GEOMETRY HOP

AQA computer science project, June 2020

2019 - 2020
ASHCROFT TECHNOLOGY ACADEMY, 100 WEST HILL SW15 2UT
Alan Wu

Contents

Analysis 4

Background and investigation of the problem 4

Outline of Research 5

"Fireboy and Watergirl" Series 5

Super Mario Bros. (1985) 6

Terraria 7

Summary of elements to possibly include into my game 8

Details and explanation of my choices for my platform game 9

Consideration of potential users 10

Relevant Flowcharts for my project 11

Flowchart of game screens 11

Flowchart of in-game progression 12

Exchange of data with the database 14

Data required by the system 15

Data from external files 15

User Input 15

Key Objectives 16

Primary objectives 16

Secondary objectives 17

Design 18

Design Overview 18

How will the title and leader-board screen work? 18

How will the game communicate with the scores database? 18

How will the game interact with text files to load each level of my game? 18

How will the game's side scroll and jumping mechanics work? 19

Design of screens 20

Screen One – Title Screen 20

Screen Two – game screen 21

Screen Three – End game screen 22

Screen Four – Leader boards screen 23

Key Variables 24

Use of appropriate data structures and types 26

Jumping Mechanism 27

Side Scrolling Mechanism 29

- Respawning Mechanism 29
- Storage of data between different levels 29
- Game level loading mechanism 30
- Leaderboard screen mechanics 31
  - Reading data into the leaderboard screen 31
  - Writing data into the leaderboard screen 31

Implementation 32

- Development plan 32
- Prototype evaluation 34
  - Object orientated programming approach 37
  - Game pausing feature 41
  - Player collision detection 42
  - User input validation and data entry to the database (which stores all scores) 44
  - Presenting the top ten scores in the leaderboard screen 49

Testing 52

- Testing the leaderboards screen (LB Test No) 52
  - Test Results for testing the leaderboards screen (LB Test No) 53
  - Test Results for data validation 60
- Testing the main menu (MM Test No) 64
- Testing the game form (G.   Test No) 64

ADDLINKOFVIDEO 65

- Testing the game's file reading abilities (TR. Test No) 65
  - Test Results for the game's file reading ability 68
- Did I meet the objectives I set out? 78

Evaluation 81

- End User feedback 81
- Improvements to my design 83

Text files the game will read from 84

- Game File 1 – text file contents 84
- Game File 2 – text file contents 85
- Game File 3 – text file contents 86
- Game File 4 – text file contents 87
- Game File 5 – text file contents 88

Code 89

- Main Menu 89

cLevelSetUp **90**
Form1 (Game Form) **92**
Leaderboards **101**

Videos: Red are of my old prototype, Green are of the second iteration of my prototype.

- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)
- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

Analysis

Background and investigation of the problem

In 2018, the gaming industry generated just shy of $135bn, and in 2025 this figure is projected to be $300bn; these figures simply go to show how big and innovative the video game industry is. In the future I aspire to be a software developer, I feel that by developing my own indie-genre game I will be able to strengthen my capabilities for development, of which will help me to build foundations to fulfil my future aspirations.

An indie game is a game which is developed by an individual or a small team; indie titles are often much smaller and less popular than mainstream titles. Indie game developers also have less resources to work with as compared to a triple A game due to the lack of a high budget and funding from a game publishing firm such as "Tencent games" and "EA". However, the small development budget does not mean that indie games are not successful – there are multiple popular indie games (at the time of development) to date including Minecraft, Plants vs. Zombies and Garry's Mod.

I have had a fascination for Platform games - where the game features a two-dimensional environment where the character is controlled by the player by jumping and traversing solid platforms at different positions on the screen. My interest for platform games stems from the simplicity of them; this is proof due to how almost all platform games do not feature tutorials but yet are still enjoyed by large audiences (such as the Super Mario series).

As an indie developer, I will be mostly developing this game by myself. However, there are some limitations as to what I can do as I will have to balance the development of my game alongside all my other school work. As a result of this, there will be some features which I would consider to include into my game, but not include as my time is limited.

In my game, I would like to adopt a simple mouse and keyboard control scheme to allow for the player to control the movement of the character. The main objective for the player is to ideally finish the level they are playing on with the highest possible score that they can obtain. Scores for each level will be calculated through the time the player takes to complete a level as well as how many lives they have used to complete each level The lives remaining in my game will be deducted a number of ways; if the player hits an obstacle which kills the character; if the character falls off the map; and if the player runs out of time.

Besides the gameplay, I am going to incorporate a main menu of which will make my game easier to navigate for the user. In the main menu, I will also include an options tab which will allow for the player to change their control settings.

Outline of Research

My game will implement gameplay styles found in other popular platform games, of which may include some element of problem-solving. In order to make my game as great as it can be, I will analyse the factors of gameplay which I would like to use as inspiration, as well as the factors which make the given game successful in the Platform genre.

"Fireboy and Watergirl" Series

In "Fireboy and Watergirl" two (or one) user(s) use keyboard controls to work together to complete puzzles which are vital to reaching the end of a level; these puzzles include turning switches on and off, opening doors for one another and moving mirrors around in order to direct light into a sensor (of which will open a door). There are also numerous obstacles in "Fireboy and Watergirl", these include pools of liquids of which may or may not be accessible to each individual;

- Green pools – neither player can enter them
- Red pools – only the red character (FireBoy) can enter them
- Blue pools – only the Blue character (Watergirl) can enter them

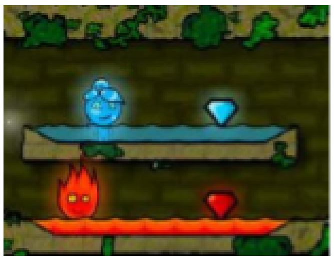


Figure SEQ Figure * ARABIC 1: Pools in which each player of FireBoy and WaterGirl can enter

On top of this, if one player dies, then they will automatically fail the level, of which then prompts them to restart or pick another level. The players are also graded on their performances by seeing how quickly they can complete a stage. However, in order for the players to complete a stage they need to complete to tasks:

- They must collect all of the gems (which will be either blue or red), the gems are specific for each player; only Fireboy can collect the red gems and only Watergirl can collect the blue gems
- Both players have to reach the end of the stage and wait for the doors to open before they can proceed to the next stage

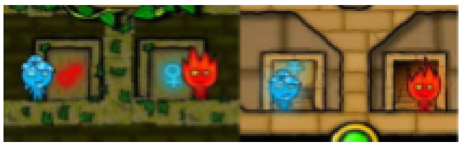


Figure SEQ Figure * ARABIC 2: Doors to end the stage will only open if the correct character is Infront of them

● Factors to use as inspiration in my project	Unsuitable elements of this game for my project
There are numerous types of obstacles that can be found in this game such as the different coloured pools of liquid which can/ cannot be accessed by each character	Fireboy and Watergirl requires two players in order to beat each stage; not suitable because I intend on making my game a single player game
Simple keyboard controls which make the game easy to play	Not the most suitable idea to create individually unique missions and puzzles with increasing difficulty for each stage as it would be too time consuming
Theming of the game (looks like lost ruins) which fits in with the overall aim of collecting all gems in order to complete a given level	
Difficulty of the different stages increases as you beat more levels, this engages the player(s) through challenging them	There are 20+ levels for every game in the Fireboy and Watergirl series, creating many unique stage environments would also be too time consuming
The player(s) have to complete a set of missions/puzzles before they can complete the level, they are on	

Super Mario Bros. (1985)

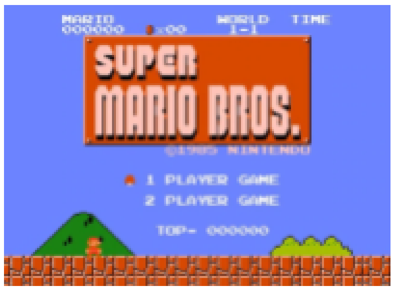


Figure SEQ Figure * ARABIC 3: Main menu of the 1985 Super Mario Bros

Super Mario Bros. was released in 1985 for Nintendo's game consoles meaning that the game utilised a physical game controller in order to control the character. In Super Mario Bros, the player plays the role of the main character Mario in order to play through each level to face the final boss, Bowser, in order to free and save Princess Toadstool. Super Mario Bros is a roll-playing, **side-scrolling** platform game; this means that the camera follows the characters movements following the speed and direction of the player character. However, in Super Mario Bros, the screen will follow the player character but only scroll forward, not backwards, so once something has passed off the back of the screen it can no longer be visited. The game features accessible goods for the player which improves the gameplay, these include:

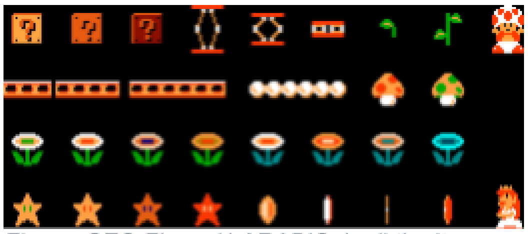


Figure SEQ Figure * ARABIC 4: all the items which can be found in Super Mario Bros

- Coins – these increase the score of the player
- Special blocks with question marks on them which release more coins or special items when hit by the player
- Invisible blocks which contain rare items
- Items that can be found in the game help to make the game easier to play for the player by giving them more powers or more lives.

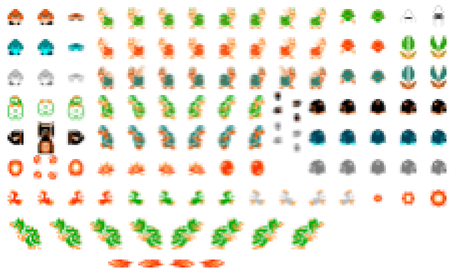


Figure SEQ Figure * ARABIC 5: all enemies of Super Mario Bros

There is are also enemies which can be killed by Mario in numerous ways, doing this increase the score the player gets and may even also drop special items and/or coins. If Mario however, is attacked by one of the enemies, he takes damage and may lose a life. The player is given 3 lives to start with and can collect more lives as they continue through the game's many stages. Mario can also lose a life if the player runs out of time or if he falls into a bottomless pit. Once Mario runs out of lives, the game ends and the player starts all over from the start again.

Factors to use as inspiration in my project	Unsuitable elements of this game for my project
Main menu feature makes the game nicely organised	Game has two player option which would not suitable because I intend on making my game a single player game
A lot of types of items available to use in the game as well as the use of enemies to enhance the gameplay of my game	Many levels/stages found in the Mario game, creating many unique stage environments would also be too time consuming
Ability for player to role play as Mario, some people may like role playing – could entice the player into the game	There is a very wide variety of interactable items in Super Mario Bros; it would not be a suitable idea to include all of them in the game as it would be far too time consuming
Game contains a storyline and gives the player a reason to go and save Princess Toadstool	Creating a storyline would take too much time and may not even be interesting to the player if there are not enough stages/levels to contain the storyline

The use of lives which limits the number of tries in which a player will have to complete the game	
Use of a Score system to track player’s performance in the game; the highest score is shown in the main menu, as can be seen in Figure 3.	
Use of a flagpole to end each stage; the higher you hit the flagpole the more points you get added to your score.	

Terraria

Terraria is a side-scrolling platform sandbox game which revolves around exploration, building, crafting, combat, and mining. The game also features pixel heavy graphics, similarly to Super Mario Bros (1985). The player in Terraria has to look for resources in order to make tools/items which can give them upper edges. The game hosts NPCs which can also give the player an upper edge as it opens room for trades which in return give the player virtual items which they can use to do as they wish. When creating a sandbox game in Terraria, the player can also choose the difficulty at which they would like to play at. Although Terraria is considered to be a sandbox title, there are also numerous enemies found in the game of which include bosses – when a boss is defeated, the player unlocks achievements which show the in-game progression of the player themselves.

Factors to use as inspiration in my project	Unsuitable elements of this game for my project
Main menu feature makes the game nicely organised	Terraria is a sandbox game, programming all the items in a sandbox game would be far too time consuming and won’t fit into my ideas
A lot of types of items available to use in the game as well as the use of enemies to enhance the gameplay of my game	There is a very wide variety of interactable items in Terraria; it would not be a suitable idea to include all of them in the game as it would be far too time consuming
Terraria features a 4 directional (up, down, left and right) side-scrolling mechanism which follows the player in all 2D directions	“Crafting recipes” and resource gathering would take too long to program
An achievements tab is present in the game which can track the player’s progression in the game	There is only one “randomly-generated” world in Terraria and these worlds are very large in size. This would not be suitable as it will take too long to make one big world.
Ability to fight enemies using the player or virtual items which can be found or crafted	

Summary of elements to possibly include into my game

- Incorporation of obstacles would immerse the player to the game as they would need to think of possible ways to get around them, there is usually a slight sense of satisfaction when a player gets past an obstacle.

  However, although I plan to include this in my game, I will probably focus on only making one or two obstacles as my time to develop this program is limited. I have chosen the compromise to make one or two as I personally think it is enough to showcase the idea of implementing it into my game.

- Score system would make the game more competitive, making the player want to retry levels to get the highest possible score. All of this will make the game more enjoyable to the player.

- The game should have an overall general theme; this will give the game some context, by giving the game a theme the game also becomes more understandable to the character as their objectives in the game will make more sense.

  This will not be a primary focus however, this is simply due to how theming an entire map (let alone the entire game) seems like a time-consuming process, of which I may or may not have enough time to carry out.

- Implementation of smaller sub-missions for every stage/level in the game as it not only gives the player a way to increase their score but also to further immerse the player into the game.

  I may choose to find a compromise to this however as designing unique missions into every level does seem quite time-consuming. A compromise of this might be for the player to collect special items (such as coins), of which contribute to the in-game score.

- Simple controls for the movement of the character because I want my game to be easy to pick up and play.

  Ideally keyboard controls, i.e. WASD keys.

- In order to complete a stage, the player has to go through a door of some sort, or hit a flagpole (like in Mario) to complete the stage.

- I will also like to include a main menu which will make the game more navigable to the player, the main menu will feature a “play” button which launches the stages, an “options” button which allows for the player to make changes to the game as well as a “leader board” button which will show the top highest scores of the game.

- Use of virtual items which can be found in interactable objects, these will give the player an upper edge and will make the game more competitive to play

- Side scrolling mechanism for the game to follow, similar to all the games used in my research

Details and explanation of my choices for my platform game

Character operation

The character will be able to move in 3 directions to navigate the level the player is playing on; the upward directions will be controlled by the player's jumps but the downwards direction will be controlled by the game through the implementation of a gravity mechanism and the "forward" direction (right) will be controlled by the program itself.

I have opted for keyboard only controls, and my choice of input keys are "p", "c" and the spacebar. The spacebar will be used to make the player's model jump, and the "p" key will be used to pause the game while the "c" key will be used to continue/un-pause the game. I have chosen to only use one key to control the character to make the game's controls as simple as possible to pick up.

Character properties

The player will be given a finite number of lives; if they use up all lives the game ends. This makes it so that the player has to think about what they are doing when they play so that they don't lose lives over small things, this makes the game more immersive as it causes the player to concentrate

I intend to make the character's player model as simple as possible due to my time constraints, as a result I have opted to make the player's model a simple red square.

Environment Properties

There will be "gravity" in the game which means that the character will free fall if there is nothing beneath them; the player will also not be able to jump unless there is solid ground beneath the character. This makes the game more realistic; everything that goes up falls, and jumping mid-air (double-jumping) is also prevented as a result.

There will be raised platforms which the player can jump up to, this adds increased difficulty to the game and acts as a good way to help the character gain altitude. The levels will also be designed to force the user/player to have to jump to navigate the level; if the user collides into the side of a platform, I intend on deducing a life and then respawning the character to the start.

My game will also feature bottomless pits* and obstacles* which will cause the player to respawn as well as causing them to lose a life.

*falling into a pit will cause death, this feature has been taken from the Super Mario Bros. game series

**obstacles such as spikes which will cause the player to die if they touch them

Consideration of potential users

Who is the program designed for?

After presenting my ideas to members of friends and family, I was given the impression that my game would be enjoyed by people of all ages, however I do feel that a more prevalent interest was evident in people aged between 5 and 30. As a result of the stronger interest within this age range of people, I have decided that I should theme the game in a way which would allow for anyone in this age range to enjoy.

Which platform is my game designed for?

My game can theoretically be played on any platform which supports games since there is really only one input key needed to control the character's movements, this means that the game can be controlled using a keyboard on computer, gamepad on console or even a touchscreen on a mobile device. Due to my limited ability to develop on game engines and other devices, I have decided to rule out for the development of my program for consoles and mobile devices. This leaves me with the only option to develop my game for computers. However, computers raise an issue of their own: of those people I presented my ideas to some had computers running Mac OS and Windows. As a result, this and my limited knowledge of programming on Mac OS, I have decided to develop my game for Windows using Visual Basic.

What is the most suitable theming for the users?

As a result of my observations from presenting my ideas, I have opted to go for a simple and plain theme. My choice not only saves me time during development, but it also means that the game would be suitable to play for my target demographic of people aged 5 to 30.

Relevant Flowcharts for my project

In the following pages a rough overview of the game screens, in-game progression as well as the exchange of data with a database will be shown in the form of flowcharts.

Flowchart of game screens

This is how I envision each of the game's screens can be accessed and how the user can navigate my program:

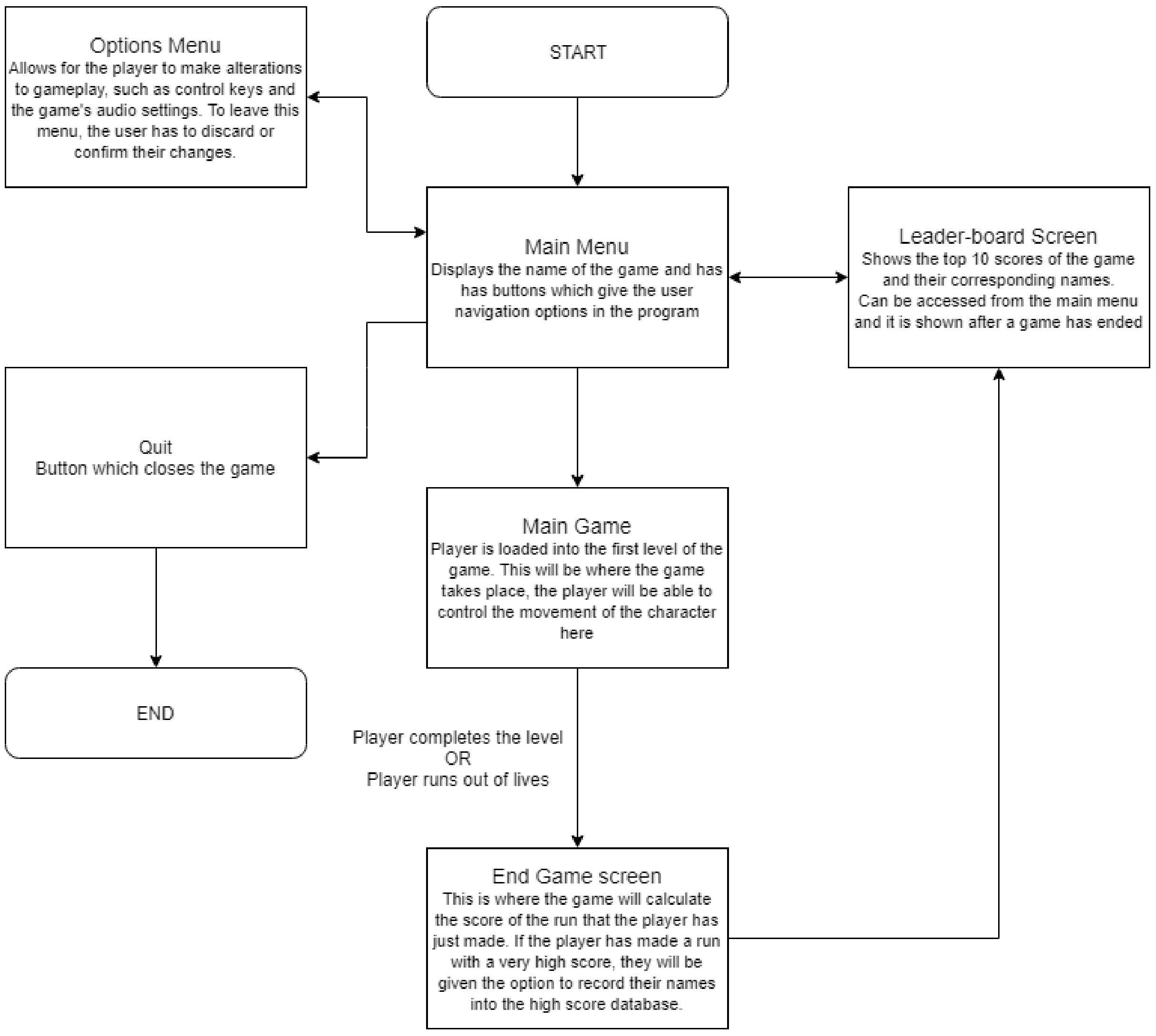
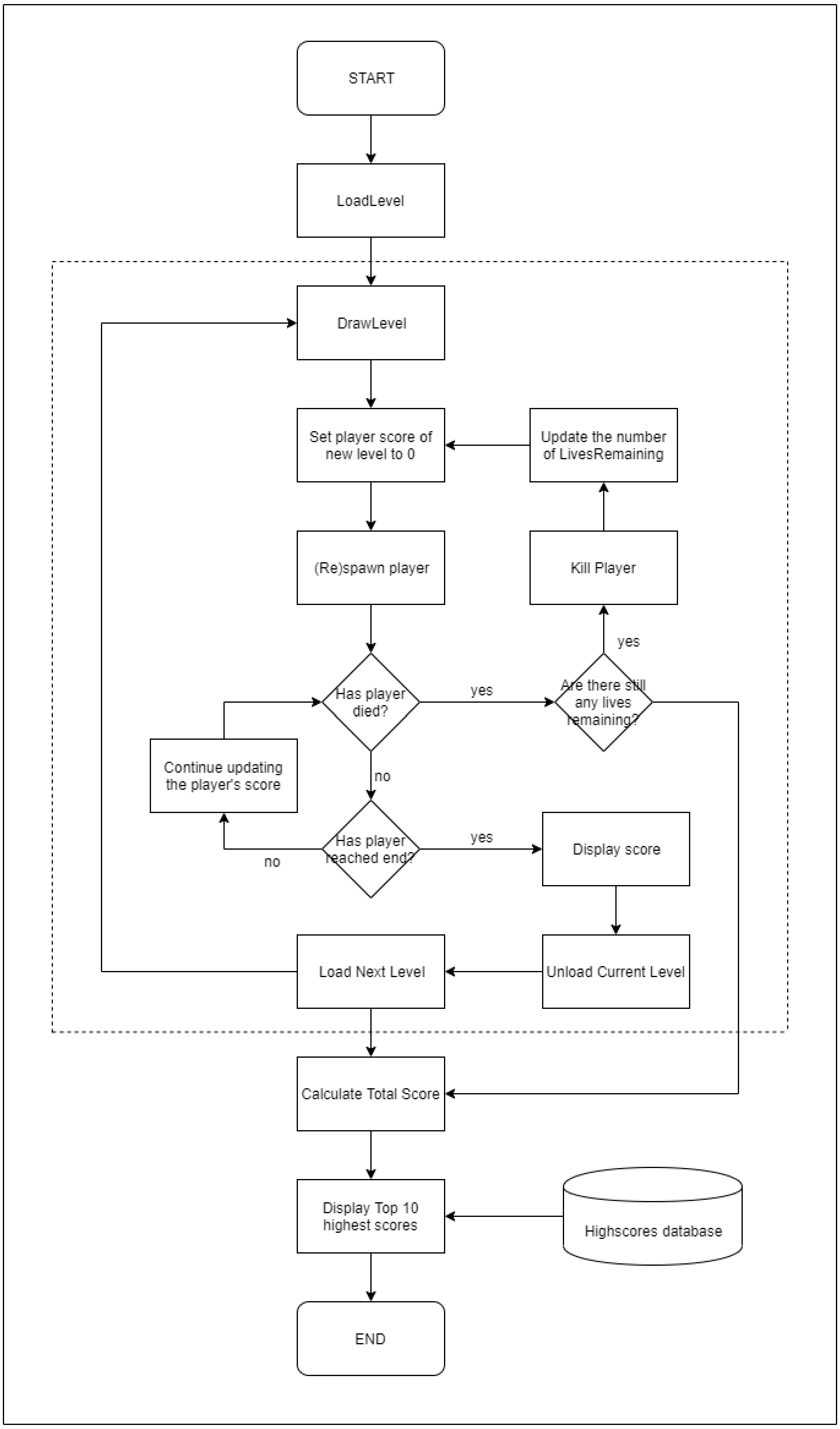

Options Menu
Allows for the player to make alterations to gameplay, such as control keys and the game's audio settings. To leave this menu, the user has to discard or confirm their changes.
START
Main Menu
Displays the name of the game and has has buttons which give the user navigation options in the program
Leader-board Screen
Shows the top 10 scores of the game and their corresponding names. Can be accessed from the main menu and it is shown after a game has ended
Quit
Button which closes the game
Main Game
Player is loaded into the first level of the game. This will be where the game takes place, the player will be able to control the movement of the character here
END
Player completes the level
OR
Player runs out of lives
End Game screen
This is where the game will calculate the score of the run that the player has just made. If the player has made a run with a very high score, they will be given the option to record their names into the high score database.

Figure 6: Labelled flowchart of the game screens and how they can be accessed by the user

Flowchart of in-game progression

The flowchart below shows how I envision the in-game progression of my game;



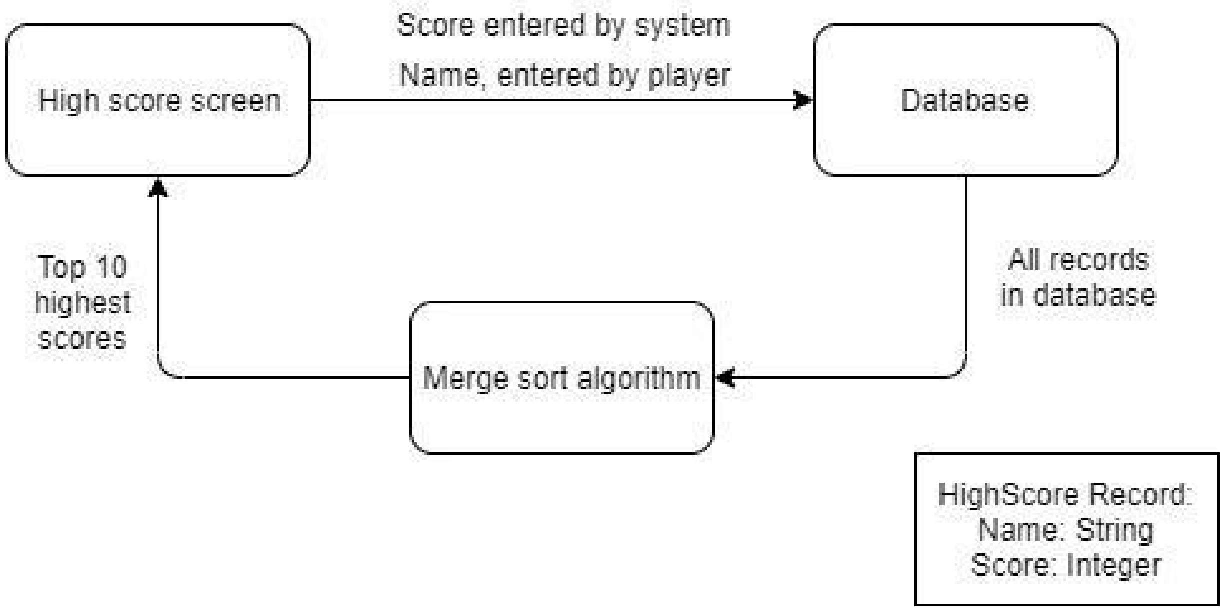
The flowchart showing in-game progression above shows how I intend for my game to operate. To begin with, the user would enter the game from the main menu and this would then result in the program to load the first level from a text file. Once this is done, the program will set the player's score of the current level to 0, this is done so that the score of a new level only counts upwards from 0; this idea was inspired from the score system in the Super Mario games. Simultaneously, the user's player particle is spawned and the user is given control of the particle. While the user plays through the level, the program will constantly check to see if the player has died; this will be done through the use of timers and collision detection algorithms (which will see if one vb.net control's boundaries have intersected another control's boundaries).

In the case where the player hits something which causes death, the program will then check to see if there are any spare lives remaining for the user; if there is, the player is respawned at the beginning of the stage and the number of lives they have remaining decreases by 1; if not, the program takes the user to the end-screen see their score of their most recent attempt of the game. The program will also check to see if the player has reached the end of the level – this will be achieved through the use of collision detection algorithms (as explained in the above paragraph). Once the player reaches the end of the level, there are two possible outcomes:

1. In the case where there is another level following the one the player is currently on, the program will load the next level, and then redo the entire process described above. If there isn't a level following the one the user is currently on, the user is taken to the end screen of all levels combined and their total score is calculated. If the user's score happens to fall in the top 10 highest, they will be given the opportunity to enter a name to correspond to their score – this data will be then stored inside a database.
2. If there isn't a level following the one the user is currently on, the user is taken to the end screen of all levels combined and their total score is calculated. If the user's score happens to fall in the top 10 highest, they will be given the opportunity to enter a name to correspond to their score – this data will be then stored inside a database.
    i. The records will be stored from highest score to lowest score; this will be done by having my program re-order all records in the database whenever a new record is added. This is done so that the records can be displayed quickly in order from highest to lowest score.

If the player completes the game, and has already entered their record to the database, I intend for the program to read and display from the database the top ten scores and their respective players on a final screen.

Exchange of data with the database



This is a rough overview of how I will be storing and retrieving data from the database.

Data required by the system

This is a summary of the data which the program will be using to perform the necessary operations to provide the user with gameplay.

Data from external files

I envision for my program to be use a number of text files to store the details of each individual level. These text files will be read by the game before the player is able to play, and then they will be drawn immediately after the text file has been read completely. My intentions are to store individual levels in their own individual text files - this is done so that I could have an easier time creating the levels themselves. Doing this also makes the program much easier to adapt and change as changing one text file means that you will only be affecting one level, rather than all the levels I was planning to include.

My program will also be using a database to store the records of the games played previously. This will be done so that my program can display all scores in the game to the player after they have finished playing through each level. The database will be storing the “Score” of the player, a value which will be generated by the program itself, as well as the player’s name – the player’s name will be inputted through a textbox in the leader-board screen. These two bits of data will be what make up a record in my database. After a record is added to the scores database, a sorting algorithm will be used to sort the scores in descending order – this is done so that scores can be quickly fetched when the leader board screens come up. Doing this will make the scores display faster as there will be less processing power needed by the computer (compared to the alternative method of sorting the records every time the database holding the scores is read).

User Input

I have chosen to make my game as simple as possible to play and for this to happen I have chosen for there to only be one input which can control the player’s particle; the user will only be able to control the jumping of the player itself. The user will use the spacebar to make the player jump; I have chosen the space key since it is usually the default assigned key for jumping in other games on the PC platform. The user will also have the option to pause the game and this will be done through another keypress.

Additionally, I have included keys to pause and un-pause/resume/continue the game; “P” for pause and “C” for continue/un-pause. Pressing these keys will cause the game to freeze altogether, or cause it to continue to run again.

The user can also use the mouse to navigate the main menu and the leader-board screens. However, in the end game screen (leader-boards screen), the user has an option to enter some text to make up their record in the database storing player scores of which will obviously be inputted via keyboard.

Key Objectives

In this section I have listed some objectives that I would like to meet throughout the development of my game/project. These have been broken down into two groups: primary objectives and secondary objectives. Primary objectives will be vital to the functionality of my program and they should all ideally be met throughout the development of my project. On the other hand, secondary objectives are additional objectives that ae not vital to the functionality of my program which I came up with while developing my project, of which I could come back and add as features in my project.

Primary objectives

1. **The game should offer fun and immersive gameplay**
    a. Hitting certain objects will cause the player to respawn
        i. Objects such as the vertical edges of the floor, or possible obstacles should cause the player to respawn
    b. The player will be respawned if they fall through the gaps in the floor of each level
        i. My game's levels will have gaps in between the floors which the player has to jump over for them to reach the end
    c. My game will have multiple levels to play through of which will increase in difficulty
    d. There should be at least 3 levels for the user to play through
        i. Levels will automatically load after the completion of the previous one
        ii. Each subsequent level should have some degree of increased difficulty
        iii. Every level should be stored as text files

2. T**he Game will feature an automatic side scrolling mechanism**
    a. The speed of this mechanism should be steady and constant
    b. The mechanism will only cause objects on the screen to move one way
    c. The player's particle should remain in the same place on the screen when the level is being shifted by this mechanism

3. **The game will feature an element of gravity**
    a. Jumps should look realistic in the sense that the player particle will accelerate into the floor after rising for a short period of time
    b. The player particle should stop falling when it reaches the floor
    c. The player should start free falling if it slides off a platform

4. **The game will also feature a score system**
    a. The score will be displayed throughout the game play in a conveniently placed location so that the player can get live updates of their score
    b. The player's score will be calculated from the number of jumps that the player has made throughout their game – there could also be more factors which contribute to the score system (see secondary objective 1)

5. **The game should give the user some ways of making inputs to control the player particle**
    a. The player particle will only be affected by the user via the spacebar to jump
    b. The player can pause and continue the game using specified keys
        i. "P" should pause the game
        ii. "C" should resume/continue the game

6. **There should be an end screen after completing all levels**

    a. This screen shows the top ten scores made by other players, and possibly the most recent run if it is high enough
        i. They should be displayed in descending order of score
        ii. All scores will be read into the screen from a database
        iii. only the top ten are displayed on the leader board /end screen
    b. There should be a feature which shows the breakdown of how the player has achieved the score of the most recent run (i.e. through the number of jumps the player has made and others - see secondary objective 1)
    c. This screen also gives the user the option to enter a name to go with their score
        i. There should be a textbox prompt on the textbox where the user will enter their name; this prompt should automatically remove itself when the user starts typing their name
        ii. The player's name should be at least 3 characters long

7. **Create a main menu for easier navigation**
    a. Features a button to leave the game/to quit the application
    b. Features a button to start the game
        i. Will close the main menu and then open up the game form/screen where the game will take place
    c. Features a "leaderboards" button
        i. Will close the main menu
        ii. Opens the leaderboard/end screen
            1. When this opens, from the main menu, the user SHOULD NOT be able to add any additional records, they will only be able to view the top scores stored in the database
            2. Scores will be displayed in descending order of score
            3. All scores will be read into the screen from a database

Secondary objectives

1. **There could be opportunities for the player to improve their scores**
    a. Possible addition of special power ups that could reward the player with a score bonus
    b. Possible addition of time; a lower time taken to complete all levels could reward the player with a score bonus
    c. Score could be deducted for each time the player has to respawn
2. **Improvements to the in-game graphics could also be done to make the game more** immersive and enjoyable for the user

Design

Design Overview

In this section, I briefly explain how I envision my project to be developed as well as how I will go about meeting the objectives, I have given myself.

How will the title and leader-board screen work?

The title screen will be the first thing the user will see when they start my program. When the program is opened it will be opened to set height and width of which will be windowed – my choice in doing this was a result of Visual Basics limitations with drawing picture boxes, of which was needed for the levels of my game; Visual Basic is unable to scale things when the window is resized. Despite this however, the size of my screen is not too big and not too small. I have settled for a size of 800 by 500 (x, y).

The title screen will consist of buttons and a background image. The background image will feature the name of my game, "geometry hop", as well as any graphics that I would like to put on the title screen. There will be a choice of 3 buttons for the user to press, those being an option to play, quit and to view the in-game leader boards. The buttons will consist of text which states the button's purpose, but I intend for the backgrounds of the buttons to be transparent.

When the user is in the leader-boards screen, they will have the option to return to the main menu as well as the ability to scroll through all the records shown in the leader boards. In order to get records to display in the leader boards screen, I will be using Visual Basic's DataGrid view tool in conjunction with a database storing all the scores of all previous attempts of my game. Using a DataGrid view will allow the user to scroll through the top ten records read from the database. The records in the database will be stored in descending order of score, and as a result the leader boards will also display the scores in descending order.

How will the game communicate with the scores database?

I aim to keep all of the scores of previous attempts of my game in a database. The database should only store the name of the player as well as the score they achieved during their attempt. The score of the player will come from the game itself and the name of the player will be up to the user's discretion in the end-game screen. In the end-game screen the user will be given the ability to add a name to go with their score. The end-game screen should also be the only way which the user can add a record to the database holding scores within my program. Additionally, the user can only add one new record to the database for each time they complete the game, where the name they have entered is also at least 3 characters in length.

When a new record is added to the database, the program should ideally sort all new records so that the database is laid out in a way which stores all the records in descending order of scores. My game will use SQL to read from the scores database in the leader boards screen (a query similar to that shown below will be used). The Database holding the scores is called "TopScores".

```
SELECT * FROM TopScores
ORDERBY score ASC
```

My game will also be using SQL to write to the TopScores database in the end-game screen, a SQL query similar to the one shown below will be used

```
INSERT INTO TopScores VALUES (ID, Name, Score)
```

How will the game interact with text files to load each level of my game?

In my project I will be using text files to store the information needed to draw each level. My reasoning for doing this was so that my game could be easily expanded to hold more levels if I chose to expand it. My thinking behind this is that to increase the number of levels available, all the developer has to do is add an additional file location to the game.

All of the text files used for storing the details for the levels follow a specific layout; the text files only store details of picture boxes that need to be drawn to represent the level. I have formatted the text files such a way where **the X-coordinate, Y-coordinate, Width, Height, Name, Colour** and **Tag** are stored for every picture box that needs to be drawn;

- The X-coordinate and Y-coordinate are stored so that the game will know where to draw a given picture box
- The Width and Height are stored so that the game will know what size to draw a given picture box
- The name of each picture box is stored so that making the text files of the levels would be less confusing for myself – by adding names, I am then able to identify what picture box is being made/saved in the text files
- The colour of each picture box is stored so that the game will know what colour to draw a given picture box
- A tag is also given to each picture box in the text files – the purpose of the tag is to group picture boxes together. Doing this will allow for the game to know when to respawn the player, draw the next level, etc.

When a new level is drawn, I aim to use Object Orientated Programming to make each level a new instance of the "Levels" class. Every time a new instance of "Levels" is made, my game will follow an algorithm which allows it to make sense of the picture box properties I have stored in my text files, and hence draw them into the game itself. I also intend to include an element of exception handling in this algorithm in case one of the text files used to store the level details is written in the wrong format; in the case where an exception is caught, a suitable message will also be displayed too, of which will detail the problem to the user. The addition of exception handling will prevent my game from crashing when it encounters a file reading error.

How will the game's side scroll and jumping mechanics work?

In order to get realistic jumps of the player and a side scrolling mechanism for the game itself, I will be using a number of timers.

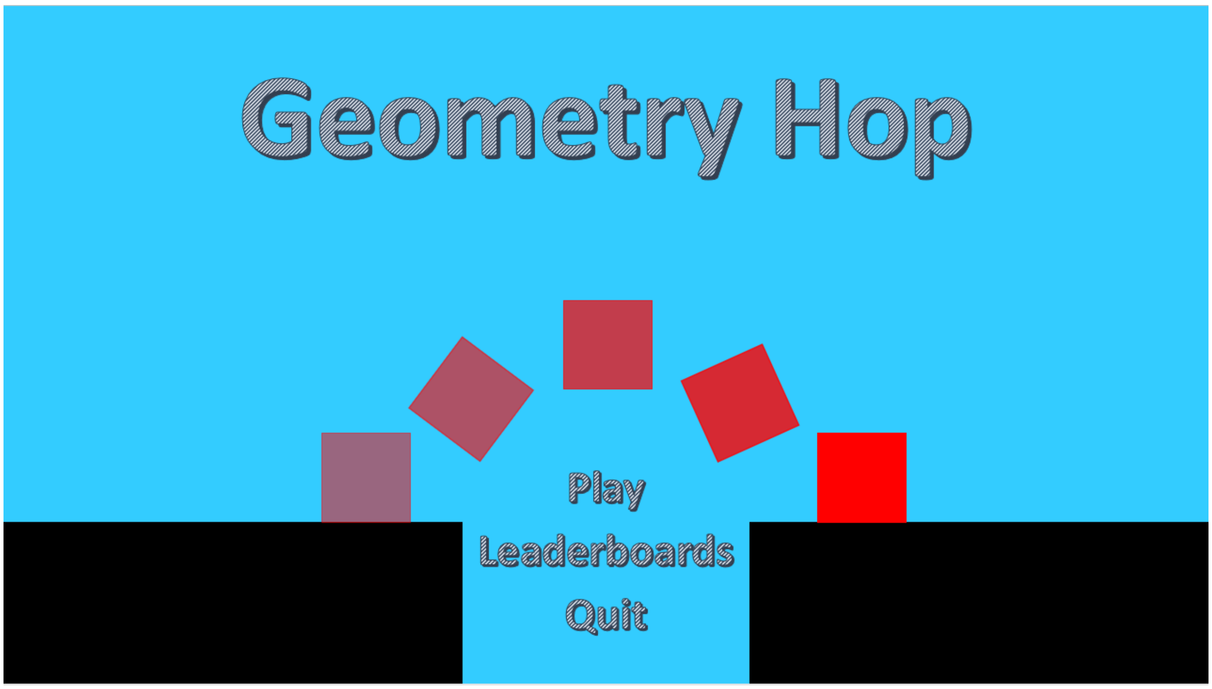
For the side scrolling mechanic, the program will shift every picture box left by a given amount of space every tick, this will create the impression that the player's particle is traversing the level. This will create an effect similar to that seen in the super Mario series when the player is moving through the level.

For the jumping mechanism, the player will be boosted upwards every time a jump is initiated by the user; this best described using the analogy of jumping – being "boosted upwards" would be similar to someone bending their knees and launching themselves into the air upon jumping. The player particle will then accelerate into the floor, the same way objects (in real life) will try to accelerate to terminal velocity when they are dropped from a given height. I feel that implementing these two key features in the jumps will make them seem more realistic.

Design of screens

I have made the following screens using PowerPoint, they only serve the purpose of providing a rough guideline to what I would like each screen to look like, as well as to give me an idea of what input and outputs each screen will have.

Screen One – Title Screen



This is the screen the user will see when they first start the game. The options can be navigated using the mouse; clicking on an option will cause it to launch that given option.

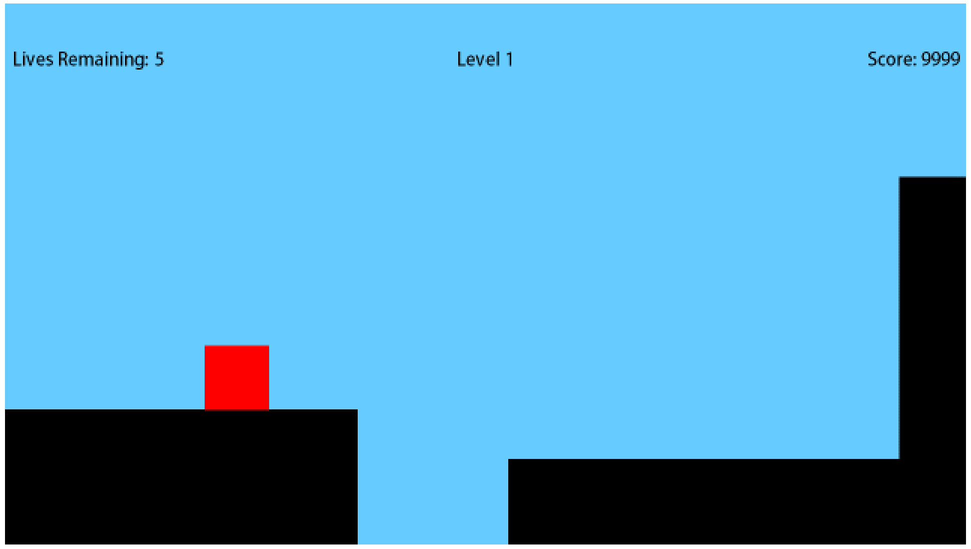
From top to bottom:
Option 1 – Starts the game
Option 2 – Opens the leader boards which displays all scores of all players
Option 3 – Exits the game

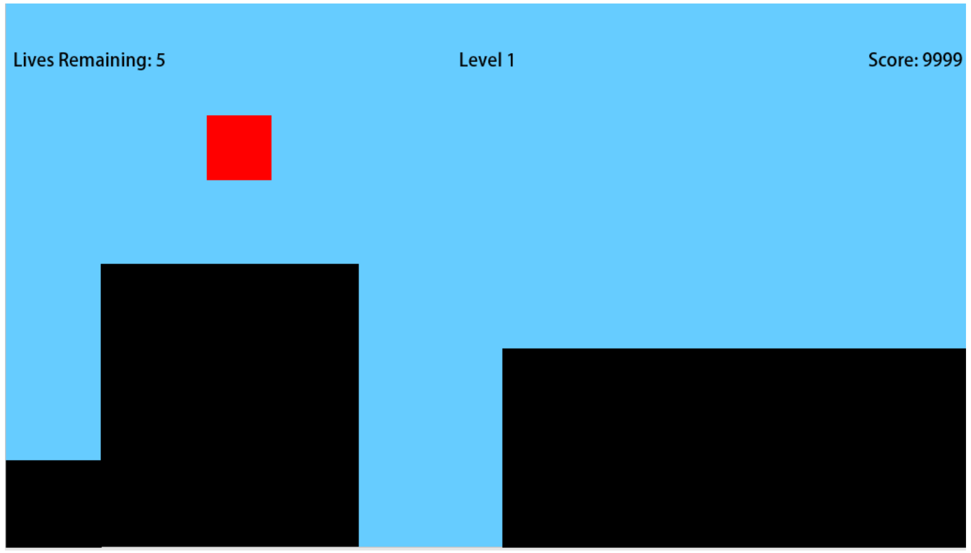
Data coming into the system:
The user's selection of the given options

Data which comes out of the system:
The option which the player has selected

Reasoning behind this design
I wanted to make my game simplistic and easily approachable, and by making the title screen as minimalist as possible, it would give users a settling welcome to start playing the game. The intention of the faded squares on the title screen also show that the game is to show movement and these are used to show that the game is fast-paced and dynamic. My choice of name for my game is also fitting for what is displayed in the title screen; a geometric shape (the red square) is seen to be hopping across the gap.

Screen Two – game screen





When the player is in the game, this will be the screen which they see. The level (represented by the black blocks) will scroll left and the player (represented by the red square) will remain locked at a given position in the x axis. The player will be able to be moved in the y direction through the use of jumps, and this is how the user will play through the level.

While the game is running, I intend for there to be a level counter so the user knows which level they are currently playing; this counter will also update itself automatically whenever the user completes a level. This counter is located in the top centre of the game form. Since I also want to include lives into my game, there will be a life counter which will decrease whenever the player dies. The score counter is also displayed for the user in the top section of the game form.

Data coming into the system:
The user's input:

- “P” – pauses the game
    - “C” – resumes the game

The Current level*

The user’s score*

The lives remaining for the user*

<u>Data that comes out of the system:</u>

The users score**/***

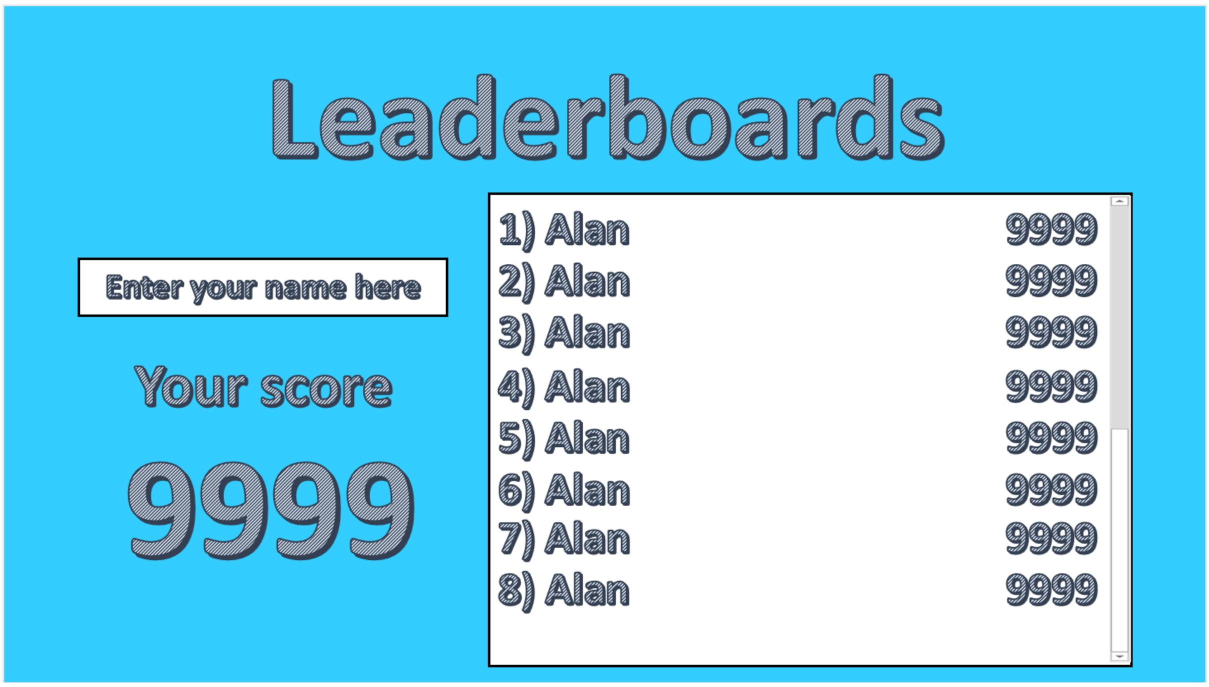
The Current Level**

The lives remaining for the user**

- Spacebar – causes the player to jump

*when the user causes the game to load the next level
**when a new level is loaded by the game
***when the leaderboard screen is about to open

Screen Three – End game screen



1) Alan	9999
2) Alan	9999
3) Alan	9999
4) Alan	9999
5) Alan	9999
6) Alan	9999
7) Alan	9999
8) Alan	9999

This is the screen the user will see when they finish playing through all the levels. They will be shown the score they have achieved as well as the option to submit a name to go with their score through the input field above where their score is shown. The leader-board on the right-hand side also has a scroll bar which allows the user to see all scores.

SQL will be used to insert/retrieve data to/from the database. All records in the database are sorted using a bubble sort algorithm when they are first added into the database; by sorting the data before it enters the database, displaying the scores in descending order can be done much faster as it will be a matter of reading the database and displaying the data suitably. Records are added to the database when the user submits a name to go with the score which they have achieved; after a name has been entered, the field to enter a name will disappear so that only one record can be added once the game is finished.

```
INSERT INTO TopScores VALUES (ID, Name, Score)
```

Above is the query I am thinking about using to insert data into the database

```
SELECT * FROM TopScores
ORDERBY score ASC
```

Above is the query I am thinking about using to retrieve data from the database

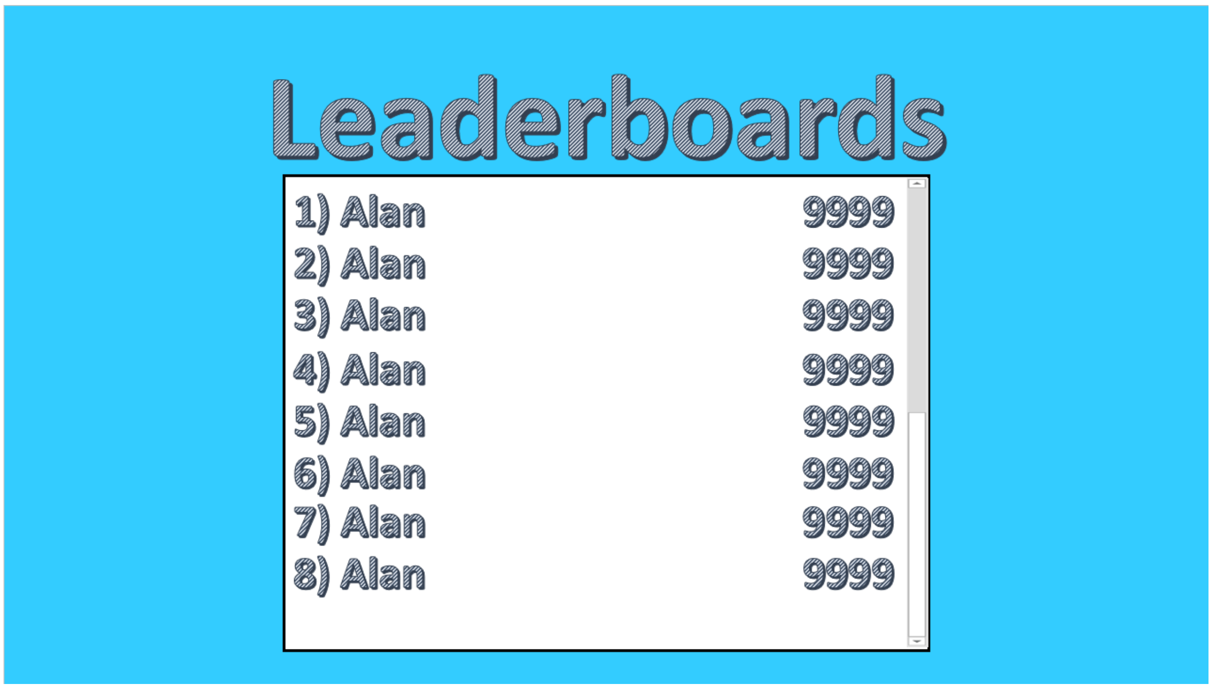
<u>Why this style was chosen:</u>
The theming behind the end-game screen should ideally have the same colours as the main menu, and the games levels if possible. As a result of this, I have chosen to use the same colour background and fonts that can be seen in the title screen for the game. Additionally, I wanted to make sure that the user can clearly see their score, hence explaining why the user score is in such a big font on the left-hand side.

Data coming into the system:
Records from the games database which stores all the scores from previous players, the name inputted by the user and the score which has been accumulated throughout the gameplay (from the game's system itself)

Data coming out of the system:

The displayed scores in from the database will be formatted in a way that is easy to read for the user.

Screen Four – Leader boards screen



This screen is the leader boards screen which can be accessed from the title screen. It functions similarly to the end game screen but does not feature the ability to add additional fields.

I want to try and make it so that the leader board screen displayed from the main menu is the same as the one displayed from the end of the game, my reasoning for this is so that I will have less to develop – this will ultimately save me the time I use for development. I aim to do this by using public methods which will call certain subroutines in the leader board's code which will either hide or show certain elements to the screen.

```
SELECT * FROM TopScores
ORDERBY score ASC
```

Above is the SQL query I am considering using to retrieve the data from the scores database to display in the leaderboards screen.

Key Variables

A number of Variables are needed by my system to store the data which will be used in each form and class to allow for the functionality of my game altogether. The following table outlines the key variables in the entirety of my program.

Form1 Script (this is the game screen)

Variable declaration	Description	Constraints	Accessed by
Const Integer GameSpeed	This constant defines how quickly my side scrolling mechanism will scroll at	Constant	Used by the TmrFowards Timer which moves all the objects on the form to simulate 2d traversal of the game's levels
Const Integer GravityStrength	This constant defines how strong the gravity of my game is – in other words the attraction between the player particle and the floor	Constant	Used by the TmrGravity Timer which moves is called every time the user wants to make a jump
Const Integer JumpingPower	This constant defines how high up the player particle can jump	Constant	Used by the windows form whenever the user presses the space key
Const Integer PointsPerJump	This constant defines how many points will affect the overall score of the run whenever the user jumps	Constant	Used by the windows form whenever the user presses the space key
Const String File1 File2 File3 File4 File5	These constants define where the game's levels can be found (as textfiles)	Constant	These constants are put into an array; this array gets called whenever a new level has to be drawn in the subroutine "Collisions"
Integer UpwardsSpeed	This variable defines how the player particle jumps in the air; it records the player's vertical speed	Integer which is greater than the value of the constant JumpingPower	Used by the windows form whenever the user presses the space key
Integer CurrentLevel	Keeps a count of which level the player is currently on, this value increments when the player reaches the end of a level.	Integer between 0 and the number of levels available to play	This is used whenever a new level has to be loaded in the subroutine "Collisions" – this value increments by 1 when this happens
Integer JumpsMade	Keeps a count of the number of jumps that the player has made. This value makes up part of the score which the user gets themselves at the end of the game.	Integer greater to or equal to 0	Used by the windows form whenever the user presses the space key Also used by the subroutine "DrawScore" to display the number of jumps the player has made during their run of the game
Boolean TouchingFloor	This Boolean value acts as a measure to prevent double jumping in the air. Implementation of this Boolean means that the user can only jump when they are touching the floor	True of False	Used by the windows form whenever the user presses the space key Also used by the sub "Collisions" to check if the user is indeed touching the floor.

LevelSetUp class

Variable declaration	Description	Constraints	Accessed by
List (of PictureBoxes) Objects	This list stores all the properties of the pictureboxes to draw from the given textfile. After each property has been stored, the object (picturebox) will be drawn onto the windows form.	List Of pictureboxes, where properties are stored as strings and integers	Used by the public Function "GetLevelInfo" which is used whenever a new level needs to be drawn.
Integer LineCount			

Leaderboards screen

Variable declaration	Description	Constraints	Accessed by
String Name	This string holds the name which the user types in that they want to go with their score. This is passed into the database via a subroutine, and then it is stored into the database. This variable can be derived from the GetName() function	String	Returned by the Private Function GetName to the Private sub InsertData
Const integer NameInDGV	Integer value which represents the column which the Name of users is stored in the data grid view control which holds the data read in from the game's database of scores	Constant	DrawLabels sub which helps to populate the top ten scores screen
Const integer ScoreInDGV	Integer value which represents the column which the Score of users is stored in the data grid view control which holds the data read in from the game's database of scores	Constant	DrawLabels sub which helps to populate the top ten scores screen
Const interger "xName" "xScore"	Both these constant integers are defined so that the Names of each record and scores of each record can be aligned in the form correctly	Constant	DrawLabels sub which helps to populate the top ten scores screen

Use of appropriate data structures and types

Integer: This data type is used to store numerical values where decimal precision is not required. I have used integers throughout my project to store constants as well as variables.

Boolean: This data type is used to store “True” or “False” values, this data type is used particularly when the program checks whether or not the player particle is touching the floor, so that the program can decide whether or not the player is able to make a jump (since jumping can only be done if the player is touching the floor)

Strings: This data type is used for the storage of text, this can either be to output to the user interface, or to input into the game’s system. Strings have been used particularly in the leader boards screen where the player is prompted (by a string) to enter a name (another string) to go with their score to add to the database.

Array: This is a static data structure which is used to store a defined amount of data so that the items in the array can be operated on/used. I have used arrays to store the game files that are to be used/run in my game.

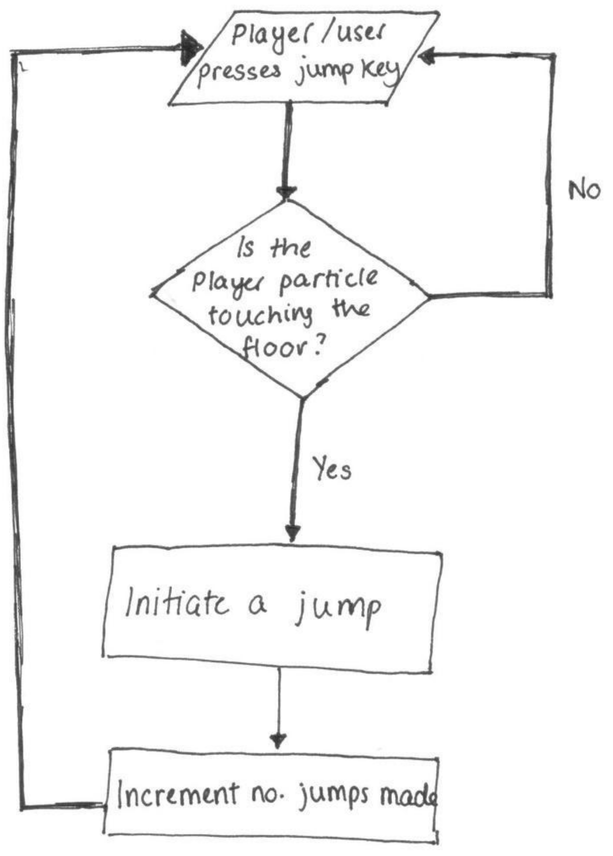
Lists: This is a dynamic data structure which is used to store an indefinite amount of data. I have used lists to store pictureboxes in my program where text files are read, this has been done because I did not know how big a text file storing a level’s details would be – hence making it indefinite.

Visual Basic controls/objects: A number of types have been provided by Visual Basic’s Toolbox, below I have detailed the types which can be found in my program

- **PictureBoxes:** These have been used to store the data needed to draw a given picturebox in my game, of which mostly make up the game’s levels.
- **Buttons:** These have been used to act as control that allows the user to directly communicate with the game and send the necessary commands to achieve a particular goal, i.e. to submit a field, or to quit the game.
- **TextBoxes:** Textboxes have been used to allow for the user to view or type in text
- **Labels:** Labels are generally used to display some informative text on the GUI which is not changed during runtime, Labels have notably been used to display the total number of jumps a player has made at the end of the game on the leader boards screen.
- **Timers:** This vb.net control has allowed me to repeatedly run the same lines of code at specific time intervals (known as ticks).

Jumping Mechanism

In my game I wanted to recreate a realistic jumping mechanism of which means that the player particle will slow down as the jump up into the air then accelerate into the floor until they hit the floor. If this is badly described, I aim for my jumping mechanism to look exactly like that found in the notorious Flappy Bird game. By doing this I would be able to add some difficulty to my game as the alternative option would make the game far too easy to complete. The alternative would have been for the player particle to jump at a constant vertical speed until it hits an upper limit, of which it would then fall back to the floor at a constant vertical speed.


Player/user presses jump key
Is the Player particle touching the floor?
No
Yes
Initiate a jump
Increment no. jumps made

My jumping mechanism will also keep a count of how many jumps a player has made; this will be done by incrementing a counter every time a successful jump is initiated, Figure 7 shows my ideas.

For the player to jump, I made it so that only the player particle would appear to move up and down on the game screen, rather than the level itself. This means that the platforms that the player traverses on are not affected at all by the jumping mechanism – only the player particle is moved by the system. As a measure to make the jumping realistic, I have also made it so that the player can only jump when they are in contact with the floor – this is because you can't jump mid-air in real life.

In order for the realistic jumping to work, initially I thought of using a timer to raise the player for a set amount of time, and then let the player free fall under gravity. Within the timer, which raised the player upwards, every tick of the timer would add height to the player. Additionally, I opted for there to be continuous gravity being applied to the player particle, which means that the player would continuously be accelerating into the floor. My thinking behind this was that after the player was raised in the jump, they would accelerate into the floor because of gravity; this thinking is shown in Figure 8. However, what actually happened with this design of the jumping mechanism is shown in Figure 9.

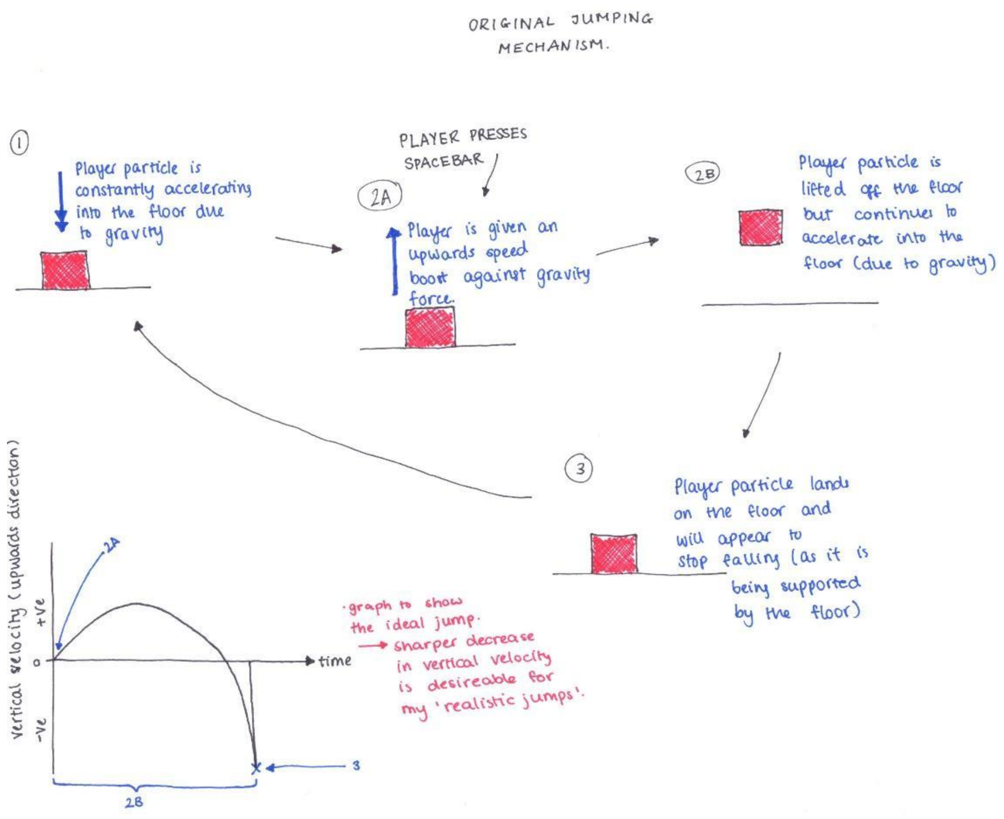

ORIGINAL JUMPING MECHANISM.
1
Player particle is constantly accelerating into the floor due to gravity
PLAYER PRESSES SPACEBAR
2A
Player is given an upwards speed boost against gravity force.
2B
Player particle is lifted off the floor but continues to accelerate into the floor (due to gravity)
3
Player particle lands on the floor and will appear to stop falling (as it is being supported by the floor)
vertical velocity (upwards direction)
+ve
0
-ve
time
2A
3
2B
·graph to show the ideal jump·
→ sharper decrease in vertical velocity is desireable for my 'realistic jumps'.

Figure 8: sketches of my initial design for the jumping mechanism

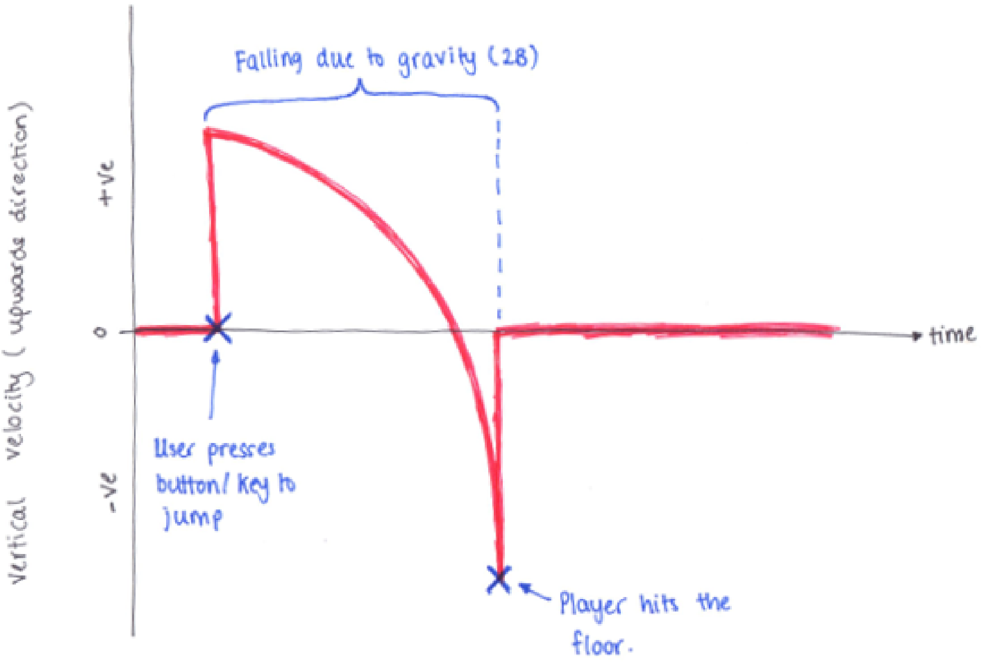


Figure 9: velocity-time graph which shows how the player particle reacted to my initial design

As a result of this, I had to re-evaluate my approach. In the end, I used a method whereby the player's vertical speed would be affected by gravity constant every tick (set to 1millisecond), and starting a jump would set the player's vertical speed to a given value (which has been set as a constant).

Additionally, for the jumping mechanism to work entirely I have used tags – a property of vb.net picturebox controls – to make it so that only the player particle moves vertically when a jump is initiated by the player; this has been implemented through the use of selection statements in the code for the timer used for gravity ("tmrGravity") in my program. Furthermore, I have also used a Boolean check value to see if the player is in contact with the floor, this was done so that the player can only initiate a jump when they are on the floor and not when they are mid-air.

Side Scrolling Mechanism

As described in my objectives, I aim to have a side scrolling mechanism in my game which will cause the level in my game to automatically move in one direction without the input of the player.

In order to implement this, I have thought of using timers and definite loops (for loops) to move every object on the screen one unit left in the x-axis to simulate the player traversing through a level. Every tick of the side scroll mechanism timer will cause my program to loop through every vb.net picturebox control in my program and move it one unit left in the x-axis.

Respawning Mechanism

Whenever the user falls off the map, or hits a vertical edge of a platform, it will cause them to die – hence this will result in them having to respawn at the start of the level.

In order to achieve this, I initially thought of making a counter which will reverse the changes made to the game's levels as described in the *Side Scrolling Mechanism* section. When a respawn is initiated this counter will be reset. Additionally, the height at which the player is at when they die may cause the player to be respawned into the floor (if they die too low) or too far up in the sky (if they die high up), as a result of this, whenever the player needs to be respawned, they will be respawned at a given height which is not too high up nor too low down.

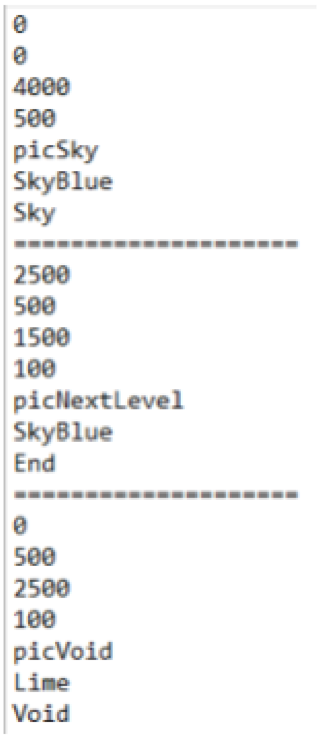
However, during the development of the system described above, I found that it made the game run increasingly slower each time the player respawned. As a result, I redesigned this mechanism entirely so that it removes the need of the distance counter. Additionally, with the old system, the player would be spawned at a pre-determined height, and this wasn't always the same as the initial height the player started at. The new system is a much simpler approach that does not cause performance issues to my program; it simply clears the console of all vb.net controls and then redraws them all.

Storage of data between different levels

As my game will feature a number of levels for the user to play through, there will be a need for the game to store details in between the different levels; for example, the current level which the user is playing through and the count of the number of jumps a player has made so far. As I have chosen for the game's levels to be instances of the class object, I feel that a good way of passing the data would be to write to an external file with this data before a new level is loaded, then read this data back in after the next level has been loaded. Although this may work, it does make my game more reliant on text files of which may fail; on top of this, my game would also be quite reliant on vb.net's timer controls which already take up quite a bit of processing power – the implementation of an external read/write file would simply add to the on-going processes for my game and as a result, could potentially cause performance issues to my game. As a compromise, I could pass data between the instances of each level using public getter methods and private setter methods in each class; this object-orientated approach would remove the need for additional external files, and hence allow data to be stored in between different levels.

Game level loading mechanism

As my game will feature multiple levels to play through for the user, I had thought up of a number of different ways of going about this in my game. In the end I settled for an Object-Orientated Programming approach – my decision to do this was heavily influenced by AQA's 2020 A2 pre-release. By using an OOP approach, I would make a class which would read the text files which contained the data needed to draw the game's levels; each time a new level was loaded, a new instance of the levels class would be made – and this would be what is displayed to the user when they are playing the game.



```
0
0
4000
500
picSky
SkyBlue
Sky
====================
2500
500
1500
100
picNextLevel
SkyBlue
End
====================
0
500
2500
100
picVoid
Lime
Void
```

*Figure SEQ Figure * ARABIC 10: example of how the details of pictureboxes will ideally be stored using textfiles*

While playing the game, the user should be updated with which level they are currently playing. The level counter should ideally automatically update itself with the correct information whenever the player loads into a new level.

My text files will be laid out in a way where it is easy for the developer of a level to see exactly what they are doing when creating a level, or maintaining it. I will be storing the data needed to recreate levels in a structured order as described; x-coordinate, y-coordinate, width of the picturebox, height of the picturebox, name of the picturebox, colour of the picturebox, followed by the tag of the picturebox (as well as a spacer – see Figure 10).

In my level set-up class, textfiles will be read from start to finish, and properties of the textbox will be assigned line by line using modular arithmetic (as there are only a given number of properties for each picturebox). After this, each picturebox and its respective properties ideally would be concatenated into a list to be drawn in the game. In the game screen itself, there will be a new instance of the levels class each time the user reaches the end of a given level.

During my development of this feature however, I have had to add a number of things to help improve it, for instance I have made a number of selection statements and the picturebox's tag properties which help to layer the pictureboxes in the game as sometimes the floor would be drawn behind the background. Additionally, I added some exception handling into the level drawing code so that the game wouldn't crash if it loaded a textfile written in the wrong way. When creating my levels however, I did find that I had to cut them down in length (size of x axis) as the longer levels tended to cause the game to run slowly – therefore as a compromise, I have made all the levels a maximum of 2500 units long.

Leaderboard screen mechanics

The leader board screen will allow for the user to not only view the top scores of players, but will also give the user an opportunity to add to the top scores after the player has completed all levels of my game.

Reading data into the leaderboard screen
I envision my leaderboard screen to have a table with the top scores of my game. The scores will be displayed in descending order, from best to worst. The fields displayed would be the score which the player has achieved as well as the name of the player who has achieved that score.

To achieve this, I will be using SQL and reading from a Microsoft access database. The database will only store the name of the player, score the player has achieved, as well as an automatically assigned ID. The data that has been read in will then be displayed in a vb.net Data grid view control, where the scores will be ordered by descending order via SQL. Initially I did want to implement some sort of sorting algorithm which would sort the records by score when the records are read into the game, however, after some thought it would be an unnecessary feature as SQL queries can assist me in doing this. On top of this, the implementation of a sorting algorithm will simply add to the processes which need to be run for my game, which in turn could cause the game to have performance issues.

Writing data into the leaderboard screen
When a user has finished the game, they will be given the opportunity to write a name to go with the score they have achieved, of which would then be submitted into the database. The name of the user will be inputted via a textbox and the score will come from the system itself – the player should not be able to change their score when they are writing to the leaderboard screen, however the user would be able to see what score they achieved.

A possible implementation of this would be through the use of SQL to write into the same Microsoft access database which stores the records which the game's leaderboard screen reads from. In order to store data into the database, I will also be using automatically assigned ID numbers to act as primary keys, this will make it so that my database is normalised of which will prevent any issues which may arise from modifying the contents of the table itself.

Implementation

Development plan

I have made a rough play for myself to follow when I develop my program. Any point which I feel that I have not fully committed to will be underlined.

1. Create the main menu & Leaderboard screen

Task	Done?
Create a suitable GUI for the leaderboard and main menu screens Buttons carry out functions they are intended for Both screens should feature similar designs to those I have made in my mock up designs	✔
Make a database which will be able to store the data from my game Database should store the Name (String), Score (integer) and ID (as a primary key) in a suitable table	✔
Leaderboard screen should be able to be opened in a number of ways by the program with varied outcomes Opening the leaderboard screen from the main menu should not allow for the user to add to the database Opening the leaderboard screen after the game has been complete should allow for the user to add to the database	✔

2. Creating the level player (part 1)

Task	Done?
Side scroll mechanism Should be automatic and should move all picturebox controls (but the player) in one direction	✔
Jumping mechanism Player should only be able to jump when they are touching the floor Jumps should look realistic Jumping should increment the Jump Counter	✔
Pausing and un-pausing mechanism "p" should pause the game entirely, the game should FREEZE "c" should resume the game as normal	✔
Respawning & collision mechanism Player should not sink through the floor Player should die if they fall off the map, or if they hit the edge of a floor segment Player should be able to hit a distinctive ending of each level which should load the next level	✔

Graphical representation and interface of the levels Levels should have backgrounds, foreground and the player should be visible to the player The Jump Counter should be visible for the player to see at all times	✔

3. Developing the level player (part 2)

Task	Done?
The program should be able to read the details of each level from text files and recreate them in the game The text files should store the data required in a suitable way which is easy for the developer to read	✔
There should be at least 3 levels for the user to play through There should therefore be at least 3 text files with different level designs Each level should have increased difficulty from the one before it	✔
Inclusion of a working level counter The counter should start at 1 The counter should increment whenever the next level is loaded	✔

4. Upgrading the level player (part 3)

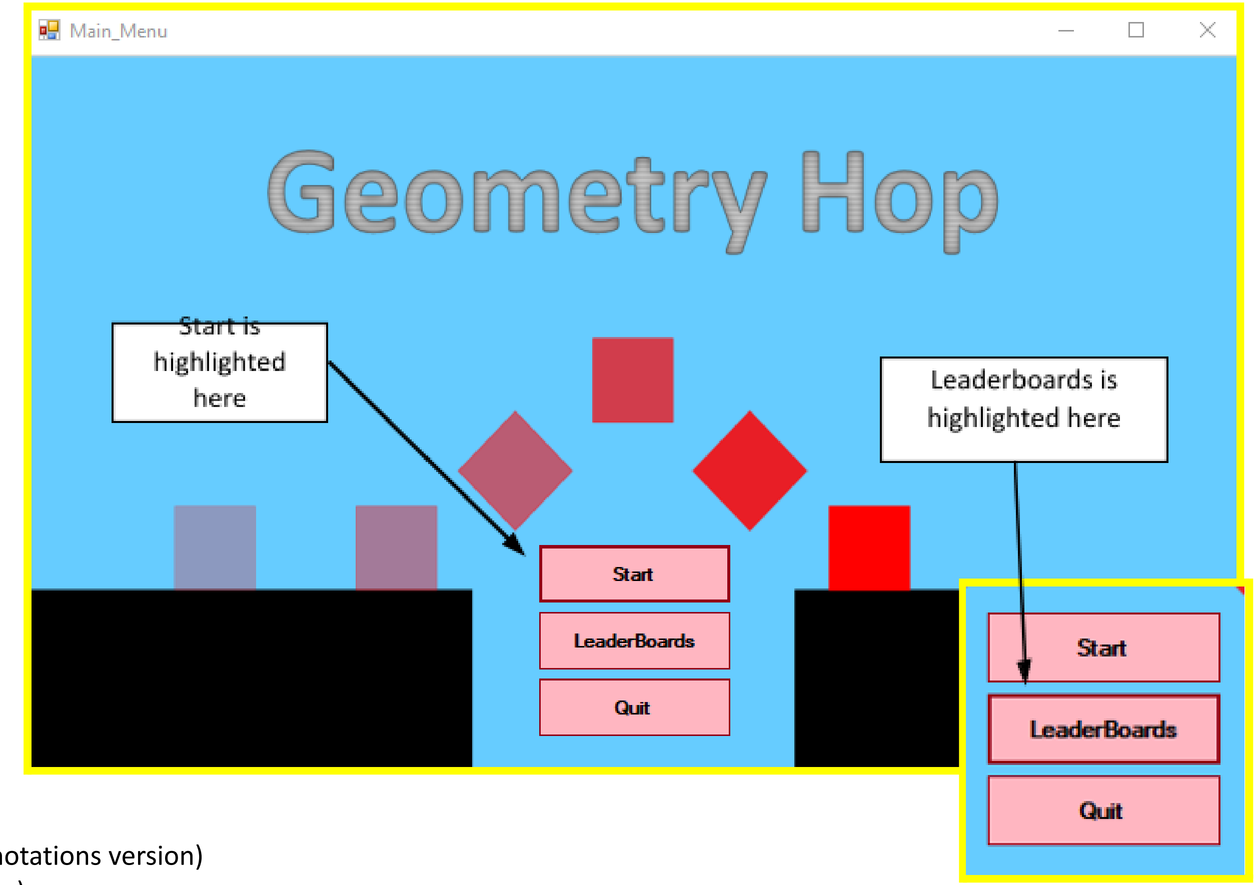
Task	Done?
Implementation of an Object-orientated approach for the game when it loads a new level This can be done by making a "levels" class where each individual level is an instance of the levels class	✔
Re-design of the level text files so that they are compatible with the new OOP system This should only be done if it is necessary but the OOP approach should still read files the same way it did before Ensure that the system works when the OOP approach is in use	✔
Implementation of exception handling for when the program reads in erroneous text files This should catch any exceptions thrown if the text file being read is stored/written in a way which the program cannot make sense of This should prevent the program from crashing if a problem is encountered	✔

5. Test that all features added work
   See the Testing section for this (page)

Prototype evaluation

Once the prototype of my game was developed, I decided that an evaluation of it should be carried out to see if my game was meeting the objectives which I gave myself. Upon doing this, I made changes to my program where the objectives were not met.

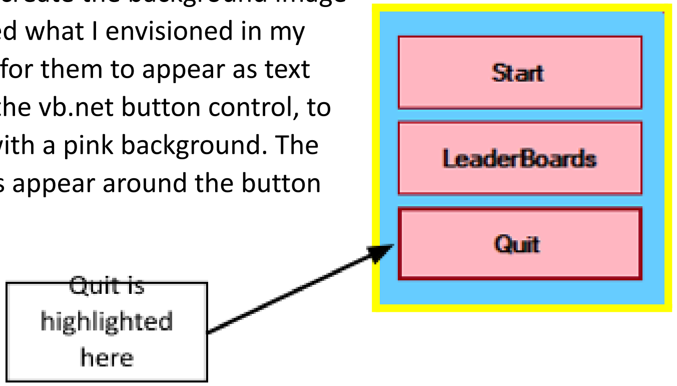
My prototype aims to meet all the requirements needed for the game to operate, such as the jumping mechanism, the respawning as well as the reading of text files for the game's levels. Despite this however, the graphical representation of my game's prototype could be improved as I feel it did not meet the expectations and ideas which I had designed in the design section.

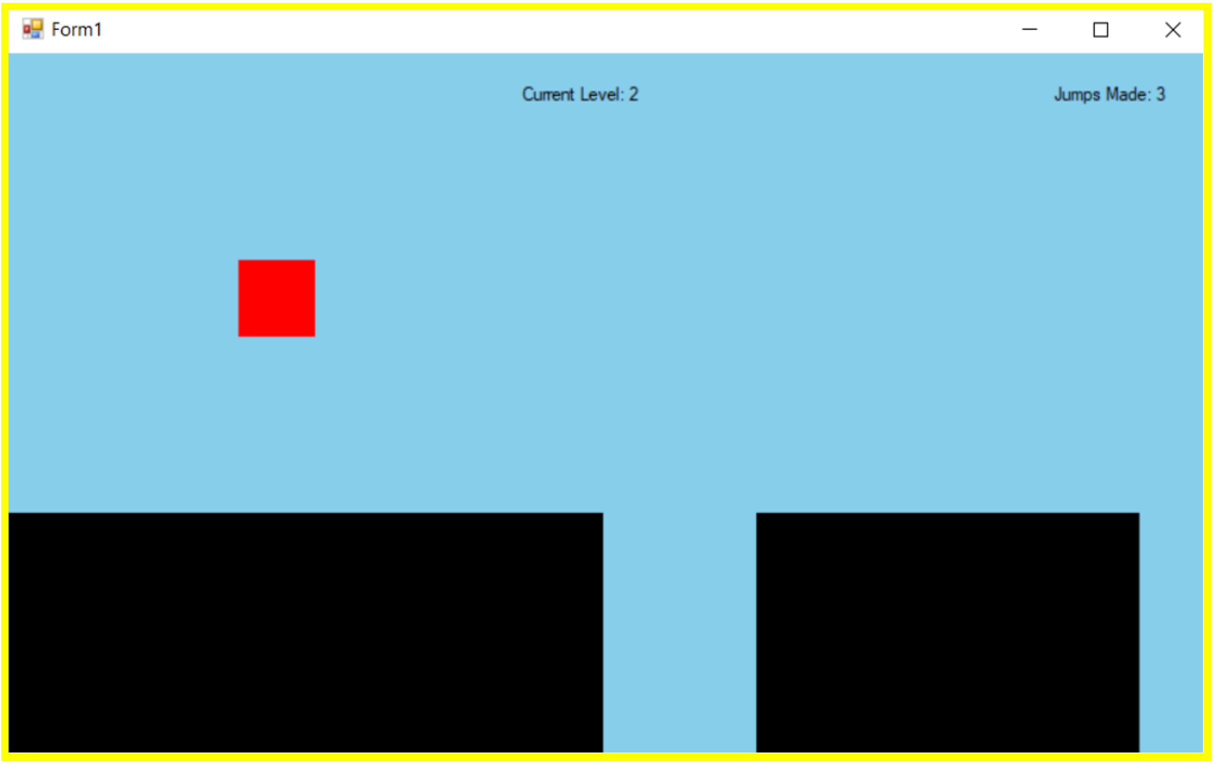


I have made some video recordings to showcase my prototype; Red are of my old prototype; Green are of the second iteration of my prototype.

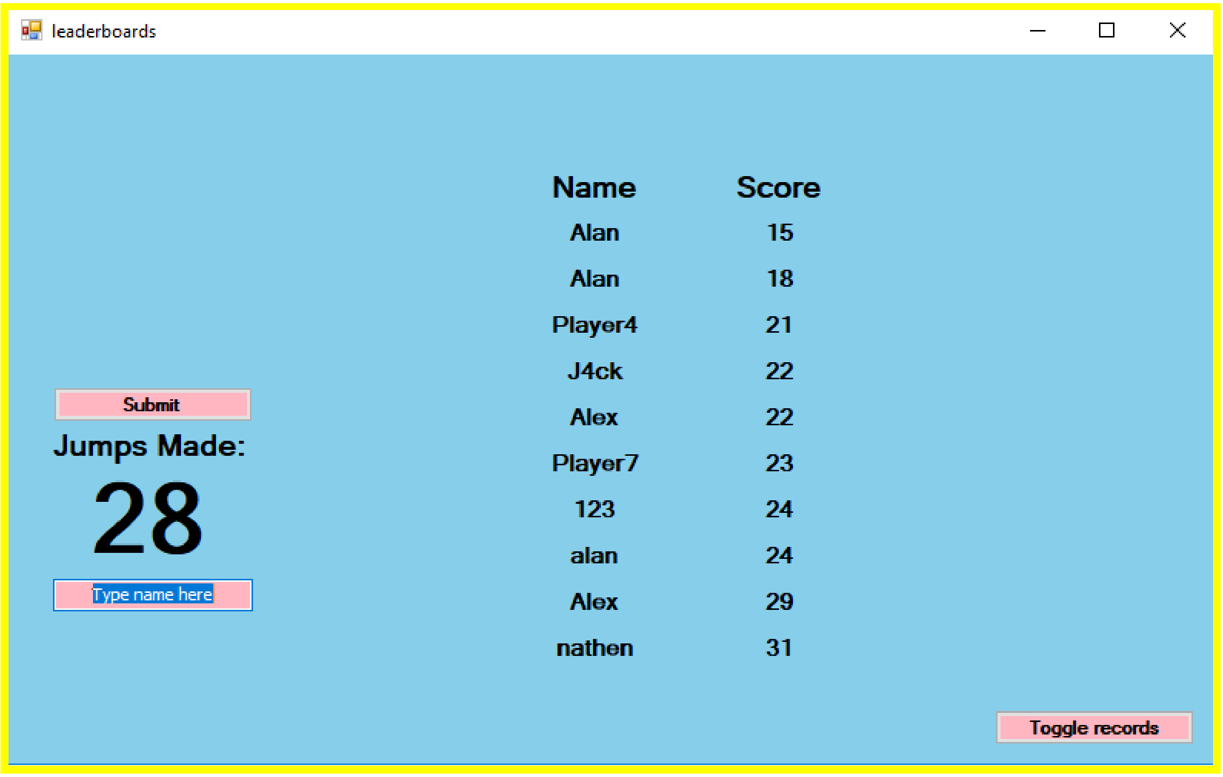
- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)
- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

My prototype's Main Menu screen does meet my desired design to some extent. I was able to recreate the background image which I wanted to use for the main menu, as well as lay out the buttons in a way which resembled what I envisioned in my design stage. One thing which I was not very happy about however were my buttons; I intended for them to appear as text with a transparent background but this did not work as I was not able to turn off the borders of the vb.net button control, to get around this issue, I instead changed the appearance of the buttons altogether to black text with a pink background. The buttons can be interacted with by using the arrow keys and the enter key, a faint red border does appear around the button which has been selected – this can be seen in the pictures attached.



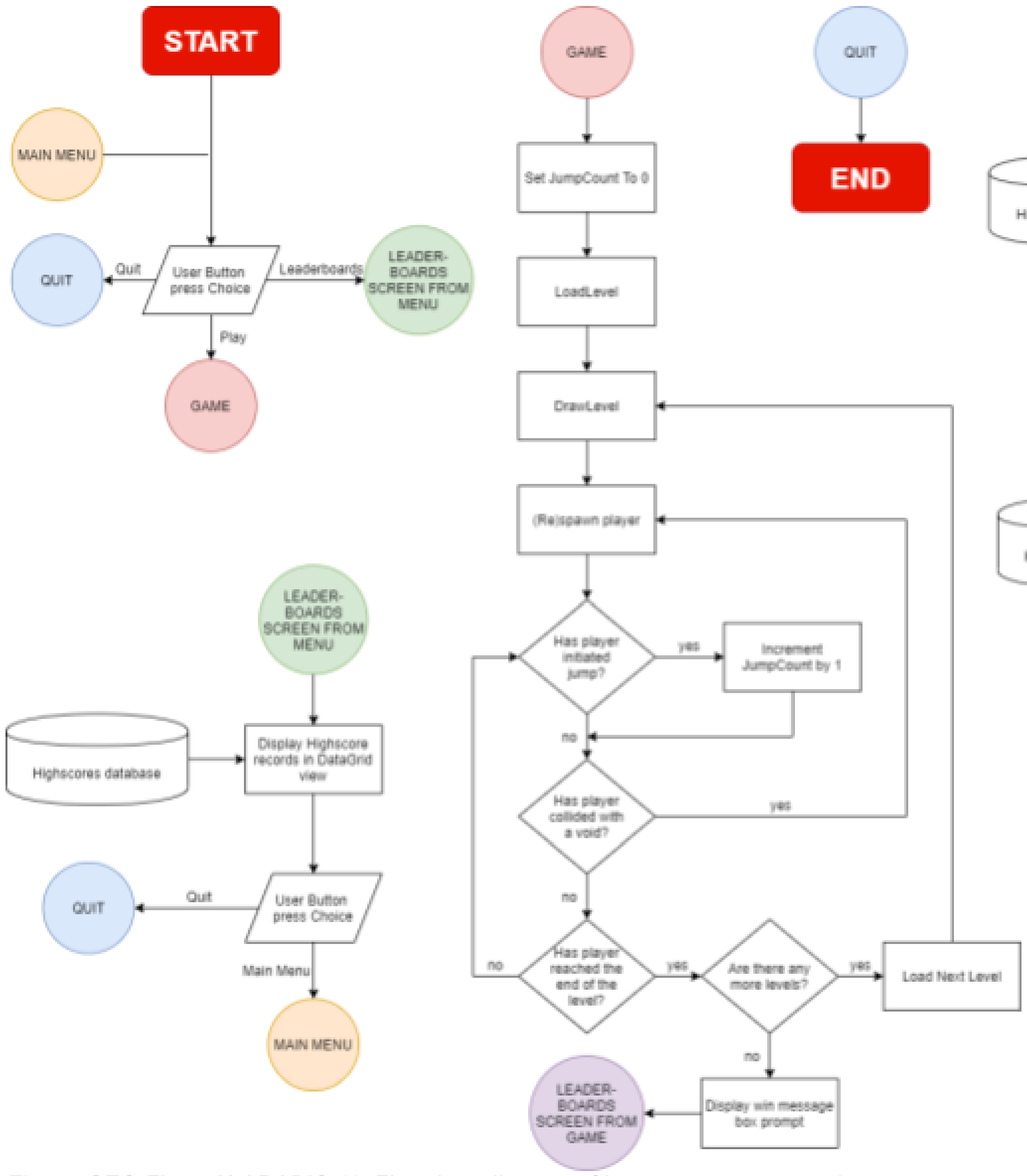


As intended in the design section of my write up, my game and it's counters work as expected. In the top right, there is a counter which increments every time a jump is made, and in the top centre of the form, there is a level counter which also increments whenever the next level is loaded.



Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

After the last level has been completed, the leaderboard screen will open and then


START
MAIN MENU
QUIT
Quit
User Button press Choice
Leaderboards
LEADER-BOARDS SCREEN FROM MENU
Play
GAME
GAME
Set JumpCount To 0
LoadLevel
DrawLevel
(Re)spawn player
Has player initiated jump?
yes
Increment JumpCount by 1
no
Has player collided with a void?
yes
no
Has player reached the end of the level?
no
yes
Are there any more levels?
yes
Load Next Level
no
Display win message box prompt
LEADER-BOARDS SCREEN FROM GAME
QUIT
END
LEADER-BOARDS SCREEN FROM MENU
Highscores database
Display Highscore records in DataGrid view
QUIT
Quit
User Button press Choice
Main Menu
MAIN MENU

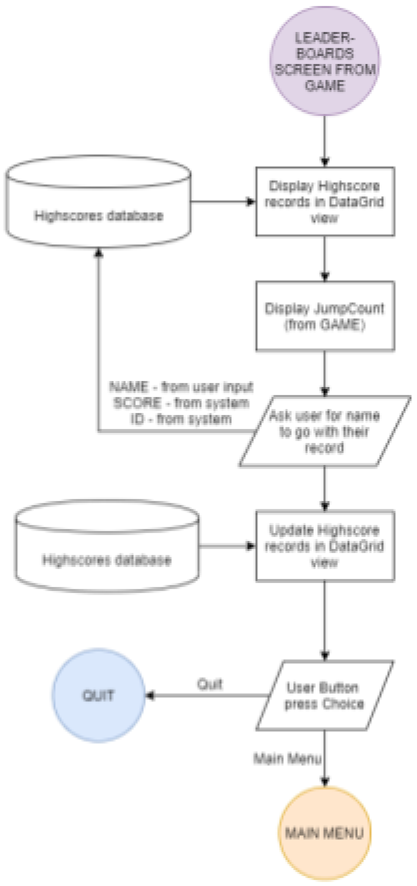

LEADER-BOARDS SCREEN FROM GAME
Highscores database
Display Highscore records in DataGrid view
Display JumpCount (from GAME)
NAME - from user input
SCORE - from system
ID - from system
Ask user for name to go with their record
Highscores database
Update Highscore records in DataGrid view
QUIT
Quit
User Button press Choice
Main Menu
MAIN MENU

*Figure SEQ Figure * ARABIC 11: Flowchart diagram of how my prototype works*

The way I envisioned my program to navigate and work has been slightly altered too, Figure 11 shows how my prototype works; elements of the flowchart diagram have been coloured too in order to make it easier to read.

I also found that my game had performance issue bug which progressively worsened itself after every respawn, but would fix itself when the next level was loaded. I did not expect this to happen in during the development of my prototype, this bug is annoying but I feel that it makes the game easier to play for the user due to how the game play becomes slowed down.

During the development of my game I also felt that at one point that the player was too small (20x20 units in visual basic properties), and because the player is supposedly the main focus for the user, I felt that the size had to be slightly bigger so that it made more of a visual prominence. As a result, I made some adjustments and the player particle is now 50x50 in size.

Object orientated programming approach

When my program loads the next level, a new instance of the “LevelSetUp” class is made, and this instance becomes what the user sees when they are playing the game. In my class, I have also included the use of reading from text files as well as exception handling as they also play a key role in the functionality of my object orientated programming approach to drawing my game’s levels.

As explained previously, I have chosen to go for an object orientated programming approach as it was heavily influenced by this year’s AQA Pre-release; by doing what I did, I felt that I would be able to gain a clearer understanding of how the paradigm works.

My Class, named “cLevelSetUp” consists of a constructor, one private function and two public functions. My constructor subroutine allows for the FileName to be passed through as a parameter into the class from the game form, the private function (called “GetColour”) allows for my program to figure out what colour a given picturebox will be; one of the public functions returns the number of pictureboxes the game has to draw for the next level, while the other public function does the text file reading.

In the text file reading function, called “GetLevelInfo” of the cLevelSetUp class, a defined loop has been used to assign the properties of each individual picturebox the game would have to draw, this has been done because each picturebox is represented by a given number of lines in my textfiles. After the properties are assigned with their values from the text file, the newly created picturebox is then added to a list called “Objects”, this list is then returned to the game when the entire textfile has been loaded into the program (this is when the textfile is completely read by the program). In this function, there has been use of exception handling too, of which handles if the textfile being read does not conform to the algorithm which reads the text files.

```
Public Class cLevelSetUp
    Public Function GetLevelInfo(ByVal FileName As String)     'ByVal FileName As
String -> Levels are stored as text files which stores the properties of pictureboxes
in the order: Xcoordinate, Ycoordinate, Width, Height, ID/name, Colour, tag
        FileOpen(1, FileName, OpenMode.Input)   'File is opened here
        Dim CurrentLine As String
        Dim Objects As New List(Of PictureBox)
        Try
            Do Until EOF(1)                                     'EOF means end of file //
This loop will continue until it has reached the end of the text file it is reading
                Dim PICTUREBOX As New PictureBox                '   "PICTUREBOX" will be a
temporary name for the newly added picturebox
                For PictureBoxProperty = 1 To 8                                'For loop
will loop through every line of the text file and assign the values to a given
pictruebox property
                    CurrentLine = LineInput(1)
                    If PictureBoxProperty = 1 Then
                        PICTUREBOX.Left = CurrentLine
                    ElseIf PictureBoxProperty = 2 Then
                        PICTUREBOX.Top = CurrentLine
                    ElseIf PictureBoxProperty = 3 Then
                        PICTUREBOX.Width = CurrentLine
                    ElseIf PictureBoxProperty = 4 Then
                        PICTUREBOX.Height = CurrentLine
                    ElseIf PictureBoxProperty = 5 Then
                        PICTUREBOX.Name = CurrentLine
                    ElseIf PictureBoxProperty = 6 Then
                        PICTUREBOX.BackColor = GetColour(CurrentLine)    'The colour is
determine via a private function/getter method which is exclusive to this class
                    ElseIf PictureBoxProperty = 7 Then
                        PICTUREBOX.Tag = CurrentLine
                    Else
                        CurrentLine = ""                        'This line is to handle
the "==========" in the text file for drawing the levels
                    End If
                    Objects.Add(PICTUREBOX) 'Each temporary picturebox control is
added to the list so that it's properties can be stored to be passed back into the
game player form
                Next
            Loop
        Catch ex As Exception   'Exception handling to prevent crashes if a badly
written textfile is read
            MsgBox("Error Loading file")
        End Try
        FileClose(1)    'Files that are opened have to be closed
        Return Objects  'The list of picturebox controls with their respective
properties is returned (to the game player form where the class is accessed)
    End Function
```

Figure 12: The GetLevelInfo public function of my cLevelSetUp class which reads textfiles and stores the contents of each individual textfile into a list. The list holds the properties and data needed to recreate each individual level in the game itself. The program then returns this list (called "objects") to the game into the subroutines which draw them out, see Figure 13 and Figure 14for this.

In order to make instances of this class, I have a subroutine which accepts the list of pictureboxes produced by the public function of the cLevelSetUp class. This subroutine begins by creating a temporary instance called "Level" which is then used to access my class's methods. After this is done, the subroutine goes through each picturebox in the list (the number of this is found using the other public function in my class) and will make an attempt to place each picturebox control at the right layer; what is meant by this is that the background will always be in the background and everything else will go in front of it. In addition to this, I also have implemented exception handling to help to prevent the program from crashing if the textfile had been written badly. This subroutine also increments the Level Counter as well as redrawing it.

```
    Private Sub DrawLevel(ByVal FileString As String)
        Static CurrentLevel As Integer  'CurrentLevel is defined once as a static to
prevent reseting everytime this sub is run
        Dim Level As New cLevelSetUp(FileString)    '"Level" is the name for the
instance of the cLevelSetUp Class; this represents the level the user will be playing
on at a given time. This instance is rewritten everytime a new level is loaded with
the details to draw the next level.
        Dim FileNotLoaded As Boolean = False    'This boolean checks if the File being
loaded was fully loaded
        Dim Objects As New List(Of PictureBox)  'A list of pictureboxes is used to
read in and draw each level; a list of pictureboxes is produced by the cLevelSetUp
class whenever a new level's textfiles is read in.
        Objects = Level.GetLevelInfo(FileString)    'The List stores the properties of
the pictureboxes that are to be drawn.
        Dim ArrayLength As Integer = Level.GetNoOfPictureboxes(FileString)  'This
value is used so that the game knows how many pictureboxes it is dealing with in the
for loop below
```

```
        Try
            For i = 0 To ArrayLength    'Will loop through each item in the List of
picturebox controls
                Me.Controls.Add(Objects(i))     'THIS LINE ADDS EACH PICTUREBOX IN THE
"OBJECTS" LIST TO THE FORM
                If Objects(i).Tag = "Sky" Then          'If statements here will
determine where the drawn pictureboxes will sit on the form
                    Objects(i).SendToBack()             ' "sky" will always be sent to
the back because it is esentially the game's background
                ElseIf Objects(i).Tag = "Floor" Then    ' "Floor" tags will be moved
fowards as they are the thing which the player will be moving on - the player needs to
see these
                    Objects(i).BringToFront()
                ElseIf Objects(i).Tag = "Void" Then     ' "Void" tags will be moved
fowards as they are the thing which the player should be avoiding; in order to avoid
it, the player must be able to detect it
                    Objects(i).BringToFront()
                ElseIf Objects(i).Tag = "Player" Then   ' "Player" has to be moved
fowards to the user can see where they are going
                    Objects(i).BringToFront()
                End If
            Next
        Catch ex As Exception       'Use of exception handling for use when the game
reads in a badly written level textfile
            MsgBox("Cannot draw the level - save file is corrupt")
            FileNotLoaded = True
        End Try
        If FileNotLoaded = False Then   'If the textfile contents are fully loaded,
the code below will run
            If Objects.Count < MinimumComponents Then   'If there arent enough
contents to draw, the user is sent to the leaderboard screen
                MsgBox("Not enough pictureboxes to draw one level")
                leaderboards.SetUpForReading()
            Else
                CurrentLevel += 1           'Increments the level counter so the game
knows what level the user is currently playing on
                DrawLevelCounterLabel(True) 'ReDraws the level counter so the user can
see it
                DrawScore()
            End If
        Else    'If the textfile contents are not loaded, the user is sent to the
leaderboard screen
            leaderboards.SetUpForReading()
        End If
    End Sub
```

Figure 13:The DrawLevel subroutineof the game player form which makes instances of the LevelSetUp class to draw each level of the game; this is used when a given level is loaded for the first time; i.e. when the game is started for the first time or when the player reaches the end of a level and needs the next one to load.

My program creates instances of the level whenever a new level has to be loaded (Figure 13 corresponds to this) and whenever the game needs to reset the level, in the case where the player is respawned by the game (Figure 14 shows the respawning instance of the cLevelSetUp class). Both subroutines in Figure 13 and Figure 14 work similarly but they do have their differences, one being that the DrawLevel sub contains exception handling since a level is being drawn for the first time whereas the ResetLevel sub doesn't contain exception handling (since already loaded levels are readable for the drawing algorithm). There are also additional checks in the DrawLevel sub to ensure that the textfiles contain enough contents to draw.

```
    Private Sub ResetLevel(ByVal FileString As String)  'works similarly to the DrawLevel sub except it doesn't increment the Level counter; this is used in the respawning mechanism. No exception handling because if the level has already been drawn it means it is already readable for the program
        Dim Level As New cLevelSetUp(FileString)    '"Level" is the name for the instance of the cLevelSetUp Class; this represents the level the user will be playing on at a given time. This instance is rewritten everytime a new level is loaded with the details to draw the next level.
        Dim Objects As New List(Of PictureBox)  'A list of pictureboxes is used to read in and draw each level; a list of pictureboxes is produced by the cLevelSetUp class whenever a new level's textfiles is read in.
        Objects = Level.GetLevelInfo(FileString)    'The List stores the properties of the pictureboxes that are to be drawn.
        Dim ArrayLength As Integer = Level.GetNoOfPictureboxes(FileString)  'This value is used so that the game knows how many pictureboxes it is dealing with in the for loop below
        For i = 0 To ArrayLength    'Will loop through each item in the List of picturebox controls
            Me.Controls.Add(Objects(i))     'THIS LINE ADDS EACH PICTUREBOX IN THE "OBJECTS" LIST TO THE FORM
            If Objects(i).Tag = "Sky" Then          'If statements here will determine where the drawn pictureboxes will sit on the form
                Objects(i).SendToBack()             ' "AIR" will always be sent to the back because it is esentially the game's background
            ElseIf Objects(i).Tag = "Coin" Then     ' "COIN" tags will be moved fowards as they are the thing which the player collects in order to achieve a higher score
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Floor" Then    ' "Floor" tags will be moved fowards as they are the thing which the player will be moving on - the player needs to see these
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Void" Then     ' "Void" tags will be moved fowards as they are the thing which the player should be avoiding; in order to avoid it, the player must be able to see it
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Player" Then   ' "Player" has to be moved fowards to the user can see where they are going
                Objects(i).BringToFront()
            End If
        Next
        DrawLevelCounterLabel(False)    'Accessing method for drawing the Level counter which will not increment the value in the level counter
        DrawScore()
    End Sub
```

Figure 14: The ResetLevel subroutineof the game player form which makes instances of the LevelSetUp class to redraw each level of the game when the player dies; respawning does not require for exception handling because if the level has already been drawn and partially played by the user, it means it is already readable for the program. This improves the performance of my game as it means less processing power is used whenever a respawn is initiated.

This feature can be seen in action on the youtube videos I have made to showcase my prototypes:
Videos: Red are of my old prototype, Green are of the second iteration of my prototype.

- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)

- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

Game pausing feature

As part of my game, I wanted there to be an option for the user to pause and continue the game as they please. In order to incorporate this, I decided to add subroutines which would be able to turn on and off the visual basic timer controls I used for the side scrolling mechanism, as well as additional user inputs for when the game is running for the user to pause/continue the game; the code for this can be seen in Figure 15 where key “P” pauses the game and key “C” resumes the game.

```
    Private Sub frmPlay_KeyDown(sender As Object, e As KeyEventArgs) Handles
MyBase.KeyDown 'Handles game's inputs
        Select Case e.KeyCode
            Case Keys.Space                     'code below is run when the user
presses space
                If TouchingFloor = True Then    'will only run if the player is
touching the floor; preventing jumping mid-air
                    UpwardsSpeed = JumpingPower 'The player is accelerated upwards by
the quantity specified in the constant JumpingPower
                    TouchingFloor = False       'When the player jumps, they are no
longer touching the floor - purpose of this line
                    JumpsMade += 1  'Increments the jumps made by the player each time
they jump
                End If
            Case Keys.P             'Pressing P pauses the game
                StopTimers()        'This is done through a sub which disables all the
timers in my program
            Case Keys.C             'Pressing C resumes/continues the game
                ActivateTimers()    'This is done through a sub which re-enables all
the timers in my program
        End Select
    End Sub
```

Figure 15: The possible user inputs of the game

Depending on the key which the user presses, either the subroutine StopTimer(), seen in Figure 16, can be called or ActivateTimer(), seen in Figure 17. StopTimer will result in all timers being stopped/paused and this means that the player particle will be locked in position both in the x and y axis. Whereas the ActivateTimer subroutine will enable all timers meaning that the player is unlocked and free to move in the x and y axis.

```
    Private Sub StopTimers()            'Disables the all timers which are used in my
game; used in the pause/continue feature
        tmrGameLogic.Enabled = False
        tmrGravity.Enabled = False
    End Sub
```

Figure 16: The StopTimer() subroutine which disables timers that control movement in the game

```
    Private Sub ActivateTimers()        'Re-enables the all timers which are used in
my game; used in the pause/continue feature
        tmrGameLogic.Enabled = True
        tmrGravity.Enabled = True
    End Sub
```

Figure 17:The ActivateTimer() subroutine which disables timers that control movement in the game

This feature can be seen in action on the youtube videos I have made to showcase my prototypes:

Videos: Red are of my old prototype, Green are of the second iteration of my prototype.

- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)
- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

Player collision detection

One of the main focuses of my game was the collision detection which the player particle has; this feature is fundamental for the game to run since this mechanism would help the game decide whether to load in the next level, respawn the player as well as decide if the player particle is able to jump. I was able to implement this using the boundaries of visual basic picturebox controls in conjunction with conditional statements and the tag property of the visual basic picturebox control. I was able to contain all of this within one subroutine, of which is run every tick of the timer meaning that the objects that the player particle touches are checked frequently.

```
Private Sub Collisions()    'Sub is responsible for checking the collisions the player
makes with the level
        Static LevelToDraw As Integer = 1    'This is used when the player reaches the
end of a level; starts of at one because the user first starts playing in level 1
        Static LowestPoint As Integer        'This is the lowest point which the player
can hit
        For Each b As Control In Me.Controls     'Checks which cause the player to
respawn; if the player hits a picturebox with the tag "void" the player is repsawned
            If TypeOf b Is PictureBox Then
                If b.Tag = "Void" Then
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    Respawn(LevelToDraw)    'Respawning mechanism will
simply redraw the entire level, the current level is passed in as a parameter so the
program knows which level to redraw
                                End If
                            End If
                        End If
                    Next
                ElseIf b.Tag = "End" Then   'Checks for when the player reaches the
end of a level; this will prompt the
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    LevelToDraw += 1    'When the user reaches the
last level the LevelToDraw is incremented by 1 in preparation for drawing the next
level
                                    DrawNextLevelOrEndScreen(LevelToDraw)   'This sub
then decides if the next level is drawn, or if the leaderboards screen is opened
                                End If
                            End If
                        End If
                    Next
                ElseIf b.Tag = "Floor" Then 'This is here so that the player doesnt 
sink through the floor due to gravity; this makes sure that the floor is solid for the 
player
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    LowestPoint = (b.Top - 50)      'The Lowest Point 
is defined as the top of a bound - 50 in the Y direction
                                    cntrl.Top = LowestPoint 'The lowest point is the 
lowest point the player can sit at, at a given time
                                    UpwardsSpeed = 0    'When the player is in contact 
with the floor, their vertical speed becomes 0 to prevent the player from forcing 
itself through the floor (this was a bug fix)
                                    TouchingFloor = True    'This check then lets the 
game know that the player is touching the floor, so that the user is given the 
opportunity to jump again
                                End If
                            End If
                        End If
                    Next    'use of nested loops and a lot of if statements for the 
collision detection
                End If
            End If
        Next 'Arguable that I couldve compared everything to "player" instead of 
comparing "player" to everything else but this was not possible - I have tried and 
this made the game have delayed collision detection. Ultimately it was better for me 
to do what I have done here
    End Sub
```

Figure 18: The collisions subroutine which checks what the player particle is touching and decides if there should be an outcome

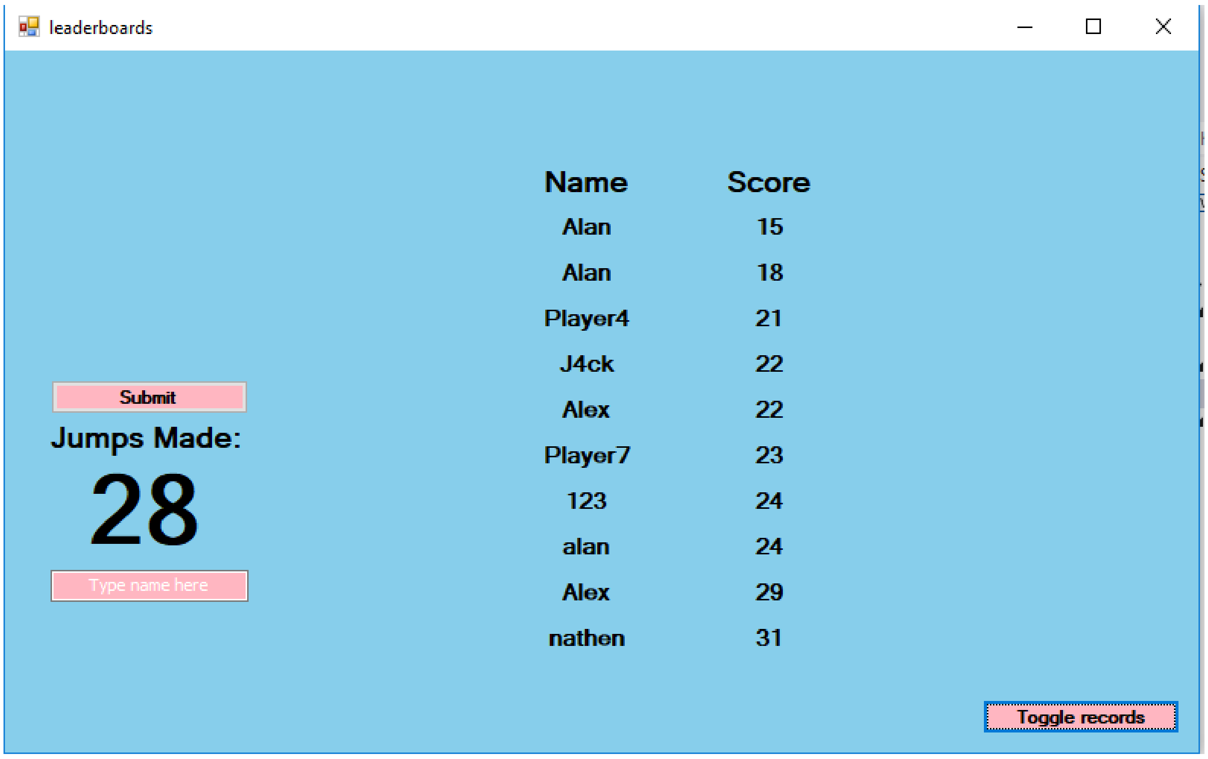
```
    Private Sub tmrGameLogic_Tick(sender As Object, e As EventArgs) Handles 
tmrGameLogic.Tick   'Game Logic check timer
        Collisions()    'Collsions will be checked every tick of this timer
        DrawUpSpeed()   'This is a test sub which helps to debug the jumping mechanism 
by showing the vertical speed of the player
        DrawScore()
        MoveLevel()     'This is the subroutine responsible for the side scrolling of 
my game, checked every tick of the game so that the level moves at the right speed
    End Sub
```

Figure 19: Shows that the Collisions subroutine is run every tick of the GameLogic timer

User input validation and data entry to the database (which stores all scores)

In order for my game to make a record to store in the game's highscores database, the user's score is given from the system and the user is prompted to produce a name to go with the score. When storing names, I wanted to make sure that the names were three to fifteen characters in length and that the names only contained exclusively letters and numbers. Through the use of string operations and Booleans I was able to validate the data given by the user.

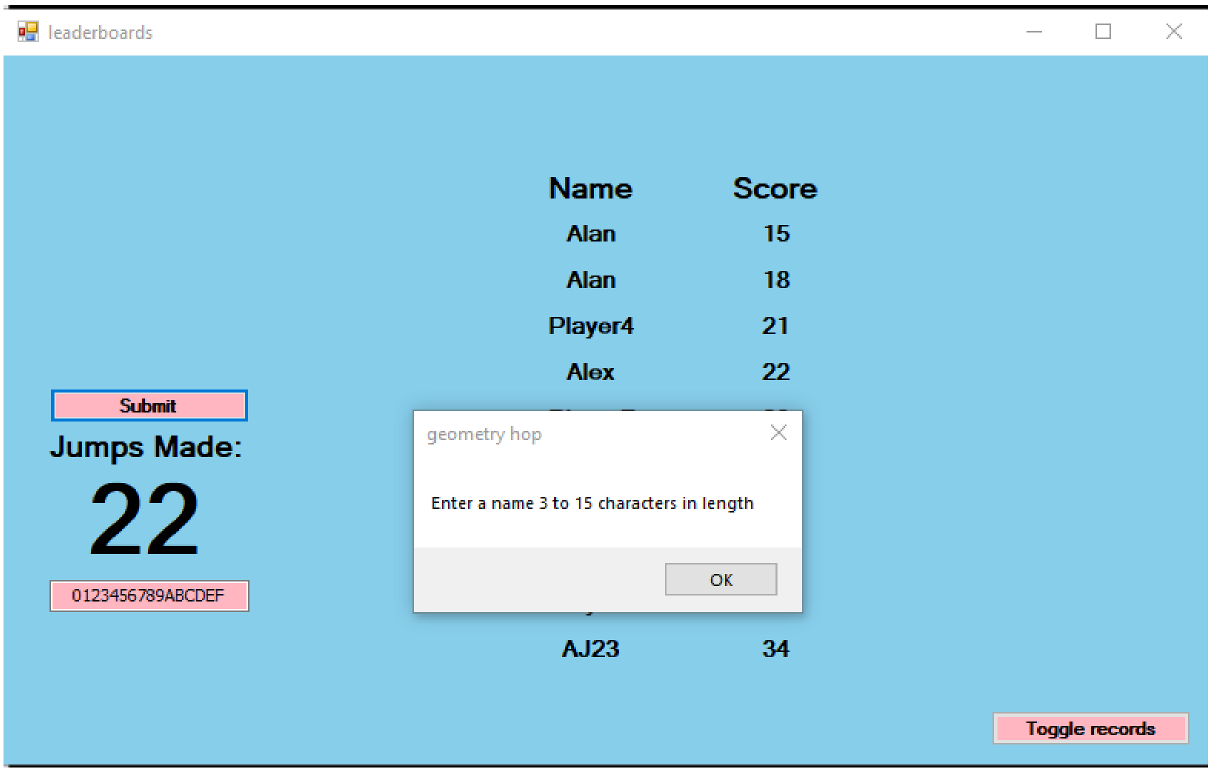
In order to validate a user's data entry, the user firstly has to attempt to submit a name using the submit button (code for this can be seen in Figure 23) on the Leaderboards screen, the screen below (Figure 20).



Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

Figure 20: The leaderboards screen after the user has finished playing a level

This will then result on the validation of the name itself via a function called "CheckValidName" (Figure 25) – validation is done using Regular expressions. Simultaneously, another function ("GetName", Figure 24) is used to get the name that the user has inputted into the textbox they are given; this function removes the textbox prompt which the user gets for the textbox too. In the case where a non-valid name is submitted, error messages are displayed by the system and the user is prompted by the system to enter a new name, this can be seen in the screenshots of the leaderboards screen on the following page in Figure 21 and Figure 22.

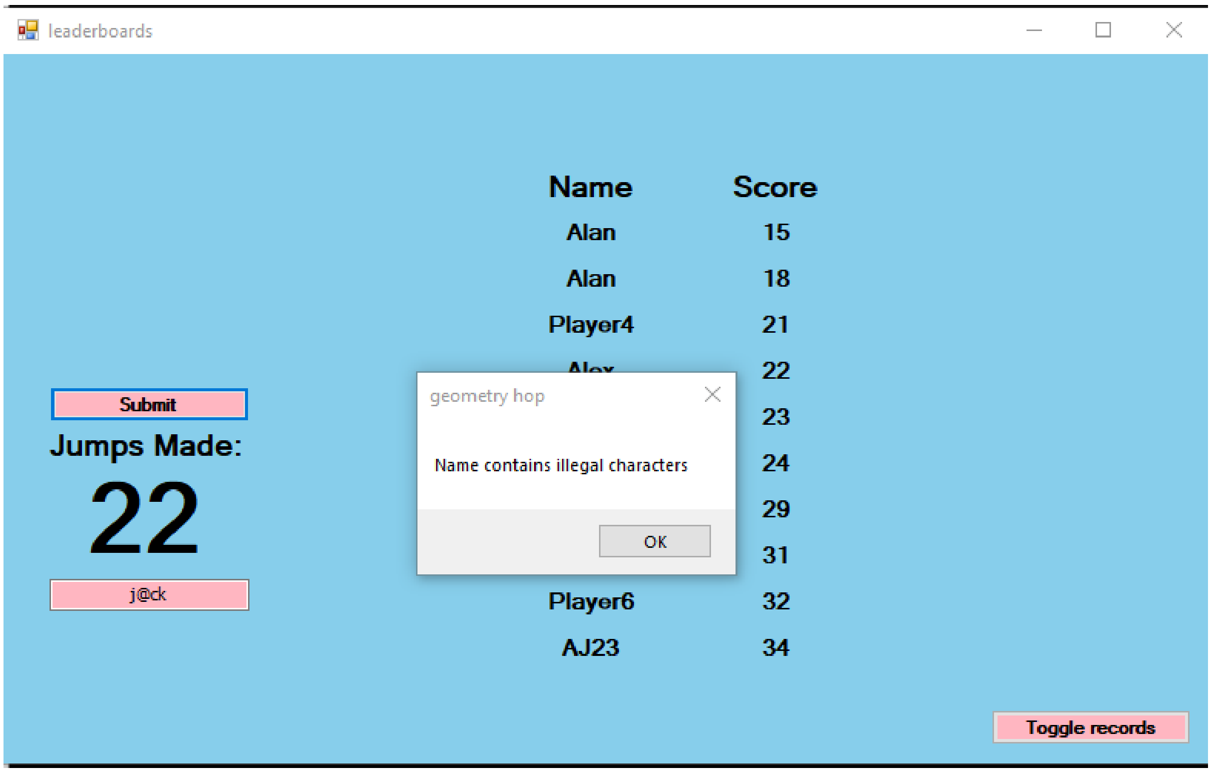

leaderboards

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
AJ23	34

geometry hop

Enter a name 3 to 15 characters in length

Figure 21: Shows the error message displayed when a name of non-valid length has been submitted


leaderboards

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
	23
	24
	29
	31
Player6	32
AJ23	34

geometry hop

Name contains illegal characters

Figure 22: Shows when a name containing characters that are not allowed are entered into the system

After all validation where the user's name entry is validated by the system, the user's name is passed into another subroutine (called "InsertData") that adds the user's records into a database using SQL.

```
    Private Sub btnSubmitFields_Click(sender As Object, e As EventArgs) Handles
btnSubmitFields.Click       'All code below is run when the user attempts to submit
their name & score (record)
        If CheckValidName(GetName()) = True Then    'A check is made on the name
entered using the CheckValidName function, where the name entered by the user is
passed in as a parameter (function parameter)
            InsertData(GetName())   'If the name is valid, then the name is inserted
into the database. The name is also passed in as a parameter here as a function
            HideFields()    'This hides the buttons which allow the user to add to the
database; prevents multiple data entries at once
            MsgBox("Record added!") 'A prompt is made for the user to let them know
that thier record was added to the database
            OpenMenu()  'Then the user is forced into the main menu so that the
leaderboards can refresh
        Else
            MsgBox("Name not valid, enter a different one") 'In the case where the
name entered is not valid, then the user gets a message box prompt notifying them thar
their name is not valid
        End If
    End Sub
```

Figure 23: The subroutine which coordinates the actions which happen when the user wants to submit a field to the database

```
    Private Function GetName()  'Handles how the program will get the name of the user
to store in the database
        Dim NameToStore As String
            If txtYourName.Text.Contains("Type name here") Then 'This is here to
remove the textbox prompt in case the prompt does not disappear by itself (this was a
bug fix)
                NameToStore = txtYourName.Text.Remove(txtYourName.Text.Length - 15)
'Removes 15 since the prompt "Type name here" is exaclty 15 characters from contents
of the Textbox where the user is to enter their name
            Else
                NameToStore = txtYourName.Text
            End If
        Return NameToStore
    End Function
```

Figure 24: GetName function which assists the program in getting the name entered by the user into the system itself. This function was necessary in order to remove the textbox prompt which would sometimes remain (another bug fix made from the original prototype).

```
    Private Function CheckValidName(ByVal NameToCheck As String)    'The name entered
by the user is passed into this function as a parameter
        Const CharCountOfPrompt As Integer = 14
        Const MinNameLength As Integer = 3
        Const MaxNameLength As Integer = 15
        Dim NotAllowedChar As Integer = 0   'This is a counter which counts how many
illegal characters the user has in thier name; it is ultimately used as a check digit
for the program itself
        Dim ValidLength, ValidChars, ValidName As Boolean   'These are the
three booleans which help to determine if a name is valid or not. ValidLength checks
that the name given is the right length; ValidChars checks that the name given only
contains Valid characters; ValidName checks that the name given is both a valid length
and that it only contains valid characters
        If txtYourName.Text.Contains("Type name here") Then 'This is here to remove
the textbox prompt in case the prompt does not disappear by itself (this was a bug
fix)
            NameToCheck = NameToCheck.Remove(NameToCheck.Length - CharCountOfPrompt) '
Removes 14 since the prompt "Type name here" is exaclty 15 characters from contents of
the Textbox where the user is to enter their name
        Else
            NameToCheck = txtYourName.Text  'If the name does not contain the prompt
then the NameToCheck is equal to the value in the textbox
        End If

        If NameToCheck.Length >= MinNameLength And NameToCheck.Length <= MaxNameLength
 Then 'If the name entered is in the alllowed range for name length the below code
will run
            If Not Regex.Match(NameToCheck, "^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9
_]*$", RegexOptions.IgnoreCase).Success Then  'This allows only Letters and numbers
                MsgBox("Name contains illegal characters")  'If the name contains a
non-allowed character, the below code will run
                ValidChars = False  'Then the Valid Cahracters boolean is set to
false
            Else
                ValidChars = True   'Otherwise the valid character boolean is set to
true
            End If
        Else
            MsgBox("Enter a name 3 to 15 characters in length") 'If the name is
outside the allowed range in length, an error message is returned
            ValidLength = False
        End If

        If (ValidLength = True) And (ValidChars = True) Then    'Checks to see if the
name is valid in terms of length and characters used
            ValidName = True    'If both length and characters used in the name are
valid, then the entire name becomes valid
        Else
            ValidName = False   'If either length or characters used in then name are
false, then the entire name is no longer valid
        End If
        Return ValidName    'The overall validity of the name is then returned to the
program
    End Function
```

Figure 25: CheckValidName function which checks if the name entered by the user is valid for storage in the database. Names must be three to fifteen characters in length and they should only be comprised of the Allowed characters by the system, of which have been defined using a constant. If the user's name is not valid, they will be notified on what makes it not valid by this exact function.

As stated previously, the system does use SQL to insert data into the database, this can be seen below in Figure 26.

```
    Private Sub InsertData(ByVal Name As String)     'name which user would like to
store in thier record is passed into this subroutine
        Dim SQLstring, SQLQuery As String
        Dim Score, ID As Integer
        Dim Connection As OleDbConnection
        Dim Command As OleDbCommand
        SQLQuery = "SELECT Count(HighScores.ID) AS CountOfID FROM HighScores"   'This
line helps the system to create an automatic ID for the record, IDs are used as
primary keys
        ID = CInt(ExectuteSQLQueryINTEGER(SQLQuery)) + 1    'Automatic ID, ID value
incremented by 1 each time a new record is added
        Score = lblYourScore.Text   'Score is taken from the system - cannot be
changed since the user cant acutally change the contents of a label
        Connection = New OleDbConnection(ConnString)
        SQLstring = "INSERT INTO HighScores VALUES(" & ID & ",'" & Name & "', " &
Score & ") "  'INSERT query applied where the ID, Name and score is inserted into the
database
        Connection.Open()
        Command = New OleDbCommand(SQLstring, Connection)   'SQL applied to database
here
        Command.ExecuteNonQuery()
        Connection.Close()
    End Sub
```

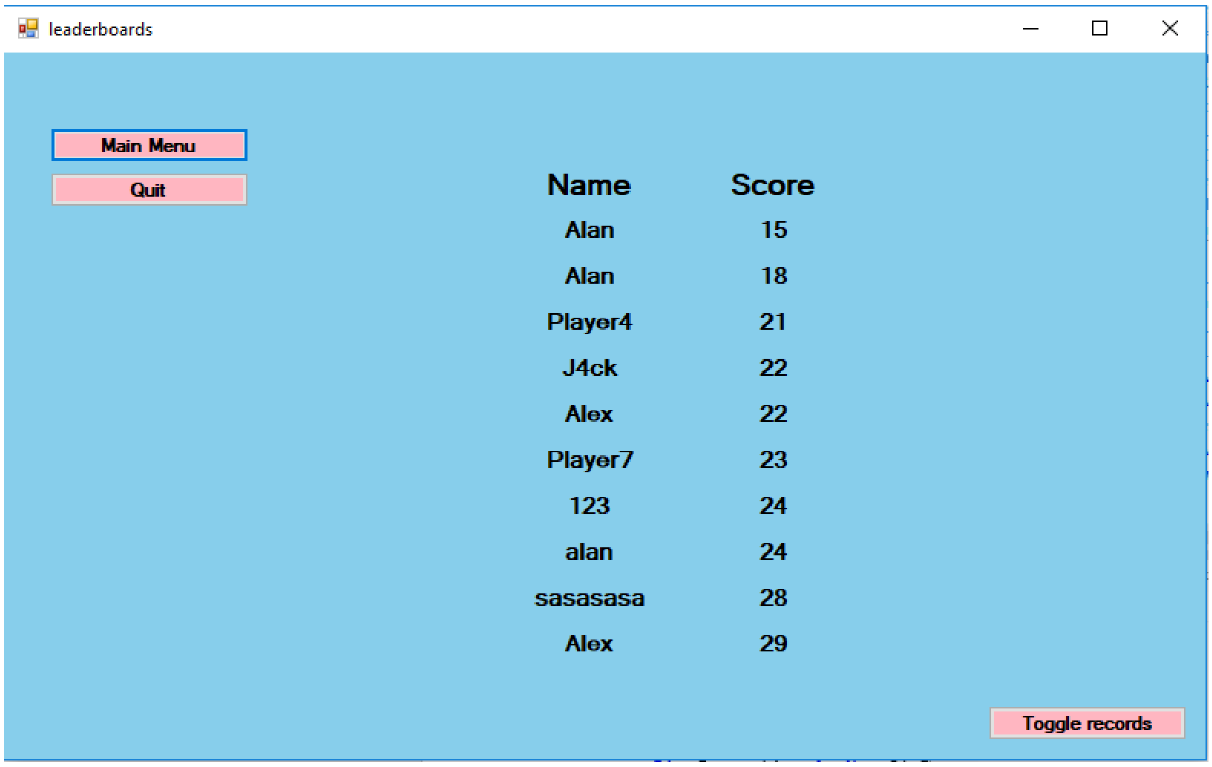
Figure 26: The InsertData subroutine of which assists in the insertion of data into the highscores database

```
INSERT INTO HighScores VALUES(" & ID & ",'" & Name & "', " & Score & ")
```

This is the SQL query used by my system to insert data into the database. An insert query is used and it will take the values ID – generated by the system, Name – generated by the user, and Score – which is generated by the system depending how well the player plays the game.

Presenting the top ten scores in the leaderboard screen

In order for me to display the top ten scores of my game, I have decided to use a combination of labels and SQL; the SQL was used to retrieve data from the database to be displayed as text for the user to read – this text was then stored in labels to make up some of the graphical user interface of the leaderboards screen.



Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
sasasasa	28
Alex	29

Figure 27: The leaderboards screen where the top ten scores have been represented using label

To begin with, data from the database is read into a vb.net data grid view control. This data grid view is then used to feed data to my program so that the top ten scores can be represented using labels; this algorithm can be seen in. Data is read into the data grid view so that the scores are ordered in ascending order; this makes it easier for my program to pick out the top ten scores.

```
    Private Sub ShowDatabase()
        Dim dtHighScores As New DataTable
        Dim Connection As New OleDbConnection
        Connection.ConnectionString = ConnString
        Connection.Open()
        Dim ds As New DataSet
        Dim dt As New DataTable
        ds.Tables.Add(dt)
        Dim da As New OleDbDataAdapter
        da = New OleDbDataAdapter("select * FROM HighScores ORDER BY score ASC",
Connection)    'SELECT query which orders the records by score in ascending order
```

```
        da.Fill(dt) 'DataTable is filled by the DataAdapter which adapts data from
database so that it can be used in vb
        dgvHighScores.DataSource = dt.DefaultView   'DataTable then fills out the data
grid view control
        Connection.Close()
```

Figure 28: ShowDatabase sub which assists in the reading in of data to the game itself, the data from the database populate a vb.net data grid view control which is used to display the top ten scores in another procedure

```
select * FROM HighScores ORDER BY score ASC
```

This is the SQL query used by my system, it will read in all records of my database and it will order them all in ascending order of score. All of this will be loaded into a visual basic Data grid view control.

After the vb.net data grid view has been populated, a grouping of subroutines is used to display the top ten scores to the leaderboard screen. During this process, the vb.net data grid view control is hidden from the user but it can be toggled (hidden or shown) so that the user can see all the records in ascending order of score if they wish to do so. I have decided to create each label making up the leaderboard screen programmatically too; by doing so I was able to use a loop to fill in all the labels.

In theory programmatically making the labels means that I cut down on processing power on the computer when the program is idling since there are less components for the computer to acknowledge when the leaderboard isn't open, in comparison to if I had a number of labels pre-made on visual studio itself.

```
    Const NameInDGV As Integer = 1   'column of the user's name in the vb.net data
grid view control
    Const ScoreINDGV As Integer = 2  'column of the user's score in the vb.net data
grid view control
    Const xName As Integer = 333    'xCoord of the name column - so that all name
labels are alligned with each other
    Const xScore As Integer = 453   'xCoord of the score column - so that all score
labels are alligned with each other
    Const yTop As Integer = 74      'the y coordinate for the top of the table
    Const ySpacing As Integer = 30  'spacing to increment by for each row
```

Figure 29: The constants which I defined in order to tidy up the code for the filling in and drawing of the labels which make up the leaderboards

```
    Private Sub ShowTopTen()
        ShowDatabase()  'the vb.net data grid view control has to be seen by the
program for this to work
        DrawTitles()    'Title lables are drawn
        Dim yCoordToPass, CurrentRow As Integer 'This value is passed as a parameter
into the DrawLabels subroutine so that the program knows where to draw a given score
and name
        For i = 0 To 9      'Repitition is used here to produce all of the labels
which make up the records part of the leaderboards screen
            yCoordToPass = yTop + (i * ySpacing + ySpacing)    'The spacing from the
top labels is increased every iteration of this loop to produce a leaderboard
            CurrentRow = i
            DrawLabels(yCoordToPass, CurrentRow) 'This sub accepts two parameters
which is used to aid the formatting of the leaderboard screen
        Next
        dgvHighScores.Hide()    'This hides the vb.net data grid view control so that
it does not get in the way of the user
    End Sub
```

Figure 30: Using a defined loop (of 10) I was able to draw the top ten scores. This was done by calling in a subroutine each iteration of the loop while passing in different parameters for a different outcome

```
    Private Sub DrawLabels(ByVal yCoord As Integer, ByVal CurrentRow As Integer)
'yCoord determines how far down a label is drawn. CurrentRow tells the program which
row of the vb.net data grid view control to read from
        Dim Name, Score As New Label
        Dim ScoreToShow, NameToShow As String
        NameToShow = dgvHighScores.Item(NameInDGV, CurrentRow).Value    'Gets the name
of the user from the vb.net data grid view control
        ScoreToShow = dgvHighScores.Item(ScoreINDGV, CurrentRow).Value  'Gets the
score of the user from the vb.net data grid view control
        'NAMES LABELS
        Name.Location = New Point(xName, yCoord)
        Name.TextAlign = ContentAlignment.MiddleCenter
        Name.BackColor = Color.SkyBlue
        Name.Font = New Font("Microsoft Sans Serif", 11, FontStyle.Bold)
        Name.Text = NameToShow
        'SCORES LABELS
        Score.Location = New Point(xScore, yCoord)
        Score.TextAlign = ContentAlignment.MiddleCenter
        Score.BackColor = Color.SkyBlue
        Score.Font = New Font("Microsoft Sans Serif", 11, FontStyle.Bold)
        Score.Text = ScoreToShow
        'ADD TO FORM
        Me.Controls.Add(Name)
        Me.Controls.Add(Score)
    End Sub
```

Figure 31: The subroutine shown here is the one which is called every iteration of the for loop; this subroutine accepts parameters which assists on the drawing and filling in of the labels which make up the leaderboards screen

Testing

After the completion of my program, it will be tested to ensure that the program is in working order and to make sure that all objectives I have given myself in the design section of my report have been met.

Testing the leaderboards screen (LB Test No)

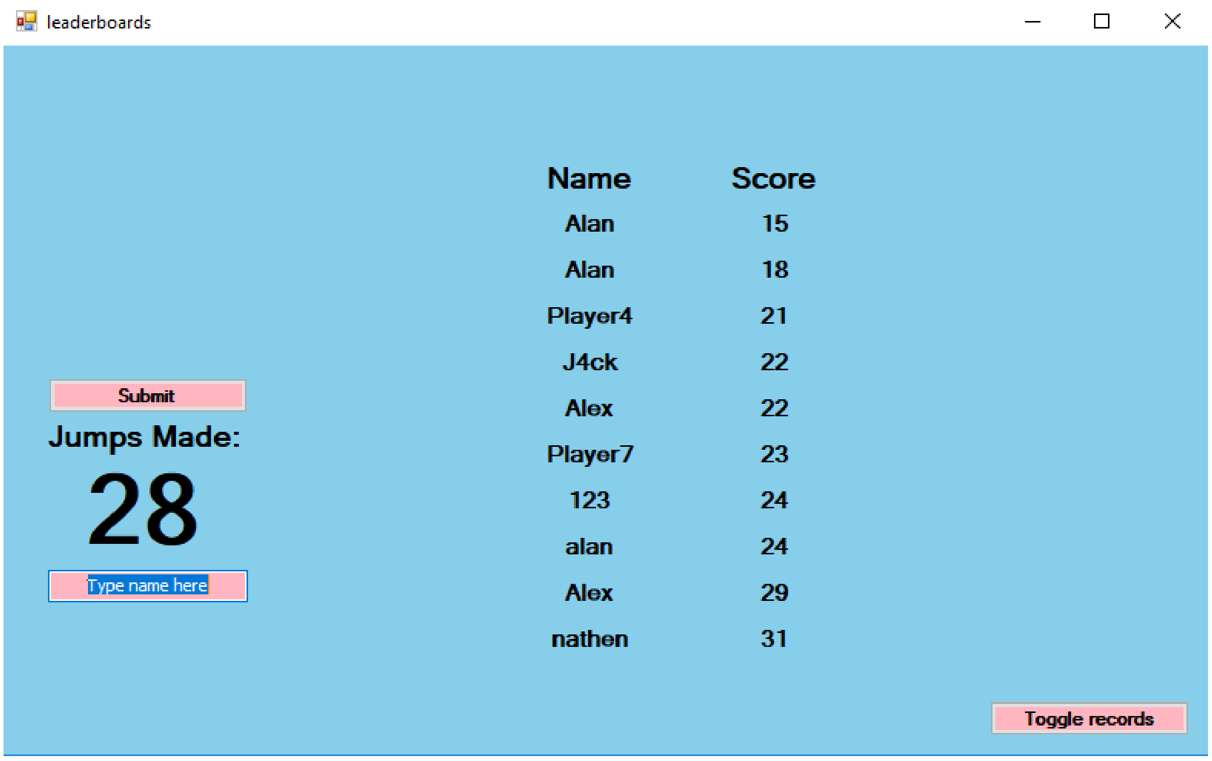
In this section I will be testing the leaderboards screen to check that it functions the way I intend it to; below is a table where I have described the tests which I have carried out on this screen/for this screen as well as the outcomes which I expected as well as the actual outcomes of the tests themselves

Test description	Test No.	Expected outcome(s)	Actual outcome
The user should only be able to add a record to the database when they have finished playing the game	LB.1	The user should only be allowed to enter a record when they have finished as game but not when they access the leader boards screen from the main menu	Same as expected outcome
Test to see that only one record can be added to the game's database at one time	LB.2	Users should not be given the option to add a record to the database when they open the leaderboards screen from the main menu Users should be given the opportunity to add a record when they finish the game	Same as expected outcome
When the leaderboards screen is opened at the end of game, the user should get a prompt in the textbox	LB.3.1	There is a prompt when the screen is opened in the textbox	Same as expected outcome
	LB.3.2	The prompt should disappear when the user begins to start typing	Same as expected outcome
	LB.3.3	The prompt should disappear when the user hovers their mouse over the textbox	Same as expected outcome
	LB.3.4	The prompt should reappear when the mouse leaves the textbox	Same as expected outcome
	LB3.5	The prompt should reappear when the textbox is empty	Same as expected outcome
The leaderboards screen should close when the user adds a record to the database	LB.4	After the user presses submit, after playing a game, to add a record to the database the leaderboards screen should close and the user should be taken to the main menu by my program	Same as expected outcome
Testing the toggling of the database	LB.5	Pressing the button when the database is hidden should show the database	Same as expected outcome
	LB.6	Pressing the button when the database is shown should hide the database	Same as expected outcome
Test that the scores are displayed in descending order	LB.7	The scores should be displayed in descending order of score in the leaderboards as well as the DataGridView control when it is toggled	Same as expected outcome

Test Results for testing the leaderboards screen (LB Test No)

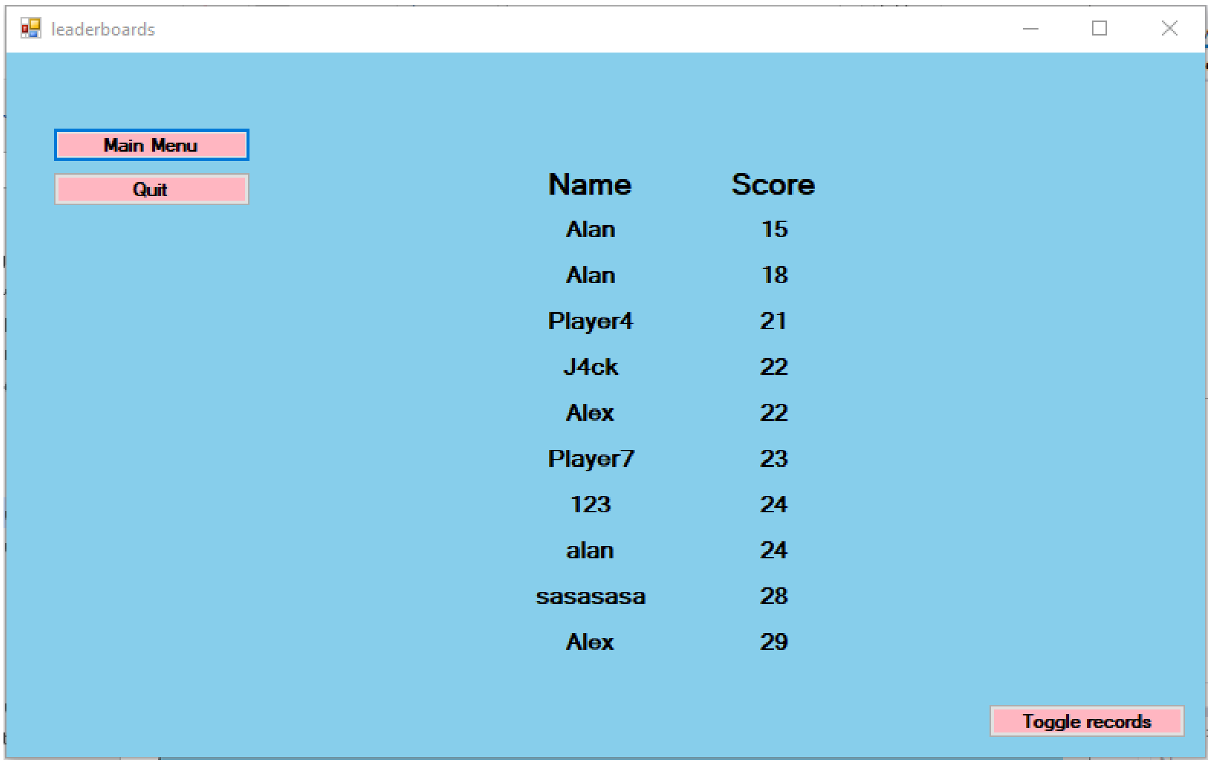
LB.1

After completing a run:



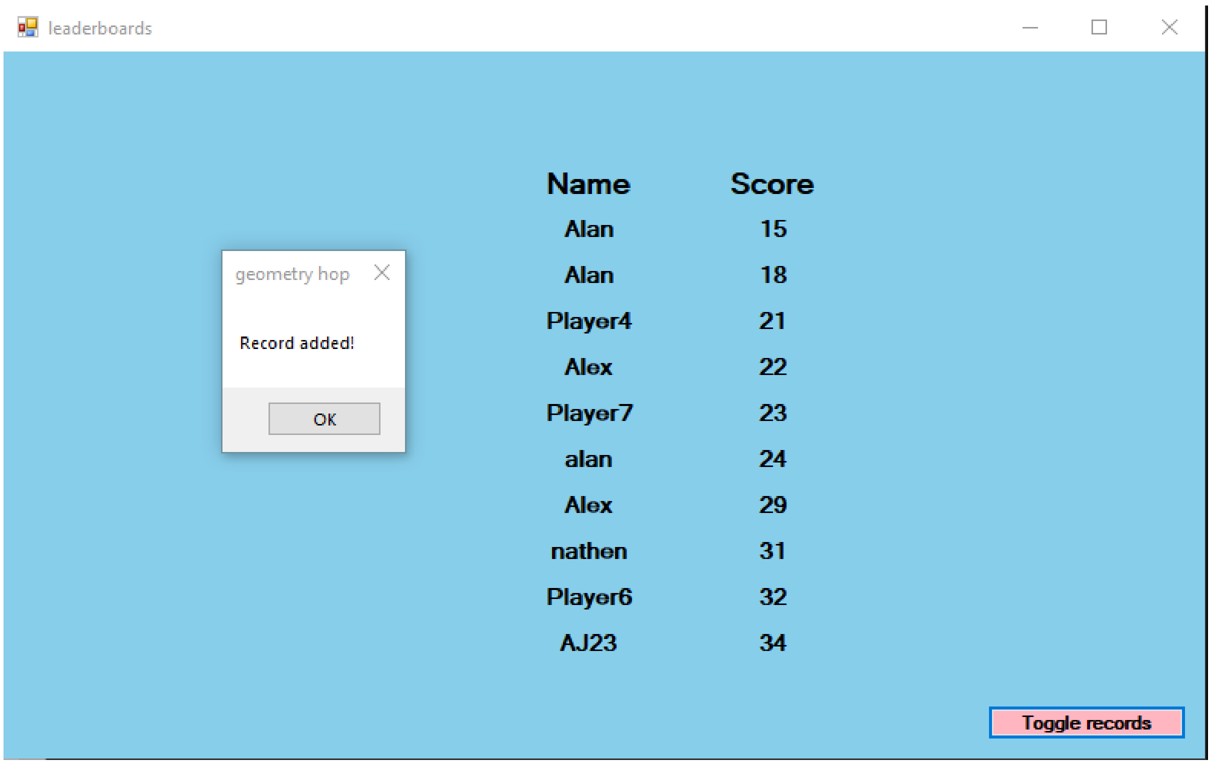
Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

When opening from the main menu:



Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
sasasasa	28
Alex	29

LB.2

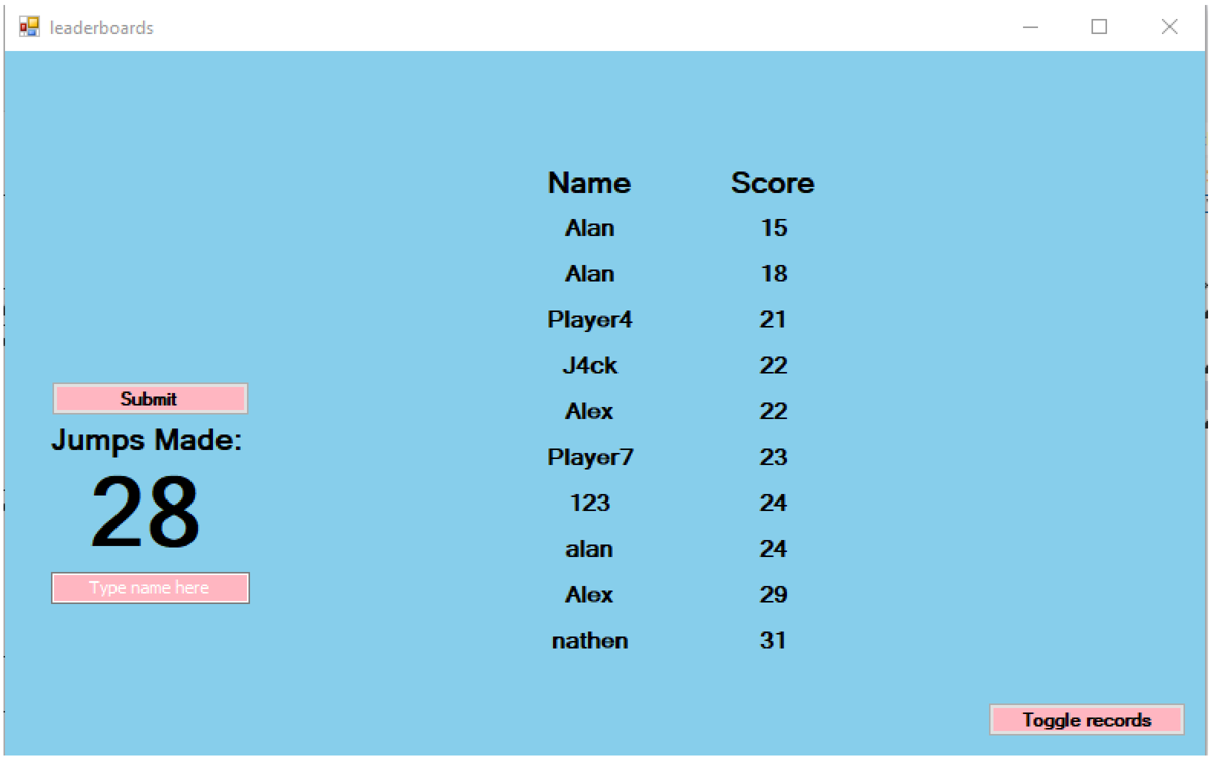


Record added!

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
Player7	23
alan	24
Alex	29
nathen	31
Player6	32
AJ23	34

Fields to enter disappear meaning test was successful

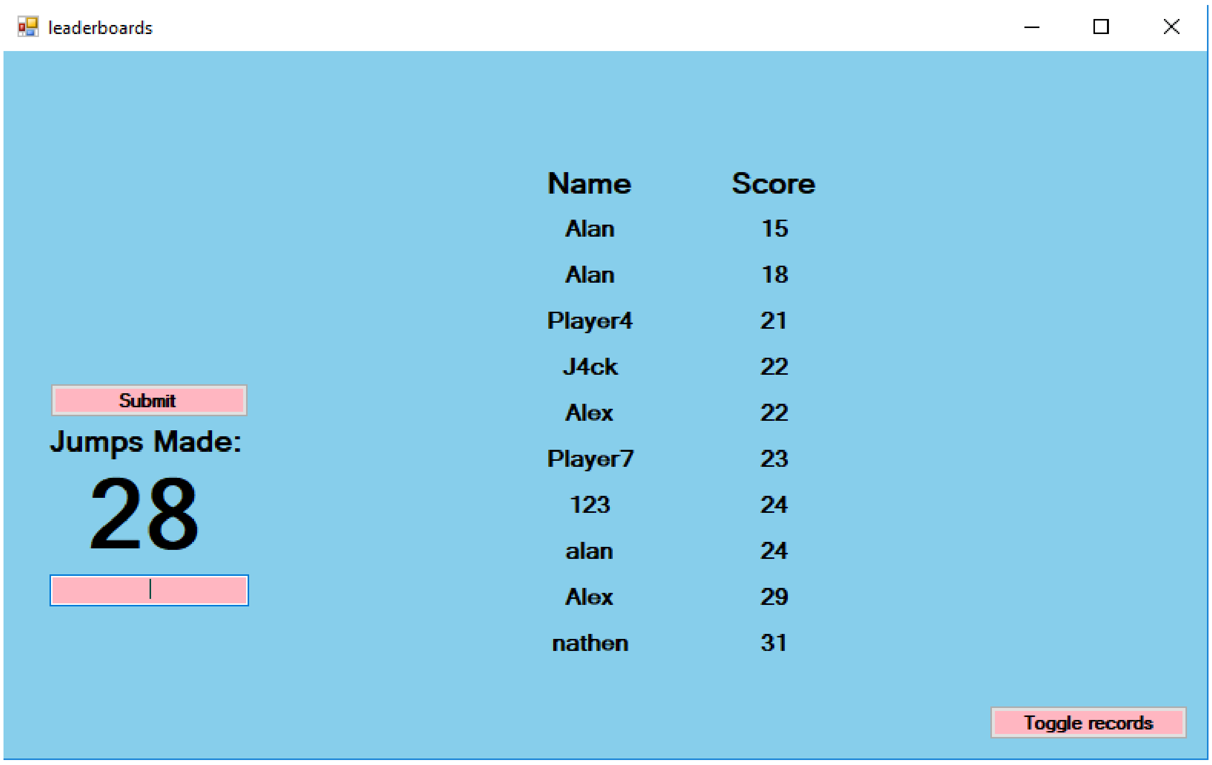
LB.3.1



Jumps Made:
28

Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

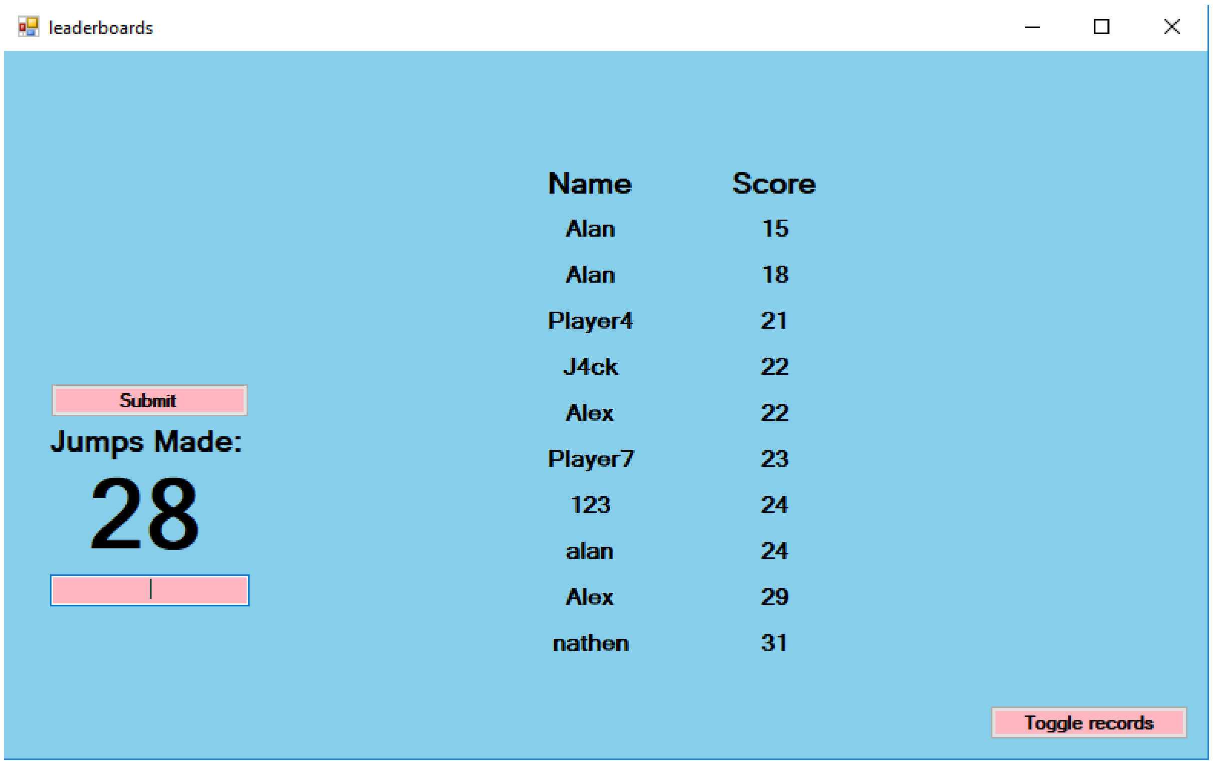
LB.3.2



Jumps Made:
28

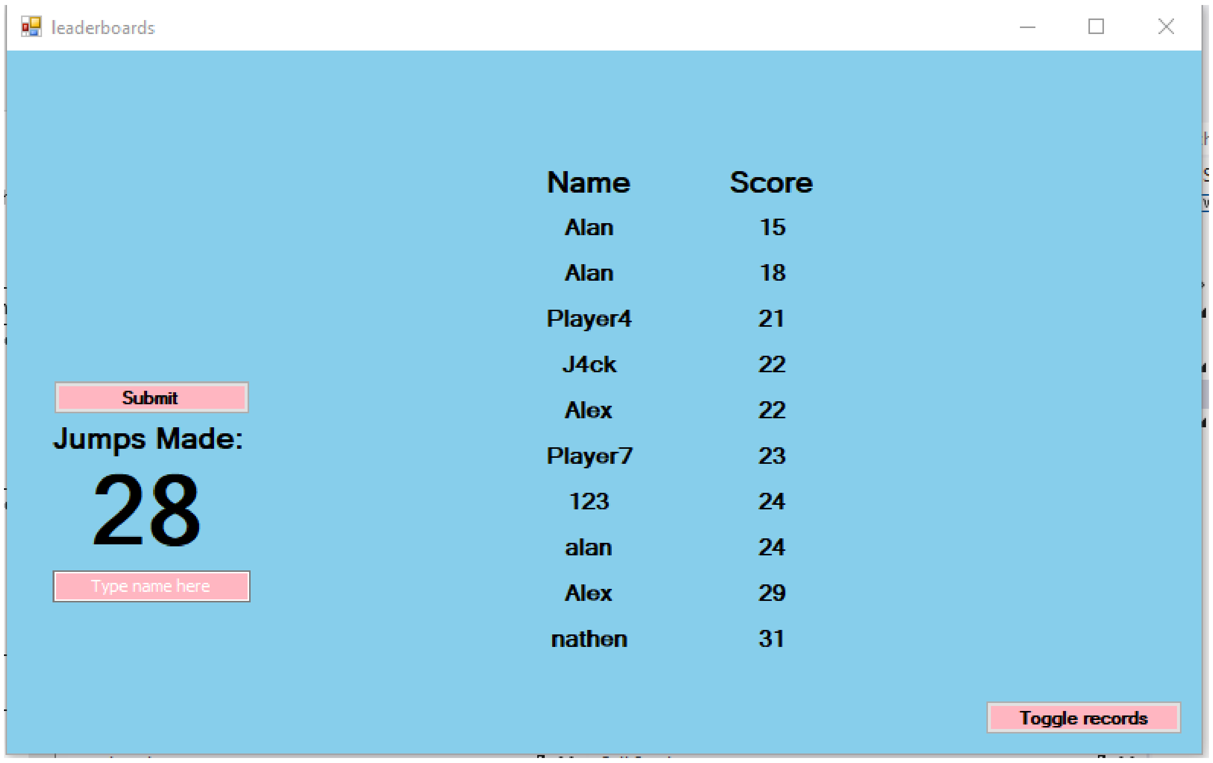
Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

LB.3.3



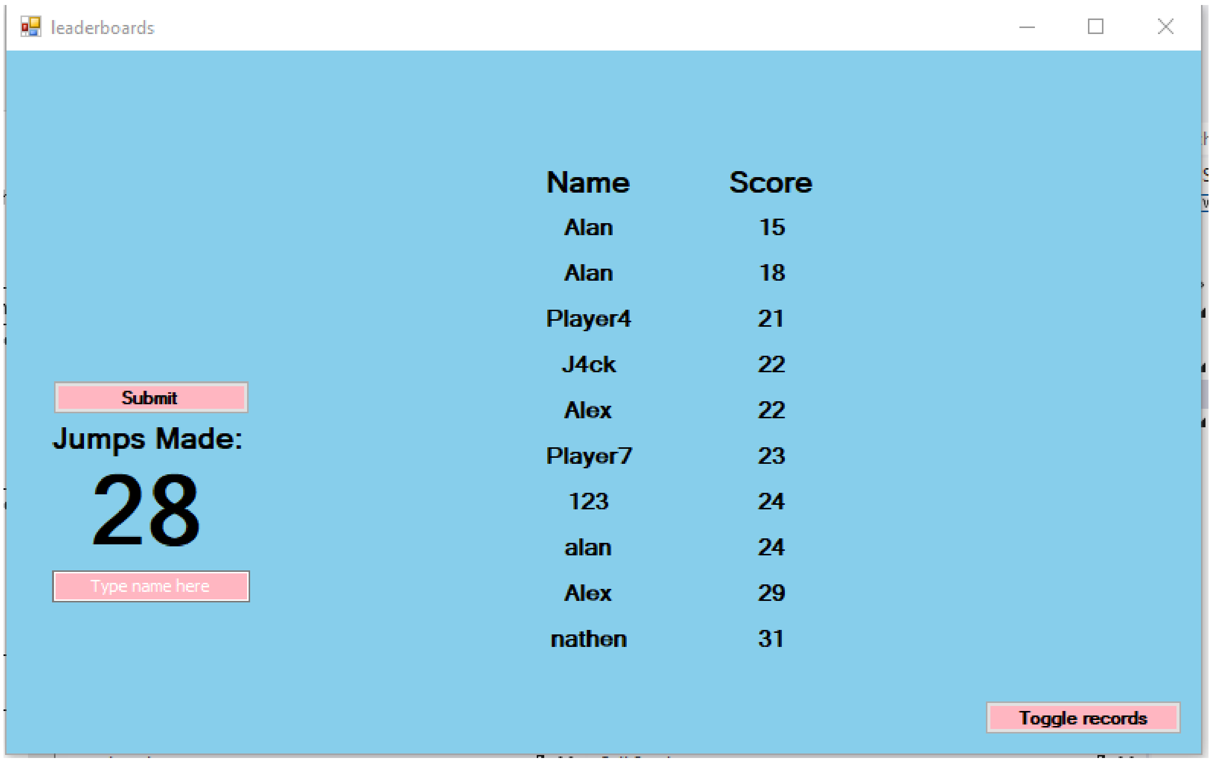
Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

LB.3.4



Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

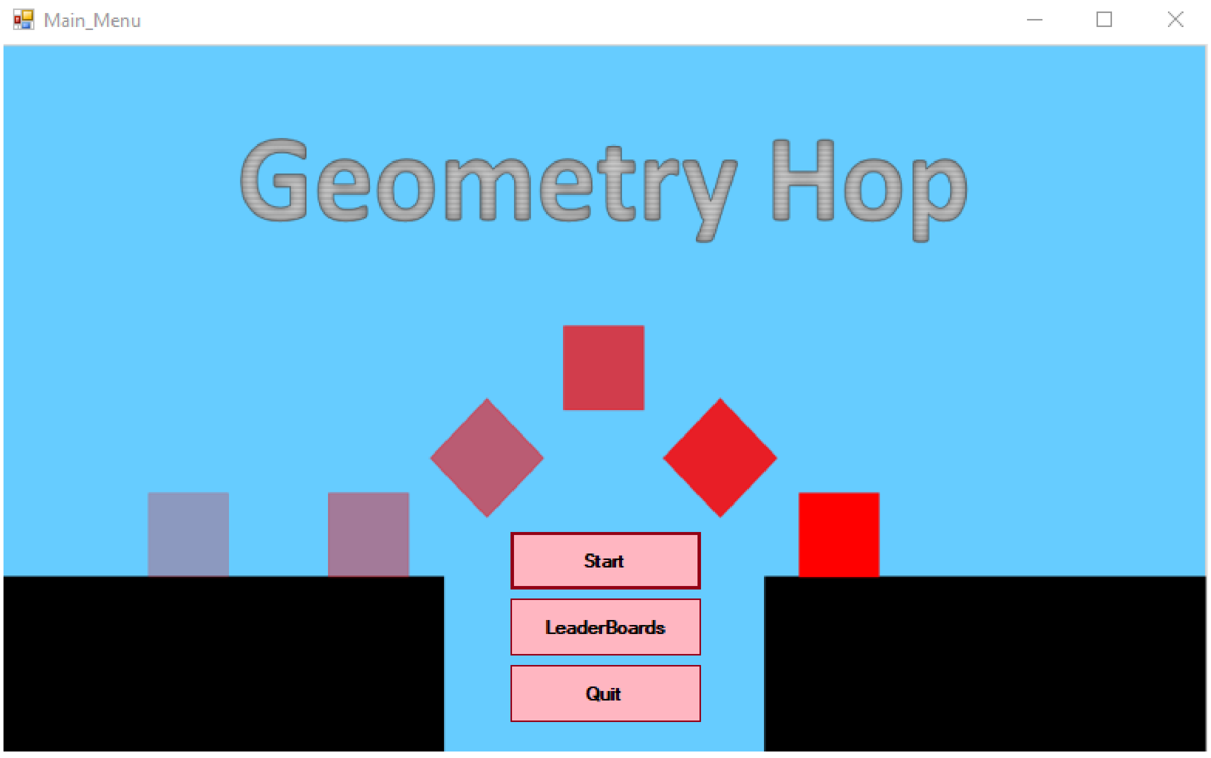
LB.3.5



leaderboards

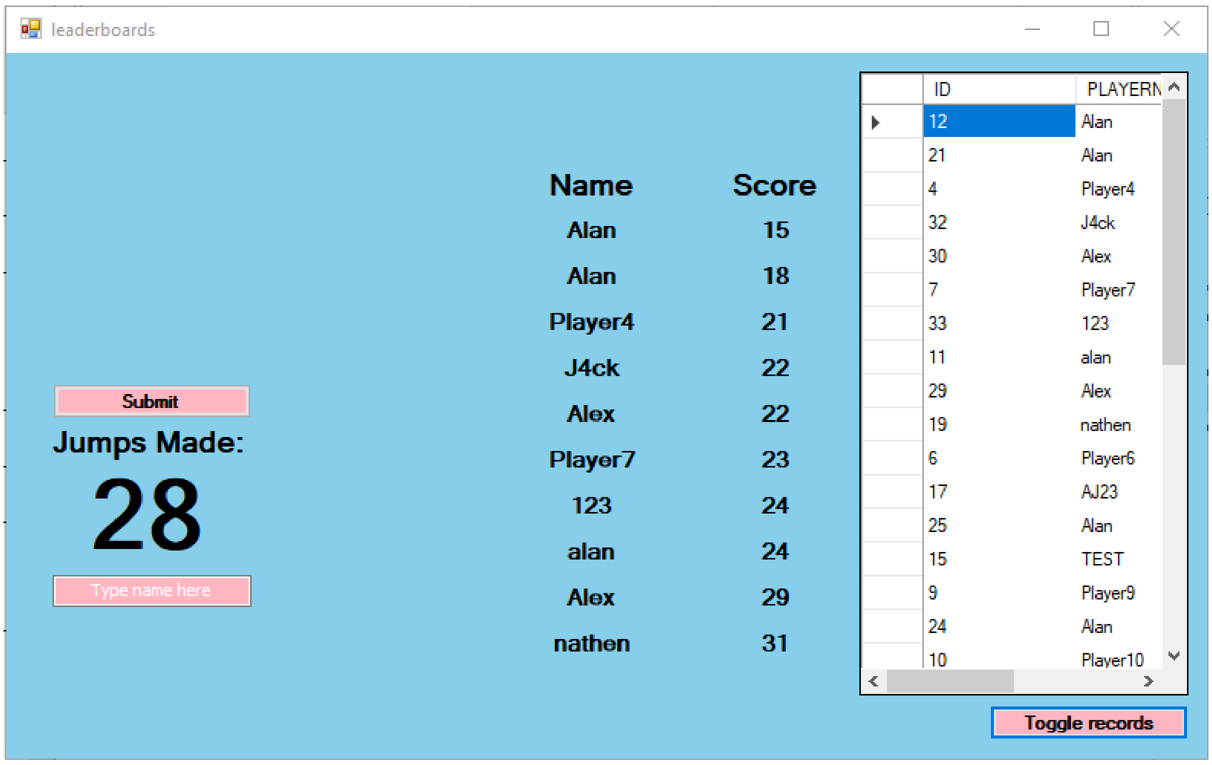
Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

LB.4



This was opened after a record was submitted

LB.5

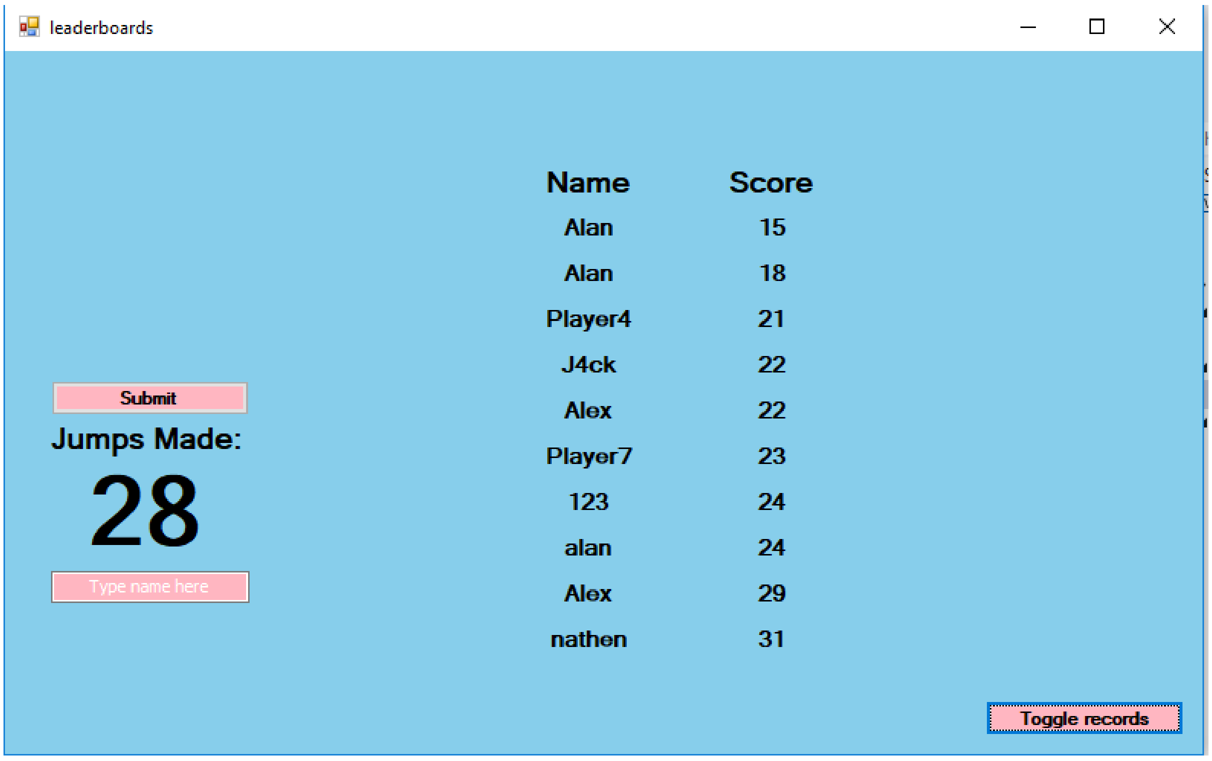


28

Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

ID	PLAYERN
12	Alan
21	Alan
4	Player4
32	J4ck
30	Alex
7	Player7
33	123
11	alan
29	Alex
19	nathen
6	Player6
17	AJ23
25	Alan
15	TEST
9	Player9
24	Alan
10	Player10

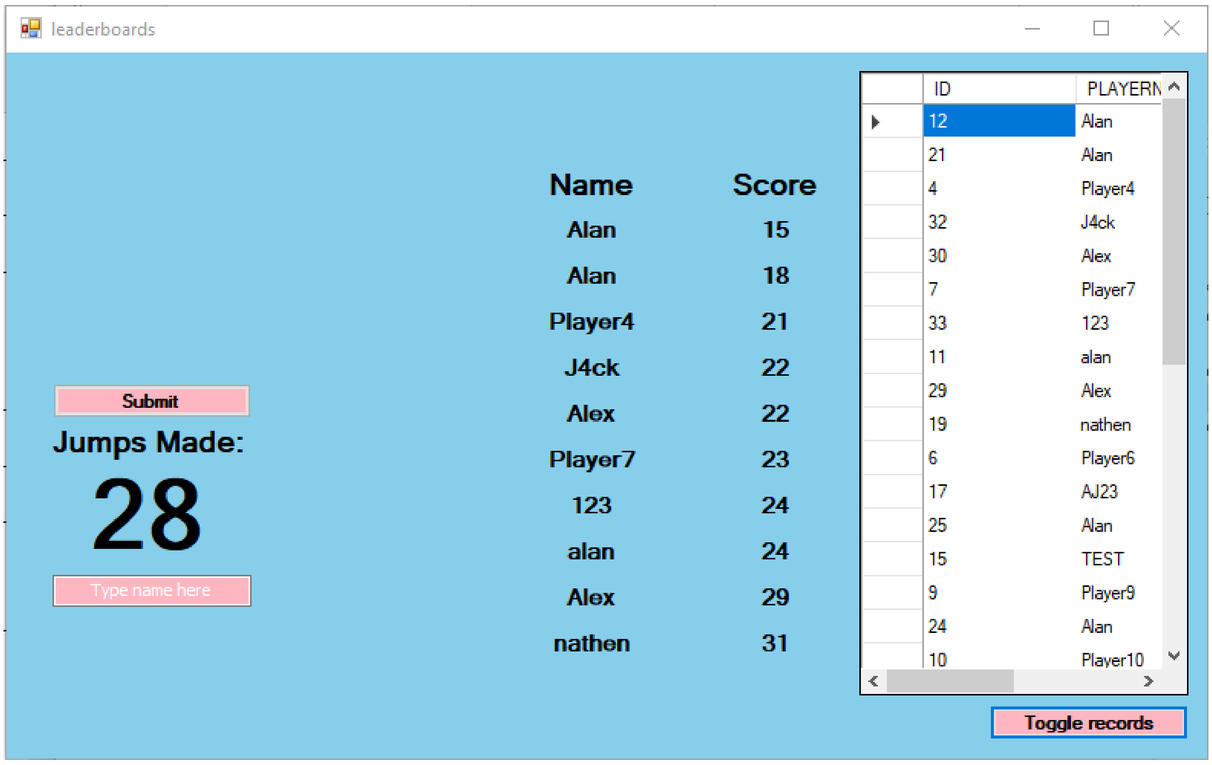
LB.6



28

Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

LB.7



Jumps Made:

28

Name	Score
Alan	15
Alan	18
Player4	21
J4ck	22
Alex	22
Player7	23
123	24
alan	24
Alex	29
nathen	31

ID	PLAYERN
12	Alan
21	Alan
4	Player4
32	J4ck
30	Alex
7	Player7
33	123
11	alan
29	Alex
19	nathen
6	Player6
17	AJ23
25	Alan
15	TEST
9	Player9
24	Alan
10	Player10

Below is the testing I carried out for the name validation in the leaderboard screen of my game. When developing my game, I made it so that the program only accepts the user's name if their name is only comprised of numbers and letters, as well as being 3 to 15 characters in length.

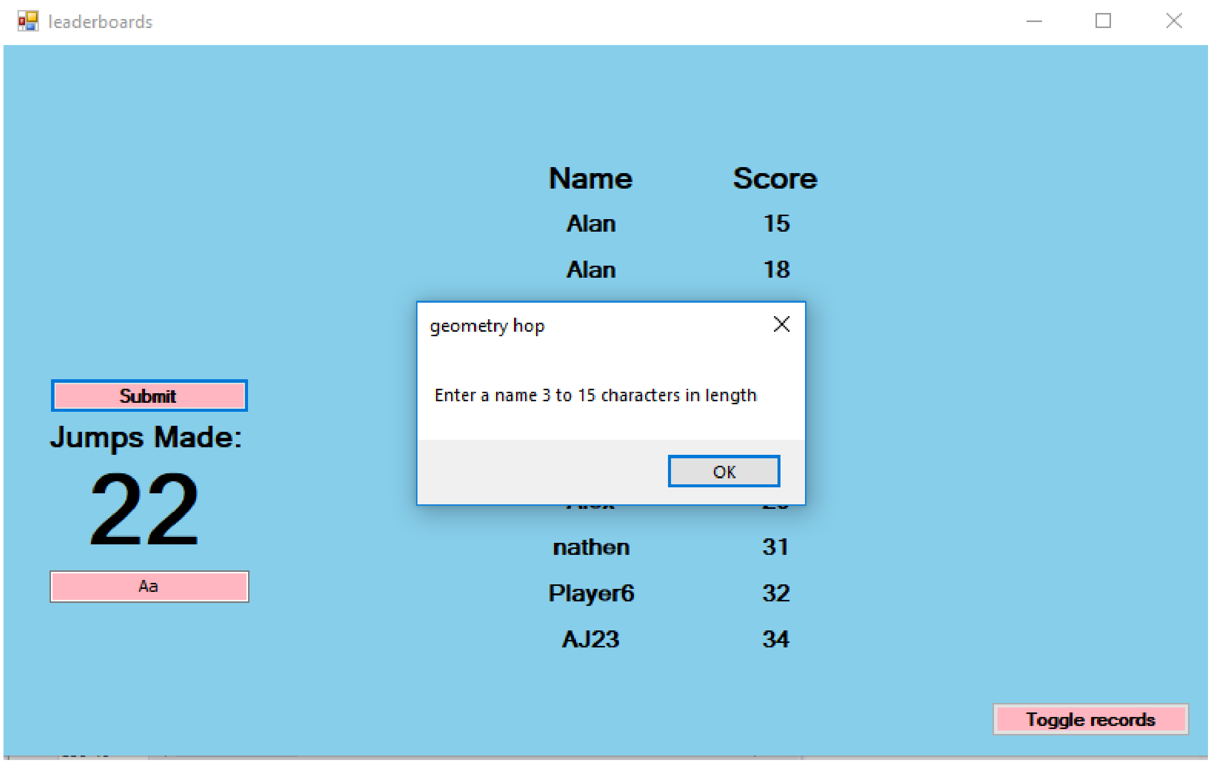
(LB.v Test No)

Data input	Test No.	Expected outcome	Actual outcome
Aa (2 characters in length)	LB.v1	Program should not accept this as a name and should return an error message saying that the name is not valid.	Same as expected outcome
0123456789ABCDEF (16 characters in length)	LB.v2	Program should not accept this as a name and should return an error message saying that the name is not valid.	Same as expected outcome
J@ck (4 characters; 3 letters and 1 symbol)	LB.v3	Program should not accept this as a name and should return an error message saying that the name is not valid.	Same as expected outcome
J4ck (4 characters; 3 letters and 1 digit)	LB.v4	Program should accept this as a name and should store this name into the database	Same as expected outcome
JacK (4 characters; 4 letters)	LB.v5	Program should accept this as a name and should store this name into the database	Same as expected outcome
123 (3 characters; 3 digits)	LB.v6	Program should accept this as a name and should store this name into the database	Same as expected outcome
No data inputs made; press submit as soon as the leaderboard screen loads	LB.v7	Program should not accept this as a name and should return an error message saying that the name is not valid.	Same as expected outcome after a fix was made
0123456789ABCDE (15 characters in length)	LB.v8	Program should accept this as a name and should store this name into the database	Same as expected outcome

Initially, upon carrying out my tests I did find that there were some ways to break my program. For example, in test LB.v7, I did find that submitting a name without an empty name field did result in the game crashing. As a result of my findings, I made changes to my program and I have documented this test in the video.

Test Results for data validation

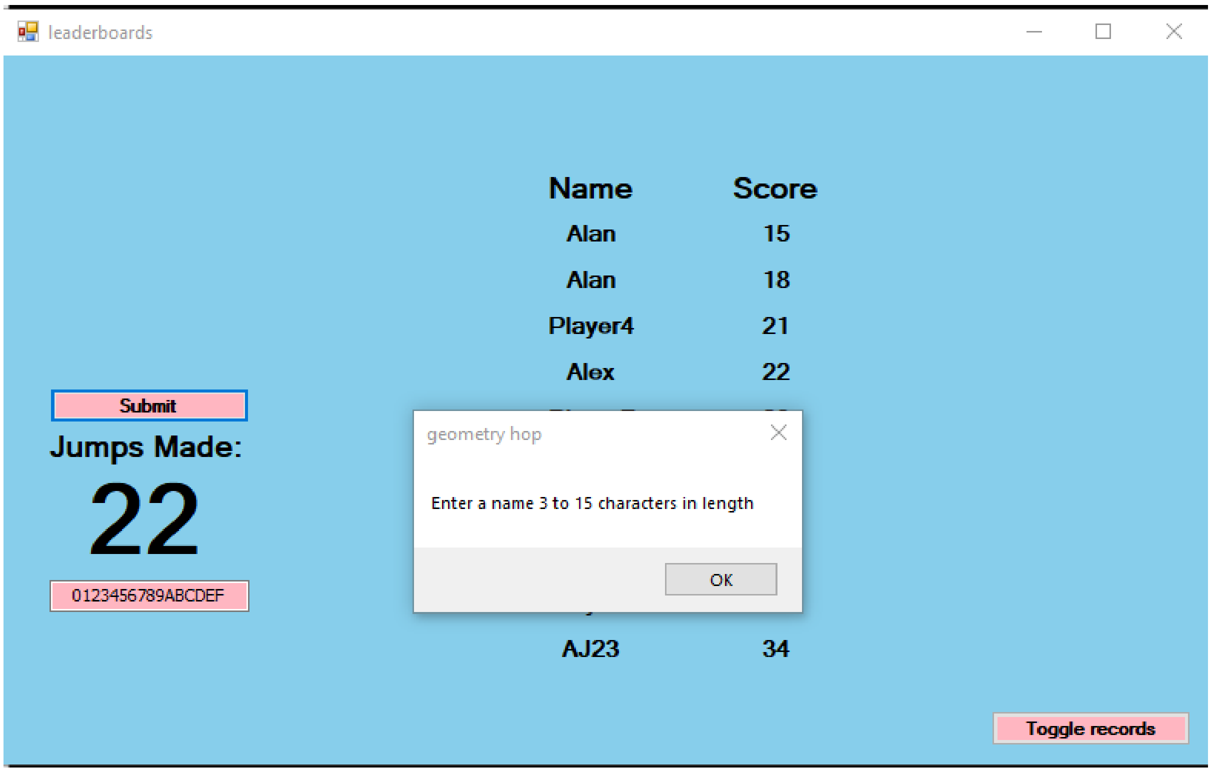
LB.v1



Name	Score
Alan	15
Alan	18
nathen	31
Player6	32
AJ23	34

Enter a name 3 to 15 characters in length

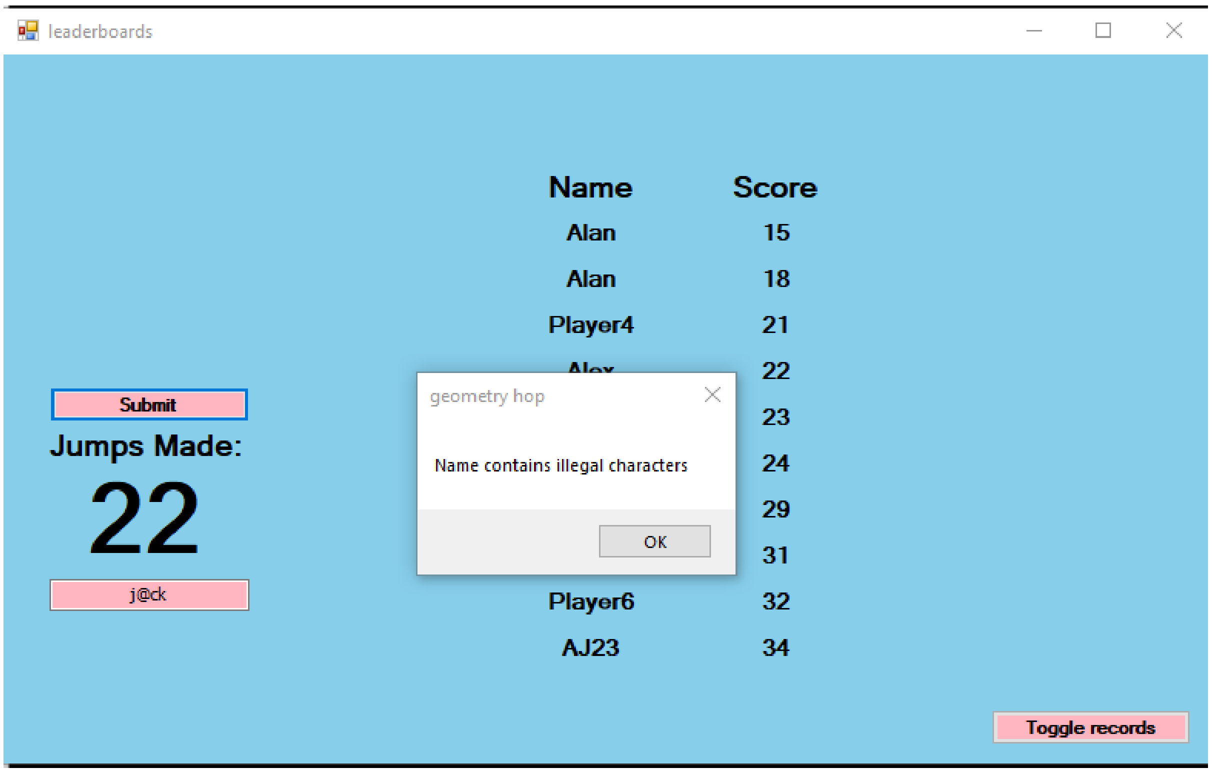
LB.v2



Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
AJ23	34

Enter a name 3 to 15 characters in length

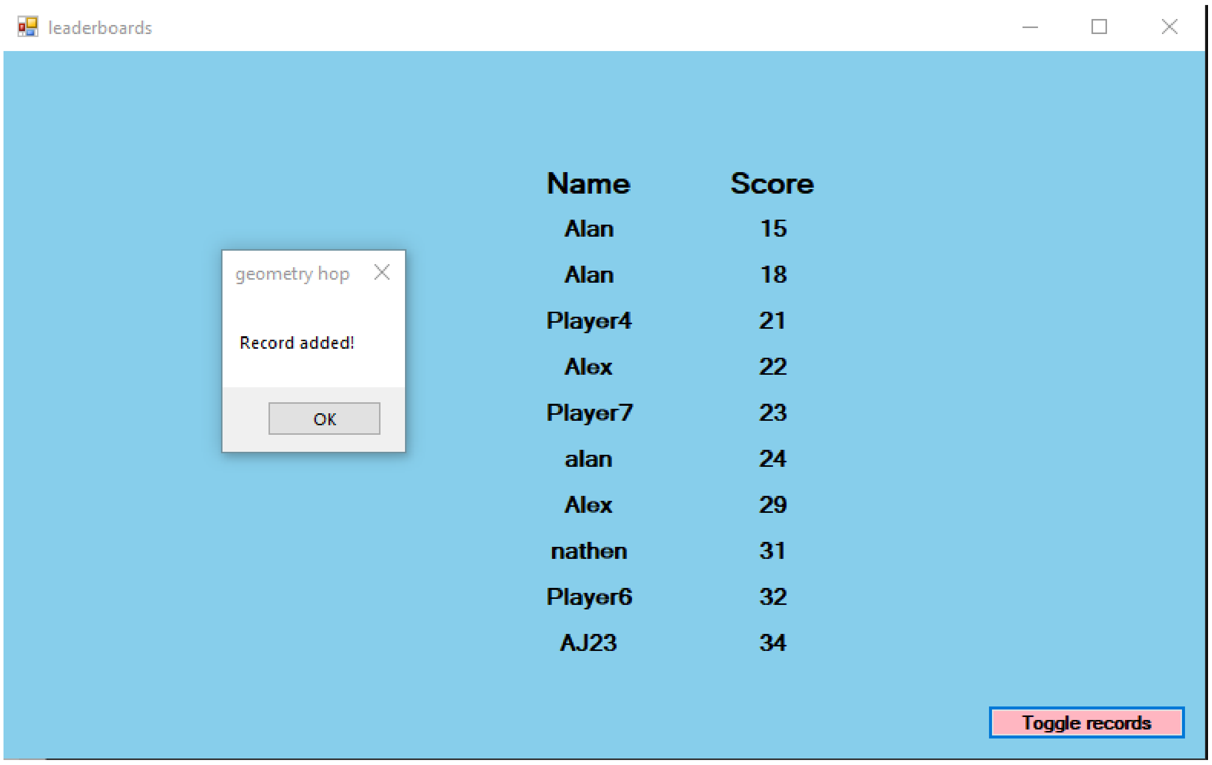
LB.v3



Name contains illegal characters

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
	23
	24
	29
	31
Player6	32
AJ23	34

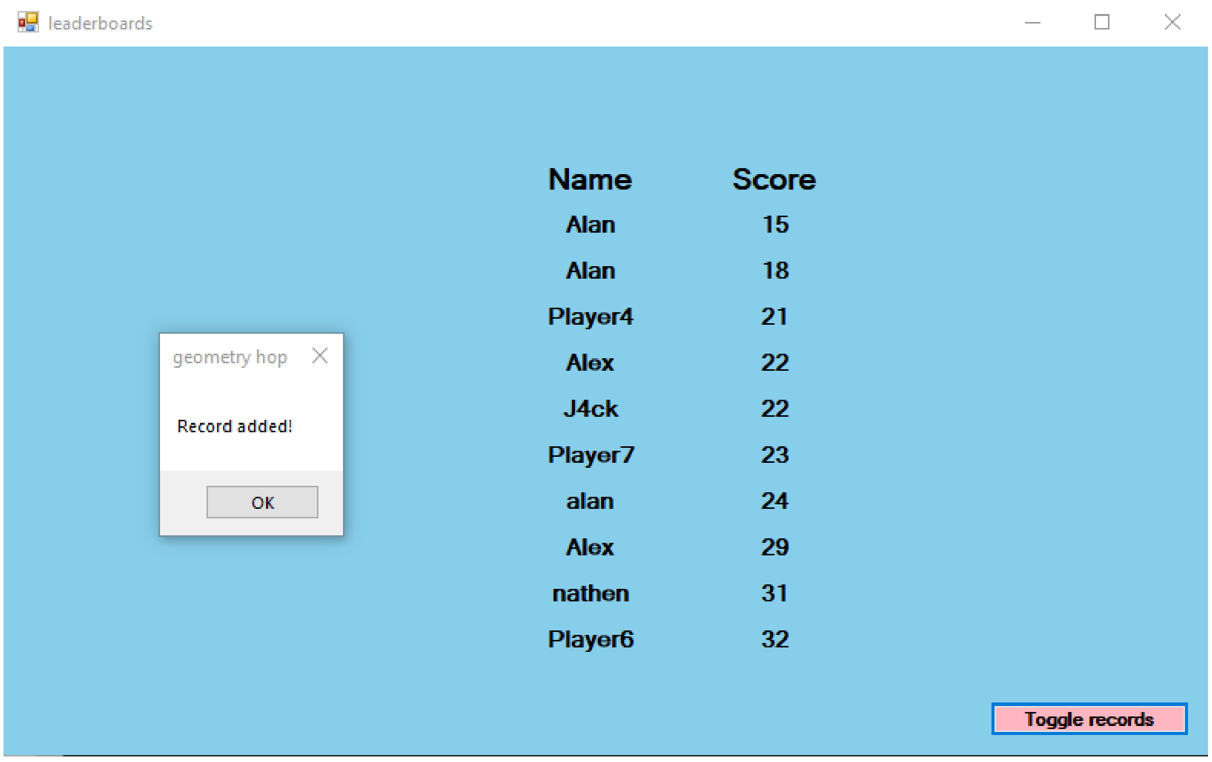
LB.v4



Record added!

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
Player7	23
alan	24
Alex	29
nathen	31
Player6	32
AJ23	34

LB.v5



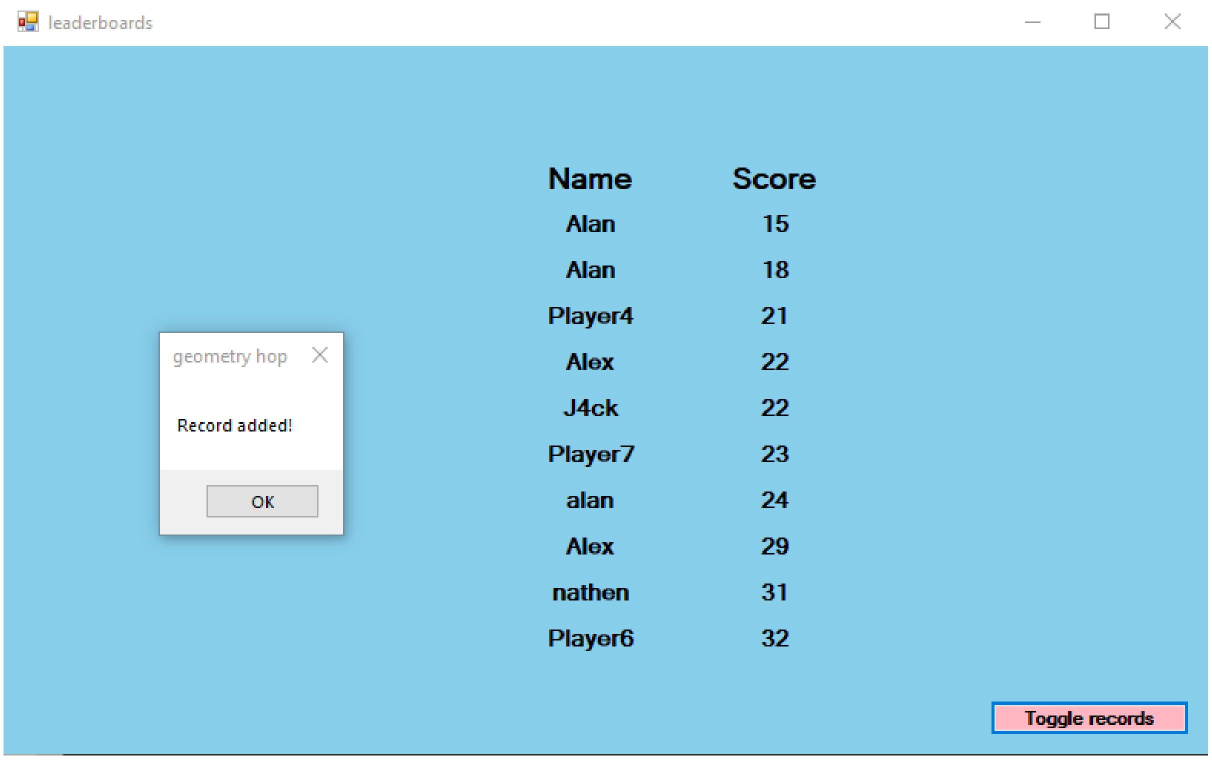
leaderboards

geometry hop

Record added!

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
J4ck	22
Player7	23
alan	24
Alex	29
nathen	31
Player6	32

LB.v6



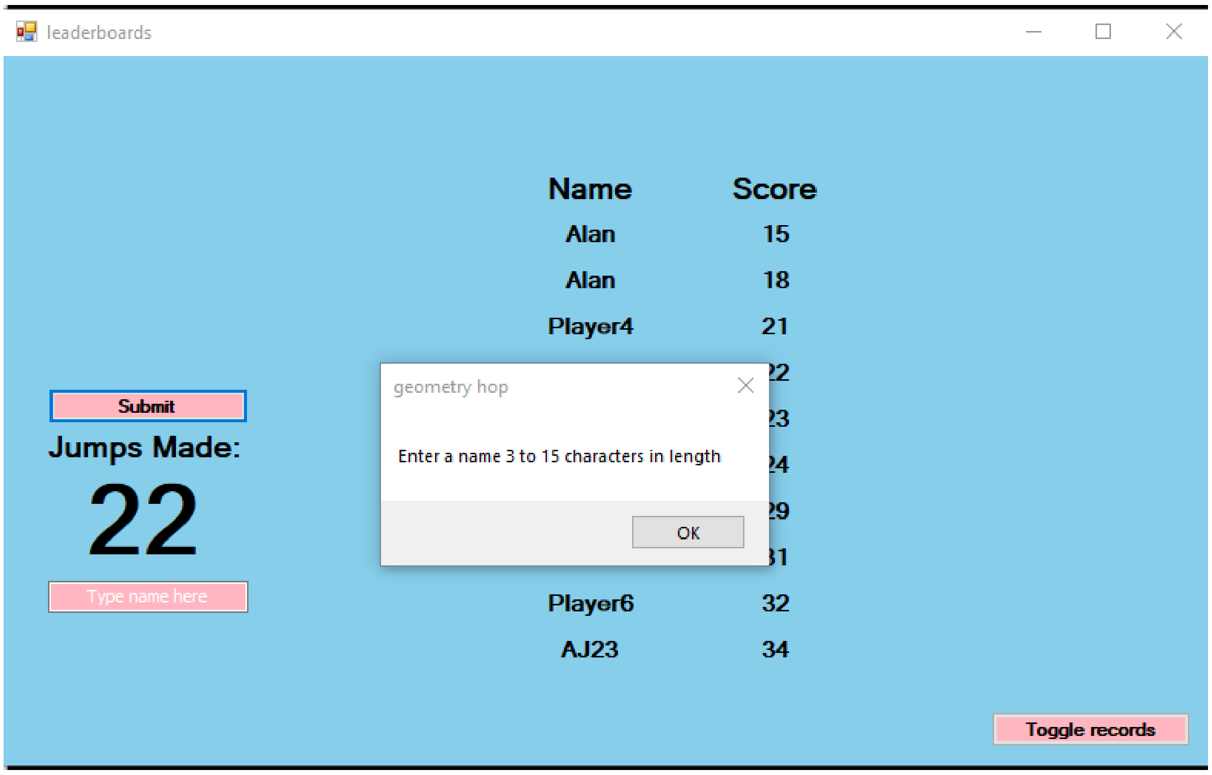
leaderboards

geometry hop

Record added!

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
J4ck	22
Player7	23
alan	24
Alex	29
nathen	31
Player6	32

LB.v7


leaderboards

Name	Score
Alan	15
Alan	18
Player4	21
	22
	23
	24
	29
	31
Player6	32
AJ23	34

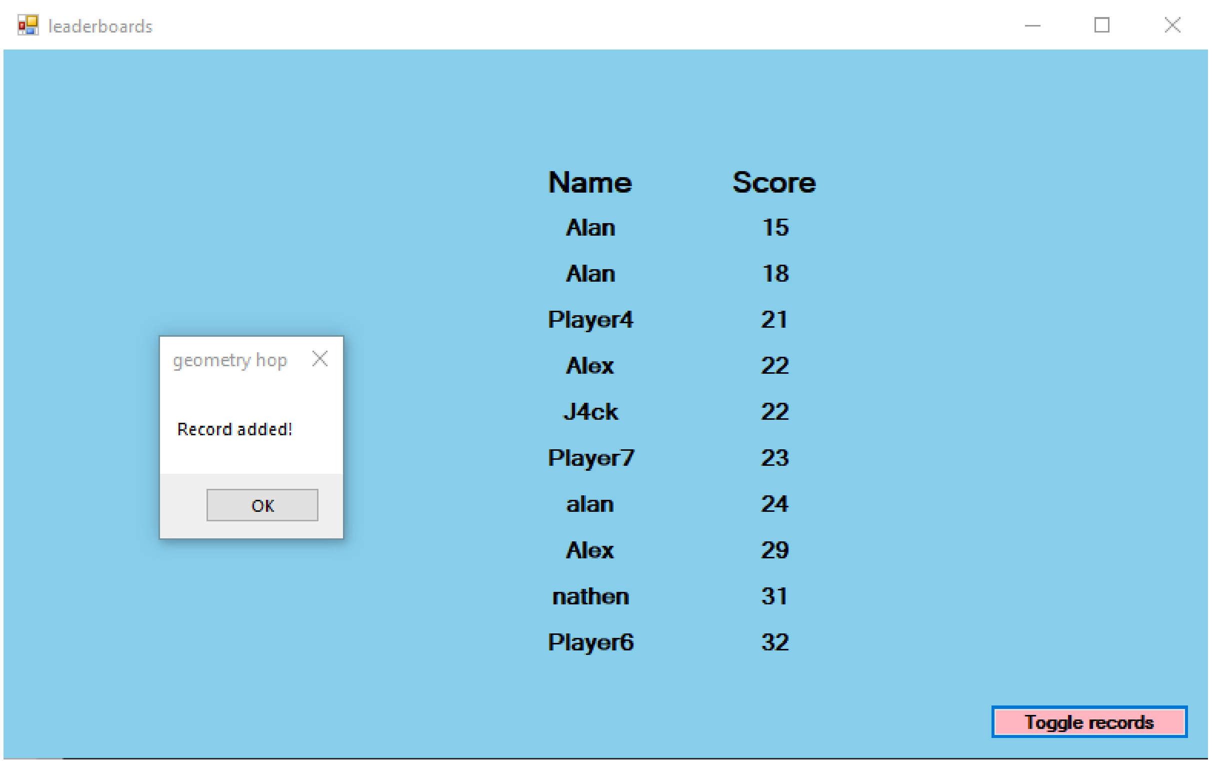
geometry hop

Enter a name 3 to 15 characters in length

Jumps Made:

22

LB.v8



Record added!

Name	Score
Alan	15
Alan	18
Player4	21
Alex	22
J4ck	22
Player7	23
alan	24
Alex	29
nathen	31
Player6	32

Testing the main menu (MM Test No)

Test description	Test No.	Expected outcome(s)	Actual outcome
Press the Play button	MM.1	The game should open and the user should be loaded into the game, the main menu will be hidden	Same as expected outcome
Press the Leaderboards button	MM.2	The program should open the leaderboards menu, which will open the leaderboards screen. However, the user should not be able to add any records to the database; the user is only allowed to view the leaderboards. This process should also hide the main menu.	Same as expected outcome
Press the Quit button	MM.3	The game should close itself	Same as expected outcome

Testing the game form (G. Test No)

Test description	Test No.	Expected outcome(s)	Actual outcome
The game should be able to read textfiles and draw them correctly	G.1.1	**Reading a valid textfile** The game should be able to read and draw whatever is contained in the textfile	Same as expected outcome
	G.1.2	**Reading a broken/corrupt textfile** The game should tell the user that the textfile for the game is corrupt and it should not draw the level.	Same as expected outcome

Test that the game's sidescroll mechanic works	G.2	When the game is playing, all pictureboxes but the player's should be moved in one direction at a constant rate	Same as expected outcome
Test that the game inputs are working	G.3.1	Pressing the spacebar should initiate a jump in the game, the player should remain in a locked position in the x axis	Same as expected outcome
	G.3.2	Pressing "P" should stop the game (when the game is running)	Same as expected outcome
	G.3.3	Pressing "C" should resume the game (when it is paused)	Same as expected outcome
Test the game's jump counter should work	G.4.1	Counter should only increment if the user initiates a jump	Same as expected outcome
	G.4.2	Counter should not increment if the user initiates a jump when the game is paused	Same as expected outcome
Test the game's level counter	G.5.1	Level counter should be set to 1 when the user first starts to play the game	Same as expected outcome
	G.5.2	Level counter should only increment when the user completes a level, and the next level is loaded	Same as expected outcome
Testing collisions and respawning	G.6.1	Falling off the map respawns the player (unless the end of the level is reached)	Same as expected outcome
	G.6.2	Hitting a wall respawns the player	
Test that the game automatically loads in new levels	G.7	When the user finishes a level, the next one will load. The level counter will also increment when this happens	Same as expected outcome
Test that the player does not sink through the floor	G.8	The player should sit on top of the floor if a jump is not initiated by the user	Same as expected outcome
Test that the gravity in my game works	G.9.1	The player should fall back to the floor eventually	Same as expected outcome
	G.9.2	The player should begin to fall if they traverse off a raised platform	Same as expected outcome

Proof of this can be seen in the videos I have made of my prototype

Videos: Red are of my old prototype, Green are of the second iteration of my prototype.

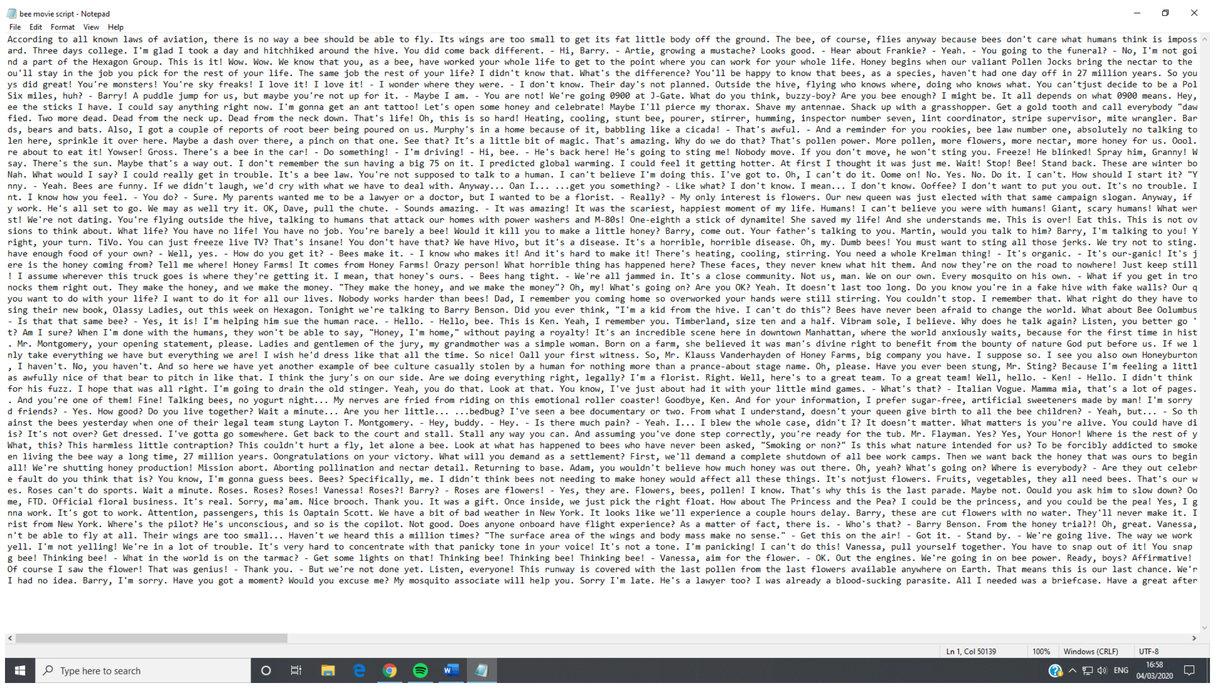
- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)
- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

Testing the game's file reading abilities (TR. Test No)

In order to properly test my game's textfile reading ability, I have made a number of different textfiles which it could read, all of which will provide different results. With each textfile, I have given my expected outcome and I have recorded the actual outcome of reading the text file too. The textfiles which I have used can also be found attached below.

Test description	Test No.	Expected outcome(s)	Actual outcome

Reading an erroneous texfile (Textfiles will be shown below)	TR.1. 1	**Reading the Bee Movie script (Figure 32)** My game should not be able to read this and as a result return an error message, while also not drawing anything to the game form	Same as expected outcome
	TR.1.2	**Reading a level textfile where the values are stored in the wrong format** My game should not be able to read this and as a result return an error message, while also not drawing anything to the game form	Same as expected outcome
	TR.1.3	**Reading an empty textfile** My game should be able to read this but will return an error message saying that there are not enough components to draw to the form	Same as expected outcome
Reading a textfile which has a non-existent file path	TR.2	**I will be using the filepath of "non-existent.filepath"** My game should not load or draw anything and should not crash	Same as expected outcome after some fixes were mad
Reading a correctly written textfile	TR.3	The game should read this with no problems, and as a result draw the contents of the textfile to the game screen as well as the level counter and jump counter	Same as expected outcome



```
According to all known laws of aviation, there is no way a bee should be able to fly. Its wings are too small to get its fat little body off the ground. The bee, of course, flies anyway because bees don't care what humans think is imposs
ard. Three days college. I'm glad I took a day and hitchhiked around the hive. You did come back different. - Hi, Barry. - Artie, growing a mustache? Looks good. - Hear about Frankie? - Yeah. - You going to the funeral? - No, I'm not goi
nd a part of the Hexagon Group. This is it! Wow. Wow. We know that you, as a bee, have worked your whole life to get to the point where you can work for your whole life. Honey begins when our valiant Pollen Jocks bring the nectar to the
ou'll stay in the job you pick for the rest of your life. The same job the rest of your life? I didn't know that. What's the difference? You'll be happy to know that bees, as a species, haven't had one day off in 27 million years. So you
ys did great! You're monsters! You're sky freaks! I love it! I love it! - I wonder where they were. - I don't know. Their day's not planned. Outside the hive, flying who knows where, doing who knows what. You can'tjust decide to be a Pol
Six miles, huh? - Barry! A puddle jump for us, but maybe you're not up for it. - Maybe I am. - You are not! We're going 0900 at J-Gate. What do you think, buzzy-boy? Are you bee enough? I might be. It all depends on what 0900 means. Hey,
ee the sticks I have. I could say anything right now. I'm gonna get an ant tattoo! Let's open some honey and celebrate! Maybe I'll pierce my thorax. Shave my antennae. Shack up with a grasshopper. Get a gold tooth and call everybody "daw
fied. Two more dead. Dead from the neck up. Dead from the neck down. That's life! Oh, this is so hard! Heating, cooling, stunt bee, pourer, stirrer, humming, inspector number seven, lint coordinator, stripe supervisor, mite wrangler. Bar
ds, bears and bats. Also, I got a couple of reports of root beer being poured on us. Murphy's in a home because of it, babbling like a cicada! - That's awful. - And a reminder for you rookies, bee law number one, absolutely no talking to
len here, sprinkle it over here. Maybe a dash over there, a pinch on that one. See that? It's a little bit of magic. That's amazing. Why do we do that? That's pollen power. More pollen, more flowers, more nectar, more honey for us. Oool.
re about to eat it! Yowser! Gross. There's a bee in the car! - Do something! - I'm driving! - Hi, bee. - He's back here! He's going to sting me! Nobody move. If you don't move, he won't sting you. Freeze! He blinked! Spray him, Granny! W
say. There's the sun. Maybe that's a way out. I don't remember the sun having a big 75 on it. I predicted global warming. I could feel it getting hotter. At first I thought it was just me. Wait! Stop! Bee! Stand back. These are winter bo
Nah. What would I say? I could really get in trouble. It's a bee law. You're not supposed to talk to a human. I can't believe I'm doing this. I've got to. Oh, I can't do it. Oome on! No. Yes. No. Do it. I can't. How should I start it? "Y
nny. - Yeah. Bees are funny. If we didn't laugh, we'd cry with what we have to deal with. Anyway... Oan I... ...get you something? - Like what? I don't know. I mean... I don't know. Ooffee? I don't want to put you out. It's no trouble. I
nt. I know how you feel. - You do? - Sure. My parents wanted me to be a lawyer or a doctor, but I wanted to be a florist. - Really? - My only interest is flowers. Our new queen was just elected with that same campaign slogan. Anyway, if
y work. He's all set to go. We may as well try it. OK, Dave, pull the chute. - Sounds amazing. - It was amazing! It was the scariest, happiest moment of my life. Humans! I can't believe you were with humans! Giant, scary humans! What wer
st! We're not dating. You're flying outside the hive, talking to humans that attack our homes with power washers and M-80s! One-eighth a stick of dynamite! She saved my life! And she understands me. This is over! Eat this. This is not ov
sions to think about. What life? You have no life! You have no job. You're barely a bee! Would it kill you to make a little honey? Barry, come out. Your father's talking to you. Martin, would you talk to him? Barry, I'm talking to you! Y
right, your turn. TiVo. You can just freeze live TV? That's insane! You don't have that? We have Hivo, but it's a disease. It's a horrible, horrible disease. Oh, my. Dumb bees! You must want to sting all those jerks. We try not to sting.
have enough food of your own? - Well, yes. - How do you get it? - Bees make it. - I know who makes it! And it's hard to make it! There's heating, cooling, stirring. You need a whole Krelman thing! - It's organic. - It's our-ganic! It's j
ere is the honey coming from? Tell me where! Honey Farms! It comes from Honey Farms! Orazy person! What horrible thing has happened here? These faces, they never knew what hit them. And now they're on the road to nowhere! Just keep still
! I assume wherever this truck goes is where they're getting it. I mean, that honey's ours. - Bees hang tight. - We're all jammed in. It's a close community. Not us, man. We on our own. Every mosquito on his own. - What if you get in tro
nocks them right out. They make the honey, and we make the money. "They make the honey, and we make the money"? Oh, my! What's going on? Are you OK? Yeah. It doesn't last too long. Do you know you're in a fake hive with fake walls? Our q
you want to do with your life? I want to do it for all our lives. Nobody works harder than bees! Dad, I remember you coming home so overworked your hands were still stirring. You couldn't stop. I remember that. What right do they have to
sing their new book, Olassy Ladies, out this week on Hexagon. Tonight we're talking to Barry Benson. Did you ever think, "I'm a kid from the hive. I can't do this"? Bees have never been afraid to change the world. What about Bee Oolumbus
- Is that that same bee? - Yes, it is! I'm helping him sue the human race. - Hello. - Hello, bee. This is Ken. Yeah, I remember you. Timberland, size ten and a half. Vibram sole, I believe. Why does he talk again? Listen, you better go '
t? Am I sure? When I'm done with the humans, they won't be able to say, "Honey, I'm home," without paying a royalty! It's an incredible scene here in downtown Manhattan, where the world anxiously waits, because for the first time in hist
. Mr. Montgomery, your opening statement, please. Ladies and gentlemen of the jury, my grandmother was a simple woman. Born on a farm, she believed it was man's divine right to benefit from the bounty of nature God put before us. If we l
nly take everything we have but everything we are! I wish he'd dress like that all the time. So nice! Oall your first witness. So, Mr. Klauss Vanderhayden of Honey Farms, big company you have. I suppose so. I see you also own Honeyburton
, I haven't. No, you haven't. And so here we have yet another example of bee culture casually stolen by a human for nothing more than a prance-about stage name. Oh, please. Have you ever been stung, Mr. Sting? Because I'm feeling a littl
as awfully nice of that bear to pitch in like that. I think the jury's on our side. Are we doing everything right, legally? I'm a florist. Right. Well, here's to a great team. To a great team! Well, hello. - Ken! - Hello. I didn't think
for his fuzz. I hope that was all right. I'm going to drain the old stinger. Yeah, you do that. Look at that. You know, I've just about had it with your little mind games. - What's that? - Italian Vogue. Mamma mia, that's a lot of pages.
. And you're one of them! Fine! Talking bees, no yogurt night... My nerves are fried from riding on this emotional roller coaster! Goodbye, Ken. And for your information, I prefer sugar-free, artificial sweeteners made by man! I'm sorry
d friends? - Yes. How good? Do you live together? Wait a minute... Are you her little... ...bedbug? I've seen a bee documentary or two. From what I understand, doesn't your queen give birth to all the bee children? - Yeah, but... - So th
ainst the bees yesterday when one of their legal team stung Layton T. Montgomery. - Hey, buddy. - Hey. - Is there much pain? - Yeah. I... I blew the whole case, didn't I? It doesn't matter. What matters is you're alive. You could have di
is? It's not over? Get dressed. I've gotta go somewhere. Get back to the court and stall. Stall any way you can. And assuming you've done step correctly, you're ready for the tub. Mr. Flayman. Yes? Yes, Your Honor! Where is the rest of y
What, this? This harmless little contraption? This couldn't hurt a fly, let alone a bee. Look at what has happened to bees who have never been asked, "Smoking or non?" Is this what nature intended for us? To be forcibly addicted to smoke
en living the bee way a long time, 27 million years. Oongratulations on your victory. What will you demand as a settlement? First, we'll demand a complete shutdown of all bee work camps. Then we want back the honey that was ours to begin
all! We're shutting honey production! Mission abort. Aborting pollination and nectar detail. Returning to base. Adam, you wouldn't believe how much honey was out there. Oh, yeah? What's going on? Where is everybody? - Are they out celebr
e fault do you think that is? You know, I'm gonna guess bees. Bees? Specifically, me. I didn't think bees not needing to make honey would affect all these things. It's notjust flowers. Fruits, vegetables, they all need bees. That's our w
es. Roses can't do sports. Wait a minute. Roses. Roses? Roses! Vanessa! Roses?! Barry? - Roses are flowers! - Yes, they are. Flowers, bees, pollen! I know. That's why this is the last parade. Maybe not. Oould you ask him to slow down? Oo
me, FTD. Official floral business. It's real. Sorry, ma'am. Nice brooch. Thank you. It was a gift. Once inside, we just pick the right float. How about The Princess and the Pea? I could be the princess, and you could be the pea! Yes, I g
nna work. It's got to work. Attention, passengers, this is Oaptain Scott. We have a bit of bad weather in New York. It looks like we'll experience a couple hours delay. Barry, these are cut flowers with no water. They'll never make it. I
rist from New York. Where's the pilot? He's unconscious, and so is the copilot. Not good. Does anyone onboard have flight experience? As a matter of fact, there is. - Who's that? - Barry Benson. From the honey trial?! Oh, great. Vanessa,
n't be able to fly at all. Their wings are too small... Haven't we heard this a million times? "The surface area of the wings and body mass make no sense." - Get this on the air! - Got it. - Stand by. - We're going live. The way we work
yell. I'm not yelling! We're in a lot of trouble. It's very hard to concentrate with that panicky tone in your voice! It's not a tone. I'm panicking! I can't do this! Vanessa, pull yourself together. You have to snap out of it! You snap
g bee! Thinking bee! - What in the world is on the tarmac? - Get some lights on that! Thinking bee! Thinking bee! Thinking bee! - Vanessa, aim for the flower. - OK. Out the engines. We're going in on bee power. Ready, boys? Affirmative!
Of course I saw the flower! That was genius! - Thank you. - But we're not done yet. Listen, everyone! This runway is covered with the last pollen from the last flowers available anywhere on Earth. That means this is our last chance. We'r
I had no idea. Barry, I'm sorry. Have you got a moment? Would you excuse me? My mosquito associate will help you. Sorry I'm late. He's a lawyer too? I was already a blood-sucking parasite. All I needed was a briefcase. Have a great after
```

Figure 32: The bee movie script which will be used as part of my testing stage for my game (Test TR.1.1)

```
zero
zero
four-thousand
five-hundred
picSky
SkyBlue
Sky
====================
two-thousand-five-hundre
d
five-hundred
one-thousand-five-hundre
d
one-hundred
picNextLevel
SkyBlue
End
====================
zero
five-hundred
two-thusand-five-hundred
one-hundred
picVoid
Lime
Void
====================
one-hundred-and-fifty
two-hundred-and-twenty-f
ive
fifty
fifty
picPlayer
Red
Player
====================
zero
three-hundred
five-hundred-and-fifty
two-hundred
picFloor1
Black
Floor
====================
eight-hundred
two-hundred-and-fifty
five-hundred-and-fifty
two-hundred-and-fifty
picFloor2
Black
Floor
====================
seven-hundred-and-ninety
two-hundred-and-sixty
ten
two-hundred-and-forty
picBarrierFloor2
Transparent
Void
====================
one-thousand-three-hundr
ed-and-fifty
three-hundred-and-fifty
five-hundred-and-fifty
one-hundred-and-fifty
picFloor3
Black
Floor
====================
one-thousand-nine-hundre
d
one-hundred-and-ninety-f
ive
six-hundred
three-hundred-and-five
picFloor4
Black
Floor
====================
one-thousand-and-ninety
two-hundred-and-five
ten
one-hundred-and-forty-fi
ve
picBarrierFloor4
Transparent
Void
====================
```

Figure 33: above is the textfile which I will be reading for test TR.1.2, it is exactly the same as level 1 on page but the integer values have been converted to strings

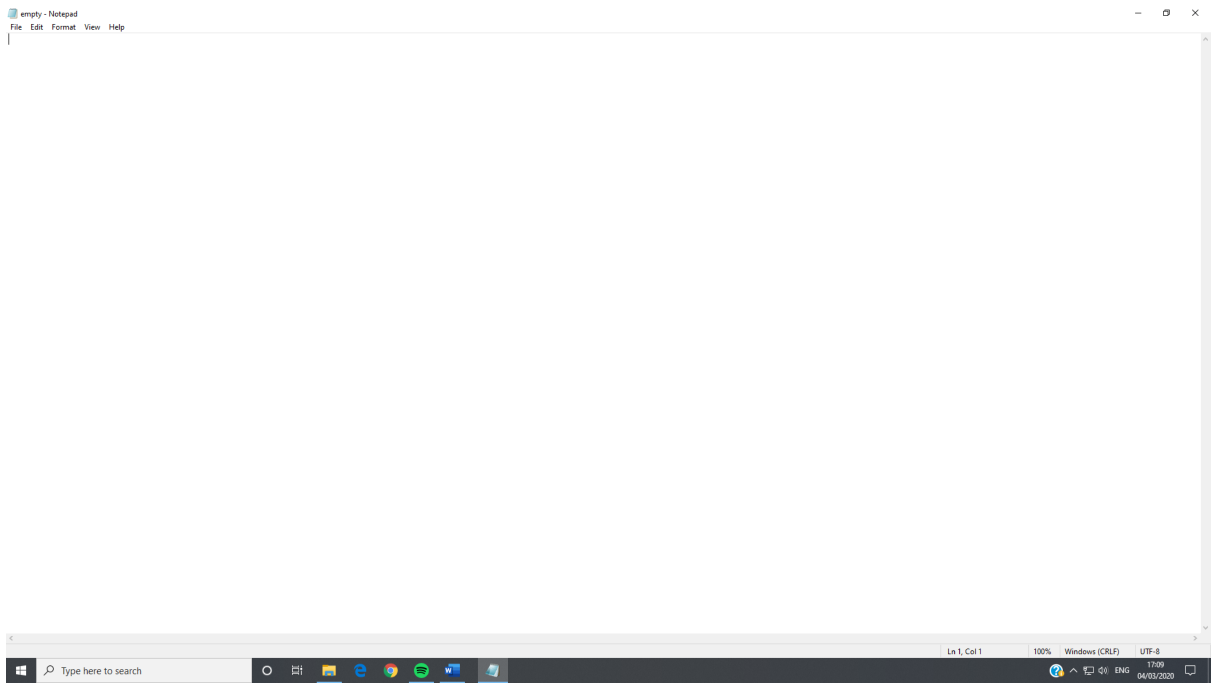
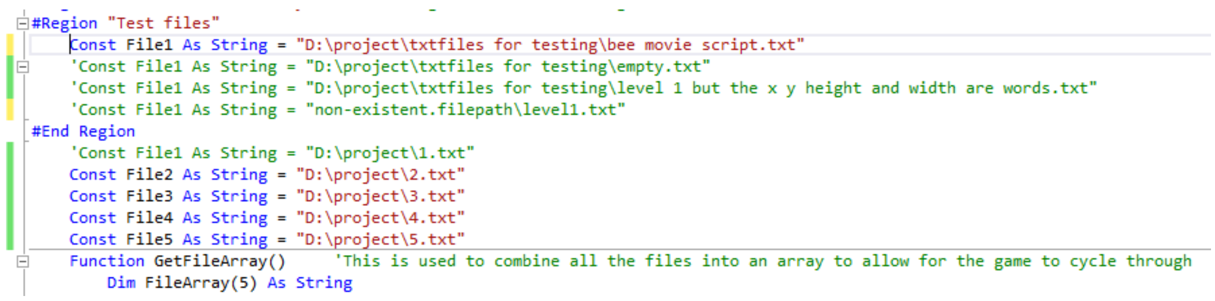


Figure 34: The empty textfile which will be used during testing (for test TR.1.3)

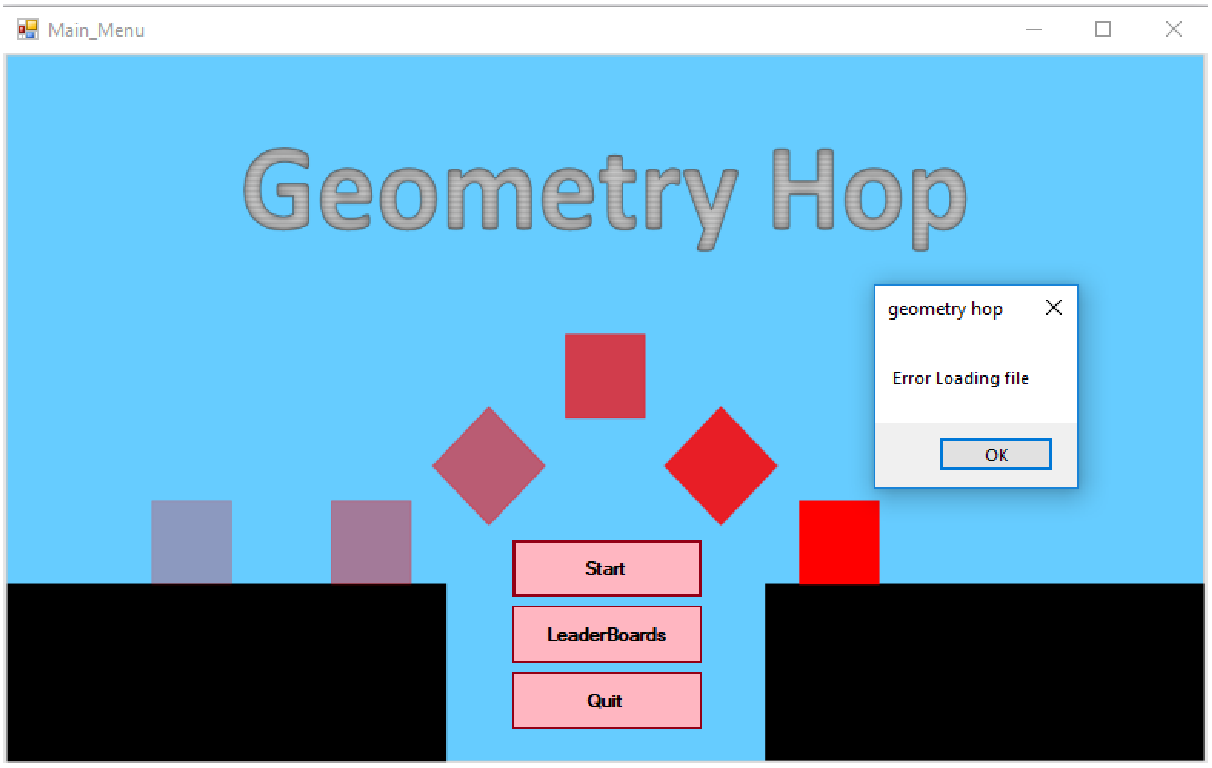
Test Results for the game’s file reading ability

<u>Test TR.1.1</u>



```
#Region "Test files"
    Const File1 As String = "D:\project\txtfiles for testing\bee movie script.txt"
    'Const File1 As String = "D:\project\txtfiles for testing\empty.txt"
    'Const File1 As String = "D:\project\txtfiles for testing\level 1 but the x y height and width are words.txt"
    'Const File1 As String = "non-existent.filepath\level1.txt"
#End Region
    'Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to combine all the files into an array to allow for the game to cycle through
        Dim FileArray(5) As String
```

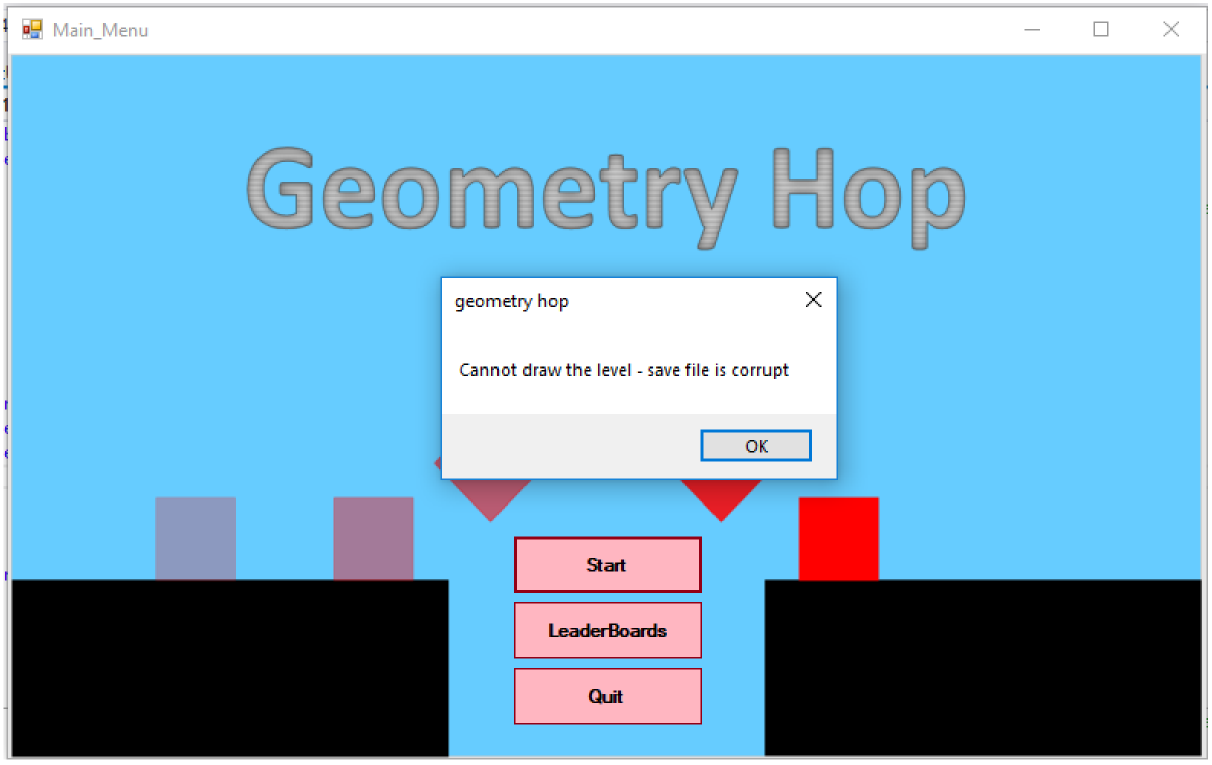
In order to test the game’s ability to read a non-related textfile to my game and to investigate the outcome of doing this, I have commented out the actual location of File1 and changed it so that the program looks for the directory of `"D:\project\txtfiles for testing\bee movie script.txt"` which will look for the bee-movie script as seen in Figure 32.



geometry hop

Error Loading file

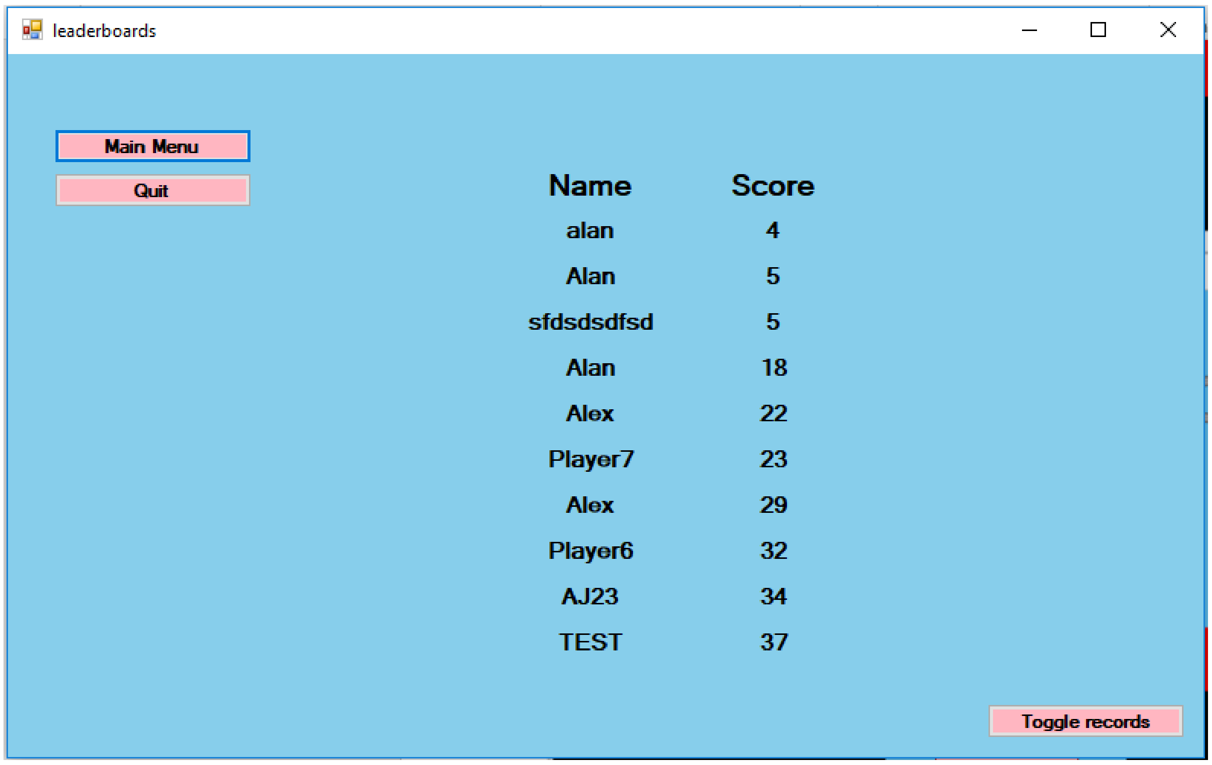
OK



geometry hop

Cannot draw the level - save file is corrupt

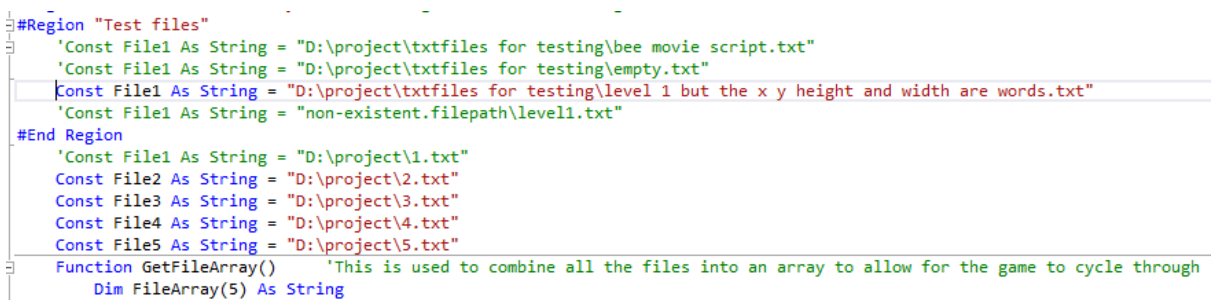
OK



Name	Score
alan	4
Alan	5
sfdsdsdfsd	5
Alan	18
Alex	22
Player7	23
Alex	29
Player6	32
AJ23	34
TEST	37

When my game opens the bee movie script, it prompts the user with two message boxes which state that the there was an error reading the file, then the user is told that the game cannot draw the file. After this, the game then opens the leaderboards screen, as expected in my testing table.

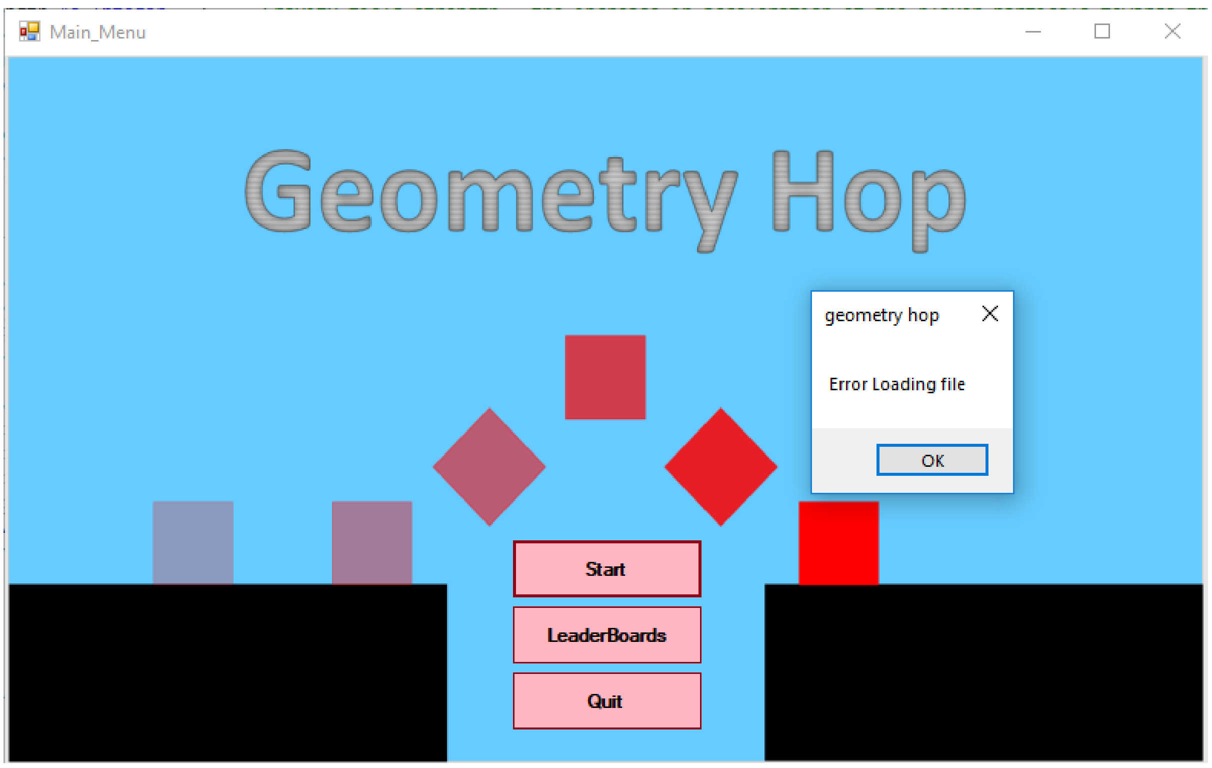
Test TR.1.2



```
#Region "Test files"
    'Const File1 As String = "D:\project\txtfiles for testing\bee movie script.txt"
    'Const File1 As String = "D:\project\txtfiles for testing\empty.txt"
    Const File1 As String = "D:\project\txtfiles for testing\level 1 but the x y height and width are words.txt"
    'Const File1 As String = "non-existent.filepath\level1.txt"
#End Region
    'Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to combine all the files into an array to allow for the game to cycle through
        Dim FileArray(5) As String
```

In order to test the game's ability to read a level textfile where the values are stored in the wrong format and to investigate the outcome of doing this, I have commented out the actual location of File1 and changed it so that the program looks for the directory of `"D:\project\txtfiles for testing\level 1 but the x y height and width are words.txt"` which will look for the bee-movie script as seen in Figure 33Figure 32.

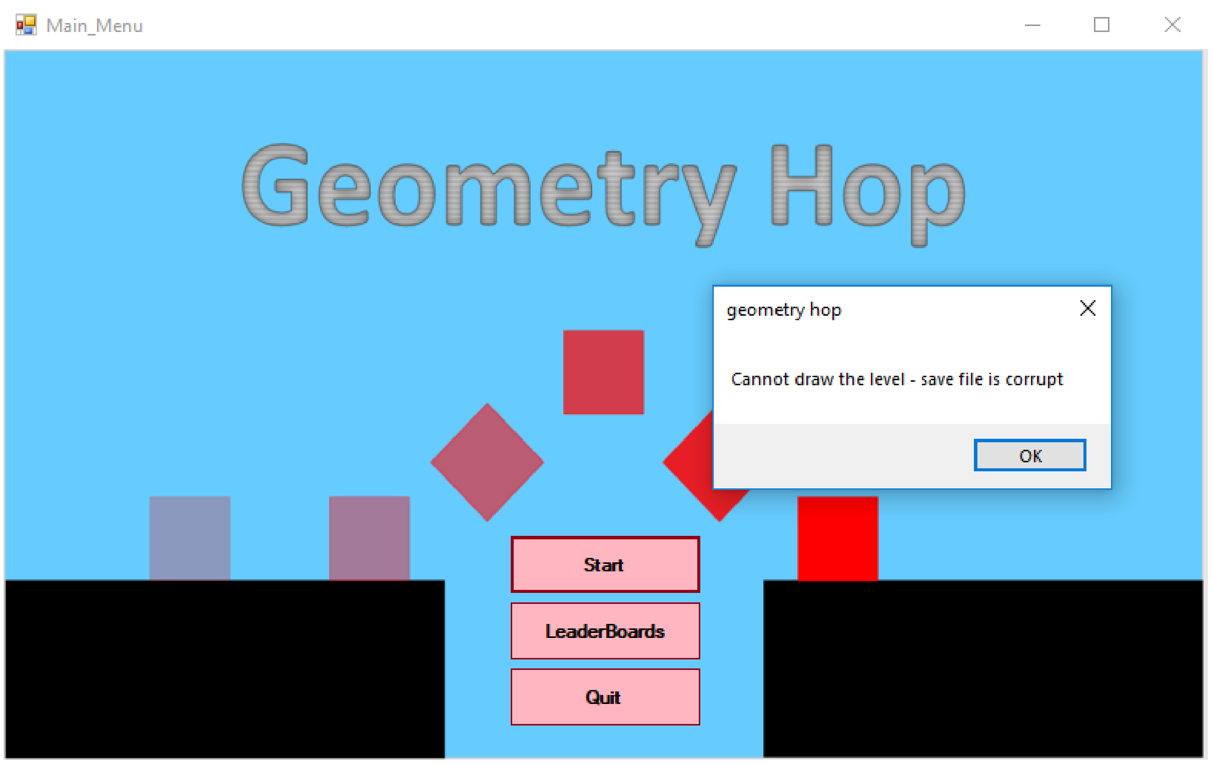
level textfile where the values are stored in the wrong format



geometry hop

Error Loading file

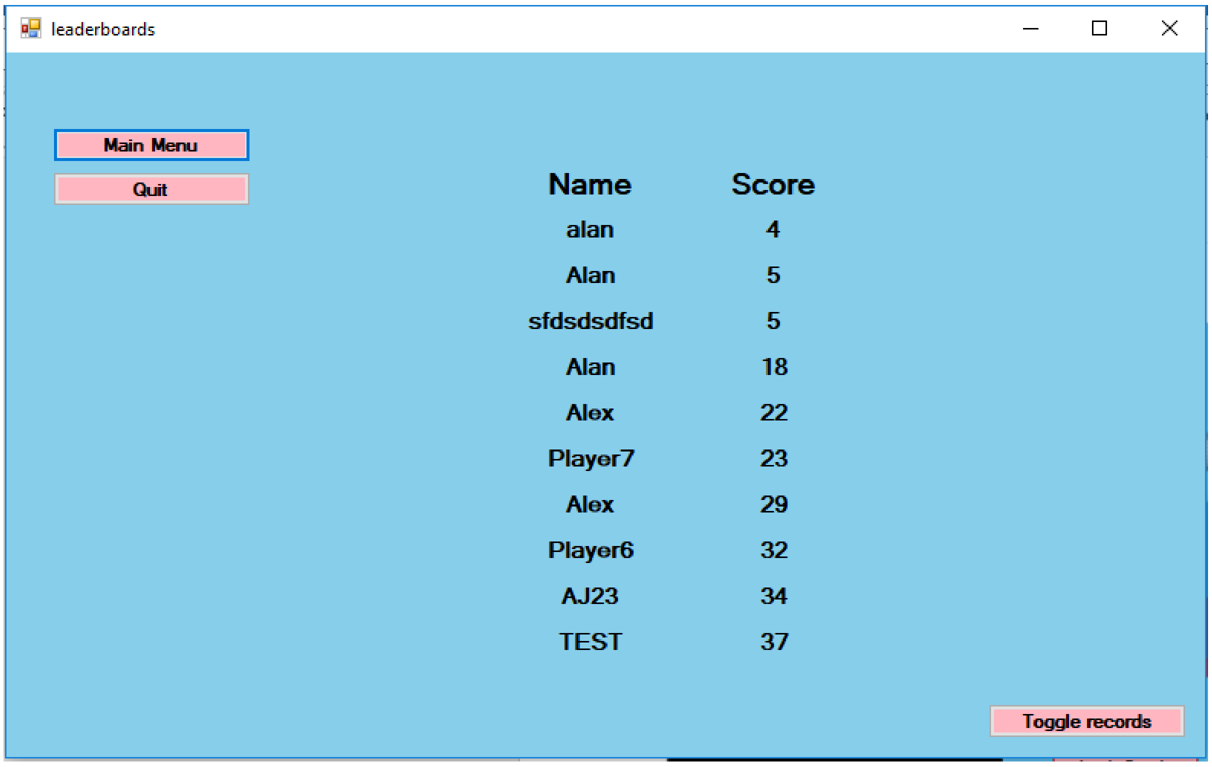
OK



geometry hop

Cannot draw the level - save file is corrupt

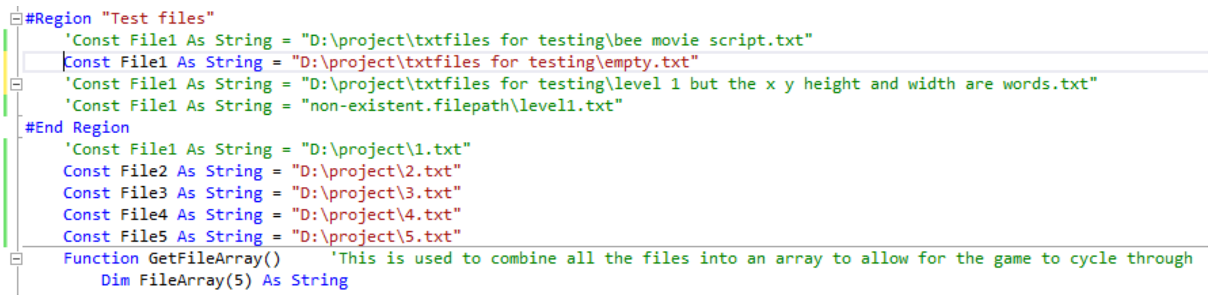
OK



Name	Score
alan	4
Alan	5
sfdsdsdfsd	5
Alan	18
Alex	22
Player7	23
Alex	29
Player6	32
AJ23	34
TEST	37

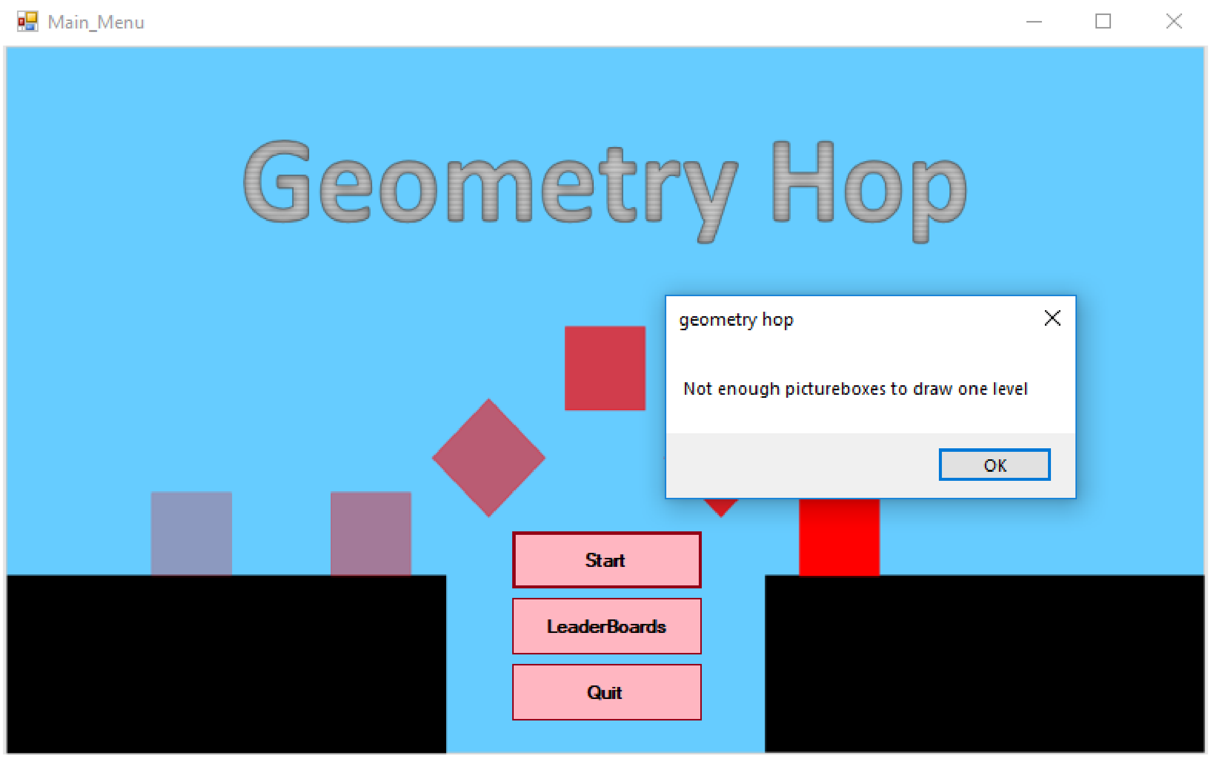
When my game opens the level textfile where the values are stored in the wrong format, it prompts the user with two message boxes which state that the there was an error reading the file, then the user is told that the game cannot draw the file. After this, the game then opens the leaderboards screen, as expected in my testing table.

Test TR.1.3



```
#Region "Test files"
    'Const File1 As String = "D:\project\txtfiles for testing\bee movie script.txt"
    Const File1 As String = "D:\project\txtfiles for testing\empty.txt"
    'Const File1 As String = "D:\project\txtfiles for testing\level 1 but the x y height and width are words.txt"
    'Const File1 As String = "non-existent.filepath\level1.txt"
#End Region
    'Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to combine all the files into an array to allow for the game to cycle through
        Dim FileArray(5) As String
```

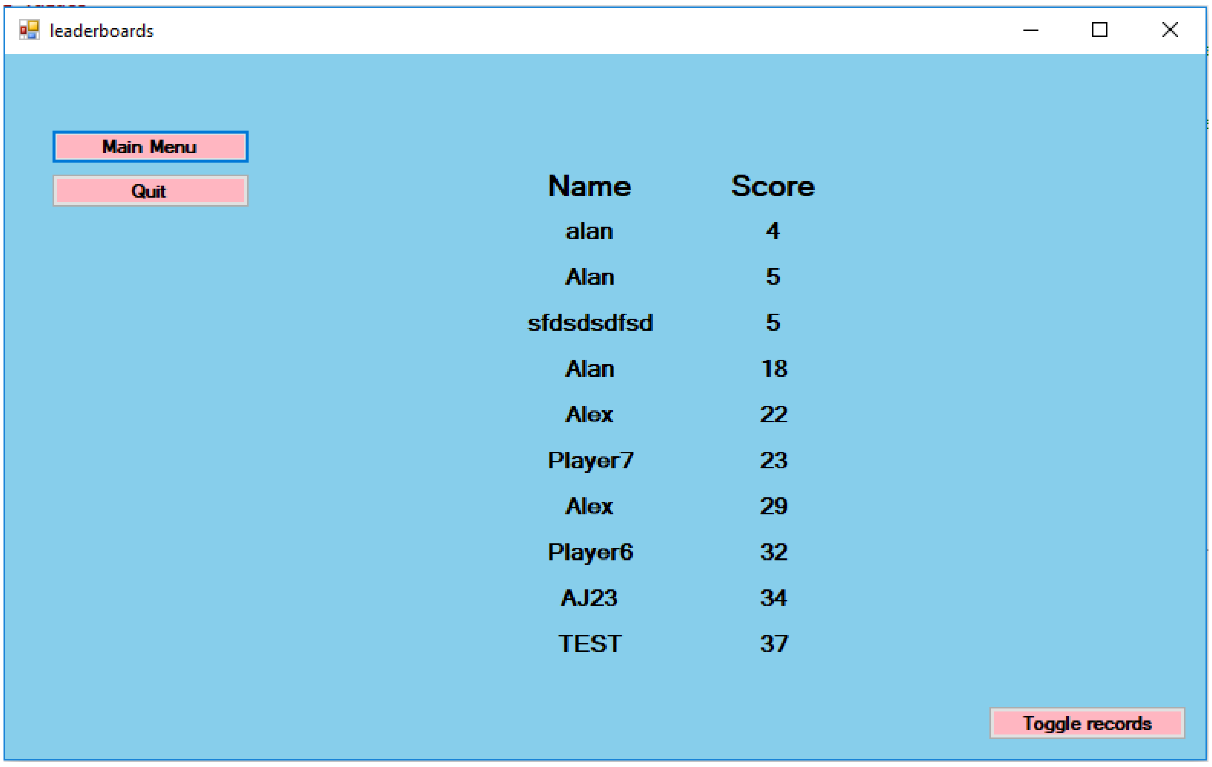
In order to test the game's ability to read an empty textfile to my game and to investigate the outcome of doing this, I have commented out the actual location of File1 and changed it so that the program looks for the directory of `"D:\project\txtfiles for testing\empty.txt"`which will look for the bee-movie script as seen in Figure 33Figure 32.



geometry hop

Not enough pictureboxes to draw one level

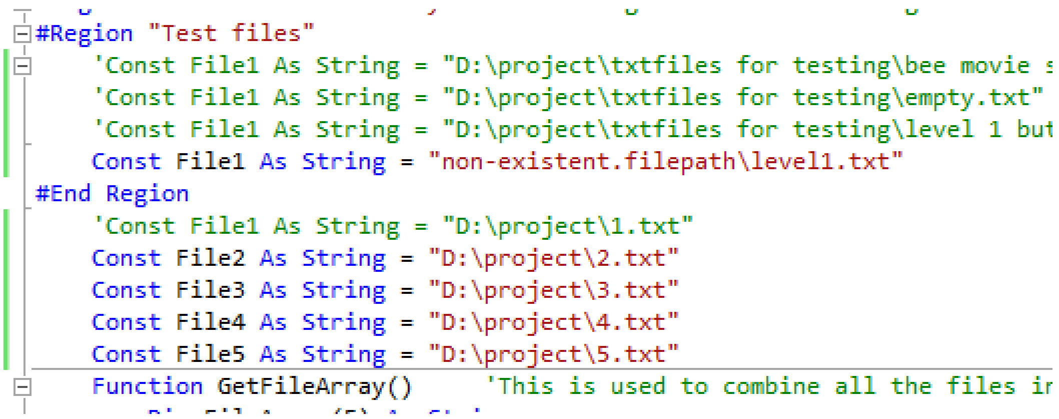
OK



Name	Score
alan	4
Alan	5
sfdsdsdfsd	5
Alan	18
Alex	22
Player7	23
Alex	29
Player6	32
AJ23	34
TEST	37

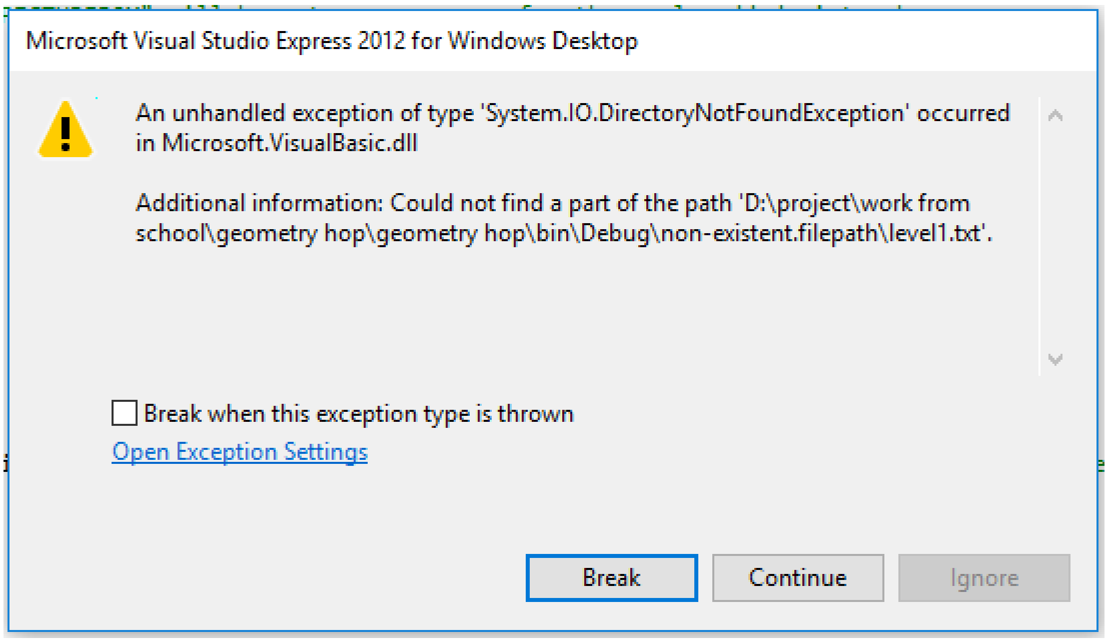
When my game opens an empty textfile, it will have no problem reading it so it will not produce a message box saying that there was an error reading the textfile. Instead, the program shows a message box which tells the user that there was not enough components to make a level, and thus does not start a game, but instead takes the user to the leaderboard screen. This outcome was the same as what I expected in my testing table.

TR.2



```
#Region "Test files"
    'Const File1 As String = "D:\project\txtfiles for testing\bee movie s
    'Const File1 As String = "D:\project\txtfiles for testing\empty.txt"
    'Const File1 As String = "D:\project\txtfiles for testing\level 1 but
    Const File1 As String = "non-existent.filepath\level1.txt"
#End Region
    'Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to combine all the files ir
```

In order to test the game's ability to look for a file not found, I have commented out the actual location of File1 and changed it so that the program looks for the directory of `"non-existent.filepath\level1.txt"`, when I tested this, the game crashed. This can be seen by the error message returned by visual basic (see Figure 35)



Microsoft Visual Studio Express 2012 for Windows Desktop

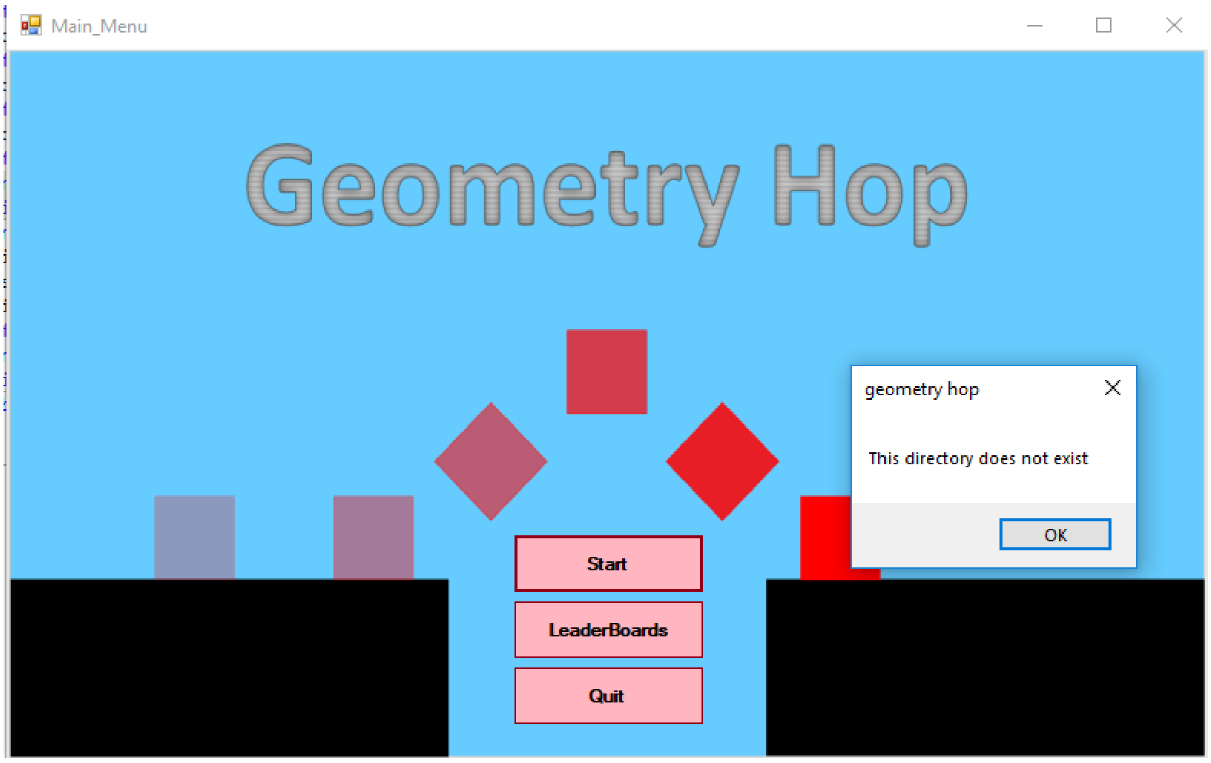
An unhandled exception of type 'System.IO.DirectoryNotFoundException' occurred in Microsoft.VisualBasic.dll

Additional information: Could not find a part of the path 'D:\project\work from school\geometry hop\geometry hop\bin\Debug\non-existent.filepath\level1.txt'.

Figure 35: Message from Visual basic which tells me that reading a textfile in a untraceable directory breaks my game

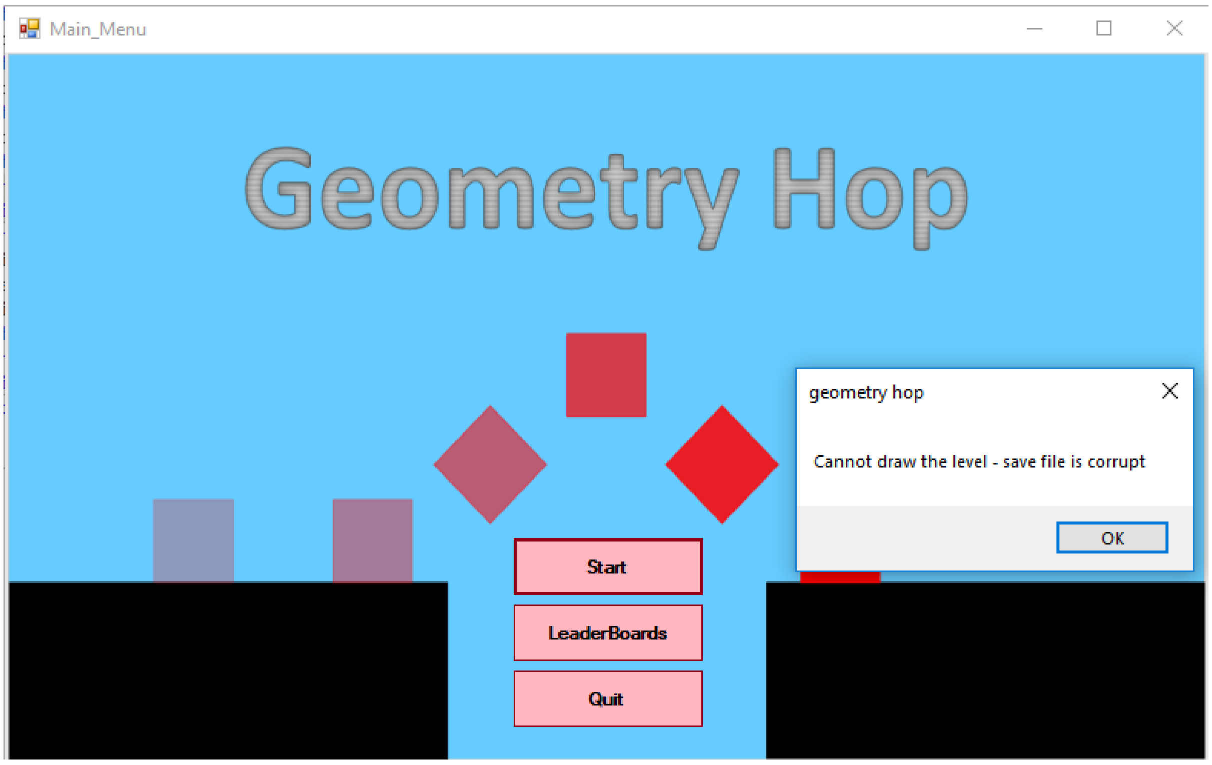
To make my game robust, I have decided that I should go back and fix this. In order to achieve this, I have added a few lines of code to my cLevelSetUp class which first checks that a file exists; it will then proceed to load the textfile if it is found; these changes have been applied to the **GetNoOfPictureboxes** function and the **GetLevelInfo** function

When I run the same test again, I get the results shown below (after pressing start).



geometry hop

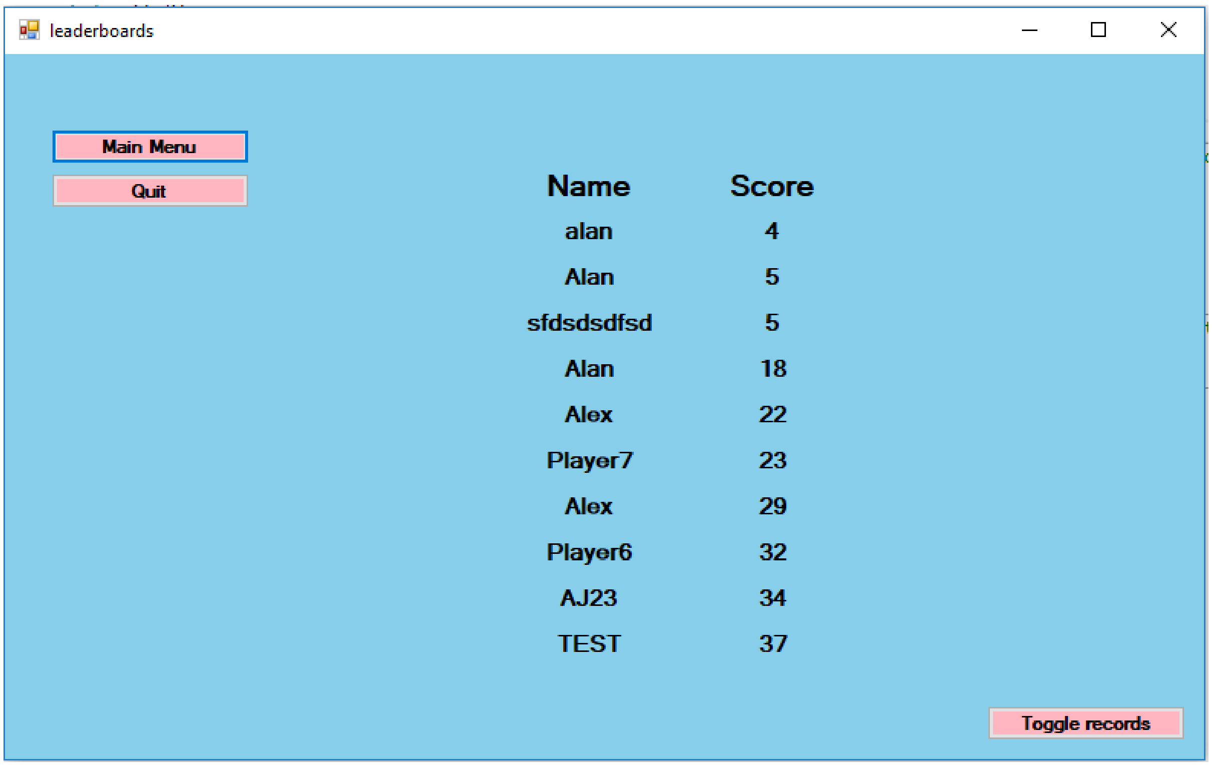
This directory does not exist



geometry hop

Cannot draw the level - save file is corrupt

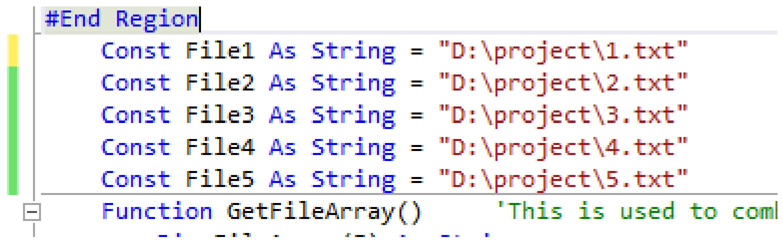
OK



Name	Score
alan	4
Alan	5
sfdsdsdfsd	5
Alan	18
Alex	22
Player7	23
Alex	29
Player6	32
AJ23	34
TEST	37

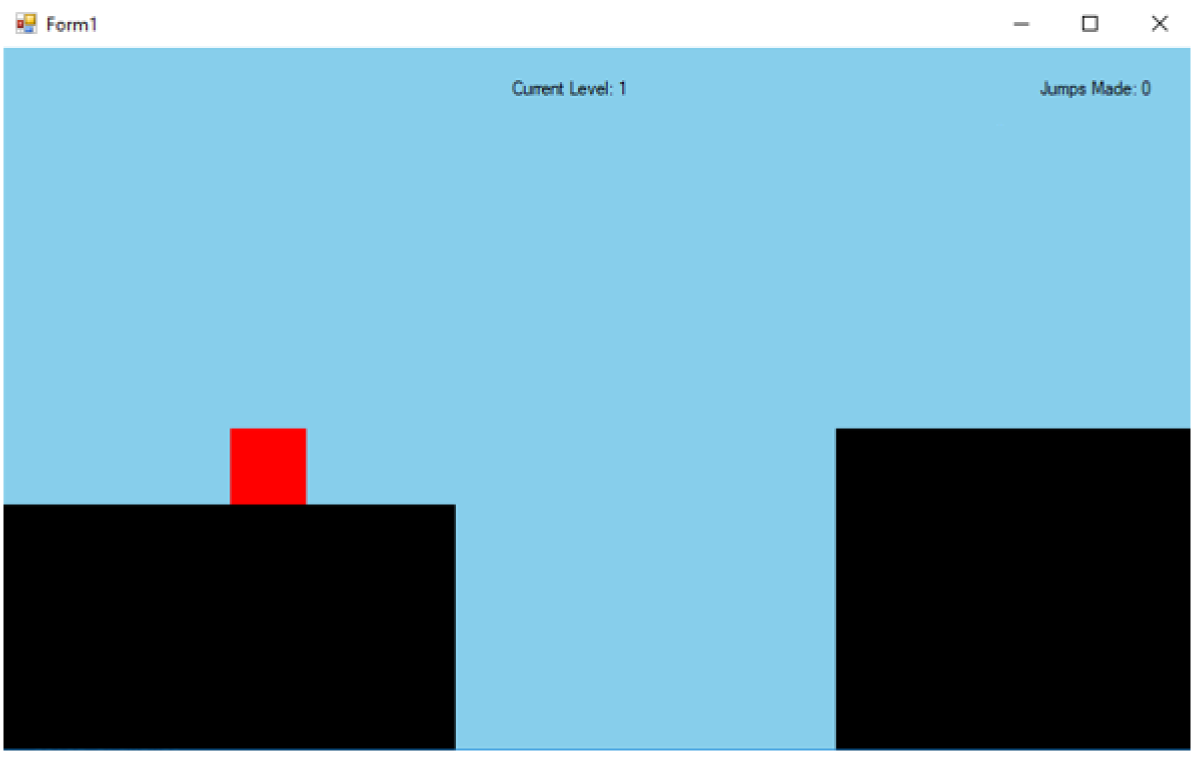
TR.4

In this test I will be testing and investigating what happens when the game reads a correctly written and formatted textfile. I expect the game to be able to read this with the textfiles with no problems, and as a result draw the contents of the textfile to the game screen as well as the level counter and jump counter. The results for this test are shown below; all level directory constants have been set to their correct values (Figure 36).



```
#End Region
    Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to coml
```

Figure 36: Directory constants have been set to their correct values.



When my game correctly reads a textfile, it loads it into the game, and then draws it along with a level counter and a jump counter. This meets my expectations and it shows that my game is working the way I intended it to.

Did I meet the objectives I set out?

In this section I will reflect on the tests I have carried out and see if I have met my objectives which I had set out for myself in the Analysis section of this document (see page *Primary objectives*, on page). I have attached the primary objectives below for the reader's convenience.

---

1. **The game should offer fun and immersive gameplay**
    a. Hitting certain objects will cause the player to respawn
        i. Objects such as the vertical edges of the floor, or possible obstacles should cause the player to respawn
    b. The player will be respawned if they fall through the gaps in the floor of each level
        i. My game's levels will have gaps in between the floors which the player has to jump over for them to reach the end
    c. My game will have multiple levels to play through of which will increase in difficulty
    d. There should be at least 3 levels for the user to play through
        i. Levels will automatically load after the completion of the previous one
        ii. Each subsequent level should have some degree of increased difficulty
        iii. Every level should be stored as text files

2. T**he Game will feature an automatic side scrolling mechanism**
    a. The speed of this mechanism should be steady and constant
    b. The mechanism will only cause objects on the screen to move one way
    c. The player's particle should remain in the same place on the screen when the level is being shifted by this mechanism

3. **The game will feature an element of gravity**
    a. Jumps should look realistic in the sense that the player particle will accelerate into the floor after rising for a short period of time
    b. The player particle should stop falling when it reaches the floor
    c. The player should start free falling if it slides off a platform

4. **The game will also feature a score system**
    a. The score will be displayed throughout the game play in a conveniently placed location so that the player can get live updates of their score
    b. The player's score will be calculated from the number of jumps that the player has made throughout their game – there could also be more factors which contribute to the score system (see secondary objective 1)

5. **The game should give the user some ways of making inputs to control the player particle**
    a. The player particle will only be affected by the user via the spacebar to jump
    b. The player can pause and continue the game using specified keys
        i. "P" should pause the game
        ii. "C" should resume/continue the game

6. **There should be an end screen after completing all levels**
    a. This screen shows the top ten scores made by other players, and possibly the most recent run if it is high enough

        i. They should be displayed in descending order of score
        ii. All scores will be read into the screen from a database
        iii. only the top ten are displayed on the leader board /end screen
    b. There should be a feature which shows the breakdown of how the player has achieved the score of the most recent run (i.e. through the number of jumps the player has made and others - see secondary objective 1)
    c. This screen also gives the user the option to enter a name to go with their score
        i. There should be a textbox prompt on the textbox where the user will enter their name; this prompt should automatically remove itself when the user starts typing their name
        ii. The player's name should be at least 3 characters long

7. **Create a main menu for easier navigation**
    a. Features a button to leave the game/to quit the application
    b. Features a button to start the game
        i. Will close the main menu and then open up the game form/screen where the game will take place
    c. Features a "leaderboards" button
        i. Will close the main menu
        ii. Opens the leaderboard/end screen
            1. When this opens, from the main menu, the user SHOULD NOT be able to add any additional records, they will only be able to view the top scores stored in the database
            2. Scores will be displayed in descending order of score
            3. All scores will be read into the screen from a database

---

Objective number	Have I met this objective?	Test which proves this	Additional notes
1.a.i	YES	G.6.2	Colliding with edges cause the respawning of the player
1.b.i (1.b)	YES	G.6.1	Falling off the map will respawn the player, unless they reach the end of a level
1.c	YES	G.7	I have 5 levels which is more than what was stated in my primary objectives
1.d	YES	G.7	I have 5 levels which is more than what was stated in my primary objectives
1.d.i	YES	G.7	The next level is indeed loaded and drawn into the game automatically without the input of the user
1.d.ii	YES	G.7	The levels arguably get increasingly harder as the user plays on
1.d.iii	YES	G.1.1, G.1.2 + All TR tests	Each level is stored in it's own textfile and the paths of them are defined as constants
2.a	YES	G.2	Scroll mechanism speed is determined by a constant, and all vb.net picturebox controls (apart from the player's) will move left
2.b	YES	G.2	All vb.net picturebox controls (with the player's as the exception) are moved left
2.c	YES	G.3.1, G.2	The player's picturebox appears to be locked at a given position in the x-axis
3.a	YES	G.3.1	Implementation of a gravity mechanism allows for the jumps to look a bit more realistic
3.b	YES	G.8, G.9.1	The player particle doesn't sink through the floor
3.c	YES	G.9.2	The player's picturebox appears to be locked at a given position in the x-axis
4.a	YES	G.4.1	Score of player (the number of jumps they have made) is clearly displayed in the top right of the game screen
4.b	NO	-	I decided to change the way how the user's score would be influenced in the game; instead of using an algorithm which took multiple factors to calculate a final score, I have instead made the user's score the number of jumps they have used to complete the game.
5.a	YES	G.3.1	Pressing the spacebar causes the player to jump, but I have also included a check that prevents double jumping from occurring
5.b.i	YES	G.3.2	Pressing P does pause the game
5.b.ii	YES	G.3.3	Pressing C does continue/resume the game
6.a.i	YES	LB.7	The most recent run will not be displayed in the leaderboards until the leaderboards are refreshed
6.a.ii	YES	LB.7	The scores in the leaderboards show the top ten in descending order
6.a.iii	YES	LB.7	Only the top ten scores are displayed in the leaderboards screen

6.b	NO	-	I decided to change the way how the user's score would be influenced in the game; instead of using an algorithm which took multiple factors to calculate a final score, I have instead made the user's score the number of jumps they have used to complete the game.
6.c.i	YES	LB.3.1 to LB.3.5	The textbox prompt works as intended; it reappears when the textbox is empty and will disappear when the user starts typing
6.c.ii	YES	All LB.v. tests	Inclusion of name validation algorithms, and the name can't be longer than 15 characters too
7.a	YES	MM.3	The quit button definitely works
7.b.i	YES	MM.1	Pressing start begins the game
7.c.i	YES	MM.2	Main Menu form hides when the user opens the leaderboards screen
7.c.ii.1	YES	MM.2	The Leaderboards button opens the leaderboards screen; the user is only able to view the records of the leaderboards when it is opened from the main menu
7.c.ii.2	YES	MM.2	Scores are displayed in descending order in the top ten, as well as in the vb.net data grid view when it is toggled by the user
7.c.ii.3	YES	MM.2	All records of my game are stored in an external database and the records are read into the program when the leaderboards screen is opened

Key: Green – exceeded objectives;
Yellow – Objective met;
Red – Just about met objective

Evaluation

End User feedback

After the development of my prototype I gave it to a number of my friends and family who matched the target demographic of my game. These users then offered their opinions on how the game's prototype could be improved. The feedback I received was overall positive but there was some point which were repeated amongst most of the people I asked to test my prototype. Below I have listed the most common issues raised by the target demographic as well as my explanation for each issue listed

1. ***"The levels are quite challenging"***
   This was a common point of feedback raised by the users who tested my game and I understand their reasoning; I do agree that the levels are quite hard to play through but unfortunately with the limited amount of time I had to develop my programme, there were other priorities other than the difficulty of each level – my main objective with the creation of levels was that there should be at least 3 levels with each subsequent level increasing in difficulty. I met the first half of this objective but the second half was arguably met. If I had more time to develop my game, I would definitely go back and make the levels more playable for people playing for the first time.

2. ***<u>"The game gets increasingly slower after each respawn, but the lag goes away when the next level is loaded"</u>***
   In my first prototype, this was definitely an issue which was made clearer to those who played my first prototype. I was not satisfied by this issue since games should be made to run smoothly, as a result I went back and fixed this issue.
   The difference can be seen between the videos which I have made of the first prototype and the final prototype.

   Videos: Red are of my old prototype, Green are of the second iteration of my prototype.
   - **https://bit.ly/2T7gdVQ** (Annotated version)
   - **https://bit.ly/2HQW0yh** (Unannotated Version)
   - **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
   - **https://bit.ly/2QSG17J** (Unannotated version)

3. ***<u>"The jumping mechanism is quite hard to get used to"</u>***
   I tried my best to implement a realistic jumping mechanism and unfortunately it is quite hard to acclimatise to for some users. In order to make the mechanism slightly more bearable for the users first trying my game, I have tried to make an attempt to make the first few levels slightly easier so that the user can adjust to the jumping mechanism.

4. ***<u>"The program is not easy on the eye"</u>***
   This was common amongst the feedback I received from the users who tested and tried my game. I completely agree with this and I regret that I had to simplify the graphical representation of my program. If I did have more time, I would add more details to the vb.net picturebox controls I used to draw my levels.

5. ***<u>"You can't quit the program properly when you are within the game itself"</u>***
   I accept this criticism and I understand why the people who tested my program are upset about this. In my game, quitting the game during a level will not properly close the program and this annoyed a number of users. If I was going to redevelop, or amend my game I would definitely make this a feature.

6. ***<u>score the scores of each player"</u>***
   I have taken this into account and I may end up changing how the score system works altogether. Currently the game only stores the number of jumps which the player has made and this becomes the final score for the player; this value is also what is stored in the database holding the highscores of the game itself. Although this is already a good way to quantify the performance of the player throughout my game, I could also add another factor – this being the number of resets/respawns which the player makes. The number of respawns could then be put into an algorithm with the jump counter to produce a final score for the game system to store in the database.

7. ***<u>"Collisions are strict; you have to make sure that the player fully lands past the edge of each black floor segement"</u>***
   My game uses a number of invisible barriers which, when come into contact with the player, cause the player to respawn. This was the system which I used to make it so that the player would die if they hit the vertical edge of a floor section; in fact this was one of my primary objectives of my game (see objective 1.a, page : "Hitting certain objects will cause the player to respawn"). The criticism made here is fair and valid however since I also agree with it, the

collision is very strict in the sense where landing even the smallest part off the floor will force a respawn for the player.

Improvements to my design

Score system
Initially I wanted a score system which produced a final score through an algorithm which took into account the time taken for the use to complete the level, the number of jumps the player used to complete the game and the number of respawns the player had. Additionally, the implementation of an algorithm could have produced negative final scores, of which is something I did not want to be possible. As a result of the possibility of negative scores, as well as my time constraints, I decided it would be more sensible for me to add a working score system which consequently was simpler than the one I initially wanted to implement. In the end I decided to store only the number of jumps a player has used as the final score instead. As a result of this, the leaderboard screen had also had some deviations made to it where instead of displaying scores in descending order, the scores are now displayed in ascending order of completion with the lowest number of jumps to the highest number of jumps needed to complete the game.

Lag Issues
During the development of the respawning mechanism described in my section on the Respawning Mechanism (page) in the Design section, I found that it made the game run increasingly slower each time the player respawned. As a result, I redesigned this mechanism entirely so that it removes the need of the distance counter. Additionally, with the old system, the player would be spawned at a pre-determined height, and this wasn't always the same as the initial height the player started at. The new system is a much simpler approach that does not cause performance issues to my program; it simply clears the console of all vb.net controls and then redraws them all. The lag issues can be seen in my old videos which showcase the prototype I developed (the game gets slower every time the player get respawned):

- **https://bit.ly/2T7gdVQ** (Annotated version)
- **https://bit.ly/2HQW0yh** (Unannotated Version)
- **https://bit.ly/2xvusfQ** (Voice-over with annotations version)
- **https://bit.ly/2QSG17J** (Unannotated version)

Use of sounds
The majority of games that are available for a wide audience to play all feature sounds which offer a more immersive gameplay for the user. Currently my game does not feature an option for game sounds and the implementation of sounds can make my game more immersive for users. If I had more time, as well as the resources to create my own sounds, I would definitely implement this feature.

Text files the game will read from

Game File 1 – text file contents

```
0
0
4000
500
picSky
SkyBlue
Sky
====================
2500
500
1500
100
picNextLevel
SkyBlue
End
====================
0
500
2500
100
picVoid
Lime
Void
====================
150
225
50
50
picPlayer
Red
Player
====================
0
300
550
200
picFloor1
Black
Floor
====================
800
250
550
250
picFloor2
Black
Floor
====================
790
260
10
240
picBarrierFloor2
Transparent
Void
====================
1350
350
550
150
picFloor3
Black
Floor
====================
1900
195
600
305
picFloor4
Black
Floor
====================
1890
205
10
145
picBarrierFloor4
Transparent
Void
====================
```

Game File 2 – text file contents

```
0
0
4000
500
picSky
SkyBlue
Sky
====================
2500
500
1500
100
picNextLevel
SkyBlue
End
====================
0
500
2500
100
picVoid
Lime
Void
====================
150
300
50
50
picPlayer
Red
Player
====================
0
400
400
100
picFloor1
Black
Floor
====================
600
350
400
150
picFloor2
Black
Floor
====================
590
360
10
140
picBarrier2
Transparent
Void
====================
1200
300
400
200
picFloor3
Black
Floor
====================
1190
310
10
190
picBarrier3
Transparent
Void
====================
1700
300
250
200
picFloor4
Black
Floor
====================
1690
310
10
190
picBarrier4
Transparent
Void
====================
2200
400
300
100
picFloor5
Black
Floor
====================
2190
410
10
90
picBarrier5
Transparent
Void
====================
```

Game File 3 – text file contents

```
0
0
4000
500
picSky
SkyBlue
Sky
====================
2500
500
1500
100
picNextLevel
SkyBlue
End
====================
0
500
2500
100
picVoid
Lime
Void
====================
150
225
50
50
picPlayer
Red
Player
====================
0
300
450
200
picFloor1
Black
Floor
====================
650
380
300
120
picFloor2
Black
Floor
====================
640
390
10
110
picBarrier2
Transparent
Void
====================
1200
300
300
200
picFloor3
Black
Floor
====================
1190
310
10
190
picBarrier3
Transparent
Void
====================
1800
350
300
150
picFloor4
Black
Floor
====================
1790
360
10
140
picBarrier4
Transparent
Void
====================
2100
200
400
300
picFloor5
Black
Floor
====================
2090
210
10
140
picBarrier5
Transparent
Void
====================
```

Game File 4 – text file contents

```
0
0
4000
500
picSky
SkyBlue
Sky
====================
2500
500
1500
100
picNextLevel
SkyBlue
End
====================
0
500
2500
100
picVoid
Lime
Void
====================
150
325
50
50
picPlayer
Red
Player
====================
0
400
400
100
picFloor1
Black
Floor
====================
500
300
100
200
picFloor2
Black
Floor
====================
490
310
10
190
picBarrier2
Transparent
Void
====================
800
300
100
200
picFloor3
Black
Floor
====================
790
310
10
190
picBarrier3
Transparent
Void
====================
1100
200
100
300
picFloor4
Black
Floor
====================
1090
210
10
290
picBarrier4
Transparent
Void
====================
1450
400
400
100
picFloor5
Black
Floor
====================
1440
410
10
90
picBarrier5
Transparent
Void
====================
2000
300
500
200
picFloor6
Black
Floor
====================
1990
310
10
190
picBarrier6
Transparent
Void
====================
```

Game File 5 – text file contents

```
0
0
4000
500
picSky
SkyBlue
Sky
================
====
2500
500
1500
100
picNextLevel
SkyBlue
End
================
====
0
500
2500
100
picVoid
Lime
Void
================
====
150
300
50
50
picPlayer
Red
Player
================
====
0
400
400
100
picFloor1
Black
Floor
================
====
500
400
25
100
picFloor2
Black
Floor
================
====
490
400
10
100
picBarrier2
Lime
Void
================
====
625
400
25
100
picFloor3
Black
Floor
================
====
615
400
10
100
picBarrier3
Lime
Void
================
====
750
300
25
200
picFloor4
Black
Floor
================
====
740
300
10
200
picBarrier4
Lime
Void
================
====
900
400
170
100
picFloor5
Black
Floor
================
====
890
400
10
100
picBarrier5
Transparent
Void
================
====
1300
250
300
250
picFloor6
Black
Floor
================
====
1290
260
10
240
picBarrier6
Transparent
Void
================
====
1700
400
200
100
picFloor7
Black
Floor
================
====
1690
410
10
90
picBarrier7
Transparent
Void
================
====
1900
350
100
150
picFloor8
Black
Floor
================
====
1890
360
10
40
picBarrier8
Transparent
Void
================
====
2000
300
150
200
picFloor9
Black
Floor
================
====
1990
310
10
40
picBarrier9
Transparent
Void
================
====
2150
250
50
250
picFloor10
Black
Floor
================
====
2140
260
10
40
picBarrier10
Transparent
Void
================
====
2300
400
200
100
picFloor11
Black
Floor
================
====
2290
410
10
90
picBarrier11
Transparent
Void
================
====
```

Code

Main Menu

```
Public Class Main_Menu  'All code here gives my buttons functionality
    Private Sub btnStart_Click(sender As Object, e As EventArgs) Handles btnStart.Click
        Form1.Show()    'Opens the game player form
        Me.Hide()   'Main Menu is HIDDEN so that the program doesnt close itself - vb requires that the main form remains open
    End Sub
    Private Sub btnQuit_Click(sender As Object, e As EventArgs) Handles btnQuit.Click
        Me.Close()  'Quits the application by closing the main form
    End Sub
    Private Sub btnLeaderBoards_Click_1(sender As Object, e As EventArgs) Handles btnLeaderBoards.Click
        OpenLeaderboards()
    End Sub
    Public Sub OpenLeaderboards()
        leaderboards.Show()
        leaderboards.SetUpForReading()  'Opens the leaderboards screen/form
        Me.Hide()   'Main Menu is HIDDEN so that the program doesnt close itself - vb requires that the main form remains open
    End Sub
End Class
```

cLevelSetUp

```
Public Class cLevelSetUp
    Public Function GetLevelInfo(ByVal FileName As String)     'ByVal FileName As String -> Levels are stored as text files which
stores the properties of pictureboxes in the order: Xcoordinate, Ycoordinate, Width, Height, ID/name, Colour, tag
        FileOpen(1, FileName, OpenMode.Input)   'File is opened here
        Dim CurrentLine As String
        Dim Objects As New List(Of PictureBox)
        Try
            Do Until EOF(1)                                   'EOF means end of file // This loop will continue until it has reached
the end of the text file it is reading
                Dim PICTUREBOX As New PictureBox              '   "PICTUREBOX" will be a temporary name for the newly added picturebox
                For PictureBoxProperty = 1 To 8                                'For loop will loop through every line of the text file
and assign the values to a given pictruebox property
                    CurrentLine = LineInput(1)
                    If PictureBoxProperty = 1 Then
                        PICTUREBOX.Left = CurrentLine
                    ElseIf PictureBoxProperty = 2 Then
                        PICTUREBOX.Top = CurrentLine
                    ElseIf PictureBoxProperty = 3 Then
                        PICTUREBOX.Width = CurrentLine
                    ElseIf PictureBoxProperty = 4 Then
                        PICTUREBOX.Height = CurrentLine
                    ElseIf PictureBoxProperty = 5 Then
                        PICTUREBOX.Name = CurrentLine
                    ElseIf PictureBoxProperty = 6 Then
                        PICTUREBOX.BackColor = GetColour(CurrentLine)   'The colour is determine via a private function/getter method
which is exclusive to this class
                    ElseIf PictureBoxProperty = 7 Then
                        PICTUREBOX.Tag = CurrentLine
                    Else
                        CurrentLine = ""                      'This line is to handle the "==========" in the text file for drawing the
levels
                    End If
                    Objects.Add(PICTUREBOX) 'Each temporary picturebox control is added to the list so that it's properties can be
stored to be passed back into the game player form
                Next
            Loop
        Catch ex As Exception   'Exception handling to prevent crashes if a badly written textfile is read
            MsgBox("Error Loading file")
        End Try
        FileClose(1)    'Files that are opened have to be closed
        Return Objects  'The list of picturebox controls with their respective properties is returned (to the game player form where
the class is accessed)
    End Function
    Private Function GetColour(ByVal Input As String) As Color  'Private Function allows for this class to determine the colour of a
picturebox from the value stored in the textfile
        Dim ColourString As Color
        If Input = "SkyBlue" Then
            ColourString = Color.SkyBlue
        ElseIf Input = "Black" Then
            ColourString = Color.Black
        ElseIf Input = "Lime" Then
            ColourString = Color.Lime
        ElseIf Input = "Orange" Then
            ColourString = Color.Orange
        ElseIf Input = "Transparent" Then
            ColourString = Color.SkyBlue
        ElseIf Input = "Red" Then
            ColourString = Color.Red
        End If
        Return ColourString
    End Function
    Public Function GetNoOfPictureboxes(ByVal FileName As String)   'Will tell the game how many picturebox controls it has to draw
for a given level by counting the number of lines in the textfile
        Dim LineCount
        LineCount = (IO.File.ReadAllLines(FileName).Length) - 1
        Return LineCount
    End Function
    Public Sub New(ByVal FileName As String)    'This instantiator is here so that the FileName can be passed into my class; without
this the FileName wouldn't be able to be passed through
    End Sub
End Class
```

Form1 (Game Form)

```
Public Class Form1
#Region "Essential values"
    Const GameSpeed As Integer = 12         'Rate at which pictureboxes are moved along the screen
    Const GravityStrength As Integer = 1    'Gravity field strength - the increase in acceleration of the player particele towards
the floor every tick
    Const JumpingPower As Integer = -22     'The increase in height which the player will recieve every jump
    Const PointsPerJump As Integer = 1      'Points to increment per jump
    Const MinimumComponents As Integer = 30 'This is the minimum number of items in the list returned by the class which are needed
to draw one level

    Private UpwardsSpeed As Integer     'This attribute allows for the jumping mechanism to work
    Private JumpsMade As Integer        'This attribute acts as a counter for the number of jumps the player makes
    Private TouchingFloor As Boolean    'This attribute prevents double jumping
#End Region
#Region "Files and file arrays"     'Storage locations of the game's files
#Region "Test files"
    'Const File1 As String = "F:\project\txtfiles for testing\bee movie script.txt"
    'Const File1 As String = "F:\project\txtfiles for testing\empty.txt"
    'Const File1 As String = "F:\project\txtfiles for testing\level 1 but the x y height and width are words.txt"
    'Const File1 As String = "non-existent.filepath\level1.txt"
#End Region
    Const File1 As String = "D:\project\1.txt"
    Const File2 As String = "D:\project\2.txt"
    Const File3 As String = "D:\project\3.txt"
    Const File4 As String = "D:\project\4.txt"
    Const File5 As String = "D:\project\5.txt"
    Function GetFileArray()     'This is used to combine all the files into an array to allow for the game to cycle through
        Dim FileArray(5) As String
        FileArray(1) = File1
        FileArray(2) = File2
        FileArray(3) = File3
        FileArray(4) = File4
        FileArray(5) = File5
        Return FileArray        'Array is returned at the end for the game to read
    End Function
#End Region
```

```
#Region "Player inputs"
    Private Sub frmPlay_KeyDown(sender As Object, e As KeyEventArgs) Handles MyBase.KeyDown 'Handles game's inputs
        Select Case e.KeyCode
            Case Keys.Space                    'code below is run when the user presses space
                If TouchingFloor = True Then   'will only run if the player is touching the floor; preventing jumping mid-air
                    UpwardsSpeed = JumpingPower 'The player is accelerated upwards by the quantity specified in the constant 
JumpingPower
                    TouchingFloor = False      'When the player jumps, they are no longer touching the floor - purpose of this line
                    JumpsMade += 1  'Increments the jumps made by the player each time they jump
                End If
            Case Keys.P             'Pressing P pauses the game
                StopTimers()        'This is done through a sub which disables all the timers in my program
            Case Keys.C             'Pressing C resumes/continues the game
                ActivateTimers()    'This is done through a sub which re-enables all the timers in my program
        End Select
    End Sub
    Private Sub DrawUpSpeed()           'This subroutine was made for debugging purposes
        Dim lblSpeed As New Label       'Made it easier for me to develop the jumping mechanism
        Me.Controls.Remove(lblSpeed)
        Dim MessageToShow As String
        lblSpeed.Location = New Point(650, 50)
        lblSpeed.Height = 20
        lblSpeed.Width = 100
        lblSpeed.BackColor = Color.SkyBlue
        Me.Controls.Add(lblSpeed)
        MessageToShow = UpwardsSpeed
        lblSpeed.Text = MessageToShow
        lblSpeed.BringToFront()
    End Sub
#End Region
#Region "Game Mechanics - jumping & gravity"
    Private Sub tmrGravity_Tick(sender As Object, e As EventArgs) Handles tmrGravity.Tick
        UpwardsSpeed += GravityStrength               'This is how the speed of the player is changed due to gravity
        For Each cntrl As Control In Me.Controls     'Becuase I have used textfiles to read in the level details the program has to 
check all picturebox objects to look for the player's so the changes only affects this picturebox
            If TypeOf cntrl Is PictureBox Then
                If cntrl.Tag = "Player" Then          'when the player object is found, it's vertical speed is changed by the program 
so that the player accelerates towards the floor
                    cntrl.Top = cntrl.Top + UpwardsSpeed
                End If
            End If
        Next
    End Sub
    Private Sub tmrGameLogic_Tick(sender As Object, e As EventArgs) Handles tmrGameLogic.Tick   'Game Logic check timer
        Collisions()    'Collsions will be checked every tick of this timer
        DrawUpSpeed()   'This is a test sub which helps to debug the jumping mechanism by showing the vertical speed of the player
        DrawScore()
        MoveLevel()     'This is the subroutine responsible for the side scrolling of my game, checked every tick of the game so that
the level moves at the right speed
    End Sub
    Sub MoveLevel()     'This is the subroutine responsible for the side scrolling of my game
        For Each cntrl As Control In Me.Controls        'Will move all picturebox controls but the player's to the left to simulate
the player traversing the level
            If TypeOf cntrl Is PictureBox Then
                If Not cntrl.Tag = "Player" Then
                    cntrl.Left = cntrl.Left - GameSpeed 'GameSpeed constant determines how fast the scrolling will happen
                End If
            End If
        Next
    End Sub
    Private Sub ActivateTimers()        'Re-enables the all timers which are used in my game; used in the pause/continue feature
        tmrGameLogic.Enabled = True
        tmrGravity.Enabled = True
    End Sub
    Private Sub StopTimers()            'Disables the all timers which are used in my game; used in the pause/continue feature
        tmrGameLogic.Enabled = False
        tmrGravity.Enabled = False
    End Sub
    Private Sub Collisions()    'Sub is responsible for checking the collisions the player makes with the level
        Static LevelToDraw As Integer = 1   'This is used when the player reaches the end of a level; starts of at one because the
user first starts playing in level 1
        Static LowestPoint As Integer       'This is the lowest point which the player can hit
        For Each b As Control In Me.Controls    'Checks which cause the player to respawn; if the player hits a picturebox with the
tag "void" the player is repsawned
            If TypeOf b Is PictureBox Then
                If b.Tag = "Void" Then
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    Respawn(LevelToDraw)    'Respawning mechanism will simply redraw the entire level, the current
level is passed in as a parameter so the program knows which level to redraw
                                End If
                            End If
                        End If
                    Next
                ElseIf b.Tag = "End" Then   'Checks for when the player reaches the end of a level; this will prompt the
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    LevelToDraw += 1    'When the user reaches the last level the LevelToDraw is incremented by 1 in
preparation for drawing the next level
                                    DrawNextLevelOrEndScreen(LevelToDraw)   'This sub then decides if the next level is drawn, or if
the leaderboards screen is opened
                                End If
                            End If
                        End If
                    Next
                ElseIf b.Tag = "Floor" Then 'This is here so that the player doesnt sink through the floor due to gravity; this makes
sure that the floor is solid for the player
                    For Each cntrl As Control In Me.Controls
                        If TypeOf cntrl Is PictureBox Then
                            If cntrl.Tag = "Player" Then
                                If cntrl.Bounds.IntersectsWith(b.Bounds) Then
                                    LowestPoint = (b.Top - 50)      'The Lowest Point is defined as the top of a bound - 50 in the Y
direction
                                    cntrl.Top = LowestPoint 'The lowest point is the lowest point the player can sit at, at a given
time
                                    UpwardsSpeed = 0    'When the player is in contact with the floor, their vertical speed becomes 0
to prevent the player from forcing itself through the floor (this was a bug fix)
                                    TouchingFloor = True    'This check then lets the game know that the player is touching the
floor, so that the user is given the opportunity to jump again
                                End If
                            End If
                        End If
                    Next    'use of nested loops and a lot of if statements for the collision detection
                End If
            End If
```

```
        Next 'Arguable that I couldve compared everything to "player" instead of comparing "player" to everything else but this was 
not possible - I have tried and this made the game have delayed collision detection. Ultimately it was better for me to do what I 
have done here
    End Sub
#End Region
#Region "Level Drawings"
    Private Function GetLevelToDraw(ByVal CurrentLevel As Integer)  'This function returns the next level which is to be drawn by the 
program
        Dim ArrayOfFiles(5) As String   'The array is 5 in length as my program features 5 games, this is done for level1 to level5 - 
not using 0th item
        ArrayOfFiles = GetFileArray()   'The files are stored in an array so that the game can move through each item
        Return ArrayOfFiles(CurrentLevel)   'The file for the next level is then returned by this function
    End Function
    Private Sub DrawNextLevelOrEndScreen(ByVal LevelToDraw As Integer)  'This sub decides whether or not the leaderboard screen is 
drawn after the completion of a given level
        Dim NextLevel As String
        Dim NumberOfLevels As Integer = GetFileArray().Length   'Determines how many levels there are so that the game knows when it 
has reached the final level
        ClearLevel()    'This will clear the screen in preparation for the next level, or screen
        If Not LevelToDraw = NumberOfLevels Then    'If the Level to be drawn isnt equal to the number of items in the FileArray then 
the game wil draw the next level
            NextLevel = GetLevelToDraw(LevelToDraw) 'Finds the textfile for the next level
            DrawLevel(NextLevel)    'Then draws this into the game itself
        Else    'In the case the LevelToDraw is equal to the number of items in the FileArray, the leaderboard screen is drawn
            DrawEndScreen() 'Used for moving the game to the leaderboard screen after level is completed
        End If
    End Sub
    Private Sub ClearLevel()    'Removes all objects off of the form, prepares the game for the next level
        Me.Controls.Clear()
    End Sub
    Private Sub DrawEndScreen() 'Used for moving the game to the leaderboard screen after level is completed
        MsgBox("You win")       'The user will be given a message box prompt to let them know they have beaten the game
        LeaderboardDisplay()
    End Sub
    Private Sub DrawLevel(ByVal FileString As String)
        Static CurrentLevel As Integer  'CurrentLevel is defined once as a static to prevent reseting everytime this sub is run
        Dim Level As New cLevelSetUp(FileString)    '"Level" is the name for the instance of the cLevelSetUp Class; this represents 
the level the user will be playing on at a given time. This instance is rewritten everytime a new level is loaded with the details to 
draw the next level.
        Dim FileNotLoaded As Boolean = False    'This boolean checks if the File being loaded was fully loaded
        Dim Objects As New List(Of PictureBox)  'A list of pictureboxes is used to read in and draw each level; a list of
pictureboxes is produced by the cLevelSetUp class whenever a new level's textfiles is read in.
        Objects = Level.GetLevelInfo(FileString)    'The List stores the properties of the pictureboxes that are to be drawn.
        Dim ArrayLength As Integer = Level.GetNoOfPictureboxes(FileString)  'This value is used so that the game knows how many
pictureboxes it is dealing with in the for loop below
        Try
            For i = 0 To ArrayLength    'Will loop through each item in the List of picturebox controls
                Me.Controls.Add(Objects(i))     'THIS LINE ADDS EACH PICTUREBOX IN THE "OBJECTS" LIST TO THE FORM
                If Objects(i).Tag = "Sky" Then          'If statements here will determine where the drawn pictureboxes will sit on
the form
                    Objects(i).SendToBack()             ' "sky" will always be sent to the back because it is esentially the game's
background
                ElseIf Objects(i).Tag = "Floor" Then    ' "Floor" tags will be moved fowards as they are the thing which the player
will be moving on - the player needs to see these
                    Objects(i).BringToFront()
                ElseIf Objects(i).Tag = "Void" Then     ' "Void" tags will be moved fowards as they are the thing which the player
should be avoiding; in order to avoid it, the player must be able to detect it
                    Objects(i).BringToFront()
                ElseIf Objects(i).Tag = "Player" Then   ' "Player" has to be moved fowards to the user can see where they are going
                    Objects(i).BringToFront()
                End If
            Next
        Catch ex As Exception       'Use of exception handling for use when the game reads in a badly written level textfile
            MsgBox("Cannot draw the level - save file is corrupt")
            FileNotLoaded = True
        End Try
        If FileNotLoaded = False Then   'If the textfile contents are fully loaded, the code below will run
            If Objects.Count < MinimumComponents Then   'If there arent enough contents to draw, the user is sent to the leaderboard
screen
                MsgBox("Not enough pictureboxes to draw one level")
                leaderboards.SetUpForReading()
            Else
                CurrentLevel += 1           'Increments the level counter so the game knows what level the user is currently playing
on
                DrawLevelCounterLabel(True) 'ReDraws the level counter so the user can see it
                DrawScore()
            End If
        Else    'If the textfile contents are not loaded, the user is sent to the leaderboard screen
            leaderboards.SetUpForReading()
        End If
```

```
    End Sub
    Private Sub Respawn(ByVal LevelToReDraw As String)  'Sub responsible for the respawning mechanism
        Dim FileString As String = GetLevelToDraw(LevelToReDraw)    'Will firstly get the file location of the level that has to be
redrawn
        ClearLevel()    'Then the form is cleared of its previous controls
        ResetLevel(FileString)  'Then the File location is passed as a parameter into the sub which redraws levels
    End Sub
    Private Sub ResetLevel(ByVal FileString As String)  'works similarly to the DrawLevel sub except it doesn't increment the Level
counter; this is used in the respawning mechanism. No exception handling because if the level has already been drawn it means it is
already readable for the program
        Dim Level As New cLevelSetUp(FileString)    '"Level" is the name for the instance of the cLevelSetUp Class; this represents
the level the user will be playing on at a given time. This instance is rewritten everytime a new level is loaded with the details to
draw the next level.
        Dim Objects As New List(Of PictureBox)  'A list of pictureboxes is used to read in and draw each level; a list of
pictureboxes is produced by the cLevelSetUp class whenever a new level's textfiles is read in.
        Objects = Level.GetLevelInfo(FileString)    'The List stores the properties of the pictureboxes that are to be drawn.
        Dim ArrayLength As Integer = Level.GetNoOfPictureboxes(FileString)  'This value is used so that the game knows how many
pictureboxes it is dealing with in the for loop below
        For i = 0 To ArrayLength    'Will loop through each item in the List of picturebox controls
            Me.Controls.Add(Objects(i))     'THIS LINE ADDS EACH PICTUREBOX IN THE "OBJECTS" LIST TO THE FORM
            If Objects(i).Tag = "Sky" Then          'If statements here will determine where the drawn pictureboxes will sit on the
form
                Objects(i).SendToBack()             ' "AIR" will always be sent to the back because it is esentially the game's
background
            ElseIf Objects(i).Tag = "Coin" Then     ' "COIN" tags will be moved fowards as they are the thing which the player
collects in order to achieve a higher score
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Floor" Then    ' "Floor" tags will be moved fowards as they are the thing which the player will
be moving on - the player needs to see these
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Void" Then     ' "Void" tags will be moved fowards as they are the thing which the player should
be avoiding; in order to avoid it, the player must be able to see it
                Objects(i).BringToFront()
            ElseIf Objects(i).Tag = "Player" Then   ' "Player" has to be moved fowards to the user can see where they are going
                Objects(i).BringToFront()
            End If
        Next
        DrawLevelCounterLabel(False)    'Accessing method for drawing the Level counter which will not increment the value in the
level counter
        DrawScore()
    End Sub
    Private Sub DrawLevelCounterLabel(ByVal increment As Boolean)   'This subroutine draws the Level counter, a boolean is passed in 
so that the subroutine knows if the value in the counter is to be incremented
        Static LevelCount As Integer    'Level is a static so it is not reset everytime this sub is run
        If increment = True Then    'If the boolean passed in is true, the value in the counter is incremented by 1
            LevelCount += 1
        End If
        Dim lblLevelCount As New Label  'Code below (re)draws the counter whenever this subroutine is run
        Dim MessageToShow As String
        lblLevelCount.Location = New Point(333, 20)  'Location of the counter itself
        lblLevelCount.Height = 20   'Dimension of the counter
        lblLevelCount.Width = 100
        lblLevelCount.BackColor = Color.SkyBlue 'Background colour of counter; had to be "SkyBlue"(same colour as the background) 
since "Transparent" doesnt work
        Me.Controls.Add(lblLevelCount)  'Adds the counter to the form
        MessageToShow = "Current Level: " & LevelCount
        lblLevelCount.Text = MessageToShow  'Updates the message/value inside of the counter
        lblLevelCount.BringToFront()    'Brings the counter in front of the background so that the player can see it
    End Sub
    Private Sub DrawScore()   'This subroutine draws the Jump counter, a boolean is passed in so that the subroutine knows if the 
value in the counter is to be incremented
        Dim lblScore As New Label  'Code below (re)draws the counter whenever this subroutine is run
        Me.Controls.Remove(lblScore)
        Dim MessageToShow As String
        lblScore.Location = New Point(680, 20)  'Location of the counter itself
        lblScore.Height = 20    'Dimension of the counter
        lblScore.Width = 100
        lblScore.BackColor = Color.SkyBlue 'Background colour of counter; had to be "SkyBlue"(same colour as the background) since 
"Transparent" doesnt work
        Me.Controls.Add(lblScore)   'Adds the counter to the form
        MessageToShow = "Jumps Made: " & JumpsMade
        lblScore.Text = MessageToShow   'Updates the message/value inside of the counter
        lblScore.BringToFront()     'Brings the counter in front of the background so that the player can see it
    End Sub
#End Region
    Private Sub LeaderboardDisplay()    'Responsible for opening the leaderboards screen
        StopTimers()    'Will firstly disable all timers to lower the load on the processor
        Me.Close()  'Closes the game player form, called "Form1" (this form)
        leaderboards.Show() 'Opens the leaderboards screen/form
        leaderboards.SetUpScreenFromGame(JumpsMade) 'This will then set up the leaderboards screen so that the user will be able to
enter a name to go with their score. The parameter passed in is so that the score can be displayed by the leaderboard screen
    End Sub
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load    'Code beow is run when this form is loaded
        Me.AutoScroll = False   'Makes it so that the user doesn't have the ability to have the vertical & horizontal slidy bars on
the edge of program
        ActivateTimers()    'All the timers in my game are enabled
        DrawLevel(File1)    'The first level is then drawn; "File1" is always going to be first; this is also the same for the way I
have designed the FileArray - "File1" is the first item in the array
    End Sub
End Class
```

Leaderboards

```
Imports System.Data.OleDb   'This is imported to allow for SQL functionality
Imports System.Text.RegularExpressions   'This allows for the use of RegEx in my name validation algorithm
Public Class leaderboards
    Private dgvHidden As Boolean = True
#Region "SQL"
    Const ConnString As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\project\highscores.accdb; Persist Security Info=
False;" 'Location of the database; it is a constant because it will never be changed
    Private Function ExectuteSQLQueryINTEGER(ByVal SQLString As String) 'This Function gets the next ID to store in the database with
the records
        Dim cn As OleDbConnection
        Dim cmd As OleDbCommand
        Dim Reader As OleDbDataReader
        Dim NextID As Integer
        cn = New OleDbConnection(ConnString)
        cn.Open()
        cmd = New OleDbCommand(SQLString, cn)
        Reader = cmd.ExecuteReader
        Dim items = Reader.GetEnumerator()
        While (Reader.Read())
            NextID = Reader(0)
        End While
        Reader.Close()
        cn.Close()
        Return NextID
    End Function
    Private Sub ShowDatabase()
        Dim dtHighScores As New DataTable
        Dim Connection As New OleDbConnection
        Connection.ConnectionString = ConnString
        Connection.Open()
        Dim ds As New DataSet
        Dim dt As New DataTable
        ds.Tables.Add(dt)
        Dim da As New OleDbDataAdapter
        da = New OleDbDataAdapter("select * FROM HighScores ORDER BY score ASC", Connection)    'SELECT query which orders the
records by score in ascending order
```

```
        da.Fill(dt) 'DataTable is filled by the DataAdapter which adapts data from database so that it can be used in vb
        dgvHighScores.DataSource = dt.DefaultView   'DataTable then fills out the data grid view control
        Connection.Close()
    End Sub
    Private Sub InsertData(ByVal Name As String)    'name which user would like to store in thier record is passed into this
subroutine
        Dim SQLstring, SQLQuery As String
        Dim Score, ID As Integer
        Dim Connection As OleDbConnection
        Dim Command As OleDbCommand
        SQLQuery = "SELECT Count(HighScores.ID) AS CountOfID FROM HighScores"   'This line helps the system to create an automatic ID
for the record, IDs are used as primary keys
        ID = CInt(ExctuteSQLQueryINTEGER(SQLQuery)) + 1    'Automatic ID, ID value incremented by 1 each time a new record is added
        Score = lblYourScore.Text   'Score is taken from the system - cannot be changed since the user cant acutally change the
contents of a label
        Connection = New OleDbConnection(ConnString)
        SQLstring = "INSERT INTO HighScores VALUES(" & ID & ",'" & Name & "', " & Score & ") "  'INSERT query applied where the ID,
Name and score is inserted into the database
        Connection.Open()
        Command = New OleDbCommand(SQLstring, Connection)   'SQL applied to database here
        Command.ExecuteNonQuery()
        Connection.Close()
    End Sub
#End Region
#Region "Draw TopTen"
    Const NameInDGV As Integer = 1   'column of the user's name in the vb.net data grid view control
    Const ScoreInDGV As Integer = 2  'column of the user's score in the vb.net data grid view control
    Const xName As Integer = 333    'xCoord of the name column - so that all name labels are alligned with each other
    Const xScore As Integer = 453   'xCoord of the score column - so that all score labels are alligned with each other
    Const yTop As Integer = 74      'the y coordinate for the top of the table
    Const ySpacing As Integer = 30  'spacing to increment by for each row
    Private Sub ShowTopTen()
        ShowDatabase()  'the vb.net data grid view control has to be seen by the program for this to work
        DrawTitles()    'Title lables are drawn
        Dim yCoordToPass, CurrentRow As Integer 'This value is passed as a parameter into the DrawLabels subroutine so that the
program knows where to draw a given score and name
        For i = 0 To 9      'Repitition is used here to produce all of the labels which make up the records part of the leaderboards
screen
            yCoordToPass = yTop + (i * ySpacing + ySpacing)    'The spacing from the top labels is increased every iteration of this
loop to produce a leaderboard
```

```
            CurrentRow = i
            DrawLabels(yCoordToPass, CurrentRow) 'This sub accepts two parameters which is used to aid the formatting of the 
leaderboard screen
        Next
        dgvHighScores.Hide()    'This hides the vb.net data grid view control so that it does not get in the way of the user
    End Sub
    Private Sub DrawTitles()
        Dim lblScoreTitle, lblNameTitle As New Label
        'NAME LABEL
        lblNameTitle.Location = New Point(xName, yTop)
        lblNameTitle.TextAlign = ContentAlignment.MiddleCenter
        lblNameTitle.BackColor = Color.SkyBlue
        lblNameTitle.Font = New Font("Microsoft Sans Serif", 14, FontStyle.Bold)
        lblNameTitle.Text = "Name"
        'SCORE LABEL
        lblScoreTitle.Location = New Point(xScore, yTop)
        lblScoreTitle.TextAlign = ContentAlignment.MiddleCenter
        lblScoreTitle.BackColor = Color.SkyBlue
        lblScoreTitle.Font = New Font("Microsoft Sans Serif", 14, FontStyle.Bold)
        lblScoreTitle.Text = "Score"
        'ADDING LABEL TO FORM
        Me.Controls.Add(lblNameTitle)
        Me.Controls.Add(lblScoreTitle)
    End Sub
    Private Sub DrawLabels(ByVal yCoord As Integer, ByVal CurrentRow As Integer)    'yCoord determines how far down a label is drawn. 
CurrentRow tells the program which row of the vb.net data grid view control to read from
        Dim Name, Score As New Label
        Dim ScoreToShow, NameToShow As String
        NameToShow = dgvHighScores.Item(NameInDGV, CurrentRow).Value    'Gets the name of the user from the vb.net data grid view 
control
        ScoreToShow = dgvHighScores.Item(ScoreInDGV, CurrentRow).Value  'Gets the score of the user from the vb.net data grid view 
control
        'NAMES LABELS
        Name.Location = New Point(xName, yCoord)
        Name.TextAlign = ContentAlignment.MiddleCenter
        Name.BackColor = Color.SkyBlue
        Name.Font = New Font("Microsoft Sans Serif", 11, FontStyle.Bold)
        Name.Text = NameToShow
        'SCORES LABELS
        Score.Location = New Point(xScore, yCoord)
        Score.TextAlign = ContentAlignment.MiddleCenter
        Score.BackColor = Color.SkyBlue
        Score.Font = New Font("Microsoft Sans Serif", 11, FontStyle.Bold)
        Score.Text = ScoreToShow
        'ADD TO FORM
        Me.Controls.Add(Name)
        Me.Controls.Add(Score)
    End Sub
#End Region
    Public Sub SetUpScreenFromGame(ByVal Score As Integer)
        txtYourName.Text = "Type name here"
        txtYourName.Show()
        lblYourScore.Text = Score
        lblYourScore.Show()
        btnSubmitFields.Show()
        lblJumpsMade.Show()
        btnQuit.Hide()
        btnToMain.Hide()
        ShowTopTen()
    End Sub
    Public Sub SetUpForReading()
        ShowTopTen()
        HideFields()
    End Sub
    Private Sub OpenMenu()
        Main_Menu.Show()    'Opens the main menu screen
        Me.Close()  'Closes the leaderboard screen
    End Sub
    Private Function GetName()  'Handles how the program will get the name of the user to store in the database
        Dim NameToStore As String
        NameToStore = txtYourName.Text
        Return NameToStore
    End Function
    Private Function CheckValidName(ByVal NameToCheck As String)    'The name entered by the user is passed into this function as a
parameter
        Const CharCountOfPrompt As Integer = 14
        Const MinNameLength As Integer = 3
        Const MaxNameLength As Integer = 15
        Dim NotAllowedChar As Integer = 0   'This is a counter which counts how many illegal characters the user has in thier name;
it is ultimately used as a check digit for the program itself
```

```
        Dim ValidLength, ValidChars, ValidName As Boolean   'These are the three booleans which help to determine if a name is valid
or not. ValidLength checks that the name given is the right length; ValidChars checks that the name given only contains Valid
characters; ValidName checks that the name given is both a valid length and that it only contains valid characters
        If txtYourName.Text.Contains("Type name here") Then 'This is here to remove the textbox prompt in case the prompt does not
disappear by itself (this was a bug fix)
            NameToCheck = NameToCheck.Remove(NameToCheck.Length - CharCountOfPrompt) 'Removes 14 since the prompt "Type name here"
is exaclty 15 characters from contents of the Textbox where the user is to enter their name
        Else
            NameToCheck = txtYourName.Text  'If the name does not contain the prompt then the NameToCheck is equal to the value in
the textbox
        End If

        If NameToCheck.Length >= MinNameLength And NameToCheck.Length <= MaxNameLength Then 'If the name entered is in
the alllowed range for name length the below code will run
            If Not Regex.Match(NameToCheck, "^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]*$", RegexOptions.IgnoreCase).Success Then  'This
allows only Letters and numbers
                MsgBox("Name contains illegal characters")  'If the name contains a non-allowed character, the below code will run
                ValidChars = False  'Then the Valid Cahracters boolean is set to false
            Else
                ValidChars = True   'Otherwise the valid character boolean is set to true
            End If
        Else
            MsgBox("Enter a name 3 to 15 characters in length") 'If the name is outside the allowed range in length, an error message
is returned
            ValidLength = False
        End If

        If (ValidLength = True) And (ValidChars = True) Then    'Checks to see if the name is valid in terms of length and characters
used
            ValidName = True    'If both length and characters used in the name are valid, then the entire name becomes valid
        Else
            ValidName = False   'If either length or characters used in then name are false, then the entire name is no longer valid
        End If
        Return ValidName    'The overall validity of the name is then returned to the program
    End Function
#Region "Buttons"
    Private Sub btnShowAll_Click(sender As Object, e As EventArgs) Handles btnShowAll.Click
        If dgvHidden = True Then
            dgvHighScores.Show()
            dgvHidden = False
        Else
            dgvHighScores.Hide()
            dgvHidden = True
        End If
    End Sub
    Private Sub btnQuit_Click(sender As Object, e As EventArgs) Handles btnQuit.Click   'Code to allow for the user to leave/quit the
app
        Me.Close()  'Closes the leaderboard screen
        Main_Menu.Close()   'Closes the Main_Menu form to close the entire game; this is the main form
    End Sub
    Private Sub btnToMain_Click(sender As Object, e As EventArgs) Handles btnToMain.Click   'Code to allow the user to naviagte to
the main menu
        OpenMenu()
    End Sub
    Private Sub btnSubmitFields_Click(sender As Object, e As EventArgs) Handles btnSubmitFields.Click       'All code below is run
when the user attempts to submit their name & score (record)
        If CheckValidName(GetName()) = True Then    'A check is made on the name entered using the CheckValidName function, where the
name entered by the user is passed in as a parameter (function parameter)
            InsertData(GetName())   'If the name is valid, then the name is inserted into the database. The name is also passed in as
a parameter here as a function
            HideFields()    'This hides the buttons which allow the user to add to the database; prevents multiple data entries at
once
            MsgBox("Record added!") 'A prompt is made for the user to let them know that thier record was added to the database
            OpenMenu()  'Then the user is forced into the main menu so that the leaderboards can refresh
        Else
            MsgBox("Name not valid, enter a different one") 'In the case where the name entered is not valid, then the user gets a
message box prompt notifying them thar their name is not valid
        End If
    End Sub
    Private Sub HideFields()
        txtYourName.Hide()
        lblYourScore.Hide()
        btnSubmitFields.Hide()
        lblJumpsMade.Hide()
    End Sub
#End Region
#Region "Textbox related"
    Private Sub txtYourName_KeyDown(sender As Object, e As KeyEventArgs) Handles txtYourName.KeyDown
        If txtYourName.Text = "Type name here" Then
            txtYourName.Text = ""
            txtYourName.ForeColor = Color.Black
        End If
    End Sub
    Private Sub txtYourName_MouseEnter(sender As Object, e As EventArgs) Handles txtYourName.MouseEnter
        If txtYourName.Text = "Type name here" Then
            txtYourName.Text = ""
            txtYourName.ForeColor = Color.Black
        End If
    End Sub
    Private Sub txtYourName_MouseLeave(sender As Object, e As EventArgs) Handles txtYourName.MouseLeave
        If txtYourName.Text = "" Then
            txtYourName.Text = "Type name here"
            txtYourName.ForeColor = Color.White
        End If
    End Sub
#End Region
End Class
```