move with the arrow keys. To do this I am going to declare 4 variables each representing a direction where the Player can move. Alongside, that I am also going to declare the variable that will hold the players speed.

Dim MoveUp As Boolean = False 'This Variable is used to allow the player to move their character up

Dim MoveDown As Boolean = False 'This Variable is used to allow the player to move their character down

Dim MoveLeft As Boolean = False 'This Variable is used to allow the player to move their character left

Dim MoveRight As Boolean = False 'This Variable is used to allow the player to move their character right

Dim PlayerSpeed As Integer = 10 'This Variable holds the speed the player can move at

Now I am going to use the .NET Framework to allow sprite to move. Using the KeyDown event from the Form Events, I am going to enable all of these when the corresponding key is pressed. Then using the KeyUp event I will disable all of the corresponding variables when the correct key is released.

If LeftKeyDown 🡪 True and MoveLeft 🡪 True Then

PlayerMovesLeft

ElseIf RightKeyDown 🡪 True and MoveRight 🡪 True Then

PlayerMovesRight

ElseIf UpKeyDown 🡪 True and MoveUp 🡪 True Then

PlayerMovesUp

ElseIf DownKeyDown 🡪 True and MoveDown 🡪 True Then

PlayerMovesDown

End If

I have converted the Pseudo-Code above to actual code (which can be seen below). I am now going to add following code to the timer which will use the variables that I made with the KeyDown and KeyUp to make the player move:

If MoveUp = True Then 'If MoveUp is True then the following code is ran

pbBoy.Top -= PlayerSpeed 'The player moves up

ElseIf MoveDown = True Then 'If MoveDown is True then the following code is ran

pbBoy.Top += PlayerSpeed 'The player moves down

ElseIf MoveLeft = True Then 'If MoveLeft is True then the following code is ran

pbBoy.Left -= PlayerSpeed 'The player moves left

ElseIf MoveRight = True Then 'If MoveRight is True then the following code is ran

pbBoy.Left += PlayerSpeed 'The player moves right

End If

Currently the Player can move anywhere in the Form even out of bounds, to prevent this I am going to add boundarys to limit the players movement:

If Player CollidesWith HorizontalBoundarys Then

PlayerLocation 🡪 AtBoundary

ElseIf Player CollidesWith VerticalBoundary Then

PlayerLocation 🡪 AtBoundary

End If

***Making Gem***

The player can now move around the form, so I now need to allow it to interact with the Gem, as well as make the Gem move to a random location when the player interacts with it, this should cause the score to increase.

If Player CollidesWith Gem Then

GemLocation 🡪 RandomLocation

plyrScore += 1

End If

I am going to check if the player and the gem are touching/overlapping. If they are I want the score to increase by one as well as move the gem to a random location on the form:

If pbBoy.Bounds.IntersectsWith(pbGem.Bounds) Then 'If the player collects a Gem then the following code is ran

Score += 1 'The player gains 1 point to their Score

pbGem.Visible = False 'The Gem temporarly becomes invisible

pbGem.Location = AssignRandomLocation(pbGem.Location) 'The Gem gets a new Random Location

pbGem.Visible = True 'The Gem becomes Visible again

End If

The code above checks if the player and the gem are in contact if they are then the players score is increased by 1 and the gem is moved to a random location. I now have a player controlled sprite as well a gem which moves to a random location when it collides with the player. The gem also increases the players score. I now have to make the enemies chase the player.

***Making the Enemies***

To do this I am going to need to track the player’s location and move all the enemies to that location. As the player won’t stay idle, the tracking code would need to be consistently updated with the player’s location as a result it would either need to be made in the timer or called from a sub routine inside the timer. The Pseudocode for this can be seen below:

If Enemy.Distance.X > Player.Distance.X Then

EnemyMove.x 🡪 -EnemySpeed

Else

EnemyMove.X 🡪 EnemySpeed

End If

If Enemy.Distance.Y > Player.Distance.Y Then

EnemyMove.Y 🡪 -EnemySpeed

Else

EnemyMove.Y 🡪 EnemySpeed

End If

My Pseudocode above states that if the Enemy is further down the X-Axis then the player then the enemy is to the right of the player and thus needs to move left to get to the player, with the Else it allows it to check if the Enemy is to the left of the player and this move right to go get to the player. The same principles are used vertically as well. The adaption of this in actual code can be seen below:

For i = 1 To CurrentNumberOfEnemies 'This FOR loop is used to loop through all the numbers up to the integer stored inside CurrentNumberOfEnemies

If pbBoy.Location.X > EnemyArray(i).Location.X Then 'If the player is to the right of the enemy then the following code is ran

EnemyArray(i).Left += EnemySpeed 'The enemy moves to the right

Else 'If the intial requirement isn't fulfilled then the following code is ran

EnemyArray(i).Left -= EnemySpeed 'The enemy moves to the left

End If

If pbBoy.Location.Y > EnemyArray(i).Location.Y Then 'If the player is below the player then the following code is ran

EnemyArray(i).Top += EnemySpeed 'The enemy moves down

Else 'If the inital requirement isn't fulfilled then the following code is ran

EnemyArray(i).Top -= EnemySpeed 'The enemy moves up

End If

Next

In my objectives, I have stated that after a certain amount of points, the number of enemies should increase, to do this I am going to track the current number of enemies and increase this number as the game progresses. Also as I am going to have a set number of enemies I can store them all in a array of a fixed size and affect them all through the use of the Array.

Dim CurrentNumberOfEnemies As Integer = 1 'This Variable holds the number of enemies on the form when the game is running

Dim EnemyArray(5) As PictureBox 'Stores all the Enemys in a Array

Initally I will have all of the enemies stored inside the array. As this will have to be done before the game beigns, this code will need placing inside the Form Load event and should be the first thing inside it. To activily increase the number of enemies in the game, I will need to continuously monitor the score, this will be done with one of the Timers made prior. With all this set, I will then require a new enemy to appear when the player reaches a score milestone which will be every 5 points. This will be done by incrementing CurrrentNumberOfEnemies then making the enemy at that point in the array visible.

If plyrScore 🡪 5 Then

CurrentNumberOfEnemies 🡪 2

EnemyArray(CurrentNumberOfEnemies) 🡪 BecomeVisible

ElseIf plyrScore 🡪 10 Then

CurrenNumberOfEnemies 🡪 3

EnemyArray(CurrentNumberOfEnemies) 🡪 BecomeVisible

ElseIf plyrScore 🡪 15 Then

CurrentNumberOfEnemies 🡪 4

EnemyArray(CurrentNumberOfEnemies) 🡪 BecomeVisible

ElseIf plyrScore 🡪 20 Then

CurrentNumberOfEnemies 🡪 5

EnemyArray(CurrentNumberOfEnemies) 🡪 BecomeVisible

End If

The Pseudocode above checks the score of the player and increases the number of enemies appropritatly all the way up to the maximum number of enemies available which is 5. CurrentNumberOfEnemies is initially 1 so when the first milestone is reached it’s set to 2.

I have also learned that when the game is run, all the enemy’s merge together when they chase the player as they all follow the players location thus coming together as there will eventually come a point where the overall displacement is reduced between the player and enemies when they all merge together. To counter this I have made a some code inside the enemyTimer which makes the enemies bounce off each other:

If Enemy1 CollidesWith OtherEnemies Then

Enemy1 🡪MoveAway

ElseIf Enemy2 CollidesWith OtherEnemies Then

Enemy2 🡪 MoveAway

ElseIf Enemy3 CollidesWith OtherEnemies Then

Enemy3 🡪 MoveAway

ElseIf Enemy4 CollidesWith OtherEnemies Then

Enemy4 🡪 MoveAway

End If

For j = 2 To 5 'FOR loop goes through values from 2 to 5

If pbEnemy1.Bounds.IntersectsWith(EnemyArray(j).Bounds) Then 'Checks if enemy 1 has collided with any of the other enemies

pbEnemy1.Left += 5 'If a collision has occured enemy 1 moves away

End If

Next

For k = 3 To 5 'FOR loop goes through values from 3 to 5

If pbEnemy2.Bounds.IntersectsWith(EnemyArray(k).Bounds) Then 'Checks if enemy 2 has collided with any of the other enemies

pbEnemy2.Left -= 5 'If a collision has occured enemy 2 moves away

End If

Next

For l = 4 To 5 'FOR loop goes through values from 4 to 5

If pbEnemy3.Bounds.IntersectsWith(EnemyArray(l).Bounds) Then 'Checks if enemy 3 has collided with any of the other enemies

pbEnemy3.Left += 5 'If a collision has occured enemy 3 moves away

End If

Next

If pbEnemy4.Bounds.IntersectsWith(pbEnemy5.Bounds) Then 'Checks to see if enemy 4 and enemy 5 have collided

pbEnemy4.Left -= 5 'If a collison has occured then enemy 4 moves away

End If

Basically, the code above checks if the two enemy’s have collided, if they have the enemy being checked is move away from all the other. This movement pattern then occurs in an alternating way with the next enemy being checked moving in the opposite direction to the enemy being checked before it. This pattern means that all enemy’s will at one point all be away from each other and will not align over each other.

Right now I have a gem which can interact with the player and provide some points and move to a random location. I also have a movable player which the player can control, as well as enemy’s which can chase the player.

***Enemy and Player Interaction***

Now I just need to make it so the player when hit by an enemy the player loses 1 of 3 lives and all the enemy’s and player position are reset. To do this all I am going to is use the Collision Function with an IF statement like below:

If Player CollidesWith Enemys Then

plyrLive -= 1

PlayerLocation 🡪 StartingPosition

EnemysLocation 🡪 RandomLocation

End If

If plyrLive = 0 Then

TimerStop 🡪 True

MessageBox(“Game Over Your Score is (Score)”)

End If

The Life and Score variable are both integer variable are declared globally as seen below:

Dim Lives As Integer = 3 'This Variable stores the number of Lives the player has

Dim Score As Integer = 0 'This Variable holds the players actual Score

When the player dies, I want all the enemy’s to move to random location hence I have created a function which outputs a random location on the form this same function is also used to assign the gems a random location as well:

Private Function AssignRandomLocation(ByRef pbLocation As Point) 'This code is used to give a pb a random location

pbLocation = New Point(RandomX.Next(1, 550), RandomY.Next(-3, 413)) 'Used to Assign a pb a Random Location on the Form

Return pbLocation 'Outputs this new Location

End Function

This function will be used to assign the gem a random location and the enemy a random location when required.

I am now going to implement invincibility into the game through an integer variable called InvincibilityTimer. The invincibility should run every 5 points (up to the limit) and at the start of the game for a few seconds. To do this I have assigned a check before the collision for ready so a collision can only occur when ready is below 0:

Dim InvincibilityTimer As Integer = 1000 'This Variable is used to give the player temporary invincibility

If InvincibiltyTimer <= 0 Then

CollisionCodeActive 🡪 True

Else

CollisionCodeActive 🡪 False

End If

When the InvincibilityTimer is active the player sprite should be changed so the player knows that they are currently invincible. This will done through a similar check with an IF statement.

The InvincibilityTimer will need counting down, this will be done with

InvincibilityTimer -= 10

Everytime the player collides with an enemy, the InvincibiltyTimer variable should reset to 1000 as well as everytime the number of enemies increases. This should allow the player to temporarily become invincible, however to counter the game becoming too easy, I have also made the enemies get faster everytime the number of enemies is increased. There is also a limit to the number of times the player can become invincible.

In this game I am also going to display the best user to the player in a label on the screen. My logic to do this, is to save the score and username of the best player or the first player to a text file which is then read and compared to the current score, if the current score is greater than the high score then the old users data is removed from the text file and the new users data is written. To do this I would use the .NET Framework File commands. The code below shows how I implanted this:

When the Form Loads I want the following to take place (in form of pseudocode):

Temp 🡪 ReadCurrentHighScore

CurrentHighScore 🡪 ConverToInteger Temp

If User beats current High Score:

If Score > CurrentHighScore

Write Score ToFile

End If

When the Timer ticks the following code is run, this code is only ran when the game ends (Adaption of Pseudo-Code):

If Lives = 0 Then 'When the player runs out of Lives then the following code is ran

gameTimer.Stop() 'The gameTimer is stopped so the events in this timer no longer take place

enemyTimer.Stop() 'The enemyTimer is stopped so the events in this timer no longer take place

If Score > HighScore Then

Dim UserInput As String = InputBox("What is Your UserName?") 'Holds the users username

Using Writer As IO.StreamWriter = New IO.StreamWriter("BestPlayer.txt") 'Used to write to stated file

Writer.WriteLine(UserInput) 'Writes the users username

Writer.WriteLine(Score) 'Writes the users score

End Using

MessageBox.Show("Well Done You Have the Current High Score") 'Informs user that they have beaten the current high score

End If

MessageBox.Show("Game Over Your Score is " & Score) 'A MessageBox appears which tells the user their final score

frmScoreMenu.CurrentUserScore = Score 'The users Score is transferred from this form to the score menu

Me.Hide() 'This form is closed

frmScoreMenu.Show() 'The score menu form is shown to the user

End If

I now have a fully working game called Bling Bling Boy, for this game all I have to do is to develop the high score screen which utilises merge sort and binary search on all of the scores that get saved to the text file which acts as a flat file database.

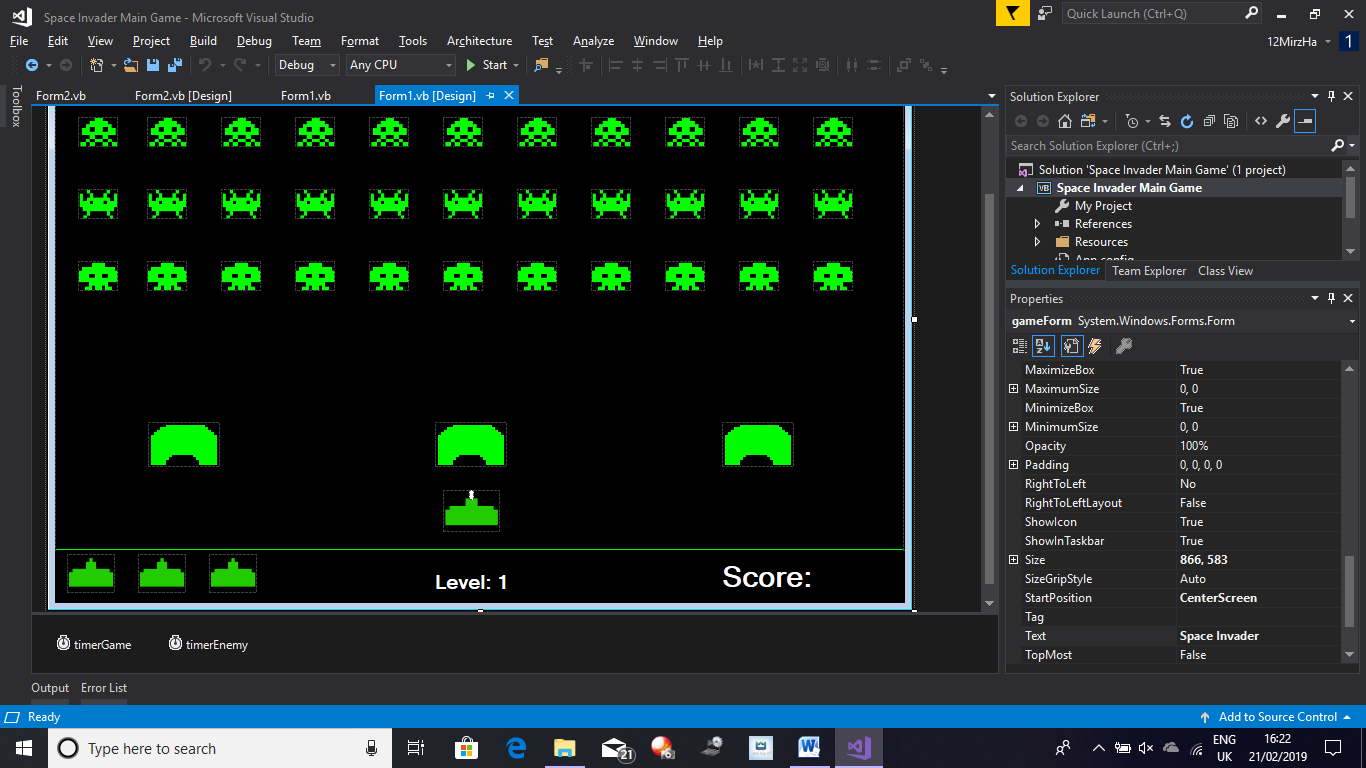
**Making Space Invaders**

To make the base game of Space Invaders, I first set up my form with a black background to match it to its original counterpart. I then added 3 rows of enemies with 11 pictureboxes these boxes will contain the 3 different sprites for the different enemies and each different enemy corresponds to a different amount of points being added to the players score when the player kills an enemy.



Design Figure 14:

Space Invader Sprites

The sprites above are all the sprites I am going to use for making Space Invaders. After I make the rows of enemies, I am going to make a label which I will place just above the bottom of the form. Inside the label I typed a line of “\_\_” which I used to act as a separator between the actual gameplay and the User information (so lives etc.). Above this line I am going to place the player sprite, this is the sprite the player can shoot with and move left and right with in the centre of the ship I added my bullet. Below the line, I am going to place 3 more player sprites in pictureboxes which would be smaller than the actual player sprite. These should all be in a line and are used to indicate to the player the number of lives they have remaining. In the centre of the line I placed a label which would tell the user what level they are on. Finally on the far right I am going to place a label which will indicate to the player there score, as well as this I am also going to drag on two timers one to update all game events and one to manage the enemies. My form layout can be seen below:

Design Figure 15:

Space Invader Form Layout

Now that I have my form setup, I am now going to start programming first of all I am going to make the player moveable.

***Making the Player Ship***

Following the same method as my other games, I am going to make two variables called MoveLeft and MoveRight and as their name suggests they allow the player to move left and right:

Dim MoveRight As Boolean = False 'Used to tell if the player can move right or not

Dim MoveLeft As Boolean = False 'Used to tell if the player can move left or not

Dim PlayerSpeed As Integer = 4 'Holds the Players Speed

I also made a PlayerSpeed Variable to hold the speed the player moves at. Now to change the value of MoveRight and MoveLeft I used the KeyDown Event and the KeyUp event, so when the left and right arrow keys are pressed the Boolean value for these variables turns to True and when the Keys are released they return to false. Now to limit the player’s movement so they cannot go of the screen I tracked the player’s location and placed picturebox boundaries where the player cannot move thus only allowing the player to move inside the area I want them to move. This is going to to be done with the .NET Framework and the .Bounds and .IntersectsWith commands. If the player does reach inside the boundary they are placed inside the boundary thus not allowing the player to move of the form, the code I implemented can be seen below:

If pbPlayer.Bounds.IntersectsWith(pbLeftWall.Bounds) Then 'If the player reaches the boundary of the form the following code is ran

pbPlayer.Location = New Point(0, 431) 'The player is placed at the boundary of the form

End If

If pbPlayer.Bounds.IntersectsWith(pbRightWall.Bounds) Then 'If the player reaches the boundary of the form then the following code is ran

pbPlayer.Location = New Point(794, 431) 'The player is placed at the boundary

End If

Now I am going to start making the enemies move.

***Making Enemies***

The enemies movement pattern in the original Space Invaders was that they moved left all the way then down and then changed to move right all the way then down. Following this pattern the enemies would slowly strafe down. I am going to mimic this. First of all I need all of the enemy’s on the form to be added to an array so all the enemies can be coded simultaneously, to do this I assigned each row of enemy’s a different tag. I then created a function called InitaliseEnemys() which would be called when the form loads and it will add all of the enemies on the form to an array. The Pseudocode for this can be seen below.

For Each PictureBox on Form

If PictureBox.Tag = AEnemyTag Then

PictureBox 🡪 EnemyArray()

End If

Next

I also want to store the location of each enemy so they can be moved back each time the level changes, to do this I made another array called EnemyLocation which would store the location of each enemy.

Dim EnemyArray(32) As PictureBox 'Holds the enemys in an array

Dim EnemyLocation(32) As System.Drawing.Point 'Keeps track of the location of each enemy in an array

These arrays have a static size of 32 as the number of enemies will not be changing so there is no need to create a dynamic array. To reduce the clutter inside the enemy timer handler, I decided to create another sub routine which would allow the enemies to move in the same manner as discussed above. I am going to take one of the 3 enemies (vertically) on the far left hand side and one of the 3 enemies (vertically) on the far right hand side. In total these two enemies are going to act as indicators for the entire number of enemy’s on the form. So when the far left enemy reaches the furthest left point on the form, all of the enemies should move down a set number of pixels and then start moving right, until the far right enemy reaches the furthest right point on the form, all the enemies should move down. This should consistently repeat until the enemies have reached the player: My Flowchart illustration of this can be seen in the Appendix labelled “Space Invaders Enemy Movement Flowchart”. My Implementation can be seen below:

Private Sub EnemyMoveMent()

'Each Enemy Moves left or right until they reach the boundary then they move down

For i = 0 To 32

EnemyArray(i).Left = EnemyArray(i).Left + EnemySpeed

Next

If PictureBox13.Bounds.IntersectsWith(pbRightWall.Bounds) Then

EnemySpeed = EnemySpeed \* -1

For i = 0 To 32

EnemyArray(i).Top += 10

Next

End If

If PictureBox2.Bounds.IntersectsWith(pbLeftWall.Bounds) Then

EnemySpeed = EnemySpeed \* -1

For i = 0 To 32

EnemyArray(i).Top += 10

Next

End If

End Sub

The sub routine above demonstrates how I programmed the enemy movement in. This sub routine needs to be called inside the enemy timer. Now to make the enemies shoot down, firstly I need a missile for each enemy. To do this I created I them dynamically instead of using more pictureboxes. The method remains the same from the pipe generators from Flappy Bird. I also needed an array to hold all of these shots so I created another array called EnemyShot() as well as another array which would hold the Boolean values for each shot to check if they have been shot or not called EnemyShot2():

Dim EnemyShot(32) As PictureBox 'Keeps track of the bullet of each enemy

Dim EnemyShot2(32) As Boolean 'Used to tell if an enemy has shot or not

The sub routine to create each missile can be seen below:

Private Sub EnemyShoot(ByVal Number As Integer)

Dim i As Integer 'Used for loops

'Code Below dynamically creates each missile when one is shot by the enemy

For i = 0 To Number

Dim EnemyMissile As New PictureBox

Me.Controls.Add(EnemyMissile)

EnemyMissile.Height = 10

EnemyMissile.Width = 5

EnemyMissile.BorderStyle = BorderStyle.FixedSingle

EnemyMissile.BackColor = Color.GhostWhite

EnemyMissile.Top = EnemyArray(i).Top + EnemyArray(i).Height / 2 - EnemyMissile.Height / 2

EnemyMissile.Left = EnemyArray(i).Left + EnemyArray(i).Width / 2 - EnemyMissile.Width / 2

EnemyMissile.BringToFront()

EnemyShot(i) = EnemyMissile

EnemyShot(i).Visible = False

Next

End Sub

The sub routine above uses a FOR loop to loop through the number off values declared inside the parameters when the routine is called. This will be 32 and will be called inside the Form Load Event. All the missiles made are placed in the centre of each enemy only becoming visible when they are shot as when they are all made they are initially hidden due to .visible = false. Now to make the shots move, I am going to create another sub routine called EnemyShotMovement. This should select a random missile and fire it horizontally down. As I am going to use the .NET Framework random number generator, I am going to call the Randomize function in the form load. I am going to loop through the EnemyShot2 array and check for false values (the values will be false if the missile in that location hasn’t been fired). If the value at i has a false value I am going to use the random number generator to change the value of an integer and then check to see if that integer value matches up with the value required for a shot to be fired. If it does then the value at i changes from false to true. Then in another loop I am going to check to see if the value at i in EnemyShot2 array is true if it is then the missile should become visible and move downwards, then I would check for a collision and remove the missile and place it back in its original location. Inside the Collision IF statement I should also be checking the number of lives and making the appropriate changes to the screen to allow the user to see if they have lost a life or not, currently I haven’t added the Life system in but this is where I would add it. The Collision will be done using the .NET Framework with Bounds and IntersectsWith() commands. Now I have a Player who can move and enemies who can also move but also shoot down.

***Making Player Shoot***

I am now going to allow the player to shoot this will be done by making a missileon the form and as we should only be able to shoot 1 missile there should only be one missile. I am also going to require a Boolean variable which will be used by to tell the computer if the missile has been shot or not:

Dim MissleShoot As Boolean = False 'Tells the Computer if the Missile is allowed to shoot or not

The player missile is a picturebox placed at the centre of the players ship. The Missile will be fired by pressing the space bar. When the space bar is pressed the missile should be released from the centre of the ship and move upwards. If it collides with any of the objects it’s supposed to be able to hit the missile should disappear and the object should react accordingly. To make the actual Missile move up we have to check if the missile has been fired, this will have to be done with a Boolean Variable which will tell the program if the missile has been fired or not. If it has been fired then the Missile should become visibile and move up. If this created Boolean Variable is false then the Missile should not be visible. If the missile hits a collidable object then the missile should disappear as well and this Boolean Variable should also become false.

If SpaceBarPressed 🡪 True Then

MissileIsAllowedToShoot 🡪 True

End If

If MissileIsAllowedToShoot 🡪 True Then

MissileMovesUp

End If

If Missile CollidesWith Object Then

ObjectReactAccordingly 🡪 True

MissileIsAllowedToShoot 🡪 False

Missile 🡪 BecomeInvisibile

End If

***Score and Leveling***

Now I am going to make the scoring and the levelling system of Space Invaders. Below are the Variables that I have declared for this:

Dim Score As Integer = 0 'Holds the Score of the Player

Dim Lives As Integer = 3 'Holds the Lives of the User

Dim Complete As Integer = 0 'Holds the number of enemys killed

Dim Level As Integer = 1 'Holds the level of the player

Now that I have my Lives variable I can make the Life check inside the function which allows the enemies to shoot. In Space Invaders different enemies provide a different number of points. As I have already tagged all the enemies before when I was making the enemies move, so I can just reuse them as each row should have a common tag with its horizontal neighbour and an alternate tag to its vertical neighbour. By checking these tags I can assign a different amount of points to the score for the different enemies:

'Checks to see which typeof enemy was killed and adds the correct amount of points to the score

If EnemyArray(j).Tag = ("Enemy2") Then

Score += 20

End If

If EnemyArray(j).Tag = ("Enemy3") Then

Score += 40

End If

If EnemyArray(j).Tag = ("Enemy1") Then

Score += 10

End If

I also have to increment Complete as I just killed an Enemy, as well as make the missile involved in the collision disappear. Inside the score label I should also display the score:

lblScore.Text = ("Score: " & Score) 'Displays the Score in the Label to show the User what their score is

I also have to check Complete against the total number of enemies which is 33. Instead of writing the whole of this code in the single IF statement I produced a subroutine which would be called at the end of the collision IF statement. Inside this sub routine I will compare complete against 33 using an IF statement, If the IF statement is fulfilled then Level is incremented and the level label should display the new level as well as reset Complete back to 0. I also have to reset all of the enemy positions and make them all visible again. This concludes the game timer.

***Enemy and Boundary Collision***

Now to finish the Enemy timer, I need to add a check for when the enemies cross/reach the bottom of the screen. When this happens instead of losing one life the player instantly dies. To do this I used the position of my label and the .Bounds and .IntersectsWith() commands to determine when the player has reached the bottom of the form. Here the game would end and all the enemies would be reset to their original locations and all the timers would be stopped and a message box would appear which would tell the player their score. Also inside the Enemy Timer I will also check the barricades life and image. The pseudocode for the barricades can be seen below:

Pseudocode:

If AnyMissile CollidesWIth Barricade Then

AnyMissileBecomesInvisible 🡪 True

BarricadeLife -= 1

End If

If BarricadeLife 🡪 2 Then

BarricadeImage 🡪 Image2

ElseIf BarricadeLife 🡪 1 Then

BarricadeImage 🡪 Image1

ElseIf BarricadeLife 🡪 0 Then

BarriaceBecomesInvisible 🡪 True

End If

The Pseudocode above outlines the basic function of the barricade and what they should do, my implementation of my pseudocode can be seen below:

Private Sub Barricade()

Dim i As Integer 'Used for Loops

Dim j As Integer 'Used for Loops

Dim BarricadeArray(2) As PictureBox 'Stores all the Barricades in an Array

'The Code below loops through each picturebox in the form and adds it to the array if it has the barricade tag

For i = 0 To 2

For Each Picturebox In Me.Controls

If Picturebox.Tag = ("Barricade") Then

BarricadeArray(i) = Picturebox

i += 1

ReDim Preserve BarricadeArray(i)

End If

Next

Next

'Nested FOR Loop checks to see if the player shot the barricade, if they did then the barricade loses a life and the image of the barricade changes accordingly

For j = 0 To 2

For i = 0 To 1

If BarricadeLife(j) > 0 AndAlso BarricadeArray(j).Visible = True Then

If (pbMissile.Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

pbMissile.Visible = False

MissleShoot = False

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

BarricadeLife(j) = BarricadeLife(j) - 1

End If

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

Next

Next

'Nested FOR Loops checks to see if a Enemy Missile has hit the barricade, this causes the barricade to lose one life and the barricade image changes accordingly

For j = 0 To 2

For i = 0 To 32

If (EnemyShot(i).Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

Me.Controls.Remove(EnemyShot(i))

EnemyShot(i).Location = EnemyArray(i).Location

BarricadeLife(j) = BarricadeLife(j) - 0.25

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

End If

Next

Next

End Sub

Initially, I tagged all of my barricade with the barricade tag, then using a FOR EACH loop I added them all to an array. The variables I added in were:

Dim BarricadeLife(2) As Single 'Used to Hold the Barricade Lives

This is a global variable and when the form loads this variable is assigned a value of 3 meaning the barricades can only take 3 hits before going down. When I have all of my barricades in an array, I can then check for any hits against it, this is why it was placed inside the EnemyTimer as it needs constant updating. I have 2 instances of a Nested FOR loop being used in this sub routine as I need to check 2 arrays for a collision so:

If (pbMissile.Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

And

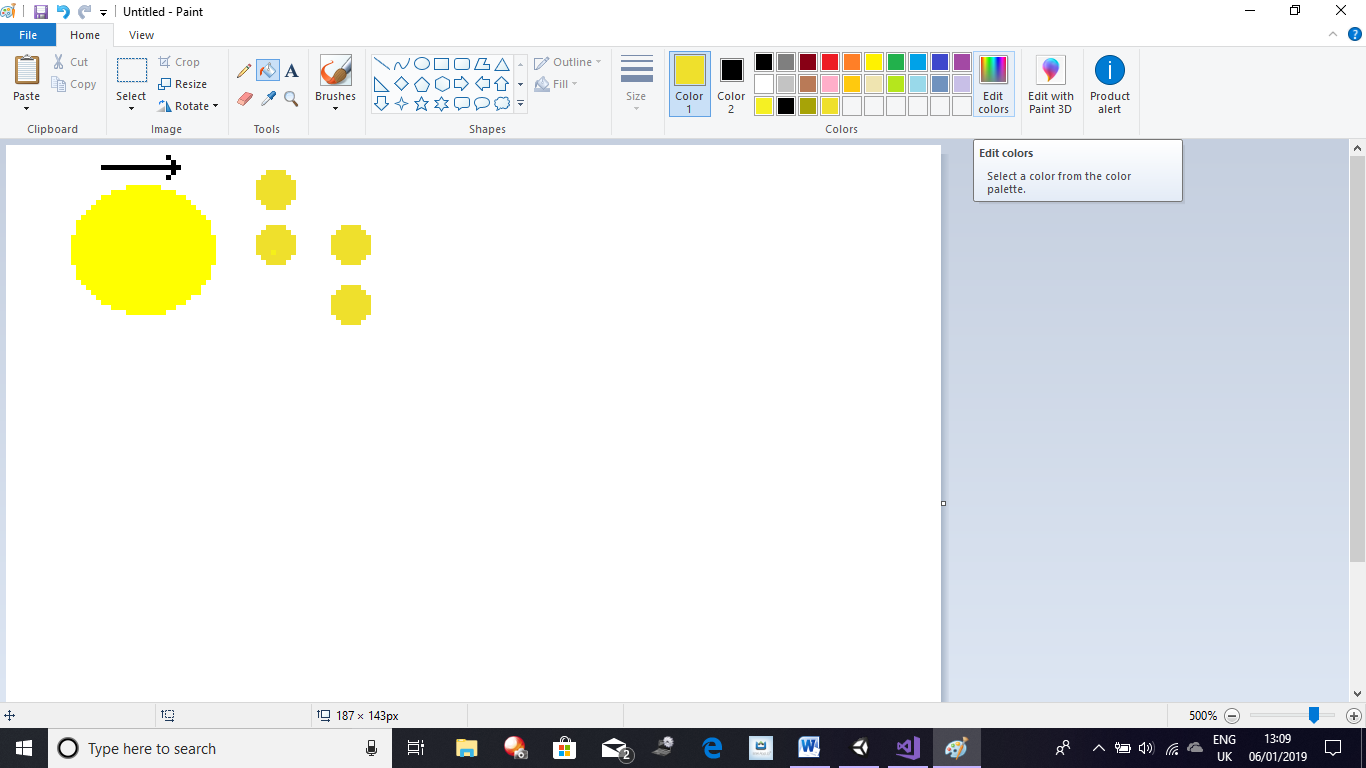
If (EnemyShot(i).Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

If one of these collisions occurs a life is taken away from the barricade life array and is compared to static values of lives, so 3, 2, 1 and 0. When the number of lives decreases from 3 and is equal to or between 2 and 3 the image of the barricade changes allowing the user to know the barricade has been damaged. The same occurs with 2 and 1. However when the number of lives is equal to 0 or below 1 then the barricade disappears through .Visible= false.

**Making PacMan**

Now I am going to produce PacMan. I have decided to make PacMan in Unity with the use of C# instead of VB.NET, I am still going to use Visual Studio however. I am doing this as with further research, PacMan is much easier to produce with the use of C# compared to VB.NET

Direction of Movement



Design Figure 16:

PacMan movement design

Pellets

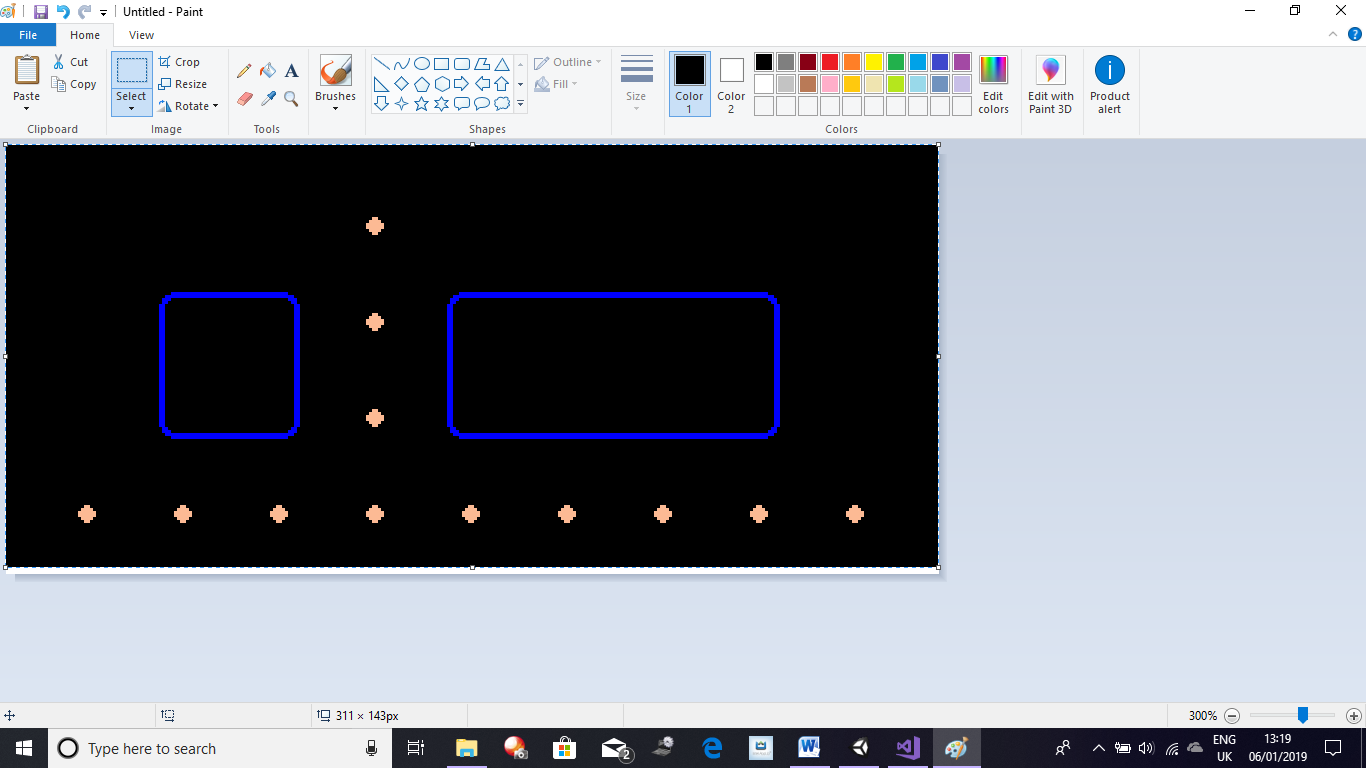
Pacman

4

3

2

1



Design Figure 17:

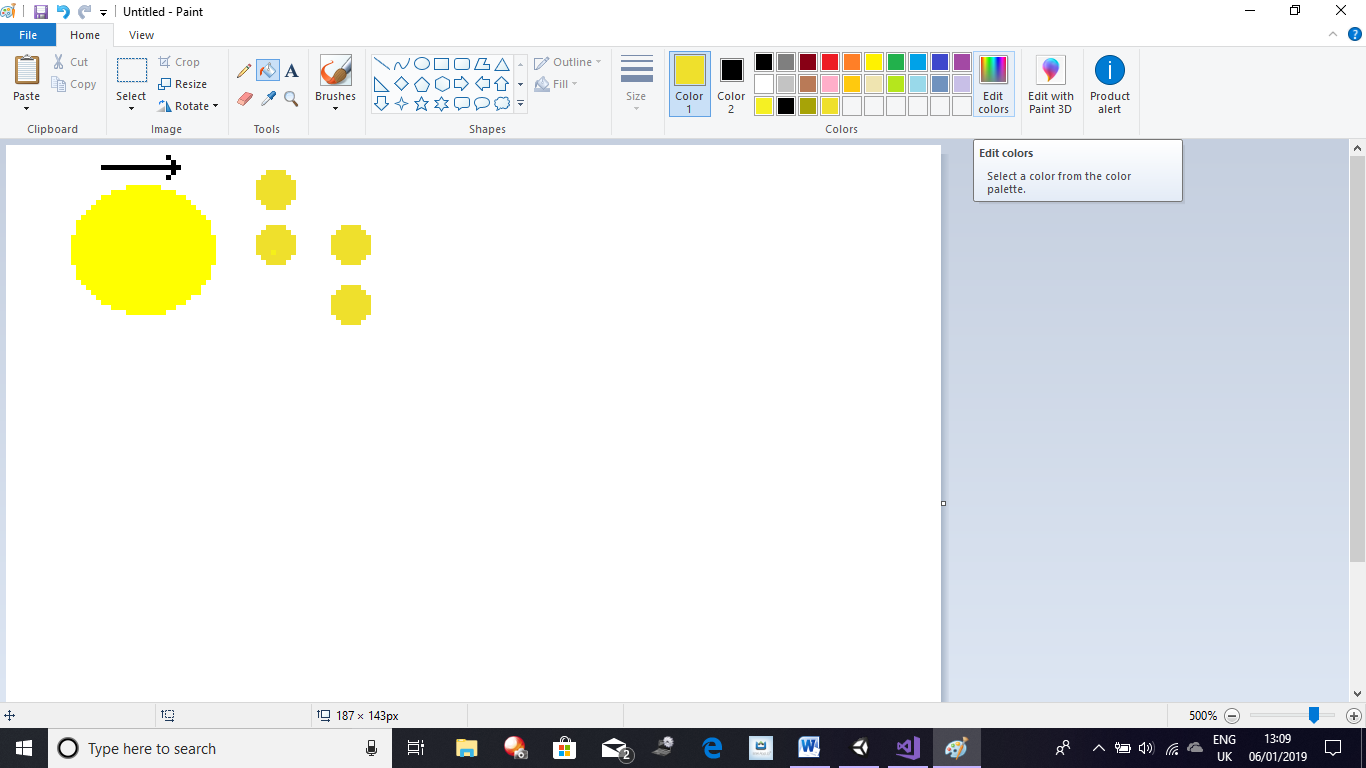
PacMan Nodes

Node 2

Node 3

Node 1

In Design Figure 17 I have a screenshot from my Map Maker showing some tiles from PacMan. The pellets labelled Node indicate the point where PacMan can move, so for instance PacMan can move from Node 2 to Node 3 but not from Node 2 to Node 1. By doing this I don’t have to have collision detection between every tile as the program will not let PacMan move up when there is no node to travel to, so for instance, see figure 16 and consider Pellets 1-4 are all node. PacMan can move from 1 to 2 and 3 but not directly to 4. This is because 4 is not directly vertical or horizontal to Pellet 1 but it is diagonal, this works the same way as Vectors work:

In the triangle adjacent, I have demonstrated how the nodes are going to be working as Vectors. In the diagram, for PacMan to move from 1 to 4, PacMan must travel to A (3) and then from A to B (4). The line labelled AB is drawn to show the total distance PacMan has to travel and shouldn’t be mistaken for an indication that PacMan can move from 1 to 4 directly. The pellets in between the node so from 2 to 3 are non-nodes and are used as normal pellets that PacMan can just consume to increase their score.

4

**A**

**B**

**AB**

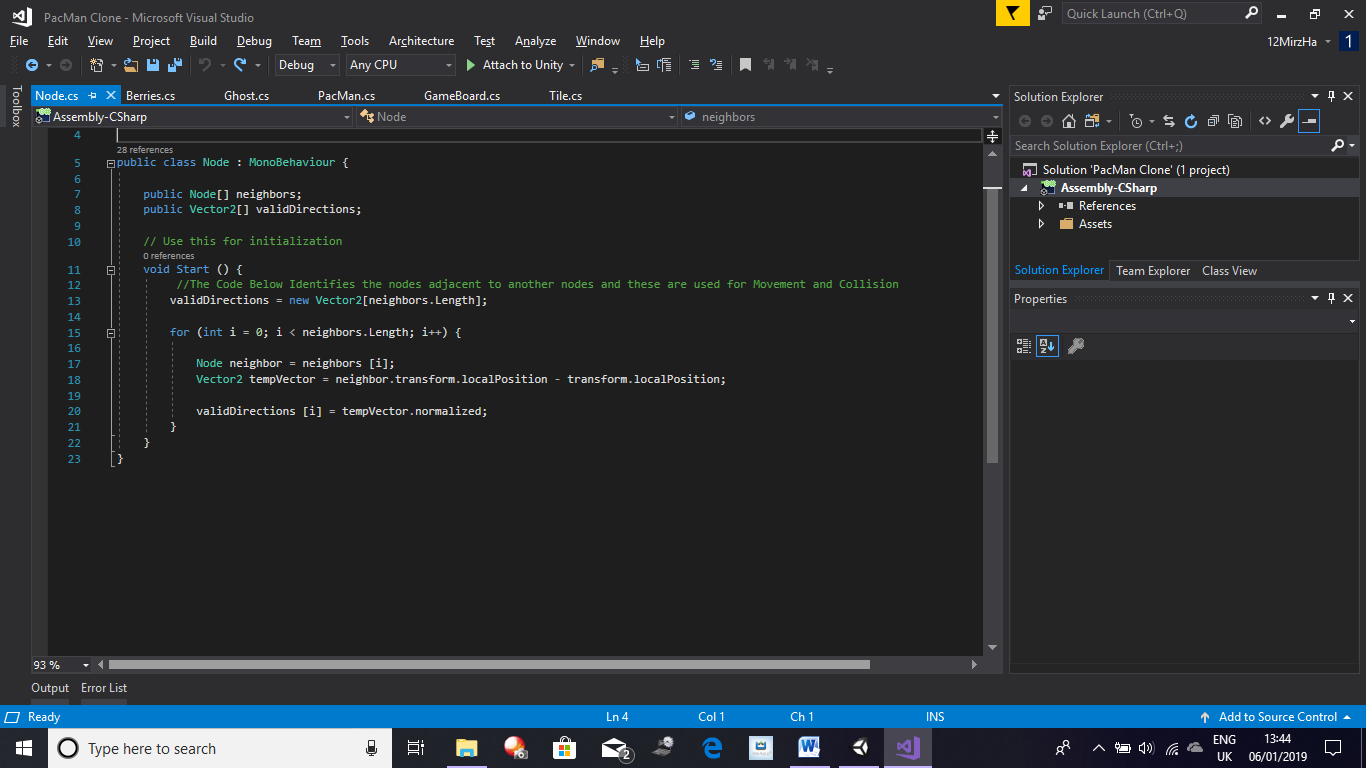
3

1

If the player tries to move up whilst PacMan is travelling to a node and there is no node above and just a tile, PacMan should ignore the input and continue moving to its original target. If a node is present it should start moving up towards the node. After some research into how to implement this I came across *The Weekly Coders PacMan tutorial which built the game in the manner that fitted my requirement, so to produce PacMan and the original Ghost AI I am going to follow his tutorials directly thus use the code he makes via his tutorials.*

To allow PacMan to distinguish what tile he is on I have used a script called Tile which will have all the different tiles that are possible as public Boolean varaibles which I can then assign inside Unity. Also to tell PacMan if a node is available, I have made the script Design Figure 18:

Design Figure 18: Node Script



In the script adjacent, I have made two public variables, one is neighbours which I will use to tag all the node neighbours of a single for example in design figure 16, the neighbours of pellet 1 are 2 and 3 as those are the ones it is next to. Valid Directions is just a check for direction to make sure PacMan can move that way. The rest of the code identifies the neighbours of a node and calculates if it’s possible for PacMan to move in that direction by checking all the neighbours of its current location and checking if PacMan can move there or not. To get PacMan to move now that I have identified where it can move and how through the Nodes I now need to allow Pacman to move to those nodes and animate them with PacMans movement. To animate PacMan I will use the inbuilt animator inside Unity using the images below in Design Figure 19 *which were taken from the mentioned tutorials*:

E:\College Subjects\Computer Science\Computer Science NEA\My Games\PacMan Unity\Help\PacMan Clone\PacMan Clone\Assets\Graphics\pacman_2.pngE:\College Subjects\Computer Science\Computer Science NEA\My Games\PacMan Unity\Help\PacMan Clone\PacMan Clone\Assets\Graphics\pacman_3.pngE:\College Subjects\Computer Science\Computer Science NEA\My Games\PacMan Unity\Help\PacMan Clone\PacMan Clone\Assets\Graphics\pacman_1.png

Design Figure 19: PacMan Animation Images

***Making PacMan Move***

Now that I have my animations done, I now need to allow PacMan to move between the nodes. First of all, I’m going to allow PacMan to change direction so it faces the direction the player is trying to make him move. This will also allow me to check if PacMan is responding correctly to user input. To do this I need a sub routine which will check PacMans current direction and compare it to the direction the player is trying to move. If PacMan isn’t moving in that direction than we need to make our next direction equal to the direction the user wants us to. Once this is done we need to check the tile of PacMan to see if PacMan is on a Node or not. If PacMan is on a Node, then we check the neighbour of that node and see if there is a Node in the direction we are trying to move, if there is than PacMan moves to that Node. Now we need to assign the user input direction to this, I did this by passing the user input when the sub routine is called. So the code which checks for a Node is only called when a directional key is pressed. The sub routine which checks for Input is called inside the Update void which is called by Unity once every frame. To make PacMans animation change with the direction it’s facing we have to update the animations every time PacMan changes direction. Now that PacMan changes direction I will need to record its orientation every time PacMan changes direction, I am going to this by checking the direction PacMan is moving in and as this is a 2D game with only 4 direction as PacMan cannot move diagonally, I am going to check the direction that PacMan is moving in and assign the orientation variable the corresponding direction. To allow PacMan to consume the pellets I will use the Tile script and assign each pellet a Boolean value which will check if the pellet is a pellet or not as well as checking if it’s been consumed. I will also need to check if it’s a super pellet or not. To do this I will have to produce an array which will contain all present game objects and future, for instance pellets, ghosts and PacMan will need storing inside this. This will need to be done in a separate script and to do this I will need to assign each object a tag for instance PacMan has the tag PacMan and the Pellets have a tag pellet or super pellet etc. Then to allow PacMan to check if the tile it’s on is a pellet or not I will have to check the array and compare it to PacMans location to make sure PacMan is on a tile which consists of something, once I’ve done this I will check if the tile is a pellet, if it is a pellet then I will check if the pellet has already been consumed, if it has then nothing happens however if it hasn’t then PacMan gains some points to their score. At this point I should have a PacMan who can move around the map and eat pellets.

***Making Ghosts***

The ghosts in PacMan each follow their own AI and they follow 4 different modes. These modes are Chase Mode, Scatter Mode, Frightened Mode and Consumed Mode. Frightened Mode and Consumed Mode are only available once one of the four super pellets is consumed. All of the Ghosts initially start off with Scatter Mode. In Scatter Mode all of the ghosts will move towards their home nodes which should all be in a corner of the map and one for each ghost. After a few seconds the ghosts change to Chase Mode where they will chase PacMan according to their AI. If PacMan consumes a Super Pellet then all of the ghosts change to Frightened Mode, then when PacMan consumes a ghost that ghost enters Consumed Mode and moves back to the Ghost House at a rapid pace then when the ghost reaches the Ghost House it changes back to its original speed. I have already written the Pseudocode for the ghost AI in my Overview, my actual implantation of it *using the tutorials* can be seen below:

Red Ghost:

Vector2 GetRedGhostTargetTile()

{

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = new Vector2(Mathf.RoundToInt (pacManPosition.x), Mathf.RoundToInt (pacManPosition.y)); //Rounds PacMans position to an Integer

return targetTile; //Function outputs targetTile

}

Pink Ghost:

Vector2 GetPinkGhostTargetTile()

{

//Has to be 4 tiles ahead of PacMan and need to take into account the positon and orientation

Vector2 pacManPosition = pacMan.transform.localPosition; //Variable stores PacMans location

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Stores PacMans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Rounds PacMans x coordinate to a whole number

int pacManPositionY = Mathf.RoundToInt(pacManPosition.y); //Rounds PacMans y coordinate to a whole number

Vector2 pacManTile = new Vector2(pacManPositionX, pacManPositionY);

Vector2 targetTile = pacManTile + (4 \* pacManOrientation); //Multiplies all of the vector2 by 4 so both x and y coordinate seperatly and then adds pacManTile to it so Pinky is always 4 tiles ahead of PacMan

return targetTile; //Function outputs targetTile

}

Blue Ghost:

Vector2 GetBlueGhostTargetTile()

{

//Most Complex AI out of all of the Ghosts

//Select the Position two tiles in front of PacMan then draw a vector from blinky to that position then double the length of that vector

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan> ().orientation; //Gets Pacmans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Stores PacMan X coordinate in this variable

int pacmanPositionY = Mathf.RoundToInt(pacManPosition.y); //Stores PacMan Y cooridinate in this variable

Vector2 pacManTile = new Vector2 (pacManPositionX, pacmanPositionY); //Get PacMan Tile

Vector2 targetTile = pacManTile + (2 \* pacManOrientation); //Create a TargetTile for where PacMan is

Vector2 tempBlinkyPosition = GameObject.Find("Red\_Ghost").transform.localPosition; //Create a temp variable for Blinky position

int blinkyPositionX = Mathf.RoundToInt(tempBlinkyPosition.x); //Stores Blinkys x position in this variable

int blinkyPositionY = Mathf.RoundToInt(tempBlinkyPosition.y); //Stores Blinkys y position in this variable

tempBlinkyPosition = new Vector2(blinkyPositionX, blinkyPositionY); //Stores Blinkys coordinate in a new vector2 variable

float distance = GetDistanceApart(tempBlinkyPosition, targetTile); //Gets the distance apart between blinkys position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempBlinkyPosition.x + distance, tempBlinkyPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

return targetTile; //Outputs the Target Tile

}

Orange Ghost:

Vector2 GetOrangeGhostTargetTile()

{

//Calculate the Distance that Clyde is from PacMan then if the distance > 8 tiles then targetting is same as Blinky else the target is his Home Node, so same as Scatter Mode

//This is because Clyde never catches PacMan is similar to a bluff to trick the player into breaking logic and moving into risky situations

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Clyde to PacMan

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

if (distance > 8) //If Distance is greater than 8 then the following code is ran

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y));

}

else if (distance < 8) //If Distance is greater than 8 then the following code is ran

{

targetTile = homeNode.transform.position; //Makes the TargetTile the Home Node

}

return targetTile; //Outputs the targetTile

}

It should also be noticed that each ghost is released at different times from the Ghost House when the game starts. Each ghosts AI was discussed in the overview, but to elaborate the Red Ghost always follows PacMan when it is in Chase Mode by always tracking PacMans tile and moving to that tile. The distance is calculated from another sub routine using the distance formula:

The Implementation of this can be seen below:

float GetDistanceApart (Vector2 posA, Vector2 posB) //Function allows the ghosts to know the distance apart they are from two positions A and B

{

float dx = posA.x - posB.x;

float dy = posA.y - posB.y;

float distance = Mathf.Sqrt(dx \* dx + dy \* dy); //Simple Mathematical formula used to calculate distance: distance = sqroot(x1 + x2)^2 + (y1 + y2)^2

return distance; //Returns the distance apart

}

The Pink Ghost always has to be 4 tiles in front of PacMan in Chase Mode, it does this by looking at the way PacMan is facing so its orientation and then looking at where PacMan is moving by looking into its current location each time PacMan and trying to move 4 tiles in front. The Blue Ghost looks at the two tiles in front of PacMan using the same method the Pink Ghost uses by taking into account PacMans orientation and direction that PacMan is moving in then choosing the two tiles in front of PacMan, we then find the vector from the Red Ghost to that point and double that quantity. This means that when the Red Ghost is far away from PacMan, the Blue Ghost is also far away. But as the Red Ghost closes in so does the Blue Ghost. The Orange Ghost follows the same algorithm as the Red Ghost by locating PacMans location and then moving to that tile unless the number of tiles between the Orange Ghost and PacMan is less than 8 in which case the Orange Ghost moves towards its home node. The Orange Ghosts acts as a bluff to trick the player. In PacMan the player also finds berries which if consumed further increase the players score. These berries however appear for a short amount of time after which they disappear.

***Making Berries, Lives and Leveling Systems***

To program all the berries I made two counters one to count how many appeared which shouldn’t exceed one as well as one to count the number of berries consumed. The berries should initially appear invisible and will appear after a set amount of time has passed and disappear after a set amount of time passes. After a few levels the berry that appears should be random between all the available ones, here I will use a random number generator where I will select a number from a range and the number chosen will have its corresponding berry appear. *The berry pseudocode and actual code are all my code and have not been taken from the The Weekly Coders Tutoral nor is that code related to the mentioned, only the original Ghost AI, modes and PacMan programming for Map and movement with interaction with pellets has been done using tutorials.*

If BerryVisible 🡪 True Then

If Player CollidesWith Berry Then

plyrScore += BerryPoints

End If

End If

If TimeVisiblie > TimeAllowed Then

BerryVisible 🡪 False

End If

At this point I should have a working PacMan which can consume berries as well as eat pellets and super pellets. The ghosts should react appropriately to the time on the timer by using the appropriate mode and reacting to PacMan when a Super Pellet is consumed by changing their mode to Frightened Mode and to Consumed Mode when they are consumed. I also have berries appearing to the screen now. Now I am now going to implement the level system in PacMan so the berry that appears changes dependent on the level that PacMan is on. To do this I will be using a static integer variable which will increment from 1 by 1 each time retaining its new value every time the level changes. To judge when the level is completed I am going to get the program to count up the number of pellets by incrementing it when the game is loaded by counting up all of the objects that are identified as pellets. I will also count up the number of pellets that get consumed by PacMan and when this number equals the total number of pellets the level will be increase with the ghosts back in the ghost house and the pellets all visible again however the player should retain their score when this happens.

BerryThatAppears = Level

If NumOfPelletsConsumed 🡪 TotalNumOfPellets Then

PacMan.Location 🡪 OriginalLocation

Ghosts.Location 🡪 OriginalLocation

Score 🡪 Static

Level += 1

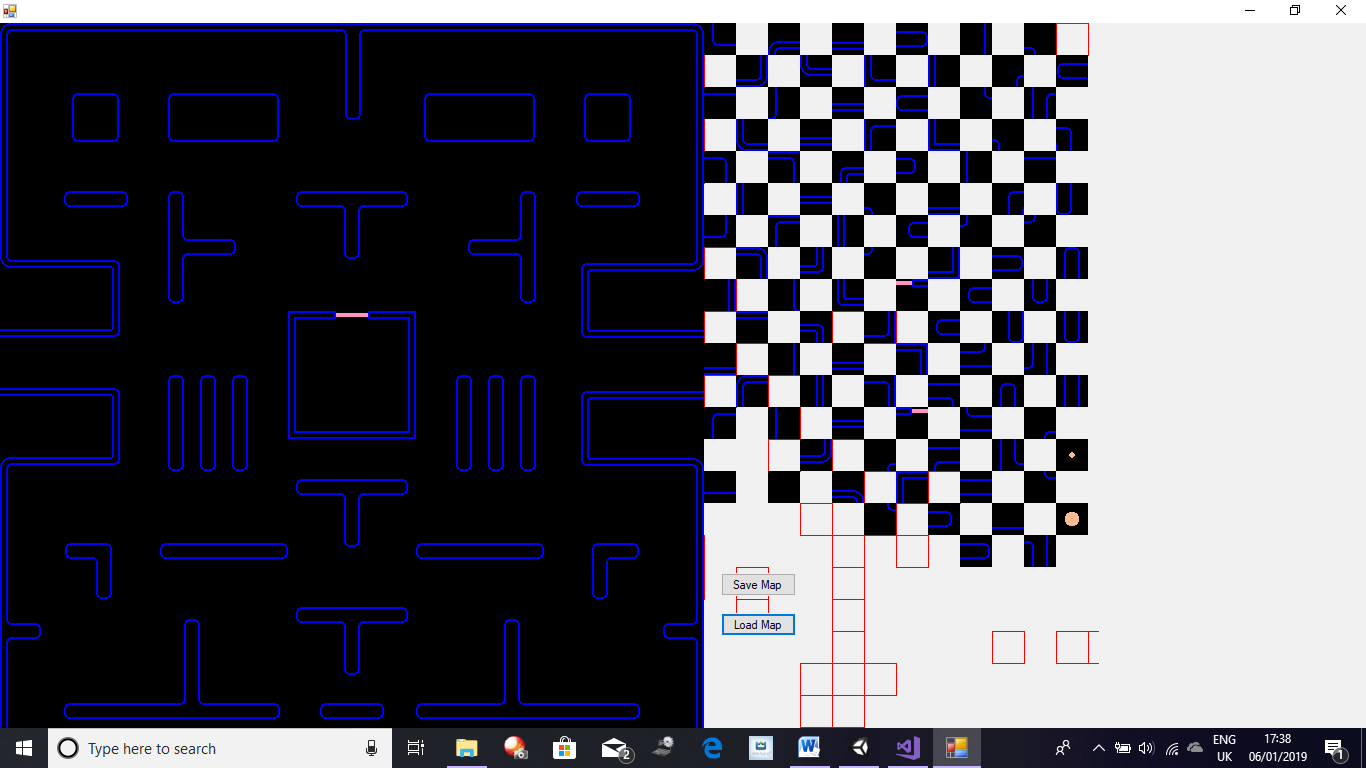
End If

To finish of the base PacMan game I am going to add a lives system as well as collision with the ghosts. Initially when the game loads PacMan will have three lives which will be displayed to the user in the same way as in Space Invaders, with visual images. These images will change as the integer variable for lives changes. This life variable should also be static so when the scene is reset for when the player gets to a new level the number of lives isn’t reset. To detect a collision with PacMan and the ghosts, I am going to make rectangles round PacMan and all of the ghosts which should get updated each time the ghosts and PacMan move. When a Collision occurs the two rectangles will be overlapping, the program should detect this and reset the game back to how it started with the ghosts inside the ghost house, however the number of pellets should stay the same as before PacMans death, also in the base PacMan map there are two portals located either end of the map horizontally. At these points I will place two invisible objects which when PacMan is on top of, which will be found through the detection of what tile PacMan is on, when PacMan or the ghosts are on this tile it should teleport them to the other side of the map as well as keep their movement going. With this done I should have a working base PacMan game.

**My Implementation on Each Game**

**PacMan**

As I now have a base working program for all 5 games, I am now going to produce their implementation. For PacMan, when the user selects their map from one of the three already produced the game should load up the same way it does in the base game with all of the ghosts inside the ghost house. This should just consist of me designing the maps in the Map Maker and then importing the design into Unity. To do this I will have to alter the location of the nodes and the Super Pellets. Alongside adding new Maps, I will also be *adapting the Weekly Coders Tutorial Code and using it to develop a Ghost AI of my own. The Ghost AI I make will only work when the user plays the custom Maps so the maps Imake.* One of the Maps I have designed alters slightly from PacMans original Map can be seen below in Design Figure 20:



Design Figure 20: One of the designs for a PacMan Map

Though the difference from the original isn’t too great, so it doesn’t take away from the original PacMan experience however it alters enough to cause the player to make mistakes and get caught by the ghosts. This is just one of the three maps for my PacMan. I am going to design all of maps using my Map Maker. One map is the original map, the map on pg 68 (above) is the second map and third will be another map which will use *my adapted Ghost AI* and the third map will also be designed on my map maker.

***Making Scoring Mechanic***

Now to build the scoring mechanic in PacMan, I am going to be using the streamwriter to write the players score from the game to a text file. The score will be grabbed from the score integer and should be saved alongside the Players username. The program will find out the Players username by displaying an input box and submit button when the player reaches 0 lives. This will also cause the movement of the ghost and PacMan to stop so they can no longer move around the Map and so the score stays static beyond this point. This means when the player reaches 0 lives a input box and button should appear when the player loses all their lives and when the button is pressed the players score is saved from the base game to a text file. This score system will only be used in the base games and not in any of my implementation to keep the games fair as some of the implementations are easier to win then others. The code for saving the score can be seen below alongside some basic Pseudocode of how SaveData will work, this code is general code and has *not* been taken from the mentioned tutorial:

UserInput 🡪 UserName

StateMent 🡪 (UserName & “, ” & UserScore)

StreamWriter(TextFile)

Write 🡪 Statement

Close StreamWriter

private void SaveData()

{

//Path of the File

string path = ("path");

//Gets writer which will write to the file in path

StreamWriter writer = new StreamWriter(path);

//Writes the line in the parameters

writer.WriteLine(textField.text + ", " + playerOneScoreText.text);

//Closes the Writer

writer.Close();

//Path of the File

string path2 = ("path");

//Gets writer which will write to the file in path

StreamWriter writer2 = new StreamWriter(path2);

//Writes the line in the parameters

writer2.WriteLine(playerOneScoreText.text);

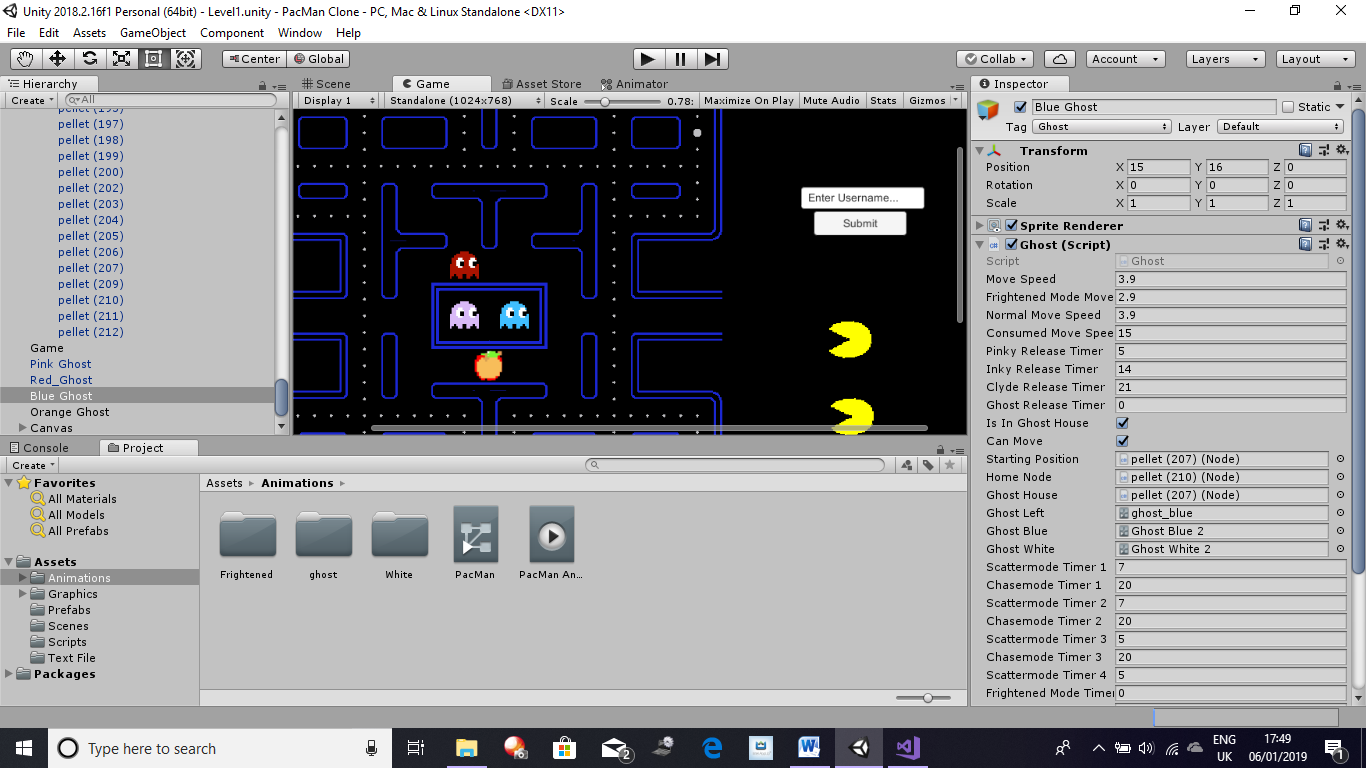
//Closes the Writer

writer2.Close();

Application.Quit(); //Closes the Application

}

The code above will be ran when the submit button is pressed. The UI used to save the score can be seen below:



Input Field

Design Figure 21:

PacMan Save UI

Submit Button

To reach the high score screen for PacMan, I am going to use the button on the menu screen. When that button is pressed the score from the PacMan internal game text file should be read in and stored in another file. I will then use the streamreader to check if the high score text file is empty or not by reading it, if it is then the data read from the PacMan file is added in, if it isn’t then we clear the file which read the data from the PacMan text file and write the new data from PacMan into it, this is done to remove any old data in the file so the new one can be added in. I will then read the file that was just cleared and entered with new data and I will also make the program read the internal high score text file to check if the data being entered in already exists. I am going to do this by reading in the data from both files into two different arrays, I will then compare the values of these two arrays and with the use of a counter, when the data is seen to be repeated the counter will be reset back to 0 whilst if the data doesn’t exist then the counter will be incremented. Once both arrays are searched through the counter will be checked to see if its value is greater than its base value which was 0. If it is then the data is written in if it isn’t then the data should not be written in. Once this is done the data from the high score text file should be read in to another array, this array should be read and it should separate the score from the username and store the score in a separate array. This can be done using delimiters or using tryparse. Once just the scores are read in to this array, the User should be able to sort the data with the use of a button which will run Merge Sort on the score array. This will sort the score array. My Merge Sort algorithm can be seen below:

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length – 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

The Merge sort algorithm I used is also a recursive algorithm as it calls itself with the lines:

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

The parameters consist of the arrays, then the low value which when the array is called will always be 0 and a high value which is going to be the arrays length – 1. The idea of Merge Sort is that it’s quicker to solve two small lists then combine them together then to sort one large list. Merge Sort has a Big O of O(nlog(n)), this means that it’s a very efficient algorithm. The logic behind my Merge Sort Algorithm is that it splits the list into two parts from the roughly the centre of the data (count wise so if 20 pieces of data it will be split into 2 sets of 10 data). This division continues until every piece of data is separated from all the other data. The data is then merged together into a list roughly double its previous size, whilst simultaneously sorting each item into order, this repeats and the data continuously keeps merging until all the data is back together again:

For i As Integer = 0 To Length – 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

Once the data is sorted the data can be searched through using Binary Search, the user should only have access to binary search once the data is sorted as Binary Search requires the data to be sorted. I am going to program it so that the Binary Search is ran from a button which only becomes available after the Merge Sort Button is pressed, this makes it so the data is always going to be sorted before the user can search through the data. My Binary Search Algorithm can be seen below:

Sub BinarySearch()

'Binary Search Algorithm can be seen Below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score?")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

Binary Search requires an ordered list and always pin points the value by looking at the centre value of the list and always splits the data into half at each comparison. Binary Search is also a very efficient algorithm as its Big O is O(logn). When the search is taking place the data gets looked at to see if it’s bigger or smaller than the point that it is currently on. If it is greater than one half is taken and the other half is discarded. This process repeats until the data being searched is found.

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

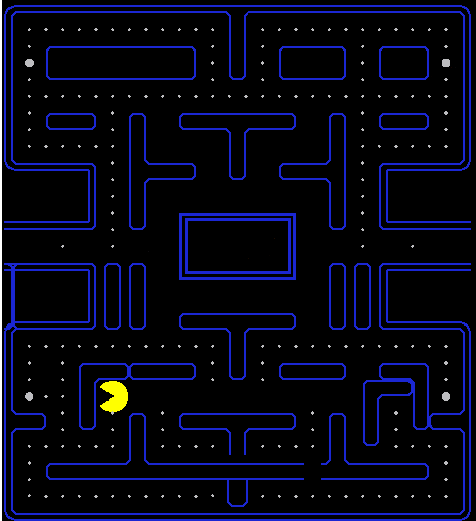
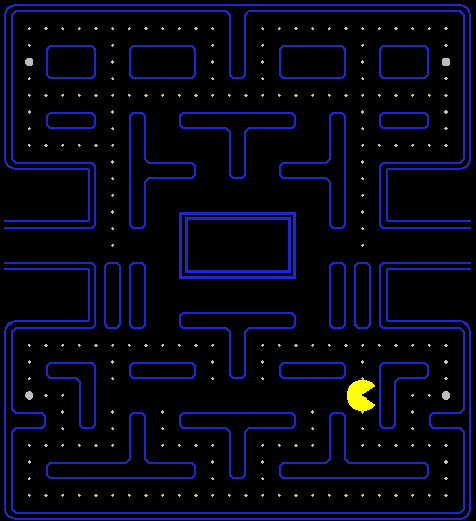
The code above is taken from the Binary Search code and this code checks if the lowest value is the lower or equal to the highest value as well as checking if value being searched is found or not. If it is then Result will equal Middle which would be the result being searched. If the inital conditions are fulfilled then the centre value of the data is located and saved to a variable Middle. Then a check occurs to see if the UserInput is greater than, less than or equal to the centre value. If it is greater than or less than the Middle than either the High or Low value is altered appropriately by either adding or subtracting from the High or Low value respectively. This is done to make the data look at the data lower than the centre or greater than the centre. As a result has not been reached the loop continues and repeats this process than the value we are looking for is the only thing left, thus it becomes the middle so we make the result equal to the middle. In my PacMan high score screen the player will also have the ability to search for scores. I will do this by checking the array that stores the username and the score. I will then check the array for any data that starts with the username provided by userinput, the userinput will be received through an input box. To provide the searched for users score the data will be split from the user username to just get the score from the array, one way of doing this is using delimiters. So to conclude in my PacMan high score screen the user should be able to search for any player to get the score of that user and see the position of the score of that user, this will be done through the use of Merge Sort and Binary Search.

The code below inverses the order of the data in the array, as the way I made my Search and Sorting Algorithms the greatest value appears at the last index of Array so it would be at the greatest index in the array, to solve this I implemented the code below into my Binary Search Algorithm:

Result = JustScore.Length – Result

This line just reverses the array in a sense by allowing the highest score which would be at the bottom of the array, have its position reversed when it is stored into result so, for example if the highest score is located in the 11th spot in an array of size 11 where the first index is 0. By doing this instead of the highest score being in the 12th spot this line does the following: 12 – 11 = 1 so the player with the highest score sees that their score is in 1st place instead if 11th.

***Making Multiple Maps***

Finally, to add my maps to PacMan, first I designed them with my Map Maker then implemented them into my game without the Ghosts. The two maps can be seen below:

Design Figure 22:

PacMan Maps

As it can be seen the maps both differ from the original map, a greater difference can be seen on the map on the Left than the map on the right. This is because the idea behind the map on the right is that it varies slightly as the user would be used to the original map, so when they come on to this map there is a only one path from the top part of the map to the bottom part of the map.This means that the player will have to pass through the centre which means that they will have to go through all the ghosts as well which will also have to pass through the centre of the map to get to the bottom of the map. The map on the left limits the number of passages in total at the bottom of the map even blocking it off at points forcing the player to reconsider their choices. Due to the layout of each Map I have decided to implement more bluff AI like that of the Orange Ghost and chase AI like that of the Red Ghost, however I still want to keepthe number of Ghosts constant to the number of Ghosts in the original game.

***Making the New Ghost AI***

As I want more chase Ghosts, I kept the Red Ghost AI the same for these Maps. However for the Pink Ghost I changed its AI so it acts similar to the Red and Orange Ghosts. What I though of doing for the Pink Ghost was that it will continuously follow PacMan until it gets to close to PacMan in which case it moves towards the centre of the Map. This would make it so that it will always chase PacMan as well as block up the centre making it harder for the player to move between the top and bottem section of the maps. The pseudocode for this can be seen below:

If DistanceFromPacMan > 2 Then

PinkGhostTarget 🡪 PacManLocation

ElseIf DistanceFrom PacMan <= 2 Then

PinkGhostTarget 🡪 CentreOfMap

End If

The pseudocode above basically illustrates the AI of the Pink Ghost. For the Blue Ghost what I did was take into account where the Pink Ghost was and what it was doing, if the Pink Ghost was chasing PacMan the Blue Ghost tries to get 2 tiles in front of PacMan otherwise it takes into account the location of the Red Ghost and tries to get infront of PacMan following its original AI. The pseudocode for it taking into account the Pink Ghost can be seen below:

If PinkGhostMode🡪 ChasePacMan Then

BlueGhostLocation 🡪 PacManOrientation + 2

End If

In essence, this makes it so the Blue Ghost will also patrol between the centre of the Map as well as trying to get in front PacMan, this will make it difficult for the player tos switch between top and bottom as well as avoid the Blue Ghost when it tries to get in front of the player. The Orange Ghost will also try to get infront of PacMan or run away from PacMan dependent on what the Blue Ghost is doing. If the Blue Ghost is taking into account the Location of the Pink Ghost then the Orange Ghost will try to get 4 tiles infront of PacMan, however if the Blue Ghost takes into account the Red Ghosts Location then the Orange Ghost will chase PacMan and run away if it gets too close. So basically if the Blue Ghost is following its original AI then so will the Orange Ghost. The Pseudocode for the Orange Ghost trying to get 4 tiles infront of PacMan can be seen below:

If BlueGhostMode 🡪 PinkGhost Then

OrangeGhostLocation 🡪 PacManOrientation + 2

End If

Overall, I think this Ghost AI will work effectivily to neutralise the player when they come to the centre and in trying to block the player out. Also as we have a balance between the Ghosts trying to get infront of PacMan and chasing it normally, this should also make it difficult for the player when traversing the map in general. *All other PacMan code will remain the same with the original which was achieved by following the weekly coders PacMan tutorial directly.*

**Pong**

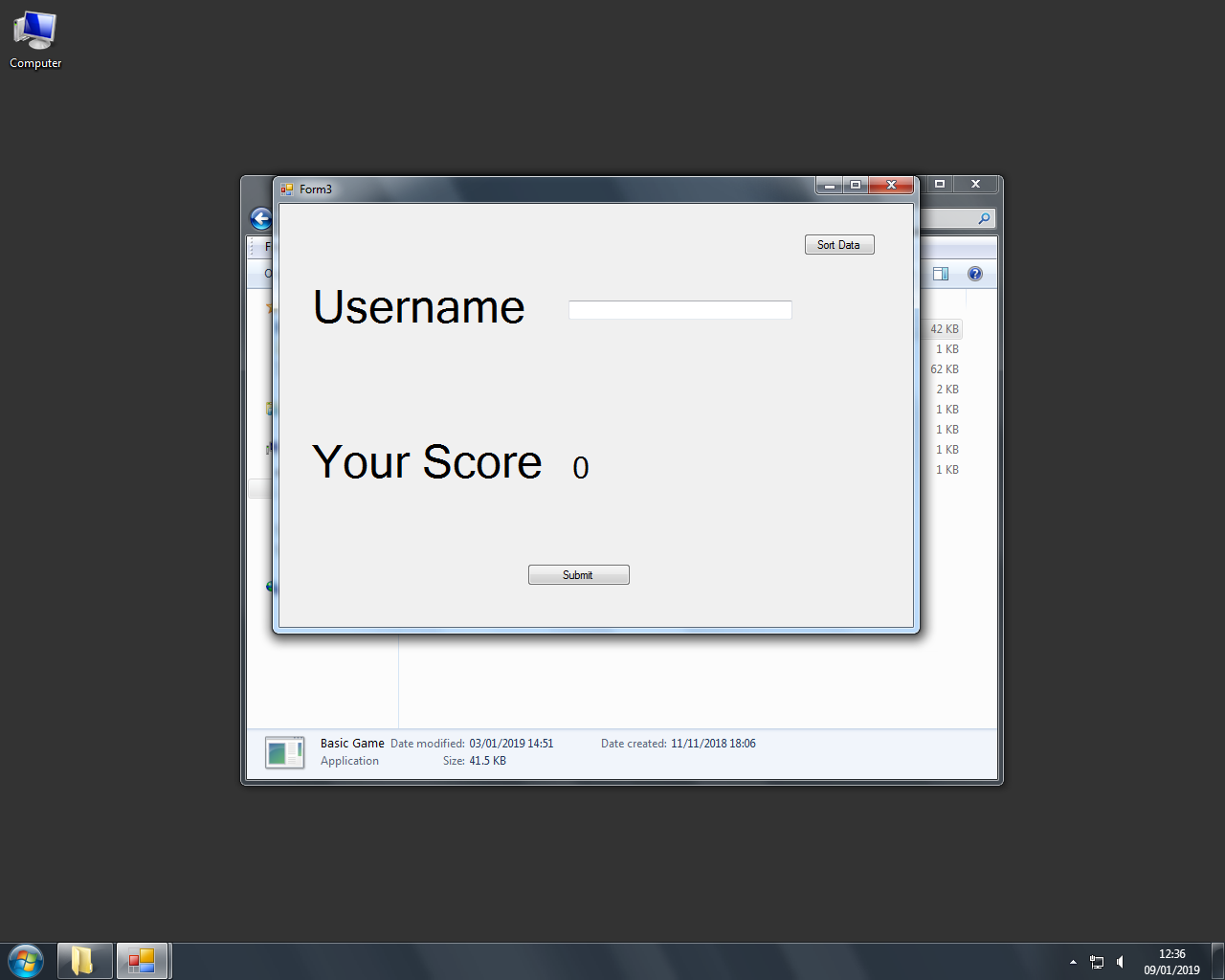
To produce Mad Mode in Pong, I am going to use the .NET Framework random number generator again so I can choose a random number between a set range. I am also going to use a counter to count when an event occurs so I can repeat an event by just referring to the number that is associated with it. Initially, I want the computer to cycle through all of these different modes, these modes will include but will not be restricted to the balls speed increasing to another speed as well as the ball moving in the other direction, another mode is that the computer will either instantly hit the ball or ignore it. This means the computer paddle has a chance of either hitting the ball or not. Another mode can be that the ball speeds up or slows down. Another Mode is that the ball moves up horizontally when approaching the player. The Pong game will not use any High Score coding as the goal of the game is to beat the computer. To conclude in Mad Mode the ball will be given the ability to try to trick the player whether it is by bluff by increasing the speed of the ball or by directly changing what it’s doing. Once I had a fully working Mad Mode, I combined it into my original program and gave the user the option to access Mad Mode when the program loads up via InputBox. This should conclude all of my implementations for Pong

**Flappy Bird**

For the Random Event Mode in Flappy Bird, I am going to implement a timer to constantly update the game with the events underneath. So every time the timer ticks I want it to call subroutine which would cause a random event to occur. I am also going to use the .NET Framework Random Number Generator to generate a random number which I can then check to see if the number generated corresponds with an event, if it does an event takes place, if it doesn’t then nothing takes place and the user carry’s on like normal. Some Random Events I am going to include are; the player becomes invincible so they can fly through the pipes for a few seconds; the player gets more points for crossing each pipe; the player doesn’t gain any points for going through a pipe; the player gains one life letting the player continue once they have died; the player temporarily becomes invisible so the player cannot see the Bird; the game slows down in pace so the pipes come at the Bird at a slower speed; the pace of the game increases so the pipes come at the bird at a quicker speed. These are my current ideas of random events to include in Flappy Birds Random Mode. Also I am going to put both Random Mode and the Normal mode in the same program with a User Input which lets the User decide which mode they want to play between the two modes when the game loads up, it gets the UserInput via InputBox. I will also include a label, so I can indicate to the player what event is taking place if they are playing in Random Mode, so the user can see what event they are experiancing.

***Making Scoring System***

In the original Flappy Bird game I will also be producing a scoring system which will save the players score to a text file. The player should then be able to search for their score and see their ranking in the game. This will be done using Merge Sort and Binary Search, the code for both of these algorithms can be seen above (in PacMan high score section as well as in appendix) as I will be reusing the code I have already made. Also as all my games use VB except for PacMan I can *use the same High Score Screen for all of the games as well as code*. As a result in all my games the High Score screen except for Pong and PacMan, I am going to transfer the score from the game form to the high score form. In that Form then the program will read all of the scores from the text file into an array and then separate the scores and the username so the score can be sorted using the Merge Sort Algorithm. Once the score is sorted the user should be able to search for their score and receive the position of their score in comparison. My form layout can be seen in Desing Figure 23:



Design Figure 23:

VB Game High Score Screen

This button will sort the data

The user will enter their username in this text box

The Users Score will be taken from the game form and it will be saved to the label here which will display it to the user

This button will save the Users data to the text file which will act as a flat file database.

Desing Figure 22 shows what the High Score Screen will look like for all of my games made in VB except for Pong as the main objective of that game is to beat the AI and PacMan as the High Score Screen for that game differs slightly. Once the User has pressed the Sort Data button the Binary Search button should become visible as the data will be sorted via Merge Sort. This will allow the user to search for their position within the data. A Sequence diagram and User Interaction diagram can be seen below:

Design Figure 24: Sequence Diagram (See Appendix)

Design Figure 23: User-Interaction Diagram (See Appendix)

Both of these diagrams illustrate how the user interacts with the scoring system in each game. The sequence diagram also illustrates what takes place between the front end and the back end. This would conclude how the scoring system in each game.

I am now going to make the scoring system which will take all of the scores from each game and form a total, which will then be sorted using Merge Sort and Searched using Binary Search.

**Overall Scoring System**

For the total score screen, I will need to form a top 5 list. I am going to do this using the Merge sort Algorithm and when the data is sorted, I am going to use the array to find the position of each score. As my Merge Sort Code sorts the data with the largest score at the bottom of the array, for the number 1 spot I am going to take the score stored in the arrays last index. Using this logic, I am going to take one away from the index of the greatest score for the second score and one away from the index of the score in second to get the score at third place etc. As this will require the data to be sorted I will have to run Merge Sort before this is done. As a result the data will be sorted autonomously by the code. As the high score screen does not require the user to save their scores on this form, the best place to call the Merge Sort would be in the Form Load event.

To get the sum of the score for one user what I am going to do is read all of the usernames from each game and then save them all into a list, as it is likely there will be repeated usernames, I will then have to remove duplicate data. To do this I will use the .NET Framework and use Distinct().ToArray() with the list. This will remove all repeated data from the list and save it all to an array that is equated to it. I am then going to loop through the array with the distinct Usernames and search for scores that correspond with that user. To do this my logic is to read all of the scores that are saved with the usernames into lists. I am then going to then loop through each username saved into the array I made prior. Using the .StartsWith I am going to check if the data starts with that username, this can be done as the data is saved as:

Test1, Score

Test2, Score

Test3, Score

Once I have found the index where that users score is saved I am then going to split the data at the comma (,) to get the score and the username separately, I am then going to use Integer.Parse to just get the score of the users. This process will be repeated for all of the games and once all of the scores of a user are found, they will be added together to form a total score. This process will be repeated for each Username inside the Username array. There shouldn’t be any duplicate data, so as a result I can then save the total of each username with their username into a text file. To sort the data I am going to separate the data at the comma again using either delimiter or split to get the username and integer. Then using Integer.Parse I will save the score of each user into a separate text file which will then be read into an array, this array should then be sorted using my Merge Sort algorithm. At this point I would have the sum of each user scores saved into an array which would be sorted. Therefore, I can then form my top 5 list using the method stated at the start below Design Figure 23 and 24. I will then allow the user to search for their score using their username, they can then use this score to find their position against the other users. This should conclude the scoring, storing and high score section of my games. The Pseudocode for a broad view of how this will work can be seen below:

For Each DistinctUserName

If DistinctUserName 🡪 FlappyBird Then

Score1 🡪 Integer.Parse(DistinctUserName Line)

ElseIf DistinctUserName 🡪 SpaceInvaders Then

Score2 🡪 Integer.Parse(DistinctUserName Line)

ElseIf DistinctUserName 🡪 BlingBlingBoy Then

Score3 🡪 Integer.Parse(DistinctUserName Line)

ElseIf DistinctUsernmae 🡪 PacMan Then

Score4 🡪 Integer.Parse(DistinctUserName Line)

End If

OverallScore 🡪 Score1 + Score2 + Score3 + Score4

MergeSort(OverallScores)

Write OverallScores To CorrectRankLabels

BinarySearchEnabled 🡪 True

**Space Invaders**

I am now going to make my implementation on Space Invaders. This will differ from the original Space Invaders in design however the gameplay remains similar. Also, I do not intend on using sprites for this game instead I am going to use coloured pictureboxes. As I want to keep a base difficulty level in this game despite the user options as a result, I am going to make it so the enemies moving down consistently strafe left and right. By having the enemies continuously strafing the player has a greater chance to miss as the enemy could move out of the path of the missile. Also, another way I am going to make a consistent difficulty is by making the enemies move back to the top of the form each time they are killed. This differs from the original Space Invaders game as in the original game; once the enemies got hit they would just disappear from the form so the player did not have to worry about them. In my adaption once an enemy gets hit it respawns at the top of the form and just carries on moving down. Furthermore, as the players score increases in this game the enemies start moving down at an increased speed this would result in the player having to make decisions more quickly as the pace of the game would increase as the players score increases. Finally, I have made the form small on purpose to make it so there is a large concentration of enemies in a small area which would make the player feel like an enemy has nearly reached the bottom, this would result in the player taking risks so they can prevent an enemy from reaching the bottom of the form. For this game, I am not going to be designing anything on to the form except for a timer which I will use to update events taking place in the game and a ship which will be the player ship; this is because I will be dynamically generating everything else. The code for this should be similar to the code used to generate the Pipes for Flappy Bird. I will be dynamically generating the the missiles the player shoots and the enemies. As this implementation also takes into account the setting the player wants to play at, I am going to make it so that the variables to make the number of enemies and produce the missiles and vice versa rely on different quantities variables the player can change when the game loads up, otherwise if the player does not want to change the settings for the game they can play on default mode. I am going to ask the user what options they want to change at the start of the program so when the Form loads, the game is made how the user wants to play it. Settings the player can change are going to be; the number of missiles the player can shoot; the number of enemies; the speed of the missiles; speed the enemies move down at. To allow the missiles to collide with the enemies I am going to using the .NET Framework and the .Bounds and .IntersectsWith() commands as used previously. As the player can fire multiple missiles, I am going to put all the missiles into an array, as well as all of the enemies. Then using a nested FOR loop I should be able to loop through both arrays to check if a collision has occurred. As the form is going to be small in size I am not going to give the enemies the ability to fire projectiles down, as a result the game will end when the enemies reach the bottom of the form. This should conclude my implementation for my Space Invaders game.

This concludes my current method of producing all of my base games as well as their implementations.

**Data Dictionary**

**Pong:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| Speed | Store the Balls Speed | Single | 1.1 |
| rndInst | Random Instance | New Random | ‘A Random Number’ |
| BallShot | Tells Computer if Ball has been put into play or not | Boolean | True/False |
| xVel | Horizontal Velocity of Ball | Single | 1.2 |
| yVel | Vertical Velocity of Ball | Single | 1.3 |
| AIActive | Tells Computer if AI is activated or not | Boolean | True/False |
| AIChecker | Counts down the time until the AI is activated | Integer | 400 |
| AICoolDown | The Cool Down Period of the AI once it has been used | Integer | 15 |
| OriginalLocation | Stores the Ball’s initial position | Point | (1, 2) |
| CheatBox | Used in Testing | Boolean | True/False |
| plrScore | Stores the Players Score | Integer | 6 |
| compScore | Stores the Computer Score | Integer | 9 |
| r | Used to generate Random Number with the use of the .NET Framework | Random | ‘Random Number’ |
| t | Integer which acts as temporary storage | Integer | 8 |
| AdvBallSpeed | Controls the Balls Speed for Enhancements in Mad Mode | Integer | 58 |
| Count | Used to load different Enhancements in Mad Mode | Integer | 0 |
| MadModeEnabled | Used to check if Mad Mode Conditions are enabled | Boolean | True/False |
| Temp | Temporarily Stores a Value | Integer | 47 |
| CheatBoxCount | Used for Testing and will allow me to change the Count Variable | Boolean | False |
| CheatBoxScore | Used for Testing and allows me to change the score | Boolean | True |

**Bling Bling Boy (My Retro Game)**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| MoveUp | Used to allow the player to move up | Boolean | True |
| MoveDown | Used to allow the player to move down | Boolean | False |
| MoveLeft | Used to allow the player to move left | Boolean | True |
| MoveRight | Used to allow the player to move right | Boolean | False |
| PlayerSpeed | Holds the speed the player moves at | Integer | 124 |
| Lives | Holds the players lives | Integer | 24 |
| Count | Used to count the number of times an event takes place | Integer | 0 |
| Score | Holds players score | Integer | 34 |
| RandomX | Used to store a random value | New Random | 4213,32532 ie a random number |
| RandomY | Used to store a random value | New Random | 2142,325353 ie a random number |
| CurrentNumberOfEnemies | Counts the current number of enemies chasing the player | Integer | 33 |
| EnemyArray() | Holds all of the enemies so they can all be affected by code simultaneously | PictureBox | VB PictureBoxes |
| HighScore | Holds the current high score | Integer | 3425 |
| temp | Used to read high score from text file | String | Hello |
| null | Used to hold irrelevant data | String | World |
| UserInput | Allows the User to Input Data which can be logged | String | Bye |
| I | Used for Loops and iterations | Integer | 12412 |
| J | Used for Loops and Iterations | Integer | 214 |
| K | Used for Loops and Iterations | Integer | 4856 |
| PlayerName | Holds the High Score owners Username | String | dfghj |
| InvincibilityTimer | Holds the value which indicates when player can go Invincible | Integer | 847454 |

**Flappy Bird**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| Yspeed | Vertical Speed of the Bird | Integer | 3 |
| Gravity | Hold how fast the Bird falls | Integer | 5 |
| Pipe() | Array holds the number of pipes at the bottom | PictureBox | Pipe(pipe1, pipe2) |
| TopPipe() | Holds the number of pipes at the top | PictureBox | Pipe(pipe1, pipe2) |
| GapBetweenPipes | Holds the size of the gap between Top Pipes and Bottom Pipes | Integer | 500 |
| PipeSpeed | Variable Controls how fast the Pipes move | Single | 3.5 |
| Score | Stores the Users Score | Integer | 99 |
| r | Used to generate Random Number with the use of the .NET Framework | Random | ‘Random Number’ |
| Lives | Variable holds the number of lives the player has | Integer | 3 |
| InvincibilityTimer | Used as a countdown for when collision detection will be enabled and will allow the bird to have some invincibility | Integer | 1500 |
| x | Stores a Number | Integer | 348 |

**Space Invaders**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| MissileArray() | Stores the Missiles that are generated | PictureBox | MissileArray(Missile, Missile2) |
| MissileOnScreen() | Stores the number of Missiles that can be on the screen | Boolean | True/False |
| MissileNumber | Used to hold the Missiles which are made by the generator | Integer | 55 |
| EnemyArray() | Holds all of the Enemies inside an array | PictureBox | EnemyArray(enem1, enem2, enem3) |
| EnemyLocation() | Keeps track of the location of the enemies | Point | EnemyLocation((2, 1), (2, 2), (2, 3)) |
| EnemyShot | Keeps track of the bullet of each enemy | PictureBox | EnemyShot(bul1, bul2, bul3) |
| EnemyShot2() | Used to tell if an enemy has shot or not | Boolean | True/False |
| BarricadeLife() | Used to hold the 3 barricades lives | Single | 6.5 |
| EnemySpeed | Holds the speed of how fast the enemies move | Integer | 3 |
| MoveRight | Used to allow the Enemy to move Right | Boolean | True/False |
| MoveLeft | Used to allow the Enemy to move Left | Boolean | True/False |
| PlayerSpeed | Stores the speed that the player moves at | Integer | 4 |
| Score | Holds the players score | Integer | 56 |
| Lives | Holds the players lives | Integer | 45 |
| Complete | Holds the Number of Enemies killed by player | Integer | 86 |
| Level | Holds what level the player is on | Integer | 7777 |
| i | Used for loops and calculations | Integer | 987 |
| j | Used for loops and calculations | Integer | 666 |
| MaxMissileNumber | Holds the max number of missiles | Integer | 4 |
| MissileSpeed | Holds how fast the Missiles move | Integer | 78 |
| MaxEnemyNumber | Holds the Maximum number of Enemy | Integer | 0 |
| UserInput | Holds the Input provided by the User | String | Bye |
| Sound | This variable is used to run sound files | Media.SoundPlayer | ‘Sound File’ |
| EnemyOnScreen() | Used to count the number of enemies on screen and to limit it | Boolean | True/False |
| Count | Used for Loops | Integer | 1 |
| r | Used to hold a random number to 2 decimal places | Double | 2.05 |

**PacMan**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| NumOfValues | Used to count the size of Arrays | Integer | 45 |
| PacManTextFile() | Holds all the Data from PacMan data file | String | PacManTextFile(data1, data2, data3) |
| Username() | Stores all of the Username and the scores corresponding to it for all the users | String | Username(userdata1, userdata2, userdata3) |
| temporaryCheck() | Used to Check for Duplicate Data | String | temporaryCheck(data1, data2, data3) |
| JustScore() | Holds all of the Users scores | Integer | JustScore(45, 68, 85) |
| JustUserName() | Used to hold every Users Username | String | JustUserName(Username1, Username2, Username3) |
| JustUserNameVal | Holds the Username when its separated from the score | String | Username |
| CheckExists() | Used with temporaryCheck() to find and remove duplicate data | String | CheckExists(data1, data2, data3) |
| UserInput | Will hold any Input by Username | String | Apple |
| ScoreOfUser | Will hold the score of a single User | Integer | 45 |
| temp | Will hold data which needs temporary storage | String | Temporary |
| GlobalNum | Used to turn local value into global value | Integer | 79 |
| count | Used with the arrays to check and remove duplicate data | Integer | 12 |
| count2 | Used to remove duplicate data | Integer | 0 |
| Score() | Reads all the Usernames and Scores to this Array | String | Score(UserData1, UserData2, UserData3) |
| isPortal | Used to see if a tile is a portal or not | Boolean | True/False |
| isPellet | Used to check and differentiate between SuperPellet and NormalPellet | Boolean | True/False |
| isSuperPellet | Used to check if Pellet is SuperPellet | Boolean | True/False |
| didConsume | Used to check if Pellet had been Consumed or Not | Boolean | True/False |
| isGhostHouseEntrance | Stores the Nodes outside the Ghost House | Boolean | True/False |
| portalReciever | Used with isPortal to allow you to teleport to the other Portal | GameObject | ‘A object like PacMan’ |
| pacManLives | Keeps track of PacMan Lives | Static Integer | 3 |
| playerLevel | Keeps track of what level is player on | Static Integer | 1 |
| boardWidth | Sets the Width of the gameBoard | Static Integer | 65 |
| boardHeight | Sets the height of the gameBoard | Static Integer | 10 |
| totalPellets | Used to count how many pellets there are overall | Integer | 455 |
| Score | Stores the Players Score | Integer | 78 |
| ghostConsumedRunningScore | Stores the score gained from consuming Ghosts | Static Integer | 784 |
| speed | Holds the Speed that PacMan moves at | Float | 4.0f |
| canMove | Dictates weather PacMan can move or not | Boolean | True/False |
| pelletConsumed | Counts the Number of Pellets that are Conusmed | Static Integer | 45 |
| moveSpeed | Holds the speed that the Ghosts move at | Float | 45.0f |
| frightenedMoveSpeed | Holds the speed that the Ghosts move at when in Frightened mode | Float | 14.0f |
| normalMoveSpeed | Speed that the Ghosts move at and is used for Robustness | Float | 88.0f |
| consumedMoveSpeed | Stores the Speed the Ghosts move at when they’ve been consumed | Float | 8485.0f |
| previousMoveSpeed | Holds the speed that the Ghost previously moved at | Float | 65485.0f |
| pinkyReleaseTimer | Holds the time the pink ghost is released at | Integer | 544 |
| inkyReleaseTimer | Holds the time the blue ghost is released at | Integer | 484 |
| clydeReleaseTimer | Holds the time the orange ghost is released at | Integer | 8785 |
| ghostReleaseTimer | Time it takes for the ghosts to start chasing PacMan | Integer | -98 |
| frightenedModeDuration | Holds the time that frightened Mode lasts for | Integer | 874 |
| startBlinkingAt | Holds the time for when the Ghosts start blinking when in Frightened Mode | Integer | 345 |
| isInGhostHouse | Used to tell the computer if that Ghost is in the Ghost House or not | Boolean | True/False |
| canMove | Dictates weather the Ghost can move or not | Boolean | True/False |
| scatterModeTimer1 | Used to count when the initial ScatterMode takes place | Integer | 653 |
| scatterModeTimer2 | Used to count when the second iteration of ScatterMode takes place | Integer | 124 |
| scatterModeTimer3 | Used to count the timer and tell when the third iteration of ScatterMode takes place | Integer | 3453 |
| scatterModeTimer4 | Used to count when the fourth and final iteration of ScatterMode takes place | Integer | 346 |
| chaseModeTimer1 | Used to count the time for when the first iteration of chase mode takes place | Integer | 3254 |
| chaseModeTimer2 | Used to count when the second iteration of ChaseMode occurs | Integer | 3465 |
| chaseModeTimer3 | Used to count when the third iteration of ChaseMode takes place | Integer | 734 |
| modeChangeIteration | Used to count what iteration we are currently on | Integer | 5467 |
| modeChageTimer | Used as a timer which counts up and is what causes the different modes to occur as well as the iteration to chage | Float | 60.0f |
| frightenedModeTimer | Used to count the duration of FrightenedMode | Float | 922.0f |
| blinkTimer | Counts how long the Ghost blinks | Float | 14.0f |
| count1 | Used to track how many times an event can take place | Integer | 213 |
| count2 | Used to check if the berry has been consumed or not | Integer | 1 |
| berryPoint | Holds the amount of points gained from consuming a berry | Integer | 3512 |
| berryTimer | Keeps track of the time the berry is visible | Float | 34.0f |
| privCounter | Used to make berry disappear if it has been visible for too long | Float | 34.0f |
| isConsumed | Checks if Berry has been Consumed or not | Boolean | True/False |
| isVisible | Checks to see if berry can be seen or not | Boolean | True/False |
| canTick | Used to tell timer when to start and stop | Boolean | True/False |
| modeChecker | Used to tell what mode Blue Ghost is on in PacMan multiple map implementation | Integer | 6554 |

**Overall High Score**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| PacManUserNamePath | Holds the Path to the PacMan Username and Score File | String | E://path |
| MyGameUserNamePath | Holds the Path to Bling Bling Boys Username and Score File | String | E://path |
| SpaceInvadersUserNamePath | Holds the path to the Space Invaders Username and Score File | String | E://path |
| FlappyBirdUserNamePath | Holds the path to the Flappy Bird Username and Score File | String | E://path |
| PacManUserNameArray() | Holds all of the Usernames from PacMan in this array | String | PacManUserNameArray(user1, user2, user 3) |
| MyGameUserNameArray() | Holds all of the Usernames from Bling Bling Boy in this array | String | MyGameUserNameArray(user1, user2, user3) |
| SpaceInvadersUserNameArray() | Holds all of the Usernames from Space Invaders in this Array | String | SpaceInvadersUserNameArray(user1, user2, user3) |
| FlappyBirdUserNameArray() | Holds all of the Usernames from Flappy Bird in this Array | String | FlappyBirdUserNameArray(user1, user2, user3) |
| PacManList() | Used to pass values between and too convert data into array also used to remove repeated data | List(Of String) | PacManList(user1, score1) |
| MyGameList() | Used to pass values between and too convert data into array, also used to remove repeated data | List(Of String) | MyGameList(user1, score1) |
| SpaceInvadersList() | Used to pass values between and too convert the data into array also used to remove repeated data | List(Of String) | SpaceInvadersList(user1, score1) |
| FlappyBirdList() | Used to pass values between and too convert the data into array also used to remove repeated data | List(Of String) | FlappyBirdList(user1, score1) |
| UserNamesList()() | Used to store all the usernames from each game and to remove repeated Usernames | List(Of String) | UserNamesList(user1, user2, user3) |
| UserNames() | Stores all the Usernames into array | String | UserNames(user1, user2, user3) |
| UserNamesWithSumList()() | Used to store all the scores alongside the sum of that Users scores | List(Of String) | UserNamesWithSumList(user1, user2)(score1, score2) |
| DistinctUserNamesWithSum | Used to remove repeated data from UserNamesWithSumList | List(Of String) | DistinctUserNamesWithSum((User1, Score1), (User2, Score2)) |
| UserNamesWithSum() | Stores all usernames with their score inside this array | String | UserNamesWithSum((user1, score1), (user2, score2)) |
| PlainUserName | Used to pass each UserName through the use of loops where required | String | User1 |
| JustScoreList()() | List stores just the scores of each user | List(Of Integer) | JustScoreList(score1, score2) |
| JustScore() | Array will stores just the scores of each user | Integer | JustScore(3421, 21421, 452) |
| UserInput | Used to pass user inputs into the program | String | Hello World |
| ScoreOfUser | Stores the Score a Single User | Integer | 28748 |
| pacIndex | Provides Index to use when going through PacMan List | Integer | 435 |
| myGameIndex | Provides Index when going through MyGame List | Integer | 35 |
| spaceInvadersIndex | Provides Index for when going through Space Invaders List | Integer | 34 |
| flappyBirdIndex | Provides Index for when going through Flappy Bird List | Integer | 4532 |
| pacSplit() | Used to store data split from PacMan List | String | pacSplit(user1, score1) |
| mygameSplit() | Used to store data split from My Game List | String | mygameSplit(user1, score1) |
| flappyBirdSplit() | Used to store data split from Flappy Bird List | String | flappyBirdSplit(user1, score1) |
| sum | Used to calculate the sum of each score | Integer | 143 |
| finalText | Stores the structure of how the final data is stored so the username alongside that users overall score | String | User1, OverallScore1 |

**VB Individual Game High Score Screen**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Purpose | Type | Example Data |
| UsernameArray() | Stores Usernames with Scores and is read from the text file which contains all of this data | String | Hellow |
| CurrentUserScore/Score2 | Used to transfer the current User Score between Forms | Integer | 58484 |
| NumOfValues | Used as a counter for the Array | Integer | 878965 |
| JustScore() | Only stores the scores of all users and not the Usernnames | String | World |
| JustUserName() | Only stores the usernames of the users and not the scores | String | Bye |
| ScoreArray() | Used to read score file | String | World |
| CheckExists() | Used to check for redundant data | String | Again |
| Count | Used to tell if data has been repeated or not | Integer | 848888 |

**Test Strategy**

First of all, I am going to be using black box testing to check if my Compilation meets my initial objectives (black box testing is a method of testing where the program is checked to see if it seems to work by just using the program and not looking into the internal structures that make up the program), ensuring my project meets the requirements set and thus satisfying the needs of the end users. For each objective I am going to provide the objective number and a brief description of the objective being tested, the evidence will be in the form of videos which will be uploaded onto YouTube. The video will be one video going through all of the tests in the order they appear on the table.

After I have tested my project with black box testing, I am then going to use white box testing which is a testing method which tests the internal structures of a program. Using white box testing I am going to be testing my Merge Sort and Binary Search algorithms, to make sure they are working as intended. I am going to do this as Merge Sort and Binary Search are central to my objectives and the purpose of the game, as the scoring system will provide the competitive high score which was one of the many reasons why arcade gaming was popular as many people wanted to be the top player on the games they played and to do this they had to achieve the highest score in the game. To ensure they algorithms are working properly as intended, I am going to test them with a dry run the algorithms and include evidence in the form of screenshots.

Finally, to end my testing I am going to allow my peers to use the program, once I have fully developed and tested it. I am going to do this as this has the potential that other users might discover errors which I might not have found as the actual user is more likely to use a variety of data like Normal, Errornous or Boundary data. This might be because I know how my game functions whilst a new user will not know it as much as me, this might result in the new user using the program in a way I wouldn’t anticipate.

While developing my program I also tested each part to see if it was working in its intended manner and as a result I am confident that my Retro Compilation will pass the tests presented to it.

**Technical Solution**

The full Code can be found in the Appendix. *All the code mentioned is mine and there is no thid party code. Any adaptaions of tutorials or taken code have been mentioned where used*. The variables used were identified in the Data Dictionary. The evidence for the technical solution can be found at the start of the Black Box testing. The evidence is provided in the form of YouTube Videos. Throughout my project I used a range of Group A, B and C algorithms and coding styles, these can be seen below:

***Group A:***

|  |  |
| --- | --- |
| Algorithm Name | Location |
| Tree Traversal (*Through my adaption of The Weekly Coders PacMan AI Pathfinding code* in PacMan Multiple Map Implementation) | See Ghost Script (In PacMan Implementation code Listing) for just Ghost AI see Implementation Figure 1 |
| List Operations | See Code for Overall High Score Code listing as used throughout in Overall High Score Screen |
| Recursive Algorithms | See Merge Sort Algorithm or Implementation Figure 2 |
| Complex User Defined Algorithms (PacMan Path Finding Adaptation and High Scoring System) | See Ghost Script (In PacMan Implementation code listing) and See Overall High Score Code or Implementation Figure 1 and 3 |
| Merge Sort | See Impelementation Figure 2 |
| Files Organised for Direct Access | See all paths for each game. Used throughout in each High Scoring System for every game except Pong. Also used in Overall High Score Code |

***Group B:***

|  |  |
| --- | --- |
| Algorithm Name | Location |
| Multi-Dimensional Array (Map-Maker) | See Map Maker Code Lisitng. Used in Map Maker as each point refers to what tile it is and its corresponding location of tile map. Output when file is saved and read is a 2D array. |
| Binary Search | See Implementation Figure 4 |
| Text Files | Used throughout in each High Scoring System for every game except Pong. Also used in Overall High Score Code as well. |
| Simple User Defined Algorithms (Get Sum of Single User/Barricade/Enemy Shot Movement ) | See Implementation Figure:  5, 6 and 7 |
| Simple Mathematical Formula Used (Trigonometry) | See Pong Coding or Implementation Figure 8 also see page 39 for design of this |
| Writing and Reading Files | Used throughout in each High Scoring System for every game except Pong. Also used in Overall High Score Code as well. |

Group C:

|  |  |
| --- | --- |
| Algorithm Name | Location |
| Single Dimensional Array | Used throughout code |
| Appropriate Choice of Simple Data Types | Used throughout code |
| Simple Mathematical Calculations (Addition/Subtraction/Multiplication/Division used to achieve values using Logic for instance in Score Compiling) | Used throughout code, especially see code for getting sum of all scores in Overall Score System |

Implementation Figure 1:

PacMan Adapted Ghost AI Code Only

Vector2 GetRedGhostTargetTile()

{

//This Ghost will follow PacMan as Normal

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y)); //Rounds PacMans position to an Integer

return targetTile; //Function outputs targetTile

}

Vector2 GetPinkGhostTargetTile()

{

//This Ghost will follow PacMan until it gets 2 tiles away from PacMan in which case it will return to the center of the Map (towards Ghost House)

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Ghost to PacMan

if (distance > 2) //Checks to see how far away the Ghost is from PacMan

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y)); //Rounds PacMans position to an Integer

}

else if (distance <= 2) //If the inital requirement isn't fulfilled then the following code is ran

{

targetTile = GameObject.Find("pellet (206)").transform.localPosition; //Makes the Ghost go back to its starting position

}

return targetTile; //Function outputs targetTile

}

Vector2 GetBlueGhostTargetTile()

{

//Takes into account where the Pink Ghost is and doubles the distance however if the Pink Ghost is at its starting position then this Ghost takes into account the Red Ghosts Position

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Gets Pacmans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Stores PacMan X coordinate in this variable

int pacmanPositionY = Mathf.RoundToInt(pacManPosition.y); //Stores PacMan Y cooridinate in this variable

Vector2 pacManTile = new Vector2(pacManPositionX, pacmanPositionY); //Get PacMan Tile

Vector2 targetTile = pacManTile + (2 \* pacManOrientation); //Create a TargetTile for where PacMan is

Vector2 tempPinkPosition = GameObject.Find("Pink Ghost").transform.localPosition; //Create a temp variable for Pinks position

if (tempPinkPosition.x != Mathf.RoundToInt(GameObject.Find("pellet (206)").transform.localPosition.x) & (tempPinkPosition.y != Mathf.RoundToInt(GameObject.Find("pellet (206)").transform.localPosition.y))) //Checks to see if the Pink Ghost is at its starting position or not

{

int PinkPositionX = Mathf.RoundToInt(tempPinkPosition.x); //Stores Pinks x position in this variable

int PinkPositionY = Mathf.RoundToInt(tempPinkPosition.y); //Stores Pinks y position in this variable

tempPinkPosition = new Vector2(PinkPositionX, PinkPositionY); //Stores Pinks coordinate in a new vector2 variable

float distance = GetDistanceApart(tempPinkPosition, targetTile); //Gets the distance apart between Pinks position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempPinkPosition.x + distance, tempPinkPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

modeChecker = 1;

return targetTile;

}

else //If the Pink Ghost is at its starting position then the following code is run

{

Vector2 tempBlinkyPosition = GameObject.Find("Red\_Ghost").transform.localPosition; //Create a temp variable for Blinky position

int blinkyPositionX = Mathf.RoundToInt(tempBlinkyPosition.x); //Stores Blinkys x position in this variable

int blinkyPositionY = Mathf.RoundToInt(tempBlinkyPosition.y); //Stores Blinkys y position in this variable

tempBlinkyPosition = new Vector2(blinkyPositionX, blinkyPositionY); //Stores Blinkys coordinate in a new vector2 variable

float distance = GetDistanceApart(tempBlinkyPosition, targetTile); //Gets the distance apart between blinkys position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempBlinkyPosition.x + distance, tempBlinkyPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

modeChecker = 0;

return targetTile; //Outputs the Target Tile

}

}

Vector2 GetOrangeGhostTargetTile()

{

//Always tries to get in front of PacMan or run away from PacMan dependent on which mode the Blue Ghost is on

Vector2 temp = Vector2.zero; //Used to temporarly hold a value

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Stores PacMans Orientation

if (modeChecker == 0)

{

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Ghost to PacMan

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

if (distance > 8) //If Distance is greater than 8 then the following code is ran

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y));

temp = targetTile;

return targetTile;

}

else //If Distance is less than 8 then the following code is ran

{

targetTile = homeNode.transform.position; //Makes the TargetTile the Home Node

temp = targetTile;

return targetTile; //Function outputs targetTile

}

}

else if (modeChecker == 1)

{

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Rounds PacMans x coordinate to a whole number

int pacManPositionY = Mathf.RoundToInt(pacManPosition.y); //Rounds PacMans y coordinate to a whole number

Vector2 pacManTile = new Vector2(pacManPositionX, pacManPositionY);

Vector2 targetTile = pacManTile + (4 \* pacManOrientation); //Multiplies all of the vector2 by 4 so both x and y coordinate seperatly and then adds pacManTile to it so Pinky is always 4 tiles ahead of PacMan

temp = targetTile;

return targetTile; //Function outputs targetTile

}

return temp; //Outputs the value

}

Implementation Figure 2:

Merge Sort Algorithm

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length – 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Implementation Figure 3:

Overall Score Algorithm

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

File.WriteAllText("Username.txt", "") 'Clears the Username Text File when the Program is Launched

File.WriteAllText("Score.txt", "") 'Clears the Score Text File When the Program is Launched

'Code below calls sub routine by the same name and passes the UserName array through it

StoreUserName(PacManUserNameArray)

StoreUserName(MyGameUserNameArray)

StoreUserName(SpaceInvadersUserNameArray)

StoreUserName(FlappyBirdUserNameArray)

'Gets the Program to get the UserName of each player and get the Sum of each user with the GetSumOfSinglePlayer Sub Routine

For i = 0 To UserNames.Length - 1

PlainUserName = UserNames(i)

GetSumofSingleUser(PlainUserName)

Next

UserNamesWthSum = DistinctUserNamesWithSum.ToArray() 'Makes a New Array with the Distinct Values in only

WriteToTextFileUserNames() 'Calls Sub Routine by the same name here

GetValue() 'Calls Sub Routine by the same name here

'FOR Loop Below goes through each value in just score and writes it to the Score text file

For i = 0 To JustScore.Length - 1

Using Writer As New StreamWriter("Score.txt", True)

Writer.WriteLine(JustScore(i))

End Using

Next

MergeSort(JustScore) 'Sorts the Data with the Merge Sort Algorithm by passing through the JustScore array

WriteToLabels() 'Calls the SUb Routine by the same name here

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Launches the Binary Search Algorithm

End Sub

Private Sub butFind\_Click(sender As Object, e As EventArgs) Handles butFind.Click

UserInput = InputBox("What is Your UserName?") 'Asks the Player who wants to know their score, stores their username in UserInput

GetValueDependentonUser() 'Calls SUb Routine by the same name here

MessageBox.Show("Your Score is " & ScoreOfUser) 'Tells the player their score through a message box

End Sub

Private Sub GetSumofSingleUser(ByVal WrittenUserNames As String)

'Adds the data from each games Username text file to the correct list

PacManList.AddRange(IO.File.ReadAllLines(PacManUserNamePath))

MyGameList.AddRange(IO.File.ReadAllLines(MyGameUsernamePath))

SpaceInvadersList.AddRange(IO.File.ReadAllLines(SpaceInvadersUsernamePath))

FlappyBirdList.AddRange(IO.File.ReadLines(FlappyBirdUsernamePath))

'Provides Indexs to use when going through lists

Dim pacIndex As Integer = 0

Dim myGameIndex As Integer = 0

Dim spaceinvadersIndex As Integer = 0

Dim flappybirdIndex As Integer = 0

'Each FOR Loop for each game finds all the Username and Score from their Lists

For i As Integer = 0 To PacManList.Count - 1

If PacManList(i).StartsWith(WrittenUserNames) Then

pacIndex = i

End If

Next

For i As Integer = 0 To MyGameList.Count - 1

If MyGameList(i).StartsWith(WrittenUserNames) Then

myGameIndex = i

End If

Next

For i As Integer = 0 To SpaceInvadersList.Count - 1

If SpaceInvadersList(i).StartsWith((WrittenUserNames)) Then

spaceinvadersIndex = i

End If

Next

For i As Integer = 0 To FlappyBirdList.Count - 1

If FlappyBirdList(i).StartsWith((WrittenUserNames)) Then

flappybirdIndex = i

End If

Next

'Seperated the data into two parts when a comma is present

Dim pacSplit() As String = Split(PacManList(pacIndex), ",")

Dim mygameSplit() As String = Split(MyGameList(myGameIndex), ",")

Dim spaceinvadersSplit() As String = Split(SpaceInvadersList(spaceinvadersIndex), ",")

Dim flappybirdSplit() As String = Split(FlappyBirdList(flappybirdIndex), ",")

'Finds the Sum of all of the Scores from each game

Dim sum As Integer = Integer.Parse(Trim(pacSplit(1))) + Integer.Parse(Trim(mygameSplit(1))) + Integer.Parse(Trim(spaceinvadersSplit(1))) + Integer.Parse(Trim(flappybirdSplit(1)))

'Used to reduce bugs by removing any empty usernames

If WrittenUserNames <> "" Then 'If the Username field isn't empty then we write it otherwise we don't

Dim finaltext As String = (WrittenUserNames & ", " & sum.ToString) 'Stores the Username of the user alongside the sum of their scores

UserNamesWithSumList.Add(finaltext) 'Adds the Username and the sum of each user to the List

DistinctUserNamesWithSum = UserNamesWithSumList.Distinct().ToList 'Removes Repeated Entrys

End If

End Sub

Private Sub StoreUserName(ByVal arr() As String)

'Adds all the Username to the List below

For i = 0 To arr.Length - 1

UserNamesList.Add(arr(i))

Next

'Removes Duplicated values when moving data to array

UserNames = UserNamesList.Distinct().ToArray

End Sub

Private Sub WriteToTextFileUserNames()

'Writes all data to the UserName Text File from the DistinctUserNameWithSum List

For i = 0 To DistinctUserNamesWithSum.Count - 1

Using Writer As New StreamWriter("Username.txt", True)

Writer.WriteLine(DistinctUserNamesWithSum(i))

End Using

Next

End Sub

Private Sub GetValue()

'Goes through each UserName and adds the sum of their scores to JustScore

For Each Data As String In UserNamesWthSum

For i = 0 To UserNames.Length - 1

PlainUserName = UserNames(i)

If Data.StartsWith(PlainUserName & ",") Then

JustScoreList.Add(Integer.Parse(Data.Split(","c)(1)))

End If

Next

Next

JustScore = JustScoreList.ToArray() 'Converts List to Array

End Sub

Private Function GetValueDependentonUser()

'If a User is looking for their score and they aren't on the top 5 they can look their score up here

For Each Data As String In UserNamesWthSum

If Data.StartsWith(UserInput & ",") Then 'Checks to see if data has the users provided username

ScoreOfUser = Integer.Parse(Data.Split(","c)(1)) 'Splits the Data so ScoreOfUser only contains the Score of the User

End If

Next

Return ScoreOfUser

End Function

Implementation Figure 4:

Binary Search Algorithm

Sub BinarySearch()

'Binary Search Algorithm can be Seen Below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score?")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

Implementation Figure 5: Get Sum Of Single User from Overall High Score Screen Code

Private Sub GetSumofSingleUser(ByVal WrittenUserNames As String)

'Adds the data from each games Username text file to the correct list

PacManList.AddRange(IO.File.ReadAllLines(PacManUserNamePath))

MyGameList.AddRange(IO.File.ReadAllLines(MyGameUsernamePath))

SpaceInvadersList.AddRange(IO.File.ReadAllLines(SpaceInvadersUsernamePath))

FlappyBirdList.AddRange(IO.File.ReadLines(FlappyBirdUsernamePath))

'Provides Indexs to use when going through lists

Dim pacIndex As Integer = 0

Dim myGameIndex As Integer = 0

Dim spaceinvadersIndex As Integer = 0

Dim flappybirdIndex As Integer = 0

'Each FOR Loop for each game finds all the Username and Score from their Lists

For i As Integer = 0 To PacManList.Count - 1

If PacManList(i).StartsWith(WrittenUserNames) Then

pacIndex = i

End If

Next

For i As Integer = 0 To MyGameList.Count - 1

If MyGameList(i).StartsWith(WrittenUserNames) Then

myGameIndex = i

End If

Next

For i As Integer = 0 To SpaceInvadersList.Count - 1

If SpaceInvadersList(i).StartsWith((WrittenUserNames)) Then

spaceinvadersIndex = i

End If

Next

For i As Integer = 0 To FlappyBirdList.Count - 1

If FlappyBirdList(i).StartsWith((WrittenUserNames)) Then

flappybirdIndex = i

End If

Next

'Seperated the data into two parts when a comma is present

Dim pacSplit() As String = Split(PacManList(pacIndex), ",")

Dim mygameSplit() As String = Split(MyGameList(myGameIndex), ",")

Dim spaceinvadersSplit() As String = Split(SpaceInvadersList(spaceinvadersIndex), ",")

Dim flappybirdSplit() As String = Split(FlappyBirdList(flappybirdIndex), ",")

'Finds the Sum of all of the Scores from each game

Dim sum As Integer = Integer.Parse(Trim(pacSplit(1))) + Integer.Parse(Trim(mygameSplit(1))) + Integer.Parse(Trim(spaceinvadersSplit(1))) + Integer.Parse(Trim(flappybirdSplit(1)))

'Used to reduce bugs by removing any empty usernames

If WrittenUserNames <> "" Then 'If the Username field isn't empty then we write it otherwise we don't

Dim finaltext As String = (WrittenUserNames & ", " & sum.ToString) 'Stores the Username of the user alongside the sum of their scores

UserNamesWithSumList.Add(finaltext) 'Adds the Username and the sum of each user to the List

DistinctUserNamesWithSum = UserNamesWithSumList.Distinct().ToList 'Removes Repeated Entrys

End If

Implementation Figure 6: Barricade from Space Invaders Main Game

End Sub

Private Sub Barricade()

Dim j As Integer 'Used for Loops

Dim BarricadeArray(2) As PictureBox 'Stores all the Barricades in an Array

'The Code below loops through each picturebox in the form and adds it to the array if it has the barricade tag

For i = 0 To 2

For Each Picturebox In Me.Controls

If Picturebox.Tag = ("Barricade") Then

BarricadeArray(i) = Picturebox

i += 1

ReDim Preserve BarricadeArray(i)

End If

Next

Next

'Nested FOR Loop checks to see if the player shot the barricade, if they did then the barricade loses a life and the image of the barricade changes accordingly

For j = 0 To 2

For i = 0 To 1

If BarricadeLife(j) > 0 AndAlso BarricadeArray(j).Visible = True Then

If (pbMissile.Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

pbMissile.Visible = False

MissleShoot = False

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

BarricadeLife(j) = BarricadeLife(j) - 1

End If

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

Next

Next

'Nested FOR Loops checks to see if a Enemy Missile has hit the barricade, this causes the barricade to lose one life and the barricade image changes accordingly

For j = 0 To 2

For i = 0 To 32

If (EnemyShot(i).Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

Me.Controls.Remove(EnemyShot(i))

EnemyShot(i).Location = EnemyArray(i).Location

BarricadeLife(j) = BarricadeLife(j) - 0.25

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

End If

Next

Next

End Sub

Implementation Figure 7: EnemyShotMovement from Space Invaders Main Game

Private Sub EnemyShotMovement()

'Variables below are used for Loops

Dim j As Integer

'Loops through to check if the enemy has shot if it hasn't it has the chance to become true through the random

For i = 0 To 32

If EnemyShot2(i) = False Then

j = CInt(Int((1000 \* Rnd()) + 1))

If j = 1000 Then

EnemyShot2(i) = True

End If

End If

Next

'Checks to see if the Enemy has shot if it has then the bullet becomes visible and the missile moves down

For i = 0 To 32

If EnemyShot2(i) = True Then

EnemyShot(i).Visible = True

EnemyShot(i).Top += 20

End If

If (EnemyShot(i).Bounds.IntersectsWith(pbPlayer.Bounds)) Then 'If a bullet hits the player then following code is ran

Me.Controls.Remove(EnemyShot(i)) 'Enemy Shot is removed from the form

EnemyShot(i).Location = EnemyArray(i).Location 'Moves the bullet back to the position of the enemy

Lives = Lives - 1 'Player loses one life

'Code below checks the life the player is on and changes the UI accordingly

If Lives = 2 Then

pbLife.Visible = False

ElseIf Lives = 1 Then

pbLife2.Visible = False

End If

End If

'If the Shot goes out of the Form then the Shot is removed and is moved back to its original location

If EnemyShot(i).Top > Me.Height Then

EnemyShot2(i) = False

EnemyShot(i).Visible = False

EnemyShot(i).Location = EnemyLocation(i)

EnemyShot(i).Top = EnemyShot(i).Top + 30

EnemyShot(i).Left = EnemyShot(i).Left + 15

End If

Next

If Lives = 0 Then 'If Player has 0 lives then the follwing code is ran

pbLife3.Visible = False 'Life Indicator is made invisible

Form2.Score2 = Score 'Player Score is transferred from this form to form2

timerGame.Stop() 'The Main Timer Stops

timerEnemy.Stop() 'The Enemy Timer Stops

MsgBox("Game Over Your Score is " & Score) 'MessageBox appears which states game over with the players score

Form2.Show() 'Open Form2

Me.Hide() 'Closes this Form

End If

End Sub

Implementation Figure 8:

Simple Mathematical Formula (Trigonometry)

Dim xVel As Single = Math.Cos(rndInst.Next(5, 10)) \* Speed 'Velcoity of the Ball Horizontally (on the x-axis)

Dim yVel As Single = Math.Sin(rndInst.Next(5, 10)) \* Speed 'Velocity of the Ball Vertically (on the y-axis)

For Implementation Figure 6, a explanation can be seen on page 39 with the triangle.

This concludes all the specific Grouped Algorithms and Methods I used.

**Testing**

<https://www.youtube.com/watch?v=cpXzxkwo9RE&feature=youtu.be>

***Space Invaders***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamp |
| 1 | 1.1.1 | Player will have a ship which they can move left/right | Player moves ship left/right with arrows key | PASS | 4:49 |
| 2 | 1.1.2 | Player should only be able to fire one missiles at a time | Player cannot fire more than one missile everytime | PASS | 4:51 |
| 3 | 1.1.3 | There is cover for the player which the player can move underneath. The cover should take the missiles that are fired from enemies/player | Player can move underneath cover which will save the player from damage from enemy missiles | PASS | 4:56 , 5:11 |
| 4 | 1.2.1 | Three Rows of Enemies which can fire down | Enemy can fire down | PASS | 5:04 |
| 5 | 1.2.2 | Enemies move left all the way then move down and then move right all the way then move down and they should follow this pattern until they reach the bottom of the form. | Enemies move in the pattern explained. | PASS | 4:46 – 4: 51 |
| 6 | 1.2.3 | Each row of enemy provides a different amount of points to the player | Player gains a different amount of points when an enemy from each row is hit | PASS | 4:47 – 5:00 |
| 7 | 1.2.4 | The enemies should be arranged in a manner where they are hierarchal where the highest points are gained from hitting an enemy at the top and lower amounts are gained by hitting enemies lower down. | Player gains a different amount of points when an enemy is hit, with the scores decreasing with the lower rows | PASS | 4:47 |
| 8 | 1.3.1 | Player score is saved in a text file alongside the username | The players score and username is saved to the score text file | PASS | 6:24  See Appendix |
| 9 | 1.4.1 | Menu possesses buttons | Appropriate num of buttons available on the menu | PASS | 4:37 |
| 10 | 1.4.2 | When the player presses Start Game the game is started either loading up the normal Space Invaders or my implementation of it dependent on the UserInput | Buttons work as desired and explained | PASS | 4:37 – 4:44 and 6:46 – 7:32 |
| 11 | 1.4.3 | The Custom Game will allow the user to change some options of the game | Player can play the game as they desire | PASS | 7:10 – 7:31 |
| 12 | 1.4.3.1 | The Custom Game maintains a Level of Difficulty | Custom Game possess a default mode as well as a base difficulty on lowest settings | PASS | 7:59 – 8:00 and 8:44 – 8:46 |
| 13 | 1.4.4 | The Exit Button Closes the Program | Button works as desired | PASS | 8:53 |
| 14 | 1.5.1 | The player should have three lives, once the player runs out of lives the game ends | A Lives system is implemented and works | PASS | 4:45 – 6:12 |
| 15 | 1.5.2 | When the player dies a messagebox should appear which will tell the player “Game Over” as well as displaying the score to the user | Messagebox tells player that game has ended as well as showing their score | PASS | 6:12 |
| 16 | 1.5.3 | After the player dies the player should be asked for their Username | User can enter their Username to save their Score | PASS | 6:16 |
| 17 | 1.5.4 | Once the User saves their Username it should be saved to the text file alongside the users score | User can save their details to the score text file | PASS | 6:24 |

***PacMan***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 2.1.1 | Ghosts should follow their original AI algorithms (Chase Mode, Scatter Mode) | Ghosts work as they would in the original PacMan | PASS | 13:19 – 15:39 |
| 2 | 2.1.1.1 | Red Ghost always tracks PacMans location and moves to that location in effect chasing PacMans (in Chase Mode) | Red Ghost chases PacMan by following PacMan | PASS | 13:19 – 15:39 |
| 3 | 2.1.1.2 | The Pink Ghost tries to get in front of PacMan (By 4 tiles) (In Chase Mode) | Pink Ghost takes alternative routes to get in front of PacMan | PASS | 13:19 – 15:39 |
| 4 | 2.1.1.3 | The Blue Ghost takes into account PacMans location as well as the Red Ghosts and tries to move two tiles in front of PacMan (In Chase Mode) | Blue Ghost tries to get in front of PacMan by also considering the Red Ghosts position | PASS | 13:19 – 15:39 |
| 5 | 2.1.1.4 | The Orange Ghost acts as a bluff and will chase PacMan using the same AI as the Red Ghost but once it gets too close it moves back to its Home Node (in Chase Mode) | Orange Ghost goes to PacMan then runs away once it gets too close | PASS | 13:19 – 15:39 |
| 6 | 2.2.1 | The players score increases by collecting pellets | Players score increases each time a pellet is eaten | PASS | 13:19 – 15:39 |
| 7 | 2.2.2 | Player Score also increases each time a berry is eaten | Collecting Berries increases Score | PASS | 17:20 – 17:24 |
| 8 | 2.3.1 | Menu has a button which will start the game with the chosen map | Player can Play Game with a chosen Map | PASS | 16:40 – 16:43 |
| 9 | 2.3.2 | Menu has a button which the user can use to change the Map they want to play on | Player can select the Map they want | PASS | 16:40 – 16:43 |
| 10 | 2.3.3 | Menu contains a High Score Button which when pressed will take the player to the high score screen | Player can enter the High Score Screen from Menu | PASS | 16:01 – 16:02 |
| 11 | 2.3.4 | Exit Button will present on the Menu which will close the program upon pressing it | Exit Button will close the program | PASS | 12:52 |
| 12 | 2.4.1 | On the menu the player will have the ability to choose the Map they want to play on | Player can choose the Map they want to play on | PASS | 16:40 – 16:43 |
| 13 | 2.5.1 | The player can control an animated PacMan Sprite. PacMan should also change the animation dependent on the direction it is travelling in | PacMan is animated | PASS | 13:19 – 15:39 |
| 14 | 2.5.2 | When PacMan hits a wall he should stop. When PacMan eats a super pellet the ghosts should enter frightened mode and when PacMan eats a ghost in frightened Mode the ghost should return to the ghost house at a high speed. When the player eats a berry or any pellet the score should increase appropriately | PacMan should interact with the environment | PASS | 13:19 – 15:39 |
| 15 | 2.5.3 | When PacMan enters a Portal he should come out of the opposing Portal | PacMan should interact with the portals appropriately | PASS | 14:14 – 14:17 |
| 16 | 2.6.1 | When PacMan loses all his lives the player loses and the players score is saved | The Players score gets saved alongside their username when PacMan runs out of lives | PASS | 15:42,  16:04 – 16:14 |

*Please Note: There is a point in my testing video where the PacMan game stops responding (15:59) when I try to close it, this is due to the Recording Software and my Machine not being able to handle the interrupt thus resulting in the game showcasing that it’s not responding.*

***Pong***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 3.1.1 | The ball is contained in the form and can bounce of the walls | The ball can bounce of the top and bottom walls | PASS | 9:06 – 9:23 |
| 2 | 3.1.2 | If the ball passes either the player or the computer paddle the other player | Both computer and player gain points appropriately | PASS | 9:11 – 9:12 |
| 3 | 3.2.1 | Two paddles on the form, one is controlled by the computer the other is controlled by the player | The player can move the paddle and the computer paddle also moves trying to hit the ball | PASS | 9:06 – 9:23 |
| 4 | 3.3.1 | The player can access Normal Mode from the Menu | The player can play the game in normal Mode | PASS | 9:00 – 9:03 |
| 5 | 3.3.2 | The player can access Mad Mode | The Player can play in Mad Mode | PASS | 10:06 – 10:10 |
| 6 | 3.4.1 | The ball will move randomly to try to trick the player | The Ball acts randomly | PASS | 10:45 – 10:55 |
| 7 | 3.4.1.1 | The AI looks at the players score and acts differently by increasing the difficulty | The ball changes how it acts according to the score | PASS | 10:10 – 12:29 |
| 8 | 3.4.1.2 | The AI will alternate with how it plays, for instance the ball the player receives won’t always be a straight ball there will be a chance for the ball to change how it was coming | The ball has a chance to change how it comes towards the ball | PASS | 10:10 – 12:29 |
| 9 | 3.4.1.3 | The ball has the ability to gain speed | The ball can increase its speed | PASS | 10:10 – 12:29 |
| 10 | 3.4.1.4 | The ball has the ability to slow down | The ball can decrease its speed | PASS | 11:27 – 11:33 |
| 11 | 3.5.1 | The ball rebounds appropriately when it hits a paddle | The ball rebounds in a realistic manner | PASS | 9:40 – 10:03 |
| 12 | 3.5.2 | The program detects when the ball passes a paddle and awards the appropriate point. The balls position should be reset | The game resets when either player or computer gains a point, score needs to stay the same | PASS | 10:00 – 10:04 |
| 13 | 3.5.3 | The ball rebounds of the walls in a realistic manner | The ball acts appropriately when it hits either the top or the bottom walls | PASS | 9:05 – 9:19 |
| 14 | 3.6.1 | Ball rebounds of the balls with the use of trigonometry | Ball bounces of walls and paddles with the use of trigonometry | PASS | See design & appendix:  9:05 – 9:19 |
| 15 | 3.6.2 | Ball should inverse its speed and bounce of at an appropriate when it rebounds of a paddle. | Ball rebounds of paddles in an appropriate and realistic manner | PASS | 9:05 - 9:19 |
| 16 | 3.7.1 | There is no high score system | Scores aren’t saved | PASS | N/A as no high score to show |
| 17 | 3.7.2 | The first to 10 wins the game weather it’s the player or the computer | When the score is 10, whoever has the score wins and the player gets an appropriate message to tell them weather they won or not | PASS | 11:17 – 11:19 |
| 18 | 3.7.3 | The player has the option to carry on playing once they have reached 10 points | The player should be able to carry on playing even after they have 10 points | PASS | 11:17 – 11:19 |

***Flappy Bird***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 4.1.1 | The player controls a bird which is affected by gravity resulting in it falling if no user input is detected | The bird go down if there is no user input | PASS | 2:23 – 3:12 |
| 2 | 4.1.2 | The longer the bird falls the quicker it falls | The speed of the bird increases as it falls | PASS | 2:35 – 2:37 |
| 3 | 4.1.3 | The player can make the bird go up by pressing a key | The player has the ability to make the bird hop | PASS | 2:23 – 3:12 |
| 4 | 4.2.1 | There can only be two pipes on the screen at one given time | There can only be 2 pipes bottom (max) on the screen at one given time | PASS | 2:23 – 3:12 |
| 5 | 4.2.2 | Each Pipe is unique as each Pipe has a different length thus making the gap between each Pipe different | The gap between each pipe is | PASS | 2:23 – 3:12 |
| 6 | 4.2.3 | There should be two pipes where there is a pipe above each two pipes at the bottom | There should be two pipes at the bottom with two pipes directly above | PASS | 2:23 – 3:12 |
| 7 | 4.3.1 | The players score should increment each time the player goes through a pipe | The players score goes up each time the bird passes through a pipe | PASS | 2:23 – 3:12 |
| 8 | 4.3.2 | When the player dies the player scores is saved and the bird is reset to its original locations | When player dies the game is reset and the players score is saved | PASS | 3:12 – 3:21 |
| 9 | 4.4.1 | Random Events occur when a random number is generated, the event corresponding with that number takes place | Random Events take place when a random event is generated | PASS | 3:40 – 4:15 |
| 10 | 4.4.2 | As the players score increases the chance of a random event increases | As the players score increases there is a higher chance for a random event to take place | PASS | 3:40 – 4:15 |
| 11 | 4.4.3 | Random Events include the pace of the game either increasing or decreasing or the bird becoming invisible or invincible or double score or the player gains additional lives or instantly dies | The stated random events occur | PASS | 3:40 – 4:15 |
| 12 | 4.5.1 | When the player dies their score is saved | Players score is saved when player dies | PASS | 3:11 – 3:21 |
| 13 | 4.5.2 | Score is transferred to the high score screen where the player can search for the position of their score using binary search | Players score is transferred to the high score screen where the player can find the position of their score | PASS | 3:21 – 3:30 |
| 14 | 4.6.1 | There should be a menu for Flappy Bird that has a start game button and a functionality to start Random Mode | There is a functioning menu with described functions | PASS | 2:16 -2:19 |
| 15 | 4.6.2 | There should also an exit button which will close the program | A Exit Button exists which closes the program when pressed | PASS | 2:15 |

***Bling Bling Boy (My Retro Inspired Game)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 5.1.1 | The player should control a sprite in a restricted play area. The sprite should be in a similar style to other retro styled games | There is a sprite which the player can move around in a play area | PASS | 0:06 – 1:01 |
| 2 | 5.1.2 | The player should be able to collect gems which will increase the players score, they should do this while avoiding the enemies | When the player collects a gem their score increases | PASS | 0:06 – 1:01 |
| 3 | 5.1.3 | The Player Sprite can be moved around the play area at a constant speed | The player sprite moves at a constant speed in all directions | PASS | 0:06 – 1:01 |
| 4 | 5.1.4 | The player has the ability to gain invincibility after reaching a point milestone (every 5 points up to a max) | The player becomes invincible temporarily at every milestone | PASS | 0:10 – 0:13 |
| 5 | 5.1.5 | The player has 3 lives when the game loads up | Player has 3 lives initially | PASS | 0:06 |
| 6 | 5.2.1 | The enemy’s in the game should also have the same art that matches with other retro games | Enemy Sprite shouldn’t seem out of place | PASS | 0:06 |
| 7 | 5.2.2 | After a certain milestone (5 points) the number of enemy’s should increase to make the game difficult as it progresses. However, there should also be a limit on the number of enemies. | The number of enemies increases by one every 5 points until the max number of enemies is reached | PASS | 0:06 – 1:01 |
| 8 | 5.2.3 | The Enemy AI should chase the player and not overlap each other | The enemies should chase the player and shouldn’t overlap with each other | PASS | 0:06 – 1:01 |
| 9 | 5.2.4 | When the Enemy interacts with the Player (Overlaps) the player should lose one life as well as resets its position. | When Player gets caught by enemies one life should be lost and the player sprite position should be reset | PASS | 0:50 – 0:53 ,  0:57 – 1:00 |
| 10 | 5.3.1 | When the player gets a gem and gains a point the gem should move to a random location on the map | Gem moves to a random location on the map once the player gains their point | PASS | 0:06 – 1:01 |
| 11 | 5.3.2 | When an Enemy collides with the player and the player loses one life the player should also lose one point | Player loses one point when caught by enemy | PASS | 0:50 – 0:53 ,  0:57 – 1:00 |
| 12 | 5.3.3 | Once the player loses all their lives, they will be asked for their username which they can enter; this will save their scores to the text file. | When the player has 0 lives they should have the option to save their score by inputting their Username | PASS | 1:01 – 1:17 |
| 13 | 5.3.4 | The player with the highest score should have their score displayed on the main menu | The highest scoring player has their name displayed on the game menu | PASS | 0:04 – 0:05 |
| 14 | 5.3.5 | All Scores that are saved should be saved to a text file as long as the User wants to save their score | All of the Scores should be saved to a text file if the user wants to save it | PASS | 1:01 – 1:17 |
| 15 | 5.4.1 | The game Menu should contain instructions on how to play the game | There should be instructions on the game menu | PASS | 0:04 – 0:05 |
| 16 | 5.4.2 | Menu should contain a start game button which will start the game | The player starts the game by pressing the start game button | PASS | 0:04 – 0:05 |
| 17 | 5.4.3 | There should also be a button which takes the player to the score screen where the player can use Binary Search to search for their score and its position | The player can access the score screen from the menu and search for score positions | PASS | 1:46 – 2:02 |

***C# Menu***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 6.1.1 | A Menu made using C#.NET that will allow the user to access all of the games. From here the User can select which game they want to play. | The Menu displays all the options for all 5 games and the user can choose and play any game on display | PASS | 0:00 |
| 2 | 6.2.1 | The Menu contains button underneath the screenshot of the game that the button will load up. There should also be a high score button which when pressed will allow the user to see the top 5 scores and search for their overall score using Binary Search. The scores will be sorted with Merge Sort | The Menu should contain buttons which load up the game corresponding to that button. There should also be a button which loads up the high score screen which will allow the user to see the top 5 overall scores and the users as well as search for their own scores and receive the position of it | PASS | 0:00 |

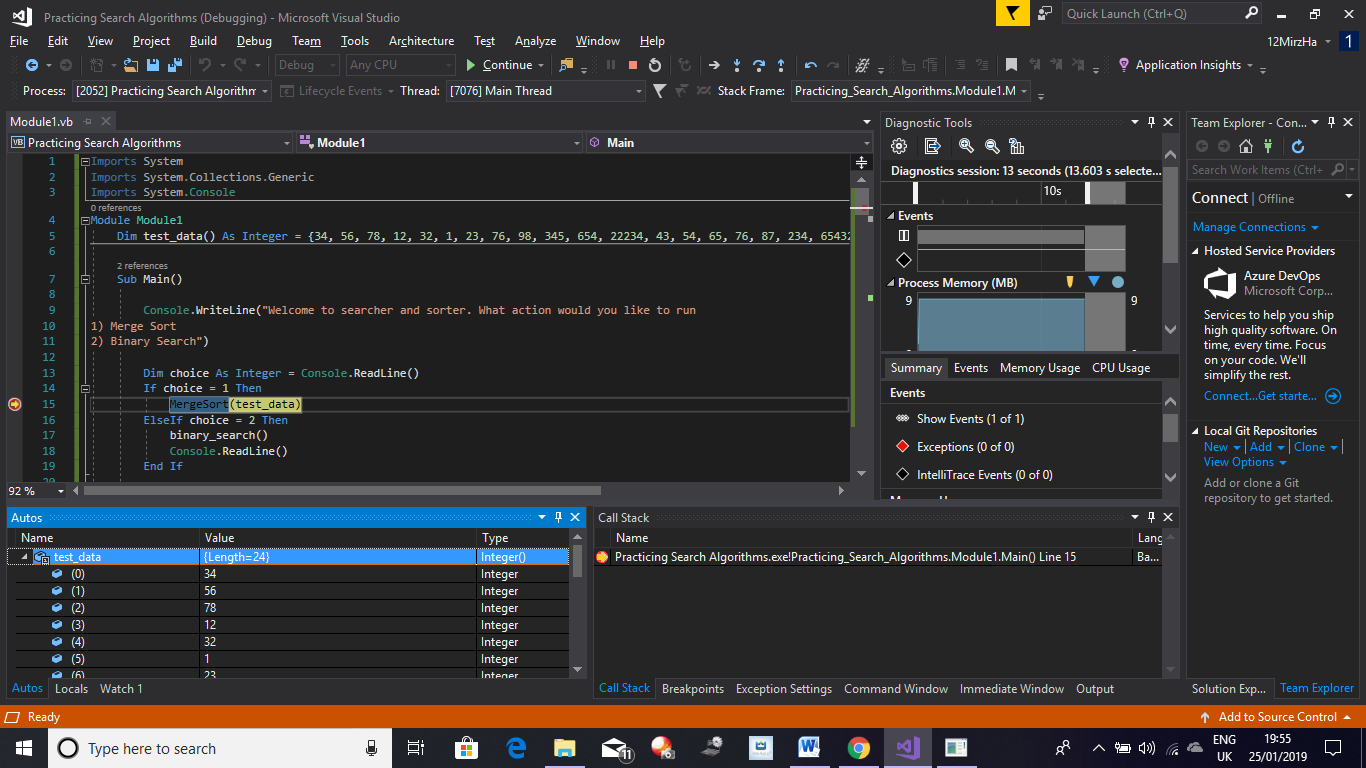
***Data Storage***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Objective Number | Objective Description | Expected Result | PASS/FAIL | Time Stamps |
| 1 | 7.1.1 | Text Files should be used to save the Players Username and Scores for each game. The data for each game should have Merge Sort ran on it to sort the data. Which should then allow the User to search for their score using Merge Sort | Data for each game is written to a text file where its sorted using Merge Sort and can be searched through using Binary Search | PASS | 1:17 – 1:34,  3:15 – 3:32,  6:16 – 6:37,  16:02 - 16:24 |
| 2 | 7.1.2 | The total score will be calculated by adding together the scores from each game for every existing user. This will then be saved to another text file. In the overall High Score Screen the top 5 users will be displayed. | The sum of the scores for each user is saved to another text file. This data will be used to display the top 5 scores | PASS | 20:24 – 20: 27  See Code in Appendix |
| 3 | 7.1.2.1 | All the scores in the summed text file will be sorted using Merge Sort and then used to provide a top 5 leader board. This will allow the user to search for their score using Binary Search which requires the data to be sorted. | The user should be able to search through the data for their score as well as see the top 5 leader board. They will require the data to be sorted before they can search for their score as Binary Search requires the data to be sorted and I am going to sort the data using Merge Sort | PASS | 20:24 – 20:27  See Code in Appendix |

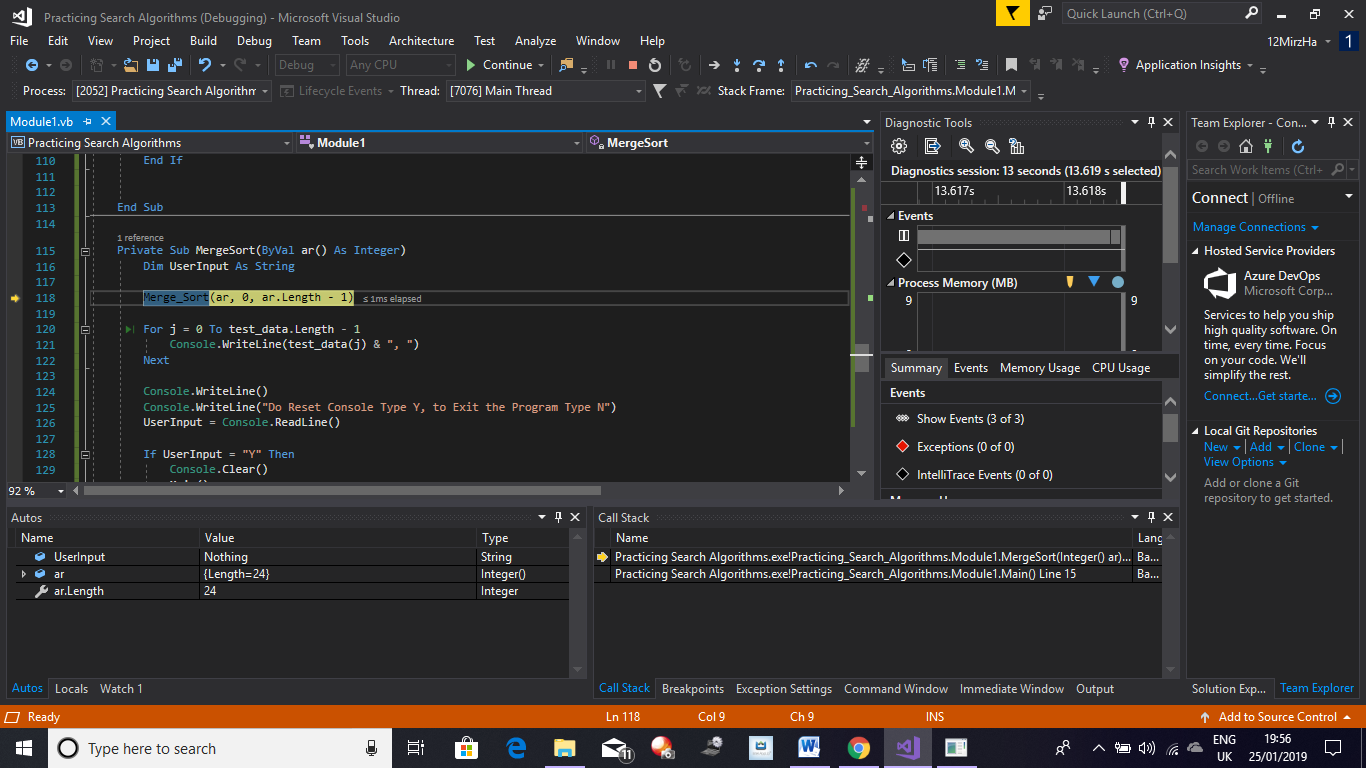
As it can be seen through my Black Box testing my game fulfils all of my objectives. As a result the games fully works and interacts with the scoring system as initially intended. As my scoring system is the core of my project in my White Box testing I have tested my Merge Sort and Binary Search algorithms using break points in the execution of the code with some sample data. This can be seen on page 112:

Code for Program used to test data can be found in the Appendix

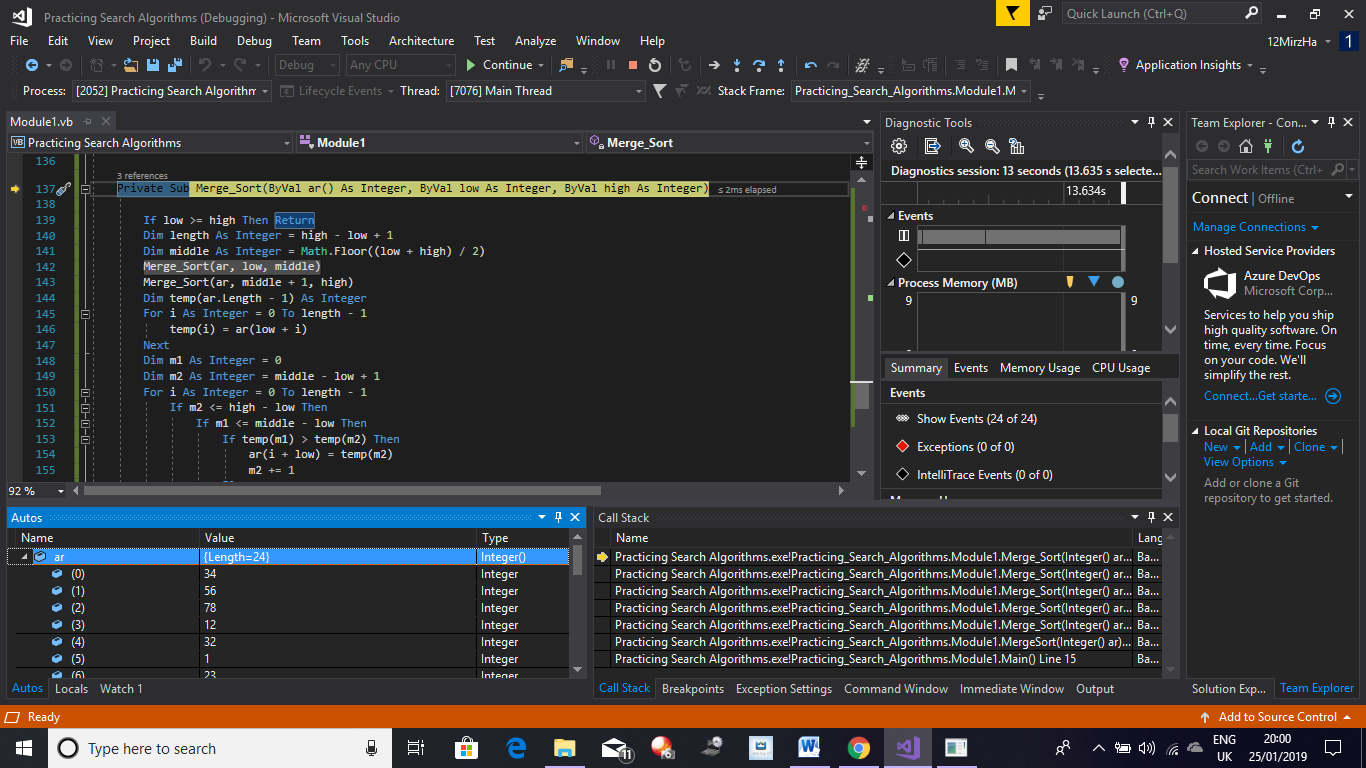
Sorting Data:



Initially Test Data isn’t sorted as can be seen above



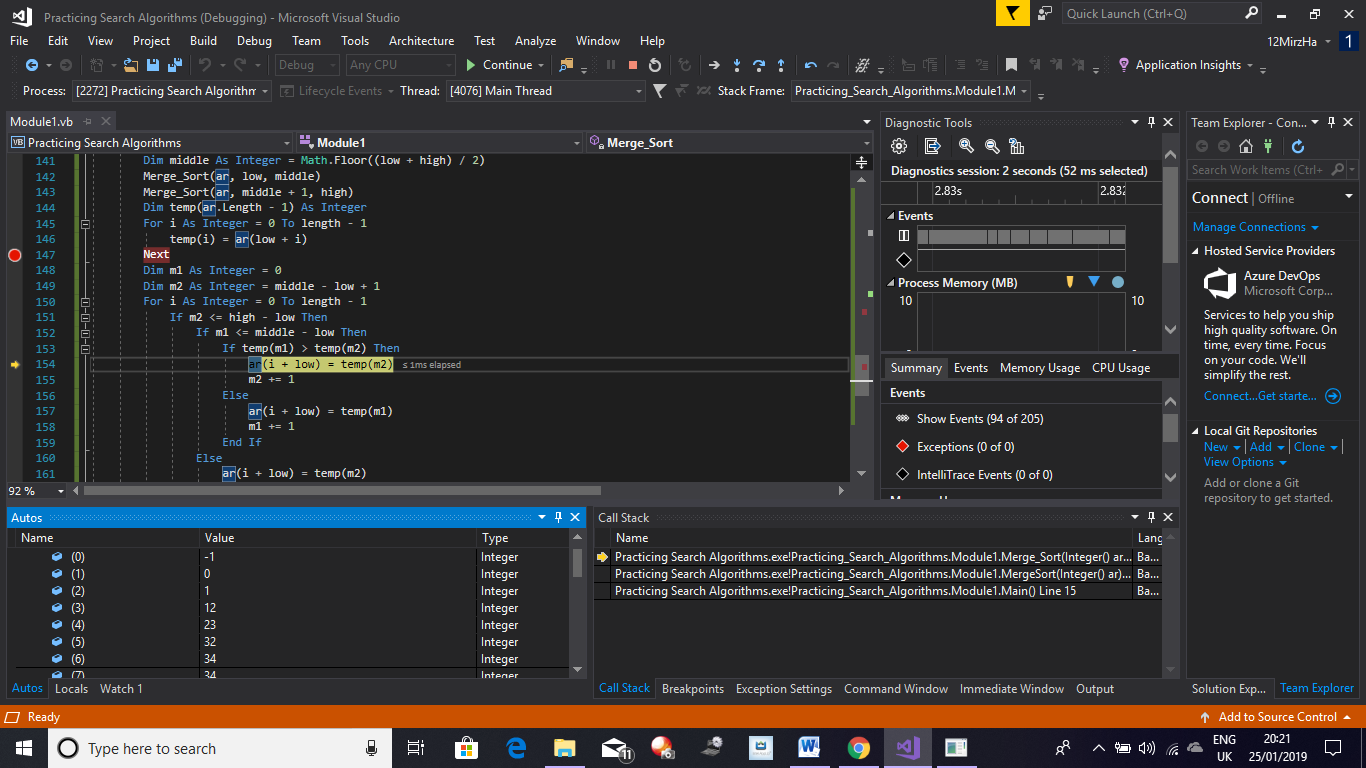
In the screenshot above the data has been passed through to Merge Sort and the correct data has been written to the parameters of the actual Merge Sort algorithm.



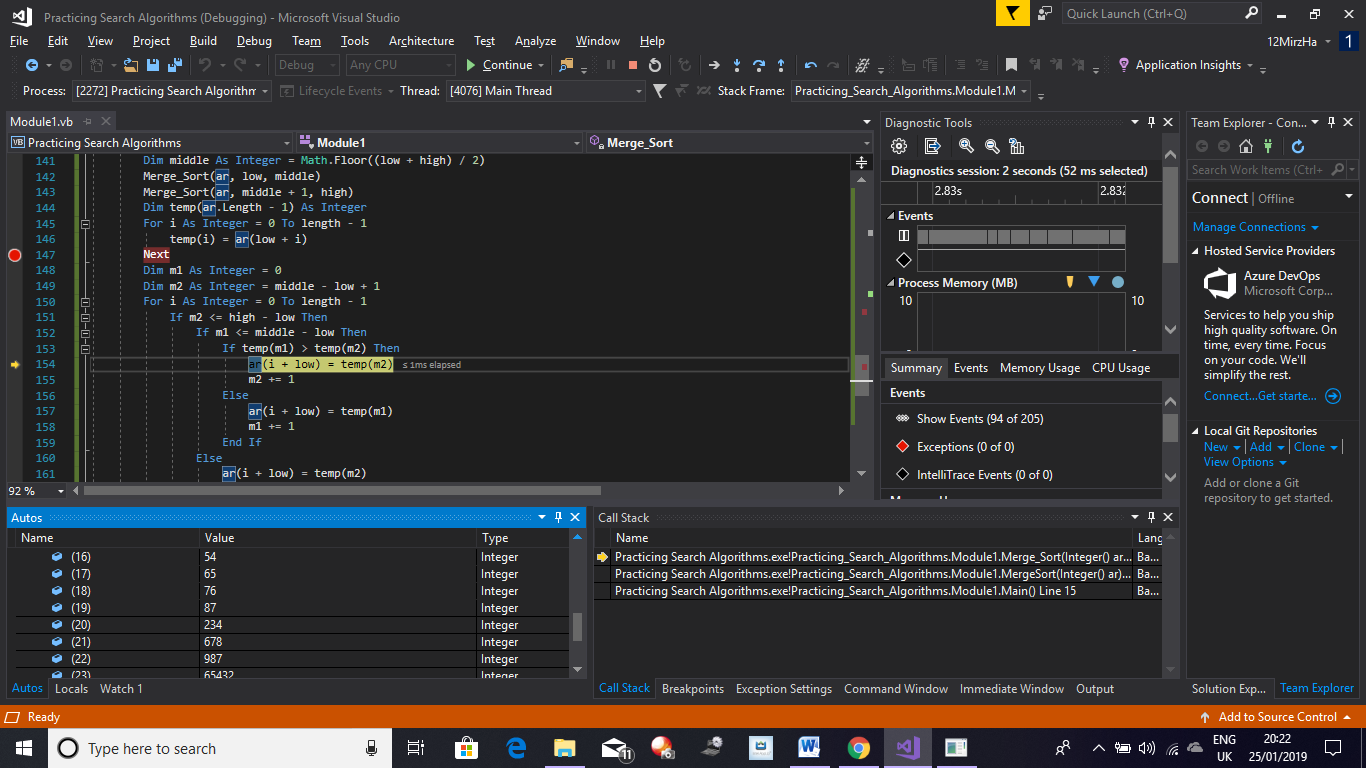
Here the code is repeatedly cut down to two lists by changing middle high and low values.

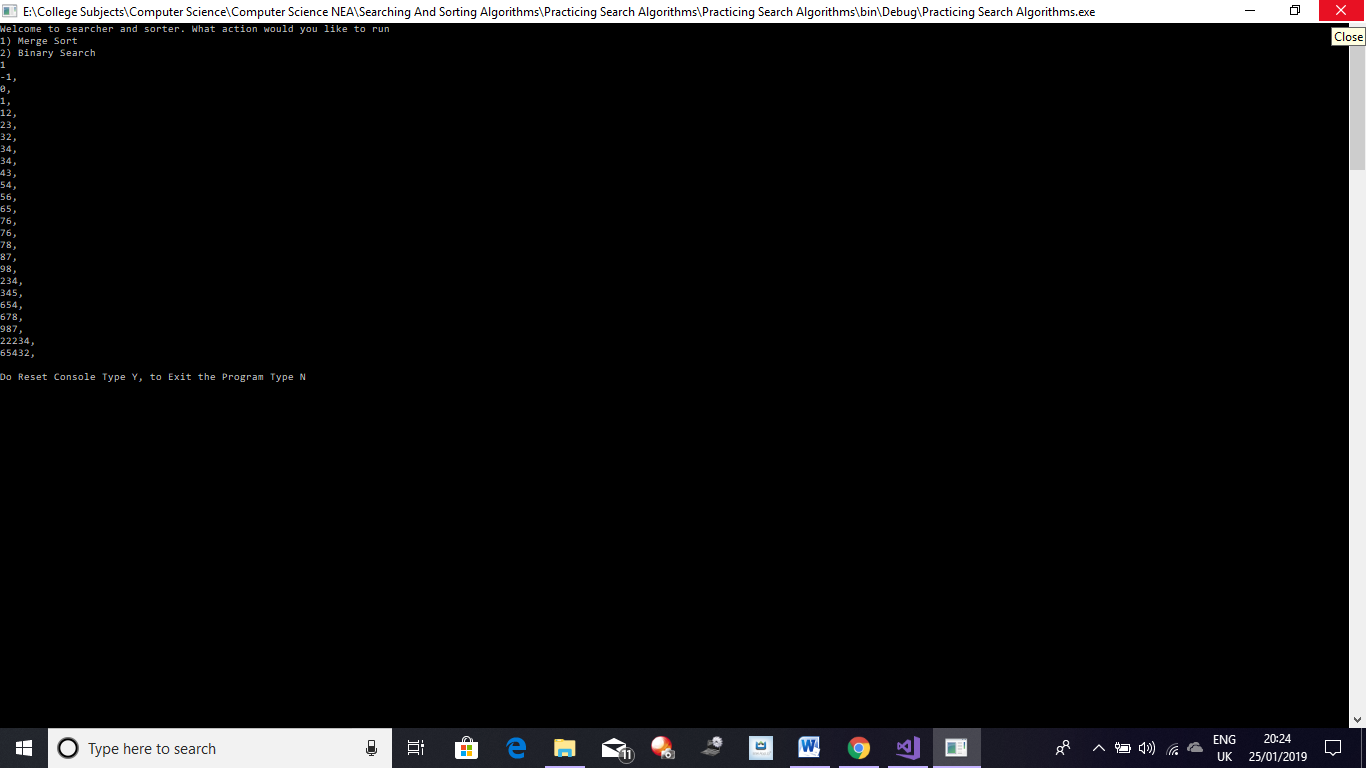


Here the data has been split into two separate and one list has been sorted. The program will now move on to sort the other list formed when data was halved



Here it can be seen the Merge Sort Algorithm has fully sorted both lists, all the data can be seen to be sorted.



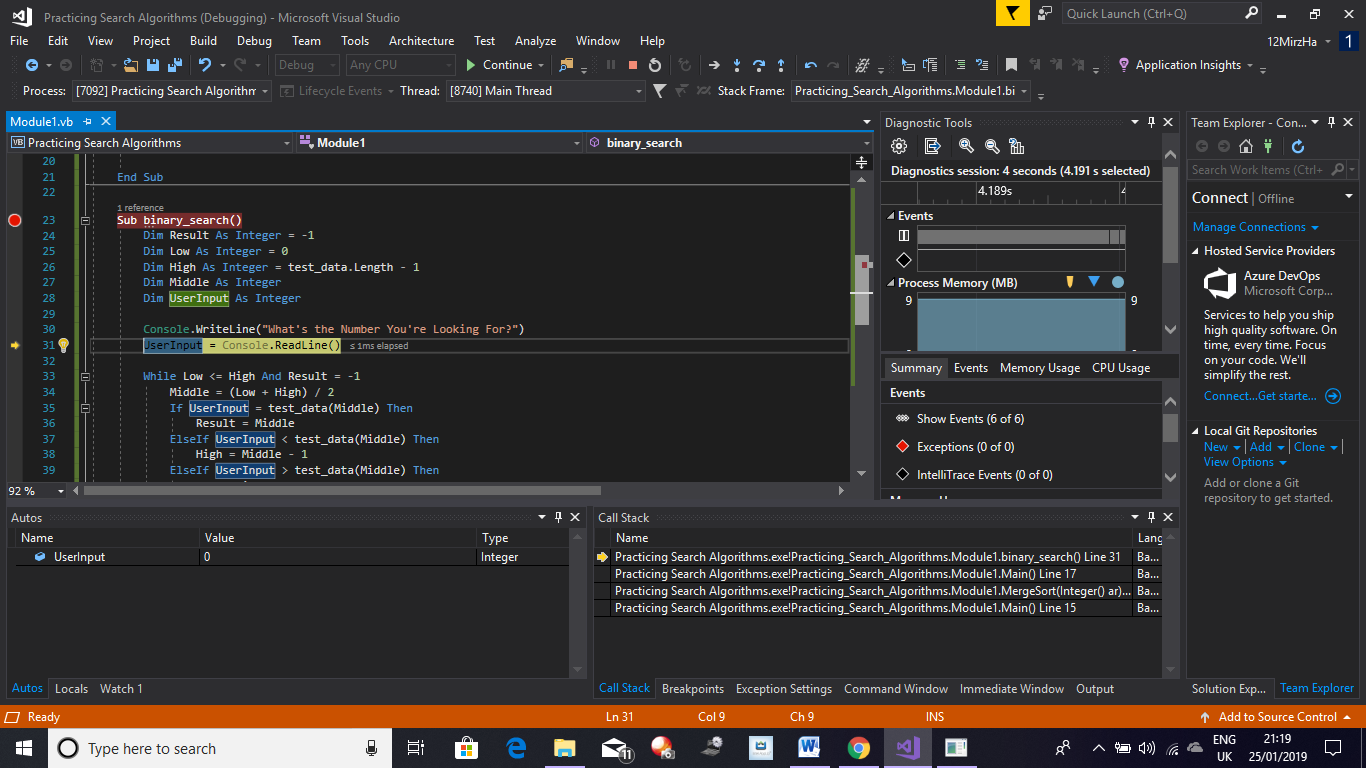


Once the data gets sorted, my program writes it out; here the whole test data can be seen sorted.

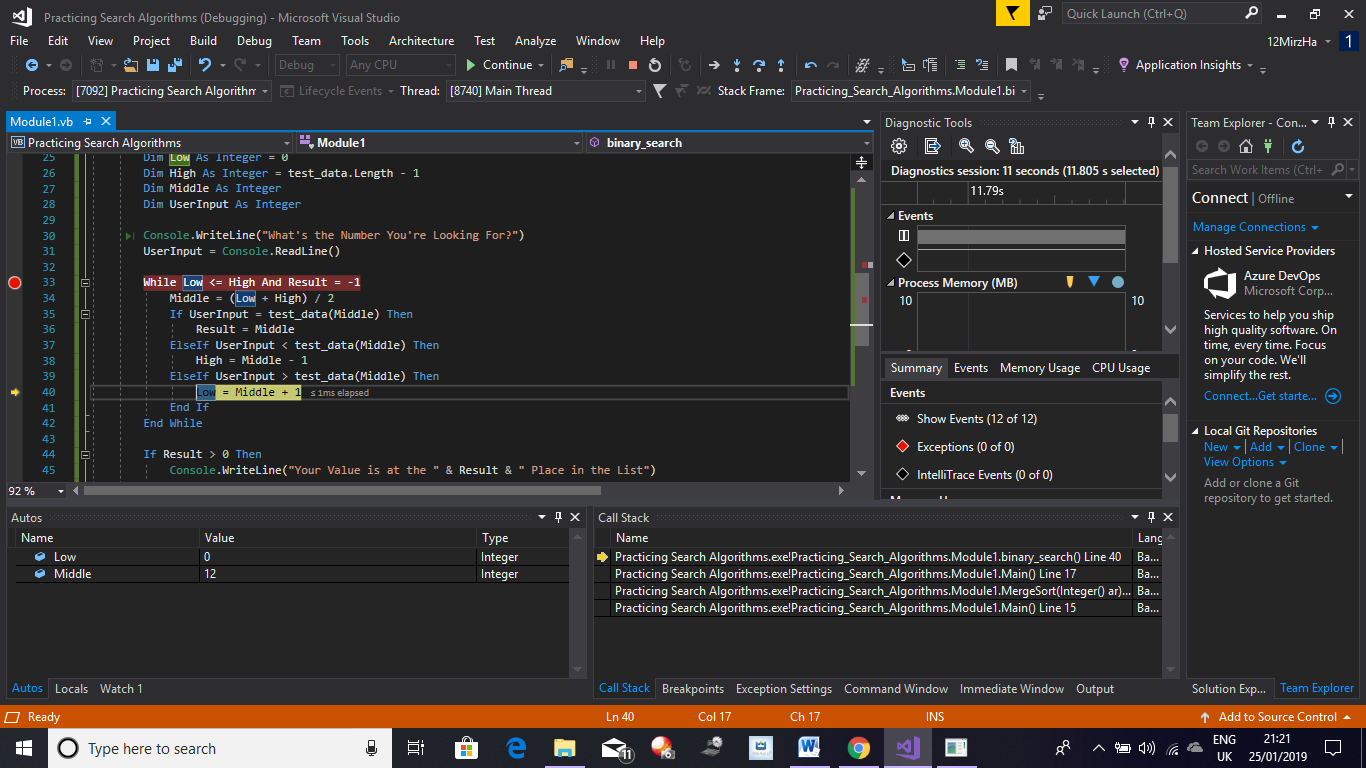
This screenshot shows the second part of the data that was sorted

Searching Data:

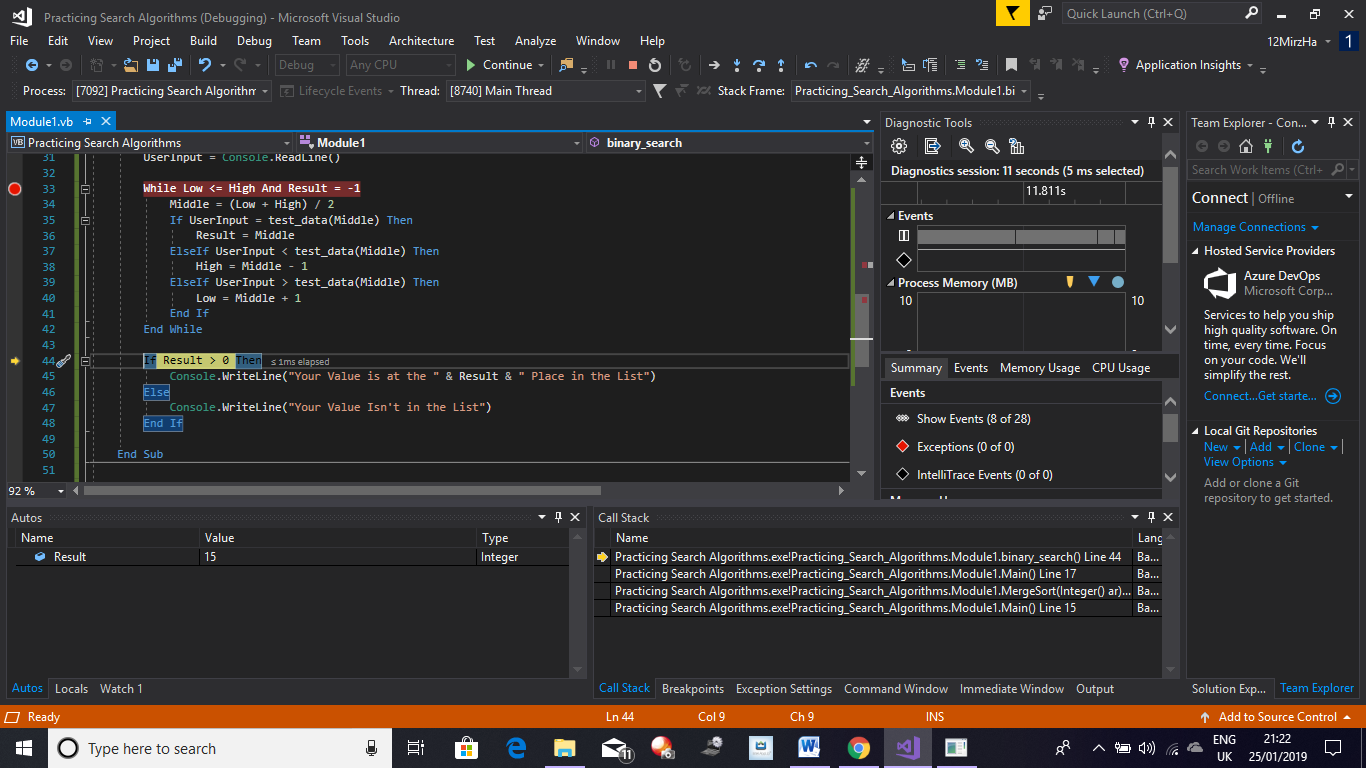
From the test data which is now sorted I am now going to search for a value using my Binary Search Algorithm, the value that I am going to search for is going to be 87 which should be in the 15th place in the array (counting the first index as 0) and in the 16th place (counting the first index of array as 1). As my Binary Search Algorithms counts from 0 it should output its position to be in the 15th spot.



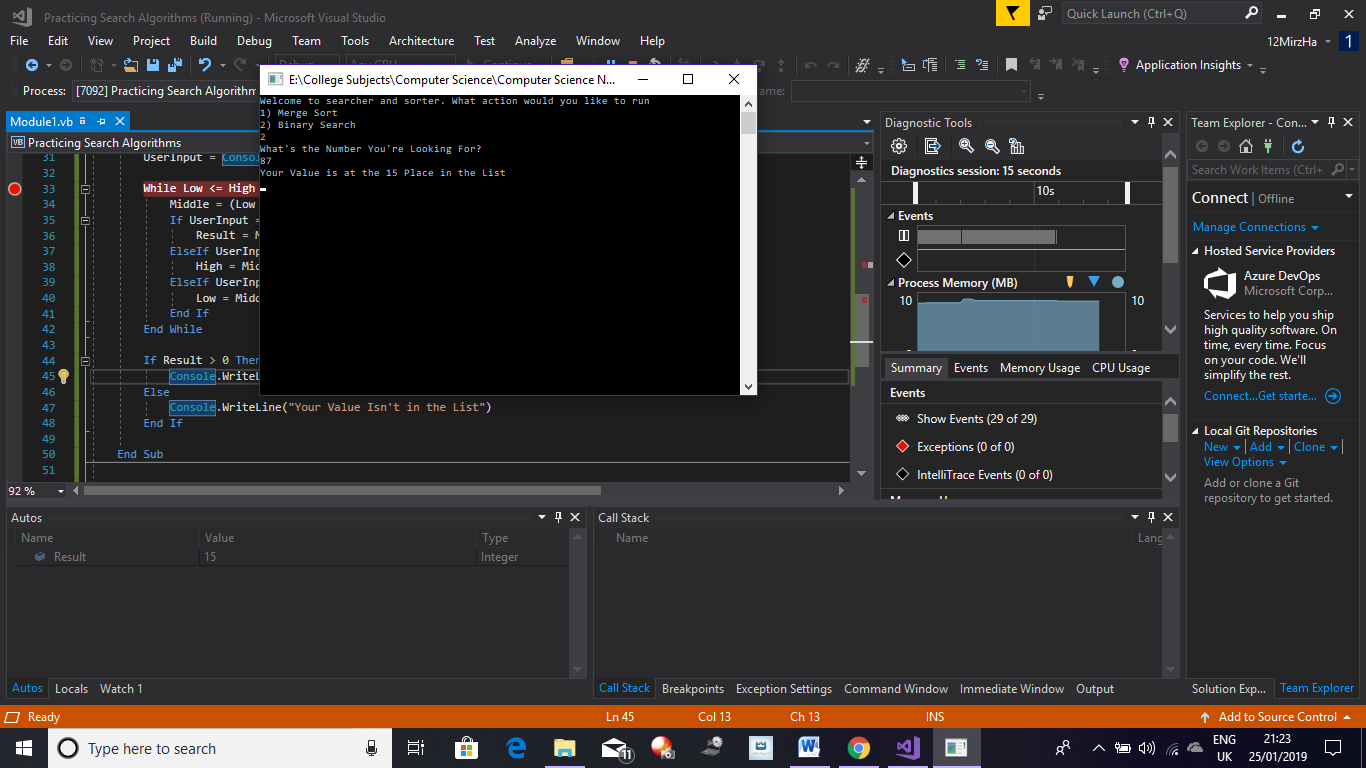
Here the Program asks the User what number they are looking for. So here my input was 87.



Here the data is continuously halved and the value of the Low or High are incremented until the number being searched for is found.



Here it can be seen that the Binary Search code has found the position of the number 87 in the sorted list as the WHILE Statements requirement have been met.



Here it can be seen that the output of my code was the number 15 which as seen indicates the position of the value. This meets my expected result and as such shows the algorithm to be working.

My White Box testing has shown both of my Searching and Sorting Algorithms to be working thus fulfilling my criteria of being able to sort my scores and search through the sorted scores for the position of the score.

**Evaluation**

Having completed my project I will now discuss positive and possible improvement for my project. I have also allowed the people who I initially interviewed with the survey to play my project and to help me come up with possible improvements. My objectives which have been stated in my Analysis and in my Test table, have been fully completed to a level at which I am happy with, they all work as required with no error messages and interact with each other appropriately. Looking back at all of my Objectives:

1: I was able to complete and fulfil all of my Objectives for Objective Point 1 which is the Space Invaders game; this can be seen in my testing as the game works as intended alongside my implementations to the game, it can also be seen in my Testing Table which shows that Space Invaders passed all of its tests. Furthermore, my Space Invaders game also communicates and shares data with its own personal score screen as well as the overall score screen. The proof for all of this can be seen in the link provided.

2: All of my Objectives for Objective Point 2 which is PacMan were. As a result I have a fully working PacMan game which works with all 3 maps I intended to make. The games works as expected with the ghost each following their original path-finding algorithms and moving in the same way as they did in the original, as well as my score system fully working which was done by having the game communicate with the menu by transferring data. All 3 maps work as required as well as does my adaptation of the Ghost AI on all of my custom Maps. The proof for all of the successful tests can all be seen in the link provided.

3: I was able to complete and fulfil all of my Objectives for Objective Point 3 which is Pong. It also passed all of its tests as seen in the Testing Table. The game works as intended alongside my implementation of the ball and computer paddle acting as an AI which would try to trick the player out. The ball rebounds of the walls and paddle in a realistic manner and the computer paddle also hits the ball following a similar AI as in the original game with the AI having a chance of missing the ball. Pong was not having its score saved as that wasn’t the requirement of the game as a result it wasn’t required to communicate with any score screen at all. The proof for all the tests of Pong can be seen in the link provided.

4: I was able to complete and fulfil all of my Objectives for Objective Point 4 which is Flappy Bird. This can be seen in my Testing Table. The game works as intended alongside my implementation of their being a chance for random events to occur. The game responds appropriately to any User Input and communicates with its personal score menu and the overall score as well. The proof for all of this can be seen in the link provided.

5: I was able to complete and fulfil all of my Objectives for Objective Point 5 which is Bling Bling Boy (My Retro Inspired Game). It passed all its test as seen in my Testing Table. The game also works as intended as well as communicating and sharing data within itself and externally with the overall score system. The AI in the game works as intended as well as the player character and his abilities. The proof for all of this can be seen in the link provided.

6: I was able to complete and fulfil all of my Objectives for Objective Point 6 which is the C# menu which compiles all of the games and the overall score system into one UI. The menu passed all tests performed on it as evident in the Test Table. The menu also works as required with each button working as expected and required. The proof for all of the objectives can be seen in the link provided.

7: I was able to complete and fulfil all of my Objectives for Objective Point 7 which is the Scoring System and the Data Storage. The Scoring System and Data Storage (this includes the overall score screen) have been proven to work as shown in my Test Table as they passed all tests performed. All data is retrieved by each game by reading their personal text files which store all of the scores for each user that played that game. All the data is also sorted through using the Merge Sort algorithm which was proven to work as evident in my White Box Testing of my Searching and Sorting Algorithms. Binary Search was also proven to work and can be used by the user to retrieve the position of their score. The data from each game is summed to form an overall score which is used to identify the top 5 players who are displayed in order. The user can also search for their overall score and its position. The proof for all of the objectives can be seen in the link provided.

My games were quite easy to understand by the people who played with it as they were quite basic controls and as many of the games I made had been played by the users they know what controls to use and the users who hadn’t easily understood the controls and picked them up quite quickly due to the lack of complex controls which matches my user requirements and my initial requirement to produce a compilation of Retro Style games. The User requirements fulfilled by just this is that the games are approachable as well as the project being easy to play (User Requirements 1 and 3 respectively). Users also appreciated the small inclusion of instructions at the start of Bling Bling Boy which was My Retro Styled game and as they hadn’t played it before they liked that they knew what the objective was straight of the menu screen. Some users felt that the inclusion of controls would have also been beneficial in Bling Bling Boy as it wasn’t a game they had played before so they didn’t know what key did what. I agree with this critique as the game wasn’t played or heard of before, so any new user would find any help in understanding what to do and how to do it useful in the game, as the controls for Bling Bling Boy are left for the User to figure out. Overall all of the games were easily understood by most of the users.

I made all of the Users play the games on a low spec computer to see how it performed and the game played with little to no lag meaning that they can be played by anyone on any computer as the requirements of the projects seem to be quite low. This fulfils my User requirement of the games being able to run on a low spec computer (User Requirement 4 respectively).

As most of the games I made with the exception of Bling Bling Boy were either from the’ Golden Age’ of the retro gaming industry many user got to experience it through the use of my project and after taking each user who played the projects, most agreed that the project was overall of a good standard and the games were quite fun to play in general. A few users critiqued that the games often got repetitive after a few plays resulting in them wanting to change games more often, however most user agreed that my implementation on the games solved this problem for them. I agree with the critique as I did not add any difficulty scaling to all of the games for instance PacMan as it wasn’t in my objectives however if it was added it would improve my project. Overall my project was seen to be a good example of the ‘Golden Age’ of retro arcade gaming and thus fulfils User requirement 2 and as the majority of the Users agreed, they found my games to be fun and entertaining which fulfils my 5th and final User requirement. Some Users stated that they might even look into Retro games due to them being ‘quick and easy to play especially on the go’. This fulfils my initial purpose of producing this project as I have got a few people to look into the retro and arcade genre of games.

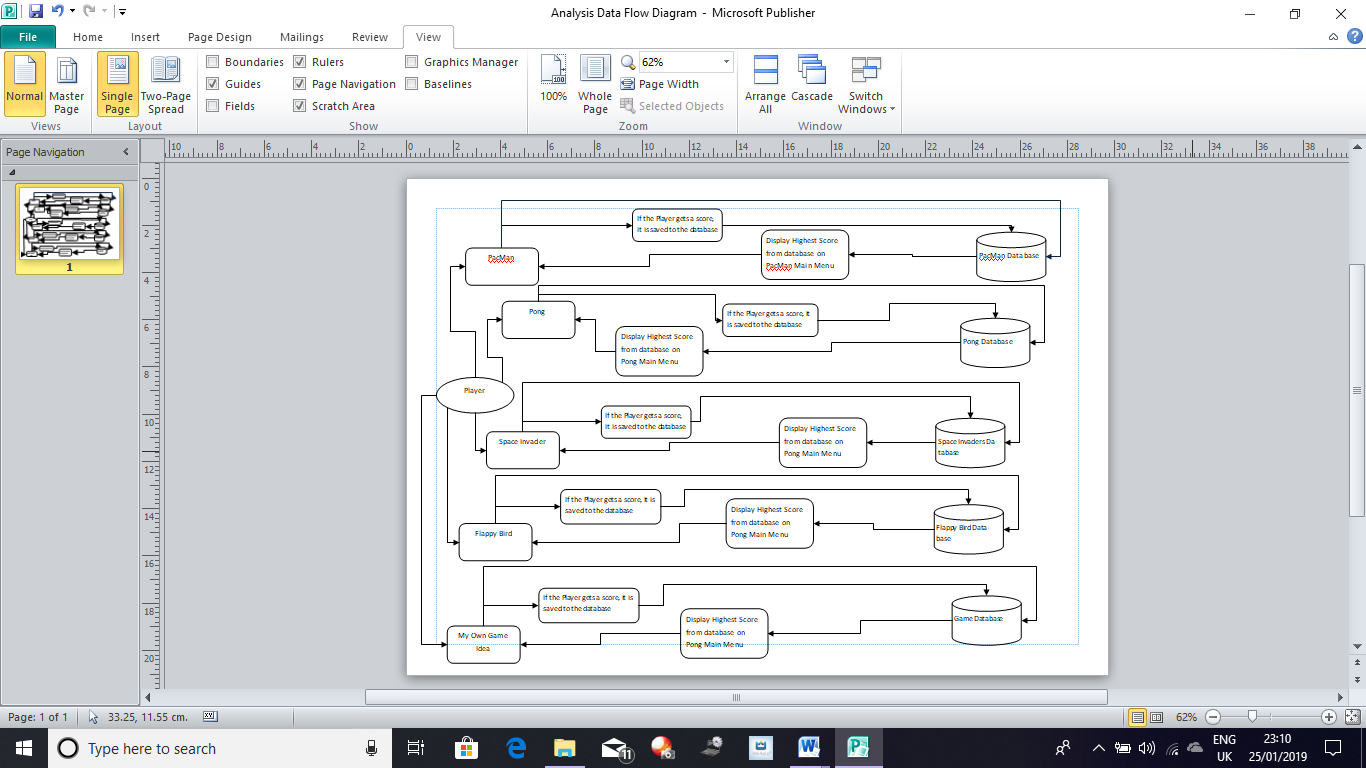
My project does take some data from the User being data that they want to save which is their score and Username. If a User does not want this to be saved, they don’t have to as it’s not a requirement to play the games. However, due to Data Protection I have made it so the users can only play local and not global so their scores will not be compared with other users around the world and can only be seen and accessed by the user themselves on their PC as all data will be saved locally.

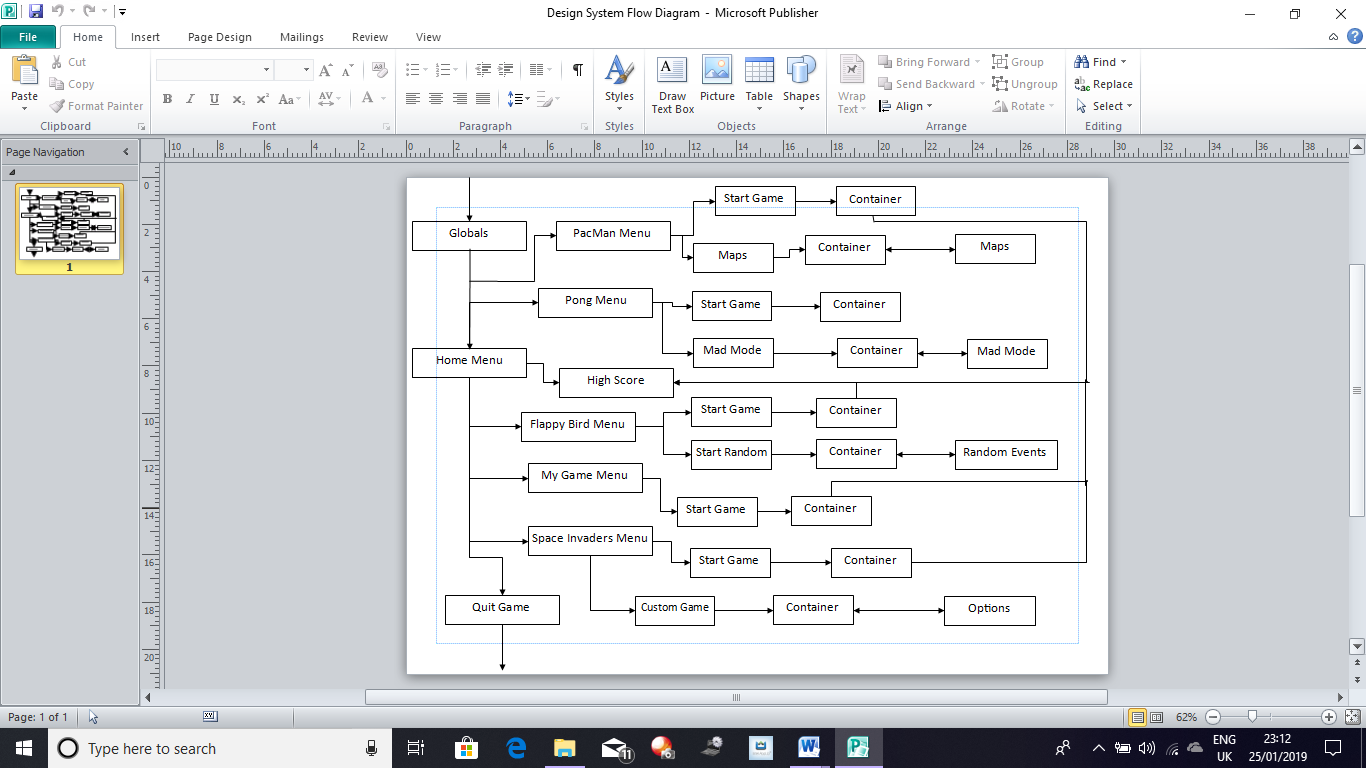
Below a list can be seen of some possible methods of improving my project:

* Add different difficulty levels or scaling difficulty which increases as the player advances in the game, this would also increase the replay value of the games
* Add the Map Maker to PacMan to improve its replay value by allowing users to create their own maps and add a fresh experience to the game
* More informative Instructions so the Users clearly understand how to play the game and what to do

**Conclusion**

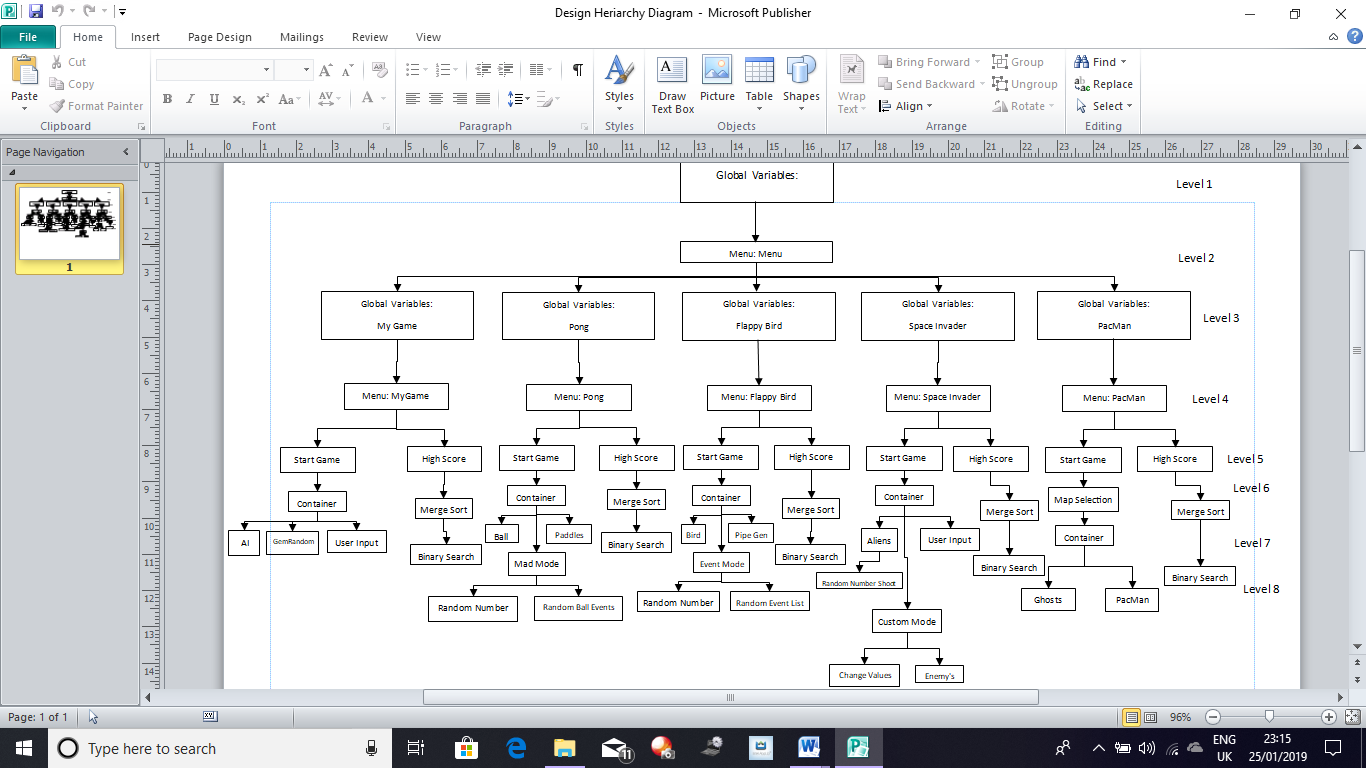
Overall, I am pleased with my projects as it fulfils most of my objectives as well as meeting all of my User requirements as well as fulfilling my initial purpose of this project as stated above by the users who tested my project. I was able to create a remake of all of my games as well as implementations on most of them as well as produce a game of my own making which takes inspiration from the retro genre. The reaction from the users who tested the project seemed quite positive overall which was satisfying with many stating that they enjoyed the games and would play it again. Many users were also competing against their peers to get their username on the top 5 list on the overall score screen. In the end I believe my project as a success which has allowed many users who probably would not have explored the retro arcade genre to see what it offers and it resulted in some users stating that they would explore further into this genre, this fulfils my initial purpose of this project to allow more users to experience the retro arcade genre.

**Appendix**



Appendix Figure 1: Data Flow Diagram

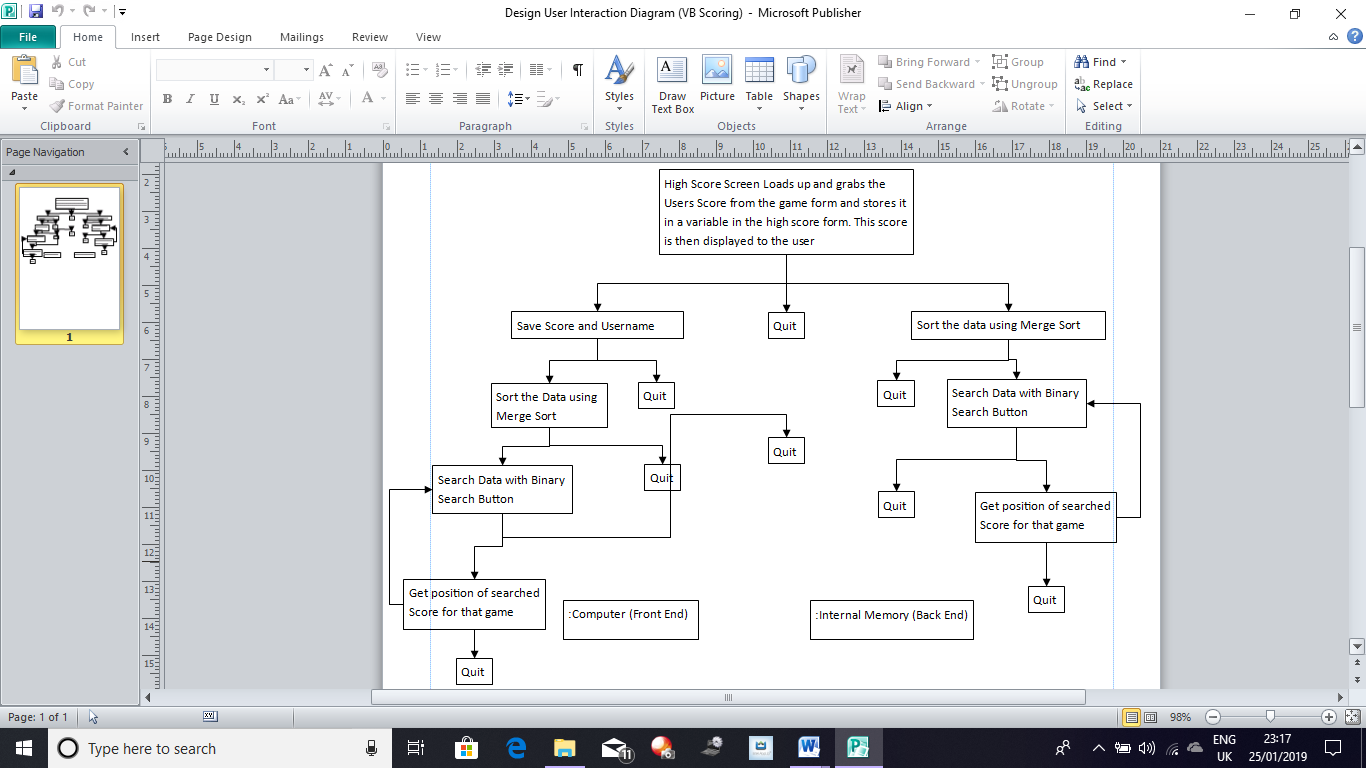
Appendix Figure 2: System Flow Diagram

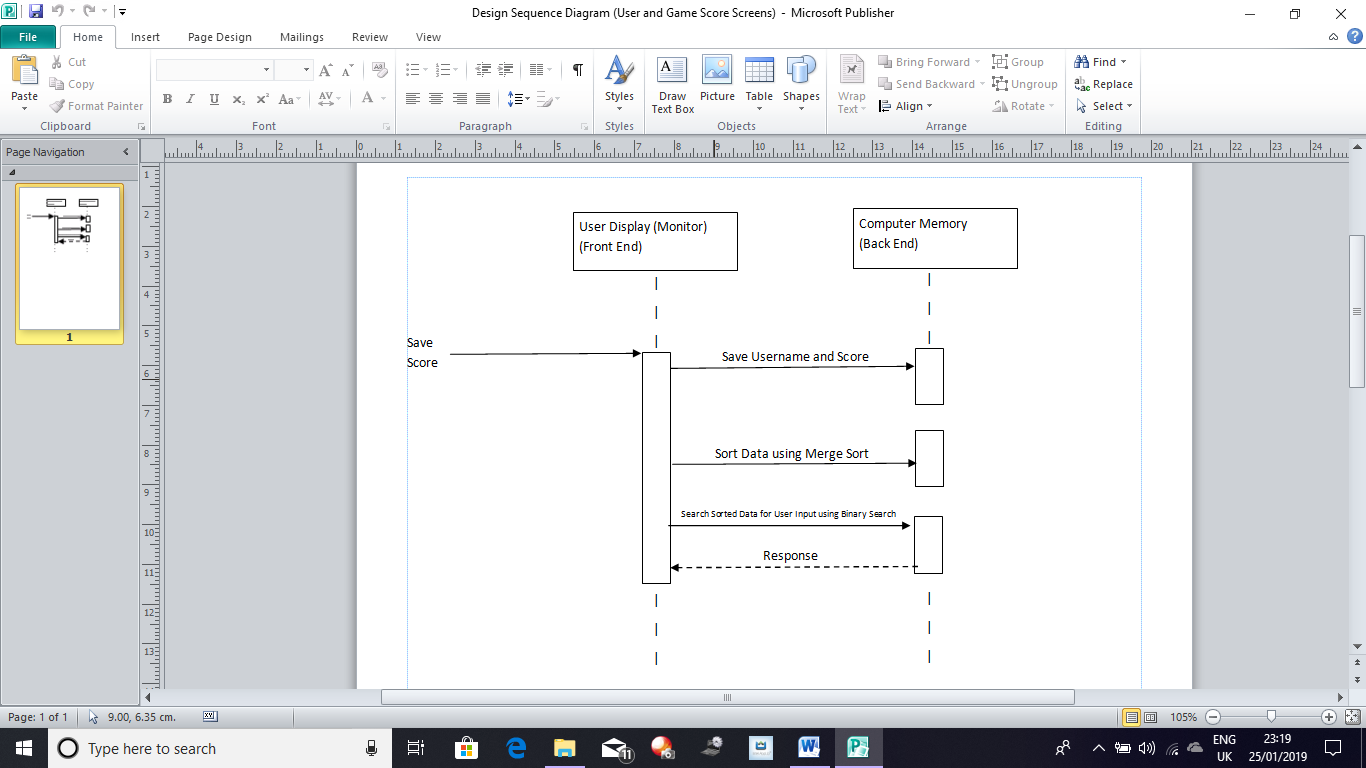


Appendix Figure 3: Hierarchy Diagram

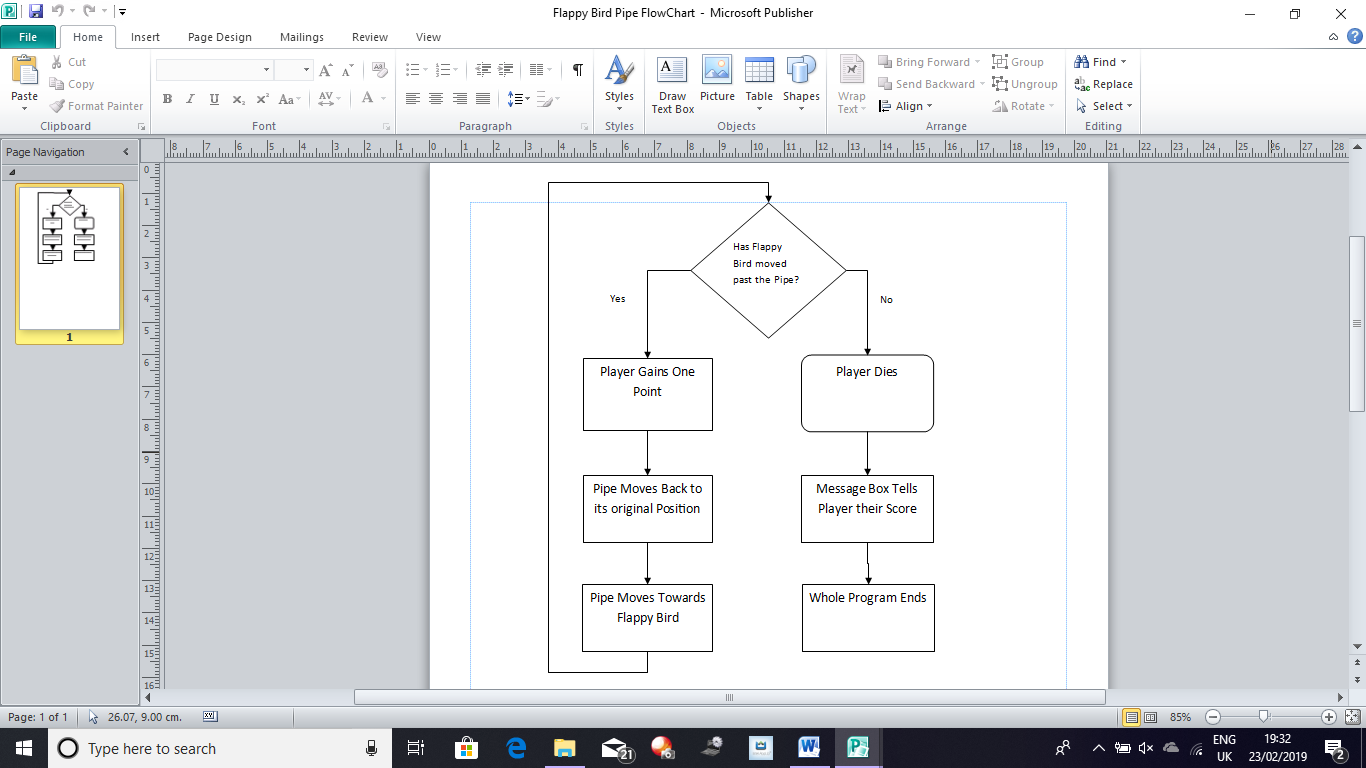
Appendix Figure 1: Data Flow Diagram

Appendix Figure 4: User Interaction Diagram

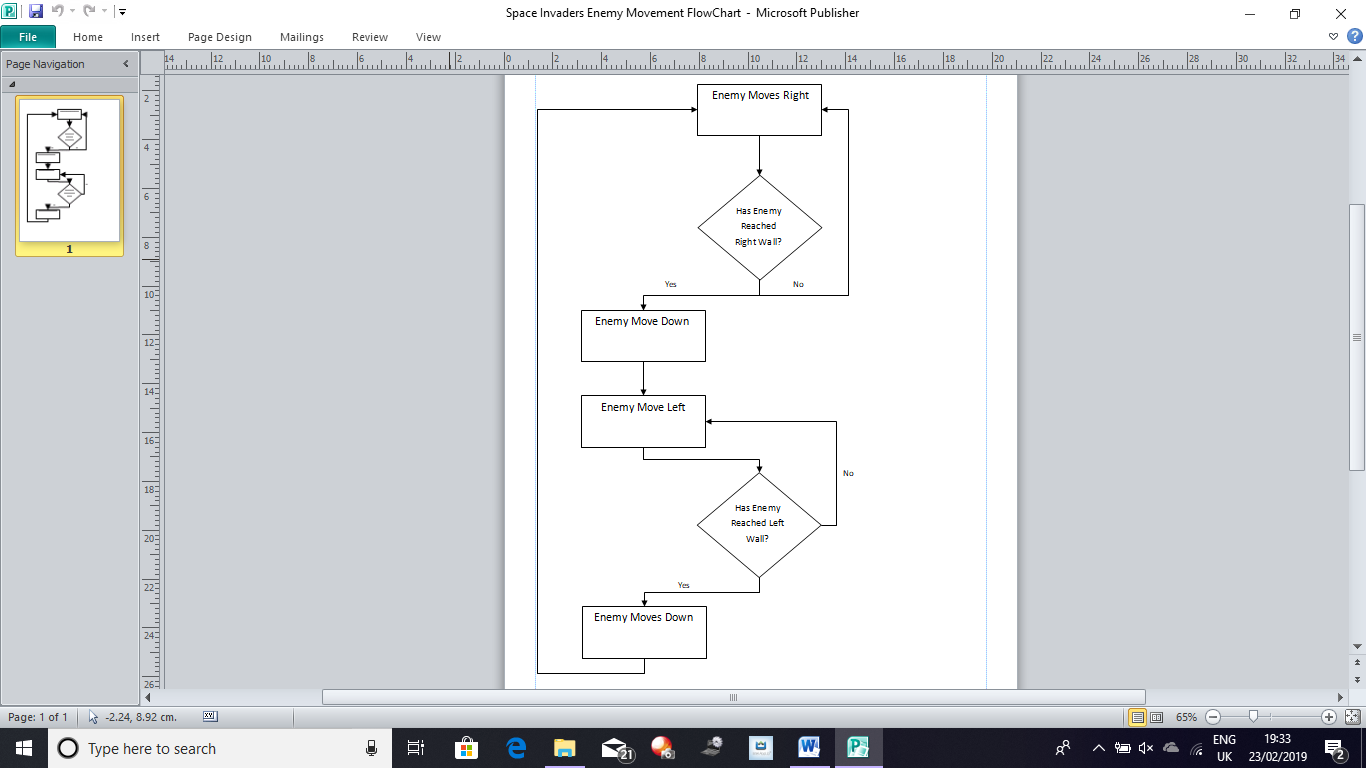




Appendix Figure 5: Sequence Diagram



Appendix Figure 6: Flappy Bird Pipe Flow Chart



Appendix Figure 7:

Space Invaders Enemy Movement Flow Chart

Code for My Retro Inspired Game (Bling Bling Boy):

***Menu:***

Public Class frmMenu

Public PlayerName As String 'Stores the Best Players Name

Public HighScore As Integer 'Stores the Best Players Score

Private Sub Form2\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim temp As String

Using Reader As IO.StreamReader = New IO.StreamReader("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\BestPlayer.txt")

PlayerName = Reader.ReadLine

temp = Reader.ReadLine

End Using

HighScore = Integer.Parse(temp)

lblPlayer.Text = ("Best Score: " & PlayerName) 'Writes the best players name to a label

lblHighScore.Text = ("High Score: " & HighScore) 'Writes the best players score to a label

End Sub

Private Sub butStartGame\_Click(sender As Object, e As EventArgs) Handles butStartGame.Click 'This code is ran when the Button called butStartGame is clicked

frmGame.Show() 'The form where the game is played is shown to the user

End Sub

Private Sub butDebug\_Click(sender As Object, e As EventArgs) Handles butDebug.Click

frmScoreMenu.Show() 'The form where the user can search for their scores and/or submit them is shown to the user

End Sub

End Class

***Game:***

Public Class frmGame

#Region "Variables"

Dim MoveUp As Boolean = False 'This Variable is used to allow the player to move their character up

Dim MoveDown As Boolean = False 'This Variable is used to allow the player to move their character down

Dim MoveLeft As Boolean = False 'This Variable is used to allow the player to move their character left

Dim MoveRight As Boolean = False 'This Variable is used to allow the player to move their character right

Dim PlayerSpeed As Integer = 10 'This Variable holds the speed the player can move at

Dim Lives As Integer = 3 'This Variable stores the number of Lives the player has

Dim Count As Integer = 0 'This Variable is used to count the number of times a event takes place

Dim Score As Integer = 0 'This Variable holds the players actual Score

Dim RandomX As New Random 'This Variable stores a Random number and is used to allow the Gem to appear at a Random Location on the Form

Dim RandomY As New Random 'This Variable stores a Random number and is used to allow the Gem to appear at a Random Location on the Form

Dim CurrentNumberOfEnemies As Integer = 1 'This Variable holds the number of enemies on the form when the game is running

Dim EnemyArray(5) As PictureBox 'Stores all the Enemys in a Array

Dim InvincibilityTimer As Integer = 1000 'This Variable is used to give the player temorary invincibility

Dim EnemySpeed As Single = 1.5 'This Variable stores the Speed of the enemies

Dim HighScore As Integer 'Used to check if player has beaten current high score or not

Dim temp As String 'Used to convert string to integer

#End Region

Private Sub frmGame\_Load(sender As Object, e As EventArgs) Handles Me.Load 'This code is ran when the Form is loaded

Dim null As String 'Used to store irrelevent data

'The code below stores all of the enemies into the array

EnemyArray(1) = pbEnemy1

EnemyArray(2) = pbEnemy2

EnemyArray(3) = pbEnemy3

EnemyArray(4) = pbEnemy4

EnemyArray(5) = pbEnemy5

Randomize() 'Loads up the .NET Randomize Function

Using Reader As IO.StreamReader = New IO.StreamReader("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\BestPlayer.txt") 'Reads the stated file

null = Reader.ReadLine 'Stores the irrelvent data in null

temp = Reader.ReadLine 'Stores the required data in temp

End Using

HighScore = Integer.Parse(temp) 'HighScore is recieved by converting temp to integer

pbGem.Location = New Point(RandomX.Next(1, 550), RandomY.Next(-3, 413)) 'Used to Assign the Gem a Random Location on the Form

pbGem.Visible = True 'As initially the Gem isn't Visible this code makes it Visible so the user can see where it is

End Sub

Private Sub frmGame\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown 'This code is ran when a Key is pressed

Select Case e.KeyCode

Case Keys.Up 'If the Up Key is pressed then the following code is ran

MoveUp = True 'MoveUp becomes True

Case Keys.Down 'If the Down Key is pressed then the following code is ran

MoveDown = True 'MoveDown becomes True

Case Keys.Left 'If the Left Key is pressed then the following code is ran

MoveLeft = True 'MoveLeft becomes True

Case Keys.Right 'If the Right Key is pressed than the following code is ran

MoveRight = True 'MoveRight becomes True

End Select

End Sub

Private Sub frmGame\_KeyUp(sender As Object, e As KeyEventArgs) Handles Me.KeyUp 'This code is ran when a Key is not being pressed / released

Select Case e.KeyCode

Case Keys.Up 'If the Up Key is not being pressed then the following code is ran

MoveUp = False 'MoveUp becomes False

Case Keys.Down 'If the Down Key is not being pressed then the following code is ran

MoveDown = False 'MoveDown becomes False

Case Keys.Left 'If the Left Key is not being pressed then the following code is ran

MoveLeft = False 'MoveLeft becomes False

Case Keys.Right 'If the Right Key is not being pressed then the following code is ran

MoveRight = False 'MoveRight becomes False

End Select

End Sub

Private Sub gameTimer\_Tick(sender As Object, e As EventArgs) Handles gameTimer.Tick 'This code is ran when the gameTimer ticks

InvincibilityTimer -= 10 'Decrements from the Invincibility timer to count it down

If MoveUp = True Then 'If MoveUp is True then the following code is ran

pbBoy.Top -= PlayerSpeed 'The player moves up

ElseIf MoveDown = True Then 'If MoveDown is True then the following code is ran

pbBoy.Top += PlayerSpeed 'The player moves down

ElseIf MoveLeft = True Then 'If MoveLeft is True then the following code is ran

pbBoy.Left -= PlayerSpeed 'The player moves left

ElseIf MoveRight = True Then 'If MoveRight is True then the following code is ran

pbBoy.Left += PlayerSpeed 'The player moves right

End If

If pbBoy.Location.Y > 380 Then 'If the player goes beyond the boundary of the form then the following code is ran

pbBoy.Location = New Point(pbBoy.Location.X, 380) 'The player gets placed at the boundary

ElseIf pbBoy.Location.Y < -6 Then 'If the player goes beyond the boundary of the form then the following code is ran

pbBoy.Location = New Point(pbBoy.Location.X, -5) 'The player gets placed at the boundary

End If

If pbBoy.Location.X > 530 Then 'If the player goes beyond the boundary of the form then the following code is ran

pbBoy.Location = New Point(530, pbBoy.Location.Y) 'The player gets placed at the boundary

ElseIf pbBoy.Location.X < -10 Then 'If the player goes beyond the boundary of the form then the following code is ran

pbBoy.Location = New Point(-10, pbBoy.Location.Y) 'The player gets placed at the boundary

End If

If pbBoy.Bounds.IntersectsWith(pbGem.Bounds) Then 'If the player collects a Gem then the following code is ran

Score += 1 'The player gains 1 point to their Score

pbGem.Visible = False 'The Gem temporarly becomes invisible

pbGem.Location = AssignRandomLocation(pbGem.Location) 'The Gem gets a new Random Location

pbGem.Visible = True 'The Gem becomes Visible again

End If

lblScore.Text = ("Score: " & Score) 'Displays Score to user

lblLives.Text = ("Lives: " & Lives) 'Displays the number of Lives to user

For i = 1 To CurrentNumberOfEnemies 'FOR loop goes through all of the enemies on the screen

If InvincibilityTimer > 0 Then 'If the InvincibilityTimer Variable is greater than 0 then the following code is ran

pbBoy.Image = My.Resources.Invincible\_Main\_Sprite 'The player character changes his sprite

Else 'If the intial requirement isn't fulfilled then the following code is ran

pbBoy.Image = My.Resources.Untitled 'The player character gets his original sprite

If pbBoy.Bounds.IntersectsWith(EnemyArray(i).Bounds) Then 'If enemy and player are in contact then the following code is ran

EnemyArray(i).Location = AssignRandomLocation(EnemyArray(i).Location) 'The enemies are moved to a random location on the Form

InvincibilityTimer = 1000 'The InvincibilityTimer is reset

Lives -= 1 'The player loses a life

Score -= 1 'The player loses a point from their score

pbBoy.Location = New Point(446, 91) 'Puts the player back to their original location

EnemyArray(i).Location = AssignRandomLocation(EnemyArray(i).Location) 'Puts the enemies in random location

End If

End If

Next

If Score = 5 And Count = 0 Then

CurrentNumberOfEnemies = 2 'When this Score milestone is reacher number of enemies increases

EnemyArray(CurrentNumberOfEnemies).Visible = True 'Makes the CurrentNumberOfEnemies Visible

InvincibilityTimer = 1000 'Resets InvincibilityTimer allowing the player to go temporarily invincible

EnemySpeed = 2 'Increases EnemySpeed by 0.5

Count = 1 'Makes it so program knows this event has occured already

ElseIf Score = 10 And Count = 1 Then

CurrentNumberOfEnemies = 3 'When this Score milestone is reacher number of enemies increases

EnemyArray(CurrentNumberOfEnemies).Visible = True 'Makes the CurrentNumberOfEnemies Visible

InvincibilityTimer = 1000 'Resets InvincibilityTimer allowing the player to go temporarily invincible

EnemySpeed = 2.5 'Increases EnemySpeed by 0.5

Count = 0 'Makes it so computer knows this is the most recent event allowing the other events to take place which require count = 0

ElseIf Score = 15 And Count = 0 Then

CurrentNumberOfEnemies = 4 'When this Score milestone is reacher number of enemies increases

EnemyArray(CurrentNumberOfEnemies).Visible = True 'Makes the CurrentNumberOfEnemies Visible

InvincibilityTimer = 1000 'Resets InvincibilityTimer allowing the player to go temporarily invincible

EnemySpeed = 3 'Increases EnemySpeed by 0.5

Count = 1 'Makes it so computer knows this is the most recent event allowing the other events to take place which require count = 1

ElseIf Score = 20 And Count = 1 Then

CurrentNumberOfEnemies = 5 'When this Score milestone is reacher number of enemies increases

EnemyArray(CurrentNumberOfEnemies).Visible = True 'Makes the CurrentNumberOfEnemies Visible

InvincibilityTimer = 1000 'Resets InvincibilityTimer allowing the player to go temporarily invincible

EnemySpeed = 3.5 'Increases EnemySpeed by 0.5

Count = 0 'Makes it so computer knows this is the most recent event allowing the other events to take place which require count = 0

End If

If Lives = 0 Then 'When the player runs out of Lives then the following code is ran

gameTimer.Stop() 'The gameTimer is stopped so the events in this timer no longer take place

enemyTimer.Stop() 'The enemyTimer is stopped so the events in this timer no longer take place

lblScore.Text = ("Score: " & Score) 'Stores the Score in the Score Label

If Score > HighScore Then

Dim UserInput As String = InputBox("What is Your UserName?") 'Holds the users username

Using Writer As IO.StreamWriter = New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\BestPlayer.txt") 'Used to write to stated file

Writer.WriteLine(UserInput) 'Writes the users username

Writer.WriteLine(Score) 'Writes the users score

End Using

MessageBox.Show("Well Done You Have the Current High Score") 'Informs user that they have beaten the current high score

End If

MessageBox.Show("Game Over Your Score is " & Score) 'A MessageBox appears which tells the user their final score

frmScoreMenu.CurrentUserScore = Score 'The users Score is transferred from this form to the score menu

Me.Hide() 'This form is closed

frmScoreMenu.Show() 'The score menu form is shown to the user

End If

End Sub

Private Sub enemyTimer\_Tick(sender As Object, e As EventArgs) Handles enemyTimer.Tick 'This code is ran when the enemyTimer Ticks

For i = 1 To CurrentNumberOfEnemies 'This FOR loop is used to loop through all the numbers up to the integer stored inside CurrentNumberOfEnemies

If pbBoy.Location.X > EnemyArray(i).Location.X Then 'If the player is to the right of the enemy then the following code is ran

EnemyArray(i).Left += EnemySpeed 'The enemy moves to the right

Else 'If the intial requirement isn't fulfilled then the following code is ran

EnemyArray(i).Left -= EnemySpeed 'The enemy moves to the left

End If

If pbBoy.Location.Y > EnemyArray(i).Location.Y Then 'If the player is below the player then the following code is ran

EnemyArray(i).Top += EnemySpeed 'The enemy moves down

Else 'If the inital requirement isn't fulfilled then the following code is ran

EnemyArray(i).Top -= EnemySpeed 'The enemy moves up

End If

Next

For j = 2 To 5 'FOR loop goes through values from 2 to 5

If pbEnemy1.Bounds.IntersectsWith(EnemyArray(j).Bounds) Then 'Checks if enemy 1 has collided with any of the other enemies

pbEnemy1.Left += 5 'If a collision has occured enemy 1 moves away

End If

Next

For k = 3 To 5 'FOR loop goes through values from 3 to 5

If pbEnemy2.Bounds.IntersectsWith(EnemyArray(k).Bounds) Then 'Checks if enemy 2 has collided with any of the other enemies

pbEnemy2.Left -= 5 'If a collision has occured enemy 2 moves away

End If

Next

For l = 4 To 5 'FOR loop goes through values from 4 to 5

If pbEnemy3.Bounds.IntersectsWith(EnemyArray(l).Bounds) Then 'Checks if enemy 3 has collided with any of the other enemies

pbEnemy3.Left += 5 'If a collision has occured enemy 3 moves away

End If

Next

If pbEnemy4.Bounds.IntersectsWith(pbEnemy5.Bounds) Then 'Checks to see if enemy 4 and enemy 5 have collided

pbEnemy4.Left -= 5 'If a collison has occured then enemy 4 moves away

End If

End Sub

Private Function AssignRandomLocation(ByRef pbLocation As Point) 'This code is used to give a pb a random location

pbLocation = New Point(RandomX.Next(1, 550), RandomY.Next(-3, 413)) 'Used to Assign a pb a Random Location on the Form

Return pbLocation 'Outputs this new Location

End Function

End Class

***Storing Scores:***

Public Class frmScoreMenu

Public CurrentUserScore As Integer 'Used to transfer the current users score between game form to score form

'Variable Below Stores the Usernames with the Scores of the Players

Dim UsernameArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\Username.txt")

'Used as a Counter for the Array

Dim NumOfValues As Integer = 0

'Just Stores the Scores

Dim JustScore(NumOfValues) As Integer

'Just Stores the Usernames

Dim JustUserName() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\JustUserName.txt")

Private Sub Form3\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'As soon as Form Loads the Scores are read in to a new variable

Dim ScoreArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\score.txt")

lblScore.Text = CurrentUserScore 'Gets the Score from the game form and displays it to the user

UpdateData() 'Calls Sub Routine by the same name here

End Sub

Private Sub butSubmit\_Click(sender As Object, e As EventArgs) Handles butSubmit.Click

'Write to Username file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\Username.txt", True)

Writer.WriteLine(txtUsername.Text & ", " & lblScore.Text)

End Using

'Write to the Score file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\score.txt", True)

Writer.WriteLine(lblScore.Text)

End Using

'Varaibles below are used to check and makes sure data isn't repeated

Dim CheckExists() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\JustUserName.txt") 'Reads username

Dim count As Integer = 0

If CheckExists.Length < 1 Then 'If the File has no values the following code is ran

'The File has the data written to as there is no data to compare against

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

Else 'If there is data in the file then the following code is ran

'Checks if the Score already exists in the database

For i = 0 To CheckExists.Length - 1

If CheckExists(i) = txtUsername.Text Then

count = 0

ElseIf CheckExists(i) <> txtUsername.Text Then

count += 1

End If

Next

'If the value of count is greater then the value used to check if the data is there or not then the data is written to the file

If count > 0 Then

'Write to File

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

JustUserName = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\Username.txt")

End If

End If

'Increases the size of JustScore by NumOfValues

ReDim JustScore(NumOfValues)

UpdateData() 'Calls Sub Routine by the same name here

'Message Box tells user their data is saved and makes the submit button false so it cannot be spammed and potentially crash the program

MessageBox.Show("Saved")

butSubmit.Visible = False

End Sub

Private Sub butMergeSort\_Click(sender As Object, e As EventArgs) Handles butMergeSort.Click

MergeSort(JustScore) 'Calls the Merge Sort Routine and used it on the array called JustScore

MessageBox.Show("Data is Sorted, You can Now use the Searching Operation") 'Message Box appears when data is sorted

butSearch.Visible = True 'Makes Search Button visible

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Calls the Binary Search Routine here

End Sub

Private Sub UpdateData()

'Data is read to Score Array

Dim ScoreArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\score.txt")

For i = 0 To ScoreArray.Length - 1 'For the number of data there is

ScoreArray(i) = CInt(ScoreArray(i)) 'Data is converted to integer

JustScore(NumOfValues) = ScoreArray(i) 'Stores Data in JustScore

NumOfValues += 1 'Increases the value of NumOfValues by 1

ReDim Preserve JustScore(NumOfValues) 'Redeclares JustScore and inreases its size as NumOfValues has increased, also preserves data that is already in array

Next

End Sub

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length - 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Sub BinarySearch()

'Binary Search Algorithm can be seen below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result 'Changes Position of the Data from the order in array to output the correct position on the leaderboard

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

End Class

Code for Flappy Bird:

***Menu:***

Public Class Form1

Private Sub butStartGame\_Click(sender As Object, e As EventArgs) Handles butStartGame.Click 'The code inside is ran when the Start Game Button is pressed

Me.Close() 'Closes the Form

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Flappy Bird.exe") 'This path leads to the Original game which the player can access and play

End Sub

Private Sub butExit\_Click(sender As Object, e As EventArgs) Handles butExit.Click 'When this Button is pressed the following code is ran

Application.Exit() 'Closes the Program

End Sub

End Class

***Game and Implementation Combined:***

Public Class Form1

#Region "Variables"

Dim Yspeed As Integer = 0 'Vertical Speed of pbBird

Dim Gravity As Integer = 2 'This Value Determines how quick pbBird Falls, the bigger it is the quicker it falls

Dim Pipe(2) As PictureBox 'Acts like an Array with the number of Pipes inside the Parameter

Dim TopPipe(2) As PictureBox 'Acts Like an Array with the Number of Pipes inside the Parameter

Dim GapBetweenPipes As Integer = 475 'Allows the User to change the Gap between the Pipes if they think the game is either too easy or hard

Dim PipeSpeed As Single = 3.5 'Variable controls how fast the Pipes Move

Dim r As Random = New Random 'Used to generate random numbers

Dim Score As Integer 'Holds the Players Score

Dim Lives As Integer = 0 'Stores the Players Lives

Dim InvincibilityTimer As Integer = 1000 'Times how long player can stay invincible

Dim UserInput As String 'Holds UserInput

#End Region

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Randomize() 'Calls the Random Function

UserInput = InputBox("Do you want to play with Random Mode or Not?") 'Lets the User Choose if they want to play with Random Mode or Not

Timer1.Start() 'Starts the named Timer

If UserInput = "Yes" Then 'If Random Mode is Active then the following code is ran

RandomEvent.Start() 'Starts the named Timer

End If

CreatePipes(1) 'Calls the Private Sub with the Same Name

CreateTopPipes(1) 'Calls the Private Sub with the Same Name

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

If UserInput = "Yes" Then

InvincibilityTimer = InvincibilityTimer - 10 'When InvincibiltyTimer is below or equal to 0 then collsion can occur

Yspeed += Gravity 'Everytime Timer Ticks Yspeed increases by 2 so from 2 to 4 to 6 following a geometric progrssion

pbBird.Top += Yspeed 'Increase speed varaible by gravity and Top position of pbBird increases by whatever the speed variable is, starting at 0

For i = 0 To 1 'This For Loop allows the Pipes to move to the Left Hand side of the Screen

Pipe(i).Left -= PipeSpeed

TopPipe(i).Left -= PipeSpeed

If Pipe(i).Location.X < pbBird.Location.X Then

Score += 1

TopPipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Top = r.Next(125, 350) 'Makes sure a Random Size is given each time

TopPipe(i).Top = Pipe(i).Top - GapBetweenPipes 'This makes sure their is a gap of a constant size between the top pipe and bottom pipe

'The IF Statement checks to see if the players score is a multiple of 5 if it is then the player becomes invinsible for a short period of time

If Score Mod 5 = 0 Then

lblIndicator.Visible = True

lblIndicator.Text = ("You are Currently Invinsible")

InvincibilityTimer = 1000

End If

End If

'If Ready is below or equal to 0 then collision detection can occur else it doesn't

If InvincibilityTimer <= 0 Then

lblIndicator.Visible = False

If (pbBird.Bounds.IntersectsWith(Pipe(i).Bounds)) Or (pbBird.Bounds.IntersectsWith(TopPipe(i).Bounds)) Then 'Code makes it so if pbBird touches any Pipe the Game Ends

If Lives = 0 Then

Timer1.Stop()

MessageBox.Show("Your Score is " & Score)

Application.Exit()

End If

End If

End If

Next

'The Code below checks to see if the bird has collided with the top or bottom wall and if the bird does the game endsf

If (pbBird.Bounds.IntersectsWith(pbTopWall.Bounds)) Then

Timer1.Stop()

MessageBox.Show("Your Score is " & Score)

Application.Exit()

ElseIf (pbBird.Bounds.IntersectsWith(pbBottomWall.Bounds)) Then

Timer1.Stop()

MessageBox.Show("Your Score is " & Score)

Application.Exit()

End If

'Displays the Score to the User

lblScore.Text = (Score)

Else

Yspeed += Gravity 'Everytime Timer Ticks Yspeed increases by 2 so from 2 to 4 to 6 following a geometric progrssion

pbBird.Top += Yspeed 'Increase speed varaible by gravity and Top position of pbBird increases by whatever the speed variable is, starting at 0

For i = 0 To 1 'This For Loop allows the Pipes to move to the Left Hand side of the Screen

Pipe(i).Left -= PipeSpeed

TopPipe(i).Left -= PipeSpeed

If Pipe(i).Location.X < pbBird.Location.X Then

Score += 1

TopPipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Top = r.Next(125, 350) 'Makes sure a Random Size is given each time

TopPipe(i).Top = Pipe(i).Top - GapBetweenPipes 'This makes sure their is a gap of a constant size between the top pipe and bottom pipe

End If

If (pbBird.Bounds.IntersectsWith(Pipe(i).Bounds)) Or pbBird.Bounds.IntersectsWith(TopPipe(i).Bounds) Then 'Code makes it so if pbBird touches any Pipe the Game Ends

Timer1.Stop() 'The Timer, Timer1, is stopped

Form2.Score2 = Score 'Score is transferred from this form to form2

MessageBox.Show("Your Score is " & Score) 'MessageBox Appears which tells the user their score

Form2.Show() 'Form2 Opens

Me.Hide() 'This Form Closes

End If

Next

If (pbBird.Bounds.IntersectsWith(pbTopWall.Bounds)) Then 'If the bird Collides with the top wall then the following code runs

Timer1.Stop() 'The Timer, Timer1, is stopped

Form2.Score2 = Score 'Score is transferred from this form to form2

MessageBox.Show("Your Score is " & Score) 'MessageBox Appears which tells the user their score

Form2.Show() 'Form2 Opens

Me.Hide() 'This Form Closes

ElseIf (pbBird.Bounds.IntersectsWith(pbBottomWall.Bounds)) Then 'If the Bird collides with the bottom wall then the following code runs

Timer1.Stop() 'The Timer, Timer1, is stopped

Form2.Score2 = Score 'Score is transferred from this form to form2

MessageBox.Show("Your Score is " & Score) 'MessageBox Appears which tells the user their score

Form2.Show() 'Form2 Opens

Me.Hide() 'This Form Closes

End If

lblScore.Text = (Score) 'Displayes the Users Score to the User

End If

End Sub

Private Sub Form1\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown

If e.KeyCode = Keys.Space Then 'When Space is pressed the following code takes place

Yspeed = -15 'If pbBird is falling down speed will be positive, if pbBird is going up speed has to be negative for pbBird to go up

End If

End Sub

Private Sub RandomEvent\_Tick(sender As Object, e As EventArgs) Handles RandomEvent.Tick

If UserInput = "Yes" Then 'If the user inputs any of the options given Random Mode will work

RandomEventCaller() 'Calls Sub Routine by the same name here

Else

'Do Nothing

End If

End Sub

Private Sub CreatePipes(ByVal Number As Integer)

For i = 0 To Number

Dim pbTemp As New PictureBox 'Variable allows me to Dynamically make new pictureboxes on Form without using pictureboxes so program is more efficient

Me.Controls.Add(pbTemp) 'Adds pbTemp to the Form with the properties of the picturebox being given below

pbTemp.Width = 50

pbTemp.Height = 350

pbTemp.BorderStyle = BorderStyle.FixedSingle

pbTemp.BackColor = Color.Green 'Makes Pipe Green

pbTemp.Top = r.Next(70, 350) 'Generates a Number Between the Range 70-350

pbTemp.Left = i \* 200 + 300 'Uses to Calculate the Number of Pipes as well as their distance from pbBird similiar to a function f(x) on a graph

Pipe(i) = pbTemp

Pipe(i).Visible = True 'Makes the picturebox visible

Next

End Sub

Private Sub CreateTopPipes(ByVal Number As Integer)

For i = 0 To Number

Dim pbTemp As New PictureBox 'Variable allows me to Dynamically make new pictureboxes on Form without using pictureboxes so program is more efficient

Me.Controls.Add(pbTemp) 'Adds pbTemp to the Form with the properties of the picturebox being given below

pbTemp.Width = 50

pbTemp.Height = 350

pbTemp.BorderStyle = BorderStyle.FixedSingle

pbTemp.BackColor = Color.Green 'Makes Pipe Green

pbTemp.Top = Pipe(i).Top - GapBetweenPipes 'Creates a consistent gap between each pipe

pbTemp.Left = i \* 200 + 300 'Uses to Calculate the Number of Pipes as well as their distance from pbBird similiar to a function f(x) on a graph

TopPipe(i) = pbTemp

TopPipe(i).Visible = True 'Makes the picturebox visible

Next

End Sub

Private Sub RandomEventCaller()

Dim x As Integer 'Stores a Random Number

x = r.Next(0, 20) 'Chooses a Random Number Betwen 20 and 0

If x = 0 Then 'If the Number Generated is 0 then the following code runs

InvincibilityTimer = 1000

lblIndicator.Text = ("You are Invinsible")

lblIndicator.Visible = True 'Player gets indicated that they are invinsible

End If

If x = 1 Then 'If the number generated is 1 then the following code is ran

'The player gains 3 points for crossing each pipe

For i = 0 To 1

If Pipe(i).Left < 0 Then

Score += 3

lblIndicator.Text = ("You currently gain 3 Points for each Pipe") 'Informs player of this effect

lblIndicator.Visible = True

TopPipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Top = r.Next(350, 125) 'Makes sure a Random Size is given each time

TopPipe(i).Top = Pipe(i).Top - GapBetweenPipes 'This makes sure their is a gap of a constant size between the top pipe and bottom pipe

End If

Next

End If

If x = 4 Then 'If the Number Generated is 4 then the follwing code is ran

'The player doesn't gain any points for passing a pipe

For i = 0 To 1

If Pipe(i).Left < 0 Then

Score += 0

lblIndicator.Text = ("You currently gain 0 Points for each Pipe") 'Informs player of this effect

lblIndicator.Visible = True

TopPipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Left += 400 'Allow the Pipes to Loop so Pipe keep going to Right Hand Side

Pipe(i).Top = r.Next(350, 125) 'Makes sure a Random Size is given each time

TopPipe(i).Top = Pipe(i).Top - GapBetweenPipes 'This makes sure their is a gap of a constant size between the top pipe and bottom pipe

End If

Next

End If

If x = 5 Then 'If the number generated is 5 then the following code runs

'The player gains one life

Lives += 1

lblIndicator.Text = ("You gained one Life") 'Informs player of this effect

lblIndicator.Visible = True

ElseIf r.Next(0, 20) = 7 Then 'If the Number generated is 7 then the following code runs

'The player instantly dies

lblIndicator.Text = ("You Have Died Instantly") 'Informs player of this effect

lblIndicator.Visible = True

Timer1.Stop() 'Stops the Timer

MessageBox.Show("Your Score is " & Score) 'MessageBox appears telling the player their final score

Application.Exit() 'Closes the Program

End If

If x = 2 Then 'If the number generated is 2 then

'Players bird becomes invisible

While InvincibilityTimer > 0

lblIndicator.Text = ("You are now Invisible") 'Informs player of this effect

lblIndicator.Visible = True

pbBird.Visible = False

End While

End If

If x = 3 Then 'If the RandomNumber generated is 3 then the following code is ran z

lblIndicator.Text = ("Pipes are Now Moving Slower") 'Informs player of this effect

lblIndicator.Visible = True

PipeSpeed = 1 'The speed at which the pipes come slows down

ElseIf r.Next(0, 20) = 6 Then 'If the RandomNumber generated is 6 then the following code is ran

lblIndicator.Text = ("Pipes are Now Moving Faster") 'Informs player of this effect

lblIndicator.Visible = True

PipeSpeed = 6.5 'The Pipes speed up

End If

End Sub

End Class

***Storing Scores:***

Public Class Form2

#Region "Variables"

'Variabe is Used to transfer the Player Score from the game Form to this Form

Public Score2 As Integer

'The Array Below Stores the Usernames and the scores of all players

Dim UsernameArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Username.txt")

'Counter used to count and increase the size of the array

Dim NumOfValues As Integer = 0

'Variable below is used to store just the scores of the User

Dim JustScore(NumOfValues) As Integer

'Variable Below Just stores the Usernames of all the players

Dim JustUserName() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\JustUserName.txt")

#End Region

Private Sub Form3\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

lblScore.Text = Score2 'Displays the current Player their score

UpdateData() 'Calls Sub Routine by the same name here

End Sub

Private Sub butSubmit\_Click(sender As Object, e As EventArgs) Handles butSubmit.Click

'Write to Username File using StreamWriter

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Username.txt", True)

Writer.WriteLine(txtUsername.Text & ", " & lblScore.Text)

End Using

'Write to Score File using StreamWriter

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Score.txt", True)

Writer.WriteLine(lblScore.Text)

End Using

'Variables below make sure that duplicate values aren't added to the JustUserName file

Dim CheckExists() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\JustUserName.txt")

Dim count As Integer = 0

If CheckExists.Length < 1 Then 'If the File is empty data is written to it

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

Else 'If existing data is present then the following code is ran

'Checks if the Score already exists in the database

For i = 0 To CheckExists.Length - 1

If CheckExists(i) = txtUsername.Text Then 'If the data already exists count is set to 0

count = 0

ElseIf CheckExists(i) <> txtUsername.Text Then 'If the data doesn't exist count increases by 1

count += 1

End If

Next

'If Count is above base value (0) so the data being added is new then the data is written to the file

If count > 0 Then

'Write to File

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

'Reads all the values from JustUserName.txt to JustUserName

JustUserName = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Username.txt")

End If

End If

'Redeclares JustScore

ReDim JustScore(NumOfValues)

UpdateData() 'Calles Sub Routine by the same name here

MessageBox.Show("Saved") 'When all the data is saved then a messagebox appears which tells the user their data is saved

butSubmit.Visible = False 'Submit button becomes false so the user cannot spam the subit button

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Calls the Binary Search Routine when the search button is pressed

End Sub

Private Sub UpdateData()

'Data from score file is read to Score Array

Dim ScoreArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Score.txt")

'The FOR loop below converts all data from score array to integer and stores it in JustScore

For i = 0 To ScoreArray.Length - 1

ScoreArray(i) = CInt(ScoreArray(i))

JustScore(NumOfValues) = ScoreArray(i)

NumOfValues += 1 'NumOfValues is incremented by 1

ReDim Preserve JustScore(NumOfValues) 'JustScore is Redimmed to Increase its size and as Preserve is also used data in the array is also kept

Next

End Sub

Private Sub butMergeSort\_Click(sender As Object, e As EventArgs) Handles butMergeSort.Click

MergeSort(JustScore) 'Calls the MergeSort Algorithm here with the values of the JustScore array

MessageBox.Show("Data is Sorted, You can Now use the Searching Operation") 'MessageBox appears when data is sorted telling the user their data is sorted

butSearch.Visible = True 'Button used to use Binary Search is made visible, it's originally hidden as binary search cannot be ran when the data isn't sorted

End Sub

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length - 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Sub BinarySearch()

'The Binary Search Algorithm can be seen below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result 'Makes it so that when the position is given the position is inversed as originally when data is sorted largest value would be last value so the position given would be a greater number then it should be for a high score so for instance it would be 10 instead of 1

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

End Class

Code for Pong:

***Menu:***

Public Class Form1

'When the Start Button is Pressed the Game Program is launched and the Game starts

Private Sub Start\_Click(sender As Object, e As EventArgs) Handles Start.Click

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Pong Remakes\Pong - Core DONE\Pong Game\WindowsApp1\bin\Debug\WindowsApp1.exe")

Environment.Exit(0)

End Sub

'When the Exit Button is pressed the Game Exits the Program

Private Sub Close\_Click(sender As Object, e As EventArgs) Handles Close.Click

Environment.Exit(0)

End Sub

End Class

***Game and Implementation:***

Public Class pongMain

#Region "Globals"

Dim Speed As Single = 15 'Ball's Speed

Dim rndInst As New Random() 'Random Instance

Dim BallShot As Boolean = False 'Determines if Ball has been shot or not

Dim xVel As Single = Math.Cos(rndInst.Next(5, 10)) \* Speed 'Velcoity of the Ball Horizontally (on the x-axis)

Dim yVel As Single = Math.Sin(rndInst.Next(5, 10)) \* Speed 'Velocity of the Ball Vertically (on the y-axis)

Dim AIActive As Boolean = True 'Holds if the AI is on or not

Dim AIChecker As Integer = 400 'Counts down the time until the AI becomes active

Dim AICoolDown As Integer = 100 'AI Cooldown period

Dim OriginalLocation As New Point 'Stores the Balls Location

Dim r As Random = New Random 'Used to generate and use Random Numbers

Dim CheatBox As Boolean = False 'Used to Debug by allowing me to change the score

Dim UserInput As String 'Holds UserInput

'The Players Score

Dim plrScore As Integer = 0

Dim compScore As Integer = 0

'Variables used for Mad Mode

Dim Count As Integer = 0 'Count is used to look at the number of different possibilites for Mad Mode

Dim MadModeEnabled As Boolean = False 'Used to check if Mad Mode conditions are running

Dim AdvBallSpeed As Integer = 10 'Controls the Balls Speed for the Enhancements(Mad Mode Region)

Dim CheatBoxCount As Boolean = False 'Used for Testing by allowing me to change the Count variable when needed

Dim t As Integer 'Counts up the number of times an event is allowed to run

#End Region

#Region "Form Load Event"

Private Sub pongMain\_Load(sender As Object, e As EventArgs) Handles Me.Load

Randomize() 'Enables the Randomize Function

UserInput = InputBox("Do you Want to Play in Mad Mode") 'Asks the User if they want to play with my implementation of Mad Mode in the game

If UserInput = "Yes" Then 'If the User wants to play in the noted mode then the following code is ran

madmodeTimer.Start() 'The Corresponding Timer is started

gameTimer.Start() 'The named timer starts ticking

AITimer.Start() 'The named timer starts ticking

Else 'If the inital condition isn't met then the following code is ran

madmodeTimer.Stop() 'The Timer is Stopped

gameTimer.Start() 'The named timer starts ticking

AITimer.Start() 'The named timer starts ticking

End If

OriginalLocation = gameBall.Location 'When the Form Loads the Balls Original Location is stored

End Sub

#End Region

#Region "Move the Paddle According to the Mouse"

'Move the Paddle According to the Mouse Position Vertically

Private Sub pongMain\_MouseMove(ByVal sender As Object, ByVal e As MouseEventArgs) Handles Me.MouseMove

If e.Y > 5 And e.Y < Me.Height - 40 - paddlePlayer.Height Then

paddlePlayer.Location = New Point(paddlePlayer.Location.X, e.Y)

End If

End Sub

#End Region

#Region "Main Timer"

Private Sub gameTimer\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles gameTimer.Tick

If BallShot = True Then

'Move the Ball

gameBall.Location = New Point(gameBall.Location.X + xVel, gameBall.Location.Y + yVel)

End If

'Set the Computer Player to Move According to the Balls Position.

If AIActive = True Then

If gameBall.Location.Y > pbTopWall.Location.Y And gameBall.Location.Y < 333 Then 'Put boundaries on Paddle Movement for Computer so it Doesn't go off the screen

paddleComputer.Location = New Point(paddleComputer.Location.X, gameBall.Location.Y)

End If

End If

'Check for Top Wall

If gameBall.Bounds.IntersectsWith(pbTopWall.Bounds) Then

gameBall.Location = New Point(gameBall.Location.X, 0)

yVel = -yVel 'Inverses Vertical Speed

End If

'Check for Bottom Wall

If gameBall.Bounds.IntersectsWith(pbBottomWall.Bounds) Then

gameBall.Location = New Point(gameBall.Location.X, Me.Height - gameBall.Size.Height - 45)

yVel = -yVel 'Inverses Vertical Speed

End If

'Check for Player Paddle

If gameBall.Bounds.IntersectsWith(paddlePlayer.Bounds) Then

gameBall.Location = New Point(paddlePlayer.Location.X - gameBall.Size.Width, gameBall.Location.Y)

xVel = -xVel 'Inverses Horizontal Speed

End If

'Check for Computer Paddle

If gameBall.Bounds.IntersectsWith(paddleComputer.Bounds) Then

gameBall.Location = New Point(paddleComputer.Location.X + paddleComputer.Size.Width + 1, gameBall.Location.Y)

xVel = -xVel 'Inverses Horizontal Speed

End If

'Check for Left Wall

If gameBall.Bounds.IntersectsWith(pbLeftWall.Bounds) Then 'If Ball hits wall behind comp paddle then the following code is ran

plrScore += 1 'Increases Players Score

AIActive = True 'Turns AI On

gameBall.Location = OriginalLocation 'Moves Ball Back to Original Location

BallShot = False 'Makes it so the ball isn't moving anymore

plrScoreDraw.Text = Convert.ToString(plrScore) 'Writes Score

End If

'Check for Right Wall

If gameBall.Bounds.IntersectsWith(pbRightWall.Bounds) Then 'If the ball hits wall behind the player then the following code is ran

compScore += 1 'compScore increases

AIActive = True 'Turns AI On

BallShot = False 'Makes it so the ball doesn't move anymore

gameBall.Location = OriginalLocation 'Moves Ball Back to original Location

compScoreDraw.Text = Convert.ToString(compScore) 'Writes Score

End If

'Once the Player reaches 10 points they have beat the AI

If plrScore = 10 Then

'Timers are Stopped

gameTimer.Stop()

AITimer.Stop()

'MessageBox Appears which tells the player they have beat the AI

MsgBox("Well Done, You have Beat the AI. Do you want to Carry on Playing, if you do you Will Recieve a Extra Point.")

plrScore = plrScore + 1 'Player gains one Point

plrScoreDraw.Text = Convert.ToString(plrScore) 'Displays the Player Score

'Starts the Timers

gameTimer.Start()

AITimer.Start()

End If

'If the Computer has 10 Points then the player has lost

If compScore = 10 Then

'The Timers are stopped

gameTimer.Stop()

AITimer.Stop()

'MessageBox Appears Telling the Player that they have lost

MsgBox("You Lose")

Application.Exit() 'Closes the Program

End If

'If the CheatBox is enabled then the timers stop and I am given an Input Box where I can change the Score to what I want it to be

If CheatBox = True Then

gameTimer.Stop()

AITimer.Stop()

plrScore = (InputBox("What do You want your Score to be?"))

plrScoreDraw.Text = (plrScore)

gameTimer.Start()

AITimer.Start()

CheatBox = False

End If

'The IF statement checks to see if the ball has passed the boundary or not if it has then the ball is moved back to the centre. This is just for error handling just in case the ball goes past but the game does not respond accordingly

If gameBall.Location.X < pbLeftWall.Location.X Then

BallShot = False

gameBall.Location = OriginalLocation

ElseIf gameBall.Location.X > pbRightWall.Location.X Then

BallShot = False

gameBall.Location = OriginalLocation

End If

End Sub

#End Region

#Region "Enemy Timer"

Private Sub AITimer\_Tick(sender As Object, e As EventArgs) Handles AITimer.Tick

AIChecker = AIChecker - 100

If AIActive = False Then 'If the AI isn't active the following code runs

AICoolDown = AICoolDown - 50 'The Cooldown period starts to count down

'If AI CoolDown goes below 0 it's set to 0

If AICoolDown < 0 Then

AICoolDown = 0

End If

'Once AI Cool Down period is 0 and the AI is still not active then the AI is turned on to be active and the cool down period is reset

If AIActive = False And AICoolDown = 0 Then

AIActive = True

AICoolDown = 100

End If

End If

'If the AIChecker goes below 0 it's set to 0

If AIChecker < 0 Then

AIChecker = 0

End If

'Once the AI checker is 0, the AI is turned on

If AIChecker = 0 Then

AIActive = True

End If

'If the players score is less than 5 then they have an even chance of getting the AI to turn off

If plrScore < 5 Then

If r.Next(0, 10) Mod 2 = 0 Then

AIActive = True

Else

AIActive = False

End If

End If

'If the Players Score is greater than 5 but less than 10 the player has more chance for the AI to stay active

If plrScore > 5 And plrScore < 10 Then

If r.Next(1, 10) Mod 2 = 0 Then

AIActive = True

Else

AIActive = False

End If

End If

'If the players score is greater then 10 and less than 15 then the player has an increased chance of the AI staying active

If 10 < plrScore And plrScore < 15 Then

If r.Next(1, 7) Mod 2 = 0 Then

AIActive = True

Else

AIActive = False

End If

End If

'If the Players Score is greater than 15 the player has high chance of the AI turning on

If 15 > plrScore Then

If r.Next(1, 3) Mod 2 = 0 Then

AIActive = False

Else

AIActive = True

End If

End If

End Sub

#End Region

#Region "Mad Mode"

Private Sub madmodeTimer\_Tick(sender As Object, e As EventArgs) Handles madmodeTimer.Tick

MadMode()

End Sub

Private Sub MadMode()

If Count = 0 And BallShot = True Then 'If the Variable count is directly = 0 and the ball is in play then the follwing code is executed

If plrScore >= 5 Then 'If the players score is greater than or equal to 5 then the following code is rexecuted

If plrScore Mod 2 = 0 Then 'As long as the Players Score is even the following code will ran

gameBall.Location = New Point(gameBall.Location.X + xVel \* 2, gameBall.Location.Y + yVel) 'Moves the Ball, this piece of code speeds the ball up if the condition is fuffiled otherwise the balls is slowed down

MadModeEnabled = True 'Makes this true so the rest of code knows that mad mode is running

Else

gameBall.Location = New Point(gameBall.Location.X + xVel \* AdvBallSpeed / 5, gameBall.Location.Y + yVel) 'Moves the Ball, this piece of code halves the speed of the ball so slows it down if the condition isn't fufilled

MadModeEnabled = True 'Lets the rest of the program know that Mad Mode is enabled

End If

Count = Count + 1 'One is added to the count to allow the other Pieces of code to function properly

End If

End If

If Count = 1 And BallShot = True And plrScore >= 10 Then 'If the Variable Count is 1 and the ball is in play and the player score is greater than or equal to 10 then the follwing code is executed

If r.Next(0, 5) Mod 2 = 0 AndAlso t = 0 Then 'If the Generated number is even then the following code is ran

MadModeEnabled = False 'Disables Mad Mode

AIActive = True 'The AI becomes active and will now track the ball

t += 1 'Increments t by 1

Else

AIActive = False 'Disables the AI if the condition isn't fufilled

End If

Count = Count + 1 'One more is added to Count increasing its valueto 2

End If

If Count = 2 And BallShot = True Then 'If Count is 2 and the Ball is in play then the following code is executed

If r.Next(0, 1) Mod 2 = 0 And plrScore >= 15 Then 'If the Random Number generated in the range is even and the players score is at least 15 then the following code is ran

t = 0 't is reset

If gameBall.Location.X = 450 Then 'If the ball is close to the player horizontally (on the x-axis)

If r.Next(0, 2) Mod 2 = 0 Then

gameBall.Location = New Point(gameBall.Location.X + xVel, gameBall.Location.Y + yVel \* AdvBallSpeed) 'Moves the Ball, this piece of code makes the ball move faster vertically

End If

End If

Count = Count + 1 'One More is added to Count increasing its value to 3

End If

End If

If Count = 3 And BallShot = True And plrScore >= 25 Then

Dim temp As Integer 'Temporarly Stores a Value

t = 0 'Resets t

If r.Next(0, 3) Mod 2 = 0 Then

temp = xVel 'Temp stores the current value of xVel

xVel = -xVel 'xVel gets inverted

gameBall.Location = New Point(gameBall.Location.X + xVel \* 2, gameBall.Location.Y + yVel) 'Code allows ball to move

xVel = temp 'xVel gets it old value back

End If

End If

Count = r.Next(0, 4) 'A random number generated between the range 0 - 4 is given to represent Count so the AI acts randomly keeping the player on their toes at all times

If CheatBoxCount = True Then 'If the CheatBoxCount Variable is True through the Key Down event than the following code is executed

gameTimer.Stop() 'The game timer is stopped

AITimer.Stop() 'The AI timer is stopped

madmodeTimer.Stop() 'The named timer stops ticking

Count = (InputBox("What do You want the Count to be?")) 'An input Box appears which allows me to change the count variable for testing and debugging

gameTimer.Start() 'The game timer is started again

AITimer.Start() 'The AI timer is also started again

madmodeTimer.Start() 'Starts the named tiemr

CheatBoxCount = False 'The Variable used in the statement is set to False I can access it multiple times at different times when the game is running

End If

End Sub

#End Region

#Region "Pause Menu and User Input"

'When the Restart Button is Pressed the Whole Program will Restart

Private Sub Restart\_Click(sender As Object, e As EventArgs) Handles Restart.Click

Application.Restart()

End Sub

'When this Button is Pressed the Game goes back to the Main Menu

Private Sub Backto\_Click(sender As Object, e As EventArgs) Handles Backto.Click

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Pong Remakes\Pong - Core DONE\Pong Menu\Pong Menu\WindowsApp1\bin\Debug\WindowsApp1.exe")

Environment.Exit(0)

End Sub

Private Sub pongMain\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown

'When the Q Key is Pressed the Pause Menu will Display hiding everything on the Screen, also shows the position of the Cursor

If e.KeyValue = Keys.Q Then

gameBall.Hide()

paddlePlayer.Hide()

paddleComputer.Hide()

Restart.Visible = True

Backto.Visible = True

Cursor.Show()

ElseIf e.KeyValue = Keys.W Then

gameBall.Show()

paddleComputer.Show()

paddlePlayer.Show()

Restart.Visible = False

Backto.Visible = False

Cursor.Hide()

ElseIf e.KeyValue = Keys.Space Then 'Allows the Ball to be Fired

BallShot = True

ElseIf e.KeyValue = Keys.A Then 'Used to Access the CheatBox

CheatBox = True

ElseIf e.KeyValue = Keys.Z Then 'Used to change Count for Mad Mode

CheatBoxCount = True

End If

End Sub

#End Region

End Class

Code for Space Invaders:

***Menu:***

Public Class gameForm

Private Sub startButton\_Click(sender As Object, e As System.EventArgs) Handles startButton.Click 'The Following Code is ran when the start game Button is pressed

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Space Invader Main Game.exe") 'This path leads to the original game which the player can access and play

Environment.Exit(0) 'Terminates (Ends/Closes) this Form

End Sub

Private Sub startcustomButton\_Click(sender As Object, e As System.EventArgs) Handles startcustomButton.Click 'The Following code is ran when the start custom game button is pressed

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Shoot em Up - Core DONE\Shoot em Up\bin\Debug\Shoot em Up.exe") 'This path leads to the custom game which the player can access and play

Environment.Exit(0) 'Terminates (Ends/Closes) this Form

End Sub

Private Sub exitButton\_Click(sender As Object, e As System.EventArgs) Handles exitButton.Click 'The Following Code is ran when the exit button is pressed

Environment.Exit(0) 'Terminates (Ends/Closes) this form

End Sub

End Class

***Game:***

Public Class gameForm

#Region "Variables"

Dim MissleShoot As Boolean = False 'Tells the Computer if the Missile is allowed to shoot or not

Dim EnemyArray(32) As PictureBox 'Holds the enemys in an array

Dim EnemyLocation(32) As Point 'Keeps track of the location of each enemy in an array

Dim EnemyShot(32) As PictureBox 'Keeps track of the bullet of each enemy

Dim EnemyShot2(32) As Boolean 'Used to tell if an enemy has shot or not

Dim BarricadeLife(2) As Single 'Used to Hold the Barricade Lives

Dim EnemySpeed As Integer = 3 'Holds how fast the enemys move

Dim MoveRight As Boolean = False 'Used to tell if the player can move right or not

Dim MoveLeft As Boolean = False 'Used to tell if the player can move left or not

Dim PlayerSpeed As Integer = 4 'Holds the Players Speed

Dim Score As Integer = 0 'Holds the Score of the Player

Dim Lives As Integer = 3 'Holds the Lives of the User

Dim Complete As Integer = 0 'Holds the number of enemys killed

Dim Level As Integer = 1 'Holds the level of the player

#End Region

Private Sub GameForm\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown

Dim i As Integer = 0 'Used for Loops

Select Case e.KeyValue

Case Keys.Right 'When the Right Key is Pressed MoveRight becomes True

MoveRight = True

Case Keys.Left 'When the Left Key is Pressed MoveLeft becomes True

MoveLeft = True

Case Keys.Space 'When the Space Key is pressed the following code is ran

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

MissleShoot = True

pbMissile.Enabled = True

pbMissile.Visible = True

End Select

End Sub

Private Sub GameForm\_KeyUp(sender As Object, e As KeyEventArgs) Handles Me.KeyUp

Select Case e.KeyValue

Case Keys.Right

MoveRight = False 'When the Right Key is Released MoveRight becomes False

Case Keys.Left

MoveLeft = False 'When the Left Key is Released MoveLeft Becomes False

End Select

End Sub

Private Sub GameForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Randomize() 'Calls Randomize through FrameWork

InitaliseEnemys() 'Calls Sub Routine by the same name here

EnemyShoot(32) 'Calls Sub Routine by the same name here

'When the Form Loads each Barricade is given three lives, thus each barricade can endure three hits

For i = 0 To 2

BarricadeLife(i) = 3

Next

End Sub

Private Sub TimerGame\_Tick(sender As Object, e As EventArgs) Handles timerGame.Tick 'The Following events occur everytime TimerGame Ticks

If MissleShoot = False Then

pbMissile.Visible = False

End If

'If MoveRight is True and the Player then the player moves right the same number of pixels as the value of playerspeed

If MoveRight = True Then

pbPlayer.Left += PlayerSpeed

End If

'If MoveLeft is True then the player moves left the same number of pixels as the value of playerspeed

If MoveLeft = True Then

pbPlayer.Left -= PlayerSpeed

End If

If pbPlayer.Bounds.IntersectsWith(pbLeftWall.Bounds) Then 'If the player reaches the boundary of the form the following code is ran

pbPlayer.Location = New Point(0, 431) 'The player is placed at the boundary of the form

End If

If pbPlayer.Bounds.IntersectsWith(pbRightWall.Bounds) Then 'If the player reaches the boundary of the form then the following code is ran

pbPlayer.Location = New Point(794, 431) 'The player is placed at the boundary

End If

If MissleShoot = True Then

pbMissile.Top -= 5 'The Missile Moves up

End If

If pbMissile.Location.Y < 1 Then 'If the Missile are off the screen then then the following code is executed

pbMissile.Visible = False 'The Missile Dissappears

MissleShoot = False

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

End If

For j = 0 To 32 'This Loop checks to see if the Missile Hits the Enemy or Not

If EnemyArray(j).Visible = True Then

If (pbMissile.Bounds.IntersectsWith(EnemyArray(j).Bounds)) Then 'If the missile and enemy at the j respectivily collide the following code is executed

pbMissile.Visible = False 'The missile dissapears

MissleShoot = False

EnemyArray(j).Visible = False 'The Enemy Dies (Dissapears)

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

Complete += 1 'Increments Complete by 1 as an enemy was killed

'Checks to see which typeof enemy was killed and adds the correct amount of points to the score

If EnemyArray(j).Tag = ("Enemy2") Then

Score += 20

End If

If EnemyArray(j).Tag = ("Enemy3") Then

Score += 40

End If

If EnemyArray(j).Tag = ("Enemy1") Then

Score += 10

End If

lblScore.Text = ("Score: " & Score) 'Displays the Score in the Label to show the User what their score is

End If

End If

Next

LevelControl() 'Calls Sub Routine by the same name here

End Sub

Private Sub TimerEnemy\_Tick(sender As Object, e As EventArgs) Handles timerEnemy.Tick

EnemyMoveMent() 'Calls Sub Routine by the same name here

EnemyShotMovement() 'Calls Sub Routine by the same name here

Barricade() 'Calls Sub Routine by the same name here

For i = 0 To 32 'Loops through all the enemys

If (pbPlayer.Bounds.IntersectsWith(lblBoundary.Bounds)) Then 'If the enemy reaches the label boundary the follwoing code is ran

Dim j As Integer 'Used for Loops

'Resets the Positions of all the enemy

For j = 0 To 32

EnemyArray(j).Location = EnemyLocation(j)

Next

'Makes all the Life indicators invisible

pbLife.Visible = False

pbLife2.Visible = False

pbLife3.Visible = False

Form2.Score2 = Score 'transfers teh Player score to Form2

Form2.Show() 'Opens Form2

timerGame.Stop() 'The Main Timer Stops

timerEnemy.Stop() 'The Enemy Timer Stops

MsgBox("Game Over Your Score is " & Score) 'MessageBox appears which states game over with the players score

Me.Hide()

End If

Next

End Sub

Private Sub InitaliseEnemys()

Dim i As Integer

For Each Picturebox In Me.Controls 'For every object in the Form the following code is ran

If Picturebox.Tag = ("Enemy1") Then 'If the object is a picturebox and is also a enemy then the IF statement is executed. The AndAlso will make sure the code is only ran when both statements (considered as Boolean) are True

EnemyArray(i) = Picturebox 'Every object which satisfied the given conditions is added to the Array

i += 1 'The value for i is incremented by 1 until all the objects in the Form are checked over

ReDim Preserve EnemyArray(i) 'Each time the loop is ran through it's redim (redeclared) with Preserve to stop any previous data which was added from being deleted

ElseIf Picturebox.Tag = ("Enemy2") Then

EnemyArray(i) = Picturebox 'Every object which satisfied the given conditions is added to the Array

i += 1 'The value for i is incremented by 1 until all the objects in the Form are checked over

ReDim Preserve EnemyArray(i) 'Each time the loop is ran through it's redim (redeclared) with Preserve to stop any previous data which was added from being deleted

ElseIf Picturebox.Tag = ("Enemy3") Then

EnemyArray(i) = Picturebox 'Every object which satisfied the given conditions is added to the Array

i += 1 'The value for i is incremented by 1 until all the objects in the Form are checked over

ReDim Preserve EnemyArray(i) 'Each time the loop is ran through it's redim (redeclared) with Preserve to stop any previous data which was added from being deleted

End If

Next

'Loop gives each Enemy their right location

For i = 0 To 32

EnemyLocation(i) = EnemyArray(i).Location

Next

End Sub

Private Sub EnemyMoveMent()

'Each Enemy Moves left or right until they reach the boundary then they move down

For i = 0 To 32

EnemyArray(i).Left = EnemyArray(i).Left + EnemySpeed

Next

If PictureBox13.Bounds.IntersectsWith(pbRightWall.Bounds) Then

EnemySpeed = EnemySpeed \* -1

For i = 0 To 32

EnemyArray(i).Top += 10

Next

End If

If PictureBox2.Bounds.IntersectsWith(pbLeftWall.Bounds) Then

EnemySpeed = EnemySpeed \* -1

For i = 0 To 32

EnemyArray(i).Top += 10

Next

End If

End Sub

Private Sub EnemyShoot(ByVal Number As Integer)

'Code Below dynamically creates each missile when one is shot by the enemy

For i = 0 To Number

Dim EnemyMissile As New PictureBox

Me.Controls.Add(EnemyMissile)

EnemyMissile.Height = 10

EnemyMissile.Width = 5

EnemyMissile.BorderStyle = BorderStyle.FixedSingle

EnemyMissile.BackColor = Color.GhostWhite

EnemyMissile.Top = EnemyArray(i).Top + EnemyArray(i).Height / 2 - EnemyMissile.Height / 2

EnemyMissile.Left = EnemyArray(i).Left + EnemyArray(i).Width / 2 - EnemyMissile.Width / 2

EnemyMissile.BringToFront()

EnemyShot(i) = EnemyMissile

EnemyShot(i).Visible = False

Next

End Sub

Private Sub EnemyShotMovement()

'Variables below are used for Loops

Dim j As Integer

'Loops through to check if the enemy has shot if it hasn't it has the chance to become true through the random

For i = 0 To 32

If EnemyShot2(i) = False Then

j = CInt(Int((1000 \* Rnd()) + 1))

If j = 1000 Then

EnemyShot2(i) = True

End If

End If

Next

'Checks to see if the Enemy has shot if it has then the bullet becomes visible and the missile moves down

For i = 0 To 32

If EnemyShot2(i) = True Then

EnemyShot(i).Visible = True

EnemyShot(i).Top += 20

End If

If (EnemyShot(i).Bounds.IntersectsWith(pbPlayer.Bounds)) Then 'If a bullet hits the player then following code is ran

Me.Controls.Remove(EnemyShot(i)) 'Enemy Shot is removed from the form

EnemyShot(i).Location = EnemyArray(i).Location 'Moves the bullet back to the position of the enemy

Lives = Lives - 1 'Player loses one life

'Code below checks the life the player is on and changes the UI accordingly

If Lives = 2 Then

pbLife.Visible = False

ElseIf Lives = 1 Then

pbLife2.Visible = False

End If

End If

'If the Shot goes out of the Form then the Shot is removed and is moved back to its original location

If EnemyShot(i).Top > Me.Height Then

EnemyShot2(i) = False

EnemyShot(i).Visible = False

EnemyShot(i).Location = EnemyLocation(i)

EnemyShot(i).Top = EnemyShot(i).Top + 30

EnemyShot(i).Left = EnemyShot(i).Left + 15

End If

Next

If Lives = 0 Then 'If Player has 0 lives then the follwing code is ran

pbLife3.Visible = False 'Life Indicator is made invisible

Form2.Score2 = Score 'Player Score is transferred from this form to form2

timerGame.Stop() 'The Main Timer Stops

timerEnemy.Stop() 'The Enemy Timer Stops

MsgBox("Game Over Your Score is " & Score) 'MessageBox appears which states game over with the players score

Form2.Show() 'Open Form2

Me.Hide() 'Closes this Form

End If

End Sub

Private Sub Barricade()

Dim j As Integer 'Used for Loops

Dim BarricadeArray(2) As PictureBox 'Stores all the Barricades in an Array

'The Code below loops through each picturebox in the form and adds it to the array if it has the barricade tag

For i = 0 To 2

For Each Picturebox In Me.Controls

If Picturebox.Tag = ("Barricade") Then

BarricadeArray(i) = Picturebox

i += 1

ReDim Preserve BarricadeArray(i)

End If

Next

Next

'Nested FOR Loop checks to see if the player shot the barricade, if they did then the barricade loses a life and the image of the barricade changes accordingly

For j = 0 To 2

For i = 0 To 1

If BarricadeLife(j) > 0 AndAlso BarricadeArray(j).Visible = True Then

If (pbMissile.Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

pbMissile.Visible = False

MissleShoot = False

pbMissile.Location = New Point(pbPlayer.Location.X + 26, pbPlayer.Location.Y)

BarricadeLife(j) = BarricadeLife(j) - 1

End If

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

Next

Next

'Nested FOR Loops checks to see if a Enemy Missile has hit the barricade, this causes the barricade to lose one life and the barricade image changes accordingly

For j = 0 To 2

For i = 0 To 32

If (EnemyShot(i).Bounds.IntersectsWith(BarricadeArray(j).Bounds)) Then

Me.Controls.Remove(EnemyShot(i))

EnemyShot(i).Location = EnemyArray(i).Location

BarricadeLife(j) = BarricadeLife(j) - 0.25

If BarricadeLife(j) >= 2 And BarricadeLife(j) < 3 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade1

ElseIf BarricadeLife(j) >= 1 And BarricadeLife(j) < 2 Then

BarricadeArray(j).Image = My.Resources.DamagedBarricade2

ElseIf BarricadeLife(j) >= 0 And BarricadeLife(j) < 1 Then

BarricadeArray(j).Visible = False

End If

If BarricadeLife(j) < 0 Then

BarricadeLife(j) = 0

End If

End If

Next

Next

End Sub

Private Sub LevelControl()

'If all the enemys have been killed then the level is incremented and shown to the user and the value for complete is reset

If Complete = 33 Then

Level += 1

lblLevel.Text = ("Level: " & Level)

Complete = 0

'Code below resets all enemy locations and makes them all visible again

For i = 0 To 32

EnemyArray(i).Location = EnemyLocation(i)

EnemyArray(i).Visible = True

Next

End If

End Sub

***Storing Scores:***

Public Class Form2

#Region "Variables"

Public Score2 As Integer 'Gets the Score of the User from the game form and tranfers it to this form

'Variables below Read Files and Store the Username with the Score and Just the Username in arrays respectivly

Dim UsernameArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Username.txt")

Dim JustUserName() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\JustUserName.txt")

Dim NumOfValues As Integer = 0 'Used to increase the size of the JustScore Array

Dim JustScore(NumOfValues) As Integer

#End Region

Private Sub Form3\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

lblScore.Text = Score2 'Writes the current Users score to the label

UpdateData() 'Calls Sub Routine by the same name here

End Sub

Private Sub butSubmit\_Click(sender As Object, e As EventArgs) Handles butSubmit.Click

'Writes Username and score to the username text file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Username.txt", True)

Writer.WriteLine(txtUsername.Text & ", " & lblScore.Text)

End Using

'Writes Users Score to Score Text File

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Score.txt", True)

Writer.WriteLine(lblScore.Text)

End Using

'Variables below are used to spot duplicated data and prevent it from being saved to the text file

Dim CheckExists() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\JustUserName.txt") 'Reads username

Dim count As Integer = 0

If CheckExists.Length < 1 Then 'If the File is empty then Data is written to it

'Writes the Users Username to JustUsername text file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

Else 'If the Text File isn't empty then the following code is ran

'Checks if the Score already exists in the database

For i = 0 To CheckExists.Length - 1

If CheckExists(i) = txtUsername.Text Then 'If the score already exists count is set to 0

count = 0

ElseIf CheckExists(i) <> txtUsername.Text Then 'If Score doesn't already exist then count is incremented by 1

count += 1

End If

Next

If count > 0 Then 'If count is greater then its base value then data is written to file

'Write Just the Username to JustUsername File

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(txtUsername.Text)

End Using

'Reads Just Usernames to JustUserName Array

JustUserName = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Username.txt")

End If

End If

ReDim JustScore(NumOfValues) 'Redeclares the JustScore array

UpdateData() 'Calls Sub Routine by the same name here

MessageBox.Show("Saved") 'MessageBox Appears which tells the user that their data is saved

butSubmit.Visible = False 'Makes the Submit Button Invisible

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Calls the BinarySearch Algorithm here

End Sub

Private Sub UpdateData()

'Reads the Score text file to array

Dim ScoreArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Score.txt")

'Code belwo converts all of the Value in ScoreArray to Integer and into the JustScore Array

For i = 0 To ScoreArray.Length - 1

ScoreArray(i) = CInt(ScoreArray(i))

JustScore(NumOfValues) = ScoreArray(i)

NumOfValues += 1 'Value of NumOfValues is incremented

ReDim Preserve JustScore(NumOfValues) 'JustScore is Redeclared and has its size increased due to NumOfValues and as Preserve is used all previous data is also kept intact

Next

End Sub

Private Sub butMergeSort\_Click(sender As Object, e As EventArgs) Handles butMergeSort.Click

MergeSort(JustScore) 'Calls the Merge Sort Algorithm here and passes the JustScore Array through in the parameters

MessageBox.Show("Data is Sorted, You can Now use the Searching Operation") 'When the data is sorted, a message box appears which informs the user that the data is sorted

butSearch.Visible = True 'Makes search button visible, so the user can now use binary search to search for their score

End Sub

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length - 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Sub BinarySearch()

'Binary Search Algorithm can be Seen Below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result 'Inverses the Positon so greatest value is at the top of the array instead of being the last value

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

End Class

***Implememtation:***

Public Class Form1

#Region "Variables"

Dim MoveRight As Boolean = False 'This Variable controls wheather the Ship moves Right or Not

Dim MoveLeft As Boolean = False 'This Variable controls wheather the Ship moves Left or Not

Dim MaxMissileNumber As Integer = 5 'This Variable holds the max number of Missiles, this value can be changed for user accessbility or if the game seems to difficult

Dim MissileArray() As PictureBox 'This Variable allows me to have multiple Missiles on the screen at one given time, currently I have hardcoded the value to 5

Dim MissileNumber As Integer = 0 'This Varaible is used to allocate each Missile released a Number

Dim MissileOnScreen() As Boolean 'This Variable is used to count the number of Missiles on the Screen and will be used to limit the number that can be on the screen. In this case it has been given the value from the MaxMissileNumber Variable so if the value needs changing it can be if needed

Dim MissileSpeed As Integer 'This Variable controls how fast a missile move

Dim MaxEnemyNumber As Integer = 5 'This value will hold the Maximum number of Enemy on the Screen

Dim EnemyArray() As PictureBox 'This Variable allows me to have the number of enemys on the screen to come and go, for instance if one enemy is defeated it will be removed from the array if an enemy is created the it will be added to the array as long as the MaxEnemyNumber Variable isn't exceeded

Dim EnemyOnScreen() As Boolean 'This Variable is used to count the number of enemys on the screen and will be used to linit the number that can be on the screen at one instant. In this case it has been given the same value as MaxEnemyNumber so the amount can be changed if needed

Dim Score As Integer = 0 'This Variable is Used to save the Score of the User

Dim EnemySpeed As Single = 3 'This Variable is Used to alter the Enemys speed as they move. This is represented as a Single which is a decimal with one Decimal number after the decimal point

Dim Sound As New Media.SoundPlayer 'This Variable is declared as a Sound Object used to run sound Files

Dim UserInput As String 'This Variable will hold the Users Response to Input Boxes

#End Region

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load 'This Piece of Code runs as soon as the Form is loaded up

UserInput = InputBox("Do you Want to Change some Settings?") 'If the correct response is given the user can edit their settings

If UserInput = ("Yes") Then

MaxMissileNumber = InputBox("What's the Max Missile Number you Want?") 'Asks the User How Many Missiles they want

ReDim MissileArray(MaxMissileNumber) 'This Variable allows me to have multiple Missiles on the screen at one given time, currently I have hardcoded the value to 5

ReDim MissileOnScreen(MaxMissileNumber) 'This Variable is used to count the number of Missiles on the Screen and will be used to limit the number that can be on the screen. In this case it has been given the value from the MaxMissileNumber Variable so if the value needs changing it can be if needed

MaxEnemyNumber = InputBox("How many Enemy's Do you Want?") 'Asks the User how many Enemys they want

MaxEnemyNumber = MaxEnemyNumber - 1

ReDim EnemyArray(MaxEnemyNumber) 'This Variable allows me to have the number of enemys on the screen to come and go, for instance if one enemy is defeated it will be removed from the array if an enemy is created the it will be added to the array as long as the MaxEnemyNumber Variable isn't exceeded

ReDim EnemyOnScreen(MaxEnemyNumber) 'This Variable is used to count the number of enemys on the screen and will be used to linit the number that can be on the screen at one instant. In this case it has been given the same value as MaxEnemyNumber so the amount can be changed if needed

MissileSpeed = InputBox("How fast do you want your Missiles to Move?") 'Will allow the User to Chane how fast the Missiles Move

EnemySpeed = InputBox("How Fast Do You Want the Enemys to Move?") 'Will affect how fast the enemy move

Timer1.Start()

scoreTimer.Start()

enemyTimer.Start()

Else

ReDim MissileArray(MaxMissileNumber) 'This Variable allows me to have multiple Missiles on the screen at one given time, currently I have hardcoded the value to 5

ReDim MissileOnScreen(MaxMissileNumber) 'This Variable is used to count the number of Missiles on the Screen and will be used to limit the number that can be on the screen. In this case it has been given the value from the MaxMissileNumber Variable so if the value needs changing it can be if needed

ReDim EnemyArray(MaxEnemyNumber) 'This Variable allows me to have the number of enemys on the screen to come and go, for instance if one enemy is defeated it will be removed from the array if an enemy is created the it will be added to the array as long as the MaxEnemyNumber Variable isn't exceeded

ReDim EnemyOnScreen(MaxEnemyNumber) 'This Variable is used to count the number of enemys on the screen and will be used to linit the number that can be on the screen at one instant. In this case it has been given the same value as MaxEnemyNumber so the amount can be changed if needed

MissileSpeed = 5

EnemySpeed = 5

Timer1.Start()

scoreTimer.Start()

enemyTimer.Start()

End If

CreateMissiles(MaxMissileNumber) 'This calls the subroutine with the same name and makes it so only the Max Number of Missiles are made

CreateEnemys(MaxEnemyNumber) 'This calls the subroutine with the same name and makes it so only the Max Number of Enemy are made

Sound.SoundLocation = ("E:\College Subjects\Computer Science\Computer Science NEA\My Games\Space Invader -DataBase\Shoot em Up - Core DONE\Shoot em Up\bin\Debug\Resources\Explosion.wav") 'This Piece of Code loads up the Sound from its location via its name

Randomize() 'Makes it so a different Random Number is generated every single time a random number is made

End Sub

Private Sub Form1\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown 'This Code is ran when a Key is Pressed Down

Dim Count As Integer = 1

Select Case e.KeyValue

Case Keys.Right 'When the Right Key is Pressed Down, the Variable MoveRight becomes True

MoveRight = True

Case Keys.Left 'When the Left Key is Pressed Down, the Variable MoveLeft becomes True

MoveLeft = True

Case Keys.Space 'When the Space Key is Pressed Down, a Missile is released vertically upwards

Sound.SoundLocation = ("E:\College Subjects\Computer Science\Computer Science NEA\My Games\Space Invader -DataBase\Shoot em Up - Core DONE\Shoot em Up\bin\Debug\Resources\Shoot.wav") 'This Piece of Code loads up the Sound from its location via its name

Sound.Play() 'This code plays the sound from the code above

For i = 0 To MaxMissileNumber 'This Loop will count the Number of Missiles on Screen

If MissileOnScreen(i) = True Then 'If there are Missiles on screen then the following code in the IF Statement will be executed

Count = Count + 1 'Everytime the statement is true and space is pressed the Count Variable has 1 added to it. In essence this counts the number of Missiles on the Screen. This value can be used later to limit the amount of missiles on the screen

End If

Next 'The Whole Loop will always count the number of Missiles there are on the screen and limit them to the number assigned

If Count <= MaxMissileNumber Then ' This IF statement does the following: If the Number of Missiles is less than or equal to 5 then can you fire more missiles, if not you cannot fire anymore missiles.

MissileOnScreen(MissileNumber) = True 'This makes the MissileOnScreen Variable True only when count follows is requirement, then the number on missiles which can be on the screen becomes 5

MissileArray(MissileNumber).Visible = True 'This Array stores all the Misiles on the Screen in the Array and makes them Visible so the player can see them

MissileArray(MissileNumber).Location = New Point(pbShip.Location.X + 16, pbShip.Location.Y) 'Makes the Missile Vertically Centered

MissileNumber += 1 'This code increments the MissileNumber by 1

If MissileNumber = MaxMissileNumber Then 'Once the Missile Number reaches 5 from the increments (above) the following code is ran

MissileNumber = 0 'This resets the MissileNumber back to 0, so the player can send out 5 missiles everytime there aren't atleast 5 missiles on the screen, if the MissileNumber wasn't reset then the player would only be able to shoot 5 missiles a game

End If

End If

End Select

End Sub

Private Sub Form1\_KeyUp(sender As Object, e As KeyEventArgs) Handles Me.KeyUp 'This Code is ran when a Key which was Down is released so it is now Up

Select Case e.KeyValue

Case Keys.Right 'When the Right Key is released, the Variable MoveRight becomes False, so the Ship stops moving Right

MoveRight = False

Case Keys.Left 'When the Left Key is release, the Variable MoveLeft Becomes False, so the Ship stops moving Left

MoveLeft = False

End Select

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick 'The Following Code is Excecuted every time the Timer Ticks

If MoveRight = True Then 'If the Boolean Variable MoveRight is True then code inside the IF statement is executed

pbShip.Left += 4 'Everytime the Timer Ticks the Ship moves one to the Right as the distance of the ship from the left increases by one each time

End If

If MoveLeft = True Then 'If the Boolean Variable MoveLeft is True then the code inside the IF statement is executed

pbShip.Left -= 4 'Everytime the Timer Ticks the Ship moves one to the Left as the Distance from the Left decreases by one each time

End If

If pbShip.Bounds.IntersectsWith(pbLeftWall.Bounds) Then 'If the Ship gets to the boundary of the Form the following code is ran

pbShip.Location = New Point(0, pbShip.Location.Y) 'Ship gets moved back to the boundary

End If

If pbShip.Bounds.IntersectsWith(pbRightWall.Bounds) Then 'If the Ship gets to the boundary of the Form then the following code is ran

pbShip.Location = New Point(455, pbShip.Location.Y) 'Ship gets moved back to the boundary

End If

For i = 0 To MaxMissileNumber 'This FOR Loop allows the Missiles made (determined from MissileNumber which is found in Shooting) to move up

If MissileOnScreen(i) = True Then 'If the Missiles on the Screen Then the following code is executed

MissileArray(i).Top -= MissileSpeed 'The Missiles move up

End If

If MissileArray(i).Bounds.IntersectsWith(pbTopBoundary.Bounds) Then 'If the Missiles reach top of the screen then then the following code is executed

MissileOnScreen(i) = False 'The ith missile (object at the ith position) will be False when the Missile reaches the end of the screen as they won't be on the screen anymore

MissileArray(i).Visible = False 'The missile dissapears

End If

For j = 0 To MaxEnemyNumber 'This Loop checks to see if the Missile Hits the Enemy or Not

If MissileArray(i).Bounds.IntersectsWith(EnemyArray(j).Bounds) Then 'If the missile and enemy at the postion i and j respectivily collide the following code is executed

Sound.Play() 'This Piece of Code plays the sound assigned to Sound

MissileArray(i).Visible = False 'The missile dissapears

EnemyArray(j).Top = 0 'This Moves the Enemy back to the Top of the Form if a Missile and Enemy collide

EnemyArray(j).Left = CInt(Rnd() \* Width) 'This moves the enemy to a random location on the X-Axis once it collides with a missile. This makes the Game more replayable so the game doesn't just end when there are no more enemys. The CInt() converts Value inside parenthesis to an integer for instance a Double Value to Integer

End If

Next

Next

End Sub

Private Sub enemyTimer\_Tick(sender As Object, e As EventArgs) Handles enemyTimer.Tick 'This Timer Controls the Enemy's movements

Dim r As Double

For i = 0 To MaxEnemyNumber 'This Loop makes sure all the enemys from 1 to 5 are affected

EnemyArray(i).Top += EnemySpeed 'The enemy moves down at the speed assigned by variable EnemySpeed

EnemyArray(i).Top += EnemySpeed 'This makes the Enemys move Down

If EnemyArray(i).Bounds.IntersectsWith(pbBottomBoundary.Bounds) Then 'The Following IF statement does the following if the Enemy moves past the bottom of the screen

Timer1.Stop() 'The Main Timer Stops

enemyTimer.Stop() 'The Enemy Timer Stops

scoreTimer.Stop() 'The Score Timer Stops so the Player stops gaining any more points to their score

MsgBox("Game Over Your Score is " & Score) 'MessageBox appears which states game over with the players score

Application.Exit()

End If

r = Rnd() 'Makes it so Random is equal to a random number between 0 and 1, as Random is Double it can also be decimals

If r > 0.5 Then 'If r > 0.5 then the following code is executed

EnemyArray(i).Left += 5 'The Enemy Moves Right 5 Pixels each time a random number is generated which is > 0.5

Else 'Otherwise if the r Variable < 0.5 then the following code is executed

EnemyArray(i).Left -= 5 'The Enemy Moves Left 5 Pixels each time a random number is generated which is < 0.5

End If

If EnemyArray(i).Bounds.IntersectsWith(pbLeftWall.Bounds) Then 'If the Enemy moves hits the left wall then the following code moves them back in

EnemyArray(i).Location = New Point(0, EnemyArray(i).Location.Y) 'Enemy is Moved to the Boundary

End If

If EnemyArray(i).Bounds.IntersectsWith(pbRightWall.Bounds) Then 'If the Enemy hiys the right wall then the Following Code moves them back in

EnemyArray(i).Location = New Point(455, EnemyArray(i).Location.Y) 'Enemy is Moved to the Boundary

End If

Next

End Sub

Private Sub scoreTimer\_Tick(sender As Object, e As EventArgs) Handles scoreTimer.Tick 'This Timer is used to keep track of the players score every 1000ms

Score += 1 'The Score Increments by 1 every timer the Timer Ticks

lblScore.Text = ("Score: " & Score) 'The label which displays the Score will show the score using this Code

If Score Mod 5 = 0 Then 'When Score is Divided by 5 and the Remainder is 0 the following code is executed and as only multiples of 5 will have a mod of 0 this occurs everytime the score is a multiple of 5

EnemySpeed += 1.5 'If the IF statement requirement is fufilled the enemy speed increased by 1.5

End If

End Sub

Private Sub CreateMissiles(ByVal Number As Integer) 'This Sub Routine makes the Missiles

For i = 0 To Number

Dim Missile As New PictureBox 'This Variable makes the Picturebox for me so I don't have to make it on the form itself

Me.Controls.Add(Missile) 'The Me refers to the Form (in this case is Form1), so this piece of code adds the PictureBox to Form1 and allows me to control it in the sense that I can manipulate its properties

Missile.Height = 10 'The Missiles Width

Missile.Width = 5 'The Missiles Height

Missile.BorderStyle = BorderStyle.FixedSingle 'The Border of the PictureBox is a single Line which does not move away from the picturebox making sure the whole picturebox is one entity

Missile.BackColor = Color.White 'The Missiles Colour is white

Missile.Top = pbShip.Top + pbShip.Height / 2 - Missile.Height / 2 'Makes the Missile Vertically Centered

Missile.Left = pbShip.Left + pbShip.Width / 2 - Missile.Width / 2 'Makes the Missile Horizontally Centered

Missile.BringToFront() 'Brings the Missile infront of the Ship when the Space Key is Pressed

MissileArray(i) = Missile 'The Number of Missiles in the Array is i, this is done to limit the number of missiles and so the program runs more efficiently rather than having the value increment by using ReDim and Preserve and using MissileArray(MissileNumber) = Missile, in essence this Variable will hold the missiles in the Array, that will be fired off and when they are they are removed from the array

MissileArray(i).Visible = False 'Makes the Missile invisible

MissileOnScreen(i) = False 'This piece of code makes it so the missiles being made aren't considered on the screen until the Form1\_KeyDown event checks to see if there are 5 missiles on the screen or not

Next

End Sub

Private Sub CreateEnemys(ByVal Number As Integer)

For i = 0 To Number

Dim Enemy As New PictureBox 'This Variable makes the Picturebox for me so I don't have to make it on the form itself

Me.Controls.Add(Enemy) 'The Me refers to the Form (in this case is Form1), so this piece of code adds the PictureBox to Form1 and allows me to control it in the sense that I can manipulate its properties

Enemy.Height = 20 'The Enemys Width

Enemy.Width = 20 'The Enemys Height

Enemy.BorderStyle = BorderStyle.FixedSingle 'The Border of the PictureBox is a single Line which does not move away from the picturebox making sure the whole picturebox is one entity

Enemy.BackColor = Color.Red 'The Enemy Colour is Red

Enemy.Top = 50 'Makes it so the enemys are 50 pixels from the Top

Enemy.Left = i \* 80 'Spreads out the Enemys 80 pixels apart

Enemy.BringToFront() 'Brings the Enemy infront of anything that was in front

EnemyArray(i) = Enemy 'The Number of Enemys in the Array is i, this is done to limit the number of enemys and so the program runs more efficiently, in essence this Variable will hold the Enemys in the Array

EnemyOnScreen(i) = True 'This piece of code makes it so the enemys being made aren't considered on the screen until the Form1\_KeyDown event checks to see if there are 5 enemys on the screen or not

Enemy.Visible = True 'Makes the Enemy invisible until called to be visible

Next

End Sub

End Class

Code For PacMan:

***Map Maker***

Option Strict Off

Imports System

Imports System.Drawing

Imports System.Windows.Forms

Imports Microsoft.VisualBasic

Public Class Form1

'View Port

Dim ResWidth As Integer = 1099 'This Value Decides Where the Red Selector Can Go on the X-Axis

Dim ResHeight As Integer = 769.5 'This Value Decides Where the Red Selector Can Go on the Y-Axis

Dim TileSize As Integer = 32

' Graphics Variable

Dim G As Graphics

Dim BBG As Graphics 'BBG = Back Buffer Graphics

Dim BB As Bitmap 'BB = Back Buffer

Dim BMPTile As Bitmap

Dim sRect As Rectangle 'sRect = Source Rectangle

Dim dRect As Rectangle 'dRect = Distance Rectangle

'Map Variable

Dim Map(100, 100, 10) As Integer

Dim MapX As Integer = 20

Dim MapY As Integer = 20

'Game Running

Dim IsRunning As Boolean = True

'Mouse Location

Dim MouseX As Integer

Dim MouseY As Integer

Dim mMapX As Integer

Dim mMapY As Integer

'PaintBrush

Dim PaintBrush As Integer = 0

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.Show()

Me.Focus()

'Initialise Graphics Objects

G = Me.CreateGraphics

BB = New Bitmap(ResWidth, ResHeight)

BMPTile = New Bitmap(GFX.pbGFX.Image)

StartGameLoop()

End Sub

Private Sub StartGameLoop()

Do While IsRunning = True

'Keep App Responsive

Application.DoEvents()

'1) Check User Input

'2) Run AI

'3) Update Object Data (Object Positions, Status, Collision Detection, etc)

'4) Check Triggers and Condition (Death of Pacman Animation, Ghosts Being Eaten, etc)

'5) Draw Graphics

DrawGraphics()

'6) Playing Sound Effects and Music (Not as Important as Everything Else)

Loop

End Sub

Private Sub DrawGraphics()

'Fill Back Buffer

'Draw Tile, String, Characters ETC

For X = 0 To 21 '(22 Tiles Across Screen)

For Y = 0 To 25 '(26 Tiles Down)

GetSourceRectangle(MapX + X, MapY + Y, TileSize, TileSize)

dRect = New Rectangle(X \* TileSize, Y \* TileSize, TileSize, TileSize)

G.DrawImage(BMPTile, dRect, sRect, GraphicsUnit.Pixel)

Next

Next

'Characters, Menus ETC

'G.FillRectangle(Brushes.Red, 22 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Blue, 22 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Black, 22 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Orange, 22 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Pink, 22 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.SandyBrown, 22 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.DarkGreen, 22 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Indigo, 22 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 22 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Navy, 23 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Chocolate, 23 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.DarkSalmon, 23 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(160, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Gray, 23 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.OliveDrab, 23 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Plum, 23 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 23 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Turquoise, 24 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.DarkGray, 24 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Gold, 24 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.LemonChiffon, 24 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Purple, 24 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Tan, 24 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(87, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Yellow, 24 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 25 \* TileSize, 13 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(256, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 25 \* TileSize, 13 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(224, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(190, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(192, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(160, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(128, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 24 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(96, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 24 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(64, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(32, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 26 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(0, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 26 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 13 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 13 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 27 \* TileSize, 15 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 27 \* TileSize, 15 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(288, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(256, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 28 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(224, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 28 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(224, 224, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(224, 256, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(256, 256, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(320, 256, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(352, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 13 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(416, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 13 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 29 \* TileSize, 15 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 29 \* TileSize, 15 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(416, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(416, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(352, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(416, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 30 \* TileSize, 16 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(474, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 30 \* TileSize, 16 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(416, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(480, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(512, 0, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(480, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 32, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 13 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 13 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 31 \* TileSize, 15 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(480, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 31 \* TileSize, 15 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 0 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(512, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 0 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 2 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 2 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 4 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(480, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 4 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 6 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(512, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 6 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 8 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(448, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 8 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 10 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(608, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 10 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 12 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(544, 64, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 12 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 14 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(544, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 14 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 32 \* TileSize, 16 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(608, 96, TileSize, TileSize)

G.DrawImage(BMPTile, 32 \* TileSize, 16 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 1 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(544, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 1 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 3 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(576, 128, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 3 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 5 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(576, 160, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 5 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 7 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(352, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 7 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 9 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(352, 224, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 9 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 11 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(352, 256, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 11 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 13 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 192, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 13 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 33 \* TileSize, 15 \* TileSize, TileSize, TileSize)

sRect = New Rectangle(384, 224, TileSize, TileSize)

G.DrawImage(BMPTile, 33 \* TileSize, 15 \* TileSize, sRect, GraphicsUnit.Pixel)

'G.FillRectangle(Brushes.Colour, 22 \* TileSize, 13 \* TileSize, TileSize, TileSize)

G.DrawRectangle(Pens.Red, MouseX \* TileSize, MouseY \* TileSize, TileSize, TileSize)

'Copy BackBuffer to Graphics Object

G = Graphics.FromImage(BB)

'Draw BackBuffer to Screen

BBG = Me.CreateGraphics

BBG.DrawImage(BB, 0, 0, ResWidth, ResHeight)

End Sub

Private Sub Form1\_MouseClick(sender As Object, e As MouseEventArgs) Handles Me.MouseClick

If MouseX = 22 And MouseY = 2 Then

PaintBrush = 1

ElseIf MouseX = 22 And MouseY = 4 Then

PaintBrush = 2

ElseIf MouseX = 22 And MouseY = 6 Then

PaintBrush = 3

ElseIf MouseX = 22 And MouseY = 8 Then

PaintBrush = 4

ElseIf MouseX = 22 And MouseY = 10 Then

PaintBrush = 5

ElseIf MouseX = 22 And MouseY = 12 Then

PaintBrush = 6

ElseIf MouseX = 22 And MouseY = 14 Then

PaintBrush = 7

ElseIf MouseX = 22 And MouseY = 0 Then

PaintBrush = 8

ElseIf MouseX = 23 And MouseY = 1 Then

PaintBrush = 9

ElseIf MouseX = 23 And MouseY = 3 Then

PaintBrush = 10

ElseIf MouseX = 23 And MouseY = 5 Then

PaintBrush = 11

ElseIf MouseX = 23 And MouseY = 7 Then

PaintBrush = 12

ElseIf MouseX = 23 And MouseY = 9 Then

PaintBrush = 13

ElseIf MouseX = 23 And MouseY = 11 Then

PaintBrush = 14

ElseIf MouseX = 24 And MouseY = 4 Then

PaintBrush = 15

ElseIf MouseX = 24 And MouseY = 6 Then

PaintBrush = 16

ElseIf MouseX = 24 And MouseY = 8 Then

PaintBrush = 17

ElseIf MouseX = 24 And MouseY = 10 Then

PaintBrush = 18

ElseIf MouseX = 24 And MouseY = 12 Then

PaintBrush = 19

ElseIf MouseX = 24 And MouseY = 14 Then

PaintBrush = 20

ElseIf MouseX = 24 And MouseY = 2 Then

PaintBrush = 21

ElseIf MouseX = 25 And MouseY = 1 Then

PaintBrush = 22

ElseIf MouseX = 25 And MouseY = 3 Then

PaintBrush = 23

ElseIf MouseX = 25 And MouseY = 5 Then

PaintBrush = 25

ElseIf MouseX = 25 And MouseY = 7 Then

PaintBrush = 25

ElseIf MouseX = 25 And MouseY = 9 Then

PaintBrush = 26

ElseIf MouseX = 25 And MouseY = 11 Then

PaintBrush = 27

ElseIf MouseX = 25 And MouseY = 13 Then

PaintBrush = 28

ElseIf MouseX = 26 And MouseY = 2 Then

PaintBrush = 29

ElseIf MouseX = 26 And MouseY = 4 Then

PaintBrush = 30

ElseIf MouseX = 26 And MouseY = 6 Then

PaintBrush = 31

ElseIf MouseX = 26 And MouseY = 8 Then

PaintBrush = 32

ElseIf MouseX = 26 And MouseY = 10 Then

PaintBrush = 33

ElseIf MouseX = 24 And MouseY = 0 Then

PaintBrush = 34

ElseIf MouseX = 26 And MouseY = 12 Then

PaintBrush = 35

ElseIf MouseX = 26 And MouseY = 0 Then

PaintBrush = 36

ElseIf MouseX = 26 And MouseY = 14 Then

PaintBrush = 37

ElseIf MouseX = 27 And MouseY = 1 Then

PaintBrush = 38

ElseIf MouseX = 27 And MouseY = 3 Then

PaintBrush = 39

ElseIf MouseX = 27 And MouseY = 5 Then

PaintBrush = 40

ElseIf MouseX = 27 And MouseY = 7 Then

PaintBrush = 41

ElseIf MouseX = 27 And MouseY = 9 Then

PaintBrush = 42

ElseIf MouseX = 27 And MouseY = 11 Then

PaintBrush = 43

ElseIf MouseX = 27 And MouseY = 13 Then

PaintBrush = 44

ElseIf MouseX = 27 And MouseY = 15 Then

PaintBrush = 45

ElseIf MouseX = 28 And MouseY = 0 Then

PaintBrush = 46

ElseIf MouseX = 28 And MouseY = 2 Then

PaintBrush = 47

ElseIf MouseX = 28 And MouseY = 4 Then

PaintBrush = 48

ElseIf MouseX = 28 And MouseY = 6 Then

PaintBrush = 49

ElseIf MouseX = 28 And MouseY = 8 Then

PaintBrush = 50

ElseIf MouseX = 28 And MouseY = 10 Then

PaintBrush = 51

ElseIf MouseX = 28 And MouseY = 12 Then

PaintBrush = 52

ElseIf MouseX = 28 And MouseY = 14 Then

PaintBrush = 53

ElseIf MouseX = 29 And MouseY = 1 Then

PaintBrush = 54

ElseIf MouseX = 29 And MouseY = 3 Then

PaintBrush = 56

ElseIf MouseX = 29 And MouseY = 5 Then

PaintBrush = 55

ElseIf MouseX = 29 And MouseY = 7 Then

PaintBrush = 57

ElseIf MouseX = 29 And MouseY = 9 Then

PaintBrush = 58

ElseIf MouseX = 29 And MouseY = 11 Then

PaintBrush = 59

ElseIf MouseX = 29 And MouseY = 13 Then

PaintBrush = 60

ElseIf MouseX = 29 And MouseY = 15 Then

PaintBrush = 61

ElseIf MouseX = 30 And MouseY = 0 Then

PaintBrush = 62

ElseIf MouseX = 30 And MouseY = 2 Then

PaintBrush = 63

ElseIf MouseX = 30 And MouseY = 4 Then

PaintBrush = 64

ElseIf MouseX = 30 And MouseY = 6 Then

PaintBrush = 65

ElseIf MouseX = 30 And MouseY = 8 Then

PaintBrush = 66

ElseIf MouseX = 30 And MouseY = 10 Then

PaintBrush = 67

ElseIf MouseX = 30 And MouseY = 12 Then

PaintBrush = 68

ElseIf MouseX = 30 And MouseY = 14 Then

PaintBrush = 69

ElseIf MouseX = 30 And MouseY = 16 Then

PaintBrush = 70

ElseIf MouseX = 31 And MouseY = 1 Then

PaintBrush = 71

ElseIf MouseX = 31 And MouseY = 3 Then

PaintBrush = 72

ElseIf MouseX = 31 And MouseY = 5 Then

PaintBrush = 73

ElseIf MouseX = 31 And MouseY = 7 Then

PaintBrush = 74

ElseIf MouseX = 31 And MouseY = 9 Then

PaintBrush = 75

ElseIf MouseX = 31 And MouseY = 11 Then

PaintBrush = 76

ElseIf MouseX = 31 And MouseY = 13 Then

PaintBrush = 77

ElseIf MouseX = 31 And MouseY = 15 Then

PaintBrush = 78

ElseIf MouseX = 32 And MouseY = 0 Then

PaintBrush = 79

ElseIf MouseX = 32 And MouseY = 2 Then

PaintBrush = 80

ElseIf MouseX = 32 And MouseY = 4 Then

PaintBrush = 81

ElseIf MouseX = 32 And MouseY = 6 Then

PaintBrush = 82

ElseIf MouseX = 32 And MouseY = 8 Then

PaintBrush = 83

ElseIf MouseX = 32 And MouseY = 10 Then

PaintBrush = 84

ElseIf MouseX = 32 And MouseY = 12 Then

PaintBrush = 85

ElseIf MouseX = 32 And MouseY = 14 Then

PaintBrush = 86

ElseIf MouseX = 32 And MouseY = 16 Then

PaintBrush = 87

ElseIf MouseX = 33 And MouseY = 1 Then

PaintBrush = 88

ElseIf MouseX = 33 And MouseY = 3 Then

PaintBrush = 89

ElseIf MouseX = 33 And MouseY = 5 Then

PaintBrush = 90

ElseIf MouseX = 33 And MouseY = 7 Then

PaintBrush = 91

ElseIf MouseX = 33 And MouseY = 9 Then

PaintBrush = 92

ElseIf MouseX = 33 And MouseY = 11 Then

PaintBrush = 93

ElseIf MouseX = 33 And MouseY = 13 Then

PaintBrush = 94

ElseIf MouseX = 33 And MouseY = 15 Then

PaintBrush = 95

End If

Select Case PaintBrush

Case 0

Case 1 'Red

Map(mMapX, mMapY, 0) = 1

Case 2 'Blue

Map(mMapX, mMapY, 0) = 2

Case 3 'Black

Map(mMapX, mMapY, 0) = 3

Case 4 'Orange

Map(mMapX, mMapY, 0) = 4

Case 5 'Pink

Map(mMapX, mMapY, 0) = 5

Case 6 'Sandy Brown

Map(mMapX, mMapY, 0) = 6

Case 7 'Dark Green

Map(mMapX, mMapY, 0) = 7

Case 8 'Indigo

Map(mMapX, mMapY, 0) = 8

Case 9 'Navy

Map(mMapX, mMapY, 0) = 9

Case 10 'Chocolate

Map(mMapX, mMapY, 0) = 10

Case 11 'Dark Salmon

Map(mMapX, mMapY, 0) = 11

Case 12 'Gray

Map(mMapX, mMapY, 0) = 12

Case 13 'OliveDrab

Map(mMapX, mMapY, 0) = 13

Case 14 'Silver

Map(mMapX, mMapY, 0) = 14

Case 15 'Turquoise

Map(mMapX, mMapY, 0) = 15

Case 16 'Dark Gray

Map(mMapX, mMapY, 0) = 16

Case 17 'Gold

Map(mMapX, mMapY, 0) = 17

Case 18 'Lemon Chiffon

Map(mMapX, mMapY, 0) = 18

Case 19 'Purple

Map(mMapX, mMapY, 0) = 19

Case 20 'Tan

Map(mMapX, mMapY, 0) = 20

Case 21 'Yellow

Map(mMapX, mMapY, 0) = 21

Case 22 'Colour

Map(mMapX, mMapY, 0) = 22

Case 23 'Colour

Map(mMapX, mMapY, 0) = 23

Case 24 'Colour

Map(mMapX, mMapY, 0) = 24

Case 25 'Colour

Map(mMapX, mMapY, 0) = 25

Case 26 ' Colour

Map(mMapX, mMapY, 0) = 26

Case 27 'Colour

Map(mMapX, mMapY, 0) = 27

Case 28 'Colour

Map(mMapX, mMapY, 0) = 28

Case 29 'Colour

Map(mMapX, mMapY, 0) = 29

Case 30 'Colour

Map(mMapX, mMapY, 0) = 30

Case 31 'Colour

Map(mMapX, mMapY, 0) = 31

Case 32 'Colour

Map(mMapX, mMapY, 0) = 32

Case 33 'Colour

Map(mMapX, mMapY, 0) = 33

Case 34 'Colour

Map(mMapX, mMapY, 0) = 34

Case 35 'Colour

Map(mMapX, mMapY, 0) = 35

Case 36 'Colour

Map(mMapX, mMapY, 0) = 36

Case 37 'Colour

Map(mMapX, mMapY, 0) = 37

Case 38 'Colour

Map(mMapX, mMapY, 0) = 38

Case 39 'Colour

Map(mMapX, mMapY, 0) = 39

Case 40 'Colour

Map(mMapX, mMapY, 0) = 40

Case 41 'Colour

Map(mMapX, mMapY, 0) = 41

Case 42 'Colour

Map(mMapX, mMapY, 0) = 42

Case 43 'Colour

Map(mMapX, mMapY, 0) = 43

Case 44 'Colour

Map(mMapX, mMapY, 0) = 44

Case 45 'Colour

Map(mMapX, mMapY, 0) = 45

Case 46 'Colour

Map(mMapX, mMapY, 0) = 46

Case 47 'Colour

Map(mMapX, mMapY, 0) = 47

Case 48 'Colour

Map(mMapX, mMapY, 0) = 48

Case 49 ' Colour

Map(mMapX, mMapY, 0) = 49

Case 50 ' Colour

Map(mMapX, mMapY, 0) = 50

Case 51 ' Colour

Map(mMapX, mMapY, 0) = 51

Case 52 ' Colour

Map(mMapX, mMapY, 0) = 52

Case 53 ' Colour

Map(mMapX, mMapY, 0) = 53

Case 54 'Colour

Map(mMapX, mMapY, 0) = 54

Case 55 'Colour

Map(mMapX, mMapY, 0) = 55

Case 56 'Colour

Map(mMapX, mMapY, 0) = 56

Case 57 'Colour

Map(mMapX, mMapY, 0) = 57

Case 58 'Colour

Map(mMapX, mMapY, 0) = 58

Case 59 'Colour

Map(mMapX, mMapY, 0) = 59

Case 60 'Colour

Map(mMapX, mMapY, 0) = 60

Case 61 'Colour

Map(mMapX, mMapY, 0) = 61

Case 62 'Colour

Map(mMapX, mMapY, 0) = 62

Case 63 'Colour

Map(mMapX, mMapY, 0) = 63

Case 64 'Colour

Map(mMapX, mMapY, 0) = 64

Case 65 'Colour

Map(mMapX, mMapY, 0) = 65

Case 66 'Colour

Map(mMapX, mMapY, 0) = 66

Case 67 'Colour

Map(mMapX, mMapY, 0) = 67

Case 68 'Colour

Map(mMapX, mMapY, 0) = 68

Case 69 'Colour

Map(mMapX, mMapY, 0) = 69

Case 70 'Colour

Map(mMapX, mMapY, 0) = 70

Case 71 'Colour

Map(mMapX, mMapY, 0) = 71

Case 72 'Colour

Map(mMapX, mMapY, 0) = 72

Case 73 'Colour

Map(mMapX, mMapY, 0) = 73

Case 74 'Colour

Map(mMapX, mMapY, 0) = 74

Case 75 'Colour

Map(mMapX, mMapY, 0) = 75

Case 76 'Colour

Map(mMapX, mMapY, 0) = 76

Case 77 'Colour

Map(mMapX, mMapY, 0) = 77

Case 78 'Colour

Map(mMapX, mMapY, 0) = 78

Case 79 'Colour

Map(mMapX, mMapY, 0) = 79

Case 80 'Colour

Map(mMapX, mMapY, 0) = 80

Case 81 'Colour

Map(mMapX, mMapY, 0) = 81

Case 82 'Colour

Map(mMapX, mMapY, 0) = 82

Case 83 'Colour

Map(mMapX, mMapY, 0) = 83

Case 84 'Colour

Map(mMapX, mMapY, 0) = 84

Case 85 'Colour

Map(mMapX, mMapY, 0) = 85

Case 86 'Colour

Map(mMapX, mMapY, 0) = 86

Case 87 'Colour

Map(mMapX, mMapY, 0) = 87

Case 88 'Colour

Map(mMapX, mMapY, 0) = 88

Case 89 'Colour

Map(mMapX, mMapY, 0) = 89

Case 90 'Colour

Map(mMapX, mMapY, 0) = 90

Case 91 'Colour

Map(mMapX, mMapY, 0) = 91

Case 92 'Colour

Map(mMapX, mMapY, 0) = 92

Case 93 'Colour

Map(mMapX, mMapY, 0) = 93

Case 94 'Colour

Map(mMapX, mMapY, 0) = 94

Case 95 'Colour

Map(mMapX, mMapY, 0) = 95

End Select

End Sub

Private Sub Form1\_MouseMove(sender As Object, e As MouseEventArgs) Handles Me.MouseMove

MouseX = Math.Floor(e.X / TileSize)

MouseY = Math.Floor(e.Y / TileSize)

mMapX = MapX + MouseX

mMapY = MapY + MouseY

End Sub

Private Sub GetSourceRectangle(ByVal X As Integer, ByVal Y As Integer, ByVal w As Integer, ByVal h As Integer)

Select Case Map(X, Y, 0)

Case 0 'Black Tile' changes all of them

sRect = New Rectangle(87, 0, TileSize, TileSize)

Case 1 'Box Bottom Straight 'Red

sRect = New Rectangle(0, 0, TileSize, TileSize)

Case 2 'Box Top Right Curve 'Blue

sRect = New Rectangle(32, 0, TileSize, TileSize)

Case 3 'Box Bottom Right Curve 'Black

sRect = New Rectangle(64, 0, TileSize, TileSize)

Case 4 'Right Straight 'Orange

sRect = New Rectangle(96, 0, TileSize, TileSize)

Case 5 'Bottom Straight 'Pink

sRect = New Rectangle(128, 0, TileSize, TileSize)

Case 6 'Box Left Curve 'Sandy Brown

sRect = New Rectangle(0, 32, TileSize, TileSize)

Case 7 'Box Bottom Straight 'Dark Green

sRect = New Rectangle(32, 32, TileSize, TileSize)

Case 8 'Box Bottom Left Curve 'Indigo

sRect = New Rectangle(64, 32, TileSize, TileSize)

Case 9 'Right Curve 'Navy

sRect = New Rectangle(96, 32, TileSize, TileSize)

Case 10 'Left Curve 'Chocolate

sRect = New Rectangle(128, 32, TileSize, TileSize)

Case 11 'Left Straight 'Dark Salmon

sRect = New Rectangle(160, 32, TileSize, TileSize)

Case 12 'Top Right Curve 'Gray

sRect = New Rectangle(0, 64, TileSize, TileSize)

Case 13 'Top Straight 'OliverDrab

sRect = New Rectangle(32, 64, TileSize, TileSize)

Case 14 'Top Left Curve 'Silver

sRect = New Rectangle(64, 64, TileSize, TileSize)

Case 15 'Mid Left Curve 'Turquoise

sRect = New Rectangle(96, 64, TileSize, TileSize)

Case 16 'Mid Joint 'Dark Grey

sRect = New Rectangle(128, 64, TileSize, TileSize)

Case 17 'Mid Right Curve 'Gold

sRect = New Rectangle(96, 128, TileSize, TileSize)

Case 18 'Mid Left Straight 'Lemon Chiffon

sRect = New Rectangle(96, 96, TileSize, TileSize)

Case 19 'Mid Right Straight 'Purple

sRect = New Rectangle(128, 96, TileSize, TileSize)

Case 20 'Black Tile 'Tan

sRect = New Rectangle(87, 0, TileSize, TileSize)

Case 21 ' Mid Bottom Right Curve 'Yellow

sRect = New Rectangle(128, 128, TileSize, TileSize)

Case 22 ' Left Curve Right Curve Joint 'Colour

sRect = New Rectangle(0, 96, TileSize, TileSize)

Case 23 'Left Curve Bottom Straight 'Colour

sRect = New Rectangle(0, 128, TileSize, TileSize)

Case 24 'Left Curve Top Straight 'Colour

sRect = New Rectangle(32, 96, TileSize, TileSize)

Case 25 'Left Curve Bottom Right Curve 'Colour

sRect = New Rectangle(32, 128, TileSize, TileSize)

Case 26 'Left Curve Top Right Curve 'Colour

sRect = New Rectangle(64, 96, TileSize, TileSize)

Case 27 'Left Curve Right Straight 'Colour

sRect = New Rectangle(64, 128, TileSize, TileSize)

Case 28 'Right Curve Joint 'Colour

sRect = New Rectangle(256, 0, TileSize, TileSize)

Case 29 'Right Curve Straight 'Colour

sRect = New Rectangle(224, 0, TileSize, TileSize)

Case 30 'Right Curve Left Joint 'Colour

sRect = New Rectangle(193, 0, TileSize, TileSize)

Case 31 'Right Curve Left Straight 'Colour

sRect = New Rectangle(192, 160, TileSize, TileSize)

Case 32 'Right Curve Left Right Joint 'Colour

sRect = New Rectangle(160, 160, TileSize, TileSize)

Case 33 'Right Bottom Straight 'Colour

sRect = New Rectangle(128, 160, TileSize, TileSize)

Case 34 'Left Curve Straight Joint 'Colour

sRect = New Rectangle(96, 160, TileSize, TileSize)

Case 35 'Right Curve Top Bottom 'Colour

sRect = New Rectangle(64, 160, TileSize, TileSize)

Case 36 'Right Curve Botton Straight 'Colour

sRect = New Rectangle(32, 160, TileSize, TileSize)

Case 37 'Right Curve Bottom Right Joint 'Colour

sRect = New Rectangle(0, 160, TileSize, TileSize)

Case 38 'Left Bump Top 'Colour

sRect = New Rectangle(288, 0, TileSize, TileSize)

Case 39 'Left Bump Bottom 'Colour

sRect = New Rectangle(288, 32, TileSize, TileSize)

Case 40 'Left Bump Top Edge 'Colour

sRect = New Rectangle(320, 0, TileSize, TileSize)

Case 41 'Left Bump Bottom Edge 'Colour

sRect = New Rectangle(320, 32, TileSize, TileSize)

Case 42 'Right Bump Top 'Colour

sRect = New Rectangle(320, 64, TileSize, TileSize)

Case 43 'Right Bump Bottom 'Colour

sRect = New Rectangle(320, 96, TileSize, TileSize)

Case 44 'Right Bump Top Edge 'Colour

sRect = New Rectangle(288, 64, TileSize, TileSize)

Case 45 'Right Bump Bottom Edge 'Colour

sRect = New Rectangle(288, 96, TileSize, TileSize)

Case 46 'Straight Curve Right Box 'Colour

sRect = New Rectangle(320, 128, TileSize, TileSize)

Case 47 'Straight Curve Left Box 'Colour

sRect = New Rectangle(288, 128, TileSize, TileSize)

Case 48 'Small Straight Curve Right Box 'Colour

sRect = New Rectangle(320, 160, TileSize, TileSize)

Case 49 'Small Straight Curve Left Box 'Colour

sRect = New Rectangle(288, 160, TileSize, TileSize)

Case 50 'Centre Box Right 'Colour

sRect = New Rectangle(288, 192, TileSize, TileSize)

Case 51 'Right Box Curve 'Colour

sRect = New Rectangle(320, 192, TileSize, TileSize)

Case 52 'Centre Box Left 'Colour

sRect = New Rectangle(256, 192, TileSize, TileSize)

Case 53 'Left Box Curve 'Colour

sRect = New Rectangle(224, 192, TileSize, TileSize)

Case 54 'Left Box Centre Straight 'Colour

sRect = New Rectangle(224, 224, TileSize, TileSize)

Case 55 'Centre Bottom Straight Centre 'Colour

sRect = New Rectangle(256, 256, TileSize, TileSize)

Case 56 'Centre Bottom Left Curve 'Colour

sRect = New Rectangle(224, 256, TileSize, TileSize)

Case 57 'Centre Bottom Right Curve 'Colour

sRect = New Rectangle(320, 256, TileSize, TileSize)

Case 58 'T-Shape Top Left Curve 'Colour

sRect = New Rectangle(352, 0, TileSize, TileSize)

Case 59 'T-Shape Top Centre Left Curve 'Colour

sRect = New Rectangle(384, 0, TileSize, TileSize)

Case 60 'T-Shape Top Centre Right Curve 'Colour

sRect = New Rectangle(416, 0, TileSize, TileSize)

Case 61 'T-Shape Top Right Curve 'Colour

sRect = New Rectangle(448, 0, TileSize, TileSize)

Case 62 'T-Shape Left Centre Straight Curve 'Colour

sRect = New Rectangle(384, 32, TileSize, TileSize)

Case 63 'T-Shape Left Centre End Curve 'Colour

sRect = New Rectangle(384, 64, TileSize, TileSize)

Case 64 'T-Shape Right Centre Straight Curve 'Colour

sRect = New Rectangle(416, 32, TileSize, TileSize)

Case 65 'T-Shape Right Centre End Curve 'Colour

sRect = New Rectangle(416, 64, TileSize, TileSize)

Case 66 'Upside Down T-Shape Left Curve 'Colour

sRect = New Rectangle(352, 160, TileSize, TileSize)

Case 67 'Upside Down T-Shape Centre Left Curve 'Colour

sRect = New Rectangle(384, 160, TileSize, TileSize)

Case 68 'Upside Down T-Shape Centre Right Curve 'Colour

sRect = New Rectangle(416, 160, TileSize, TileSize)

Case 69 'Upside Down T-Shape Centre Right Straight 'Colour

sRect = New Rectangle(448, 160, TileSize, TileSize)

Case 70 'Upside Down T-Shape Centre Right Curve 'Colour

sRect = New Rectangle(474, 160, TileSize, TileSize)

Case 71 'Upside Down T-Shape Top Left Curve 'Colour

sRect = New Rectangle(384, 96, TileSize, TileSize)

Case 72 'Upside Down T-Shape Top Right Curve 'Colour

sRect = New Rectangle(416, 96, TileSize, TileSize)

Case 73 'Upside Down L Left Curve 'Colour

sRect = New Rectangle(480, 0, TileSize, TileSize)

Case 74 'Upside Down L Right Curve 'Colour

sRect = New Rectangle(512, 0, TileSize, TileSize)

Case 75 'Upside Down L Bottom Curve 'Colour

sRect = New Rectangle(480, 32, TileSize, TileSize)

Case 76 'Left T-Shape Top Curve 'Colour

sRect = New Rectangle(448, 32, TileSize, TileSize)

Case 77 'Left T-Shape Top Centre Curve 'Colour

sRect = New Rectangle(448, 64, TileSize, TileSize)

Case 78 'Left T-Shape Top Centre Straight 'Colour

sRect = New Rectangle(480, 64, TileSize, TileSize)

Case 79 'Left T-Shape Top Centre Curve 'Colour

sRect = New Rectangle(512, 64, TileSize, TileSize)

Case 80 'Left T-Shape Bottom Curve 'Colour

sRect = New Rectangle(448, 96, TileSize, TileSize)

Case 81 'Left T-Shape Bottom Centre Straight 'Colour

sRect = New Rectangle(480, 96, TileSize, TileSize)

Case 82 'Left T-Shape Bottom Centre Curve 'Colour

sRect = New Rectangle(512, 96, TileSize, TileSize)

Case 83 'Left T-Shape Bottom Centre 'Colour

sRect = New Rectangle(448, 128, TileSize, TileSize)

Case 84 'Right T-Shape Top Centre 'Colour

sRect = New Rectangle(608, 64, TileSize, TileSize)

Case 85 'Right T-Shape Top Centre Curve 'Colour

sRect = New Rectangle(544, 64, TileSize, TileSize)

Case 86 'Right T-Shape Bottom Centre 'Colour

sRect = New Rectangle(544, 96, TileSize, TileSize)

Case 87 'Right T-Shape Bottom Centre 'Colour

sRect = New Rectangle(608, 96, TileSize, TileSize)

Case 88 'Left Upside Down L Curve 'Colour

sRect = New Rectangle(544, 128, TileSize, TileSize)

Case 89 'Left Upside Down L Edge 'Colour

sRect = New Rectangle(576, 128, TileSize, TileSize)

Case 90 'Left Upside Down L Bottom 'Colour

sRect = New Rectangle(576, 160, TileSize, TileSize)

Case 91 'Top Curve 'Colour

sRect = New Rectangle(352, 192, TileSize, TileSize)

Case 92 'Bottom Curve 'Colour

sRect = New Rectangle(352, 224, TileSize, TileSize)

Case 93 'Bottom Curve 'Colour

sRect = New Rectangle(352, 256, TileSize, TileSize)

Case 94 'Pill 'Colour

sRect = New Rectangle(384, 192, TileSize, TileSize)

Case 95 'Power Pill 'Colour

sRect = New Rectangle(384, 224, TileSize, TileSize)

End Select

End Sub

Private Sub SaveMap(ByVal MapFile As String)

Try

Dim sw As New IO.StreamWriter(MapFile & ".map")

Dim strLine As String = ""

Dim X As Integer = 0

Dim Y As Integer = 0

For Y = 0 To 100

For X = 0 To 100

strLine = strLine & Map(X, Y, 0) & ","

Next

sw.WriteLine(strLine)

strLine = ""

Next

sw.Close()

sw.Dispose()

MsgBox("Map '" & MapFile & "' Successfully Saved.", MsgBoxStyle.OkOnly, "Success")

Catch ex As Exception

MsgBox("Map '" & MapFile & "' Could not be Written to." & vbCrLf & vbCrLf & ex.Message, MsgBoxStyle.Critical, "ERROR")

IsRunning = False

End Try

End Sub

Private Sub LoadMap(ByVal MapFile As String)

Try

Dim sr As New IO.StreamReader(MapFile & ".map")

Dim strLine As String = ""

Dim X As Integer = 0

Dim Y As Integer = 0

Do Until sr.EndOfStream

strLine = sr.ReadLine

strLine = strLine.Replace(strLine.LastIndexOf(","), "")

For Each item As String In Split(strLine, ",", -1)

If item = "" Then

item = 0

End If

If X <= 100 Then

Map(X, Y, 0) = Int(item)

End If

X = X + 1

Next

X = 0

Y = Y + 1

Loop

sr.Close()

sr.Dispose()

Catch ex As Exception

MsgBox("Map '" & MapFile & "' Could not be Loaded." & vbCrLf & vbCrLf & ex.Message, MsgBoxStyle.Critical, "ERROR")

IsRunning = False

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

SaveMap("Map1")

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

LoadMap("Map1")

End Sub

End Class

***Menu:***

Public Class Form1

Private Sub butStart\_Click(sender As Object, e As EventArgs) Handles butStart.Click

Me.Close() 'Closes the Form

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Unity PacMan Done\PacMan Clone.exe") 'This path leads to the Original game which the player can access and play

End Sub

Private Sub butQuit\_Click(sender As Object, e As EventArgs) Handles butQuit.Click

Application.Exit() 'Closes the whole program

End Sub

Private Sub butChooseMap\_Click(sender As Object, e As EventArgs) Handles butChooseMap.Click

Me.Hide() 'Closes this Form

Form3.Show() 'Shows Form3 to the User

End Sub

Private Sub butHighScore\_Click(sender As Object, e As EventArgs) Handles butHighScore.Click

Me.Hide() 'Closes the current Form

Form2.Show() 'Displays the Form2

End Sub

End Class

***Form3 For Menu (Menu for Map Selection):***

Public Class Form3

Private Sub butMap2\_Click(sender As Object, e As EventArgs) Handles butMap2.Click

Me.Close() 'Closes the Form

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Unity PacMan Done\Map 2\PacMan Clone.exe") 'This path leads to the game with this Map Starting

End Sub

Private Sub butMap3\_Click(sender As Object, e As EventArgs) Handles butMap3.Click

Me.Close() 'Closes the Form

Process.Start("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Unity PacMan Done\Map 3\PacMan Clone.exe") 'This path leads to the game with this Map Starting

End Sub

End Class

***Storing Scores:***

Imports System.IO

Public Class Form2

#Region "Variables"

Dim NumOfValues As Integer = 0 'Used to Count the Size of JustScore Array

'Variable Below Stores the Usernames and Scores from the PacMan Game

Dim PacManTextFile() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\My Games\PacMan Unity\Help\PacMan Clone\PacMan Clone\Assets\Text File\Username.txt")

Dim Username() As String 'Stores all the Usernames in this array

Dim temporaryCheck() As String 'Array is used to check for duplicate data

Dim JustScore(NumOfValues) As Integer 'Array will only store the Scores

Dim JustUserName() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\My Games\PacMan Menu\PacMan Menu\bin\Debug\JustUserName.txt") 'Array Will only store the Usernames

Dim JustUserNameVal As String 'Holds the Value of the Username when it is seperated from the score

Dim CheckExists() As String 'Used alongside temporaryCheck to check for duplicate data

Dim UserInput As String 'Will hold all UserInput

Dim ScoreOfUser As Integer 'Will Hold the score of a single User

Dim temp As String 'Temp variable used to store data which needs holding on to for a while

Dim GlobalNum As Integer 'Used to turn a local value into a Global Value

Dim count As Integer = 0 'Used with CheckExists and TemporaryCheck to find Duplicated Dataw

#End Region

Private Sub Form2\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'Stores the Username and Score from the PacMan file into temp when the Form Loads

For i = 0 To PacManTextFile.Length - 1

temp = PacManTextFile(i)

Next

End Sub

Private Sub butQuit\_Click(sender As Object, e As EventArgs) Handles butQuit.Click

Application.Exit() 'Closes the Application

End Sub

Private Sub butStart\_Click(sender As Object, e As EventArgs) Handles butStart.Click

'Code Below Checks if temp is empty, if it is then we write to username for the first time otherwise data already exists and we make temp empty and write the new value to temp

If File.ReadAllText("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\temp.txt").Length = 0 Then

MsgBox("Storing First Data")

'Writes temp to Username File

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt", True)

Writer.WriteLine(temp)

End Using

GettingJustScore() 'Calls Sub Routine by the same name here

Else 'If data is present in the temp file then

MsgBox("Existing Data...")

System.IO.File.WriteAllText("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\temp.txt", "") 'Makes temp empty

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\temp.txt", True) 'Writes temp value to temp

Writer.WriteLine(temp)

End Using

End If

temporaryCheck = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\temp.txt") 'Reads temp

CheckExists = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt") 'Reads username

'Checks if the Score already exists in the database

For i = 0 To temporaryCheck.Length - 1

For j = 0 To CheckExists.Length - 1

If CheckExists(j) = temporaryCheck(i) Then 'If the Data already exists count is set to 0

count = 0

ElseIf CheckExists(j) <> temporaryCheck(i) Then 'If the Data doesn't exist count is incremented by one

count += 1

End If

Next

If count > 0 Then 'If the Value of Count is greater then its base value then data is writen

'Writes temp to Username

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt", True)

Writer.WriteLine(temp)

End Using

'Reads the Username Text File to Username

Username = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt")

End If

GettingJustScore() 'Calls Sub Routine by the same name here

Next

Dim CheckJustUserName() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\JustUserName.txt") 'Reads username

Dim count2 As Integer = 0 'Used to remove duplicate data

Dim Score() As String = File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\score.txt") 'Reads score to Score array

Username = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt") 'Reads all the Usernames with scores to Username

If CheckJustUserName.Length < 1 Then 'If the JustUsername array is empty so checks if file is empty if it is then the following code is ran

'Nested FOR Loops Used to go through Username and Score Array

For i = 0 To Username.Length - 1

For j = 0 To Score.Length - 1

'Code Below Seperates the Username and Score by using the comma as a delimiter

For Each Data As String In Username

If Data.EndsWith(", " & Score(j)) Then

If Not Integer.Parse(Data.Split(","c)(1)) Then 'If the value cannot be turned into a integer then the following code is ran

JustUserNameVal = Data.Split(","c)(0) 'Stores the just the UserName in JustUserNameVal

'Writes just the Username to the JustUserName text file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(JustUserNameVal)

End Using

End If

End If

Next

Next

Next

Else 'If there already is data in JustUserName Text File then the following code is ran

'Loops through JustUserName array and Score Array

'The Nested FOR Loops Seperates the Username from score using Split

For i = 0 To JustUserName.Length - 1

For j = 0 To Score.Length - 1

For Each Data As String In Username

If Data.EndsWith(", " & Score(j)) Then

If Not Integer.Parse(Data.Split(","c)(1)) Then

JustUserNameVal = Data.Split(","c)(0)

'If the Username already is in file then count is set to base value of 0

If CheckJustUserName(i) = JustUserNameVal Then

count2 = 0

'If the Username doesn't exist in file then count is incremented by 1

ElseIf CheckJustUserName(i) <> JustUserNameVal Then

count2 += 1

End If

End If

End If

Next

Next

Next

'If Count2 is greater then base value (0) then data did not exist before so writes data to file

If count2 > 0 Then

'Writes just the Username to JustUserName text file

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\JustUserName.txt", True)

Writer.WriteLine(JustUserNameVal)

End Using

JustUserName = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt") 'Stores all the Usernames into JustUserName Array

End If

End If

UserInput = InputBox("What is Your Username?") 'Asks the User for their username

Username = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt") 'Reads alls the scores and usernames to username array

GetValue() 'Calls Sub Routine by the same name here

MessageBox.Show("Your Score is " & ScoreOfUser) 'MessageBox appears which tells the user their score

butMergeSort.Visible = True 'Makes the Merge Sort Button Visible to the User

End Sub

Private Sub butMergeSort\_Click(sender As Object, e As EventArgs) Handles butMergeSort.Click

MergeSort(JustScore) 'Calls the Merge Sort Algorithm passing the Just Score Array in the parameters

MessageBox.Show("Data is Sorted, You can Now use the Searching Operation") 'When data is sorted a Message box appears which tells the User that the data is sorted

butSearch.Visible = True 'Makes the Search Button Visible allowing the user to search for their position

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Calls Binary Search Algorithm here

End Sub

Private Sub GettingJustScore()

'Variables below are used to just get the score of the user from the username file

Dim data = System.IO.File.ReadAllText("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt")

Dim arrays = New List(Of Single())()

Dim lines = data.Split({vbCr, vbLf}, StringSplitOptions.RemoveEmptyEntries)

Dim Number As Integer

For Each line In lines

Dim lineArray = New List(Of Single)()

For Each s In line.Split({","c}, StringSplitOptions.RemoveEmptyEntries) 'Seperates the data at the comma

Integer.TryParse(s, Number) 'Trys to convert s to integer if the data can be parsed it is added to linearray

lineArray.Add(Convert.ToInt32(Number))

Next

arrays.Add(lineArray.ToArray()) 'Turns LineArray data to lineArray

Dim i As Integer = 0, result As Integer = 0 'Declare two more integer variables called i and integer

'Increments result

While i < lineArray.Count

result += Convert.ToInt32(lineArray(Math.Min(System.Threading.Interlocked.Increment(i), i - 1)))

End While

temporaryCheck = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\score.txt") 'Used to remove duplicate data, stores all scores

If temporaryCheck.Length < 1 Then 'If the Score file is empty data is written to it

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\score.txt", True)

Writer.WriteLine(result)

End Using

Else 'If the file isn't empty then the following code is ran

If temporaryCheck.Contains(result) Then 'If the array already holds that value then nothing happens

'Do Nothing

Else 'If the value isn't already in the array then the following code is ran

'Writes the data to Score

Using Writer As New IO.StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\score.txt", True)

Writer.WriteLine(result)

End Using

GlobalNum = result 'Gives GlobalNum the same value as result

End If

End If

Next

UpdateData() 'Calls Sub Routine by the same name here

End Sub

Private Sub UpdateData()

Dim ScoreArray() As String = IO.File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\score.txt") 'Reads and Stores all the Values from Score

'Loops through ScoreArray

For i = 0 To ScoreArray.Length - 1

If JustScore.Contains(ScoreArray(i)) Then 'If JustScore already contains the value then nothing happens

'Do Nothing

Else 'If JustScore doesn't already contain the data then the following code is ran

ScoreArray(i) = CInt(ScoreArray(i)) 'Converts the value at i to integer

JustScore(NumOfValues) = ScoreArray(i) 'Gives JustScore the same value

NumOfValues += 1 'Increments i by 1

ReDim Preserve JustScore(NumOfValues) 'Increases the Size of JustScore by ReDim and keeps the old values through Preserve

End If

Next

End Sub

Private Function GetValue()

'Goes through the Username Array and looks for Data that starts with the username the user stated

For Each Data As String In Username

If Data.StartsWith(UserInput & ",") Then

ScoreOfUser = Integer.Parse(Data.Split(","c)(1)) 'Seperates the score from the username and stores it in ScoreOfUser

End If

Next

Return ScoreOfUser 'Outputs ScoreOfUser

End Function

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length - 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Sub BinarySearch()

'Binary Search Algorithm can be seen Below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score?")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result 'Inverses the Positon so greatest value is at the top of the array instead of being the last value

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

End Class

***PacMan Game Scripts:***

***Tile:***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Tile : MonoBehaviour {

public bool isPortal; //Used to see if a tile is a portal or not

public bool isPellet; //Used to check and differentiate between a normal pellet and super pellet, if the pellet is a normal pellet this variable will be true for that only

public bool isSuperPellet; //Used to check if the pellet is a normal pellet or super pellet, if the pellet is a super pellet this variable is true for that only

public bool didConsume; //Used to check if the pellet has been consumed or not so if pacman has collied with it or not

public bool isGhostHouseEntrance; //Stores the two node in front of the ghost house

public bool isGhostHouse; //Stores the Nodes inside the Ghost House

public GameObject portalReceiver; //Used with the isPortal to allow you to teleport to the other portal

}

***GameBoard:***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System.IO;

using UnityEditor;

using UnityEngine.SceneManagement;

public class GameBoard : MonoBehaviour

{

public static int pacManLives = 3; //Keeps track of PacMans Lives

public static int playerLevel = 1; //Keeps track of what level the player is on

private static int boardWidth = 28; //This varaible is an integer which won't change its value thus is static and not dynamic, the varaible sets the games width which is the same as the width of the original PacMan game

private static int boardHeight = 36; //This varaible is an integer which won't change its value thus is static and not dynamic, the varaible sets the games height which is the same as the height of the original PacMan game

public int totalPellets = 0; //This will be used to count how many pellets there are in the game

public static int Score = 0; //This will store the players score

public static int ghostConsumedRunningScore; //Variable will store the score gained from consuming a ghost

public Text playerOneUp; //Will indicate where the player score is shown

public Text playerOneScoreText; //Will display the players score

//Holds picture of PacMan which represents the amount of lives he has left

public Image Lives;

public Image Lives2;

public Image Lives3;

public GameObject inputField; //Stores the InputBox where the player will write their high score, used to hide it

public GameObject hideButton; //Stores the button and is used to hide it

public InputField textField; //Stores the text field where the player will write their username if they want to store it

public Button submitButton; //Stores the button which when pressed will save the players data

public GameObject[,] board = new GameObject[boardWidth, boardHeight]; //This multidimenstional array will be used to store all the game objects which are inside the board (thus the width and height is referenced)

// Use this for initialization

void Start()

{

inputField.SetActive(false); //Makes the Input Field Invisible

hideButton.SetActive(false); //Makes the Button Invisible

Object[] objects = GameObject.FindObjectsOfType(typeof(GameObject)); //Makes a local variable called objects which will detect all of the GameObjects in the scene at the start of the Array and will be used with the FOR loop below to iterate through them and store them in the array

foreach (GameObject o in objects) //Iterates through all of the objects and adds them to the Array which will contain all the objects in the scene

{

Vector2 pos = o.transform.position; //Stores the position in Vector 2

if (o.name != "PacMan" && o.name != "Nodes" && o.name != "NonNodes" && o.name != "Maze" && o.name != "Pellets" && o.tag != "Ghost" && o.tag != "ghostHome" && o.name != "Canvas" && o.tag != "UIElements" && o.tag != "Berry") //If the tag/name of the object being added to the array is any of the name specified within the parameter it won't be added to the array

{

if (o.GetComponent<Tile>() != null) //If the object is a Tile then the following code is ran

{

if (o.GetComponent<Tile>().isPellet || o.GetComponent<Tile>().isSuperPellet) //If the object is a pellet or a superpellet then the following code is ran

{

totalPellets++; //Increments the value of total pellets by 1, so counts how many pellets there are initially

}

}

board[(int)pos.x, (int)pos.y] = o; //The x and y location is stored in the array for every object not called Pacman as o

}

else

{ //If the condtion in the IF statement isn't fuffiled then this ELSE statement is executed

}

}

}

// Update is called once per frame

void Update()

{

UpdateUI(); //Calls the Sub Routine by the same name here

CheckPelletsConsumed();

}

void UpdateUI() //Used to Update all objects on the UI

{

playerOneScoreText.text = Score.ToString(); //Updates the Players Score text box displaying their score so they can see it

if (pacManLives == 2) //If PacMan has two lives left then the following code is ran

{

Lives.enabled = false; //One life image is diabled to inform player how many lives they have left

Berries.count = 1; //Variable used to count is reset

}

else if (pacManLives == 1) //If PacMan has only one life left then the following code is executed

{

//Disabled the Life image to inform player how many lives they have left

Lives.enabled = false;

Lives2.enabled = false;

Berries.count = 1; //Variable used to count is reset

}

else if (pacManLives == 0) //If PacMan has 0 Lives left then the following code is ran

{

GameObject pacMan = GameObject.Find("PacMan"); //Finds PacMan

pacMan.transform.GetComponent<PacMan>().canMove = false; //Stops Pacman from moving

GameObject[] o = GameObject.FindGameObjectsWithTag("Ghost"); //Finds all of the Ghosts and stores in an array called o

foreach (GameObject ghost in o)

{

ghost.transform.GetComponent<Ghost>().canMove = false; //Stops ghosts from moving

}

//Disabled the Life image to inform player how many lives they have left

Lives.enabled = false;

Lives2.enabled = false;

Lives3.enabled = false;

//Makes the following Fields Visible

inputField.SetActive(true);

hideButton.SetActive(true);

inputField.SetActive(true);

submitButton.onClick.AddListener(SaveData); //Saves the Users Data when the button is pressed

}

//Used to reduce errors by resetting PacMan lives to 0 if they ever go below 0 for any bug related reason at all

if (pacManLives < 0)

{

pacManLives = 0;

}

}

//Code below checks to see how many pellets PacMan has consumed

void CheckPelletsConsumed()

{

if (totalPellets == PacMan.pelletConsumed) //Check if the Player has eaten all the pellets

{

PacMan.pelletConsumed = 0; //Resets the amount of Pellets eaten by PacMan

PlayerWin(1); //Launches the sub-routine by the same name

}

}

//The Code Below is ran when the player eats all of the pellets

void PlayerWin(int playerNum)

{

if (playerNum == 1)

{

GameBoard.playerLevel++; //Increases the players level via increment

}

StartCoroutine(ProcessWin(2));

}

IEnumerator ProcessWin(float delay) //starts the win process by causing a delay which is stated in the parenthesis when called

{

GameObject pacMan = GameObject.Find("PacMan");

pacMan.transform.GetComponent<PacMan>().canMove = false; //Stops Pacman from moving

GameObject[] o = GameObject.FindGameObjectsWithTag("Ghost");

foreach (GameObject ghost in o)

{

ghost.transform.GetComponent<Ghost>().canMove = false; //Stops ghosts from moving

}

yield return new WaitForSeconds(delay); //Will be used to animate the blinking of the board, if the player does win as the board blinks

StartNextLevel(); //Starts the next level

Berries.berryTimer = 0f; //Value of timer is reset

Berries.count = 1; //Count Variable is reset

Berries.count2 = 1; //Count2 Variable is reset

}

private void StartNextLevel()

{

//SceneManager.LoadScene(SceneManager.GetActiveScene().name);

SceneManager.LoadScene("Level1"); //Will restart the scene allowing the player to continue playing once they've eaten all of the pellets of one level

}

public void Restart() //This is used to restart the whole game for when PacMan is caught

{

pacManLives -= 1; //Decreases PacMans lives each time he's caught

Berries.count = 1;

GameObject pacMan = GameObject.Find("PacMan"); //Finds Pacman then runs the code below

pacMan.transform.GetComponent<PacMan>().Restart(); //Runs PacMans restart routine

GameObject[] o = GameObject.FindGameObjectsWithTag("Ghost"); //Finds all of the Ghosts in the game and stores them in an array

foreach (GameObject ghost in o) //This FOR EACH loop, loops through all the ghosts in the array and applies the restart method on them

{

ghost.transform.GetComponent<Ghost>().Restart();

}

}

private void SaveData()

{

//Path of the File

string path = ("E:/College Subjects/Computer Science/Computer Science NEA/My Games/PacMan Unity/Help/PacMan Clone/PacMan Clone/Assets/Text File/Username.txt");

//Gets writer which will write to the file in path

StreamWriter writer = new StreamWriter(path);

//Writes the line in the parameters

writer.WriteLine(textField.text + ", " + playerOneScoreText.text);

//Closes the Writer

writer.Close();

//Path of the File

string path2 = ("E:/College Subjects/Computer Science/Computer Science NEA/My Games/PacMan Unity/Help/PacMan Clone/PacMan Clone/Assets/Text File/Score.txt");

//Gets writer which will write to the file in path

StreamWriter writer2 = new StreamWriter(path2);

//Writes the line in the parameters

writer2.WriteLine(playerOneScoreText.text);

//Closes the Writer

writer2.Close();

Application.Quit(); //Closes the Application

}

}

***Nodes***:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Node : MonoBehaviour {

public Node[] neighbors;

public Vector2[] validDirections;

// Use this for initialization

void Start () {

//The Code Below Identifies the nodes adjacent to another nodes and these are used for Movement and Collision

validDirections = new Vector2[neighbors.Length];

for (int i = 0; i < neighbors.Length; i++) {

Node neighbor = neighbors [i];

Vector2 tempVector = neighbor.transform.localPosition - transform.localPosition;

validDirections [i] = tempVector.normalized;

}

}

}

***PacMan:***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PacMan : MonoBehaviour

{

public Vector2 orientation; //Holds PacMans orientation

public float speed = 4.0f; //This Variable will hold the speed at which Pacman will move at

public Sprite idleSprite; //This Variable allows PacMan to have a Idle sprite when he interacts with a tile

private Vector2 direction = Vector2.zero; //Variable will hold Pacmans direction in it so if it's moving left, right, up or down

private Vector2 nextDirection; //This variable will be used to check which direction PacMan will want to Move to

public bool canMove = true; //Dictates wheather PacMan can Move or Not

public static int pelletConsumed = 0; //Is used to count how many pellets are eaten

private Node currentNode, targetNode, previousNode; //These variables are Node type variables and will store which node pacman is currently on, TargetNode is the Node we are actually going to move to, previousNode will be used to track the node we came from

private Node startingPosition; //Stores PacMans starting Position for when he gets consumed by a ghost so he can move to it when the game is reset

// Use this for initialization

void Start() {

Node node = GetNodeAtPosition(transform.localPosition); //We get the Postion of the Node that Pacman is on as localPosition is just refernecing PacMan

startingPosition = node; //Makes startingPosition equal to the Node that PacMan origianlly starts at

if (node != null) //If the position of pacman is on is a node then the following code is executed

{

currentNode = node; //Pacman current location is changed to the node

}

direction = Vector2.left; //As in the Original game PacMan starts facing left

orientation = Vector2.left;

ChangePosition(direction); //Calls the sub routine by the same name here

}

//The Code below is used to reset PacMan when he gets caught by a ghost

public void Restart()

{

canMove = true; //Allows PacMan to Move

transform.position = startingPosition.transform.position; //Resets PacMans starting position to his original position

currentNode = startingPosition; //Sets PacMans currentNode to where he started

direction = Vector2.left; //As in the Original game PacMan starts facing left

orientation = Vector2.left; //Orientation will be left originally as PacMan moves left originally so will face left

nextDirection = Vector2.left;

ChangePosition(direction); //Restarts PacMan Position allowing it to change from its original position

}

// Update is called once per frame

void Update() {

if (canMove == true) //If PacMan can move then the following code is executed

{

CheckInput(); //Sub routine by the same name is called here

Move(); //Sub routine by the same name is called here

UpdateOrientation(); //Sub routine by the same name is called here

UpdateAnimationState(); //Sub Routine by the same name is called here

ConsumePellet(); //Sub Routine by the same name is called here

}

}

void CheckInput() {

if (Input.GetKeyDown(KeyCode.LeftArrow)) { //If the Left Key is pressed down then the following code is executed as the condition in the subroutine is fuffiled

ChangePosition(Vector2.left); //The ChangePosition Routine is called with Vector2.direction we are trying to move in

}

else if (Input.GetKeyDown(KeyCode.RightArrow)) //If the Right Key is pressed down then the following code is executed

{

ChangePosition(Vector2.right); //The ChangePosition Routine is called with Vector2.direction we are trying to move in

}

else if (Input.GetKeyDown(KeyCode.UpArrow)) //If the Up arrow key is pressed then as the condition in the IF statement is fufilled then the following code is executed

{

ChangePosition(Vector2.up); //The ChangePosition Routine is called with Vector2.direction we are trying to move in

}

else if (Input.GetKeyDown(KeyCode.DownArrow)) //If the Down key is pressed then the following code is executed

{

ChangePosition(Vector2.down); //The ChangePosition Routine is called with Vector2.direction we are trying to move in

}

}

void ChangePosition(Vector2 d) //This will allow PacMan to change his direction

{

if (d != direction) //If this statement is correct the follwing code will be executed, the condition is that when d doesn't equal our direction

nextDirection = d; //If the condition above is fuffiled then this piece of Code will be executed where out next direction = our current direction (d)

if (currentNode != null) //If we are on a Node then the following code is executed

{

Node moveToNode = CanMove(d); //We move to the Node we are trying to reach

if (moveToNode != null) //If we are trying to move to a Node the following code will be run as the postion we are trying to will not be empty as it contains a Node

{

direction = d; //The direction becomes the direction we are trying to move in

targetNode = moveToNode; //Our targetNode becomes our moveToNode as we are trying to move to that Node

previousNode = currentNode; //The Node we were on before (or on if we just moved) becomes our previous node

currentNode = null; //Current Node is reset back to Null

}

}

}

void Move() //This doesn't return anything so is a void and will control Pacman's movement

{

if (targetNode != currentNode && targetNode != null) //If the targetNode doesn't equal the node we are on and the targetNode isn't empty then the following code is ran

//The IF statement below allows me to change direction in between Nodes

if (nextDirection == direction \* -1) //IF statement makes it so that the direction we want to move changes this is done by inversing the original direction (\* -1) so it travels in the opposite direction

{

direction \*= -1; //Sets Direction to the opposite in the direction it is travelling, so if PacMan is moving up you can move it down if you want and vice versa

Node tempNode = targetNode; //Temp variable to store a Node for a while whilst other code is ran before, used to value of targetNode isn't lost

targetNode = previousNode;

previousNode = tempNode;

}

{

if (OverShotTarget()) //Calls the OverShotTarget function here to check against

{

currentNode = targetNode; //currentNode becomes equivilant to our targetNode

transform.localPosition = currentNode.transform.position; //Our position is moved to the same position as the currentNode

GameObject otherPortal = GetPortal (currentNode.transform.position); //Code moves PacMan to the other Portal

if (otherPortal != null) //If the otherPortal isn't null (if the variable isn't empty then the following code is ran)

{

transform.localPosition = otherPortal.transform.position;

currentNode = otherPortal.GetComponent<Node> ();

}

Node moveToNode = CanMove(nextDirection); //We use the variable moveToNode and set it equal to CanMove with the direction we want to move in next

if (moveToNode != null) //If the node we are trying to move to isn't empty the following code is executed

direction = nextDirection; //Our direction is updated with the new Direction

if (moveToNode == null) //If the node we are trying to move to is empty so equals null then the following code is executed

moveToNode = CanMove(direction); //The CanMove function is run with the direction we are trying to move in and is made equal to the node PacMan is trying to get to

if (moveToNode != null) //If the node we are trying to move to isn't empty the following code is executed

{

targetNode = moveToNode; //Our targetNode becomes our moveToNode as we are trying to move to that Node

previousNode = currentNode; //The Node we were on before (or on if we just moved) becomes our previous node

currentNode = null; //Current Node is reset back to Null

}

else //If the condition above is not fuffilled then the code below is ran instead

{

direction = Vector2.zero;

}

}

else //If the condition above is not fuffilled then the code below is ran instead

{

transform.localPosition += (Vector3)(direction \* speed) \* Time.deltaTime; //This piece of code basically moves Pacman in the direction it is facing at a pace of the float variable called speed

}

}

}

void UpdateOrientation() {

if (direction == Vector2.left) { //If the direction of the object (Pacman) changes than the game is updated with the new direction, here if the direction is changed to left so Pacman faces left than the following code is executed

orientation = Vector2.left; //Makes the left Orientation the same as the left vector2

transform.localScale = new Vector3(-1, 1, 1); //Pacman transform inversely so it is resized with its inital values however the -1 as the x value inverses pacman so it faces the opposite direction

transform.localRotation = Quaternion.Euler(0, 0, 0); //Pacman is not rotated as its not going to be facing up or down

} else if (direction == Vector2.right) { //If the direction of Pacman is changed to right then the following code is executed

orientation = Vector2.right; //Makes it so orientation is the same as the right vector2

transform.localScale = new Vector3(1, 1, 1); //Pacman is made to transform to all positive values so it faces right

transform.localRotation = Quaternion.Euler(0, 0, 0); //Pacman doesn't rotate as it's not facing up or down

} else if (direction == Vector2.up) { //If Pacmans direction is changed so it faces upwards

orientation = Vector2.up; //Makes it so orientation is the same as when PacMan is facing up and thus is given the same vector2

transform.localScale = new Vector3(1, 1, 1); //It's not resized

transform.localRotation = Quaternion.Euler(0, 0, 90); //Pacman is rotated so it faces 90 degrees on the set axis as a result it will face up

} else if (direction == Vector2.down) { //If Pacmans direction is changed so it faces downwards than the following code is ran

orientation = Vector2.down; //Makes it so orientation is the same as when PacMan is facing down and thus is given the same vector2

transform.localScale = new Vector3(1, 1, 1); //Pacman isn't inversed size wise so it doesn't change scale wise

transform.localRotation = Quaternion.Euler(0, 0, 270); //Pacman is set to face 270 degrees on the set axis as a result it will face downwards

}

}

void UpdateAnimationState() //This is used to update the Animation of PacMan

{

//Code below makes it so when PacMan collides with a tile (reaches a Node where he cannot move in that single direction anymore) his munching animation stops and remains in a static animation until the next userinput moves him away from that direction

if (direction == Vector2.zero)

{

GetComponent<Animator>().enabled = false; //Disables the Animator

GetComponent<SpriteRenderer>().sprite = idleSprite; //Sets the sprite to the sprite stored in idleSprite which will stop PacMan from opening and closing his mouth

}

else

{

GetComponent<Animator>().enabled = true; //Enables the Animator

}

}

void ConsumePellet() //Used to allow PacMan to eat the pellets and superpellets

{

GameObject o = GetTileAtPosition(transform.position);

if (o != null) //If the tile PacMan is occupying isn't null (empty) then the following code is ran

{

Tile tile = o.GetComponent<Tile> (); //Temp variable called tile is made which stores the tile PacMan is on

if (tile != null) //If the tile isn't empty (null) then the following code is ran

{

if (!tile.didConsume && (tile.isPellet || tile.isSuperPellet)) //If the tile is consumed and is either a pellet or super pellet than the following code is ran

{

o.GetComponent<SpriteRenderer>().enabled = false; //The visibility of the pellet/superpellet is turned off as their sprite is no longer going to render as the render is now set to false

tile.didConsume = true; //Boolean did consume becomes true

if (tile.isSuperPellet) //Short Hand IF statement as only one line of code is executed each time so we can ignore the curly brackets

GameBoard.Score += 50; //If PacMan eats a super pellet then he gains 50 points

else

GameBoard.Score += 10; //Increases the Players Score by 10 for each pellet consumed

pelletConsumed++;

if (tile.isSuperPellet) //If the tile which is eaten by PacMan is a super pellet the following code is ran

{

GameObject[] ghosts = GameObject.FindGameObjectsWithTag("Ghost"); //Stores all the Ghosts in a Array

foreach (GameObject go in ghosts) //Goes through each Ghost and executed the following code

{

go.GetComponent<Ghost>().startFrightenedMode(); //Puts all of the Ghosts in Frightened Mode

}

}

}

}

}

}

void MoveToNode(Vector2 d)

{

Node MovetoNode = CanMove(d); //This variable is used to check if the we can move to the Node or not

if (MovetoNode != null) //If the position we want to move to is a Node then the follwing code is executed

{

transform.localPosition = MovetoNode.transform.position; //Pacman is moved to the Node we are checking if we can move to or not

currentNode = MovetoNode; //Pacmans current Node becomes the node we just moved to

}

}

GameObject GetTileAtPosition (Vector2 pos)

{

int tileX = Mathf.RoundToInt(pos.x); //This code rounds the X position to a whole number and saves it in the varaible tileX

int tileY = Mathf.RoundToInt(pos.y); //This code rounds the Y position to a whole number and saves it in the variable tileY

GameObject tile = GameObject.Find("Game").GetComponent<GameBoard>().board[tileX, tileY]; //Gets allthe tiles from the gameboard into tile

if (tile != null)

return tile;

return null;

}

Node GetNodeAtPosition (Vector2 pos)

{

GameObject tile = GameObject.Find("Game").GetComponent<GameBoard>().board [(int)pos.x, (int)pos.y]; //Here it creates a game object called Tile and we are taking the game component from the game object and then using the board array we are taking the value at pos x and y

if (tile != null) //If the tile does equal null we will get a null reference error

{

return tile.GetComponent<Node> (); //Returns the value of the component of that node

}

return null; //If tile does equal null then we just return null (so nothing)

}

bool OverShotTarget() //Function returns a True or False at the end

{

//Variables are declared here and set to values which can change when the program is run

float NodetoTarget = LengthFromNode(targetNode.transform.position);

float NodetoSelf = LengthFromNode(transform.localPosition);

return NodetoSelf > NodetoTarget; //Return True or False, it will be True when we overshot our target otherwise it will be false

}

float LengthFromNode(Vector2 targetPosition) //Function is declared where in the parenthasis a Value is required for Vector 2

{

Vector2 vec = targetPosition - (Vector2)previousNode.transform.position; //Here vec is declared as a variable which equals the targetPosition - the position of our previous node

return vec.sqrMagnitude; //Returns the squared length of vec (Vector2)

}

Node CanMove(Vector2 d)

{

Node MoveToNode = null; //This variable will be used to check if we can Move to the Next Node or not, if we can then the node we can move to is set equal to this variable

for (int i = 0; i < currentNode.neighbors.Length; i++) //Iterating through all the neighbours of the Node we are currently on, at every iteration we check if all the valid direction we can move, if Pacman can move to the next node then the following code is ran

if (currentNode.validDirections[i] == d) //If the Node we are trying to go to is a valid node in that direction the following code is ran

{

MoveToNode = currentNode.neighbors[i]; //We move to the neighbour of our current node in the direction we are trying to move in

break; //This makes it so the loop is not ran over again as MovetoNode will be reset to null

}

return MoveToNode;

}

GameObject GetPortal (Vector2 pos)

{

GameObject tile = GameObject.Find ("Game").GetComponent<GameBoard> ().board [(int)pos.x, (int)pos.y]; //Finds all objects from game which fit this description

if (tile != null) //If the tile varaible isn't empty so isn't null the follwing code is ran

{

if (tile.GetComponent<Tile>() != null)

{

if (tile.GetComponent<Tile>().isPortal) //Checks to see if PacMan is on a Portal Tile (Node) if it is then the following code is ran

{

GameObject otherPortal = tile.GetComponent<Tile>().portalReceiver;

return otherPortal; //Used so you can come out of the other portal if you enter the one on the opposite side

}

}

}

return null; //Function returns nothing if the conditons aren't met

}

}

***Ghost:***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Ghost : MonoBehaviour {

//The speed the ghosts move at

public float moveSpeed = 3.9f;

//The Speed the ghosts will have when they are in frightened mode

public float frightenedModeMoveSpeed = 2.9f;

//Stores the ghosts normal move speed, used for robustness

public float normalMoveSpeed = 3.9f;

//Stores the speed the ghosts move at if they've been consumed

public float consumedMoveSpeed = 15f;

//The Speed the Ghosts had before the Mode changed allows for us to reverse it

private float previousMoveSpeed;

//The time it will take before Pinky will be able to leave the Ghost House

public int pinkyReleaseTimer = 5;

//The time it will take before Inky will be able to leave the Ghost House

public int inkyReleaseTimer = 14;

//The time it will take before Clyde will be able to leave the Ghost House

public int clydeReleaseTimer = 21;

//The Time it will take for Ghost to start Chasing PacMan

public float ghostReleaseTimer = 0;

//Time used to count the duration of frightened mode

private int frightenedModeDuration = 10;

//Time for when the ghosts start blinking during their frightened mode

private int startBlinkingAt = 7;

//Used to check if the Ghost is in the ghost house (centre rectangle) or not

public bool isInGhostHouse = false;

//Dictates wheather the ghosts can move or not

public bool canMove = true;

//The position of the ghosts at the start

public Node startingPosition;

//The variable below will be used to assign each ghost the corner they move to in scatter mode

public Node homeNode;

//Used to let ghosts know where their house is so they can return there when they have been consumed

public Node ghostHouse;

//The Variable Below resets the ghosts when they change modes

public RuntimeAnimatorController ghostLeft;

//The Variables are used to run animations when the player eats a super pellet

public RuntimeAnimatorController ghostBlue;

public RuntimeAnimatorController ghostWhite;

//The Variables below are used as Timers which will count down and lead to different mode that the ghosts move in

public int scattermodeTimer1 = 7;

public int chasemodeTimer1 = 20;

public int scattermodeTimer2 = 7;

public int chasemodeTimer2 = 20;

public int scattermodeTimer3 = 5;

public int chasemodeTimer3 = 20;

public int scattermodeTimer4 = 5;

private int modeChangeIteration = 1;

private float modeChangeTimer = 0;

//The Variables below are the Timers for Frightened Mode

public float frightenedModeTimer = 0;

public float blinkTimer = 0;

//Variable used to determmine if the ghosts are supposed to blink or not

private bool frightenedModeisWhite = false;

//The enumuration below will allow me to change through from each mode more easily

private enum Mode

{

Chase,

Scatter,

Frightened,

Consumed

}

//The starting mode is Scatter mode so that's waht the current mode will start of with

Mode currentMode = Mode.Scatter;

Mode previousMode;

//Enum will allow me to use all the different ghosts

public enum GhostType

{

Red,

Pink,

Blue,

Orange

}

//Instantiate the ghost type

public GhostType ghostType = GhostType.Red;

//This Variable will be used to track PacMan

private GameObject pacMan;

//These variables will be used to track the location the Ghosts are on and what direction they want to move in and go

private Node currentNode, targetNode, previousNode;

private Vector2 direction, nextDirection;

// Use this for initialization

void Start ()

{

pacMan = GameObject.FindGameObjectWithTag("PacMan"); //Finds PacMan and makes it equal to pacMan

Node node = GetNodeAtPosition(transform.localPosition); //New node is declared and becomes equivalent too the position of the ghost

if (node != null) //If the node isn't empty then the following code is ran

{

currentNode = node; //CurrentNode becomes equal to node so the currentNode is initially equal to the Ghosts starting position

}

if (isInGhostHouse) //If the ghost is in the ghost house the following code is ran

{

direction = Vector2.up; //The Ghost moves up to leave the ghost house

targetNode = currentNode.neighbors[0]; //This is 0 as there is only one neighbour in the index which is the node above the ghost house

}

else

{

direction = Vector2.left; //Initially the Ghost moves left

targetNode = ChooseNextNode();

}

previousNode = currentNode; //sets the previous node to the current node as the current node is going to change

UpdateAnimatorController(); //Updates the Animations

}

public void Restart() //Resets all ghost variabels to their original values and resets the ghosts position to their original position

{

canMove = true; //Allows the ghosts to move

transform.position = startingPosition.transform.position; //Moves all of the ghosts back to their original position

ghostReleaseTimer = 0; //Resets this timer, controls whent the ghosts are released from the ghost house

Berries.berryTimer = 0f; //Resets the Berry Timer

modeChangeIteration = 1; //Resets the iteration of the Ghosts so when they leave the ghost house they follow their original time limits on each mode

modeChangeTimer = 0; //Resets this Timer which controls when the Modes chage

if (transform.name != "Red\_Ghost")

isInGhostHouse = true; //Makes it so blinky (the Red Ghost) is also reset even though he doesn't originally start inside the ghost house

currentNode = startingPosition; //Resets the currentNode of the ghosts so it starts of equalling the ghosts starting points

if (isInGhostHouse) //If the Ghosts are in the ghost house then the following code is ran

{

direction = Vector2.up; //The Ghosts move upwards to leave the ghost house

targetNode = currentNode.neighbors [0];

}

else //If the Ghost isn't in the ghost house initially (so blinky i.e the red ghost) then the following code is ran

{

direction = Vector2.left; //The ghost initally moves left

targetNode = ChooseNextNode();

}

previousNode = currentNode; //Resets the value of the previous Node so it starts of intially with the currentNode that the Ghosts are on

UpdateAnimatorController(); //Updates any animation if it's required

}

// Update is called once per frame

void Update () {

if (canMove == true) //If the Ghosts can move then the following code is executed

{

ModeUpdate(); //Calls the sub routine by the same name here

Move(); //Calls the sub routine by the same name here

ReleaseGhosts(); //Calls the sub routine by the same name here

CheckCollision(); //Calls the sub routine by the same name here

CheckIsInGhostHouse(); //Calls the sub routine by the same name here

}

}

//Code below checks to see if the Ghosts are in the ghost house or not

void CheckIsInGhostHouse()

{

if (currentMode == Mode.Consumed) //If the ghosts have been eaten by PacMan when they were in frightened mode their mode is changed to consumed and if they are in consumed mode then the following code is ran

{

GameObject tile = getTileAtPosition(transform.position); //Get the Tile at the Ghosts current position

if (tile != null) //If there is a value stored in tile then the following code is ran for it

{

if (tile.transform.GetComponent<Tile> () != null)

{

if (tile.transform.GetComponent<Tile>().isGhostHouse) //Checks to see if the ghosts are in the ghost house if they are then the following code is ran

{

moveSpeed = normalMoveSpeed; //The ghost move speed is changed to the normal move speed that they move at

Node node = GetNodeAtPosition(transform.position); //Gets the node the ghosts is at

//Updates the Node Values

if (node != null)

{

currentNode = node;

direction = Vector2.up; //Makes Ghosts move up to leave the ghost house

targetNode = currentNode.neighbors[0];

previousNode = currentNode;

currentMode = Mode.Chase; //Sets Ghost in Chase Mode

UpdateAnimatorController(); //Updates Animations

}

}

}

}

}

}

//Code below checks if a collision with pacman has occured or not

void CheckCollision()

{

Rect ghostRect = new Rect(transform.position, transform.GetComponent<SpriteRenderer>().sprite.bounds.size / 4); //Used to consistently make rectangles for the ghost which will be used in collision detection

Rect pacManRect = new Rect(pacMan.transform.position, pacMan.transform.GetComponent<SpriteRenderer>().sprite.bounds.size /4); //Used to make a rectangle around PacMan of the same size as his sprite which can then be used for collision detection

//The Code Below will either return true or false which will be used to tell if we have a collision with PacMan or not

if (ghostRect.Overlaps(pacManRect))

{

if (currentMode == Mode.Frightened)

{

Consumed();

}

else //If the current mode isn't frightened then the following code is ran

{

if (currentMode != Mode.Consumed)

{

GameObject.Find("Game").transform.GetComponent<GameBoard>().Restart(); //Calls the restart method from gameboard thus causing the whole game to restart

}

}

}

}

void Consumed()

{

GameBoard.Score += GameBoard.ghostConsumedRunningScore; //Increases the players score when they eat a ghost

currentMode = Mode.Consumed; //Changes the Ghosts current Mode

previousMoveSpeed = moveSpeed; //Stores the Ghosts current speed as previous speed

moveSpeed = consumedMoveSpeed; //Updates the Ghosts Current Move speed to the speed they move at when they are consumed

UpdateAnimatorController(); //Updates the Animations

GameBoard.ghostConsumedRunningScore = GameBoard.ghostConsumedRunningScore \* 2; //Multuplies the ghosts score by 2 each time a ghost is eaten as this is what happened in the original game

}

void UpdateAnimatorController() //Updates the Animations

{

if (currentMode != Mode.Frightened) //If the Ghosts are not in frightened mode then the following code is ran

{

if (direction == Vector2.up || direction == Vector2.left || direction == Vector2.right || direction == Vector2.down) //If the ghosts are moving in any of the stated directions then the following code is ran

{

//If I wanted to add animations for each direciton this is where I would add them with the correct direction using a IF ... ELSE IF statement

transform.GetComponent<Animator>().runtimeAnimatorController = ghostLeft; //Get the Ghost moving left animation

}

}

else //If the Ghosts are in Frightened Mode then the Following code is ran

{

transform.GetComponent<Animator>().runtimeAnimatorController = ghostBlue; //The Ghosts get their blue scared animation

}

}

//Code Below allows the Ghosts to Move

void Move()

{

if (targetNode != currentNode && targetNode != null && !isInGhostHouse) //If the targetNode has a value and it isn't the currentNode then the following code and the ghost is in the ghost house then the following code is ran

{

if (OverShotTarget()) //Checks to see if we overshot the target or not

{

currentNode = targetNode; //If we did Overshoot the target then our currentNode becomes our targetNode

transform.localPosition = currentNode.transform.position;

GameObject otherPortal = GetPortal(currentNode.transform.position); //Check to see if we are on a portal

if (otherPortal != null) //Makes sure we have a portal

{

transform.localPosition = otherPortal.transform.position; //Moves our ghosts position to the other portal

currentNode = otherPortal.GetComponent<Node>(); //Makes out currentNode equal to the portal node that we just moved to

}

targetNode = ChooseNextNode(); //Launches ChooseNextNode and assigns the returned value to targetNode so where the ghost is trying to move

previousNode = currentNode; //Updates the previous node with the current node as current node is about to be updated

currentNode = null;

UpdateAnimatorController(); //Calls the sub by the same name

}

else //If we didn't overshoot our target the following code is ran

{

transform.localPosition += (Vector3)direction \* moveSpeed \* Time.deltaTime; //This code increments the ghosts position basically allowing the ghost to move

}

}

}

void ModeUpdate() //This routine allows the mode of the ghosts to change as time goes on

{

if (currentMode != Mode.Frightened) //If Ghosts aren't in Frightened Mode then the following code is ran

{

modeChangeTimer += Time.deltaTime; //The Timer is started

if (modeChangeIteration == 1) //When modeChangeIteration equals one then the following code is ran

{

if (currentMode == Mode.Scatter && modeChangeTimer > scattermodeTimer1) //Checks to see if we are in scatter mode or not and if it is for the following code to run the modechangeTimer would have to have exceeded the scattermodeTimer meaning the mode neads to change

{

ChangeMode(Mode.Chase); //Changes the Ghosts to Chase Mode at the correct iteration of the time

modeChangeTimer = 0; //Resets the Timer

}

if (currentMode == Mode.Chase && modeChangeTimer > chasemodeTimer1) //If the Timer exceeds the chasemode timer and the current mode is chase then the following code is ran

{

modeChangeIteration = 2; //Changes the Mode iteration to 2 as in PacMan the ghosts follow this loop 4 times, alternating between chase and scatter 4 times before sticking to chase mode

ChangeMode(Mode.Scatter); //Changes the Ghosts to scatter Mode

modeChangeTimer = 0; //Resets the Timer

}

}

//The ELSE IF statements below check to see which iteration we are on so do we use chasemodeTimer1 or chasemodeTimer2 etc

else if (modeChangeIteration == 2)

{

if (currentMode == Mode.Scatter && modeChangeTimer > scattermodeTimer2) //If the Ghosts are in scatter mode and the timer has exceed that time then the following code is ran

{

ChangeMode(Mode.Chase); //Ghosts change to Chase mode

modeChangeTimer = 0; //Resets the Timer

}

if (currentMode == Mode.Chase && modeChangeTimer > chasemodeTimer2) //If the Ghosts are in chase mode and timer exceeds the chase timer then the following code is ran

{

modeChangeIteration = 3; //Calls the third iteration of mode changes

ChangeMode(Mode.Scatter); //Changes the Ghosts to Scatter Mode

modeChangeTimer = 0; //Resets the timer

}

}

else if (modeChangeIteration == 3) //If we are in the third iteration then the following code is ran

{

if (currentMode == Mode.Scatter && modeChangeTimer > scattermodeTimer3) //If we are in scatterMode and modeChangeTimer exceeds scatterModeTimer then the following code is ran

{

ChangeMode(Mode.Chase); //Changes the Ghosts to Chase Mode

modeChangeTimer = 0; //Resets the Timer

}

if (currentMode == Mode.Chase && modeChangeTimer > chasemodeTimer3) //If we are in Chase Mode and the modeChangeTimer exceeds the chase Timer then the following code is ran

{

modeChangeIteration = 4; //We Move on to our 4th iteration

ChangeMode(Mode.Scatter); //Changes the ghost mode to scatter mode

modeChangeTimer = 0; //Resets the Timer

}

}

else if (modeChangeIteration == 4)

{

// The IF statement below does not contain a scatter mode as in the original PacMan game after 4 iterations of the Timer the game just let the ghosts stay in Chase Mode

if (currentMode == Mode.Scatter && modeChangeTimer > scattermodeTimer4)

{

ChangeMode(Mode.Chase);

modeChangeTimer = 0;

}

}

}

else if (currentMode == Mode.Frightened) //If Ghosts are in Frightened Mode then the following code is ran

{

frightenedModeTimer += Time.deltaTime; //Increments the Timer

if (frightenedModeTimer >= frightenedModeDuration) //Once the Frightened Mode Timer as reached it max duration the following code is ran

{

frightenedModeTimer = 0; //Value is reset

ChangeMode(previousMode); //Mode is changed back to its previous mode

}

if (frightenedModeTimer >= startBlinkingAt) //If FrightendModetimer exceeds the startBlinking period then the following code is ran

{

blinkTimer += Time.deltaTime; //Starts Blink Timer, allows ghosts to blink to inform player that frightend mode is nearly over

if (blinkTimer >= 0.1f)

{

blinkTimer = 0f; //Resets the Blink Timer

if (frightenedModeisWhite) //Checks the Boolean, initally false so the following code is ran which alternates the ghost between blue and white

{

//Code below allows Ghosts to blink by alternating a bool variable for if the ghost is white or not and by using the animator to change the ghosts animation between scared blue and white ghosts

transform.GetComponent<Animator>().runtimeAnimatorController = ghostBlue;

frightenedModeisWhite = false;

}

else

{

transform.GetComponent<Animator>().runtimeAnimatorController = ghostWhite;

frightenedModeisWhite = true;

}

}

}

}

}

void ChangeMode(Mode m) //This changes the mode we are playing at and as it isn't a function it doesn't return anything

{

if (currentMode == Mode.Frightened) //Checks if the Mode is Frightened then the movespeed gains speed as previous move speed if there is one

{

moveSpeed = previousMoveSpeed;

}

if (m == Mode.Frightened) //If the mode is frightened then the following code is ran

{

previousMoveSpeed = moveSpeed;

moveSpeed = frightenedModeMoveSpeed; //Ghosts gain the slower speed of frightened move speed

}

if (currentMode != m) //If our current mode isn't the mode we are trying to change to then the following code is ran, this is done to stop an infinite loop of a mode such as frightened mode as otherwise previous mode would get updated to be frightend mode again if it was called again thus causing an infinite loop

{

//Updates all of the Modes

previousMode = currentMode;

currentMode = m;

}

UpdateAnimatorController(); //Updates the Animations

}

public void startFrightenedMode()

{

if (currentMode != Mode.Consumed) //If the Ghosts haven't been consumed by PacMan then the following code is ran

{

GameBoard.ghostConsumedRunningScore = 200; //Everytime frightened mode is started and a ghost is consumed the score recieved from eating a ghost is always initally 200 and multiplies by 2 each time

frightenedModeTimer = 0; //Resets the Timer whenever Frightened Mode is started

ChangeMode(Mode.Frightened); //Changes the Mode to Frightened Mode

}

}

Vector2 GetRedGhostTargetTile()

{

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = new Vector2(Mathf.RoundToInt (pacManPosition.x), Mathf.RoundToInt (pacManPosition.y)); //Rounds PacMans position to an Integer

return targetTile; //Function outputs targetTile

}

Vector2 GetPinkGhostTargetTile()

{

//Has to be 4 tiles ahead of PacMan and need to take into account the positon and orientation

Vector2 pacManPosition = pacMan.transform.localPosition; //Variable stores PacMans location

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Stores PacMans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Rounds PacMans x coordinate to a whole number

int pacManPositionY = Mathf.RoundToInt(pacManPosition.y); //Rounds PacMans y coordinate to a whole number

Vector2 pacManTile = new Vector2(pacManPositionX, pacManPositionY);

Vector2 targetTile = pacManTile + (4 \* pacManOrientation); //Multiplies all of the vector2 by 4 so both x and y coordinate seperatly and then adds pacManTile to it so Pinky is always 4 tiles ahead of PacMan

return targetTile; //Function outputs targetTile

}

Vector2 GetBlueGhostTargetTile()

{

//Most Complex AI out of all of the Ghosts

//Select the Position two tiles in front of PacMan then draw a vector from blinky to that position then double the length of that vector

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan> ().orientation; //Gets Pacmans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Stores PacMan X coordinate in this variable

int pacmanPositionY = Mathf.RoundToInt(pacManPosition.y); //Stores PacMan Y cooridinate in this variable

Vector2 pacManTile = new Vector2 (pacManPositionX, pacmanPositionY); //Get PacMan Tile

Vector2 targetTile = pacManTile + (2 \* pacManOrientation); //Create a TargetTile for where PacMan is

Vector2 tempBlinkyPosition = GameObject.Find("Red\_Ghost").transform.localPosition; //Create a temp variable for Blinky position

int blinkyPositionX = Mathf.RoundToInt(tempBlinkyPosition.x); //Stores Blinkys x position in this variable

int blinkyPositionY = Mathf.RoundToInt(tempBlinkyPosition.y); //Stores Blinkys y position in this variable

tempBlinkyPosition = new Vector2(blinkyPositionX, blinkyPositionY); //Stores Blinkys coordinate in a new vector2 variable

float distance = GetDistanceApart(tempBlinkyPosition, targetTile); //Gets the distance apart between blinkys position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempBlinkyPosition.x + distance, tempBlinkyPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

return targetTile; //Outputs the Target Tile

}

Vector2 GetOrangeGhostTargetTile()

{

//Calculate the Distance that Clyde is from PacMan then if the distance > 8 tiles then targetting is same as Blinky else the target is his Home Node, so same as Scatter Mode

//This is because Clyde never catches PacMan is similar to a bluff to trick the player into breaking logic and moving into risky situations

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Clyde to PacMan

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

if (distance > 8) //If Distance is greater than 8 then the following code is ran

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y));

}

else if (distance < 8) //If Distance is less than 8 then the following code is ran

{

targetTile = homeNode.transform.position; //Makes the TargetTile the Home Node

}

return targetTile; //Outputs the targetTile

}

Vector2 GetTargetTile () //This function returns each ghosts target tile

{

Vector2 targetTile = Vector2.zero;

if (ghostType == GhostType.Red) //The red ghost will be released with the following code

{

targetTile = GetRedGhostTargetTile();

}

else if (ghostType == GhostType.Pink) //The pink ghost will be released with the following code

{

targetTile = GetPinkGhostTargetTile();

}

else if (ghostType == GhostType.Blue) //The blue ghost will be released with the following code

{

targetTile = GetBlueGhostTargetTile();

}

else if (ghostType == GhostType.Orange) //The orange ghost will be released with the following code

{

targetTile = GetOrangeGhostTargetTile();

}

return targetTile; //Returns (Outputs) the target tile

}

Vector2 GetRandomTile()

{

int x = Random.Range(0, 28); //Gets a random x value between the range of the maps width

int y = Random.Range(0, 36); //Gets a random y value between the range of the maps height

return new Vector2 (x, y); //Outputs the Random Position (so Tile/Node) the ghost moves to

}

void ReleasePinkGhost() //Used to Release the Pink Ghost from the Ghost House

{

if (ghostType == GhostType.Pink && isInGhostHouse) //If the ghost is the Pink ghost and is in the ghost house the following code is executed

{

isInGhostHouse = false; //Makes it so that the Pink Ghost leaves so is no longer in the house so the boolean variable becomes false

}

}

void ReleaseBlueGhost() //Used to release the Blue Ghost from the ghost house

{

if (ghostType == GhostType.Blue && isInGhostHouse) //If the ghost is the blue ghost and is in the ghost house then the following code is ran

{

isInGhostHouse = false; //Makes it so that the Blue ghost leaves so is no longer in the house so the boolean varaibele isInGhostHouse becomes false

}

}

void ReleaseOrangeGhost() //Used to release the Orange Ghost from the ghost house

{

if (ghostType == GhostType.Orange && isInGhostHouse) //If the ghost is the orange ghost and is in the ghost house then the following code is ran

{

isInGhostHouse = false; //Makes it so that the Orange ghost leaves so is no longer in the house so the boolean varaibele isInGhostHouse becomes false

}

}

void ReleaseGhosts()

{

ghostReleaseTimer += Time.deltaTime; //Counts the timer which will be used to execute time based events

if (ghostReleaseTimer >= pinkyReleaseTimer) //If the ghostreleasetimer has a greater value then the timer for the pinky release timer the following code is executed

ReleasePinkGhost(); //Pinky is released

if (ghostReleaseTimer >= inkyReleaseTimer) //If the ghostreleasetimer has counted over then inkys release timer then the following code is ran (this is will be literally when the timer is above the inky timer by the smallest possible amount)

ReleaseBlueGhost(); //Inky is released

if (ghostReleaseTimer >= clydeReleaseTimer) //If the ghostreleasetimer has a greater value then the timer for clyde release timer then the following code is executed

ReleaseOrangeGhost(); //Clyde is released

}

Node ChooseNextNode() //This function will tell the ghost AI which Node to move to next

{

Vector2 targetTile = Vector2.zero; //The target tile initially is (0, 0)

//The IF statement below allows for the ghosts to change mode and follow the mode that they should be in, this will be the same as the original game

if (currentMode == Mode.Chase)

{

targetTile = GetTargetTile(); //targetTile is now set by GetTargetTile which gets a different TargetTile according to each ghosts original algorithm

}

else if (currentMode == Mode.Scatter) //When program starts ghosts are initally going to be in scatter mode thus the following code will move each ghost to their home Node (their corner in the map)

{

targetTile = homeNode.transform.position; //The Ghosts move towards their home nodes

}else if (currentMode == Mode.Frightened) //If the Current Mode is Frightened Mode then the following code is ran

{

targetTile = GetRandomTile(); //The Ghosts move to random tiles across the map

}else if (currentMode == Mode.Consumed) //If the Ghosts have been consumed by pacman whilst they were in frightened mode then the following code is ran

{

targetTile = ghostHouse.transform.position; //The Ghosts move to the GhostHouse node they have been assigned

}

//A new Node variable is made here which has a initial value of nothing (null), this is used to allow ghosts to move from node to node

Node moveToNode = null;

Node[] foundNodes = new Node[4];

Vector2[] foundNodesDirection = new Vector2[4];

int nodeCounter = 0;

for (int i = 0; i < currentNode.neighbors.Length; i++)

{

if (currentNode.validDirections [i] != direction \* -1)

{

if (currentMode != Mode.Consumed) //If the Ghosts are not in Consumed Mode then the following code is ran

{

GameObject tile = getTileAtPosition(currentNode.transform.position); //Get a tile for the node the ghost is at

if (tile.transform.GetComponent<Tile>().isGhostHouseEntrance == true)

{

//If we found a Ghost House, thus we don't want to allow movement

if (currentNode.validDirections[i] != Vector2.down)

{

foundNodes[nodeCounter] = currentNode.neighbors[i];

foundNodesDirection[nodeCounter] = currentNode.validDirections[i];

nodeCounter++;

}

}

else //If we are at tile which isn't the ghost house tile then the following code is ran

{

foundNodes[nodeCounter] = currentNode.neighbors[i];

foundNodesDirection[nodeCounter] = currentNode.validDirections[i];

nodeCounter++;

}

}

else //If we are in consumed mode then we have to allow the ghosts to move in any direction so they can travel to the ghost house

{

foundNodes[nodeCounter] = currentNode.neighbors[i];

foundNodesDirection[nodeCounter] = currentNode.validDirections[i];

nodeCounter++;

}

}

}

if (foundNodes.Length == 1) //This means we have only found one Node, as a result of this the following code is ran

{

moveToNode = foundNodes[0];

direction = foundNodesDirection[0];

}

if (foundNodes.Length > 1) //If we have found more than one node

{

float leastDistance = 100000f;

for (int i = 0; i < foundNodes.Length; i++)

{

if (foundNodesDirection [i] != Vector2.zero)

{

float distance = GetDistanceApart(foundNodes[i].transform.position, targetTile);

if (distance < leastDistance)

{

leastDistance = distance;

moveToNode = foundNodes [i];

direction = foundNodesDirection[i];

}

}

}

}

return moveToNode;

}

Node GetNodeAtPosition (Vector2 pos)

{

//Creating a GameObject and setting it equal to the object in the GameBoard array at that position

GameObject tile = GameObject.Find("Game").GetComponent<GameBoard> ().board[(int)pos.x, (int)pos.y];

if (tile != null) //Checks to see if the object we grabbed is empty (null) or not

{

if (tile.GetComponent<Node> () != null) //Checks to see if the object we grabbed is a Node or not

{

return tile.GetComponent<Node>();

}

}

return null;

}

GameObject getTileAtPosition(Vector2 pos) //Used to get the Tile Position where the ghosts is at

{

//Two variables are madewhich store x and y values

int tileX = Mathf.RoundToInt(pos.x);

int tileY = Mathf.RoundToInt(pos.y);

//Stores everything in the gameboard at tileX and tileY in tile

GameObject tile = GameObject.Find("Game").GetComponent<GameBoard>().board[tileX, tileY];

if (tile != null) //If tile does equal something then the following code is ran

return tile;

return null;

}

GameObject GetPortal (Vector2 pos) //This function allows the Ghosts to use the same portals as PacMan as they could in the original game

{

GameObject tile = GameObject.Find("Game").GetComponent<GameBoard>().board[(int)pos.x, (int)pos.y]; //Searches throught the GameBoard array

if (tile != null)

{

if (tile.GetComponent<Tile> ().isPortal) //Checks to see if the Tile is a portal tile or not if it is then the following code is ran

{

GameObject otherPortal = tile.GetComponent<Tile>().portalReceiver; //Makes the tile in the other portal the reciever and stores its value in otherPortal

return otherPortal; //Outputs otherPortal

}

}

return null; //Otherwise returns null

}

float LengthFromNode (Vector2 targetPosition) //Calculatees the distance from the node to allow AI to get the fastest way to PacMan

{

Vector2 vec = targetPosition - (Vector2)previousNode.transform.position;

return vec.sqrMagnitude;

}

//Checks to see if we went over/past target or not

bool OverShotTarget()

{

float nodeToTarget = LengthFromNode(targetNode.transform.position); //Lets us get the position of the Node and thus the distance

float nodeToSelf = LengthFromNode(transform.localPosition); //Gets the distance from Ghost to Node

return nodeToSelf > nodeToTarget; //Returns a value if the Ghost is further away from the Node then the target

}

float GetDistanceApart (Vector2 posA, Vector2 posB) //Function allows the ghosts to know the distance apart they are from two positions A and B

{

float dx = posA.x - posB.x;

float dy = posA.y - posB.y;

float distance = Mathf.Sqrt(dx \* dx + dy \* dy); //Simple Mathematical formula used to calculate distance: distance = sqroot(x1 + x2)^2 + (y1 + y2)^2

return distance; //Returns the distance apart

}

}

***Berries:***

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Berries : MonoBehaviour {

public static int count = 1; //Count Variable used to keep track of how many times an even can take place

public static int count2 = 1; //Counts if a berry has been eaten or not

public int berryPoint = 100; //Amount of points gained by PacMan if he consumes a berry

public static float berryTimer = 0f; //BerryTimer keeps track of when berry can appear

public float privCounter = 0f;

public bool isConsumed; //Checks to see if berry has been consumed or not

private bool isVisible; //Checks to see if berry is visible or not

private bool canTick = true; //Used to tell if Timer Can Tick or not

private GameObject pacMan; //Stores PacMan

//Variables Below store all of the berries

private GameObject Strawberry;

private GameObject Cherry;

private GameObject Apple;

private GameObject Peach;

// Use this for initialization

void Start() {

pacMan = GameObject.FindGameObjectWithTag("PacMan"); //Finds PacMan and makes it equal to pacMan

Object[] objects = GameObject.FindObjectsOfType(typeof(GameObject));

foreach (GameObject o in objects) //Iterates through all of the objects and adds them to the Array which will contain all the objects in the scene

{

if (o.name == "Strawberry") //If object is strawberry

{

Strawberry = o; //Makes strawberry equal to its object

o.GetComponent<SpriteRenderer>().enabled = false; //Stop rendering its sprite

}

else if (o.name == "Cherry") //If object is Cherry

{

Cherry = o; //Makes Cherry equal to its object

o.GetComponent<SpriteRenderer>().enabled = false; //Stop rendering its sprite

}

else if (o.name == "Apple") //If object is Apple

{

Apple = o; //Makes Apple equal to its object

o.GetComponent<SpriteRenderer>().enabled = false; //Stop rendering its sprite

}

else if (o.name == "Peach") //If object is Peach

{

Peach = o; //Makes Peach equal to its object

o.GetComponent<SpriteRenderer>().enabled = false; //Stop rendering its sprite

}

}

}

// Update is called once per frame

void Update() {

int rnd = Random.Range(0, 5); //Gets a random number between the range 0 and 5

if (canTick == true) //If the timer is allowed to tick then the following code is ran

{

privCounter += Time.deltaTime; //privCounter is started

}

//Code Below checks to see if Berry can spawn in or not and if they are spawning in at the right timer and level

berryTimer += Time.deltaTime;

if (((berryTimer >= 45) && GameBoard.playerLevel == 1) && count2 == 1)

{

BerryGen(Cherry);

}

else if (((berryTimer >= 45) && GameBoard.playerLevel == 2) && count2 == 1)

{

BerryGen(Strawberry);

}

else if (((berryTimer >= 45) && GameBoard.playerLevel == 3) && count2 == 1)

{

BerryGen(Apple);

}

else if (((berryTimer >= 45) && GameBoard.playerLevel == 4) && count2 == 1)

{

BerryGen(Peach);

}

else if (((berryTimer >= 45) && GameBoard.playerLevel > 4) && count2 == 1)

{

if (rnd == 1)

BerryGen(Strawberry);

else if (rnd == 2)

BerryGen(Cherry);

else if (rnd == 3)

BerryGen(Peach);

else if (rnd == 4)

BerryGen(Apple);

else if (rnd == 5)

{

//Do Nothing

}

}

CheckCollision(); //Call Sub Routine by the same name here

}

//Checks to see if Berry and PacMan have collided or not

void CheckCollision()

{

Rect BerryRect = new Rect(transform.position, transform.GetComponent<SpriteRenderer>().sprite.bounds.size / 4); //Used to consistently make rectangles for the berries which will be used in collision detection

Rect pacManRect = new Rect(pacMan.transform.position, pacMan.transform.GetComponent<SpriteRenderer>().sprite.bounds.size / 4); //Used to make a rectangle around PacMan of the same size as his sprite which can then be used for collision detection

//The Code Below will either return true or false which will be used to tell if we have a collision with PacMan or not

if (BerryRect.Overlaps(pacManRect))

{

isConsumed = true; //Makes the isConsumed Variable True meaning makes it so the berry has been eaten

if (isVisible == true) //If the berry is visible so the player can see it then follwing code is ran

{

GameBoard.Score += 100; //Player gains 100 point

isVisible = false; //Makes it so Berry is no longer visible so player cannot gain an infinite amount of points

count2 -= 1; //Decreases Count by 1

}

//Code below disables all the berrys

Cherry.GetComponent<SpriteRenderer>().enabled = false;

Strawberry.GetComponent<SpriteRenderer>().enabled = false;

Apple.GetComponent<SpriteRenderer>().enabled = false;

Peach.GetComponent<SpriteRenderer>().enabled = false;

}

}

//Code Below spawns in Berrys which PacMan can eat to gain more points

void BerryGen(GameObject Berry)

{

int roll = Random.Range(0, 10); //Random Number between range 0 to 10 is given

if (count == 1 && roll == 5) //If count is 1 and the number given by roll is 5 then the following code is ran

{

Berry.GetComponent<SpriteRenderer>().enabled = true; //The Berry stated is spawned in

isVisible = true; //Berry becomes visible

isConsumed = false; //Berry hasn't been eaten

count -= 1; //Reduces count by 1 so an infinte amount of berrys cannot spawn in

}

if (privCounter >= 45) //If privCounter has exceeded the stated value then the following code is ran

{

canTick = false; //Can tick is set to false

Berry.GetComponent<SpriteRenderer>().enabled = false; //Berry is disabled

}

}

}

***PacMan Multiple Map Ghost AI Code (My AI):***

Vector2 GetRedGhostTargetTile()

{

//This Ghost will follow PacMan as Normal

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y)); //Rounds PacMans position to an Integer

return targetTile; //Function outputs targetTile

}

Vector2 GetPinkGhostTargetTile()

{

//This Ghost will follow PacMan until it gets 2 tiles away from PacMan in which case it will return to the center of the Map (towards Ghost House)

Vector2 pacManPosition = pacMan.transform.localPosition; //This variable stores PacMans position

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Ghost to PacMan

if (distance > 2) //Checks to see how far away the Ghost is from PacMan

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y)); //Rounds PacMans position to an Integer

}

else if (distance <= 2) //If the inital requirement isn't fulfilled then the following code is ran

{

targetTile = GameObject.Find("pellet (206)").transform.localPosition; //Makes the Ghost go back to its starting position

}

return targetTile; //Function outputs targetTile

}

Vector2 GetBlueGhostTargetTile()

{

//Takes into account where the Pink Ghost is and doubles the distance however if the Pink Ghost is at its starting position then this Ghost takes into account the Red Ghosts Position

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Gets Pacmans Orientation

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Stores PacMan X coordinate in this variable

int pacmanPositionY = Mathf.RoundToInt(pacManPosition.y); //Stores PacMan Y cooridinate in this variable

Vector2 pacManTile = new Vector2(pacManPositionX, pacmanPositionY); //Get PacMan Tile

Vector2 targetTile = pacManTile + (2 \* pacManOrientation); //Create a TargetTile for where PacMan is

Vector2 tempPinkPosition = GameObject.Find("Pink Ghost").transform.localPosition; //Create a temp variable for Pinks position

if (tempPinkPosition.x != Mathf.RoundToInt(GameObject.Find("pellet (206)").transform.localPosition.x) & (tempPinkPosition.y != Mathf.RoundToInt(GameObject.Find("pellet (206)").transform.localPosition.y))) //Checks to see if the Pink Ghost is at its starting position or not

{

int PinkPositionX = Mathf.RoundToInt(tempPinkPosition.x); //Stores Pinks x position in this variable

int PinkPositionY = Mathf.RoundToInt(tempPinkPosition.y); //Stores Pinks y position in this variable

tempPinkPosition = new Vector2(PinkPositionX, PinkPositionY); //Stores Pinks coordinate in a new vector2 variable

float distance = GetDistanceApart(tempPinkPosition, targetTile); //Gets the distance apart between Pinks position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempPinkPosition.x + distance, tempPinkPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

modeChecker = 1;

return targetTile;

}

else //If the Pink Ghost is at its starting position then the following code is run

{

Vector2 tempBlinkyPosition = GameObject.Find("Red\_Ghost").transform.localPosition; //Create a temp variable for Blinky position

int blinkyPositionX = Mathf.RoundToInt(tempBlinkyPosition.x); //Stores Blinkys x position in this variable

int blinkyPositionY = Mathf.RoundToInt(tempBlinkyPosition.y); //Stores Blinkys y position in this variable

tempBlinkyPosition = new Vector2(blinkyPositionX, blinkyPositionY); //Stores Blinkys coordinate in a new vector2 variable

float distance = GetDistanceApart(tempBlinkyPosition, targetTile); //Gets the distance apart between blinkys position and the target tile

distance \*= 2; //Multiplies the distance by 2 every time it changes so the Blue ghost is always two tiles in from of PacMan

targetTile = new Vector2(tempBlinkyPosition.x + distance, tempBlinkyPosition.y + distance); //The target tile is equal to Blinkys x and y position + the distance

modeChecker = 0;

return targetTile; //Outputs the Target Tile

}

}

Vector2 GetOrangeGhostTargetTile()

{

//Always tries to get in front of PacMan or run away from PacMan dependent on which mode the Blue Ghost is on

Vector2 temp = Vector2.zero; //Used to temporarly hold a value

Vector2 pacManPosition = pacMan.transform.localPosition; //Gets PacMans position

Vector2 pacManOrientation = pacMan.GetComponent<PacMan>().orientation; //Stores PacMans Orientation

if (modeChecker == 0)

{

float distance = GetDistanceApart(transform.localPosition, pacManPosition); //Gets the distance from Ghost to PacMan

Vector2 targetTile = Vector2.zero; //Sets targetTile to (0,0)

if (distance > 8) //If Distance is greater than 8 then the following code is ran

{

targetTile = new Vector2(Mathf.RoundToInt(pacManPosition.x), Mathf.RoundToInt(pacManPosition.y));

temp = targetTile;

return targetTile;

}

else //If Distance is less than 8 then the following code is ran

{

targetTile = homeNode.transform.position; //Makes the TargetTile the Home Node

temp = targetTile;

return targetTile; //Function outputs targetTile

}

}

else if (modeChecker == 1)

{

int pacManPositionX = Mathf.RoundToInt(pacManPosition.x); //Rounds PacMans x coordinate to a whole number

int pacManPositionY = Mathf.RoundToInt(pacManPosition.y); //Rounds PacMans y coordinate to a whole number

Vector2 pacManTile = new Vector2(pacManPositionX, pacManPositionY);

Vector2 targetTile = pacManTile + (4 \* pacManOrientation); //Multiplies all of the vector2 by 4 so both x and y coordinate seperatly and then adds pacManTile to it so Pinky is always 4 tiles ahead of PacMan

temp = targetTile;

return targetTile; //Function outputs targetTile

}

return temp; //Outputs the value

}

Code for C# Menu:

using System;

using System;

using System.Diagnostics;

using System.Windows.Forms;

namespace Menu

{

public partial class frmCMenu : Form

{

public frmCMenu()

{

InitializeComponent();

}

private void butMyGame\_Click(object sender, EventArgs e) //When the MyGame Button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/Bling Bling Boy/Bling Bling Boy/Bling Bling Boy/bin/Debug/Bling Bling Boy.exe"); //Launches Bling Bling Boy (MyGame)

this.Close(); //Closes the Menu

}

private void butFlappyBird\_Click(object sender, EventArgs e) //When the FlappyBird Button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/Flappy Bird -DataBase/Flappy Bird Menu/Flappy Bird Menu/bin/Debug/Flappy Bird Menu.exe"); //Launches the Flappy Bird Menu

this.Close(); //Closes the Menu

}

private void butSpaceInvaders\_Click(object sender, EventArgs e) //When the SpaceInvaders Button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/Space Invader -DataBase/Menu/WindowsApp1/bin/Debug/WindowsApp1.exe"); //Launches the Space Invaders Menu

this.Close(); //Closes the Menu

}

private void butPong\_Click(object sender, EventArgs e) //When the Pong button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/Pong Remakes/Pong - Core DONE/Pong Menu/Pong Menu/WindowsApp1/bin/Debug/WindowsApp1.exe"); //Launches the Pong Menu

this.Close(); //Closes the Menu

}

private void butPacMan\_Click(object sender, EventArgs e) //When the PacMan button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/PacMan Menu/PacMan Menu/bin/Debug/PacMan Menu.exe"); //Launches the PaMan Menu

this.Close(); //Closes the Menu

}

private void butHighScore\_Click(object sender, EventArgs e) //When the HighScore Button is pressed the following code is ran

{

Process.Start("E:/College Subjects/Computer Science/Computer Science NEA/FINAL/High Score Screen/WindowsApp1/WindowsApp1/bin/Debug/WindowsApp1.exe"); //Launches the High Score Screen

this.Close(); //Closes the Menu

}

}

}

Code for Overall High Score Screen:

Imports System.IO

Public Class Form1

#Region "Variables"

'The Path of the Files are stored here, the variable name states which file it is referring to

Dim PacManUserNamePath As String = ("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\Username.txt")

Dim MyGameUsernamePath As String = ("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\Username.txt")

Dim SpaceInvadersUsernamePath As String = ("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\Username.txt")

Dim FlappyBirdUsernamePath As String = ("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\Username.txt")

'The Values below store all the usernames and scores from the username file into a array

Dim PacManUserNameArray() As String = File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\PacMan Menu\PacMan Menu\bin\Debug\JustUserName.txt")

Dim MyGameUserNameArray() As String = File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Bling Bling Boy\Bling Bling Boy\Bling Bling Boy\bin\Debug\JustUserName.txt")

Dim SpaceInvadersUserNameArray() As String = File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Space Invader -DataBase\Space Invader Main Game - Database\Space Invader Main Game\bin\Debug\JustUserName.txt")

Dim FlappyBirdUserNameArray() As String = File.ReadAllLines("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\Flappy Bird -DataBase\Flappy Bird - Core DONE\Flappy Bird\Flappy Bird\bin\Debug\JustUserName.txt")

'Below all the Lists are declared which are used to pass values through

Dim PacManList As New List(Of String)

Dim MyGameList As New List(Of String)

Dim SpaceInvadersList As New List(Of String)

Dim FlappyBirdList As New List(Of String)

Dim UserNamesList As New List(Of String)() 'List is used to store all the Usernames of the users

Dim UserNamesWithSumList As New List(Of String)() 'List is used to store all the Username of the users alongisde the sum of their scores from each game

Dim DistinctUserNamesWithSum As New List(Of String) 'Used to remove repeated data from the List Above

Dim UserNames() As String 'Stores all the Usernames in a array

Dim UserNamesWthSum() As String 'Stores all the the Username with their sum in a array

Dim PlainUserName As String 'Used to pass the UserName of the Username where it's required

Dim JustScoreList As New List(Of Integer)() 'List stores just the Scores of the Users

Dim JustScore() As Integer 'Array will store just the scores of the users

Dim UserInput As String 'Used to Pass User Inputs into the program

Dim ScoreOfUser As Integer 'Stores the Score of a single User

#End Region

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

File.WriteAllText("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\High Score Screen\WindowsApp1\WindowsApp1\bin\Debug\Username.txt", "") 'Clears the Username Text File when the Program is Launched

File.WriteAllText("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\High Score Screen\WindowsApp1\WindowsApp1\bin\Debug\Score.txt", "") 'Clears the Score Text File When the Program is Launched

'Code below calls sub routine by the same name and passes the UserName array through it

StoreUserName(PacManUserNameArray)

StoreUserName(MyGameUserNameArray)

StoreUserName(SpaceInvadersUserNameArray)

StoreUserName(FlappyBirdUserNameArray)

'Gets the Program to get the UserName of each player and get the Sum of each user with the GetSumOfSinglePlayer Sub Routine

For i = 0 To UserNames.Length - 1

PlainUserName = UserNames(i)

GetSumofSingleUser(PlainUserName)

Next

UserNamesWthSum = DistinctUserNamesWithSum.ToArray() 'Makes a New Array with the Distinct Values in only

WriteToTextFileUserNames() 'Calls Sub Routine by the same name here

GetValue() 'Calls Sub Routine by the same name here

'FOR Loop Below goes through each value in just score and writes it to the Score text file

For i = 0 To JustScore.Length - 1

Using Writer As New StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\High Score Screen\WindowsApp1\WindowsApp1\bin\Debug\Score.txt", True)

Writer.WriteLine(JustScore(i))

End Using

Next

MergeSort(JustScore) 'Sorts the Data with the Merge Sort Algorithm by passing through the JustScore array

WriteToLabels() 'Calls the SUb Routine by the same name here

End Sub

Private Sub butSearch\_Click(sender As Object, e As EventArgs) Handles butSearch.Click

BinarySearch() 'Launches the Binary Search Algorithm

End Sub

Private Sub butFind\_Click(sender As Object, e As EventArgs) Handles butFind.Click

UserInput = InputBox("What is Your UserName?") 'Asks the Player who wants to know their score, stores their username in UserInput

GetValueDependentonUser() 'Calls SUb Routine by the same name here

MessageBox.Show("Your Score is " & ScoreOfUser) 'Tells the player their score through a message box

End Sub

Private Sub GetSumofSingleUser(ByVal WrittenUserNames As String)

'Adds the data from each games Username text file to the correct list

PacManList.AddRange(IO.File.ReadAllLines(PacManUserNamePath))

MyGameList.AddRange(IO.File.ReadAllLines(MyGameUsernamePath))

SpaceInvadersList.AddRange(IO.File.ReadAllLines(SpaceInvadersUsernamePath))

FlappyBirdList.AddRange(IO.File.ReadLines(FlappyBirdUsernamePath))

'Provides Indexs to use when going through lists

Dim pacIndex As Integer = 0

Dim myGameIndex As Integer = 0

Dim spaceinvadersIndex As Integer = 0

Dim flappybirdIndex As Integer = 0

'Each FOR Loop for each game finds all the Username and Score from their Lists

For i As Integer = 0 To PacManList.Count - 1

If PacManList(i).StartsWith(WrittenUserNames) Then

pacIndex = i

End If

Next

For i As Integer = 0 To MyGameList.Count - 1

If MyGameList(i).StartsWith(WrittenUserNames) Then

myGameIndex = i

End If

Next

For i As Integer = 0 To SpaceInvadersList.Count - 1

If SpaceInvadersList(i).StartsWith((WrittenUserNames)) Then

spaceinvadersIndex = i

End If

Next

For i As Integer = 0 To FlappyBirdList.Count - 1

If FlappyBirdList(i).StartsWith((WrittenUserNames)) Then

flappybirdIndex = i

End If

Next

'Seperated the data into two parts when a comma is present

Dim pacSplit() As String = Split(PacManList(pacIndex), ",")

Dim mygameSplit() As String = Split(MyGameList(myGameIndex), ",")

Dim spaceinvadersSplit() As String = Split(SpaceInvadersList(spaceinvadersIndex), ",")

Dim flappybirdSplit() As String = Split(FlappyBirdList(flappybirdIndex), ",")

'Finds the Sum of all of the Scores from each game

Dim sum As Integer = Integer.Parse(Trim(pacSplit(1))) + Integer.Parse(Trim(mygameSplit(1))) + Integer.Parse(Trim(spaceinvadersSplit(1))) + Integer.Parse(Trim(flappybirdSplit(1)))

'Used to reduce bugs by removing any empty usernames

If WrittenUserNames <> "" Then 'If the Username field isn't empty then we write it otherwise we don't

Dim finaltext As String = (WrittenUserNames & ", " & sum.ToString) 'Stores the Username of the user alongside the sum of their scores

UserNamesWithSumList.Add(finaltext) 'Adds the Username and the sum of each user to the List

DistinctUserNamesWithSum = UserNamesWithSumList.Distinct().ToList 'Removes Repeated Entrys

End If

End Sub

Private Sub StoreUserName(ByVal arr() As String)

'Adds all the Username to the List below

For i = 0 To arr.Length - 1

UserNamesList.Add(arr(i))

Next

'Removes Duplicated values when moving data to array

UserNames = UserNamesList.Distinct().ToArray

End Sub

Private Sub WriteToTextFileUserNames()

'Writes all data to the UserName Text File from the DistinctUserNameWithSum List

For i = 0 To DistinctUserNamesWithSum.Count - 1

Using Writer As New StreamWriter("E:\College Subjects\Computer Science\Computer Science NEA\FINAL\High Score Screen\WindowsApp1\WindowsApp1\bin\Debug\Username.txt", True)

Writer.WriteLine(DistinctUserNamesWithSum(i))

End Using

Next

End Sub

Private Sub GetValue()

'Goes through each UserName and adds the sum of their scores to JustScore

For Each Data As String In UserNamesWthSum

For i = 0 To UserNames.Length - 1

PlainUserName = UserNames(i)

If Data.StartsWith(PlainUserName & ",") Then

JustScoreList.Add(Integer.Parse(Data.Split(","c)(1)))

End If

Next

Next

JustScore = JustScoreList.ToArray() 'Converts List to Array

End Sub

Private Function GetValueDependentonUser()

'If a User is looking for their score and they aren't on the top 5 they can look their score up here

For Each Data As String In UserNamesWthSum

If Data.StartsWith(UserInput & ",") Then 'Checks to see if data has the users provided username

ScoreOfUser = Integer.Parse(Data.Split(","c)(1)) 'Splits the Data so ScoreOfUser only contains the Score of the User

End If

Next

Return ScoreOfUser

End Function

Private Sub MergeSort(ByVal Array() As Integer)

'Calls the Merge Sort Algorithm with the correct values inside the parameters using the array that it MergeSort was called with

MergeSortAlgorithm(Array, 0, Array.Length - 1)

End Sub

Private Sub MergeSortAlgorithm(ByVal Array() As Integer, ByVal Low As Integer, ByVal High As Integer)

'Merge Sort Algorithm can be seen below

If Low >= High Then Return

Dim Length As Integer = High - Low + 1

Dim Middle As Integer = Math.Floor((Low + High) / 2)

MergeSortAlgorithm(Array, Low, Middle)

MergeSortAlgorithm(Array, Middle + 1, High)

Dim Temporary(Array.Length - 1) As Integer

For i As Integer = 0 To Length - 1

Temporary(i) = Array(Low + i)

Next

Dim Point1 As Integer = 0

Dim Point2 As Integer = Middle - Low + 1

For i As Integer = 0 To Length - 1

If Point2 <= High - Low Then

If Point1 <= Middle - Low Then

If Temporary(Point1) > Temporary(Point2) Then

Array(i + Low) = Temporary(Point2)

Point2 += 1

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Else

Array(i + Low) = Temporary(Point2)

Point2 += 1

End If

Else

Array(i + Low) = Temporary(Point1)

Point1 += 1

End If

Next

End Sub

Sub BinarySearch()

'Binary Search Algorithm can be Seen Below

Dim Result As Integer = -1

Dim Low As Integer = 0

Dim High As Integer = JustScore.Length - 1

Dim Middle As Integer

Dim UserInput As Integer

UserInput = InputBox("What is Your Score?")

While Low <= High And Result = -1

Middle = (Low + High) / 2

If UserInput = JustScore(Middle) Then

Result = Middle

ElseIf UserInput < JustScore(Middle) Then

High = Middle - 1

ElseIf UserInput > JustScore(Middle) Then

Low = Middle + 1

End If

End While

If Result < 0 Then

MessageBox.Show("Your Value Isn't in the List")

Else

Result = JustScore.Length - Result

MessageBox.Show("Your Value is at the " & Result & " Place in the List")

End If

End Sub

Private Sub WriteToLabels()

'Variables below are used to write the correct data to the labels on Form

Dim Position As Integer

Dim temp As String

'Finds the Username of the User with the highest score and stores it in temp then writes data to label

For Each Data As String In UserNamesWthSum

If Data.EndsWith(", " & JustScore.Last) Then

temp = Data.Split(","c)(0)

lblBest.Text = ("1:" & temp & "-" & JustScore.Last)

End If

'Finds the Username of the User with the second highest score and stores it in temp then writes data to label

Position = JustScore.Length - 2

If Data.EndsWith(", " & JustScore(Position)) Then

temp = Data.Split(","c)(0)

lblSecond.Text = ("2:" & temp & "-" & JustScore(Position))

End If

'Finds the Username of the User with the third highest score and stores it in temp then writes data to label

Position = JustScore.Length - 3

If Data.EndsWith(", " & JustScore(Position)) Then

temp = Data.Split(","c)(0)

lblThird.Text = ("3:" & temp & "-" & JustScore(Position))

End If

'Finds the Username of the User with the fourth highest score and stores it in temp then writes data to label

Position = JustScore.Length - 4

If Data.EndsWith(", " & JustScore(Position)) Then

temp = Data.Split(","c)(0)

lblFourth.Text = ("4:" & temp & "-" & JustScore(Position))

End If

'Finds the Username of the User with the fifth highest score and stores it in temp then writes data to label

Position = JustScore.Length - 5

If Data.EndsWith(", " & JustScore(Position)) Then

temp = Data.Split(","c)(0)

lblFifth.Text = ("5:" & temp & "-" & JustScore(Position))

End If

Next

End Sub

End Class