Programming Project By Nathaniel Lowis

Contents

Analysis

Basic Program Specification and Justification of Program

The program I want to create is a platform game where the user will use projectile motion to track the motion of bullets. The user will be able to move to an endpoint, with enemies which you can shoot who guard the way. Once you have reached the endpoint you will have completed the level. You will have a set number of bullets to use and when you fire one, the number you have decreases. You will score points by the time taken, the amount of health left, how many bullets you have left and how many enemies you kill. The different guns will use different number of bullets. The user will try and complete levels and gain the most points which will then be put into a database which the user will be able to see.

This game is hard enough as it requires a complex GUI and a way to hold the scores. It is also complex with the use of physics equations which will be used.

One of the video's (example of my game) shows a playthrough of my game so you can see what is happening and how the program works.

Describe and justify the features that make the problem solvable by computational methods and explain why it is amenable to a computational approach.

This is what my initial ideas and breakdown of the problem:

When a bullet hits either you (from an enemy) or the enemy (from you) the amount of health to be taken off will be calculated using the SUVAT equations ($v = u + at$, $s = (u + v)/2 * t$, $v ^2 = u ^2 + 2as$, $s = ut + ½ a(t^2)$, $s = vt - ½ a(t^2)$). I would set the initial velocity, the acceleration, the time (worked out by how long the bullet has been generated) and I have to work out velocity and displacement. However, this could change. I would then work out the kinetic energy ($E = ½ mV^2$) and then work out the power ($p = E/t$) with the power being the amount of health lost. Unsurprisingly, each character in the level will have their own health pool and the level will not just be a straight line (ie there will be things to jump over which will be decided by one of the SUVAT equations). I will make 2 guns (made by a parent class and then each gun inheriting from the parent class) with one making the bullet move in a parabolic curve and one moving in a horizontal direction. I will use threads to make everything work simultaneously. Finally, I will have a database with all the high scores of the user.

Thinking Abstractly

I will represent the level using rectangles and squares along with circles which are simple things for a computer to draw. A character will be represented by a rectangle along with the end point being represented by a square and a bullet by a circle.

I am going to make two different types of bullets; both types will inherit attributes and methods. The attributes they will inherit are mass, acceleration, time, and methods will be the kinetic energy equation and power equation.

I will use the equations of motion (Also known as SUVAT) to move the bullets.

I will have a collision detector where it will detect whether the character or bullet will pass a certain x axis value if the object is stationary and if not, an x value which will be worked out.

I will have different window for the menu, the game and high scores. All the characters will be rectangles to make it easy to make and also the end point will be a square as well on the screen for the user to see.

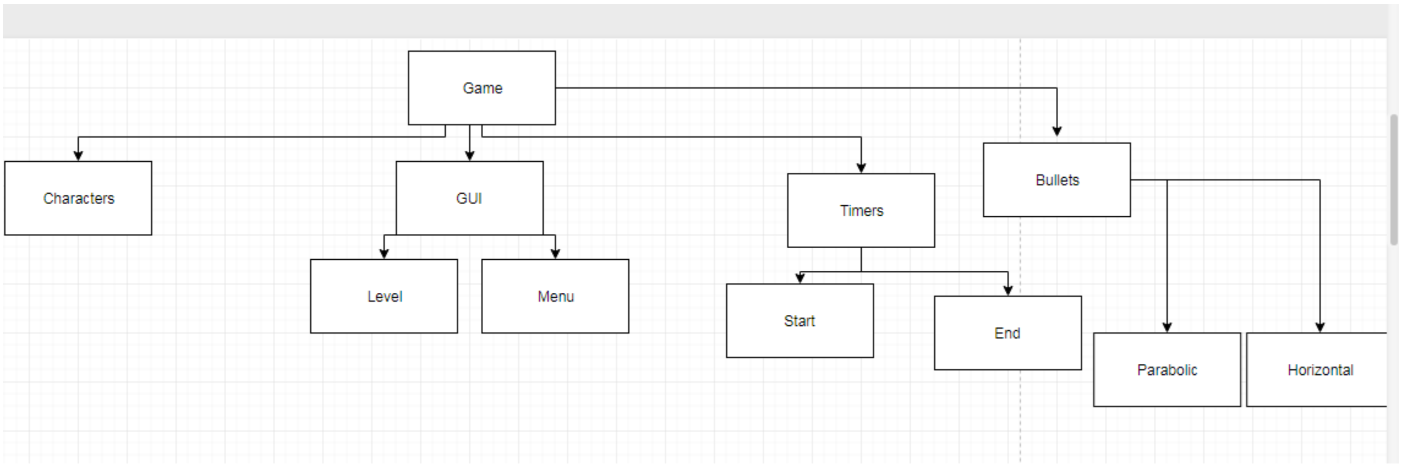
Another element of abstraction will be that the bullet does not accelerate when on the screen but will have constant speed as this is easier to implement.

I have also abstracted that the parabolic bullet will be at the angle of 45 degrees however, in real life, this is unlikely to happen.

Thinking Procedurally

I will have most of the code in procedures, functions and classes. The main classes will be the bullet and character classes and the most important modules will be the menu. They will be the most important modules as if they work the whole game will be able to work as I intend it to do. I will have a larger top down structure diagram later.

I will write the code using classes, functions and procedure with hopefully little code outside any functions, procedures or classes.



Thinking Logically

There will be a decision to see if a bullet has actually hit a character and if it has hit a character, then how much health needs to be taken off. Another decision will be whether a button has been clicked and what to do if it has been. Another decision will be where in the screen the person has pressed so if they have clicked a button it will take them to the next screen. Another decision will be whether the character has reached the endpoint and if it has the game will end.

There will also be a menu which will require decisions about which button was pressed so to decide where the user will go from there.

Another decision will be whether the bullet has 'hit' the enemy and what should happen from there.

Another decision will be which button the user has pressed and what should happen. As well, there will be a decision to decide whether you have clicked the screen allowing you to use the buttons for the game.

There will be another decision on whether the user has entered their username and what the largest Primary key is in the database. Another decision will be which score is the largest and which score is the smallest.

Another decision will be which button has been clicked thus meaning the game knows when to shoot a bullet or the character to move.

Thinking Concurrently

I will try and get threads working in Python however this whole section could be void if I cannot get it working. The different characters will all be able to work concurrently, as will the bullets. These should all be able to work independently of each other. The characters should all be able to move at the same time as the bullets. This then means that the user is facing an actual enemy. They should also be able to fire bullets to try and hit the enemy and this should be able to work without everything else stopping. This will mean you could have 2 or 3 bullets all being fired at once. I will also have a timer on another thread meaning you can get the correct time at all times.

Thinking ahead

I will reuse the code for the bullet and for the characters. I will use the Tkinter library, the time library, SQLite3 library and CSV libraries meaning I do not need to code this from the bottom up. I will also reuse code which I have already programmed.

I will not be deliberately using the cache because I do not know how to and it is not necessary for my program.

Programming Paradigm

I will be using multiple programming paradigms however I will mostly use procedural, object oriented and event driven. The object-oriented paradigm will be used as it will allow me to use classes so I can use new code for bullets and characters but am still able to call them multiple times in the code. It also allows me to inherit features from a super class meaning I can have methods and attributes which are the same for different sub classes but can still differentiate between them.

Event driven programming will be used as I am using a GUI so am waiting for the user to do something to then actually make something happen. It then allows me to wait for when the user wants to move or press a button to start the level.

I will also use logic programming for using the database as I will be using a version of SQL. This will allow me to interrogate the database when needed.

I will finally use procedural programming to execute commands which have been triggered by the user pressing a button and so the code actually works.

Computational methods

I will use visualisation so the user can see what is happening and react to what is happening in the game. It is only a representation so I will only use rectangles for the characters as anything else would be too complex. When I am programming it I will use visualisation, as I will try and solve the equations of motion, on paper beforehand, so I can use them correctly.

I have already recognised the initial problem which is at the top of the document. I will also use problem recognition meaning I can decompose what the problem is at each point of the program.

I will use pipelining with the output of the velocity of the bullet being put into the kinetic energy equation then the final kinetic energy going into the power equation. As well the power from the final equation will then go into how much health needs to be taken off.

I will also use performance modelling, as I will see how much health is taken off when I shoot a bullet and see if I need to take more or less off.

Identify suitable stakeholders for the project and describe them, explaining how they will make use of the proposed solution and why it is appropriate to their needs.

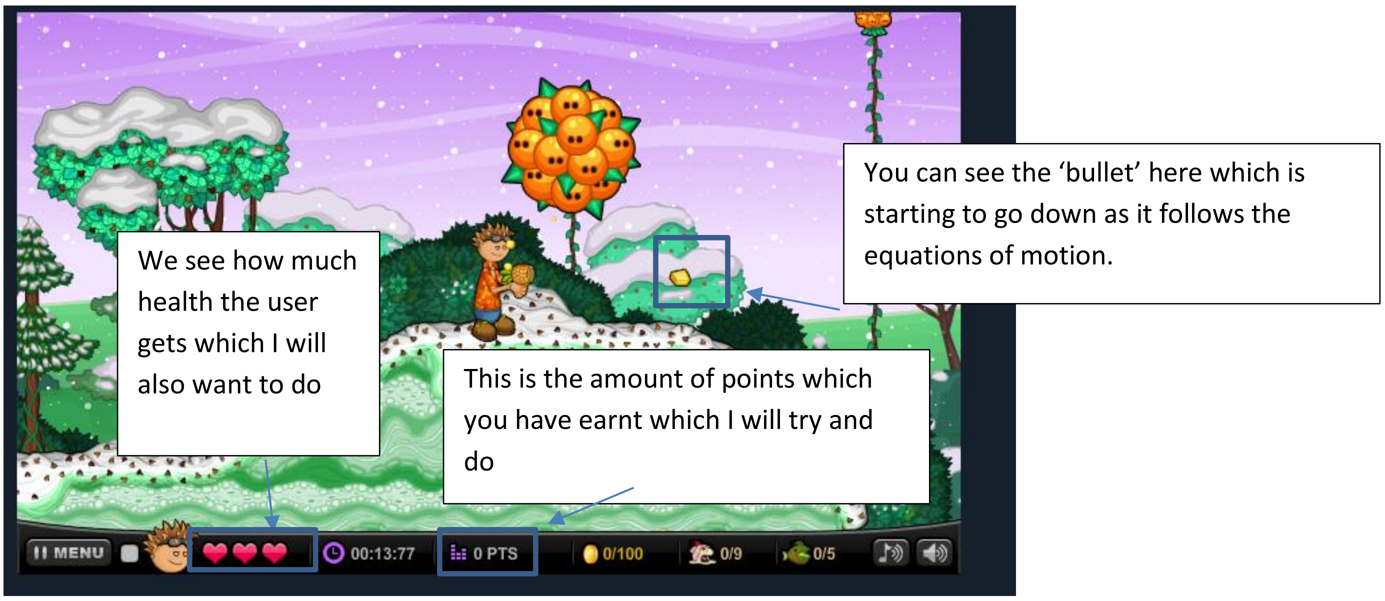
I am going to aim my game at the teenager demographic as I believe they will like the competitiveness of trying to get a high score but will also like the setting of the game.

Person	Description	How they will use it and whether appropriate to needs
Cameron Bird	17-year-old who likes computer games	Cameron enjoys playing computer games. He is an A level physicist meaning he understands how the motion will work so will be able to say whether my physics engine feels realistic
Max Bateman-Sandy	17-year-old who likes computer games	He is a casual gamer who plays a lot of his games on the computer. This makes him very good at saying whether the controls feel naturals and work. Also, he is mildly colour blind so can say whether my colour scheme is any good.
Ben Lowis	14-year-old who plays a lot of computer games	It will be appropriate for Ben as he is someone who likes games like Fortnite so should enjoy the more colourful aspect of the game. He also plays a lot of games meaning he will be able to say whether it is fun and something he would like to go back to. It also means he will enjoy playing a different game with easier objectives.

Research the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.

Papa Louie When Sundaes Attack found on Coolmaths Games

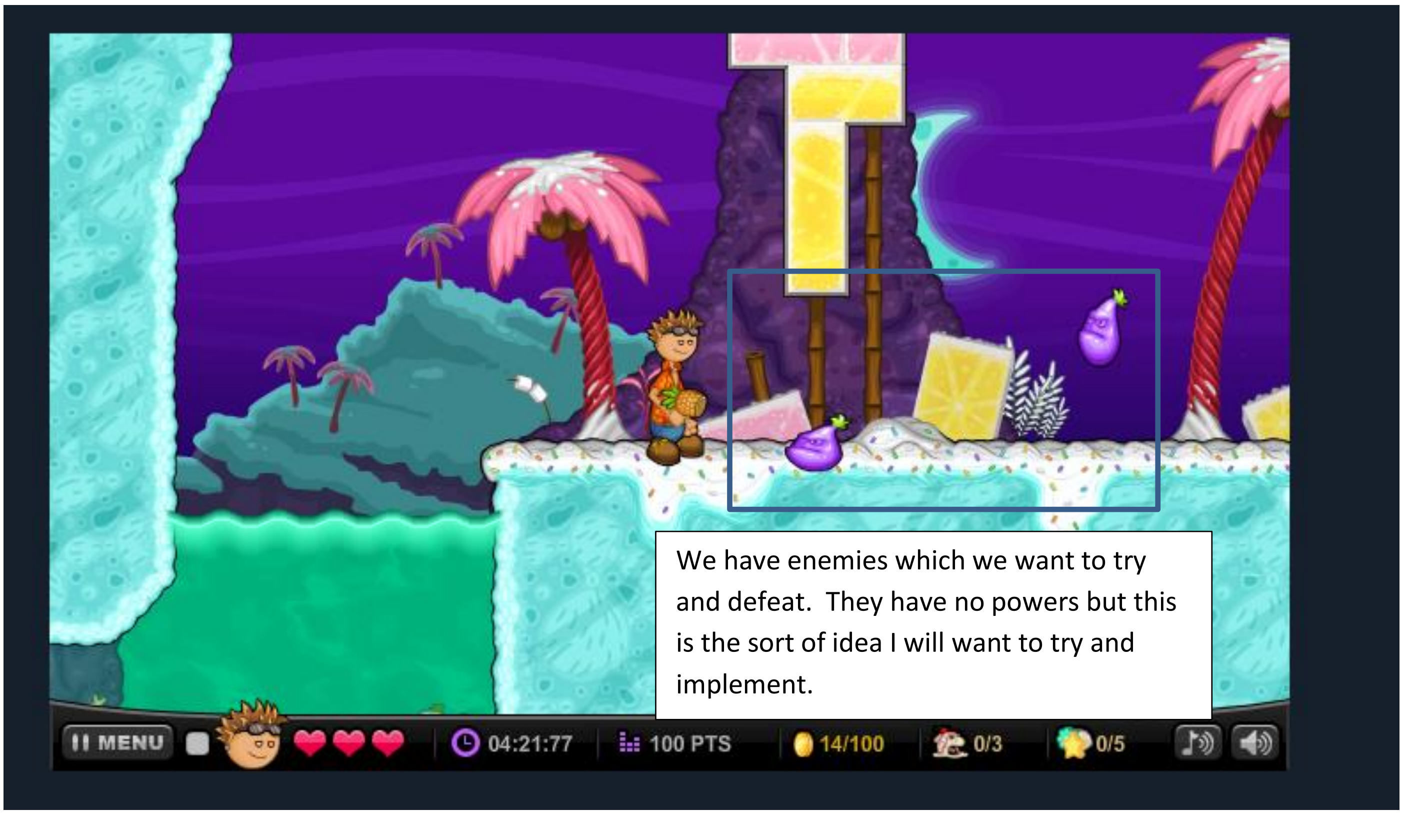
On Coolmaths Games there is a game which I have taken inspiration from, because as you move through the levels you can unlock characters which use projectiles (like in the picture below). They use these projectiles to hit enemies which you will be able to see if you can complete a few levels. One of the biggest differences between this game and the game I want to make is that Papa Louie has many different enemies and only one uses projectile motion whilst I am more likely to make just 1 enemy in the level so my game does not use too many threads. Papa Louie also has the lives and different game elements on the bottom of the screen. I will simplify this to just health pool and bullets remaining. You can also see the projectile moving on the screen which I will do and this will not be very hard to achieve on Tkinter. The GUI is also too complex for what I will be able to produce, so I will simplify it



You can see the 'bullet' here which is starting to go down as it follows the equations of motion.

We see how much health the user gets which I will also want to do

This is the amount of points which you have earnt which I will try and do



We have enemies which we want to try and defeat. They have no powers but this is the sort of idea I will want to try and implement.

I cannot get to the code but I assume it will use some sort of concurrency to try and get the enemies to bounce at the same time as you are moving. I also assume it will have its own physics engine which is a separate module which will have things like the gravity constant in it and also other elements which can help with the physics.

Projectile Motion Maker A simulator for calculating the flight of a projectile in the atmosphere.

This is a program which works out the distance of how far a projectile will go in the atmosphere which I have found in REPL. This code uses vectors to work out the distance which I will use if I use a parabolic motion for one bullet. I am more likely to take away a lot of complex parts, like drag (this is where the air pushes back on you whilst you move) and just use the equations of motion which I can use to show the motion of the bullet. The code is useful as you can see the path which the bullet would take when shot. This is similar to the movement of the bullet that the user sees when the bullet is shot. The module calls another module when it runs. It was interesting to see how the programmer makes sure that it does not reach the stack limit or slow down the running of the program. However, as the programmer has only done it a few times it rarely fails. Another interesting problem is that there can be a lot of dividing by 0 errors. This is an error which he has not caught thus making it harder for the user to play with.

However, the code is easy to read as it only uses 5 -10 lines per module which I would like to do.

Identify the essential features of the proposed computational solution

Description of feature	Solvable by computational methods and amenable to computational approach
A GUI to show what the level looks like at any one moment	A GUI is solvable by using pre-existing libraries like Tkinter. It will be connected to the function and classes of the objects on the screen being used.
Buttons to start game or go to high scores or contents page	This is a GUI feature which I can implement so when the user presses the button, it will call the appropriate function so it can run the correct part.
Algorithms to work out how far a bullet goes.	It is solvable by using maths so the developer will work out how to solve the equation then will implement it by putting it in a class.
Algorithm to work out how many points the user gets	This will be done by an equation the developer will make up then be put into a function
Threads to make different parts work at the same time	This is solvable as I can use the pre-existing library in Python. This allows me to have many different characters moving at the same time.
A sorting algorithm to put the scores in order	This is solvable with the merge sort. I already have code for it and the scores are most likely to already be in order so this will be the fastest.
Algorithms to move the character left/right	This is solvable as you can bind keys to move left or right from a pre-existing library. It is also solvable as I can move the character left or right easily.

Identify the essential features of the proposed computational solution explaining these choices:

The level: This is very important as the way to actual allow the user to play the game. It will be very basic but characters will be represented by rectangles and bullets by circles.

The menu: This will mean the user can decide whether to play the game or to see the high scores of other players.

The bullets: This will be the only way to stop the enemy and needed to win the game

An endpoint: The only way to stop the game.

User's character: The only way the user can interact with the environment

The scoring system: This is the way the user knows how well they have done and also needs to reward finishing the level, getting a good time and finally not using too many bullets.

How to win: You finish the level (win) by getting to an endpoint which you activate when you go past.

How to score: You score by completing the game as quickly as possible and by defeating all the enemies with the least amount of health lost and least number of bullets used. This is computational as I will create an equation to work it out

The GUI: This is important as it allows the user to interact with the game. During the level the user will see rectangles and circles for the characters and the bullets. In the menu there will be buttons to choose where to go.

How to play: The user will play the game by using the left/right keys which allows them to move left or right. The user will also use the different button keys to fire the bullet, change the bullet and also to jump.

Enemy's character: This is the person which you have to defeat. It will be able to use both guns and fire every 3 seconds or so and will have a high damage output. This is essential so that you have an actual objective to try and defeat.

Limitations

Limitation	Explanation	Justification
Timescale of the project	There is a limited length of time to have all documentation and programming to be completed by.	Will not be able to make many different bullets as they will take a while to work out the maths for
The GUI	The GUI will not look good and will not look professional and just uses blocks and circles	Not using a game engine as it has its own prebuilt physics engine built in so will have to use Tkinter instead. So, the graphics will look less impressive
The number of enemies	Will not have many enemies to fight	More enemies will equal more threads to make which then makes the game harder to program. I do not want many threads to work with so will not use many enemies. As well more enemies would require my basic collision detector to be overloaded.
Short levels	The whole level will probably be on the one page	This is because I do not know how to get Tkinter to scroll. If I can I could then make the level larger.
Inaccurate Bullets	The bullets will not be completely accurate. Would behave differently in real life.	This is because I am not learning new equations which bring in drag and other components which I do not know. I am abstracting the bullets down to just using the SUVAT equations making them simpler to use. As well there will be no acceleration on the screen as this would be too hard to make
Having to have a user input for anything to happen.	For the enemy to shoot a bullet there must be a user input so we need to disguise this for the user.	This is because of using Tkinter and also how I will have most of the main commands to do different things on the screen, in the main program, it will be hard for me to program the game to keep on getting the enemy to shoot bullets all the time.

Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements.

Hardware

Hardware	Justification
Intel® Core™ i5-2520M @2.50GHz	This is the core in my laptop which I am using meaning the people playing my game should have a laptop of the same specification.
4GB of RAM	This will allow the whole game to run without too many problems
2 or More Processors	This will allow the game to use more than 1 thread and will allow the program to run everything at the same time.
Basic GPU	This is so the computer will be able to render in the basic graphics. All home computers will have this GPU.
Hard Disk Drive of 1 MB	This will be enough memory to store the whole game and also means it should be able to make quite a large database of scores for different people.
Monitor	This will be needed so the user can see what is happening to their character and react to it.
Mouse	This is needed so the user can press the buttons to access the menu and also so they will be able to interact with the game.
Keyboard with arrow keys	This will be so the user is able to control how the character moves and control the bullets being shot.

Software

IDE research

If I was to use Tkinter and Python I would then need to use a better IDE than IDLE so either Anaconda (think it is specific to Python) or Visual Studios (which I've used before). I will need an IDE so I can save multiple files and access them easily and also so I can have a decent debugger and so I can also see errors before I run the code. I am going to use Thonny (which I have been shown by my teacher) and Visual Studio 2017.

Software	Justification
IDE: Thonny	Even though this is not a widely used IDE, I will use it as it has a very good debugger and also it can help show syntax errors very quickly. (When you open a bracket there will be highlighted bit until you close the bracket)
Python 3.6	Python is a programming language which was created in 1991 by Guido van Rossum which emphasises readability making the syntax very important certainly concerning tabs and whitespace. I will use Python 3.6 as this is the style of the language which I am used to the most and I do not want to confuse myself with errors as I do not know the current version of the language. The user must also have installed Python 3.6 to run the game.
Libraries: Time	This is a library which allows me to work out how long something has been on the screen for or how long it has taken the user to play the game. It is needed for these reasons thus I will be able to score the player on how well they have done
Libraries: CSV	This is a library which will allow me to use CSV files which will hold the score so this can be used in different parts of my program which are in different files
Libraries: SQLite3	This allows me to use SQL to create, interrogate and update a database. This is the library allowing me to use SQL within Python meaning I do not need a different file integrated into my program.
Libraries: Tkinter (known as tkinter in	This allows me to create a GUI for the user to be able to see what is

the code)	happening and visualise what is happening. I am using Tkinter as there is a lot of help which I can find on the internet and also it allows me to create my own physics for the game.
OS: Windows 7 (or equivalent) or better	This is the OS on my most basic laptop and this can run the game. If you went to a lower OS the game might not work as it might not be able to download Python.

Success Criteria

Statement	Justification
The program will work. The player will be able to play a game which is realistic and is a shooting 2D platform game with an enemy which you need to defeat in it. The program will be robust enough to not break if a wrong input is entered.	I do not want there to be any game breaking bugs or glitches so I will want to have identified them before I have finished. I want the game to be finished and that the user feels that it is polished and finished.
The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character and this will be the way to actually interact with the level.
The program will have an enemy	This is so there is some difficulty in the game.
The program will end when the user's character reaches an endpoint.	This means the level will end at one point and will not go on forever.
The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.
The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.
The program will make the bullet go the correct distance using SUVAT.	This make the game realistic
The program will have two bullets a parabolic one and a horizontal one.	This allows the user to have more than one weapon
The program will allow the user to change which bullet to use	This means they can use both bullets.
The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.
The program will have a timer which allows it to work out how long the user has taken.	This is to be used in the total points and allows the user to try and make it quicker
The program will work out a total.	This is to give a point for the game
The program will allow the user to see how well they have done	This will allow the user to see how well they have done
The program will put the score in a database	This is so the user can see how they compare to other people.
The program will have a database of high scores	This is so all scores can be entered and also so the user can see how well they are doing.
The program will allow the user to enter a username.	This is so it can go into the database along with their score.
The program will allow the user to choose whether to see the high scores or to play the game	This will be a menu and allow the user to see different parts of the program.
The program will display a table of high scores.	This allows the user to see how well they are doing.
The program will make sure all characters are on the ground when jumping.	This will then make sure it looks realistic and also means you can see how high they are.
The program will end if the user's character hits 0.	This is so there is a risk if you are hit by a bullet
The program will have a bullet being shot by the enemy every 5 seconds	This is so there is a continuous risk to the user.
The program will allow any characters in the username	This means it's easy for the user to have any name
The program will only allow usernames which no one else has used before.	This is so it is easy for the User to see their score in the database

The program will have an instructions screen	This will allow the user to know how to play the game
The program will have a menu	Allows the user to go to different parts of the program
The program will have a leaderboard screen	This will allow the user to see the current best scores
The program will output the leaderboard from the largest score (Best) to the smallest score (worst)	This will make it easy for the user to see the best and worst scores
The program will take away health if the user's character touches the enemy	This means there is some danger to touching the enemy and means you cannot just run straight to the endpoint
The game will do nothing if you press buttons which are not q, w or the arrow keys.	This means the game cannot break if you enter an accidental input.

Pre-Development

This research which I had done prior to the start of the project.

I will use the libraries Tkinter and time and CSV and SQLite3

I have no need for a game engine as this will already have a pre-built physics engine which I will not want to use as my main complexity for the code comes from getting the bullets to work.

Looking at Unity and Tkinter

I have looked into using Unity and this is what I thought of it:

I have never used Unity before and when getting an account, I decided I would do the lessons thus meaning I know the basics of it. From there I will then try and see what language it is used I then will find a website to try and learn that language. The first tutorial just told me where the play area is (The testing area). There are a lot of different tutorials so I will look through them over the next week. However, I cannot seem to find how to actually code with it and it looks like it already has a Physics engine with it which might not be good as that is what my project is going to be about and I would like to work out how you code in it.

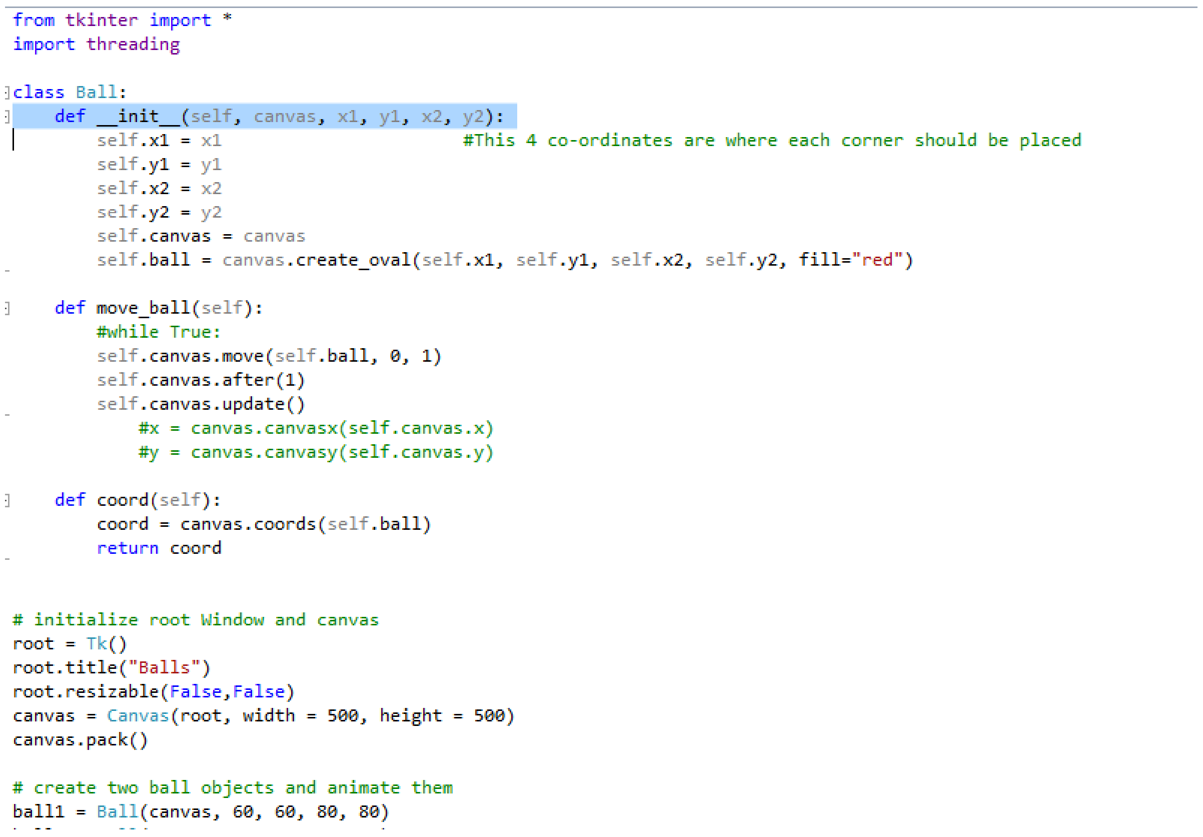
I have programmed in Python before and have been shown Tkinter. This is what I thought of that:

Tkinter means I can actually have to make my own 'physics engine'. I have found out how you can move objects and have found some documentation for a new method canvas. I have also found this which can make a ball and get it moving across the screen. I'll have the code on me. This is actually really interesting as then it means I do not have to worry about making that class. I can make it update I's position which actually is not very hard. Need to look into the canvas method more but this means using Tkinter and Python is much more viable. The problems will be making a way for it to 'see' when a ball collides but this makes it easier. Also means I can use the 'move' method for the other the characters I have. I have also been able to make a floor using the canvas model. The other things to see if I can do is to make a human.

Code I have found.

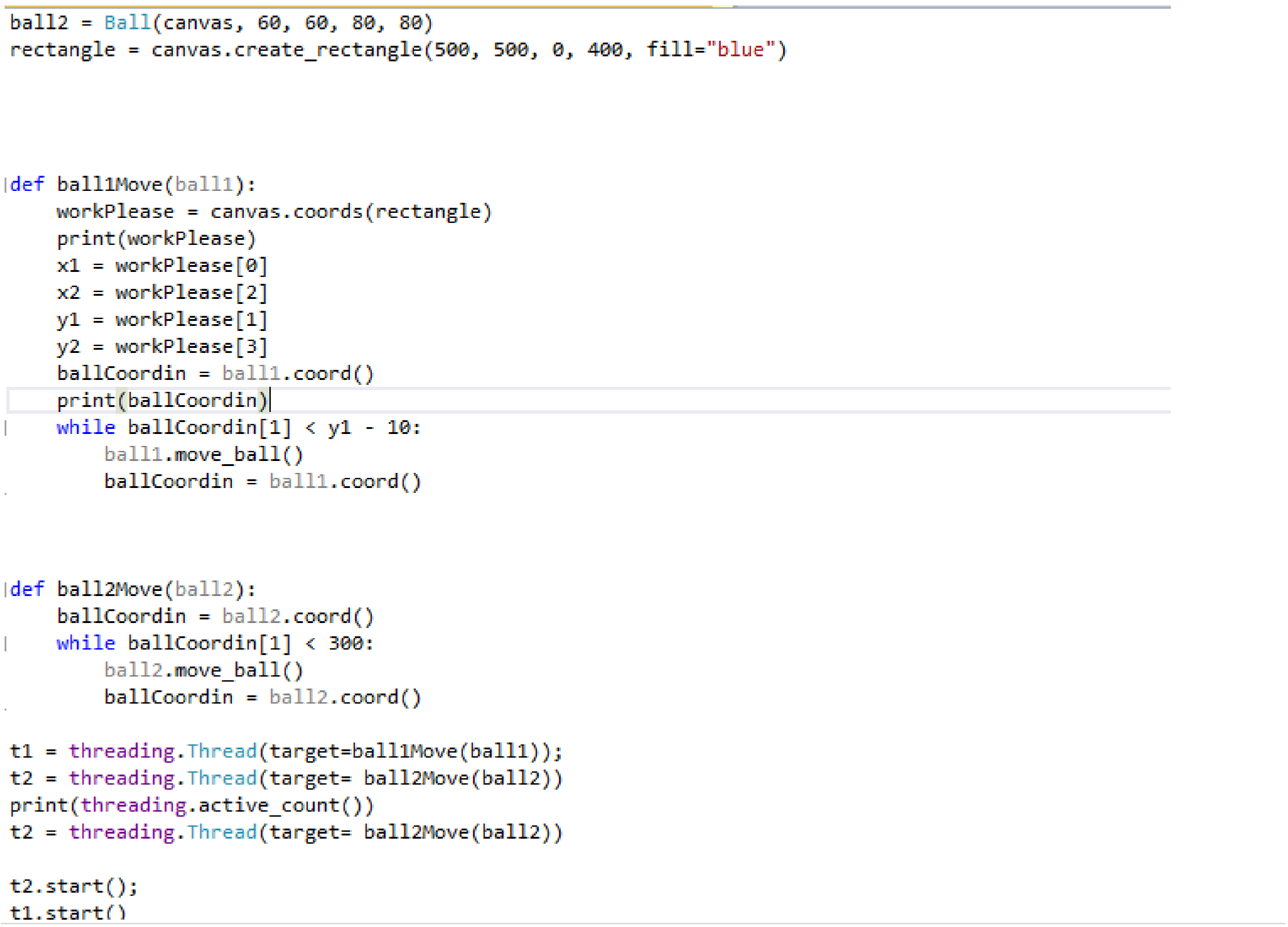
Code for bullet:

All this code is from https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas .



```
from tkinter import *
import threading

class Ball:
    def __init__(self, canvas, x1, y1, x2, y2):
        self.x1 = x1                          #This 4 co-ordinates are where each corner should be placed
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2
        self.canvas = canvas
        self.ball = canvas.create_oval(self.x1, self.y1, self.x2, self.y2, fill="red")

    def move_ball(self):
        #while True:
        self.canvas.move(self.ball, 0, 1)
        self.canvas.after(1)
        self.canvas.update()
            #x = canvas.canvasx(self.canvas.x)
            #y = canvas.canvasy(self.canvas.y)

    def coord(self):
        coord = canvas.coords(self.ball)
        return coord


# initialize root Window and canvas
root = Tk()
root.title("Balls")
root.resizable(False,False)
canvas = Canvas(root, width = 500, height = 500)
canvas.pack()

# create two ball objects and animate them
ball1 = Ball(canvas, 60, 60, 80, 80)
```

This is code which I have amended slightly. I amended by working out the coordinates of the bullet and then stopping when it reaches a certain point. This is a basic collision dectector which I will use. I also looked into using threads with Python to see if I could use them to bring in concurrent programming.

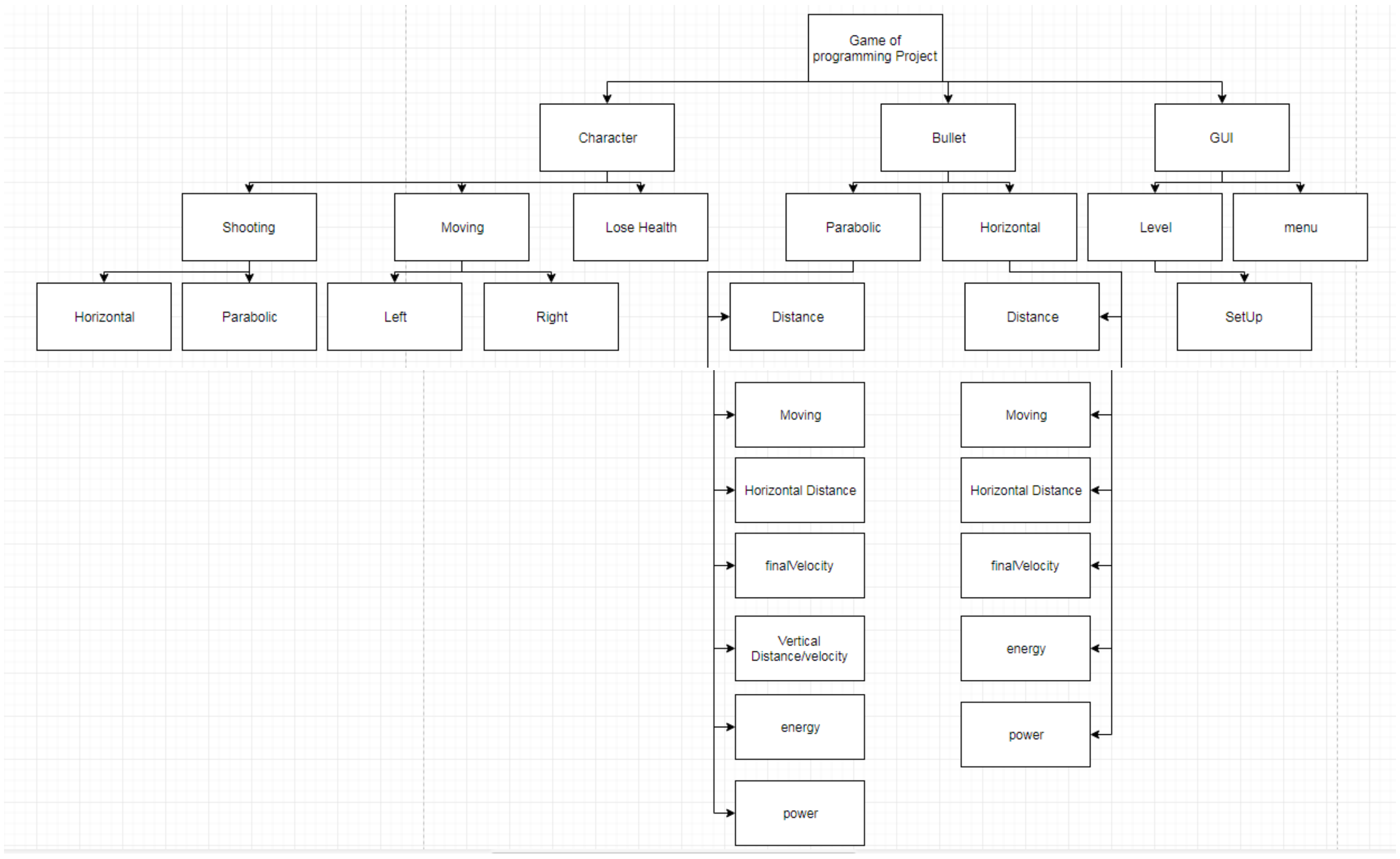


```
ball2 = Ball(canvas, 60, 60, 80, 80)
rectangle = canvas.create_rectangle(500, 500, 0, 400, fill="blue")



def ball1Move(ball1):
    workPlease = canvas.coords(rectangle)
    print(workPlease)
    x1 = workPlease[0]
    x2 = workPlease[2]
    y1 = workPlease[1]
    y2 = workPlease[3]
    ballCoordin = ball1.coord()
    print(ballCoordin)
    while ballCoordin[1] < y1 - 10:
        ball1.move_ball()
        ballCoordin = ball1.coord()



def ball2Move(ball2):
    ballCoordin = ball2.coord()
    while ballCoordin[1] < 300:
        ball2.move_ball()
        ballCoordin = ball2.coord()

t1 = threading.Thread(target=ball1Move(ball1));
t2 = threading.Thread(target= ball2Move(ball2))
print(threading.active_count())
t2 = threading.Thread(target= ball2Move(ball2))

t2.start();
t1.start()
```

Design

Break the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process

Structure Diagram



Class diagram:

CLASS BulletsShot
Attributes: howManyShot (Integer)
Methods: bullets_Sub , bullets_Show

Class diagram:

CLASS Character
Attributes: characterOnScreen (Tkinter Object) , canvasForScreen (Tkinter Object), x0 (Integer), x1(Integer), y0(Integer), y1 (Integer), health (Float)

Methods: delete_Character, lose_ Health, move_Left, move_Right, coord_Player, fire_Hori_Bullet_Class, jump, fire_Hori_Bullet_Class, fire_Hori_Bullet_Class_Enemy, fire_Para_Bullet_Class, health_Left

Class diagram:

CLASS Bullet
Attributes: intialSpeedHori (Float), intialSpeedHoriPara (Float), finalSpeedHori (Float), finalSpeedVertical (Float), finalSpeed (Float)m distanceHori (Float), distanceVertical (Float), horizontalAcceleration (Float), verticalAccelerationUp (Float), verticalAccelerationDown (Float), x1Bullet (Integer), y1Bullet (Integer), x2Bullet (Integer), y2Bullet (Integer), canvasBullet (Tkinter Object), ball (Tkinter Object), time (Integer), mass (Float), energy (Float)
Methods: delete_Bullet, energy_In_Bullet, power_At_Point, coord

Class HoriBullet
Attributes: No New
Methods: speed_At_Any_Point, move_Ball, distance_To_Work_Out

Class: ParabolicBullet
Attributes: No New
Methods: distance_Worker_Up, distance_Worker_Down, speed_Vert_Up, speed_Vert_Down, final_Speed_Parabolic, move_Ball_Para_Up, move_Ball_Para_Down

These classes are made to be the two bullets which you can fire

Sub procedure/ method	Explanation	Justification
__init__ (Character)	This will draw the character onto the level and also set up its variables. It will be the __init__ module of the class character	This allows the character to be seen and also for instances to be made of them.
__init__ (Bullet)	This will set up all the variables needed to set up a bullet	This is needed so the bullet can be made when it is shot and also so it is also outputted to the screen.
fireBullet	This will make sure the bullet is fired for class character	This allows the user to be able to fire a bullet
jump	The class character will be able to jump	Allows the user to maybe be able to jump over bullets or obstacles
move_Left	Let's the class character to move left	This means they can traverse the whole level and go back if

		needs to
lose_Health	Means the class character will be able to lose health	Means the character can actually die if there is any health.
move_Right	Let’s the class character to move right	This means they can traverse the whole level and go to the endpoint
set_Up	This will set up the level and makes sure everything is ready	This can be called when the user says they want to play the game
finish_Game	This will check to see if the game has finished and if it has to work out the highscore	This means the game can actually finish and then can work out the whole highscore and will ask for a username
energy_In_Bullet	This will use the equation e = 1/2mv^2	This will work out the energy the bullet had so I can work out the Power
power_At_Point	This will use the equation = p = e/t	This will work out the amount of health which needs to be taken off by the character
speed_At_Any_Point	This will work out the final Speed	This will mean we can work out the energy
move_Bullet_Hori	This will mean the bullet can move on the screen horizontally	Means the user can see where the bullet is moving
hero_Move_Right	This will move the hero Right and call the finishGame module	This is needed so I can actually allow the user to move the character and finish the game
hero_Move_Left	This will move the hero left and call the finishGame module	This is needed so I can actually allow the user to move the character and finish the game
delete_Bullet	This will delete the bullet from the screen	When the bullet has run its full course, it will be deleted
coord_Bullet	This will work out where the bullet is on the screen.	Must use the coords function from the canvas library but I cannot use the coords function when I have the canvas object in a class so this is how you circumnavigate this problem

distance_To_Work_Out	This will work out how far the bullet has to go with s = ut + 1/at^2	This is so the computer knows how far it needs to send the bullet
delete_Character	This will mean the program can fully delete a character when they have 'died'	This is so the enemy can be deleted when they have 'died'
coord_Player	Works out where the character is on the screen and give coordinates for it	This is so I will be able to move the character and also see if they have passed the end point
fire_Hori_Bullet_Class	Allows the character to fire the horizontal bullet	This is so the bullet can be fired by the right character
fire_Hori_Bullet_Class_Enemy	Allows the enemy to fire the horizontal bullet	This is so the bullet can be fired by the right character
fire_Para_Bullet_Class	Allows the enemy to fire the Parabolic bullet	This is so the bullet can be fired by the right character and so the correct bullet will be fired
fire_Hori_Bullet_Ene	This will allow the bullet to interact with the environment and will actually show the bullet moving when fired by an enemy.	This is needed so the bullet can actually fire by the character's character and means the code works correctly
fire_Hori_Bullet	This will allow the bullet to interact with the environment and will actually show the bullet moving	This is needed so the bullet can actually fire by the enemy's character and means the code works correctly
fire_Para_Bullet	This will allow a parabolic bullet to be fired and allow it to interact with the environment	This is needed so the code is working in the main program so different characters can get to it and also so the code is split up.
callback	This will mean the window will be pressed	This will allow the user to press the keys to actually move
distance_Worker_Up	This will work out how far the bullet should go up and to the side.	This will allow the physics of the bullet to work
distance_Worker_Down	This will work out how far the bullet should go down and to the side.	This will allow the physics of the bullet to work
speed_Vert_Up	Will work out the speed of the	This will be very realistic as it is

	bullet going vertically up	using rules which I have learnt from physics.
speed_Vert_Up	Will work out the speed of the bullet going vertically down	This will be very realistic as it is using rules which I have learnt from physics.
final_speed_Parabolic	This will resolve the two components of sped upwards and to the side and work out the final speed of the bullet.	This is needed as before this we only have the speed for the bullet going upwards and to the side. This will use Pythagoras's theorem to resolve it.
move_ball_Para_Up	This will move the bullet upwards and to the side	This is needed so the bullet will actually move in the correct direction
move_ball_Para_Down	This will move the bullet downwards and to the side	This is needed so the bullet will actually move in the correct direction
enemy_Shooting	This allows the enemy to shoot a bullet	This is needed so each module is not too long and also so you can see what needs to happen at each point.
hero_Touch_Enemy	This will take health off the hero if it is touching the enemy	This is needed so there is a danger to touching the enemy.
hero_lose_Health	This will take health off the hero if it should be taken off it.	This is needed as the code is needed throughout the program.
hero_Jump	This will allow the character to jump	Needed to allow the character to jump and check that it should jump.
checker	This will check to see what the largest unique ID is in the database so far	This will then mean I can make a new unique ID for the next thing to be added to the database
username_Entering	This will allow the user to enter their username and the score to be inputted so it can be inserted into the database	This will mean I have all the data needed to be inputted into the database
adding_To_Database	This will add all the data to the database	This is needed so I can enter all the data into the database and have most of the SQL separate

		to the rest of the code
create_Table	This will allow the person to recreate the leader board	Needed in case I have to ever delete the database so I can create a new one without any problem
display_Database	This will allow the database to be outputted to the screen whilst also being sorted	This is needed to get the whole database and then for it all to be sorted and sent back to the main program to be outputted
quick_Sort	This will sort out the database using the quick sort	Needed as it has a fast Big O natation O(nlogn) and also allows me to sort the database on a certain piece of data
instructions	This will display the instructions for the game on a separate screen	This is needed to be able to display how to play the game to the user
play_Game	This will allow the user to play the game	This is needed so the user is able to press a button and play the game
database_To_Display	This will allow the user to display the leader board on a separate screen	This is needed so the user can press a button and see the high scores
__init__ (BulletsShot)	This will set up the number of bullets the user can use	This is needed to know how many bullets the user can use
bullets_Sub	This will take away 1 from howManyShot	This is needed so the program knows how many bullets the user has left
bullets_Show	This will output how many bullets the user has left	This is needed to be used in the final score
health_Left	This will output how much health the user has	This is needed to be used in the final score
scoring	This will work out the score the user has earnt from the level	Needed to make some sort of completion for completing the level
make_Text	This will make the text to output the table	Needed to be able to output the whole table correctly and in a way Tkinter will like.

Describe the solution fully, using appropriate and accurate algorithms, justifying these algorithms form a complete solution to the problem and identify and justify the test data to be used during the iterative development of the solution

Light Brown = Procedure boxes

main_Game.py

Class Bullets Shot

Pseudocode will be at end

Name of Sub-program	Class BulletsShot __init__
Inputs	None
Outputs	None
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is has made the class	None	Can take away bullets

Name of Sub-program	Class BulletsShot bullets_Sub
Inputs	None
Outputs	None
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to take away bullets	A bullet is shot	The number of bullets you have has gone down by 1

Name of Sub-program	Class BulletsShot bullets_Show
Inputs	None
Outputs	howManyShot (Integer)

Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to output the bullet	More than 10 bullets shot	You cannot shoot anymore bullets

Pseudocode

```
CLASS BulletsShot

        public PROCEDURE new

                howManyShot = 10
        endPROCEDURE

        public PROCEDURE bullets_Sub

                howManyShot = howManyShot – 1

        endPROCEDURE

        public PROCEDURE bullets_Show

                RETURN howManyShot

        endPROCEDURE

endCLASS
```

CLASS CHARACTER

Pseudocode will be at end

Both the user's character and the enemy characters will use this class.

Name of Sub-program	Class Character __init__
Inputs	x0Given (Integer), x1Given(Integer), y1 (Integer), y0 (Integer), canvasCharacter (Tkinter Object), colour (Tkinter Object)
Outputs	Rectangle on screen
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to create a character when initialised	Any x and y coordinates, any colour	Character on screen

Name of Sub-program	Class Character health_Left
Inputs	None
Outputs	health
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to output the code	Hitting the enemy 4 times	Character is deleted and game ends

Name of Sub-program	Class Character jump
Inputs	None
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to get the character to jump up a certain amount of pixels	Up Button	Character goes up then down

Name of Sub-program	Class Character delete_Character
Inputs	characterOnScreen (Tkinter Object)
Outputs	Character deleted
Pre-conditions	There is a character on the screen

Post-conditions	No more of that character

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to delete a character when initialised	The character to be deleted	Character not on screen

Name of Sub-program	Class Character move_Left
Inputs	Left key (object), amount
Outputs	Character Moves left (Object, float)
Pre-conditions	Tkinter has all been imported, a character is on the screen
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to move a character	The left key	The character moves left.

Name of Sub-program	Class Character move_Right
Inputs	Right key (object)
Outputs	Character Moves Right (Object, float)
Pre-conditions	Tkinter has all been imported, a character is on the screen
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to move a character	The Right key	The character moves right.

Name of Sub-program	Class Character lose_Health
Inputs	healthToLose (Float), health (float)
Outputs	New health (Float) (new health lost)
Pre-conditions	That they will lose health
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to make a character lose health	health = 200 healthToLose = 50	health = 150

Name of Sub-program	Class Character fire_Hori_Bullet_Class
Inputs	coordinateInFireBulletHori
Outputs	Bullet being fired and deleted
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the program can make and delete a bullet	Coordinates worked out by where the user is	Bullet is made and moves

Name of Sub-program	Class Character fire_Hori_Bullet_Class_Enemy
Inputs	coordinateInFireBulletHoriEne
Outputs	Bullet being fired and deleted
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output

1	This will test that the program can make and delete a bullet	Coordinates worked out by where the enemy is	Bullet is made and moves

Name of Sub-program	Class Character fire_Para_Bullet_Class
Inputs	coordinateInFireBulletPara
Outputs	Bullet being fired and deleted
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the program can make and delete a bullet	Coordinates worked out by where the character is	Bullet is made and moves in a parabolic arc

Name of Sub-program	Class Character coord_Player
Inputs	characterOnScreen (Tkinter Object)
Outputs	coordinates (Array)
Pre-conditions	There is a character
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the program can work out where both the bullet and enemy character is	None	Bullet Deletes as the bullet coordinates are equal to person coordinates

Reusing this later as if bullet deletes when it hits the character I know the code work

Pseudocode

```
CLASS Character

        Public PROCEDURE new(canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour)
                characterOnScreen = Create Rectangle On Screen
                canvasForCharacter = canvasCharacter
                x0 = x0Given
```

```
            x1 = x1Given
            y0 = y0Given
            y1 = y1Given
            health = 200
      endPROCEDURE

      public PROCEDURE jump
            height = 0
            while height <= 50
                  characterOnScreen move upwards by 1 pixel
                  Update screen
                  Wait 10 ms
                  height = height + 1

            downHeight = 0
            while downHeight <= 50
                  characterOnScreen move downwards by 1 pixel
                  Update canvasForCharacter
                  Wait 10 ms
                  downHeight = downHeight + 1
      endPROCEDURE

      public PROCEDURE delete_Character
            delete CharacterOnScreen
      endPROCEDURE

      public FUNCTION lose_Health (healthToLose)
            health = health – healthToLose
            RETURN health
      endFUNCTION

      public PROCEDURE move_Left
            move characterOnScreen Left
            update canvasForCharacter
      endPROCEDURE

      public PROCEDURE move_Right
            move characterOnScreen Right
            update canvasForCharacter
      endPROCEDURE

      public FUNCTION coord_Player
            coordinates = coordinates of characterOnScreen
            RETURN coordinates
      endFUNCTION

      public PROCEDURE fire_Hori_Bullet_Class (coordinateInFireBulletHori)
            bulletShootingHori = new HoriBullet(coordinateInFireBulletHori[2], coordinateInFireBulletHori[1],
coordinateInFireBulletHori[2] + 5, coordinateInFireBulletHori[1]+5, canvasForCharacter)
            bulletShootingHori.deleteBullet()
            delete bulletShootingHori
      endPROCEDURE

public PROCEDURE fire_Hori_Bullet_Class_Enemy (coordinateInFireBulletHoriEne)
```

```
                bulletShootingHoriEne = new HoriBullet(coordinateInFireBulletHoriEne[2],
coordinateInFireBulletHoriEne[1]+ 45, coordinateInFireBulletHoriEne[2] + 5, coordinateInFireBulletHoriEne[1]+40,
canvasForCharacter )
                bulletShootingHoriEne.deleteBullet()
                delete bulletShootingHori
        endPROCEDURE

public PROCEDURE fire_Para_Bullet_Class (coordinateInFireBulletPara)
                bulletShootingPara = new HoriBullet(coordinateInFireBulletHori[2], coordinateInFireBulletHori[1],
coordinateInFireBulletHori[2] + 5, coordinateInFireBulletHori[1]+5, canvasForCharacter)
                bulletShootingPara.deleteBullet()
                delete bulletShootingPara
        endPROCEDURE


endCLASS
```

CLASS BULLET

The Pseudocode will be at the end

Name of Sub-program	Class Bullet __init__
Inputs	the canvas (Tkinter Object), x1BulletGiven (Integer), y1BulletGiven (Integer), x2BulletGiven (Integer), y2BulletGiven (Integer), canvasToUse
Outputs	Bullet on screen when initialised
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to create a bullet when initialised	Any x and y coordinates	Small Bullet on screen

Name of Sub-program	Class Bullet power_At_Any_Point
Inputs	energy (Float), timeTaken (Float)
Outputs	power (float)
Pre-conditions	Energy has been worked out and timeTaken is known
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to work out the correct Power	energy = 500 timeTaken = 2.5	200

Name of Sub-program	Class Bullet energy_In_Bullet
Inputs	finalSpeed (Float), mass(Integer)
Outputs	energy (Float)
Pre-conditions	Speed has been worked out
Post-conditions	None

Test Number	Explanation	Input	Expected Output

1	This will test that the code is able to work out the correct Energy	speed = 20
		mass = 500
			1000000

Name of Sub-program	Class Bullet delete_Bullet
Inputs	None
Outputs	Bullet deleted (Tkinter Object)
Pre-conditions	There is a bullet
Post-conditions	Bullet is deleted

Test Number	Explanation	Input	Expected Output
1	This will test that the bullet will delete after a set amount of time	Time >= 5	Bullet Deleted
2	This will test that the bullet will delete if hits a character	Hits a character	Bullet Deleted

Name of Sub-program	Class Bullet coord_Bullet
Inputs	None
Outputs	CoordinatesBullet (array)
Pre-conditions	There is a bullet
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the program can work out where both the bullet and enemy character is	None	Bullet Deletes as the bullet coordinates are equal to person coordinates

Name of Sub-program	Class HoriBullet speed_At_Any_Point
Inputs	distanceGone (Float), intialSpeed (Float),

	horizonatalAcceleration (Integer)
Outputs	finalSpeed (Float)
Pre-conditions	We have a distance
Post-conditions	None

Test Number	Explanation	Input	Expected Output

1	This will test the program can work out the speed of a Horizontal bullet	distanceGone = 7
		intialSpeed = 50
		horizonatalAcceleration = 10
			51.3809

Name of Sub-program	Class HoriBullet move_Ball
Inputs	xMovement (Integer)
Outputs	Ball Moving
Pre-conditions	We have a value for xMovement
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can move the bullet	Press q	Bullet moves

Name of Sub-program	Class HoriBullet distance_To_Work_Out
Inputs	intialSpeed (Float), horizontalAcceleration (Integer), time (Float)
Outputs	The distance the bullet must move
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output

1	This will test the program can work out how far the bullet must move	intialSpeed = 5
		horizontalAcceleration = 10
		time = 5
			150

Name of Sub-program	Class ParabolicBullet distance_Worker_Up
Inputs	None (Only variables in class)
Outputs	distanceVertical (Float), distanceHori (Float)
Pre-conditions	Time and verticalAccelerationUp are set
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can work out how far the bullet must move vertically	None	30.65625
2	This will test the program can work out how far the bullet must move horizontally	None	250

Name of Sub-program	Class ParabolicBullet distance_Worker_Down
Inputs	None (Only variables in class)
Outputs	distanceVertical (Float), distanceHori (Float)
Pre-conditions	Time and verticalAccelerationDown are set
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can work out how far the bullet must move vertically	None	30.65625
2	This will test the program can work out how far the bullet must move horizontally	None	250

Name of Sub-program	Class ParabolicBullet speed_Vert_Up
Inputs	distanceVertUp, timeTakenSoFar
Outputs	None
Pre-conditions	The values inputted are there
Post-conditions	The number is larger than or equal to 0

Test Number	Explanation	Input	Expected Output
1	This will test the program can work out the speed of the bullet	None	0

Name of Sub-program	Class ParabolicBullet speed_Vert_Down
Inputs	timeTaken
Outputs	None
Pre-conditions	The values inputted are there
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can work out the speed of the bullet	None	0

Name of Sub-program	Class ParabolicBullet final_Speed_Parabolic
Inputs	HorizontalSpeed (Float) and Vertical Speed
Outputs	finalSpeed
Pre-conditions	The values are in the class
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can work out the final speed of the bullet	VerticalSpeed = 100 horiSpeed = 50	111.80339887498948

Name of Sub-program	Class ParabolicBullet moveBall_Para_Up
Inputs	yMovement (Float), xMovement (Float)
Outputs	None
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can move the bullet	None	Bullet moving Up

Name of Sub-program	Class ParabolicBullet moveBall_Para_Down
Inputs	yMovement2(Float), xMovement2 (Float)
Outputs	None
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test the program can move the bullet	None	Bullet moving Up

Pseudocode

```
CLASS Bullet

    public PROCEDURE new(x1BulletGiven, y1BulletGiven, x2BulletGiven, y2BulletGiven, canvasToUse)

        intialSpeedHori = 50.0
        intialSpeedHoriPara = 50.0

        finalSpeedHori = 0.0
        finalSpeedVertical = 50.0
        finalSpeed = 0.0
        distanceHori =0.0
        distanceVertical = 0.0
        horizontalAcceleration = 30.0
        verticalAccelerationUp = -9.81
        verticalAccelerationDown = 9.81

        x1Bullet = x1BulletGiven
        y1Bullet = y1BulletGiven
        x2Bullet = x2BulletGiven
        y2Bullet = y2BulletGiven

        canvasBullet = canvasToUse

        ball = Create circle on screen with coordinates given
        time = 5
        mass = 0.008
        energy = 0.0

    endPROCEDURE

    public PROCEDURE delete_Bullet
        delete Ball from the screen
    endPROCEDURE

    public PROCEDURE energy_In_Bullet

        energy = .5 * mass * finalSpeed^2
    endPROCEDURE

    public FUNCTION power_At_Point(timeTaken)
        power = energy /timeTaken
        RETURN power
    endFUNCTION

    public FUNCTION coord_Bullet
        coordinatesBullet = Coordinates of ball on screen
        RETURN coordinatesBullet
    endFUNCTION

endCLASS

CLASS HoriBullet inherits Bullet
```

```
    public FUNCTION speed_At_Any_Point(distanceGone)

        finalSpeedHori =( intialSpeedHori ^2 + (2 * horizontalAcceleration * distanceGone)) ^ .5
        RETURN finalSpeed
    endFUNCTION

    public PROCEDURE move_Ball(xMovement)
        moveBullet Right by xMovement
        wait 100ms
        update canvasBullet

    endPROCEDURE

    public FUNCTION distance_To_Work_Out
        distance = (intialSpeed * time) + (.5 *horizontalAcceleration * time^2)
        RETURN distance
    endFUNCTION

endCLASS

CLASS ParabolicBullet inherits Bullet

    public FUNCTION distance_Worker_Up
        distanceVertical = (-.5 * verticalAccelerationUp *(time /2) **2)
        distanceHori = (intialSpeedHoriPara * (time /2))
        RETURN distanceVertical, distanceHori
    endFUNCTION

    public FUNCTION distance_Worker_Down
        distanceVertical = (.5 * verticalAccelerationDonw *(time /2) **2)
        distanceHori = (intialSpeedHoriPara * (time /2))
        RETURN distanceVertical, distanceHori
    endFUNCTION

    public FUNCTION speed_Vert_Up(distanceVertUp, timeTakenSoFar)
        finalSpeedVertical = (distanceVertUp + (.5 * verticalAcceleration * (timeTakenSoFar ** 2))) /
timeTakenSoFar

        IF finalSpeedVertical < 0
            finalSpeedVertical = 0.0
        endIF
        RETURN finalSpeedVertical
    endFUNCTION

    public FUNCTION speed_Vert_Donw(timeTaken)
        finalSpeedVertical = verticalAcceleration * timeTaken
        RETURN finalSpeedVertical
    endFUNCTION

    public FUNCTION final_Speed_Parabolic

        finalSpeed = ((finalSpeedVertical **2) + (intialSpeedHoriPara **2))**.5
        RETURN finalSpeed
    endFUNCTION
```

```
public PROCEDURE move_Ball_Para_Up(yMovement, xMovement)
        moveBullet Right by xMovement and yMovement
        wait 100ms
        update canvasBullet
endPROCEDURE

public PROCEDURE move_Ball_Para_Down(yMovement2, xMovement2)
        moveBullet Right by xMovement and yMovement
        wait 100ms
        update canvasBullet
endPROCEDURE
```

Sub Procedures

Name of Sub-program	Fire_Hori_Bullet
Inputs	bulletFiring (Bullet Class)
Outputs	None (But bullet moves and interacts on screen)
Pre-conditions	Tkinter has all been imported, bulletFiring is a class Bullet
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to make the bullet move the correct distance	Bullet moves 200 pixels	Bullet moves every few milliseconds
2	The bullet will stop and take health off enemy when bullet hits	Press q	Enemy is deleted as so much power is in bullet at the moment
3	The bullet will delete itself after 5 seconds	Press q	Bullet is deleted after 5 seconds

Pseudocode

```
PROCEDURE fire_Hori_Bullet(bulletFiring)
    timer1Start = Take time
    distanceToGo = bulletFiring.distance_To_Work_Out()
    distPer10Milli = distanceToGo / 50
    bulletGone = 0
    timeDoneHoriBullet = 0

    WHILE bulletGone != distanceToGo AND timeDoneHoriBullet < 5000
        enemyCoordinates = enemy.coord_Player()
        bulletCoordinates = bulletFiring.coord_Bullet()

        IF NO ERROR MESSAGE
            IF bulletCoordinates[2] > enemyCoordinates[2]
                timer1Finish = Take time
                timer1 = timer1Finish – timer1Start

                bulletFiring.speed_At_Any_Point(bulletGone)
                bulletFiring.energy_In_Bullet()
                bulletPower = bulletFiring.power_At_Point(timer1)

                bulletGone = distanceToGo
                enemyHealth = enemy.lose_Health(bulletPower)
                IF enemyHealth <= 0
                    enemy.delete_Character()
                endIF
            ELSE
                bulletFiring.move_Ball(distPer10Milli)
                bulletGone = bulletGone + distPer10Milli
                timeDoneHoriBullet = timeDoneHoriBullet + 100
            endIF

        IF ERROR CODE APPEARS
            bulletFiring.move_Ball(distPer10Milli)
            bulletGone = bulletGone + distPer10Milli
            timeDoneHoriBullet = timeDoneHoriBullet + 100
        endERRORCODECATCHING
    endWHILE
endPROCEDURE
```

Name of Sub-program	scoring
Inputs	timeDoneIn (Float), levelScore(Integer)
Outputs	finalScore
Pre-conditions	addingToDatabase is in same file and CSV is imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to work out a school	Getting to the red point	A score over 200

<u>Pseudocode</u>

```
FUNCTION scoring(timeDoneIn, levelScore)

    healthForHero = hero.health_Left()
    howManyBulletsLeft = bulletsleft.bullets_Show()
    finalScore = (1 /timeDoneIn) + (howManyBulletsLeft *5) + healthForHero

    OPEN "Scores.csv" in writing mode as addingScores

        writingARow = writer(addingScores)
        arrayWithScoreIn = [finalScore + "\n"]
        writingARow.writerow(arrayWithScoreIn)
    CLOSE "Scores.csv"

    IMPORT adding_To_Database

    RETURN finalScore
```

endFUNCTION

Name of Sub-program	fire_Hori_Bullet_Ene
Inputs	bulletFiringEne (Bullet Class)
Outputs	None (But bullet moves and interacts on screen)
Pre-conditions	Tkinter has all been imported, bulletFiringEne is a class Bullet
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to make the bullet move the correct distance	Bullet moves 200 pixels	Bullet moves every few milliseconds
2	That the hero is fired at when pressing q	Press q	Bullet is fired first by enemy then hero.
3	The bullet will delete itself after 5 seconds	Press q	Bullet is deleted after 5 seconds

<u>Pseudocode</u>

```
PROCEDURE fire_Hori_BulletEne(bulletFiringEne)
        timer1StartEne = Take time
        distanceToGoEne = bulletFiringEne.distance_To_Work_Out()
        distPer10MilliEne = distanceToGoEne / 50
        bulletGoneEne = 0
        timeDoneHoriBulletEne = 0

        WHILE bulletGoneEne != distanceToGoEne AND timeDoneHoriBulletEne < 5000
                heroCoordinatesEne = hero.coord_Player()
                bulletCoordinates = bulletFiringEne.coord_Bullet()

                IF bulletCoordinatesEne[0] <= heroCoordinatesEne[2] AND  bulletCoordinatesEne[2] >=
        heroCoordinatesEne[0]  AND  bulletCoordinatesEne[1] <= heroCoordinatesEne[3]
                timer1FinishEne = Take time
                        timer1Ene = timer1FinishEne – timer1StartEne
                        bulletFiringEne.speed_At_Any_Point(bulletGoneEne)
                        bulletFiringEne.energy_In_Bullet()
                        bulletPowerEne = bulletFiringEne.power_At_Point(timer1Ene)

                        bulletGoneEne = distanceToGoEne
                        hero_Lose_Health(bulletPowerEne)
        ELSE
            bulletFiringEne.move_Ball(distPer10MilliEne)
            bulletGoneEne = bulletGoneEne + distPer10MilliEne
            timeDoneHoriBulletEne = timeDoneHoriBulletEne + 100
        endIF

    endWHILE
endPROCEDURE
```

Name of Sub-program	fire_Para_Bullet
Inputs	bulletFiring2 (Bullet Class)
Outputs	None (But bullet moves and interacts on screen)
Pre-conditions	Tkinter has all been imported, bulletFiring2 is a class Bullet
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to make the bullet move the correct distance	Bullet moves 200 pixels	Bullet moves every few milliseconds
2	This will test that the bullet moves correctly	Press w	The bullet moves in a parabolic arc
3	The bullet will delete itself after 5 seconds	Press w	Bullet is deleted after 5 seconds

Pseudocode

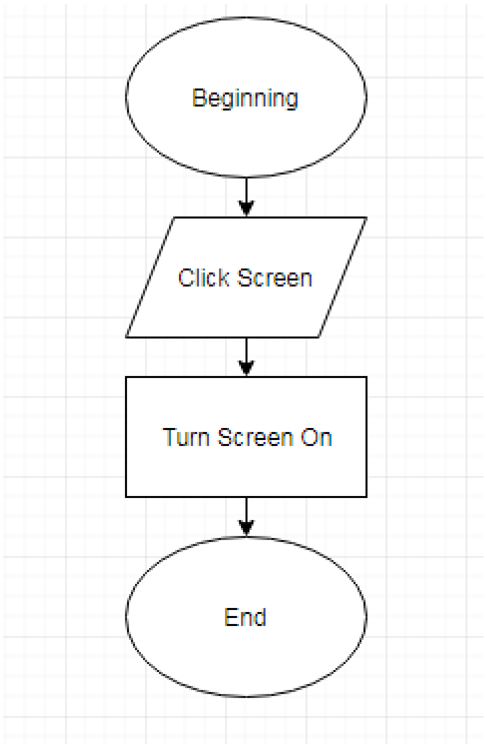
```
PROCEDURE fire_Para_Bullet(bulletFiring2)
    timer2Start = Take time
    distanceToGoUp, distanceLeft = bulletFiring2.distance_WorkerUp()
    distLeft = distanceLeft / 25
    distPer10MilliUp = distanceToGoUp / 25
    bulletGoneUp= 0
    timeDoneParaBullet = 0
    goDown  = True

    WHILE bulletGoneUp != distanceToGoUp AND timeDoneParaBullet < 2500
        enemyCoordinates = enemy.coord_Player()
        bulletCoordinates2 = bulletFiring2.coord_Bullet()

        IF NO ERROR MESSAGE
```

```
                IF bulletCoordinates2[0] > enemyCoordinates[0] AND bulletCoordinates2[1] >=
enemyCoordinates[1] -5 AND bulletCoordinates2[0] < enemyCoordinates[2]
                    timer2Finish = Take time
                    timer2 = timer2Finish – time2Start

                    bulletFiring2.speed_Vert_Up(bulletGoneUp, timer2)
                    bulletFiring2.final_Speed_Parabolic()

                    bulletFiring2.energy_In_Bullet()
                    bulletPowerPara = bulletFiring2.power_At_Point(timer1)

                    bulletGoneUp = distanceToGoUp
                    enemyHealth2 = enemy.lose_Health(bulletPowerPara)

                    IF enemyHealth2 <= 0
                        enemy.delete_Character()
                    endIF
                ELSE
                    bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)
                    bulletGoneUp = bulletGoneUp + distPer10MilliUp
                    timeDoneParaBullet = timeDoneParaBullet + 100
                endIF

            IF ERROR CODE APPEARS
                bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)
                bulletGoneUp = bulletGoneUp + distPer10MilliUp
                timeDoneParaBullet = timeDoneParaBullet + 100
            endERRORCODECATCHING
        endWHILE

        if goDown == True
            timer3Start = Take Time
            distanceToGoDown, distanceLeft2 = bulletFiring2.distance_Worker_Down()
            distLeftToGo = distanceLeft / 25
            distPer10MilliDown = distanceToGoDown / 25
            bulletGoneDown = 0
            timeDoneParaBullet2 = 0

        WHILE bulletGoneDown != distanceToGoDown AND timeDoneParaBullet2 < 2500
            enemyCoordinates2 = enemy.coord_Player()
            bulletCoordinates3  = bulletFiring2.coord_Bullet()
            IF NO ERROR MESSAGE
                IF bulletCoordinates3[0] > enemyCoordinates2[0] AND bulletCoordinates3[1] >=
enemyCoordinates2[1] -5 AND bulletCoordinates3[0] < enemyCoordinates2[2]
                    timer3Finish = Take Time
                    timer3 = timer3Finish – timer3Start

                    bulletFiring2.speed_Vert_Down(timer3)
                    bulletFiring2.final_Speed_Parabolic()

                    bulletFiring2.energy_In_Bullet()
                    bulletPowerPar2a = bulletFiring2.power_At_Point(timer1)

                    bulletGoneDown = distanceToGoDown
                    enemyHealth3 = enemy.lose_Health(bulletPowerPara2)
                    IF enemyHealth2 <= 0
                        enemy.delete_Character()
                    endIF
                ELSE
                    bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)
                    bulletGoneDown = bulletGoneDown + distPer10MilliDown
                    timeDoneParaBullet2 = timeDoneParaBullet2 + 100

                endIF
            IF ERROR CODE APPEARS
                bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)
                bulletGoneDown = bulletGoneDown + distPer10MilliDown
                timeDoneParaBullet2 = timeDoneParaBullet2 + 100
            endERRORCATCHING
        endWHILE


endPROCEDURE
```

Name of Sub-program	callback
Inputs	event (Tkinter Object), pressing screen
Outputs	Can use screen
Pre-conditions	Tkinter has all been imported
Post-conditions	Can now use screen

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to let the user use the screen	Click the screen	Can now use the keyboard with Tkinter

<u>Pseudocode</u>



Name of Sub-program	hero_Move_Left
Inputs	event (Tkinter Object), pressing left key
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to move a character	The left key	The character moves left.

<u>Pseudocode</u>

```
PROCEDURE hero_Move_Left(event)
    hero.move_Left(2)
    hero_Touch_Enemy
    finish_Game
endPROCEDURE
```

Name of Sub-program	hero_Move_Right
Inputs	event (Tkinter Object), pressing Right key
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to move a character	The Right key	The character moves right.

Pseudocode

```
PROCEDURE hero_Move_Right(event)
        hero.move_Right()
        hero_Touch_Enemy
        finish_Game
endPROCEDURE
```

Name of Sub-program	finish_Game
Inputs	None
Outputs	Checks If Game has finished
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code ends if character gets to the endpoint	None	The window deletes itself if it reaches the endpoint.

Pseudocode

```
PROCEDURE finish_Game
        x0Endpoint = 375
        x1Endpoint = 400
        y0Endpoint = 375
        y1Endpoint = 400

        characterPosistion = hero.coord_Player()
        IF characterPosistion[2] > x0Endpoint

                finalTimerEnd = Get Time
                DELETE EVERYTHING
                finalTime = finalTimerStart – finalTimerEnd
                scoring(finalTime, 100)
        endIF
endPROCEUDRE
```

Name of Sub-program	fire_Bullet
Inputs	letter (String)
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code to see if when the user presses q a bullet is fired	q	A Horizontal Bullet is fired

<u>Pseudocode</u>

```
PROCEDURE fire_Bullet(letter)

        bulletLeftHori = bulletsLeft.bullets_Show

        IF bulletLeftHori > 0

                bulletsLeft.bullets_Sub
                ERROR CATCHING IF NO ERROR
                        enemy_Shooting
                IF THERE IS AN ERROR
                        PASS
                endERRORCATCHING

                characterPosistion = hero.Coord_Player()
                hero.fire_Hori_Bullet_Class(characterPosistion)
        endIF
endPROCEDURE
```

Name of Sub-program	fire_Bullet_Para
Inputs	letter (String)
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

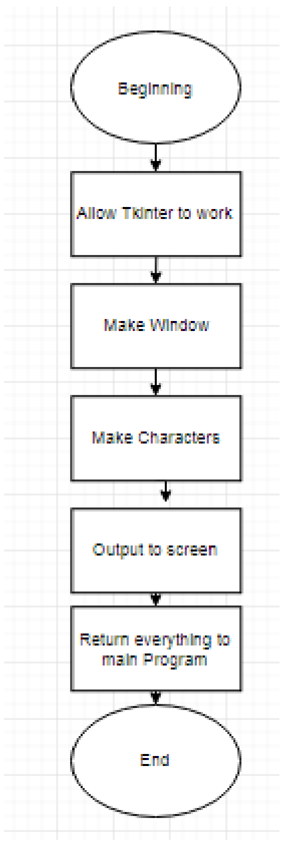
Test Number	Explanation	Input	Expected Output
1	This will test that the code to see if when the user presses w a bullet is fired	w	A Parabolic Bullet is fired

<u>Pseudocode</u>

```
PROCEDURE fire_Bullet(letter)

    bulletLeftHori = bulletsLeft.bullets_Show

    IF bulletLeftHori > 0

        bulletsLeft.bullets_Sub()
        characterPosistion = hero.coord_Player()


        characterPosistion = hero.Coord_Player()
        hero.fire_Para_Bullet_Class(characterPosistionPara)
    endIF

endPROCEDURE
```

Name of Sub-program	set_Up
Inputs	None
Outputs	A GUI which looks like a level, rootSetUp (Tkinter Object), canvasSetUp (Tkinter Object), floorSetUp (Tkinter Object), endPointSetUp (Tkinter Object), heroSetUp (Tkinter Object), enemySetUp (Tkinter Object), bulletsLeft (Class)
Pre-conditions	That there is no screen on. Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code can make a working level	None	A level is made



Name of Sub-program	enemy_Shooting
Inputs	None
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code to see if when the user presses q a bullet is fired from the enemy	q	A Horizontal Bullet is fired from the enemy going towards the hero.

Pseudocode

```
PROCEDURE enemy_Shooting()
    characterPosistionForEnemy = enemy.coord_Player()
    enemy.fire_Hori_Bullet_Class_Enemy(characterPosistionForEnemy)
endPROCEDURE
```

Name of Sub-program	hero_Touch_Enemy
Inputs	None
Outputs	None
Pre-conditions	The hero has touched the enemy
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that if the hero touches the enemy it loses health and dies	Touch the enemy multiple times	The hero dies

<u>Pseudocode</u>

```
PROCEDURE hero_Touch_Enemy()

    heroCurrentCoordinates = hero.coord_Player()
    enemyCurrentCoordinates = enemy.coord()

    IF THERE IS NO ERROR

        IF heroCurrentCoordinates[2] >= enemyCurrentCoordinates[0] and heroCurrentCoordinates[3] >=
enemyCurrentCoordinates[1]
            hero_Lose_Health(50)
            hero.move_Left(20)
        endIF
    ERROR FOUND
        PASS
    EndERRORCATCHING
endPROCEDURE
```

Name of Sub-program	hero_Lose_Health
Inputs	healthToLoseFunction (Integer/Float)
Outputs	None
Pre-conditions	The hero is losing health
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the hero dies if it loses too much health	Touch the enemy multiple times/Get shot at by the enemy	The hero dies

<u>Pseudocode</u>

```
PROCEDURE hero_Lose_Health()

    heroHealth = hero.lose_Health(healthToLoseFunction)

    IF heroHealth <= 0

        finalTimerEnd2 = Get time
        DELETE EVERYTHING

        finalTime2 = finalTimerStart – finalTimerEnd2

        scoring(finalTime2, 0)
    endIF

endPROCEDURE
```

Name of Sub-program	hero_Jump
Inputs	Event (Tkinter Object)
Outputs	None
Pre-conditions	Tkinter has all been imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to move a character	The Up key	The character Jumps

<u>Pseudocode</u>

```
PROCEDURE hero_Jump(event)

    heroCoordinatesJumpFunction = hero.coord_Player()

    IF heroCoordinatesJumpFunction[3] != 400
        PASS
    ELSE
```

```
                    hero_Jump()
          endIF
endPROCEDURE
```

Name of Sub-program	main
Inputs	None
Outputs	Game which works
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code to see if you can play the game	q,w, left key, right key	Something happens on the screen

Pseudocode

```
IMPORT tkinter
IMPORT time
IMPORT csv

finalTimerStart = Get time

root, frame, floor, endPoint, hero, enemy, bulletsLeft= set_Up
Press Left = hero_Move_Left
Press Right = hero_Move_Right
Press Up = hero_Jump
Press Left Key = callback
Press “q” = fire_Bullet
Press “w” = fire_Bullet_Para
Send frame to screen

INFINITE LOOP frame
```

adding_To_Database.py

Name of Sub-program	checker
Inputs	None
Outputs	Largest (Integer)
Pre-conditions	There is a database to connect to and SQLite3 is imported
Post-conditions	The database is shut

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to find the largestID	None	The next record has then next number from the previous record

Pseudocode

```
FUNCTION checker

        connNew = Connect to database
        curser = curser for database

        largest = 0
        curser.execute (SELECT UniqueID FROM Highscores)
        uniqueID = get results

        FOR idRecord in uniqueID
                IF idRecord[0] > largest
                        largest = idRecord[0]
                endIF
        endFOR

        close Database
        RETURN largest
endFUNCTION
```

Name of Sub-program	username_Entering()
Inputs	None
Outputs	None
Pre-conditions	csv is imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to add the correct data to the database	None	(1, Nat, 49….)

Pseudocode

```
PROCEDURE user_Entering

        OPEN Scores.csv as readingScores
                reader  = Set a readerScores

                FOR recordToBeRead in reader

                        IF recordToBeRead == []
                                PASS
                        ELSE
                                finalScoreDatabaseFile = recordToBeRead[0]
                        endIF
                endFOR
        CLOSE Scores.csv
        username = Get from addingUserName
        uniqueIDFunction = checker

        newID = uniqueIDFunction + 1

        labelChange (Change to read: "Your score is being added to the database.  Your score was
{}".format(finalScoreDatabaseFile) )
        adding_To_Database(newID, username, finalScoreDatabaseFile)
endPROCEDURE
```

Name of Sub-program	adding_To_Database
Inputs	uniqueIDToUse (Integer), usernameToAdd (string), scores (Float)
Outputs	None
Pre-conditions	There is a database, all inputs have values
Post-conditions	Database is closed

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to add the correct data to the database	None	(1, Nat, 49….)

Pseudocode

```
PROCEDURE adding_To_Database(uniqueIDToUse, usernameToAdd, scores)

        connectionNew = Connect To highscores.db
        curserNew = Curser for connectionNew

        toAdd = [(uniqueIDToUse, userNameToAdd, scores)]

        curserNew.executemany(INSERT INTO high scores VALUES from toAdd)
        Save connectionNew
        Close connectionNew

endPROCEDURE
```

Name of Sub-program	main
Inputs	None
Outputs	GUI
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce the GUI	None	A GUI

Pseudocode

```
IMPORT tkinter
IMPORT sqlite3
IMPORT csv

databaseScreen = make Window
Name databaseScreen "Bullet Game"
Size databaseScreen (500x500)

addingUserName = Entry to databaseScreen
addingUserName Sent to screen

labelChange = Label saying "Enter Username"
labelChange Sent to screen

buttonAskingForUserName = Button labelled "Enter Username" And when pressed runs username_Entering
buttonAskingForUsername Sent to screen

Infinite Loop databaseScreen
```

database_Creater.py

Name of Sub-program	create_Table
Inputs	name (String)
Outputs	None
Pre-conditions	None
Post-conditions	A database has been made

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a table	highscores.db	A database named highscores.db

Pseudocode

```
PROCEDURE create_Table (name)
        connection = Make a connection
        cursor = Create a cursor  for the database
```

Name of Sub-program	main
Inputs	None
Outputs	None
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a table	None	A database named highscores.db

Pseudocode

```
IMPORT sqlite3
create_Table("highscores.db")
```

displaying_Database.py

Name of Sub-program	display_Database
Inputs	None
Outputs	sortedTable (List of tuples)
Pre-conditions	There is a database
Post-conditions	Table is sorted, database is shut

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a sorted table	Table	A sorted table

Pseudocode

```
FUNCTION display_Database
    connectionDisplay = Connect to "highscores.db"
    cursorDisplay = Make a cursor

    cursorDisplay.execute(SELECT * FROM Highscores)
    table = Fetch result

    sortedTable = quick_Sort(table)

    CLOSE connectionDisplay
    RETURN sortedTable
endFunction
```

Name of Sub-program	quick_Sort
Inputs	dataArray (list Of Tuples)
Outputs	first_part + second_Part
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a sorted table	Table	A sorted table

Pseudocode

```
FUNCTION quick_Sort(dataArray)

    IF len(dataArray) == 1 or len(dataArray) == 0
        RETURN dataArray
    ELSE
        pivot = dataArray[0][2]
        i = 0

        FOR j in range(len(dataArray)-1)
            IF dataArray[j + 1][2] > pivot
                tempDataArray = dataArray[j + 1]
                dataArray[j + 1] = dataArray[i + 1]
                dataArray[i + 1] = tempDataArray

                i = i + 1
            endIF
        endFOR

        tempDataArray = dataArray[0]
        dataArray[0] = dataArray[i]
        dataArray[i] = tempDataArray
        first_part = quick_Sort(dataArray[:i])
        second_part = quick_Sort(dataArray[i + 1:])
        first_Part.append(dataArray[i])

        RETURN first_part + second_part
    endIF
endFUNCTION
```

Name of Sub-program	make_Text
Inputs	None
Outputs	tableDisplay (String)
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce text	None	A sorted table in the GUI

Pseudocode

```
FUNCTION make_Text
    tableToDisplay = display_Database
```

```
        tableDisplay = “HighScores \n Place, UniqueID, Username, Score \n”
        place = 0

        FOR recordToDisplay IN tableToDisplay
                place = place + 1
                placeString = str(place)

                recordToDisplayStr = str(recordToDisplay)

                tableDisplay = tableDisplay, placeString,  recordToDisplayStr + "\n"
        endFOR
endFUNCTION
```

Name of Sub-program	main
Inputs	None
Outputs	A GUI
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a GUI	None	A GUI with a high scores table

<u>Pseudocode</u>

```
IMPORT sqlite3
IMPORT tkinter

databaseShowScreen = Make Window
Name databaseShowScreen  “Bullet Game”
Size DatabaseShowScreen 500x500

databaseShowScreen Colour Navy

displayTable = make_Text

labelWithTable = Make Label with displayTable being outputted
Send labelWithTable to screen

Infinite Loop databaseShowScreen
```

final_Game_File

Name of Sub-program	instructions
Inputs	None
Outputs	None
Pre-conditions	Tkinter is imported
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a GUI with instructions in it	None	A GUI with the instructions on it

Pseudocode

```
PROCEDURE instructions

        instructionsScreen = Make a window
        Title instructionsScreen = “Bullet Game”
        Size instructionsScreen = 500x500

        Background Colour instructionsScreen = Navy

        instructionsToDo = Label saying "To play the game click on the screen.  Controls (Do not have caps lock on) \n q: Fires a horizontal Bullet.  Powerful however the enemy will shoot \n w: Fires a Parabolic Bullet.  Not so powerful \n Aim: Get to the red block as fast as you can using the least number of bullets possible and having the most health at the end. \nCyan Block - Enemy\nDark Blue Block - You"
        instructionsToDo Sent to screen

endPROCEDURE
```

Name of Sub-program	play_Game
Inputs	None
Outputs	None
Pre-conditions	Program is in the same file
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to play the game	None	A GUI with the game on it which you can play

Pseudocode

```
PROCEDURE play_Game

        IMPORT main_Game
endPROCEDURE
```

Name of Sub-program	database_To_Display
Inputs	None
Outputs	None
Pre-conditions	Program is in same file
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a GUI the database in it	None	A GUI with the leader board

Pseudocode

```
PROCEDURE database_To_Display

        IMPORT displaying_Database

endPROCEDURE
```

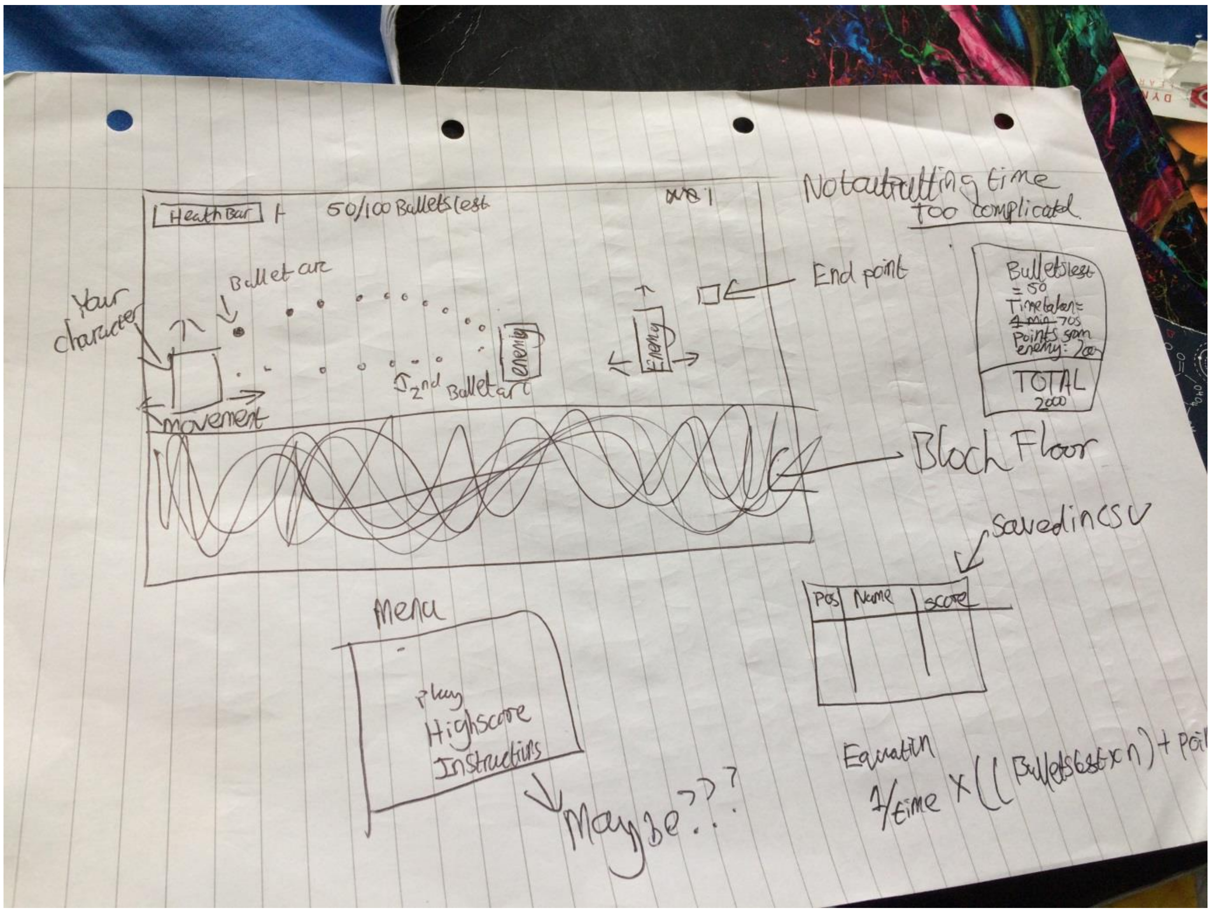
Name of Sub-program	main
Inputs	None
Outputs	A GUI
Pre-conditions	None
Post-conditions	None

Test Number	Explanation	Input	Expected Output
1	This will test that the code is able to produce a GUI with a menu	None	A GUI with the menu
2	This will test the user can press each button and the GUI do something	Pressing a button	A New screen appears

<u>Pseudocode</u>

```
IMPORT tkinter

window = Make a screen

Title window “Bullet Game”
Size window 500x500

Background Colour window “Navy”

labelWelcome = Make a label saying Bullet Shot
Sent labelWelcome to the screen

buttonInstructions = Make a button labelled Instructions With a procedure of instructions
Sent buttonInstructions to the screen

buttonGame = Make a button labelled Game and with a procedure play_Game
Sent buttonGame to the screen

buttonLeaderboard = Make a button labelled Leaderboard with a procedure database_To_Display
Sent buttonLeaderboard to the screen

Infinite Loop window
```

Storyboard



Heath Bar
50/100 Bullets left
Bullet arc
Your character
movement
2nd Bullet arc
enemy
Enemy
End point
Bloch Floor
~~Not~~ cultrulting time
Too complicated
Bullets left = 50
Time taken= ~~4 min~~ 70s
Points from enemy: 200
TOTAL 2000
Saved in CSV
Pos
Name
score
Menu
Play
Highscore
Instructions
Maybe???
Equation
1/time x ((Bullets left x n) + poi

Describe, justifying choices made, the usability features to be included in the solution

Usability Feature	Explanation	Justification
Different colours for the different objects	All the different objects made will be different colours.	This is done to make the user able to identify the different objects on the screen.
High contrast colours	All the different colours will be very different	This makes it easier for mildly colour-blind users see the level
Instructions	There will be a choice for the instructions to make it easier for the user to know what is happening.	Means the user can know what is happening and helps the user understand how to play the game.
The buttons are close to each other.	This means that I am using buttons which are close to each other so q and w for shooting	This makes it easier for people who have limited mobility meaning they can still play the game

Identify and justify the key variables/ data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation

Variable Name	Initial Value	Global or Local (Where)	Variable/ Constant	Purpose	Justification	Validation
rootSetUp	Tkinter	Local (setUp)	Constant (Variable in Python)	This means I can use Tkinter	This allows me to use Tkinter as my GUI	Check it has been imported to Python
canvasSetUp	Window	Local (setup)	Variable	This means I have a window for the GUI	Allows me to use a window.	Checks that the window is created
floorSetUp	Object	Local (setUp)	Variable	This means I have a floor that the user can see	This means the user can see where they need to stand and where to go	Checks the floor has been added
endPointSetUp	Object	Local (setUp)	Variable	This means I have an endpoint	Ends the game	Check it has been added to the screen
characterHeroSetUp	Character	Local (setUp)	Variable	This means the user has a character to use	Means the user can see what they are doing. Sent as Global so anything can use it	Check it has been added to the screen
Root	Tkinter	Global	Constant (Variable in Python)	This means I can use Tkinter	This allows me to use Tkinter as my GUI. Sent as Global so anything can use it	None
frame	Window	Global	Variable	This means I have a window for the GUI	Allows me to use a window. Sent as Global so anything can use it	None
endpoint	Object	Global	Variable	This means I have an	Ends the game. Sent as	None

				endpoint	Global so anything can use it	
hero	Charac ter	Global	Variable	This means the user has a character to use	Means the user can see what they are doing. Sent as Global so anything can use it	Check it is all working so can move left, right and shoot
x0EndPoint	375	Local (finishGame)	Constant (Variable in Python)	I have one coordinate so I know where the final block is	Allows me to know where one edge of the block is so I can end it at the right time	Check this works by going over it and ending the game
enemy	Charac ter	Global	Variable	This the enemy will be able to fight against	This gives the user actual danger to fight against.	
Character	Class	Global	-	This means I can have characters in my game	This will mean I can use the same code for both my Hero and Enemy	Validation – Check it can be put on screen, shoot, move left or right, and lose health
Bullet	Class	Global	-	This will allow the program to have weapons and means the user can score points	This is a superclass so it will have code which both sub classes will need	Validation – Power, energy are both correct, it can move when need to and move the correct distance
HoriBullet	Class	Global	-	This will mean the user can shoot a horizontal bullet which moves the correct distance	This will mean the User can use different bullets in different situations.	Validation – Goes correct distance, will delete if hits something.
power	0	Local (Bullet)	Variable	This will be where the power is stored when it has been worked	This will mean we can work out how much health should	Checks if the right power has been worked out

				out	be lost	
energy	0	Local (Bullet)	Variable	This will be where the energy is stored when it has been worked out	This will mean the program can work out how much power is needed	Works out the correct energy
Ball	Tkinter Object	Local (Bullet)	Variable	Shows the bullet on the screen	This will mean the user can see the bullet on the screen	The bullet is on the screen
finalSpeed	0	Local (Bullet)	Variable	Works out the final speed on the bullet	This will mean we can work out the correct power of the bullet. Needed in both sort of bullets	Works out the correct final speed
distanceHori	0	Local (Bullet)	Variable	Works out the distance the bullet goes horizontally	Means the bullet can move the correct distance	Works out the correct distance
distanceVertical	0	Local (Bullet)	Variable	Works out the distance the bullet goes vertically	Means the bullet can move the correct distance	Works out the correct distance
health	200	Local (Character)	Variable	This will be the amount of health the character have	This will be used to see whether either of the characters are dead yet	That when health is less than 0 the character dies.
bulletShootingHori	HoriBullet	Local (fireHoriBulletClass)	Variable	This will be the horizontal bullet being shot	This will be the instance of the bullet and is needed for the game to be worked	When bullet shot it appears.

distPer10Milli	distanceToGo /50	Local (fireHoriBullet)	Variable	Works out how far the bullet should move on the screen	This is needed so the bullet looks natural on the screen whilst moving	When bullet shot it appears on the screen a few times
timer1	Time Bullet has moved	Local (fireHoriBullet)	Variable	Works out how long the bullet has been on the screen	Needed for working out the power of the bullet	Checks it takes a reading between 0 and 5.
bulletFiring	HoriBullet	Local (fireHoriBullet)	Variable	This is the bullet which is being fired at that point	This is needed so the bullet can interact with the environment.	Check it moves when it is fired
ParabolicBullet	Class	Global	Class	This is the bullet which will go in a parabolic arc	This is needed to get a bullet moving	That the bullet moves in a parabolic arc
finalSpeedVertical	0	Local (ParabolicBullet)	Variable	This will work out the vertical speed of a bullet	Needed to work out the energy at the end	Check that the values work out
bulletGoneUp	0	Local (fireParaBullet)	Variable	This will say how high the character has gone	Needed to check if it has reached the correct height and how far the bullet has gone	Check it is added after every loop in the while loop
bulletGoneDown	0	Local (fireParaBullet)	Variable	This will say how low the character has gone	Needed to check if it has reached the correct height and how far the bullet has gone	Check it is added after every loop in the while loop
goDown	True	Local (fireParaBullet)	Variable	This will say whether the bullet needs to go down	This is needed as if the bullet has hit the character if it is going up it does not need to go down	If the bullet hits the character when going up the bullet does not go down.

heroHealth	-	Local (hero_Lose_ Health)	Variable	This will be how much health the character has	If this is less than 0 the hero is killed	If this goes below 0 the Hero died
finalTimerStar t	-	Global	Variable	This will hold the time when the user started the game	This will used to work out the final time	This will be tested to check we get a float in the final score
finalTimerEnd 2 / finalTimerEnd	-	Local	Variable	This will hold the time the user ends the game	Needed to work out the time for the final game	Check the score is a float
finalScore	-	Local (Scoring)	Variable	This will hold the final score of the game	Needed to hold the score for the user and work it out	Output a score
Scores.csv	-	Global	CSV file	This will hold the score so it can be transferred across the different files which need it	This is needed so the score can be added onto the database which program is in a completely separate file to the game	That the score is inputted to the database
highscores.db	-	Global	Database	This will hold the scores of all the people who have played the game and also the user's username	This is needed to show all the scores of different players who have played the game	That I can output it into Tkinter
newID	-	Local (username_E ntering)	Variable	This will hold the unique primary key for the next entry for the database	Needed so I have a unique user ID for the database	Each entry in the database has a unique entry into the database
toAdd	-	Local(adding _To_Databas e)	Variable	This will hold what needs to be added to the database	Needed so I can add the next entries into the	AN entry is added onto the database

					database	
databaseScreen	Tkinter window	Global	Variable	This will be the screen for the database	Needed to output asking for the username	This will be tested that a new screen is added after the game ends
addingUsername	-	Global	Variable	This will allow the user to enter their username	Needed so the user can enter their username	This will be tested to see if it is on the screen
labelChange	-	Global	Variable	This is needed to say what to do	Needed to say to enter the username and also so the user knows what is happening	Text changes after the username is entered
buttonAskingForUsername	-	Global	Variable	When pressed it enters the username	Needed so the program knows it can take the username	Enter a username then press the button and name appears in screen
cursor	-	Local (create_Table)	Variable	This will help create the table	Needed so the program can create the table correctly	A database is created
table	Database Table	Local (display_Table)	Variable	This will hold the current table	Needed so it can be sorted	A database can be shown
sortedTable	-	Local (display_Database)	Variable	This will hold the sorted table	Needed so the user can see a sorted table	Table is outputted which is sorted to largest scores to smallest scores
dataArray	-	Local (quick_Sort)	Variable	Holds what needs to be sorted	Needed so the table can be sorted	Database which is sorted
pivot	-	Local (quick_Sort)	Variable	Holds the number which is used to say whether the score is larger or	Needed so it can be sorted	Database is sorted when outputted

				smaller		
first_part	-	Local (quick_Sort)	Variable	This holds the first half of the dataArray	Needed to sort out each half of the database	Database is sorted correctly when outputted
second_part	-	Local (quick_Sort)	Variable	This holds the first half of the dataArray	Needed to sort out each half of the database	Database is sorted correctly when outputted
tableDisplay	-	Local (make_Text)	Variable	The final text to be outputted for the leaderboard	Needed to add all the data in a way which is consumable to the user	Database is outputted in rows.
databaseShow Screen	-	Global	Variable	This will be the GUI	This will mean the user can see the database	Window is made when leaderboard is pressed
labelWithTabl e	-	Global	Variable	This will be the label for the database to be outputted	This is needed so the user can see the current leaderboard	Leaderboard is outputted when leaderboard button pressed
instructionScr een	-	Local (instructions)	Variable	The GUI for the instructions screen	Needed to output the instructions	New window when instructions button pressed
instructionsTo Do	-	Local (instructions)	Variable	This will be the label which outputs the instructions for the game	Needed to output the instructions for the user to read	Instructions are outputted to the instructions screen
window	-	Global	Variable	This will be the window for the menu	Needed to show the menu and for the user to start the game	When program started the screen is made
buttonInstruct ions	-	Global	Variable	This will be the button to go to the instructions screen	Needed for the user to go to the instructions	Button saying instructions is in the menu
buttonGame	-	Global	Variable	This will be the button to go to	Needed for the user to play	Button saying Game is in the

				the Game	the game	menu
buttonLeader board	-	Global	Variable	This will be the button to go to the leaderboard screen	Needed for the user to go to the leaderboard	Button saying Leaderboard is in the menu

Identify and justify any further data to be used in post development phase.

Test Number	Statement	Justification	Input	Expected Output	Actual Output	Notes
1	The program will work. The player will be able to play a game which is realistic and is a shooting 2D platform game with an enemy which you need to defeat in it. The program will be robust enough to not break if a wrong input is entered.	I do not want there to be any game breaking bugs or glitches so I will want to have identified them before I have finished. I want the game to be finished and that the user feels that it is polished and finished.	-	The game does not unexpectedly crash.		
2	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character and this will be the way to actually interact with the level.	Valid - Right Key	The character moves to the right		
3	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character and this will be the way to actually interact with the level.	Valid - Left Key	The character moves to the Left		
4	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character and this will be the way to actually interact with the level.	Valid - Up key	The character moves up then down landing on the floor.		
5	The program will have an enemy	This is so there is some difficulty in the game.	-	There is more than one character on the level		
6	The program will end when the user's character	This means the level will end at one point and will not go on	Valid - Reach the endpoint	The Game ends showing the		

	reaches an endpoint.	forever.		next screen		
7	The program will allow the user to change which bullet to use	This means they can use both bullets.	Valid – Press q, e, q	This should shoot 2 bullets		
8	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid -Press q	The enemy should shoot a bullet. Hits the user's character and disappears		
9	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid - Press q	Once the enemy's bullet disappears the user will fire a bullet and when it hits the enemy, this too will be deleted		
10	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid -Press w	The hero will fire a bullet and when it hits the enemy it will disappear		
11	The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Valid -Press q	When bullet hits the hero, it will output the power of it and the health remaining		
12	The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Valid -Press q	When the bullet hits the enemy, it will output the power and the health remaining		
13	The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Valid -Press w	When the bullet hits the enemy, it will output the power and the health remaining		
14	The program will	This make the game	Valid – Press q	The program		

	make the bullet go the correct distance using SUVAT.	realistic		will output how far the enemy bullet is going (Will be a float)		
15	The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Valid – Press q	The program will output how far the hero's bullet is going (Will be a float)		
16	The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Valid – Press q	The program will output how far the bullet is going (Will be a float)		
17	The program will have two bullets a parabolic one and a horizontal one.	This allows the user to have more than one weapon	-	There is two bullets the user can use		
18	The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.	q	The number of bullets left decreases by 1		
19	The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.	w	The number of bullets left decreases by 1		
20	The program will have a timer which allows it to work out how long the user has taken.	This is to be used in the total points and allows the user to try and make it quicker	-	There is a time at the end which is similar to the time I will take on my phone		
21	The program will work out a total.	This is to give a point for the game	-	At the end the program will work out a score		
22	The program will allow the user to see how well they have done	This will allow the user to see how well they have done	-	The score is outputted		
23	The program will have a database of high scores	This is so all scores can be entered and also so the user can see how well they are doing.	Running database_Creator	highscore.db is made in same file		

24	The program will put the score in a database	This is so the user can see how they compare to other people.	-	The score is added into the database (We can see it on the high scores page)		
25	The program will allow the user to enter a username.	This is so it can go into the database along with their score.	Valid: 10low1	This is accepted		
26	The program will only allow usernames which no one else has used before.	This is so it is easy for the User to see their score in the database	Invalid: 10low1	This is not accepted		
27	The program will allow the user to choose whether to see the high scores or to play the game	This will be a menu and allow the user to see different parts of the program.	-	There is a menu showing different options		
28	The program will display a table of high scores.	This allows the user to see how well they are doing.	Press Leaderboard	There is a leaderboard		
29	The program will make sure all characters are on the ground when jumping	This will then make sure it looks realistic and also means you can see how high they are.	Invalid: Press up button multiple times	Only jumps once until it reaches the ground		
30	The program will end if the user's character hits 0.	This is so there is a risk if you are hit by a bullet	Touch enemy 4 or more times	The program will open the page asking for the username		
31	The program will have a bullet being shot by the enemy every 5 seconds	This is so there is a continuous risk to the user.	-	There is a bullet shooting every 5 seconds		
32	The program will allow any characters in the username	This means it's easy for the user to have any name	Valid: hiTh1s @h!	This is accepted		
33	The program will have an instructions screen	This will allow the user to know how to play the game	Press Instructions in menu	Instructions are outputted		
34	The program will have a menu	Allows the user to go to different parts of the program	-	There is a menu		
35	The program will have a	This will allow the user to see the	Press Leaderboard	This will output the		

	leaderboard screen	current best scores		leaderboard		
36	The program will output the leaderboard from the largest score (Best) to the smallest score (worst)	This will make it easy for the user to see the best and worst scores	-	Out of the multiple scores largest is top smallest bottom		
37	The program will take away health if the user's character touches the enemy	This means there is some danger to touching the enemy and means you cannot just run straight to the endpoint	Touch enemy	Enemy will lose health		
38	The game will do nothing if you press buttons which are not q,w or the arrow keys.	This means the game cannot break if you enter an accidental input.	Press g	Nothing		

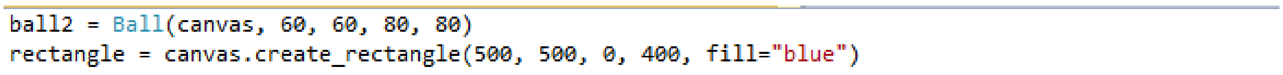
Development

Provide evidence of testing at each stage of the iterative development process and provide evidence of any failed tests and the remedial actions taken with full justification for any actions taken.

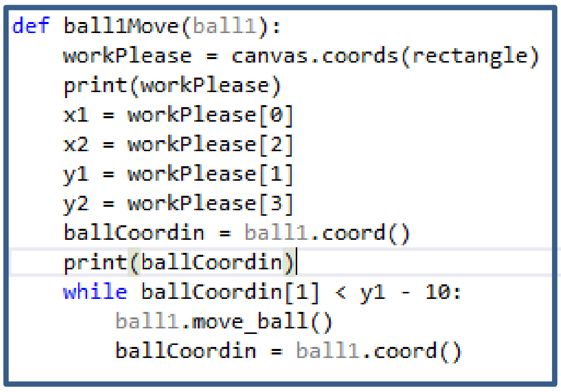
Wherever **Testing** appears, it means that there is some iterative testing at this point of the development

Prototype 0

This is code where I have been able to move an object however it is completely autonomous, meaning it will start to run as soon as the program starts. It is proof of concept that I can get objects to move in Tkinter thus meaning my project will be able to go ahead.

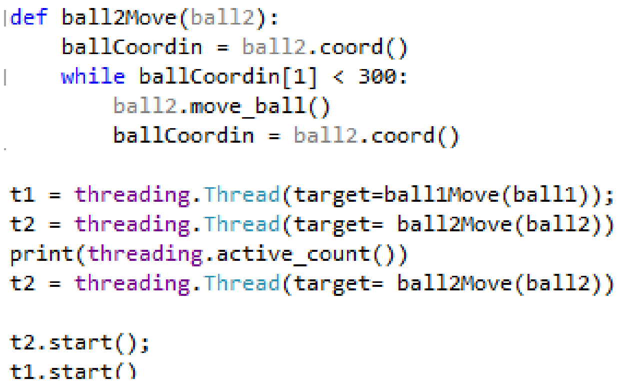


```
ball2 = Ball(canvas, 60, 60, 80, 80)
rectangle = canvas.create_rectangle(500, 500, 0, 400, fill="blue")
```



```
def ball1Move(ball1):
    workPlease = canvas.coords(rectangle)
    print(workPlease)
    x1 = workPlease[0]
    x2 = workPlease[2]
    y1 = workPlease[1]
    y2 = workPlease[3]
    ballCoordin = ball1.coord()
    print(ballCoordin)
    while ballCoordin[1] < y1 - 10:
        ball1.move_ball()
        ballCoordin = ball1.coord()
```

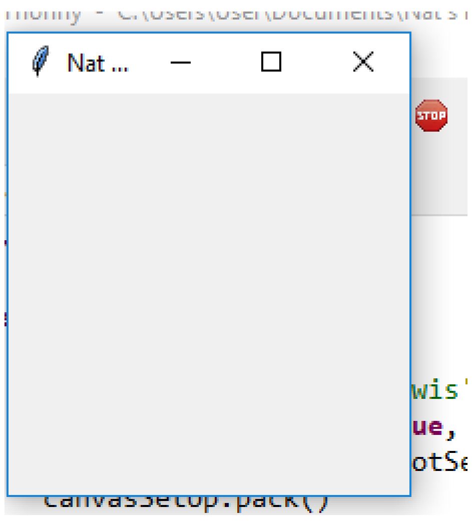
This part of the code I edited for it to stop before reaching the 'floor' from the code mentioned earlier.



```
def ball2Move(ball2):
    ballCoordin = ball2.coord()
    while ballCoordin[1] < 300:
        ball2.move_ball()
        ballCoordin = ball2.coord()

t1 = threading.Thread(target=ball1Move(ball1));
t2 = threading.Thread(target= ball2Move(ball2))
print(threading.active_count())
t2 = threading.Thread(target= ball2Move(ball2))

t2.start();
t1.start()
```

First prototype

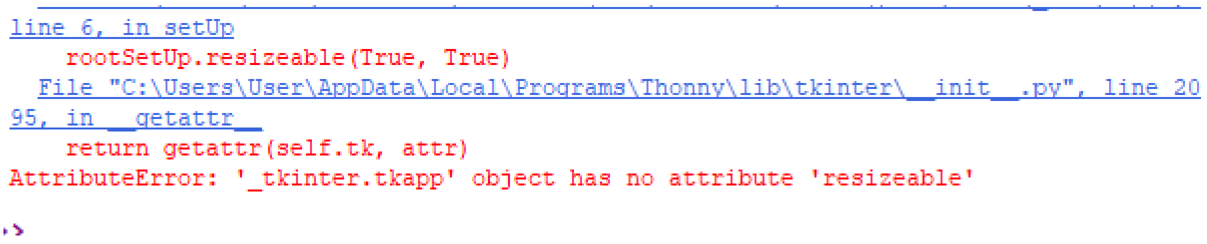
My first prototype will be very simple. I will make a start on the character class (move left and right methods). I will also program a simple level where once reaching the end, the canvas deletes itself. To make things slightly simpler I will do each class in separate windows then add them all together at the end. This is much simpler for me to write the code and also able to test each class by itself.



```
wis'
ue,
otSe
canvasSetup.pack()
```

Testing I started by making the canvas and this happened:

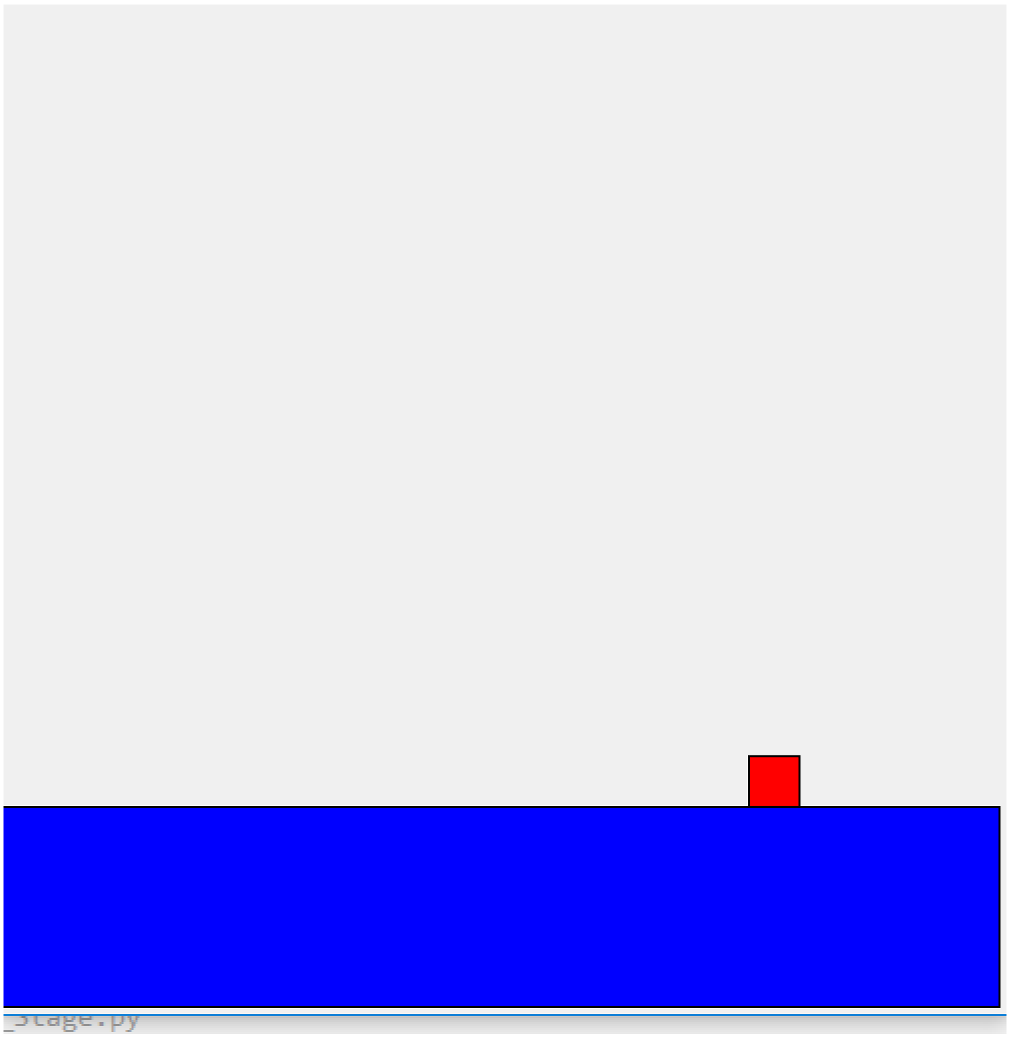
This is ever so slightly too small. The problem is that the Tkinter has no attribute resizable.



```
line 6, in setUp
    rootSetUp.resizeable(True, True)
  File "C:\Users\User\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 20
95, in __getattr__
    return getattr(self.tk, attr)
AttributeError: '_tkinter.tkapp' object has no attribute 'resizeable'
```

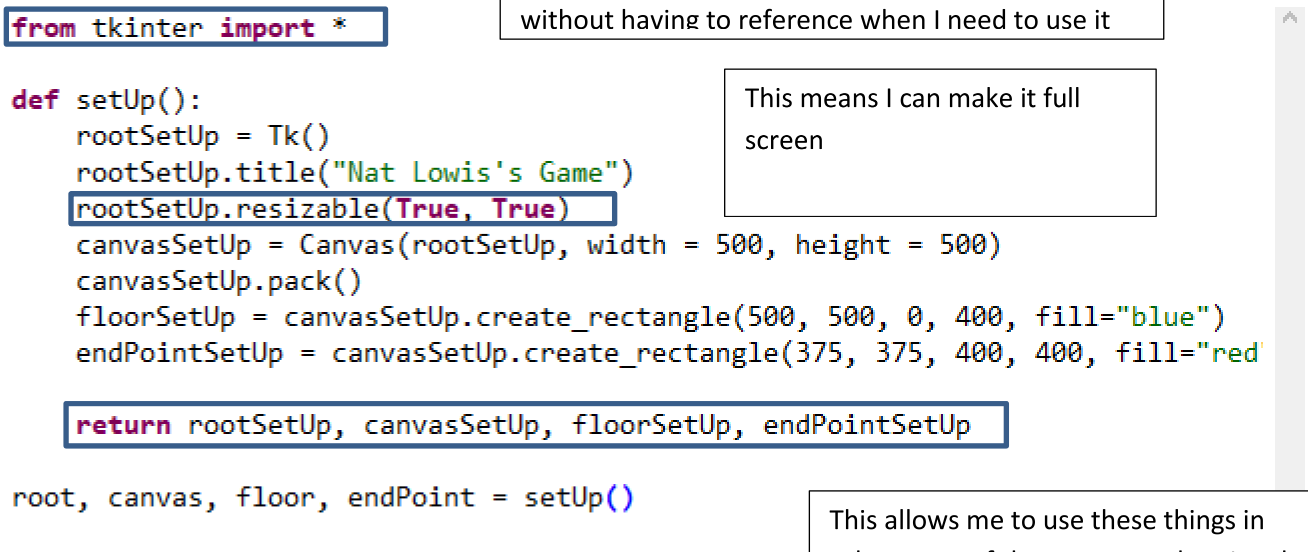
This is just a spelling error and after changing it I was able to get the canvas to appear the correct size. This will allow the user to be able to play the game by being able to see everything.

I then wrote code to try and add a floor, but I was not able to get it into the right position along with an endpoint as they were too long. I finally used a website which helped me[1] and I have it just above the floor like this:



Code for setUp prototype 1:

This makes it easier to use the methods from tkinter without having to reference when I need to use it



```
from tkinter import *

def setUp():
    rootSetUp = Tk()
    rootSetUp.title("Nat Lowis's Game")
    rootSetUp.resizable(True, True)
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)
    canvasSetUp.pack()
    floorSetUp = canvasSetUp.create_rectangle(500, 500, 0, 400, fill="blue")
    endPointSetUp = canvasSetUp.create_rectangle(375, 375, 400, 400, fill="red

    return rootSetUp, canvasSetUp, floorSetUp, endPointSetUp

root, canvas, floor, endPoint = setUp()
```

This means I can make it full screen

This allows me to use these things in other parts of the program when I make other parts of the game

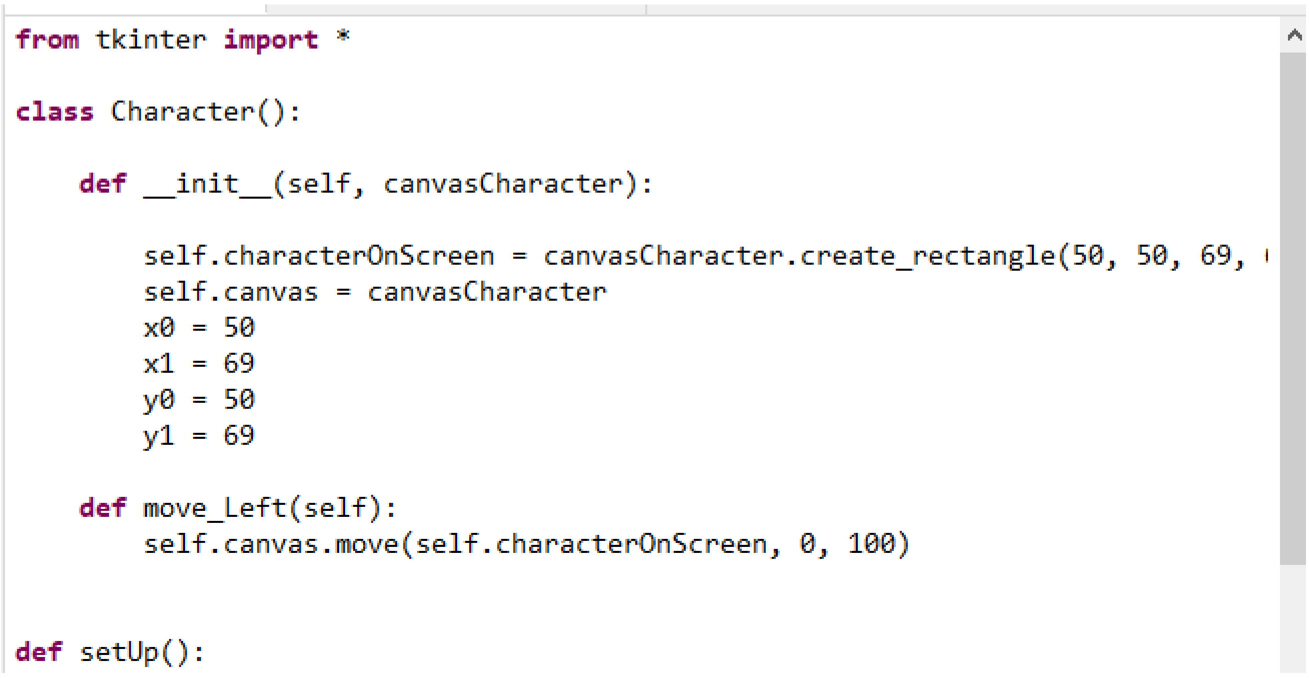
[1] http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_rectangle.html In Bibliography

I have made comments on parts which are important but the parts which do not have comments will be commented in the actual program. This code allows me to make the canvas the correct size whilst a floor and endpoint are set up as well.

For the character I have taken code off effbot[2] and am hopefully going to use it to make the character move left. The code I have taken is a way to move objects in Tkinter thus allowing the user to move the character.

Testing I then tried to validate the character to see whether it would move when the user inputted the left key. When I first ran it, I had a couple of errors due to spelling mistakes but once sorted nothing happened. So, I copied and pasted some code off effbot[3] again and this worked in a separate file. I put this code in my actual file and it did not work. It has made the window which holds my game larger than it should be, so I am going to work out why it does this.

The code at the moment:

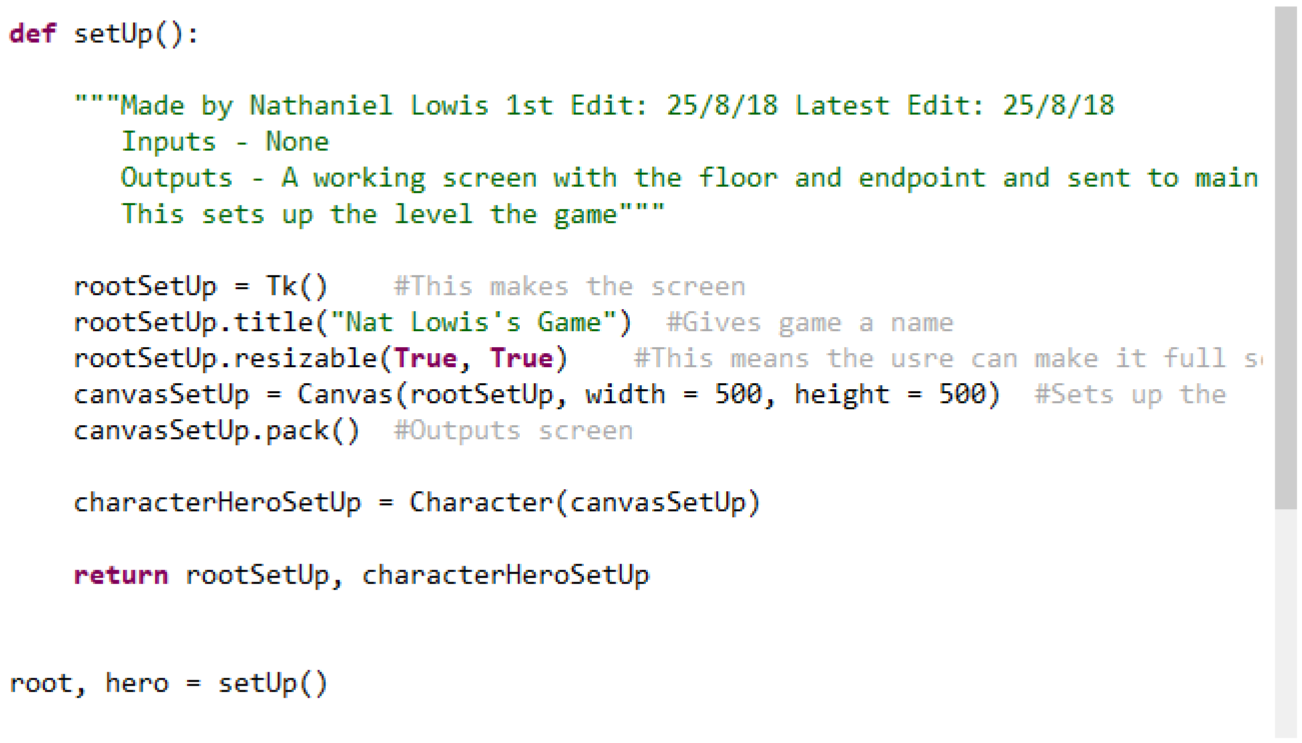


```
from tkinter import *

class Character():

    def __init__(self, canvasCharacter):

        self.characterOnScreen = canvasCharacter.create_rectangle(50, 50, 69, 
        self.canvas = canvasCharacter
        x0 = 50
        x1 = 69
        y0 = 50
        y1 = 69

    def move_Left(self):
        self.canvas.move(self.characterOnScreen, 0, 100)


def setUp():
```

This shows the basic class of Character where I have specified where two of the corners are.

[2] http://effbot.org/tkinterbook/canvas.htm In Bibliography
[3] http://effbot.org/tkinterbook/canvas.htm In Bibliography

The setUp module is a simplified version from the one I have made making it easier for me to just focus on the character working:



```
def setUp():

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/8/18
       Inputs - None
       Outputs - A working screen with the floor and endpoint and sent to main
       This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the usre can make it full s
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)  #Sets up the
    canvasSetUp.pack()  #Outputs screen

    characterHeroSetUp = Character(canvasSetUp)

    return rootSetUp, characterHeroSetUp


root, hero = setUp()
```

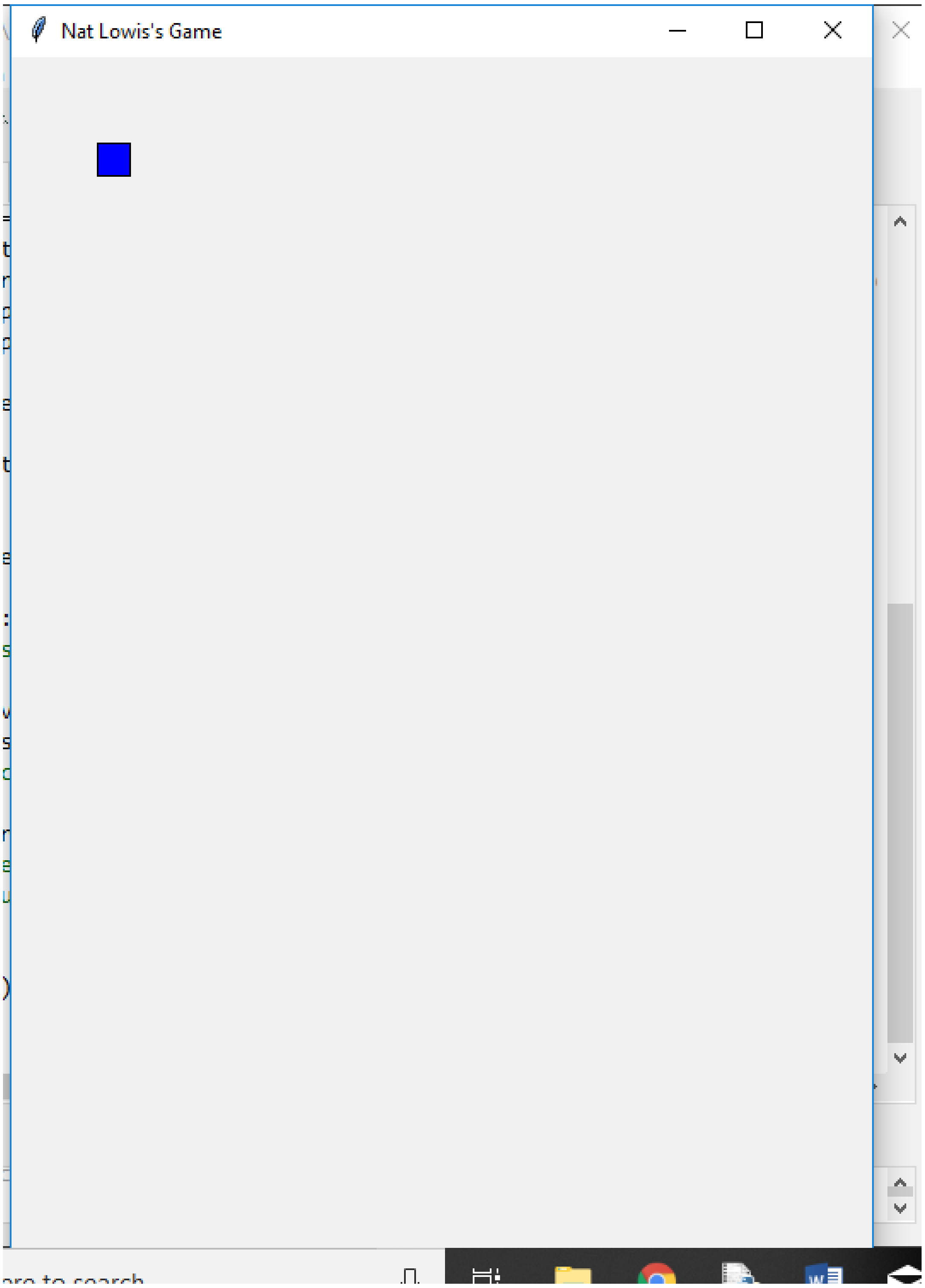
Finally, I have code which I am using to = test to see whether it is working. It is not:

```
def key(event):
    print("pressed", repr(event.char))

def callback(event):
    frame.focus_set()
    print("clicked at", event.x, event.y)

frame = Frame(root, width=500, height=500)
frame.bind("<Key>", key)
frame.bind("<Button-1>", callback)
frame.pack()

root.mainloop()
```

This is taken from effbot but is allowing me to test if my code is behaving as it should[4].

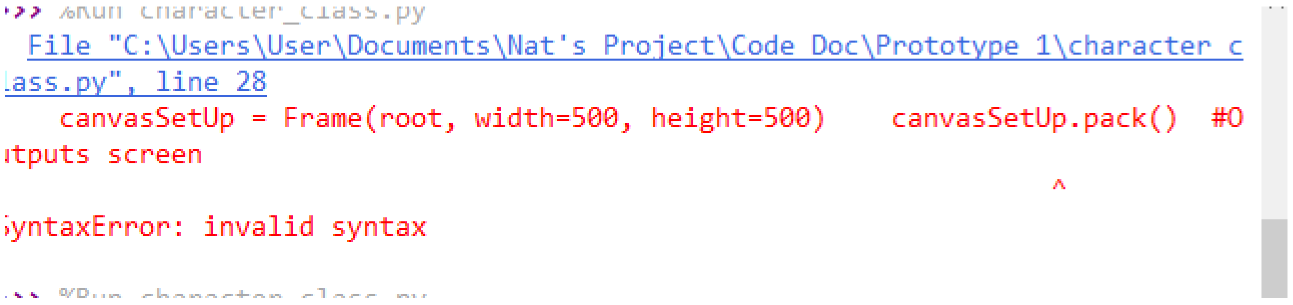
[4] http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm In Bibliography

Testing When I run the game this is the window which I get:

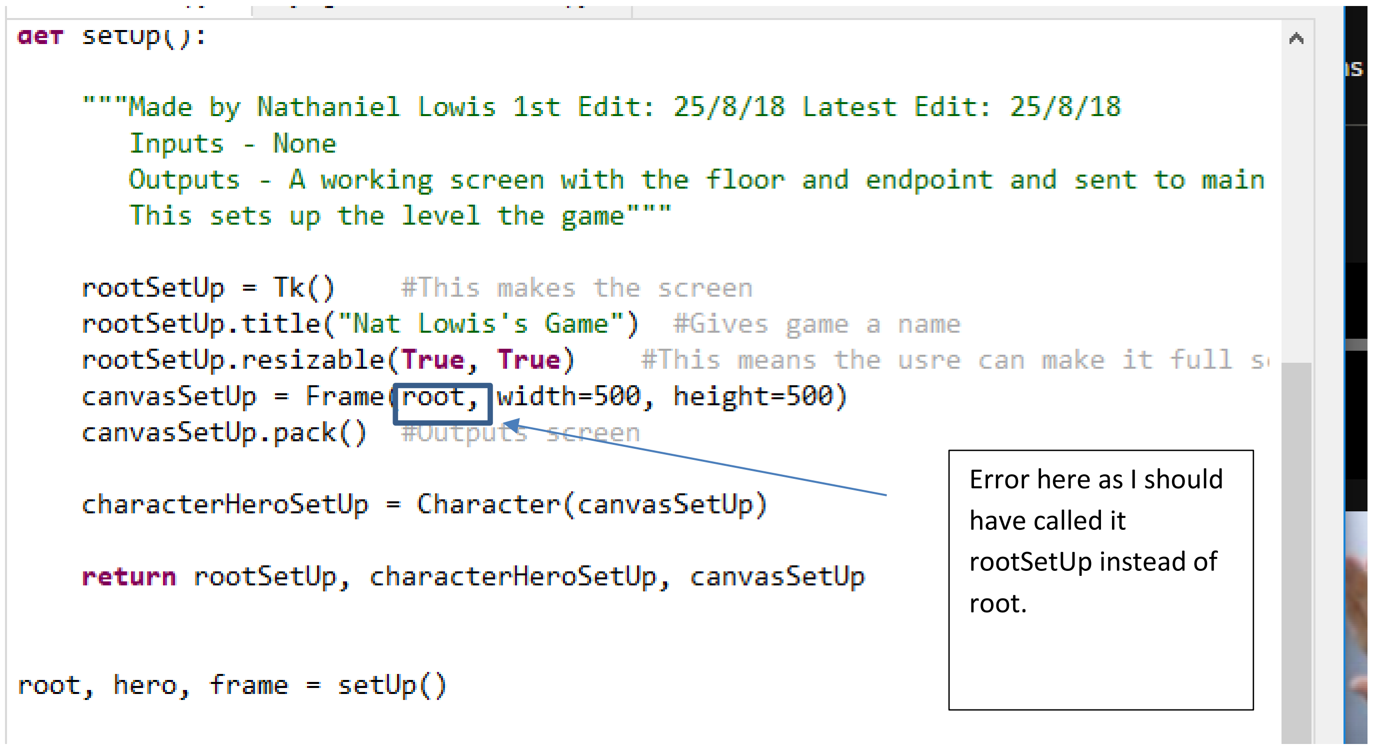


The problem is that I cannot interact with the screen by pressing the left key to move the object. I have decided that to try and remedy this, in the setUp I will make the window a frame instead of a canvas. This hopefully means I can start to get the player moving so the user can use it.

When I first ran it, I got a syntax error as I did not have part of the code on the right line:


```
>>> %Run character_class.py
  File "C:\Users\User\Documents\Nat's Project\Code Doc\Prototype 1\character_c
lass.py", line 28
    canvasSetUp = Frame(root, width=500, height=500)     canvasSetUp.pack()   #O
utputs screen
                                                                     ^
SyntaxError: invalid syntax

>>> %Run character_class.py
```

Basic SetUp Code at the moment:

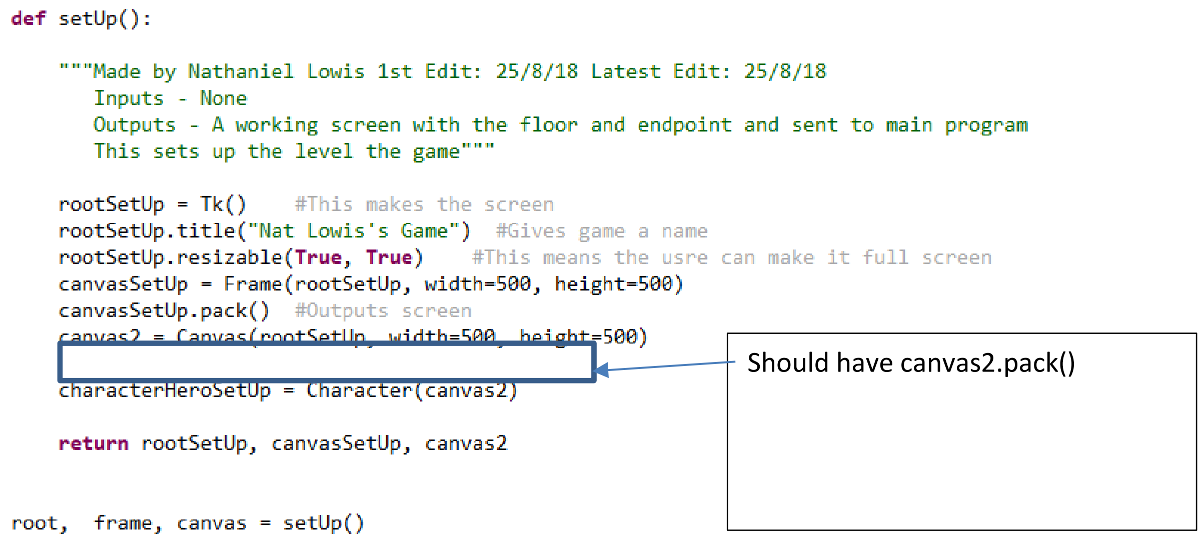

```
def setUp():

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/8/18
       Inputs - None
       Outputs - A working screen with the floor and endpoint and sent to main
       This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the usre can make it full s
    canvasSetUp = Frame(root, width=500, height=500)
    canvasSetUp.pack()  #Outputs screen

    characterHeroSetUp = Character(canvasSetUp)

    return rootSetUp, characterHeroSetUp, canvasSetUp


root, hero, frame = setUp()
```

Error here as I should have called it rootSetUp instead of root.

Unfortunately, frame cannot make the rectangle object.

I have realised that a canvas can create the character but cannot seem to bind keys to move objects. A frame can get something to bind to keys but not create the character model. Seemingly, I cannot have both a character as a canvas object and move it using user inputs.

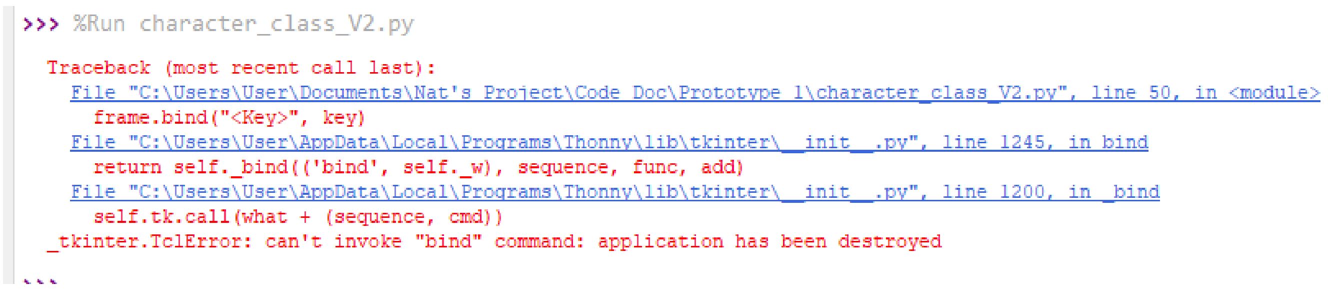
I then added a new canvas, but it did not work because I forgot to pack it:


```
def setUp():

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/8/18
       Inputs - None
       Outputs - A working screen with the floor and endpoint and sent to main program
       This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the usre can make it full screen
    canvasSetUp = Frame(rootSetUp, width=500, height=500)
    canvasSetUp.pack()  #Outputs screen
    canvas2 = Canvas(rootSetUp, width=500, height=500)

    characterHeroSetUp = Character(canvas2)

    return rootSetUp, canvasSetUp, canvas2


root,  frame, canvas = setUp()
```

Should have canvas2.pack()

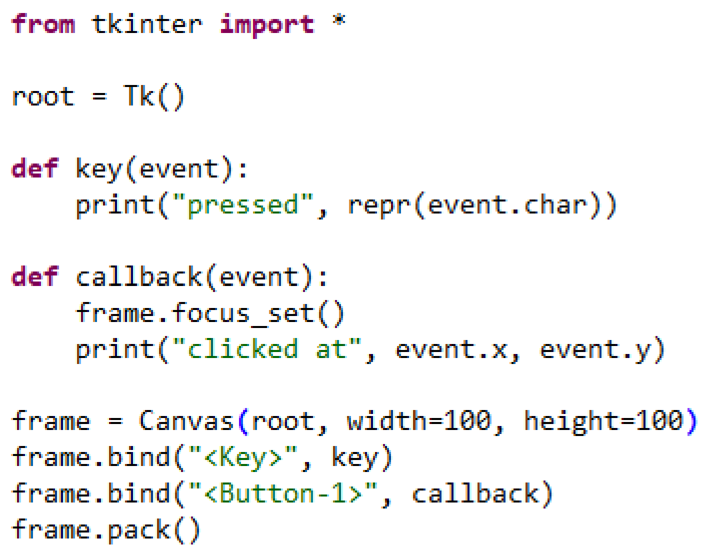
When I ran it, I came across the same problem as before.

I thought that when I run the code it was adding the frame and canvas together. However, this does not seem to be the problem either. I get this error code:



```
>>> %Run character_class_V2.py

  Traceback (most recent call last):
    File "C:\Users\User\Documents\Nat's Project\Code Doc\Prototype 1\character_class_V2.py", line 50, in <module>
      frame.bind("<Key>", key)
    File "C:\Users\User\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1245, in bind
      return self._bind(('bind', self._w), sequence, func, add)
    File "C:\Users\User\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1200, in _bind
      self.tk.call(what + (sequence, cmd))
  _tkinter.TclError: can't invoke "bind" command: application has been destroyed
```

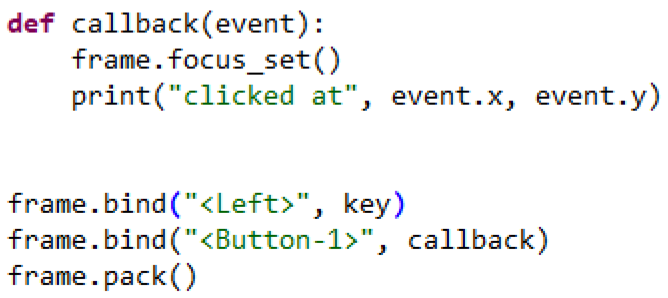
Obviously when I make the canvas I destroy the frame object which is unhelpful as it means I cannot use a canvas object and also move it. However, I changed frame to canvas on my testing code and it still worked.



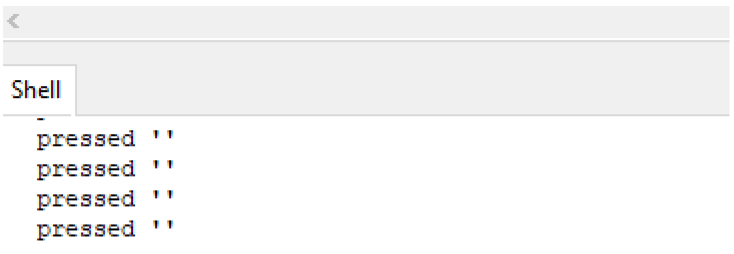
```
from tkinter import *

root = Tk()

def key(event):
    print("pressed", repr(event.char))

def callback(event):
    frame.focus_set()
    print("clicked at", event.x, event.y)

frame = Canvas(root, width=100, height=100)
frame.bind("<Key>", key)
frame.bind("<Button-1>", callback)
frame.pack()
```

I took out frame on my main code and changed it to a canvas and it worked.

Next, I will try and get the character to move left. This will allow the user to actually start controlling the character. I have changed frame.bind("<key>", key) to frame.bind("<Left>", key) which does work.

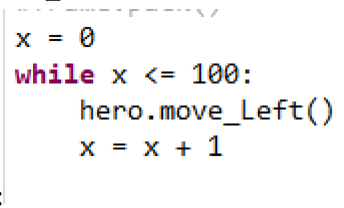


```
def callback(event):
    frame.focus_set()
    print("clicked at", event.x, event.y)


frame.bind("<Left>", key)
frame.bind("<Button-1>", callback)
frame.pack()
```



```
pressed ''
pressed ''
pressed ''
pressed ''
```

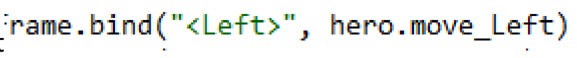
Testing Next I will try and change this to move the hero left a little bit. This did not work. After changing other parts of the code, which did not work, (like changing the button-1 callback to move_Left) I decided to look at some code from the internet[5] which I then have used a part of. I then noticed I did not use the update and after methods for the canvas so I added these in the move_Left method which did not help. However, I wrote this testing code into the move_Left function which did work:



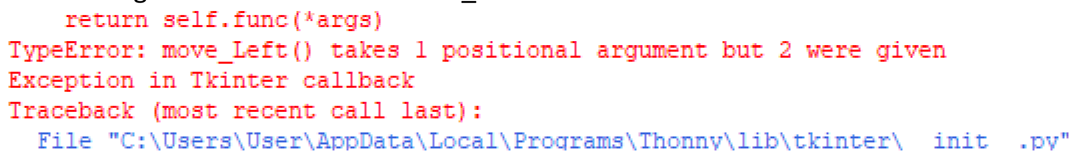
```
x = 0
while x <= 100:
    hero.move_Left()
    x = x + 1
```

So, I know my code works but I do not know why the character is not moving left when I press the left button.

I tried taking the brackets off the move_left which did not work:



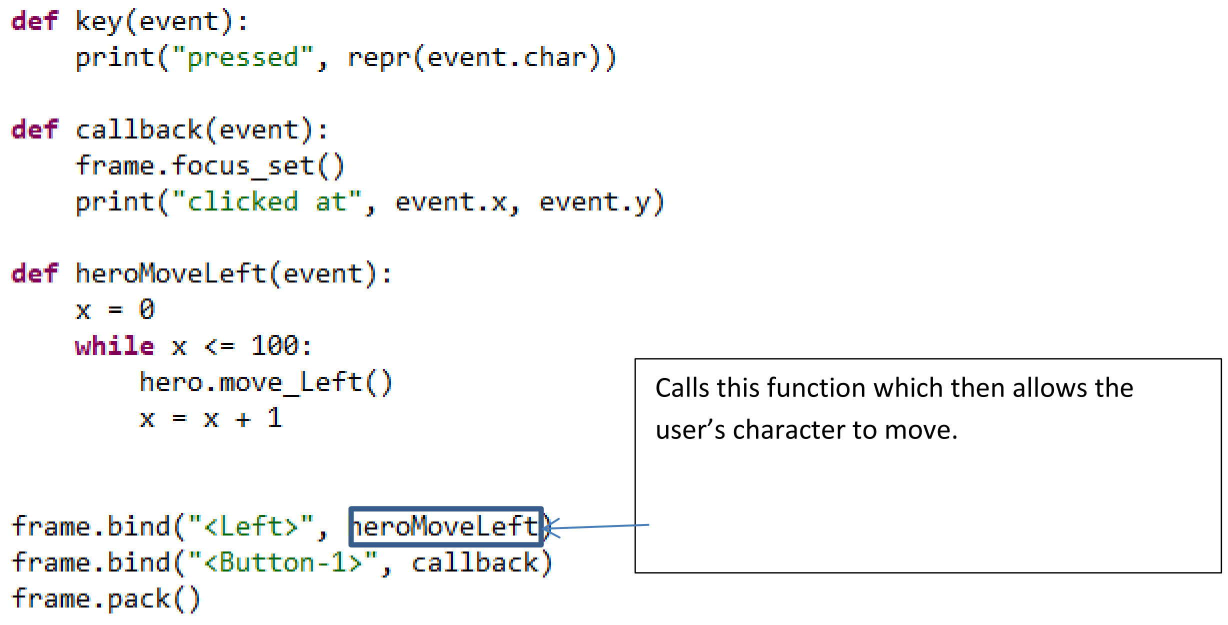
```
rame.bind("<Left>", hero.move_Left)
```



```
    return self.func(*args)
TypeError: move_Left() takes 1 positional argument but 2 were given
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\User\AppData\Local\Programs\Thonny\lib\tkinter\  init  .py"
```

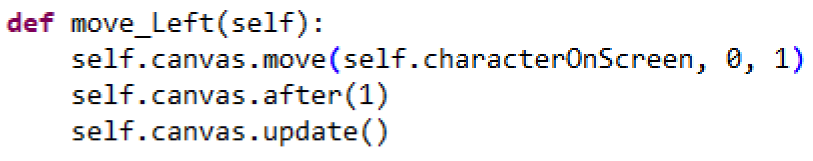
I then found the solution. I put the move_Left method into another procedure. Now, when the user presses the left button, it will call the procedure which will then call the move_Left method. The object moved downwards but I can forgive that as it is moving.

The code for the parts I have been programming:



```
def key(event):
    print("pressed", repr(event.char))

def callback(event):
    frame.focus_set()
    print("clicked at", event.x, event.y)

def heroMoveLeft(event):
    x = 0
    while x <= 100:
        hero.move_Left()
        x = x + 1


frame.bind("<Left>", heroMoveLeft)
frame.bind("<Button-1>", callback)
frame.pack()
```

Calls this function which then allows the user's character to move.

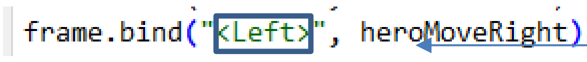


```
    def move_Left(self):
        self.canvas.move(self.characterOnScreen, 0, 1)
        self.canvas.after(1)
        self.canvas.update()
```

This moves the hero down quite a bit. I will take out the while loop and just have hero.Move_Left() in the module heroMoveLeft and change self.canvas.moveto (-10, 0) which moves the character left! I tried this couple of times and it kept moving left. I copied the move_Left code converted it to make the character move right and then copied

[5] https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas In Bibliography

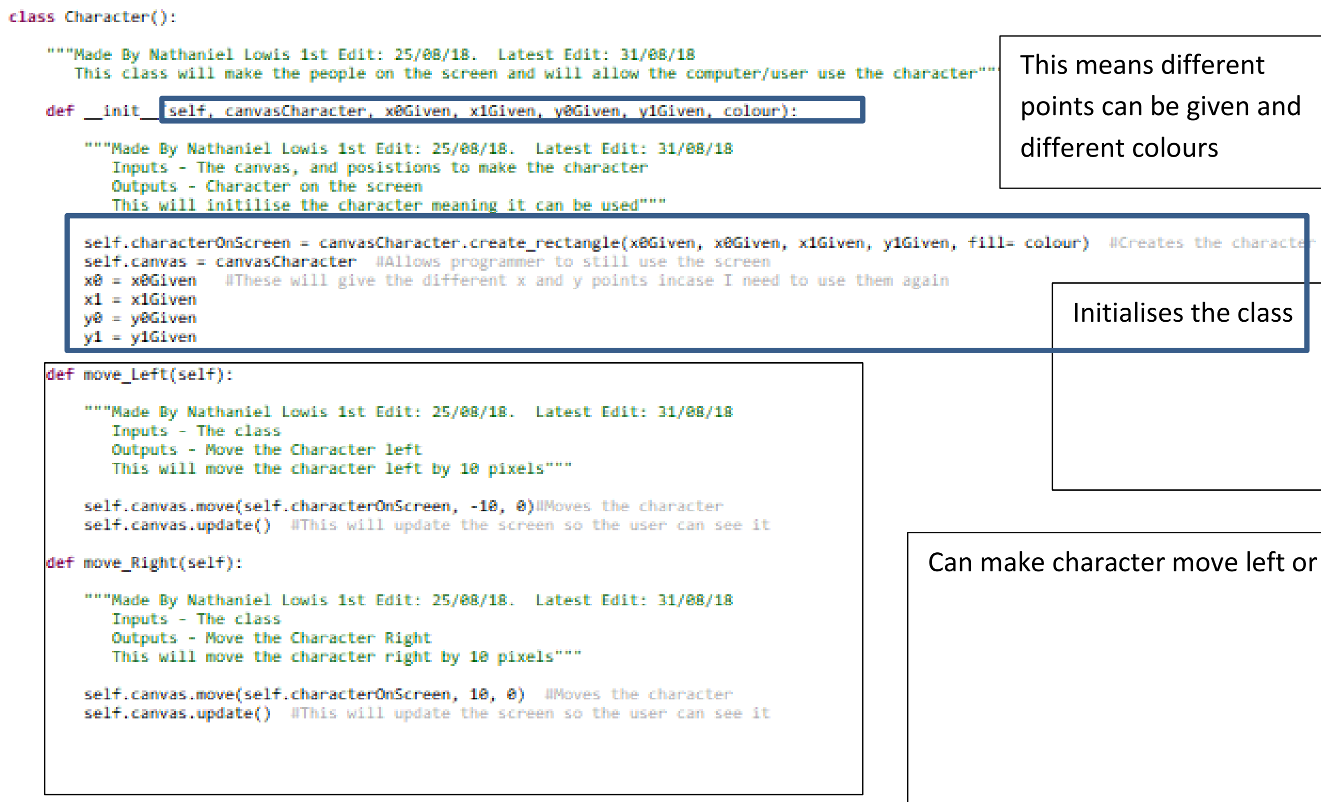
and pasted the heroMoveLeft code, called it heroMoveRight and converted it and it failed.



```
frame.bind("<Left>", heroMoveRight)
```

Need to change to Right

This then worked! So, I now have code to move the character left and right, meaning the user can start to control the character. Now, when the Character gets to the endpoint the level is deleted so the user can have a goal to the game. Before that I will comment the code and put docstrings in it. Code for class character:

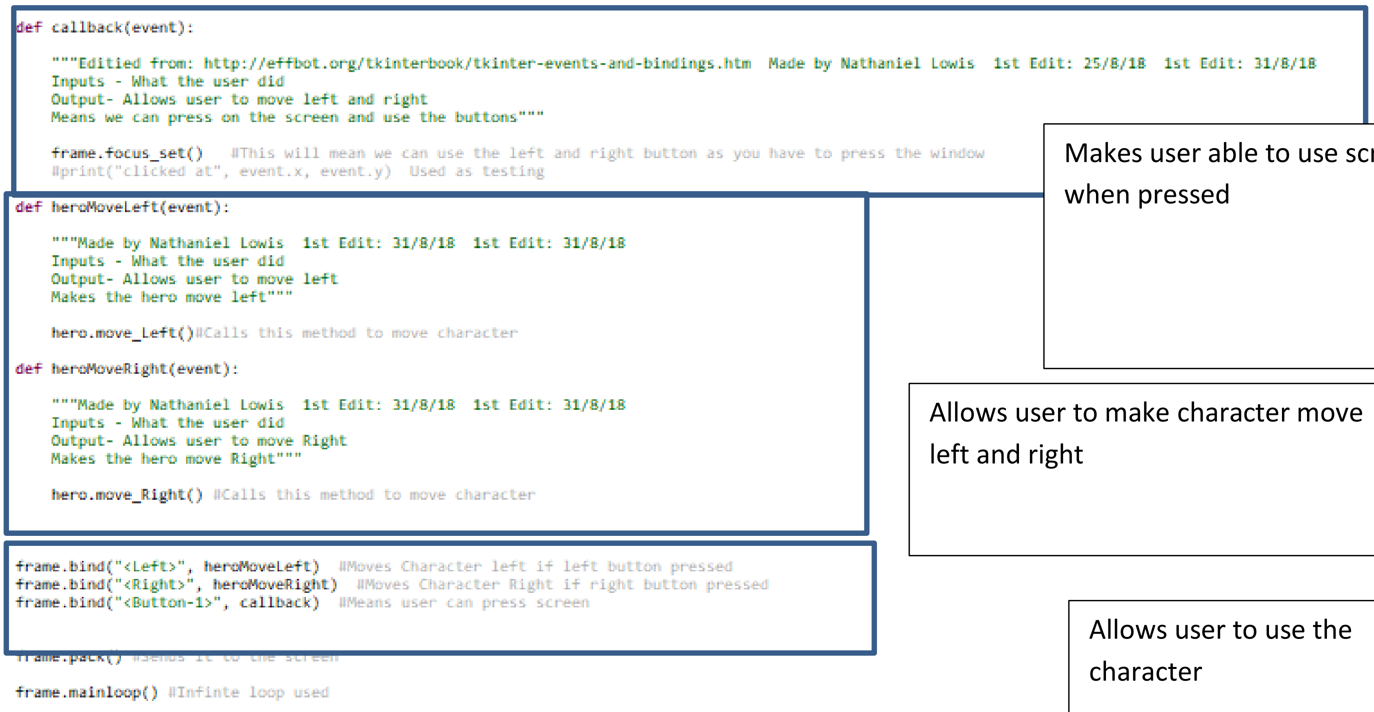


```
class Character():

    """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
       This class will make the people on the screen and will allow the computer/user use the character"""

    def __init__(self, canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
           Inputs - The canvas, and posistions to make the character
           Outputs - Character on the screen
           This will initilise the character meaning it can be used"""

        self.characterOnScreen = canvasCharacter.create_rectangle(x0Given, x0Given, x1Given, y1Given, fill= colour)  #Creates the characte
        self.canvas = canvasCharacter  #Allows programmer to still use the screen
        x0 = x0Given   #These will give the different x and y points incase I need to use them again
        x1 = x1Given
        y0 = y0Given
        y1 = y1Given

    def move_Left(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
           Inputs - The class
           Outputs - Move the Character left
           This will move the character left by 10 pixels"""

        self.canvas.move(self.characterOnScreen, -10, 0)#Moves the character
        self.canvas.update()  #This will update the screen so the user can see it

    def move_Right(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
           Inputs - The class
           Outputs - Move the Character Right
           This will move the character right by 10 pixels"""

        self.canvas.move(self.characterOnScreen, 10, 0)  #Moves the character
        self.canvas.update()  #This will update the screen so the user can see it
```

This means different points can be given and different colours

Initialises the class

Can make character move left or right



```
def callback(event):

    """Editied from: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm  Made by Nathaniel Lowis  1st Edit: 25/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left and right
    Means we can press on the screen and use the buttons"""

    frame.focus_set()   #This will mean we can use the left and right button as you have to press the window
    #print("clicked at", event.x, event.y)  Used as testing
def heroMoveLeft(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left
    Makes the hero move left"""

    hero.move_Left()#Calls this method to move character

def heroMoveRight(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move Right
    Makes the hero move Right"""

    hero.move_Right() #Calls this method to move character

frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button pressed
frame.bind("<Button-1>", callback)  #Means user can press screen

frame.pack() #Sends it to the screen

frame.mainloop() #Infinte loop used
```

Makes user able to use screen when pressed

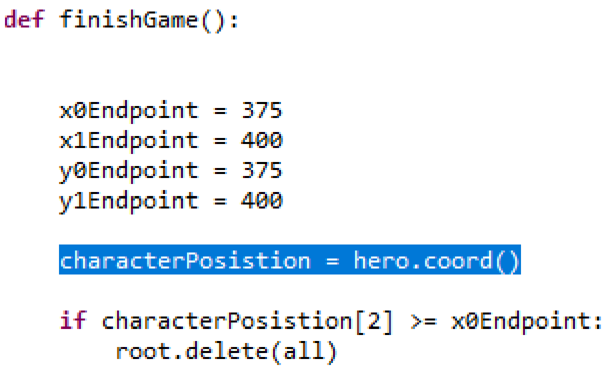
Allows user to make character move left and right

Allows user to use the character

The final part of my first prototype will be to make the character finish the level.   Initially this requires me to put the first two parts together.

This was accomplished without many problems (Minor problems were caused by not naming everything correctly).

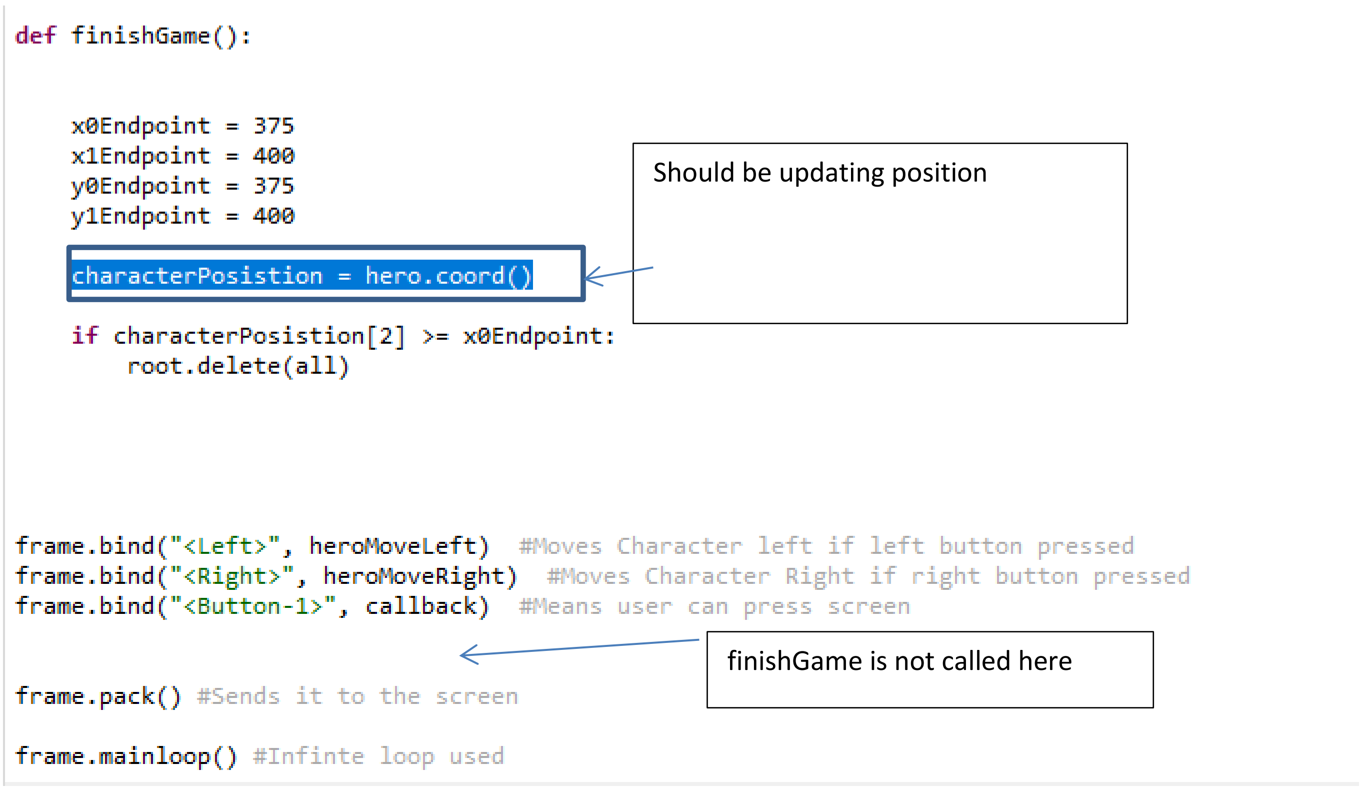
I have made a new method in the class character called coord which will give the coordinates of the character. I need this to see whether I have finished the game. This is the first iteration of finishGame:



```
def finishGame():


    x0Endpoint = 375
    x1Endpoint = 400
    y0Endpoint = 375
    y1Endpoint = 400

    characterPosistion = hero.coord()

    if characterPosistion[2] >= x0Endpoint:
        root.delete(all)
```

The endpoint will be constant, therefore I will not find the coordinates but will have the endpoint as a programmed variable.

Testing I got to the endpoint and the game did not finish as I did not update the character's position and did not call it in the main program.

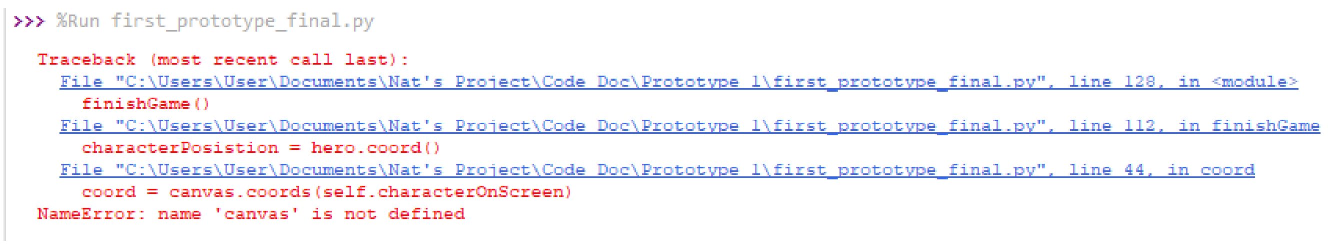


```
def finishGame():


    x0Endpoint = 375
    x1Endpoint = 400
    y0Endpoint = 375
    y1Endpoint = 400

    characterPosistion = hero.coord()

    if characterPosistion[2] >= x0Endpoint:
        root.delete(all)




frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button pressed
frame.bind("<Button-1>", callback)  #Means user can press screen


frame.pack() #Sends it to the screen

frame.mainloop() #Infinte loop used
```

Should be updating position

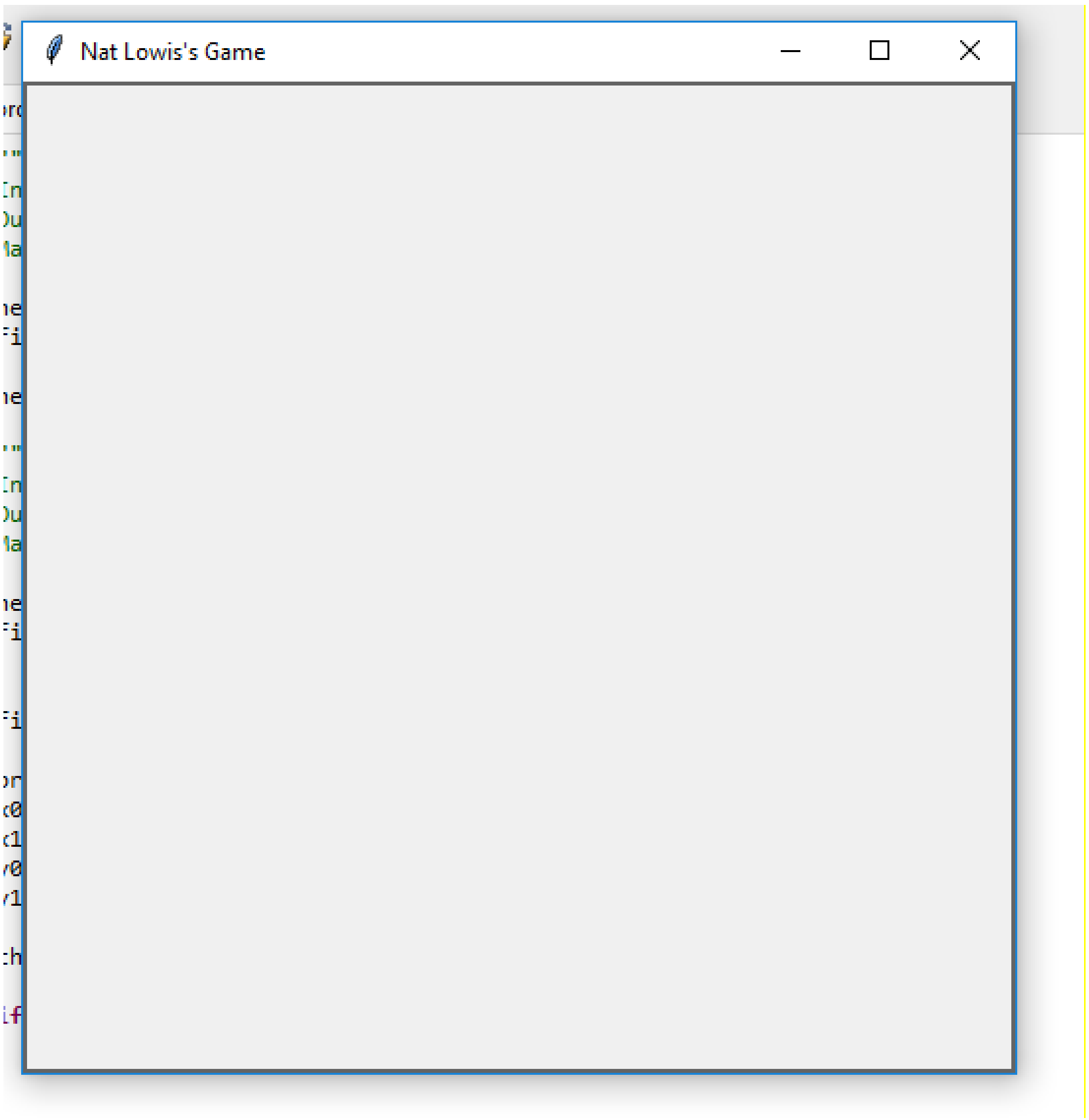
finishGame is not called here

Testing There was then an error which said:



```
>>> %Run first_prototype_final.py

  Traceback (most recent call last):
    File "C:\Users\User\Documents\Nat's Project\Code Doc\Prototype 1\first_prototype_final.py", line 128, in <module>
      finishGame()
    File "C:\Users\User\Documents\Nat's Project\Code Doc\Prototype 1\first_prototype_final.py", line 112, in finishGame
      characterPosistion = hero.coord()
    File "C:\Users\User\Documents\Nat's Project\Code Doc\Prototype 1\first_prototype_final.py", line 44, in coord
      coord = canvas.coords(self.characterOnScreen)
  NameError: name 'canvas' is not defined
```

I have forgotten to change the name for the canvas which meant it was not working. This got rid of the red text, but it still was not finishing and deleting everything on the screen. **Testing** I added a print("TRYING") and this showed me it was not being tried more than once. So, I decided to add it to moveHeroLeft and the moveHeroRight. It then worked (Almost). This was the screen I got when the user moved past the endpoint.



Not entirely the end result that I wanted but acceptable for my first prototype. This does show that the game ends when the user passes the endpoint. I will now finish all the comments and docstrings then I have completed my first prototype.

Review

I have now been able to make a screen and get the user to start interacting with the game by moving a character and ending the game at a certain point. Next I want to add bullets and an enemy. In the longer term, I want there to be a menu and a database.

Prototype 1 Code

```
from tkinter import *     #This means I can use Tkinter

class Character():

  """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
     This class will make the people on the screen and will allow the computer/user use the character"""
```

```
    def __init__(self, canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
          Inputs - The canvas, and positions to make the character
          Outputs - Character on the screen
          This will initialise the character meaning it can be used"""

        self.characterOnScreen = canvasCharacter.create_rectangle(x0Given, x0Given, x1Given, y1Given, fill= colour)
#Creates the character
        self.canvas = canvasCharacter  #Allows programmer to still use the screen
        x0 = x0Given   #These will give the different x and y points in case I need to use them again
        x1 = x1Given
        y0 = y0Given
        y1 = y1Given

    def move_Left(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
          Inputs - The class
          Outputs - Move the Character left
          This will move the character left by 10 pixels"""

        self.canvas.move(self.characterOnScreen, -10, 0)#Moves the character
        self.canvas.update()  #This will update the screen, so the user can see it

    def move_Right(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
          Inputs - The class
          Outputs - Move the Character Right
          This will move the character right by 10 pixels"""

        self.canvas.move(self.characterOnScreen, 10, 0)  #Moves the character
        self.canvas.update()  #This will update the screen so the user can see it

    def coord(self):

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 31/8/18
          Inputs  - The character
          Outputs - Where the character is
          This will show where the character is"""

        coordinates = self.canvas.coords(self.characterOnScreen)  #This gives the coordinates for the character
        return coordinates  #Returns it to the main program


def setUp():

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/8/18
      Inputs - None
      Outputs - A working screen with the floor and endpoint and sent to main program
      This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the user can make it full screen
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)  #Sets up the
    canvasSetUp.pack()  #Outputs screen
    floorSetUp = canvasSetUp.create_rectangle(500, 500, 0, 400, fill="blue")  #Sets up the floor
    endPointSetUp = canvasSetUp.create_rectangle(375, 375, 400, 400, fill="red")  #Sets up the endpoint

    characterHeroSetUp = Character(canvasSetUp, 50, 69, 50, 69, "Blue")  #This will make the user’s character

    return rootSetUp, canvasSetUp, floorSetUp, endPointSetUp, characterHeroSetUp  #Sends everything back to main
program.

root, frame, floor, endPoint, hero = setUp()

#def key(event):

 #   """Used as testing """

  #  print("pressed", repr(event.char))  TESTING

def callback(event):

    """Edited from: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm  Made by Nathaniel Lowis  1st Edit:
25/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left and right
    Means we can press on the screen and use the buttons"""

    frame.focus_set()   #This will mean we can use the left and right button as you have to press the window
    #print("clicked at", event.x, event.y)  Used as testing

def heroMoveLeft(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left
    Makes the hero move left and checks if the game has finished"""

    hero.move_Left()#Calls this method to move character
    finishGame()

def heroMoveRight(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move Right
    Makes the hero move Right and checks if the game has finished"""

    hero.move_Right() #Calls this method to move character##
    finishGame()


def finishGame():

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  Latest Edit: 31/8/18
      Inputs - Nothing
    Outputs - Checks if game has finished
    This will check if you have passed the ending point and if you have deletes the screen"""


   #print("TRYING")  Used for testing
   x0Endpoint = 375  #These are the known coordinates for the endpoint
   x1Endpoint = 400
   y0Endpoint = 375
   y1Endpoint = 400

   characterPosistion = hero.coord()  #Works out where the user's character is

   if characterPosistion[2] >= x0Endpoint:   #Checks if the character has passed the endpoint
      frame.delete(ALL)  #If they have it will delete all objects on the canvas




frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button pressed
frame.bind("<Button-1>", callback)  #Means user can press screen


frame.pack() #Sends it to the screen



frame.mainloop() #Infinte loop used
```

Second Prototype

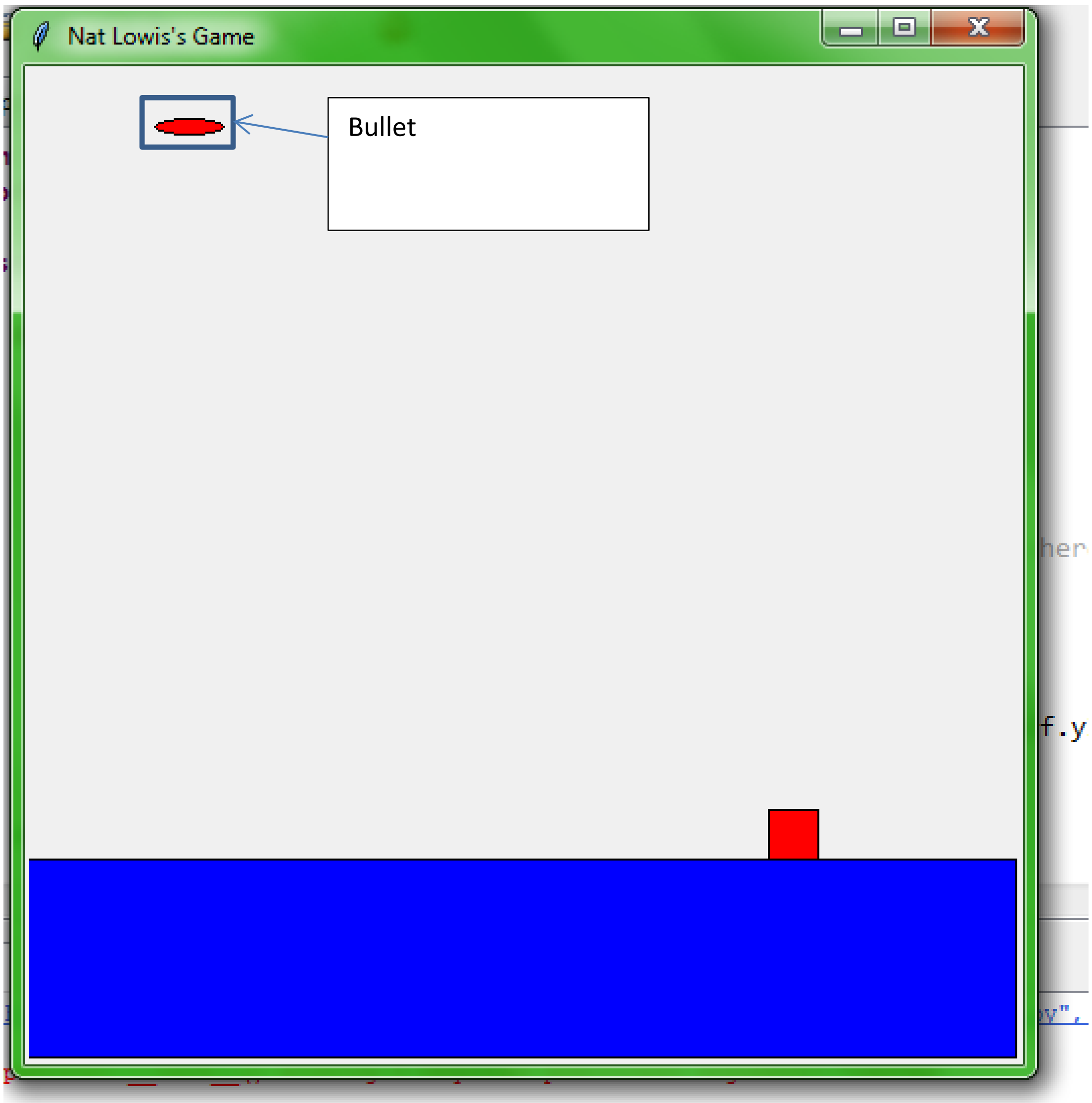
- Create bullet and villain

This prototype will focus on creating the bullets and also create an enemy which will give an element of danger to the game.

I have started creating the new bullet class and have started the initialisation class:

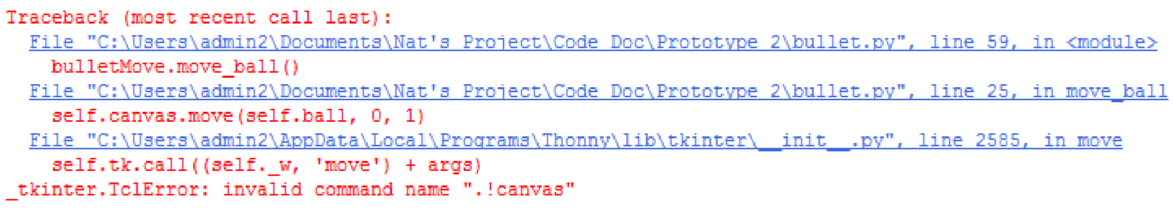
```
class bullet():

    def __init__(self, x1, y1, x2, y2, canvas):

        self.intialSpeed = 0
        self.finalSpeed = 0
        self.distance = 0
        self.horizontalAcceleration = 10
        self.x1 = x1                    #This 4 co-ordinates are where each corner should be placed
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2
        self.canvas = canvas
        self.ball = canvas.create_oval(self.x1, self.y1, self.x2, self.y2, fill="red")
```

Testing And it looks like this:



I have not made the bullet look right at the moment (This will be accomplished at a later point). However, it shows when a bullet is initialised it will create a bullet on the screen. I will now try and use code which I have found to make the bullet move properly[6].

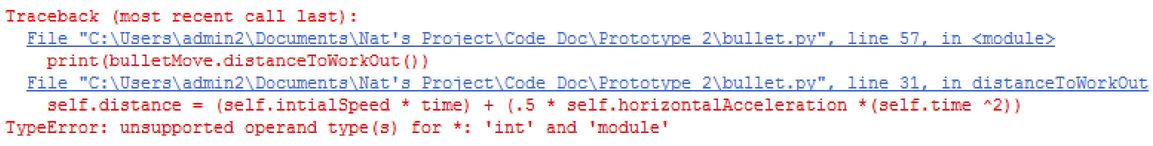
I changed the code I found making it move every 10 milliseconds thus meaning it can look smoother. I have programmed how far it will move every 10 milliseconds. **Testing** When the program runs however I get this error message:



```
Traceback (most recent call last):
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 2\bullet.py", line 59, in <module>
    bulletMove.move_ball()
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 2\bullet.py", line 25, in move_ball
    self.canvas.move(self.ball, 0, 1)
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 2585, in move
    self.tk.call((self._w, 'move') + args)
_tkinter.TclError: invalid command name ".!canvas"
```

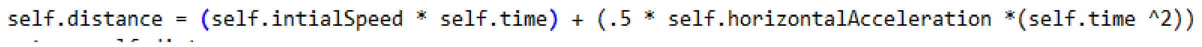
The bullet still moves but for some reason error messages were produced. The first way I tried to fix it was changing the name in the class which worked.

I am now trying to see if the program can work out the distance that the bullet should move which it is not doing at the moment. That was because I forgot some brackets. **Testing** I then could not multiply some numbers together as I had this error message:



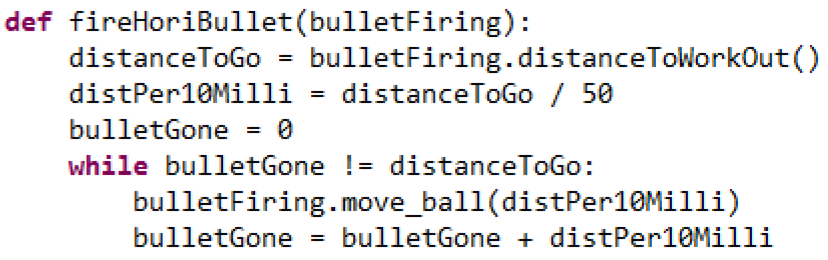
```
Traceback (most recent call last):
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 2\bullet.py", line 57, in <module>
    print(bulletMove.distanceToWorkOut())
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 2\bullet.py", line 31, in distanceToWorkOut
    self.distance = (self.intialSpeed * time) + (.5 * self.horizontalAcceleration *(self.time ^2))
TypeError: unsupported operand type(s) for *: 'int' and 'module'
```

I decided to use the debugger and I found the problem was that the time variable was not a number as I forgot to add self.time and had just written time. I then got a total of 35 which is incorrect unless I have done the maths incorrectly. **Testing** This was because I used in the line:



```
self.distance = (self.intialSpeed * self.time) + (.5 * self.horizontalAcceleration *(self.time ^2))
```

Instead of ^ I should have **. When I ran it this time I got the correct score of 125. I now am going to try and get it moving by the 125 'metres' now. This is the code at the moment:



```
def fireHoriBullet(bulletFiring):
    distanceToGo = bulletFiring.distanceToWorkOut()
    distPer10Milli = distanceToGo / 50
    bulletGone = 0
    while bulletGone != distanceToGo:
        bulletFiring.move_ball(distPer10Milli)
        bulletGone = bulletGone + distPer10Milli
```

[6] https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas In Bibliography

I am dividing the distance by 50 so when I am updating the screen it updates the bullet to move the correct distance every 10 milliseconds.

Testing However the bullet is not moving for 5 seconds. It is only moving for about 2 seconds so I will try and change it to move for 5 seconds. I have changed the .after statement to 100 which now means the bullet moves for about 5 seconds.

I am now going to add everything into the first prototype meaning the user can now start to fire the bullet. **Testing** I have a couple of problems however. One is this:

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1699, in __call__
    return self.func(*args)
TypeError: fireBullet() takes 0 positional arguments but 1 was given
```

To counter this, I have made a local variable which takes this unknown quantity. I will now finish writing the function. This worked.

Another good thing is that you can move both the character and shoot at basically the same time. It is a little unrealistic as the person is faster than the bullet however that is easy to change.

Next, I have to delete the bullet after it has finished and bring in a new character. I can make a bullet be intialised where the user is.

Testing I am trying to delete the bullet which is not working at the moment.

However, I have changed how fast the character can move and have moved the character downwards so that it can move on the floor.

I have now been able to delete the bullet. After attempting a variety of different ways, I have created a method which can delete the object on the screen and then delete the actual variable. This means after a set amount of time (5 seconds or so) the bullet will delete itself. Deleting the bullet is important so the user can see that the bullet has gone.

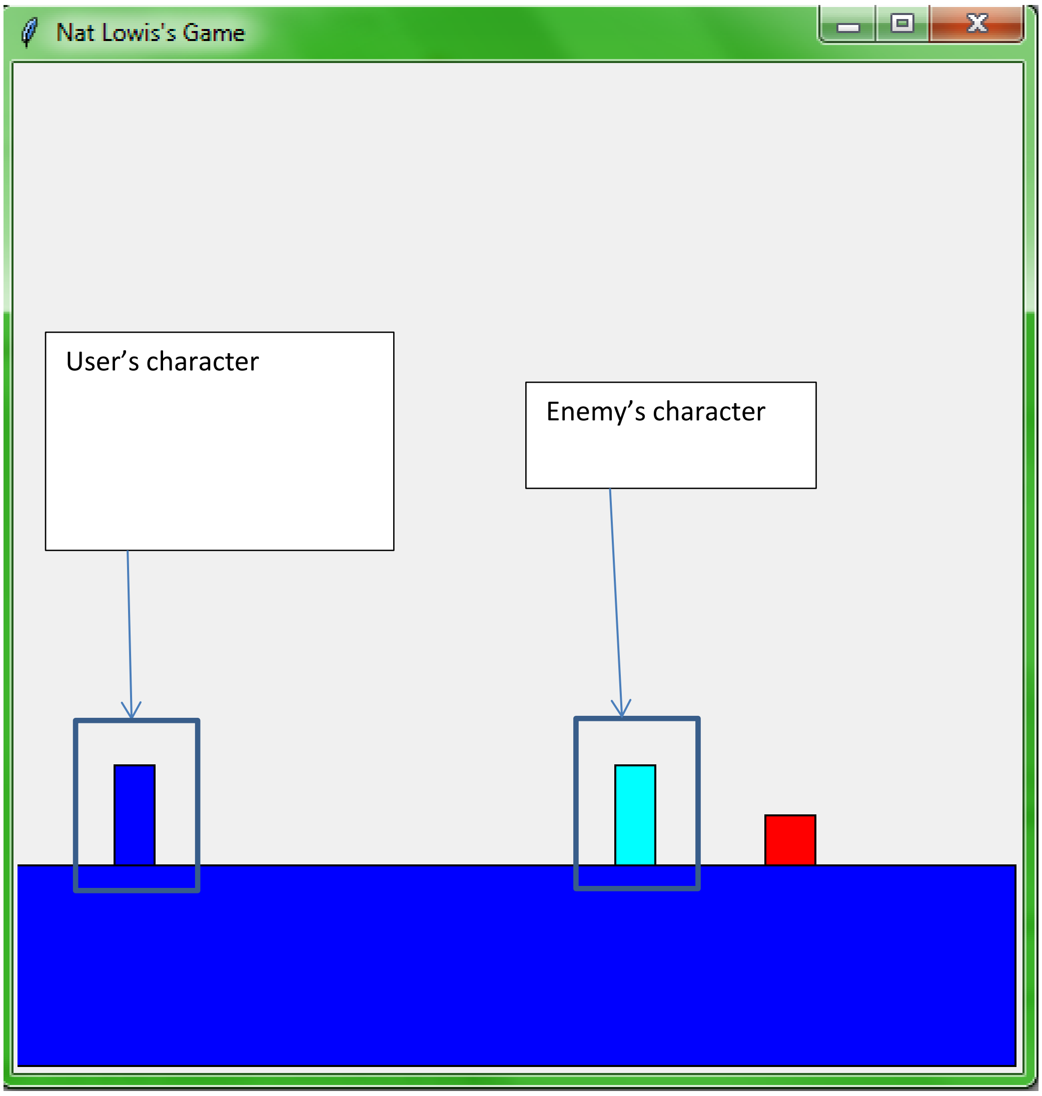
I will now create new attributes and methods to the class character and move the user's character down and create an enemy character.

Testing The user can now press the button q and the bullet will move the correct distance which is realistic

In my analysis, I mentioned that I was going to use threads throughout the project however I am not going to use threads as they are very complicated on Python. However, you can move the user's character whilst the bullet moves (a bit slower than usual but I do not mind) which solves a major problem which could have come later on.

I have chosen the mass to be about 8 grams as this is fairly accurate to real bullets.

I have now added both the user's character and the enemy to the screen and it looks like this:



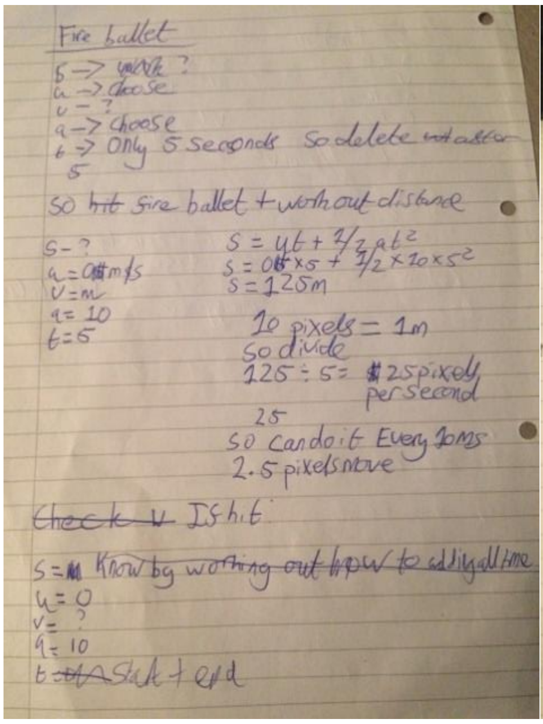
This also shows that I can initialise a character and get it to output to the screen

Now I will have the bullet be able to detect the enemy (which is the cyan object) and to take health away and also to delete that bullet which has been shot once it hits the character. At the moment I am having a few errors concerning spelling and not defining everything.

Testing Firstly, I ran the program to test with the numbers from my iterative programming part and all the numbers worked out perfectly

```
51.38093031466052
100000.0                    150.0
200.0
```

Testing I then ran it and the power in the bullet was only 1.9. I worked the maths out manually:



Fire bullet

s -> work ?
u -> choose
v - ?
a -> choose
t -> only 5 seconds So delete [illegible]
5

So ~~hit~~ fire bullet + work out distance

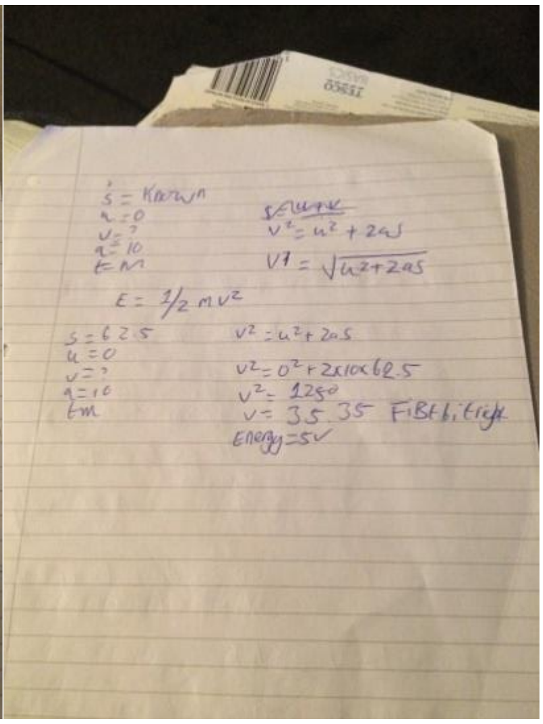
s - ?
u = ~~0~~ m/s
v = m
a = 10
t = 5

$s = ut + \frac{1}{2}at^2$
$s = 0 \times 5 + \frac{1}{2} \times 10 \times 5^2$
$s = 125m$

10 pixels = 1m
so divide
125 ÷ 5 = 25 pixels per second
25
so can do it Every 10ms
2.5 pixels move

~~Check v~~ Is hit

s = Know ~~by working out how to add [illegible] time~~
u = 0
v = ?
a = 10
t = ~~[illegible]~~ Start + end



s = Known
u = 0
v = ?
a = 10
t = M

[illegible]
$v^2 = u^2 + 2as$
$v = \sqrt{u^2+2as}$

$E = \frac{1}{2}mv^2$

s = 62.5
u = 0
v = ?
a = 10
t m

$v^2 = u^2 + 2as$
$v^2 = 0^2 + 2 \times 10 \times 62.5$
$v^2 = 1250$
v = 35.35 [illegible]
Energy = 5v

and it all worked out. So, to counteract this I will up the initial speed and acceleration. With a bit of research, I found the acceleration of bullet is 4.4x10^5 and the initial speed is 335 m/s so I will use these values. The only problem with these vaues is that the bullet is too fast! I will scale it down a little to make it easier to use the bullet. I now have the values both at 100 which makes it fast (It cannot multitask anymore) but easier to use.

Testing I will get the enemy to lose health and then this prototype is finished which I could do with the enemy losing 50 health as needed when it got hit by a bullet:

```
150
```

Testing I am doing this well, but I have got this error:

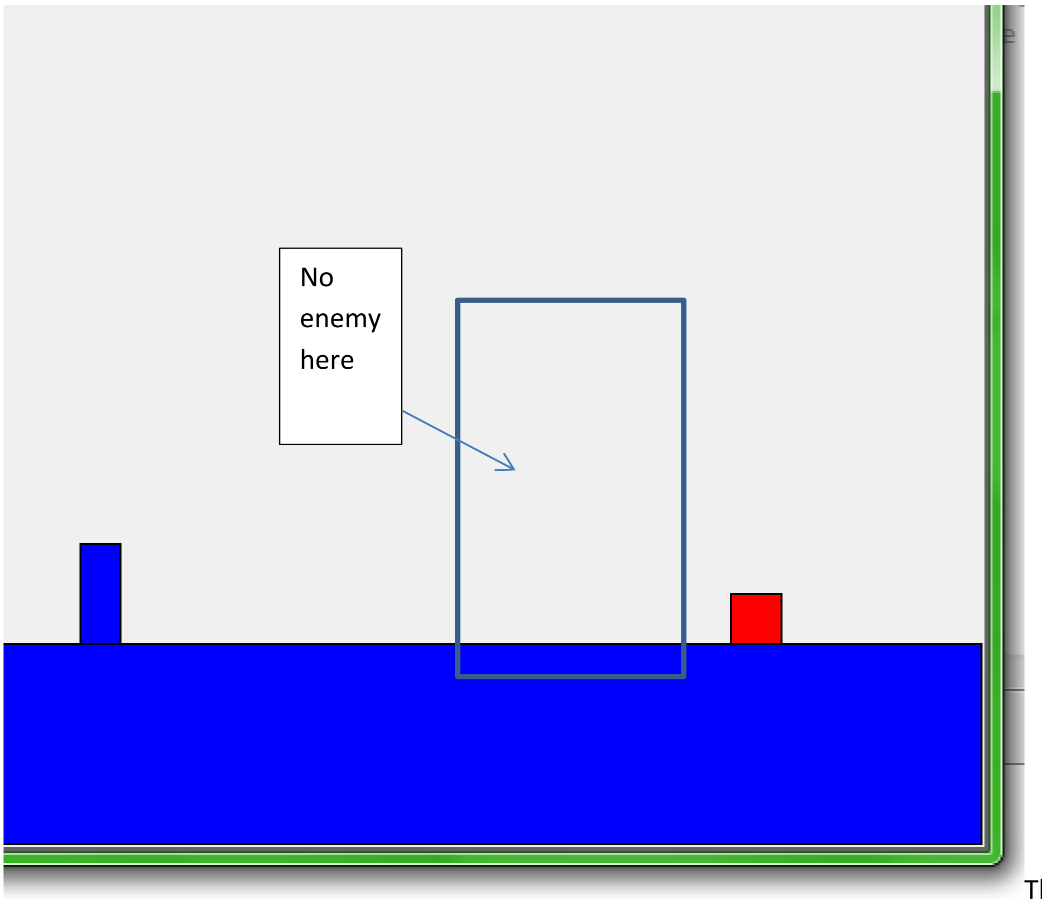
```
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 2\bullet_a
    enemyCoordinates = enemy.coord()
UnboundLocalError: local variable 'enemy' referenced before assignment
```

I started by commenting out all the code in the if statement to work out where the code is going wrong. I discovered that it was due to the lines I commented out and the bullet deleted itself when it was supposed to.

Testing When the user presses q after the enemy is defeated the game crashes as there is no enemy for it to check for, so I will use a try, except clause. The problem is that enemy is deleted meaning it does not have anything to use to see whether it has hit the enemy.

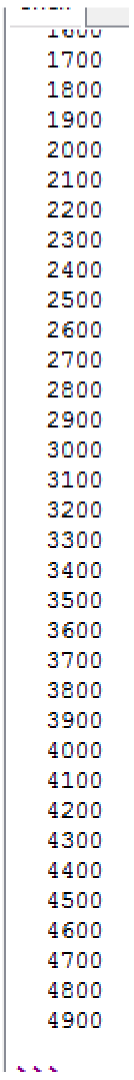
I will quickly add the code to allow the bullet to move on the screen and then stop and this was successful. I will comment the program and the second prototype will be done!

What the game looks like when the bullet hits a character:



The one character has been deleted when the bullet hit. This shows I could delete the character meaning the user can get to the end.

<u>Testing</u> When I was testing my game I saw that the bullet would be on the screen for longer than 5 seconds. To get around this I made a new variable called timeDone and every time the piece moved it added 100 to it until it reached 5000 and then it deleted itself.



```
1700
1800
1900
2000
2100
2200
2300
2400
2500
2600
2700
2800
2900
3000
3100
3200
3300
3400
3500
3600
3700
3800
3900
4000
4100
4200
4300
4400
4500
4600
4700
4800
4900
...
```

I also ran the code to check whether the bullet ran for 5 seconds and this was the output: As you can see the bullet ended when it reached 5000 which is 5 seconds.

Review

I have now got a bullet to be fired for about 5 seconds and to move in a realistic direction. I have had to give up on the idea of using threads to try and get all the pieces to work together. I will now want to make another bullet and get the enemy to start firing at the user.

Prototype 2 Code

```
from tkinter import *     #This means I can use Tkinter

import time   #Allows me to use time and measure how long bullets

class Bullet():

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 25/9/18
      This is the superclass which will allow me to model how the bullet moves and output it to the screen.  Parts of
code has been adapted from:https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas
      """

    def __init__(self, x1BulletGiven, y1BulletGiven, x2BulletGiven, y2BulletGiven, canvasToUse):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 25/9/18
          Inputs - the canvas, coordinates (Integers)
          Outputs - None
          Sets up the class when initialised"""

        self.intialSpeed = 100  #These will set up the intial speed, acceleration for use in the rest of the code
        self.finalSpeed = 0
        self.distance = 0
        self.horizontalAcceleration = 30
        self.x1Bullet = x1BulletGiven                    #These 4 co-ordinates are where each corner of the object should be
placed
        self.y1Bullet = y1BulletGiven
        self.x2Bullet = x2BulletGiven
        self.y2Bullet = y2BulletGiven
        self.canvasBullet = canvasToUse  #Sets up the code
        self.ball = canvasToUse.create_oval(self.x1Bullet, self.y1Bullet, self.x2Bullet, self.y2Bullet, fill="green")  #Outputs
to screen
        self.time = 5      #More constants made.
        self.mass = 0.008
        self.energy = 0

    def deleteBullet(self):   #Change to delete_Bullet

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - The class
          Outputs - Deletes the bullet from the screen
          Deletes the bullet from the screen"""

        self.canvasBullet.delete(self.ball)  #Deletes bullet


    def energy_In_Bullet(self):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Class
          Outputs - The energy of the bullet
          Works out the amount of energy in the bullet"""

        self.energy = .5 * self.mass * ( self.finalSpeed ** 2)   #Uses E = 1/2 mv^2
        #print(self.energy)    #testing

    def powerAtPoint(self, timeTaken):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Time, class
          Outputs - The amount of power from the bullet
          Works out the amount of power in the bullet"""

        power = self.energy / timeTaken  #Uses p = E/t

        #print(power)   #Testing

        return power

    def coord(self):  #CHANGE TO COORD_BULLET

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 12/9/18
          Inputs  - The class
          Outputs - Where the bullet is
          This will show where the bullet is"""

        coordinatesBullet = self.canvasBullet.coords(self.ball)  #This gives the coordinates for the bullet
        return coordinatesBullet  #Returns it to the main program



class HoriBullet(Bullet):

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      This is a subclass of bullet which calculates and output it to the screen."""

    def speedAtAnyPoint(self, distanceGone):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs -  Distance Gone (Float), class
          Outputs - The final speed of bullet
          This works out the final speed of the bullet"""

        self.finalSpeed = ((self.intialSpeed ** 2) + (2 * self.horizontalAcceleration * distanceGone) )** .5   #Uses V^2 =
u^2 + (2as)
        #print(self.FinalSpeed)   #Testing

        return self.finalSpeed

    def move_ball(self, xMovement):   #CHANGE TO MOVE_BULLET_HORI

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Input - How much to move by
          Output - Moving on the screen"""

        self.canvasBullet.move(self.ball, xMovement, 0)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen

    def distanceToWorkOut(self):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - class
          Outputs - The distance to go"""

        self.distance = (self.intialSpeed * self.time) + (.5 * self.horizontalAcceleration *(self.time **2)) #Uses S = ut +
1/2at^2
        return self.distance



class Character():

    """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 25/9/18
      This class will make the people on the screen and will allow the computer/user use the character"""

    def __init__(self, canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour):
        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
           Inputs - The canvas, and positions to make the character
           Outputs - Character on the screen
           This will initialise the character meaning it can be used"""

        self.characterOnScreen = canvasCharacter.create_rectangle(x0Given, y0Given, x1Given, y1Given, fill= colour)
#Creates the character
        self.canvas = canvasCharacter  #Allows programmer to still use the screen
        x0 = x0Given   #These will give the different x and y points incase I need to use them again
        x1 = x1Given
        y0 = y0Given
        y1 = y1Given
        self.health = 200



    def deleteCharacter(self):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
           Input - Character Class
           Outputs - Deletes the character
           Deletes character if they are killed"""

        self.canvas.delete(self.characterOnScreen)  #Deletes the character

    def lose_Health(self, healthToLose):

        """Made by Nathaniel Lowis 1st Edit: 11/9/18  Latest Edit: 12/9/18
           Inputs - The health to lose (Float)
           Outputs - Health left (Float)
           Makes the character lose health"""

        self.health = self.health - healthToLose  #Lose health
        return self.health



    def move_Left(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 11/9/18
           Inputs - The class
           Outputs - Move the Character left
           This will move the character left by 2 pixels"""

        self.canvas.move(self.characterOnScreen, -2, 0)#Moves the character
        self.canvas.update()  #This will update the screen, so the user can see it

    def move_Right(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 11/9/18
           Inputs - The class
           Outputs - Move the Character Right
           This will move the character right by 2 pixels"""

        self.canvas.move(self.characterOnScreen, 2, 0)  #Moves the character
        self.canvas.update()  #This will update the screen, so the user can see it

    def coord(self):   #coord_Player CHANGE TO THIS

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 31/8/18
          Inputs  - The character
          Outputs - Where the character is
          This will show where the character is"""

        coordinates = self.canvas.coords(self.characterOnScreen)  #This gives the coordinates for the character
        return coordinates  #Returns it to the main program

    def fireHoriBulletClass(self, coordinateInFireBulletHori):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - Coordinates of character in an array
          Outputs - Shooting bullet
          This controls the bullet"""


        bulletShootingHori = HoriBullet(coordinateInFireBulletHori[2], coordinateInFireBulletHori[1],
coordinateInFireBulletHori[2] + 5, coordinateInFireBulletHori[1]+5, self.canvas)  #Makes an instant of the HoriBullet
Class
        fireHoriBullet(bulletShootingHori)   #Controls the bullet
        bulletShootingHori.deleteBullet()  #These will delete the bullet afterwards
        del(bulletShootingHori)


def setUp():     #CHANGE TO SET_UP

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/9/18
      Inputs - None
      Outputs - A working screen with the floor and endpoint and sent to main program
      This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the user can make it full screen
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)  #Sets up the
    canvasSetUp.pack()  #Outputs screen
    floorSetUp = canvasSetUp.create_rectangle(500, 500, 0, 400, fill="grey")  #Sets up the floor
    endPointSetUp = canvasSetUp.create_rectangle(375, 375, 400, 400, fill="red")  #Sets up the endpoint
    heroSetUp = Character(canvasSetUp, 50, 70, 350, 400, "Blue")  #This will make the user's character
    enemySetUp = Character(canvasSetUp, 300, 320, 350 ,400, "Cyan")  #This will make the enemy class



    return rootSetUp, canvasSetUp, floorSetUp, endPointSetUp, heroSetUp, enemySetUp #Sends everything back to
main program.

def fireHoriBullet(bulletFiring):  #CHANGE TO FIRE_HORI_BULLET

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      Inputs - The Class HoriBullet
      Outputs - The bullet moving and interacting with the environment
      This will allow the bullet to interact with the environment"""

   timer1Start = time.time()    #Starts a timer
   distanceToGo = bulletFiring.distanceToWorkOut()  #Works out how far the bullet goes
   distPer10Milli = distanceToGo / 50   #Divides the distance by 50 so they are all in equal chunks
   bulletGone = 0  #Creates a variable which works out how far the bullet has gone
   timeDoneHoriBullet = 0

   while bulletGone != distanceToGo and timeDoneHoriBullet < 5000:   #Whilst the bullet has not gone as far as it
needs to

      enemyCoordinates = enemy.coord()  #Gets enemy's coordinates in an array
      bulletCoordinates = bulletFiring.coord()  #Gets the bullets coordinates in an array

      try:  #The program will go down this route when there is an enemy class

         if bulletCoordinates[2] > enemyCoordinates[2]:  #If the bullet is past the enemy
            timer1Finish = time.time()  #Stops timer
            timer1 = timer1Finish - timer1Start   #Works out length of time the  bullet has gone for

            #print(bulletGone) #testing
            #print(timer1)    #testing

            bulletFiring.speedAtAnyPoint(bulletGone)   #Works out the speed of the bullet
            bulletFiring.energy_In_Bullet()    #Works out the energy of the bullet
            bulletPower = bulletFiring.powerAtPoint(timer1)   #Works out the power of the bullet

            #print(bulletPower) #testing

            bulletGone = distanceToGo    #Means the while loop can stop
            enemyHealth = enemy.lose_Health(bulletPower)  #Makes the enemy lose health

            if enemyHealth <= 0:  #If the enemy's health is below or equal to 0
               #CULD PUT THIS IN OWN FUNCTION
              enemy.deleteCharacter()  #Deletes the enemy


         else:

            bulletFiring.move_ball(distPer10Milli)   #Moves the ball the amount it needs to
            bulletGone = bulletGone + distPer10Milli  #Added to  bulletGone
            timeDoneHoriBullet = timeDoneHoriBullet + 100

      except:  #This is when there is no enemy

         bulletFiring.move_ball(distPer10Milli)   #Moves the bullet
         bulletGone = bulletGone + distPer10Milli #Added to BulletGone
         timeDoneHoriBullet = timeDoneHoriBullet + 100

#def key(event):

 #   """Used as testing """

  #  print("pressed", repr(event.char))  TESTING
```

```
def callback(event):

    """Edited from: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm  Made by Nathaniel Lowis  1st Edit: 
25/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left and right
    Means we can press on the screen and use the buttons"""

    frame.focus_set()   #This will mean we can use the left and right button as you have to press the window
    #print("clicked at", event.x, event.y)  Used as testing

def heroMoveLeft(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move left
    Makes the hero move left and checks if the game has finished"""

    hero.move_Left()#Calls this method to move character
    finishGame()

def heroMoveRight(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 31/8/18
    Inputs - What the user did
    Output- Allows user to move Right
    Makes the hero move Right and checks if the game has finished"""

    hero.move_Right() #Calls this method to move character##
    finishGame()



def finishGame():

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  Latest Edit: 31/8/18
      Inputs - Nothing
      Outputs - Checks if game has finished
      This will check if you have passed the ending point and if you have deletes the screen"""


    #print("TRYING")  Used for testing
    x0Endpoint = 375  #These are the known coordinates for the endpoint
    x1Endpoint = 400
    y0Endpoint = 375
    y1Endpoint = 400

    characterPosistion = hero.coord()  #Works out where the user's character is

    if characterPosistion[2] >= x0Endpoint:   #Checks if the character has passed the endpoint
        frame.delete(ALL)  #If they have it will delete all objects on the canvas


def fireBullet(letter):
    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 11/9/18
       Inputs - The q which must (for some reason) be included
       Outputs - None
       This will start the process of firing a bullet"""

    #MAYBE GET BOTH TO GO HERE AND USE IF STATMENT TO GO EITHER WAY

    characterPosistion = hero.coord() #Gets the coordinates for the user's character
    hero.fireHoriBulletClass(characterPosistion)  #Fires bullet



root, frame, floor, endPoint, hero, enemy= setUp()  #Sets up game





frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button pressed

frame.bind("<Button-1>", callback)  #Means user can press screen
frame.bind("<q>", fireBullet)




frame.pack() #Sends it to the screen



frame.mainloop() #Infinte loop used
```

Third Prototype

– 2nd Bullet made and villain shoots

Firstly, in this prototype I am going to add a jump (as all games need a jump!). This will then mean the user can jump over the bullets which the enemy shoots. This is going to be simplistic as all it will do is jump up to a certain height then go back down. I am making it simplistic now as it is not an integral part of the program. However, I do not want the user to be able to stop mid jump and move higher than you should.

Testing When I first did it was moving left then right and then when I changed it to the vertical component it moved down then up which was quickly changed to move up then down.

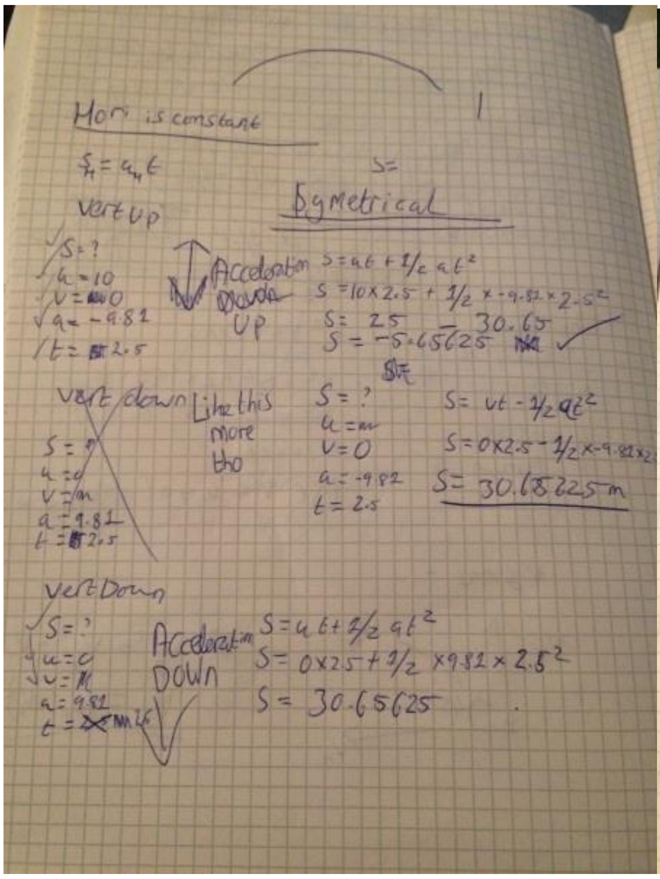
Testing At the moment you can get the character to keep on jumping higher and higher if you spam the up button. To counteract this, I will have a minimum height you can jump from. This works however when you shoot a bullet you stop jumping when you shoot. This would take too long to try and sort out but I will sort out that the enemy will need both x and y coordinates to be defeated and the endpoint will require both x and y coordinates instead of just an x component.

The code for ending the game now:

```
if characterPosistion[2] >= x0Endpoint and characterPosistion[3] >= y0Endpoint and characterPosistion[3] <= y1Endpoint and characterPosistion[2] <= x1Endpoint:
    frame.delete(ALL)  #If they have it will delete all objects on the canvas
```

I will now try and get the parabolic bullet working. To do this I will take all the bullet code and copy and paste it into another Python file. Then I will start working to build the parabolic bullet.

Firstly, I did the maths on paper for the distance and the speed:



Hori is constant

$s_h = u_h t$

vert up

$s = ?$
$u = 10$
$v = 0$
$a = -9.81$
$t = 2.5$

Symetrical

Acceleration Down Up

$s = ut + 1/2 at^2$
$s = 10 \times 2.5 + 1/2 \times -9.82 \times 2.5^2$
$s = 25 - 30.65$
$s = -5.65625$

vert down

$s = ?$
$u = 0$
$v = m$
$a = 9.81$
$t = 2.5$

Like this more tho

$s = ?$
$u = m$
$v = 0$
$a = -9.82$
$t = 2.5$

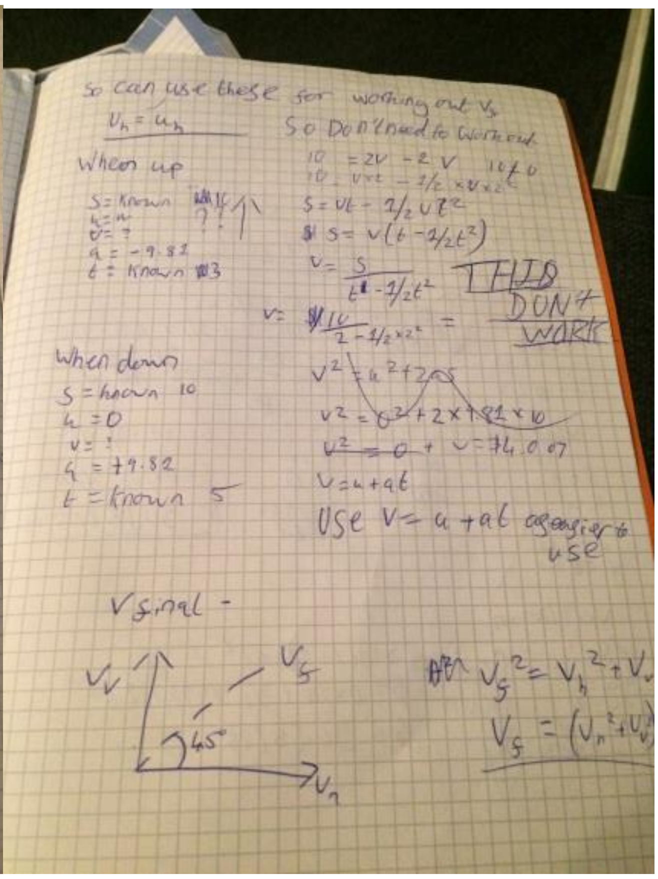
$s = vt - 1/2 at^2$
$s = 0 \times 2.5 - 1/2 \times -9.82 \times 2.5^2$
$s = 30.68225$ m

vert Down

$s = ?$
$u = 0$
$v = m$
$a = 9.82$
$t = 2.5$

Acceleration DOWN

$s = ut + 1/2 at^2$
$s = 0 \times 2.5 + 1/2 \times 9.82 \times 2.5^2$
$s = 30.65625$



So can use these for working out v_f

$v_h = u_h$

So Don't need to work out

When up

$s =$ known 10
$u = m$
$v = ?$
$a = -9.81$
$t =$ known 3

$s = vt - 1/2 vt^2$
$s = v(t - 1/2 t^2)$
$v = \frac{s}{t - 1/2 t^2}$
$v = \frac{10}{2 - 1/2 \times 2^2} =$

THIS DON'T WORK

When down

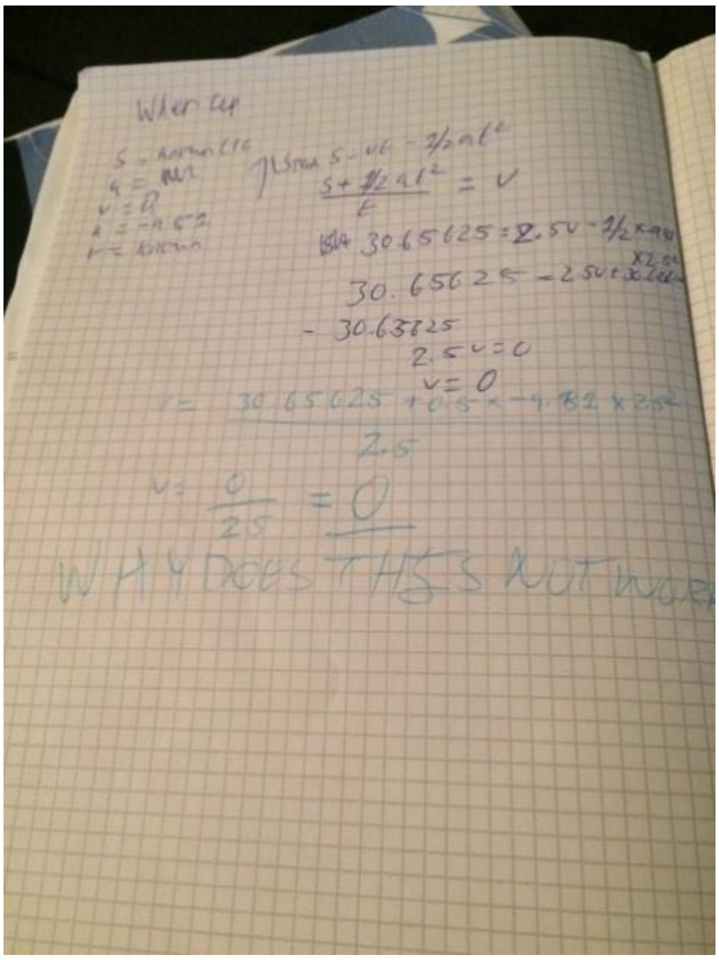
$s =$ known 10
$u = 0$
$v = ?$
$a = +9.82$
$t =$ known 5

$v^2 = u^2 + 2as$
$v^2 = 0^2 + 2 \times 9.81 \times 10$
~~$v^2 = 0$~~ + $v = 14.007$
$v = u + at$

Use $v = u + at$ as easier to use

v final -

$v_f^2 = v_h^2 + v_v^2$
$v_f = (v_h^2 + v_v^2)$



When up

s = [illegible]
u = [illegible]
v = 0
a = -9.82
t = known

[illegible] $s = ut - \frac{1}{2}at^2$

$\frac{s + \frac{1}{2}at^2}{t} = v$

$30.65625 = 2.5v - \frac{1}{2} \times 9.81 \times 2.5^2$

$30.65625 = 2.5v + 30.6[illegible]$

$- 30.63825$

$2.5v = 0$

$v = 0$

$v = \frac{30.65625 + 0.5 \times -9.82 \times 2.5^2}{2.5}$

$v = \frac{0}{25} = \underline{0}$

WHY DOES THIS NOT WORK

. **Testing** I then tried to see if the distance for when the bullet is moving up is correct and it wasn't. The vertical component was right, but the horizontal was 102.5 when it should be 250. This was because I added the numbers:

```
def distance_Worker_Up(self):

    self.distanceVertical = (-.5 * self.verticalAccelerationUp * (self.time /2)**2)
    self.distanceHori = (self.intialSpeedHoriPara + (self.time /2))
    print(self.distanceVertical, self.distanceHori)
    return self.distanceVertical, self.distanceHori
```

Should be a *

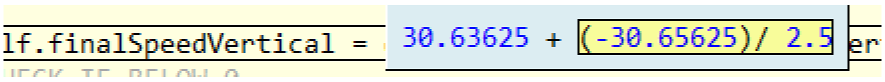
When changed to a * the answer worked. I then wrote the function for going down. **Testing** When I ran the code the horizontal distance was correct but for the vertical distance it was 7.755485961562951. This is wrong. When I used the debugger, the maths seemed to work fine so I went back to how I did it. The problem was that I was square rooting the time instead of squaring it: `(self.time /2)**.5)`. It should be **2. Once I did this the maths worked so it now outputs the right vertical component. I then decided to write the code for the speed at different points of the program. When I ran the program however it did not work as I got an error by dividing by 0. This is because of how I used the equation so when I rewrote the equation I got the correct answer (on paper).

```
self.finalSpeedVertical = distanceVertUp / (timeTakenSoFar - .5 * (timeTakenSoFar **2))
```

Now becomes:

```
self.finalSpeedVertical = distanceVertUp + (.5 * self.verticalAccelerationUp * (timeTakenSoFar ** 2))/ timeTakenSoFar
```

Testing This gave me an answer of 18.37375 which does not equal 0! I then tried it on paper again and it worked. So, I decided to step through the code. The problem is here:



```
lf.finalSpeedVertical = 30.63625 + (-30.65625)/ 2.5 er
```

It is dividing too early and I will now use more brackets to make sure it works.

Testing When I used more brackets it then gave me the answer of -0.007999999999999983 which still does not equal 0! The problem is rounding as 30.65625 – 30.65625 = -0.019999999999999574. To counteract this, I will say any speed below 0 is actually equal to 0 as there cannot be a negative speed. This worked. So, I have the speed working for it going vertically upwards and now I will work it out for vertically downwards.

Testing This is just acceleration multiplied by time so when time is 2.5 speed should be 24.525 and when I ran the code the answer was 24.525000000000002 which is close enough. The problem as you can see is the rounding errors however if the error is about 10 decimal places below this will barely affect the physics or the code. The program will be unrealistic if there is a negative speed.

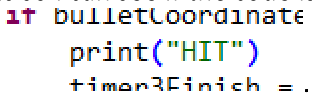
Finally, I will use both the vertical and horizontal components to then work out the final velocity using Pythagoras Theorem.

Testing Using my testing (VVertical = 100, hori = 50 or vice versa) the answer which the program gave was 111.80339887498948 which is correct!

Next step is to get the bullet moving which requires putting it in the main program. After a lot of working out with names of variables, I have the bullet going up! The problem is that pressing both q and w will shoot the parabolic bullet up not just w. This wil be the next step after that I will consider programming the bullet to go down. I just took out the code to fire a bullet from the bulletFiring module and put it in its own one, so you can now press q and a straight-line bullet moves and press w and a parabolic bullet fire.
Testing When I shot the bullet it would not notice if I hit the enemy because I had written it down as bulletCoordinates2 when instead it should have been bulletCoordinates. I changed everything to be the same so it should have worked. But it was unsuccessful. I will finish sorting out the parabolic arc by outputting it to the screen then finish programming how the bullet should be hitting the enemy.

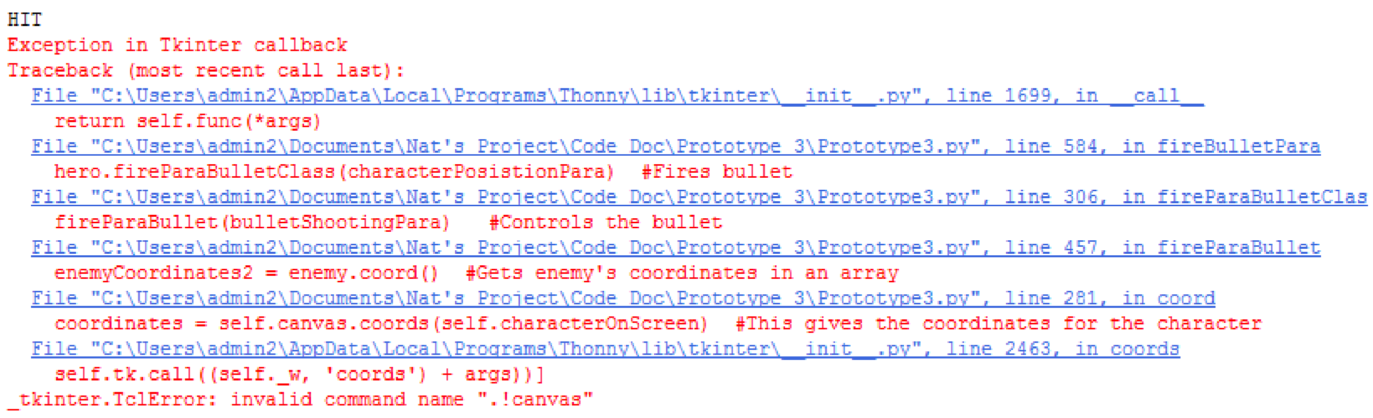
Testing First, I copied and pasted the work I did for moving the bullet up and after not naming the variables and methods correctly (At one point the bullet moved downward instead of upward!) the bullet moved in a parabolic arc. I now have two bullets which move in a way which physics suggest. I can change the x and y values to try and get the bullet to hit the enemy but that it moved is brilliant. It is pleasing that the bullet is moving but now I will need to change the x and y values to get the bullet to hit the enemy.

Next, I will try and get the bullet to hit the enemy and this will mean that the whole game works!

Testing I added a print statement which says hit so hopefully when I get the bullet to hit the character I can see whether it hits. This is so I can see if the code is registering a hit. When I ran the code and got the bullet to move it did hit the character: 

```
if bulletCoordinate
    print("HIT")
    timer3Finish = 
```

. I also got a pile of red text:



```
HIT
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1699, in __call__
    return self.func(*args)
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 3\Prototype3.py", line 584, in fireBulletPara
    hero.fireParaBulletClass(characterPosistionPara)  #Fires bullet
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 3\Prototype3.py", line 306, in fireParaBulletClas
    fireParaBullet(bulletShootingPara)  #Controls the bullet
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 3\Prototype3.py", line 457, in fireParaBullet
    enemyCoordinates2 = enemy.coord()  #Gets enemy's coordinates in an array
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 3\Prototype3.py", line 281, in coord
    coordinates = self.canvas.coords(self.characterOnScreen)  #This gives the coordinates for the character
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 2463, in coords
    self.tk.call((self._w, 'coords') + args))]
_tkinter.TclError: invalid command name ".!canvas"
```

I am not sure what this red text means however I will put a print statement to say what the enemy's health was at.

It does not reach that far however.   It does not even go past the code which asks it to get the time:

```
if bulletCoordinates3[0] > enemyCoordinates2[0] and bullet
    print("HIT")
    timer3Finish = time.time()  #Stops timer
    print(timer3)
    timer3 = timer3Finish - timer3Start   #Works out lengt
```

This is how I am testing the code by adding Hit statements in it.

Testing However, I am just looking at one aspect of the arc which is when the bullet goes down. I will look at going up as well. It printed the time (Here being 0.0 seconds). One of the reasons it might not work was that I had not changed the timer1 to timer2. When I next ran the code, it seemed to work…. But it still does not work consistently.

Testing To try and work out how to do this I firstly got the program to output HIT UP or HIT DOWN depending when the program noticed that we have 'hit' the enemy. When I ran the code next, I was able to get the character to hit the enemy and it was able to output the time.

```
HIT UP
HIT UP 0.0
```

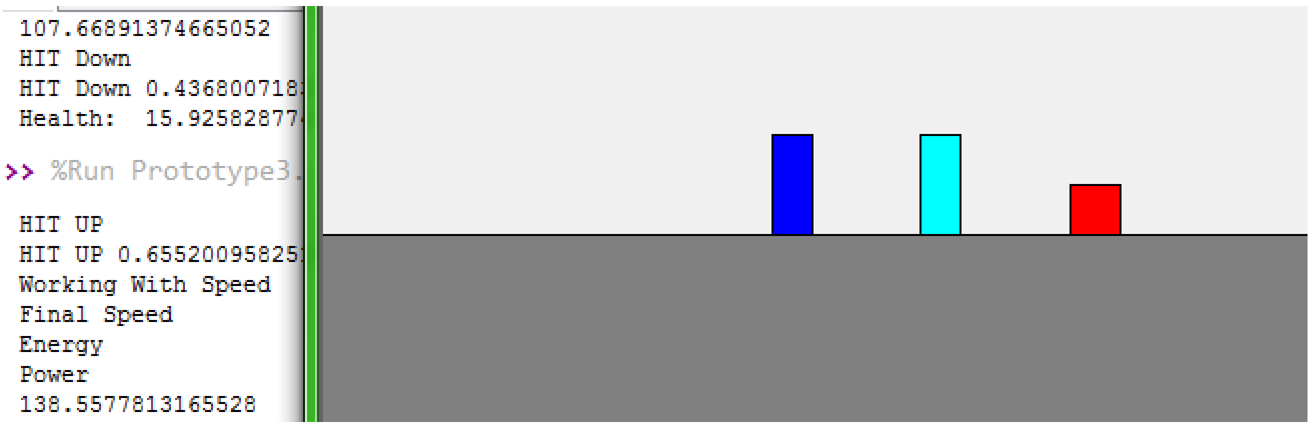
I then got the code to output what it has done:

```
bulletFiring2.speed_Vert_Up(bulletGoneUp, timer2)   #Works out the speed of the bullet
print("Working With Speed")

bulletFiring2.final_Speed_Parabolic()
print("Final Speed")
bulletFiring2.energy_In_Bullet()    #Works out the energy of the bullet
print("Energy")
bulletPowerPara = bulletFiring2.powerAtPoint(timer2) #Works out the power of the bullet
print("Power")

#print(bulletPower) #testing
```

Testing I did change the conditions of the code, so all the bullet had to do was go past the enemy and not go through it. I ran through the code and this was outputted:

```
HIT UP
HIT UP 0.4368007183074951
Working With Speed
Final Speed
Energy
Power
107.66891374665052
HIT Down
HIT Down 0.4368007183074951
Health:  15.925828774002483
```

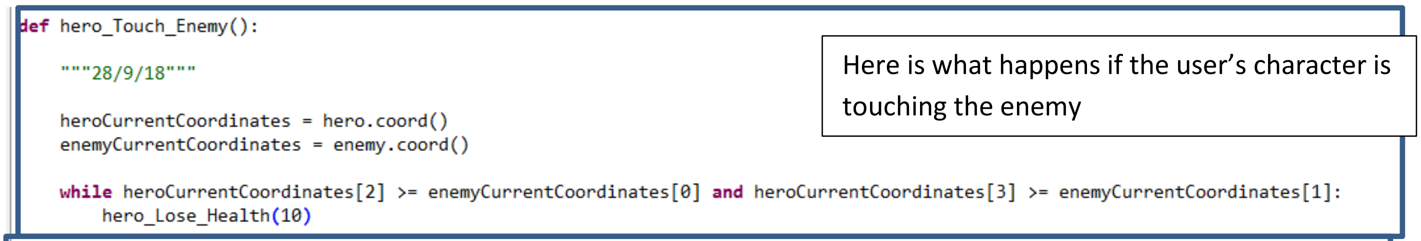
As you can see there is both health taken away from going up and down and this can be solved by adding an if statement saying whether it needs to go down. This worked.



```
107.66891374665052
HIT Down
HIT Down 0.436800718
Health:  15.92582877

>> %Run Prototype3.

HIT UP
HIT UP 0.65520095825
Working With Speed
Final Speed
Energy
Power
138.5577813165528
```

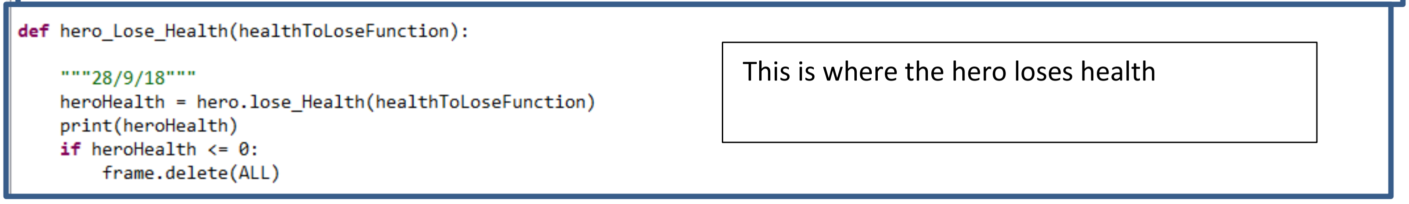
You can see there is no bullet here and this is all the code which has been outputted! So, to make this code work a bit better I will have the hitbox (the area which the user shoots at which the program will take as hitting the enemy) to be about 5 to 10 pixels higher to make it a bit easier for the user to shoot. However, at the moment you can have the bullet miles from the enemy, but it still hits the character as the hitbox is all the area behind the enemy. To change this, I will have in the while statement, code specifying that it has to be the area in which the enemy resides. I now have two bullets which the user can shoot which harms the enemy and means the user can end the game.

I now have to make the enemy dangerous so if the user's character hits the enemy, the 'hero' loses health and letting the enemy to shoot bullets. I will get the hero to lose health when touching the enemy first. This is the code at the moment:



```
def hero_Touch_Enemy():

    """28/9/18"""

    heroCurrentCoordinates = hero.coord()
    enemyCurrentCoordinates = enemy.coord()

    while heroCurrentCoordinates[2] >= enemyCurrentCoordinates[0] and heroCurrentCoordinates[3] >= enemyCurrentCoordinates[1]:
        hero_Lose_Health(10)
```

Here is what happens if the user's character is touching the enemy



```
def hero_Lose_Health(healthToLoseFunction):

    """28/9/18"""
    heroHealth = hero.lose_Health(healthToLoseFunction)
    print(heroHealth)
    if heroHealth <= 0:
        frame.delete(ALL)
```

This is where the hero loses health

If the hero loses all his health the game will end making it harder for the user. I have not programmed in any x values for the enemy's hitbox beyond the front (So does not wrap around the actual character) as it is almost impossible to jump over the enemy.

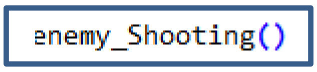
<u>Testing</u> At the moment, the output shows that the character is not being deleted if it loses all of it's health and it is in an infinite loop. This is a loop which you cannot get out of. To change this, I am going to change the while loop to an if statement and increase the amount you should lose to 50. This works so that if you were touching the enemy you lose health. Now it is a bit unfair that you lose so much health as the user is not being pushed back, so what I will do is make a pushback by 10 pixels, so you are out of the way. I could increase this if I want to, but this shows that it works. Now if you try and jump over the enemy you are going to find it very hard, so I will not program this. Finally, if the enemy is deleted you get an error as the code is expecting an enemy to touch. A simple try except clause will solve this problem. Now the program will not allow you to hit the enemy infinitely!

The next thing I need to try and do is get the enemy to shoot the character. I could have the enemy to alternate which bullet it shoots but this would be too complex. I will get the enemy to shoot a bullet every 5 seconds (So there is a bullet on the screen every 5 seconds). It will slow the game even more, but this is a trade-off which I accept will need to happen. However, before I did this I commented out all the print statements I have used to help test the bullets. Now I will try and get the enemy to shoot a bullet.

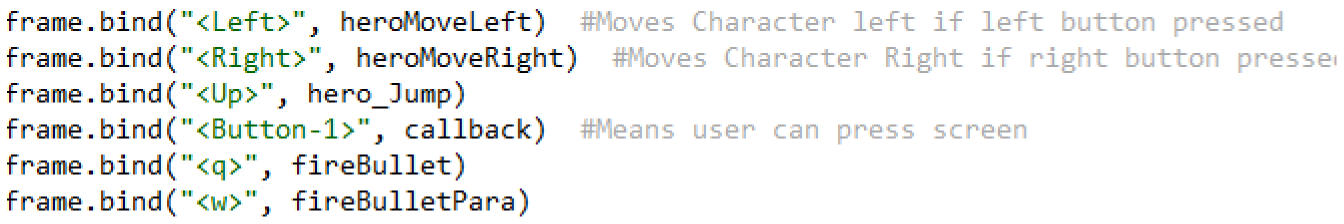
I will put in a module which has an infinite loop (which will be needed to keep on shooting a bullet) and see how the program can deal with it. It did not like it when I had it before all the movement buttons.

When I put it after the movement buttons it did not like that at all! What I will do is try and see what happens if I put it in the callback function which is when the user presses the screen to get the character to move. When it's here the game dies. So, I need to think of how to get the enemy to fire the bullet.

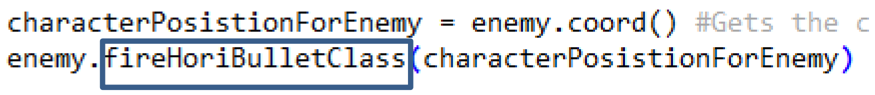
What I decided to do was actually write the code to get the bullet to be shot. I copied and pasted the code for a horizontal bullet and added an Ene at the end of each variable so they could all work. Now what I have to do is get the bullet to shoot. I added it to the main code (To get the bullet to shoot):


```
enemy_Shooting()
```

Calls the way to shoot a bullet

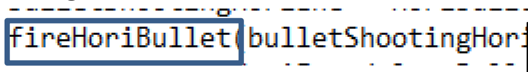

```
frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button presse
frame.bind("<Up>", hero_Jump)
frame.bind("<Button-1>", callback)  #Means user can press screen
frame.bind("<q>", fireBullet)
frame.bind("<w>", fireBulletPara)
```

<u>Testing</u> This did not do anything, so I put it in fireBullet. But when I did that the bullet was going the wrong way and was hurting the enemy! The problem was that I was not calling the right function:


```
characterPosistionForEnemy = enemy.coord() #Gets the c
enemy.fireHoriBulletClass(characterPosistionForEnemy)
```

This should be fireHoriBulletClassEnemy

Testing This did not work either! I put the bullet to -5 which could work however the bullet still was going the wrong way! So, what I decided to do was go and see if I could see what was wrong later on in the code. I found the problem. Another spelling error:



```
fireHoriBullet(bulletShootingHor
```

Should be fireHoriBulletEne

The bullet is still going the wrong way! I will now check the fireHoriBulletEne module. This worked. Next, I will get the bullet to hit the hero and the hero to be able to jump over the bullet. The firing of the enemy's bullet needs to be automatic.

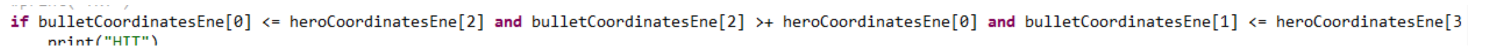
Thinking through the problem I might keep it that when you fire a horizontal bullet the enemy will fire their bullet first. One of the reasons for this is that during my testing of the enemy's bullet I have used the firing of the horizontal bullet as a prompt to also fire the enemy's bullet. I believe this will be an effective method in the actual game, because using the horizontal bullet is too powerful as it only requires using this bullet 2 or 3 times to kill the enemy. Therefore, the user will require some skill to be able to jump over the enemy's bullet before they can fire their own horizontal bullet or use the parabolic bullet to hit the enemy. Consequently, I do not have to program the code to fire a bullet every 5 seconds making it easier for me to complete the project on time.

Testing So I started changing the enemy's bullet and on the first attempt the bullet was too large. Then I modified the size of the bullet. Next, I will consider how to position the bullet so that it is fired from the hip area (Middle of the rectangle) enabling the user to jump over the bullet. After modifying the values, I achieved this. However, it was still hard to jump over so as a consequence I will modify the code so the bullet is fired from a lower position. I also upped how high you could jump to make it easier to jump over the bullet. I changed how high the user could jump to 100 pixels but this might be too large, so I can still change this value if needed.

I need to program the code so that when the bullet hits the hero it will hurt the hero. **Testing** The problem is that the bullet is not registering it has hit the hero when I have tested it. I believe, the bullet does not like the canvas object however I am not completely sure.

I took out the try except comment I had (will need to add this earlier) and also fixed another spelling error which meant I could get the enemy to shoot at the user's character. I took out the try statement as this was only used if there was no enemy, but it still allows the character to shoot however this is not needed as if there is no hero there is no way the enemy can shoot as the game will be finished.

Testing I then wrote the code make the bullet hit the character allowing the user's character to jump over it and then not be hit:



```
if bulletCoordinatesEne[0] <= heroCoordinatesEne[2] and bulletCoordinatesEne[2] >+ heroCoordinatesEne[0] and bulletCoordinatesEne[1] <= heroCoordinatesEne[3
    print("HIT")
```

When you defeat the enemy it then means that part of the code brings up red text. This can be fixed by a try except clause. Once this is added and I have added docstrings and comments this prototype will be finished. I will also modify the values for my bullets ensuring the game is hard but is achievable and is a good game for the user. This will require me to change the amount which you can jump and the starting values for the 2 bullets. I changed the jump values to 50 pixels which requires more skill to jump over the bullet from the user.

I have now commented and put docstrings on the whole program so now it should all work well. I have now finished the game side of my project and now need to add a scoring system and to try and store and save this then also to have a menu.

Review

I have now finished the game part of my project and now have got a working basic game however I will now want to add a better UI by adding a menu and add a scoring system.

Prototype 3 Code

```
from tkinter import *     #This means I can use Tkinter

import time   #Allows me to use time and measure how long bullets

class Bullet():

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 3/10/18
      This is the superclass which will allow me to model how the bullet moves and output it to the screen.  Parts of
code has been adapted from:https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas
      """

    def __init__(self, x1BulletGiven, y1BulletGiven, x2BulletGiven, y2BulletGiven, canvasToUse):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 3/1018
          Inputs - the canvas (Tkinter Object), coordinates (Integers)
          Outputs - None
          Sets up the class when initialised"""

        self.intialSpeedHori = 50.0  #These will set up the intial speed, acceleration for use in the rest of the code for
both the Parabolic bullet and Horizontal Bullet
        self.intialSpeedHoriPara = 50.0

        self.finalSpeedHori = 0.0
        self.finalSpeedVertical = 50.0
        self.finalSpeed = 0.0

        self.distanceHori = 0.0
        self.distanceVertical = 0.0

        self.horizontalAcceleration = 30.0
        self.verticalAccelerationUp = -9.81
        self.verticalAccelerationDown = 9.81

        self.x1Bullet = x1BulletGiven                    #These 4 co-ordinates are where each corner should be placed
        self.y1Bullet = y1BulletGiven
        self.x2Bullet = x2BulletGiven
        self.y2Bullet = y2BulletGiven

        self.canvasBullet = canvasToUse  #Sets up the code

        self.ball = canvasToUse.create_oval(self.x1Bullet, self.y1Bullet, self.x2Bullet, self.y2Bullet, fill="green")  #Outputs
to screen
        self.time = 5      #More constants made.
        self.mass = 0.008
        self.energy = 0.0

    def deleteBullet(self):   #Change to delete_Bullet

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - The class
          Outputs - Deletes the bullet from the screen
          Deletes the bullet from the screen"""

        self.canvasBullet.delete(self.ball)  #Deletes bullet
    def energy_In_Bullet(self):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Class
          Outputs - The energy of the bullet (float)
          Works out the amount of energy in the bullet"""

        self.energy = .5 * self.mass * ( self.finalSpeed ** 2)   #Uses E = 1/2 mv^2
        ##print(self.energy)    #testing

    def powerAtPoint(self, timeTaken):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Time (float), class
          Outputs - The amount of power from the bullet (float)
          Works out the amount of power in the bullet"""

        power = self.energy / timeTaken  #Uses p = E/t

        ##print(power)   #Testing

        return power

    def coord(self):  #CHANGE TO COORD_BULLET

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 12/9/18
          Inputs  - The class
          Outputs - Where the bullet is (Array of floats)
          This will show where the bullet is"""

        coordinatesBullet = self.canvasBullet.coords(self.ball)  #This gives the coordinates for the bullet
        return coordinatesBullet  #Returns it to the main program



class HoriBullet(Bullet):

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      This is a subclass of bullet which calculates and output it to the screen."""

    def speedAtAnyPoint(self, distanceGone):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs -  Distance Gone (Float), class
          Outputs - The final speed of bullet (Float)
          This works out the final speed of the bullet"""

        self.finalSpeedHori = ((self.intialSpeedHori ** 2) + (2 * self.horizontalAcceleration * distanceGone) )** .5   #Uses
V^2 = u^2 + (2as)
        ##print(self.FinalSpeed)   #Testing

        self.finalSpeed = self.finalSpeedHori  #This means the program can use it later
        return self.finalSpeed

    def move_Ball(self, xMovement):   #CHANGE TO MOVE_BULLET_HORI

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Input - How much to move by (Float)
          Output - Moving on the screen (Tkinter Object)
          This will move the bullet on the screen"""

        self.canvasBullet.move(self.ball, xMovement, 0)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen

    def distanceToWorkOut(self):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - class
          Outputs - The distance to go (Float)
          This will work out how far the bullet has to go"""

        self.distanceHori = (self.intialSpeedHori * self.time) + (.5 * self.horizontalAcceleration *(self.time **2)) #Uses S = 
ut + 1/2at^2
        return self.distanceHori



class ParabolicBullet(Bullet):

    """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
      This is a subclass of Bullet which works out the stuff for a parabolic Bullet"""

    def distance_Worker_Up(self):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Vertical Distance (Float), Horizontal Distance (Float)
          This will work out how far the bullet will have to move when the bullet moves upwards"""

        self.distanceVertical = (-.5 * self.verticalAccelerationUp * (self.time /2)**2)  # Uses S = vt - (1/2)at^2
        self.distanceHori = (self.intialSpeedHoriPara * (self.time /2))  #Uses S = ut
        ##print(self.distanceVertical, self.distanceHori)  #Testing
        return self.distanceVertical, self.distanceHori


    def distance_Worker_Down(self):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Vertical Distance (Float), Horizontal Distance (Float)
          This will work out how far the bullet will have to move when the bullet moves downwards"""

        self.distanceVertical = (.5 * self.verticalAccelerationDown * (self.time /2)**2)    #Uses S = ut + (1/2)at^2
        self.distanceHori = (self.intialSpeedHoriPara * (self.time / 2))    #Uses S = ut
        ##print(self.distanceVertical, self.distanceHori)  #testing
        return self.distanceVertical, self.distanceHori
    def speed_Vert_Up(self, distanceVertUp, timeTakenSoFar):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class, Distance Gone Vertically Up (Float), time (Float)
          Outputs - The final Vertical speed (Float)
          This will work out the final vertical speed of the bullet when the bullet goes upwards"""

        self.finalSpeedVertical = (distanceVertUp + (.5 * self.verticalAccelerationUp * (timeTakenSoFar ** 2)))/
timeTakenSoFar  #Uses S = vt - .5at^2

        if self.finalSpeedVertical < 0:  #If the speed is worked to be less than 0 it sets the final speed to 0
            self.finalSpeedVertical = 0.0

        ##print(self.finalSpeedVertical)  #Testing
        return self.finalSpeedVertical

    def speed_Vert_Down(self, timeTaken):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class, time (Float)
          Outputs - Final Vertical Speed (Float)
          This will work out the final vertical speed of the bullet when the bullet goes upwards"""

        self.finalSpeedVertical = self.verticalAccelerationDown * timeTaken  #Uses v = u + at and u is assumed to be 0
        ##print(self.finalSpeedVertical) #Testing
        return self.finalSpeedVertical

    def final_Speed_Parabolic(self):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Final Overall Speed (Float)
          This uses vector working to work out the final speed of a parabolic bullet"""

        self.finalSpeed = ((self.finalSpeedVertical  **2) + (self.intialSpeedHoriPara **2))**.5  #Uses Pythagoras Theorem
to resolve into 1 vector
        ##print(self.finalSpeed)  #Testing
        return self.finalSpeed

    def move_Ball_Para_Up(self, yMovement, xMovement):   #CHANGE TO MOVE_BULLET_HORI

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 27/9/18
          Input - yMovement (Float), xMovement (Float)
          Output - Moving on the screen (Tkinter Object)
          Moves the bullet on the screen"""

        self.canvasBullet.move(self.ball, xMovement, -yMovement)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen

    def move_Ball_Para_Down(self, yMovement2, xMovement2):   #CHANGE TO MOVE_BULLET_HORI

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 27/9/18
          Input - How much to move by
          Output - Moving on the screen
          Works out how far the bullet has to be moved """

        self.canvasBullet.move(self.ball, xMovement2, yMovement2)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen


class Character():

    """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 2/10/18
      This class will make the people on the screen and will allow the computer/user use the character"""

    def __init__(self, canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
          Inputs - The canvas (Tkinter Object), and coordinates (Integers)
          Outputs - Character on the screen
          This will initialise the character meaning it can be used"""

        self.characterOnScreen = canvasCharacter.create_rectangle(x0Given, y0Given, x1Given, y1Given, fill= colour)
#Creates the character
        self.canvasForCharacter = canvasCharacter  #Allows programmer to still use the screen
        x0 = x0Given   #These will give the different x and y points incase I need to use them again
        x1 = x1Given
        y0 = y0Given
        y1 = y1Given
        self.health = 200

    def jump(self):

        """Made By Nathaniel Lowis 1st Edit: 25/9/18 Latest Edit: 3/10/18
          Inputs - Class
          Outputs - None
          This will allow the character to jump and it be shown on the screen"""

        height = 0  #Sets how high the character is

        while height <= 50:  #Whilst the character has not reached the maximum height (50)

            self.canvasForCharacter.move(self.characterOnScreen, 0, -1)#Moves the character up
            self.canvasForCharacter.update()  #This will update the screen, so the user can see it
            self.canvasForCharacter.after(10)  #Waits 10 ms until running the program.  Allows the user to move here I
think
            height = height + 1  #Increments height by 1

        downHeight = 0  #Sets how high the character  Needed to get the character down

        while downHeight <=50:  #Whilst the character has not reached the floor (Need to go down 50 pixels)

            self.canvasForCharacter.move(self.characterOnScreen, 0, 1)#Moves the character
            self.canvasForCharacter.update()  #This will update the screen, so the user can see it
            self.canvasForCharacter.after(10)  #Waits 10ms until it does the anything
            downHeight = downHeight + 1  #Increments downHeight by 1
```

```
    def deleteCharacter(self):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Input - Character Class
          Outputs - Deletes the character
          Deletes character if they are killed"""

        self.canvasForCharacter.delete(self.characterOnScreen)  #Deletes the character

    def lose_Health(self, healthToLose):

        """Made by Nathaniel Lowis 1st Edit: 11/9/18  Latest Edit: 12/9/18
          Inputs - The health to lose (Float)
          Outputs - Health left (Float)
          Makes the character lose health"""

        self.health = self.health - healthToLose  #Lose health
        return self.health



    def move_Left(self, amount):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 26/9/18
          Inputs - The class
          Outputs - Move the Character left
          This will move the character left by 'amount' pixels"""

        self.canvasForCharacter.move(self.characterOnScreen, -amount, 0)#Moves the character
        self.canvasForCharacter.update()  #This will update the screen, so the user can see it

    def move_Right(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 11/9/18
          Inputs - The class
          Outputs - Move the Character Right
          This will move the character right by 2 pixels"""

        self.canvasForCharacter.move(self.characterOnScreen, 2, 0)  #Moves the character
        self.canvasForCharacter.update()  #This will update the screen, so the user can see it

    def coord(self):   #coord_Player CHANGE TO THIS

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 31/8/18
          Inputs  - The character
          Outputs - Where the character is
          This will show where the character is"""

        coordinates = self.canvasForCharacter.coords(self.characterOnScreen)  #This gives the coordinates for the
character
        return coordinates  #Returns it to the main program

    def fireHoriBulletClass(self, coordinateInFireBulletHori):
        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - Coordinates of character in an array
          Outputs - Shooting bullet
          This controls the bullet"""


        bulletShootingHori = HoriBullet(coordinateInFireBulletHori[2], coordinateInFireBulletHori[1],
coordinateInFireBulletHori[2] + 5, coordinateInFireBulletHori[1]+5, self.canvasForCharacter)  #Makes an instant of
the HoriBullet Class
        fireHoriBullet(bulletShootingHori)   #Controls the bullet
        bulletShootingHori.deleteBullet()  #These will delete the bullet afterwards
        del(bulletShootingHori)

    def fireHoriBulletClassEnemy(self, coordinateInFireBulletHoriEne):

        """Made by Nathaniel Lowis 1st Edit: 2/10/18 Latest Edit: 2/10/18
          Inputs - Coordinates of enemy character in an array
          Outputs - Shooting bullet
          This controls the bullet for the enemy"""


        bulletShootingHoriEne = HoriBullet(coordinateInFireBulletHoriEne[2], coordinateInFireBulletHoriEne[1] + 45,
coordinateInFireBulletHoriEne[2] +5 , coordinateInFireBulletHoriEne[1] + 40 , self.canvasForCharacter)  #Makes an
instant of the HoriBullet Class
        fireHoriBulletEne(bulletShootingHoriEne)   #Controls the bullet
        bulletShootingHoriEne.deleteBullet()  #These will delete the bullet afterwards
        del(bulletShootingHoriEne)


    def fireParaBulletClass(self, coordinateInFireBulletPara):

        """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 28/9/18
          Inputs - Coordinates of character in an array
          Outputs - Shooting bullet
          This controls the Parabolic bullet """


        bulletShootingPara = ParabolicBullet(coordinateInFireBulletPara[2], coordinateInFireBulletPara[1],
coordinateInFireBulletPara[2] + 5, coordinateInFireBulletPara[1]+5, self.canvasForCharacter)  #Makes an instant of
the ParabolicBullet Class
        fireParaBullet(bulletShootingPara)   #Controls the bullet
        bulletShootingPara.deleteBullet()  #These will delete the bullet afterwards
        del(bulletShootingPara)


def setUp():     #CHANGE TO SET_UP

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/9/18
      Inputs - None
      Outputs - A working screen with the floor and endpoint and sent to main program
      This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Nat Lowis's Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the user can make it full screen
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)  #Sets up the
    canvasSetUp.pack()  #Outputs screen
    floorSetUp = canvasSetUp.create_rectangle(500, 500, 0, 400, fill="grey")  #Sets up the floor
    endPointSetUp = canvasSetUp.create_rectangle(375, 375, 400, 400, fill="red")  #Sets up the endpoint
    heroSetUp = Character(canvasSetUp, 50, 70, 350, 400, "Blue")  #This will make the user's character
    enemySetUp = Character(canvasSetUp, 300, 320, 350 ,400, "Cyan")  #This will make the enemy class



    return rootSetUp, canvasSetUp, floorSetUp, endPointSetUp, heroSetUp, enemySetUp #Sends everything back to
main program.

def fireHoriBulletEne(bulletFiringEne):  #CHANGE TO FIRE_HORI_BULLET

    """Made by Nathaniel Lowis 1st Edit: 2/10/18 Latest: 3/10/18
       Inputs - The Class HoriBullet
       Outputs - The bullet moving and interacting with the environment
       This will allow the bullet to interact with the environment for the enemy bullet"""

    timer1StartEne = time.time()    #Starts a timer
    distanceToGoEne = bulletFiringEne.distanceToWorkOut()  #Works out how far the bullet goes
    distPer10MilliEne = distanceToGoEne / 50   #Divides the distance by 50 so they are all in equal chunks
    bulletGoneEne = 0  #Creates a variable which works out how far the bullet has gone
    timeDoneHoriBulletEne = 0

    while bulletGoneEne != distanceToGoEne and timeDoneHoriBulletEne < 5000:   #Whilst the bullet has not gone as
far as it needs to

        heroCoordinatesEne = hero.coord()  #Gets hero's coordinates in an array
        #print(heroCoordinatesEne) #Testing
        bulletCoordinatesEne = bulletFiringEne.coord()  #Gets the bullets coordinates in an array
        #print(bulletCoordinatesEne) #Testing

        #try:  #The program will go down this route when there is an enemy class #Not needed anymore
        #print("TRY") #Testing
        if bulletCoordinatesEne[0] <= heroCoordinatesEne[2] and bulletCoordinatesEne[2] >= heroCoordinatesEne[0]
and bulletCoordinatesEne[1] <= heroCoordinatesEne[3]: #If the bullet has hit the hero

            #print("HIT") #Testing
            timer1FinishEne = time.time()  #Stops timer
            timer1Ene = timer1FinishEne - timer1StartEne   #Works out length of time the  bullet has gone for

            ##print(bulletGone) #testing
            ##print(timer1)    #testing

            bulletFiringEne.speedAtAnyPoint(bulletGoneEne)   #Works out the speed of the bullet
            bulletFiringEne.energy_In_Bullet()    #Works out the energy of the bullet
            bulletPowerEne = bulletFiringEne.powerAtPoint(timer1Ene)   #Works out the power of the bullet

            #print(bulletPower) #testing

            bulletGoneEne = distanceToGoEne    #Means the while loop can stop
            hero_Lose_Health(bulletPowerEne) #Makes the hero lose health
    else:

        negativeDistPer10MilliEne = distPer10MilliEne * -1  #This will allow the bullet to move towards the hero
        bulletFiringEne.move_Ball(negativeDistPer10MilliEne)   #Moves the ball the amount it needs to
        bulletGoneEne = bulletGoneEne + distPer10MilliEne  #Added to  bulletGone
        timeDoneHoriBulletEne = timeDoneHoriBulletEne + 100


##      except:  #This is when there is no enemy     #All not needed anymore
##
##          print("Except")
##
##          negativeDistPer10MilliEne = distPer10MilliEne * -1
##          bulletFiringEne.move_ball(negativeDistPer10MilliEne)   #Moves the ball the amount it needs to
##          bulletGoneEne = bulletGoneEne + distPer10MilliEne #Added to BulletGone
##          timeDoneHoriBulletEne = timeDoneHoriBulletEne + 100
##

def fireHoriBullet(bulletFiring):  #CHANGE TO FIRE_HORI_BULLET

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      Inputs - The Class HoriBullet
      Outputs - The bullet moving and interacting with the environment
      This will allow the bullet to interact with the environment"""

    timer1Start = time.time()    #Starts a timer
    distanceToGo = bulletFiring.distanceToWorkOut()  #Works out how far the bullet goes
    distPer10Milli = distanceToGo / 50   #Divides the distance by 50 so they are all in equal chunks
    bulletGone = 0  #Creates a variable which works out how far the bullet has gone
    timeDoneHoriBullet = 0

    while bulletGone != distanceToGo and timeDoneHoriBullet < 5000:   #Whilst the bullet has not gone as far as it
needs to

        enemyCoordinates = enemy.coord()  #Gets enemy's coordinates in an array
        bulletCoordinates = bulletFiring.coord()  #Gets the bullets coordinates in an array

        try:  #The program will go down this route when there is an enemy class

            if bulletCoordinates[0] > enemyCoordinates[0] and bulletCoordinates[1] >= enemyCoordinates[1]:  #If the
bullet is past the enemy
                timer1Finish = time.time()  #Stops timer
                timer1 = timer1Finish - timer1Start   #Works out length of time the  bullet has gone for

                ##print(bulletGone) #testing
                ##print(timer1)    #testing

                bulletFiring.speedAtAnyPoint(bulletGone)   #Works out the speed of the bullet
                bulletFiring.energy_In_Bullet()    #Works out the energy of the bullet
                bulletPower = bulletFiring.powerAtPoint(timer1)   #Works out the power of the bullet

                ##print(bulletPower) #testing

                bulletGone = distanceToGo    #Means the while loop can stop
            enemyHealth = enemy.lose_Health(bulletPower)  #Makes the enemy lose health

            if enemyHealth <= 0:  #If the enemy's health is below or equal to 0
               #CULD PUT THIS IN OWN FUNCTION
              enemy.deleteCharacter()  #Deletes the enemy


         else:

            bulletFiring.move_Ball(distPer10Milli)   #Moves the ball the amount it needs to
            bulletGone = bulletGone + distPer10Milli  #Added to  bulletGone
            timeDoneHoriBullet = timeDoneHoriBullet + 100

      except:  #This is when there is no enemy

         bulletFiring.move_Ball(distPer10Milli)   #Moves the bullet
         bulletGone = bulletGone + distPer10Milli #Added to BulletGone
         timeDoneHoriBullet = timeDoneHoriBullet + 100




def fireParaBullet(bulletFiring2):  #CHANGE TO FIRE_HORI_BULLET

   """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 28/9/18
     Inputs - The Class ParabolicBullet
     Outputs - The bullet moving and interacting with the environment
     This will allow the bullet to interact with the environment"""


   #This bit is for the bullet moving upwards
   timer2Start = time.time()    #Starts a timer
   distanceToGoUp, distanceLeft = bulletFiring2.distance_Worker_Up()  #Works out how far the bullet goes
   distLeft = distanceLeft / 25  #Divides the distance by 25 so they are all in equal chunks to go left
   distPer10MilliUp = distanceToGoUp / 25  #Divides the distance by 25 so they are all in equal chunks to go up
   bulletGoneUp = 0  #Creates a variable which works out how far the bullet has gone
   timeDoneParaBullet = 0  #This will be how long the bullet has moved for
   goDown = True  #This is used to say whether the bullet should go down

   while bulletGoneUp != distanceToGoUp and timeDoneParaBullet < 2500:   #Whilst the bullet has not gone as far as
it needs to and not gone for long enough

      enemyCoordinates = enemy.coord()  #Gets enemy's coordinates in an array
      bulletCoordinates2 = bulletFiring2.coord()  #Gets the bullets coordinates in an array

      try:  #The program will go down this route when there is an enemy class

         if bulletCoordinates2[0] > enemyCoordinates[0] and bulletCoordinates2[1] >= enemyCoordinates[1] -5 and
bulletCoordinates2[0] < enemyCoordinates[2]:  #If the bullet is hitting the enemy
            ##print("HIT UP") #Testing

            timer2Finish = time.time()  #Stops timer
            timer2 = timer2Finish - timer2Start   #Works out length of time the  bullet has gone for
            ##print("HIT UP", timer2)  #Testing
```

```
                ##print(bulletGone) #testing
                ##print(timer1)    #testing

                bulletFiring2.speed_Vert_Up(bulletGoneUp, timer2)   #Works out the speed of the bullet going upwards
                ##print("Working With Speed") #Testing

                bulletFiring2.final_Speed_Parabolic()  #Works out the final speed for the bullet
                ##print("Final Speed") #Testing
                bulletFiring2.energy_In_Bullet()    #Works out the energy of the bullet
                ##print("Energy")  #Testing
                bulletPowerPara = bulletFiring2.powerAtPoint(timer2) #Works out the power of the bullet
                ##print("Power") #Testing

                print(bulletPowerPara) #testing

                bulletGoneUp = distanceToGoUp    #Means the while loop can stop
                enemyHealth2 = enemy.lose_Health(bulletPowerPara)  #Makes the enemy lose health

                #print(enemyHealth)
                goDown = False  #Means the bullet does not have to go down

                if enemyHealth2 <= 0:  #If the enemy's health is below or equal to 0
                    #CULD PUT THIS IN OWN FUNCTION
                   enemy.deleteCharacter()  #Deletes the enemy


            else:

                bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)   #Moves the ball the amount it needs to up
and left
                bulletGoneUp = bulletGoneUp + distPer10MilliUp  #Added to  bulletGoneUp how far it went
                timeDoneParaBullet = timeDoneParaBullet + 100   #Time is updated

        except:  #This is when there is no enemy

            bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)   #Moves the ball the amount it needs to up and
left
            bulletGoneUp = bulletGoneUp + distPer10MilliUp  #Added to  bulletGone how far it went
            timeDoneParaBullet = timeDoneParaBullet + 100    #Time is updated




    if goDown == True:        #Means the bullet can go downwards
        timer3Start = time.time()    #Starts a timer
        distanceToGoDown, distanceLeft2 = bulletFiring2.distance_Worker_Down()  #Works out how far the bullet goes
        distLeftToGo = distanceLeft / 25   #Divides the distance by 25 so they are all in equal chunks to go up
        distPer10MilliDown = distanceToGoDown / 25  #Divides the distance by 25 so they are all in equal chunks to go
down
        bulletGoneDown = 0  #Creates a variable which works out how far the bullet has gone
        timeDoneParaBullet2 = 0  #Creates a variable for amount of time it has been

        while bulletGoneDown != distanceToGoDown and timeDoneParaBullet2 < 2500:   #Whilst the bullet has not
gone as far as it needs to and not for long enough

            enemyCoordinates2 = enemy.coord()  #Gets enemy's coordinates in an array
            bulletCoordinates3 = bulletFiring2.coord()  #Gets the bullets coordinates in an array

            try:  #The program will go down this route when there is an enemy class

                if bulletCoordinates3[0] > enemyCoordinates2[0] and bulletCoordinates3[1] >= enemyCoordinates2[1] -5
and bulletCoordinates3[0] < enemyCoordinates2[2]:  #If the bullet is hitting the enemy

                    #print("HIT Down") #testing
                    timer3Finish = time.time()  #Stops timer
                    timer3 = timer3Finish - timer3Start   #Works out length of time the  bullet has gone for

                    #print("HIT Down", timer3) #Testing


                    ##print(bulletGone) #testing
                    ##print(timer1)    #testing

                    bulletFiring2.speed_Vert_Down(timer3)   #Works out the speed of the bullet going downwards

                    bulletFiring2.final_Speed_Parabolic()  #Works out the final speed of the bullet
                    bulletFiring2.energy_In_Bullet()    #Works out the energy of the bullet
                    bulletPowerPara2 = bulletFiring2.powerAtPoint(timer3)   #Works out the power of the bullet

                    #print(bulletPowerPara) #testing

                    bulletGoneDown = distanceToGoDown    #Means the while loop can stop
                    enemyHealth3 = enemy.lose_Health(bulletPowerPara2)  #Makes the enemy lose health
                    #print("Health: ", enemyHealth) #Testing

                    if enemyHealth3 <= 0:  #If the enemy's health is below or equal to 0
                       #CULD PUT THIS IN OWN FUNCTION
                      enemy.deleteCharacter()  #Deletes the enemy


                else:

                    bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)   #Moves the ball the amount it needs
to left and down
                    bulletGoneDown = bulletGoneDown + distPer10MilliDown #Added to  bulletGoneDown
                    timeDoneParaBullet2 = timeDoneParaBullet2 + 100  #Updates time

            except:  #This is when there is no enemy

                bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)   #Moves the ball the amount it needs to
                bulletGoneDown = bulletGoneDown + distPer10MilliDown #Added to  bulletGoneDown
                timeDoneParaBullet2 = timeDoneParaBullet2 + 100  #Updates Time


#def key(event):

 #   """Used as testing """

  #  #print("pressed", repr(event.char))  TESTING
```

```
def enemy_Shooting():

   """Made By Nathaniel Lowis 1st Edit: 2/10/18,  Latest Edit: 2/10/18
     Inputs - None
     Outputs - None
     This allows the enemy to shoot"""

   characterPosistionForEnemy = enemy.coord() #Gets the coordinates for the enemy's character
   enemy.fireHoriBulletClassEnemy(characterPosistionForEnemy)  #Fires bullet

def hero_Touch_Enemy():

   """Made By Nathaniel Lowis 1st Edit: 28/9/18, Latest Edit: 3/10/18
     Inputs - None
     Outputs - None
     This will mean the user's character is hurt if it touches the enemy"""

   heroCurrentCoordinates = hero.coord()  #Gets the hero's coordinates
   enemyCurrentCoordinates = enemy.coord() #Gets the Enemy's coordinates

   try:  #If there is an enemy

      if heroCurrentCoordinates[2] >= enemyCurrentCoordinates[0] and heroCurrentCoordinates[3] >=
enemyCurrentCoordinates[1]:  #If the hero touches the enemy

         hero_Lose_Health(50) #Hero Loses health
         hero.move_Left(20)  #Hero pushed back

   except:  #If there is no enemy nothing should happen
      pass

def hero_Lose_Health(healthToLoseFunction):

   """Made by Nathaniel Lowis 1st Edit: 28/9/18, Latest Edit: 28/9/18
     Inputs - Health to lose (Float)
     Outputs - None
     This will take away health if the hero is hit and stop the game if the hero is dead"""

   heroHealth = hero.lose_Health(healthToLoseFunction)  #Gets Hero to lose health
   #print(heroHealth)  #Testing

   if heroHealth <= 0:  #If the hero is 'dead' it should delete everything
      frame.destroy()


def callback(event):

   """Edited from: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm  Made by Nathaniel Lowis  1st Edit:
25/8/18  1st Edit: 31/8/18
   Inputs - What the user did
   Output- Allows user to move left and right
   Means we can press on the screen and use the buttons"""

   frame.focus_set()   #This will mean we can use the left and right button as you have to press the window
```

```
    ##print("clicked at", event.x, event.y)  Used as testing

def heroMoveLeft(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 28/9/18
    Inputs - What the user did
    Output- Allows user to move left
    Makes the hero move left and checks if the game has finished"""

    hero.move_Left(2)#Calls this method to move character
    hero_Touch_Enemy()
    finishGame()

def heroMoveRight(event):

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 28/9/18
    Inputs - What the user did
    Output- Allows user to move Right
    Makes the hero move Right and checks if the game has finished"""

    hero.move_Right() #Calls this method to move character##
    hero_Touch_Enemy()
    finishGame()

def hero_Jump(event):

    """Made by Nathaniel Lowis  1st Edit: 25/9/18.  Latest Edit: 3/10/18
    Inputs - What the user did
    Output- Allows user to move Jump
    Makes the hero move Right and checks if the game has finished"""

    heroCoordinatesJumpFunction = hero.coord()
    if heroCoordinatesJumpFunction[3] != 400:   #This is so the code
        pass
    else:
        hero.jump() #Calls this method to move character##

def finishGame():

    """Made by Nathaniel Lowis  1st Edit: 31/8/18  Latest Edit: 31/8/18
      Inputs - Nothing
      Outputs - Checks if game has finished
      This will check if you have passed the ending point and if you have deletes the screen"""


    ##print("TRYING")  Used for testing
    x0Endpoint = 375  #These are the known coordinates for the endpoint
    x1Endpoint = 400
    y0Endpoint = 375
    y1Endpoint = 400

    characterPosistion = hero.coord()  #Works out where the user's character is

    if characterPosistion[2] >= x0Endpoint and characterPosistion[3] >= y0Endpoint and characterPosistion[3] <=
y1Endpoint and characterPosistion[2] <= x1Endpoint:   #Checks if the character has passed the endpoint
```

```
        frame.destroy()  #If they have it will delete all objects on the canvas


def fireBullet(letter):

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 3/10/18
       Inputs - The q which must (for some reason) be included
       Outputs - None
       This will start the process of firing a Horizontal bullet and get the enemy to shoot first"""

    try:  #If there is an enemy it should start the shooting process

        enemy_Shooting()

    except: #If there is no enemy it should do nothing
        pass

    characterPosistion = hero.coord() #Gets the coordinates for the user's character
    hero.fireHoriBulletClass(characterPosistion)  #Fires bullet

def fireBulletPara(letter):

    """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 27/9/18
       Inputs - The w which must (for some reason) be included
       Outputs - None
       This will start the process of firing a Parabolic bullet"""


    characterPosistionPara = hero.coord() #Gets the coordinates for the user's character
    hero.fireParaBulletClass(characterPosistionPara)  #Fires bullet



root, frame, floor, endPoint, hero, enemy= setUp()  #Sets up game






frame.bind("<Left>", heroMoveLeft)  #Moves Character left if left button pressed
frame.bind("<Right>", heroMoveRight)  #Moves Character Right if right button pressed
frame.bind("<Up>", hero_Jump)      #Allows the user to jump
frame.bind("<Button-1>", callback)  #Means user can press screen
frame.bind("<q>", fireBullet)     #Allows the user can shoot a horizontal Bullet
frame.bind("<w>", fireBulletPara)   #Allows the user to shoot a parabolic bullet


frame.pack() #Sends it to the screen



frame.mainloop() #Infinte loop used
```

Fourth Prototype

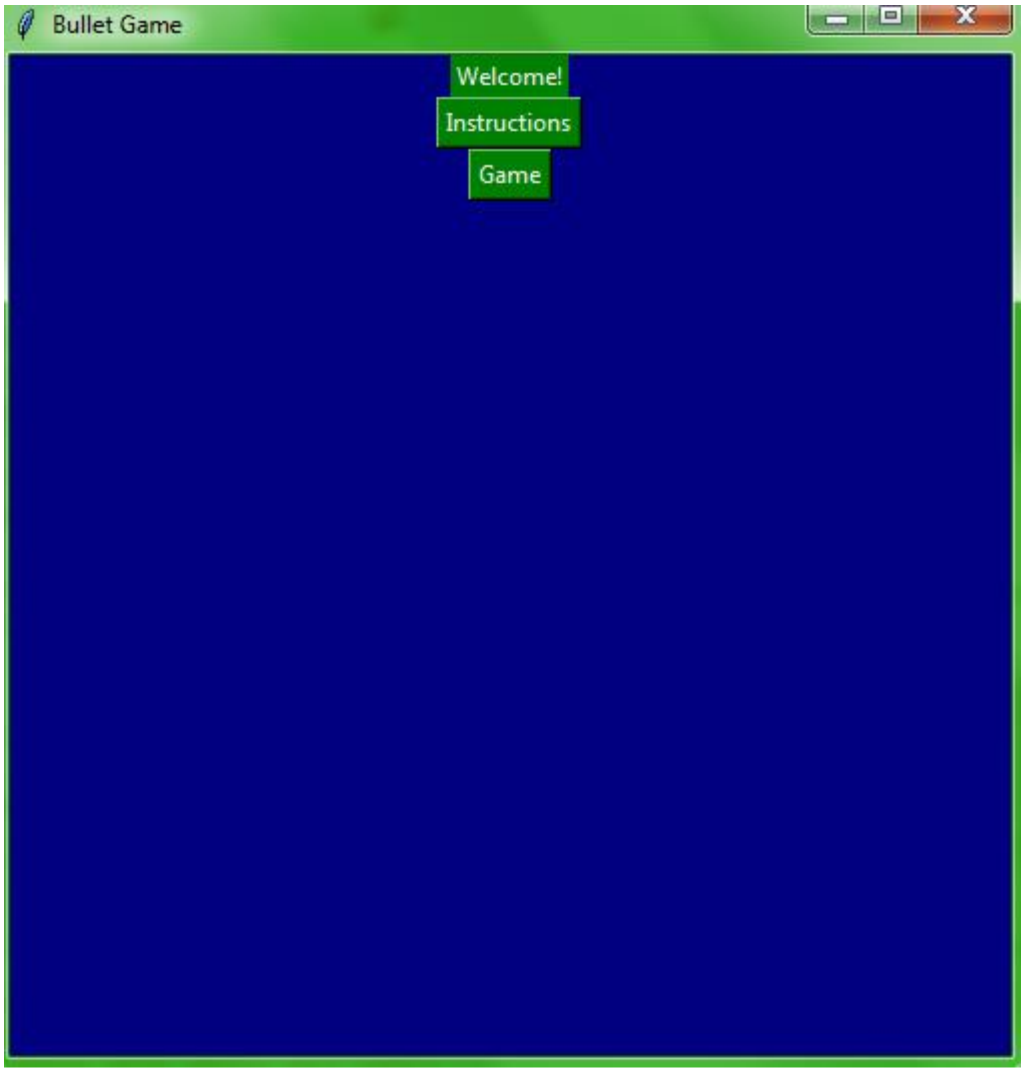
– Scoring made and saved in csv files

For my final prototype I want to work out scoring, how to change screens and create a leader board to allow the user to track their score against other players. I want a menu and more of a GUI to make it easier for the user to navigate the game.

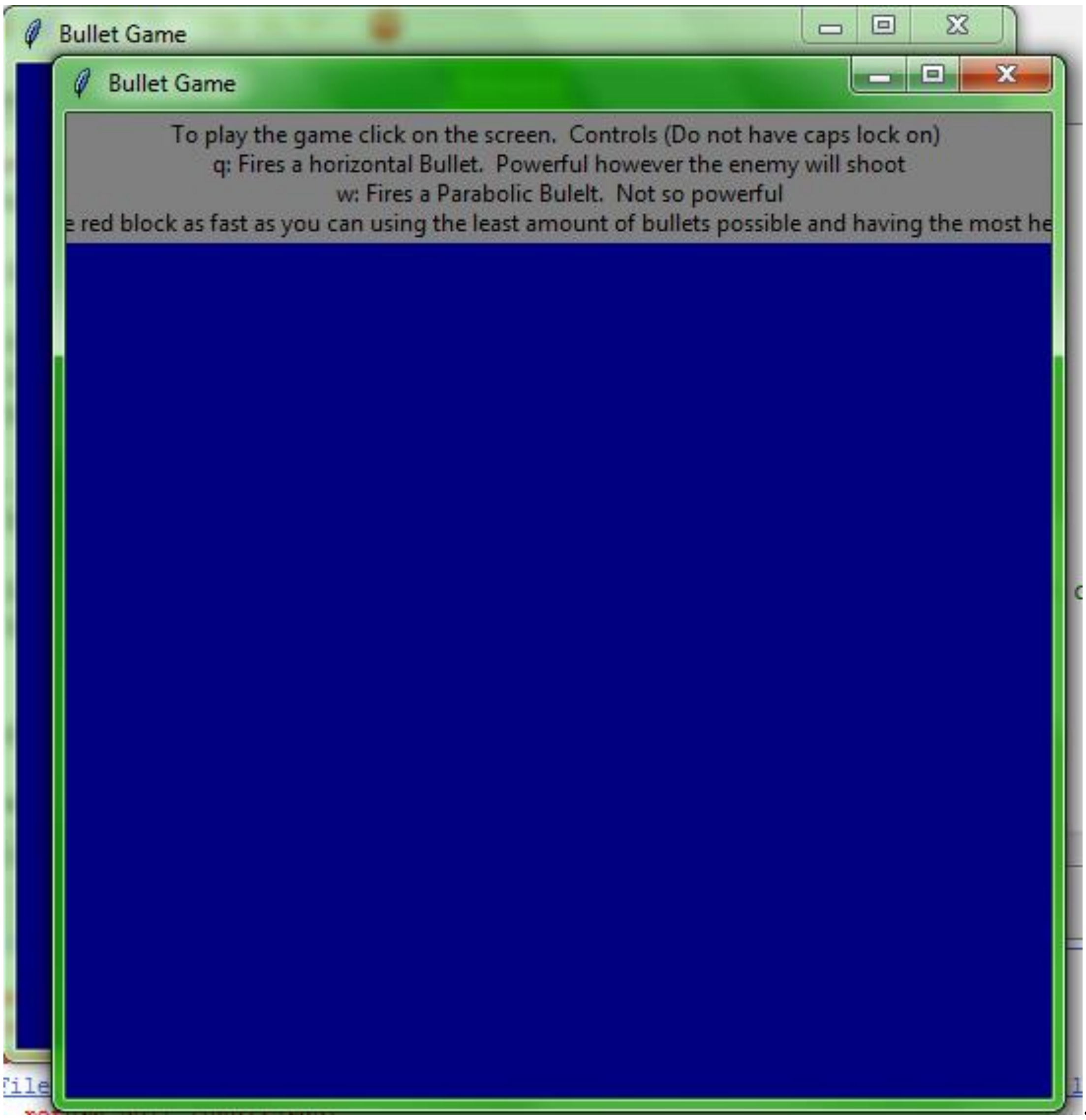
What I first did was to download my game into a new Python file like this:

```
import gamePrototype4

setUp()
```

This is so I can use a game within another file which then means I can play the whole game in one file. **Testing** I have tried to create a menu and then delete itself when a button is pressed which is not working at all. I am trying to get a file to import another file and see if it can show the instructions. However, I cannot import the file. So, at the moment I have got the menu to show instructions, but it does not delete the window so at the moment it looks like this:



Testing When I click the Instructions button it will send the user to a new screen: **Testing**

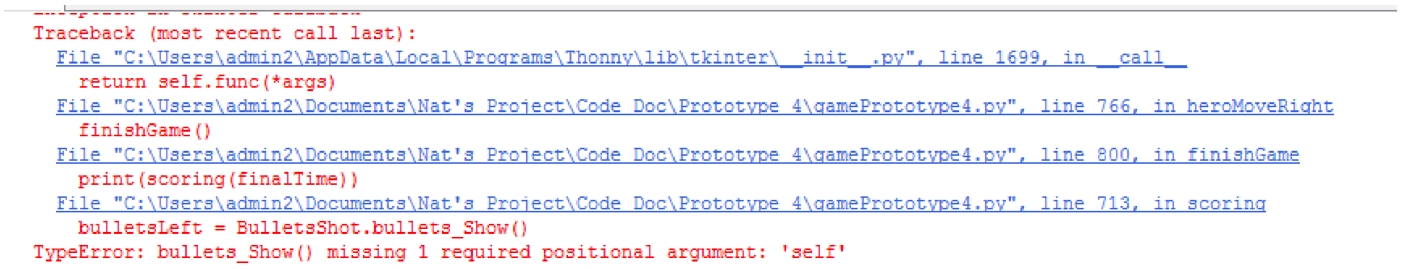

To play the game click on the screen. Controls (Do not have caps lock on)
q: Fires a horizontal Bullet. Powerful however the enemy will shoot
w: Fires a Parabolic Bulelt. Not so powerful
e red block as fast as you can using the least amount of bullets possible and having the most he

. As you can see the other window has not been deleted but I cannot work out how to delete it all.

Then I started importing the game into the main file. I firstly tried using .exec commands however I could not work with them, so I settled on using the import function. **<u>Testing</u>** When the user presses the Game it will import the game and run it.

Next, I will program the game to work out what your score is by adding a scoring system. Adding a scoring system allows the user to see how well they are actually doing. Firstly, I will now get the game to work out how many times you have shot the bullet and work out how long the game has run for.

Firstly, when I started trying to work out how many bullets has been shot, I tried setting up variables, but this never worked as I could not get them global and I was unable to reassign numbers to them. Instead I have made a class and set it up which can take bullets away and say how many bullets you have. **<u>Testing</u>** I will implement this into the game by saying whether there are more than 0 bullets which was successful. It was shooting when there were more than 0 bullets left! I then tried to shoot a bullet when I had already shot 10 (The limit is 10) and it did not allow me. Now I will try and get the game to make a score. **<u>Testing</u>** When programming the score, there were a lot of spelling errors concerning attributes and now my game is broken. I have had to try and work out my code but one of my methods is saying I am not even passing self into the method. My character cannot die as my scoring module is not working correctly at all.



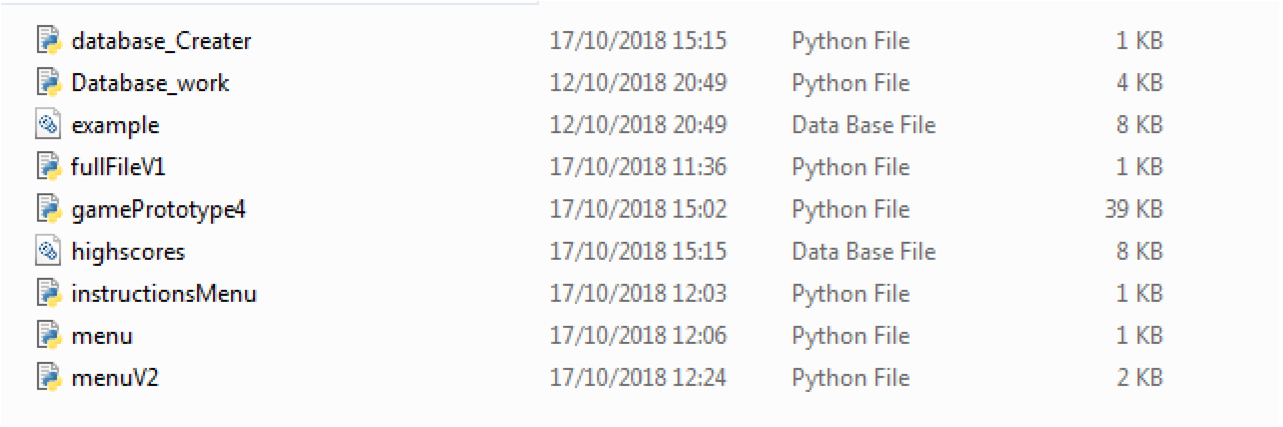
```
Traceback (most recent call last):
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1699, in __call__
    return self.func(*args)
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 4\gamePrototype4.py", line 766, in heroMoveRight
    finishGame()
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 4\gamePrototype4.py", line 800, in finishGame
    print(scoring(finalTime))
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 4\gamePrototype4.py", line 713, in scoring
    bulletsLeft = BulletsShot.bullets_Show()
TypeError: bullets_Show() missing 1 required positional argument: 'self'
```

This shows that I have not passed in self in my method.

This problem was that I was calling BulletShot not bulletsLeft but now it is saying it is an unbound error, so I am searching through my code looking for any other mentions of bulletsLeft. This was because I had given another variable the same name. Now I have changed it to howManyBulletsLeft. **Testing** After a couple more errors concerning variable names I was able to get a score of 490 or so. When I used bullets, I got a score of 209 meaning finishing the game as quickly as possible is the most important component of the score so if you died early by running into the enemy, you could get more points than if you completed the actual game by killing it. To sort this out I will give an extra 100 points for finishing the level to counteract this problem.

Now I will try and get this score onto the screen and then be able to add it to a database/csv. To do this I am going to import SQLite3 (A SQL language for Python) which then allows me to work with a database. I have code on how to program a database which I will use.[7] This code was written in one of my lessons. I will need code to make the database, to get all the records off a database and use a sorting algorithm on it and also code to add data to the database. Finally, I will need code to ask for a username and for the code to query the database checking whether there is already a username like that already in the database. All of this is so I can allow the user to track their scores and position against other users. I also will use SQLite3 as this allows me to make and manipulate a database.

I will first tackle the problem of creating a database. **Testing** My code is just adapted from code I already had[8] and I then made the database:

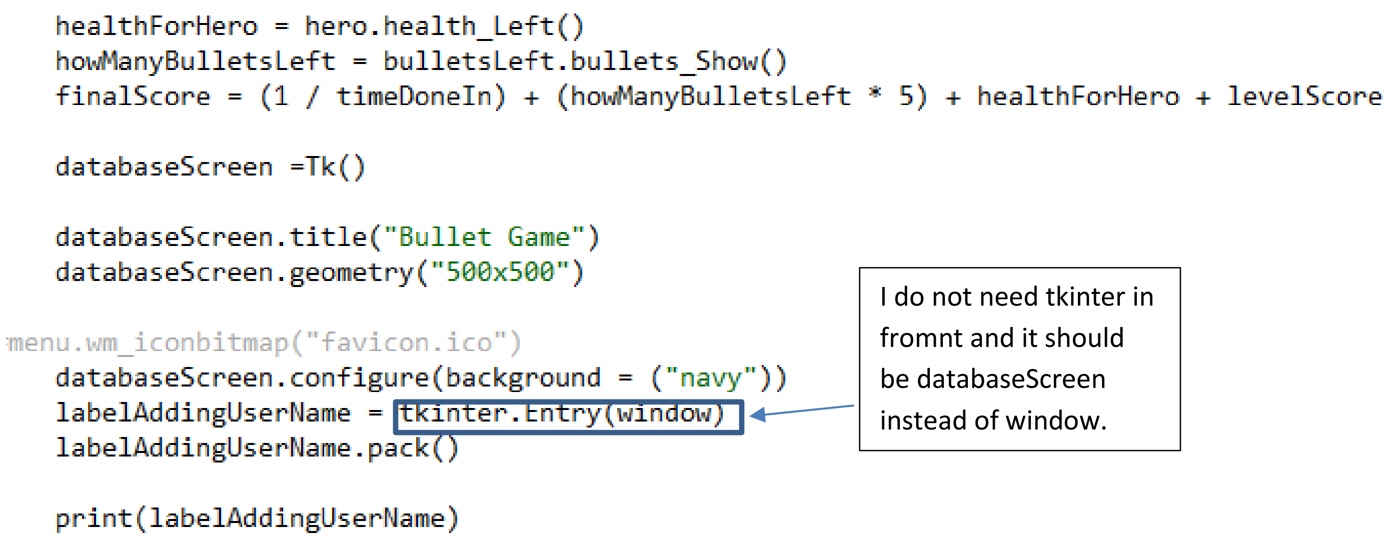


database_Creater	17/10/2018 15:15	Python File	1 KB
Database_work	12/10/2018 20:49	Python File	4 KB
example	12/10/2018 20:49	Data Base File	8 KB
fullFileV1	17/10/2018 11:36	Python File	1 KB
gamePrototype4	17/10/2018 15:02	Python File	39 KB
highscores	17/10/2018 15:15	Data Base File	8 KB
instructionsMenu	17/10/2018 12:03	Python File	1 KB
menu	17/10/2018 12:06	Python File	1 KB
menuV2	17/10/2018 12:24	Python File	2 KB

I will clear this out as there is a lot of unneeded files here (EG example, database_Work, menu). This has created my database, and this will allow me to add files to it. I will then go into gamePrototype4 (the file which has my game in it) and start to allow the user to add their username and add it to the file.

[7] https://repl.it/@NathanielLowis/Database-in-python In Bibliography
[8] https://repl.it/@NathanielLowis/Database-in-python In Bibliography

I started this by adding this code to my program:



```
    healthForHero = hero.health_Left()
    howManyBulletsLeft = bulletsLeft.bullets_Show()
    finalScore = (1 / timeDoneIn) + (howManyBulletsLeft * 5) + healthForHero + levelScore

    databaseScreen =Tk()

    databaseScreen.title("Bullet Game")
    databaseScreen.geometry("500x500")

menu.wm_iconbitmap("favicon.ico")
    databaseScreen.configure(background = ("navy"))
    labelAddingUserName = tkinter.Entry(window)
    labelAddingUserName.pack()

    print(labelAddingUserName)
```

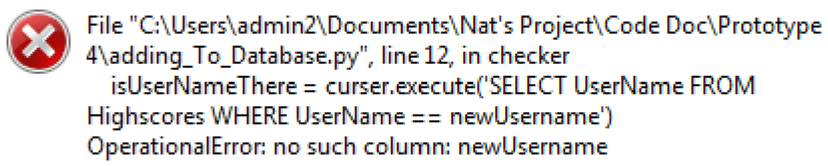
I do not need tkinter in fromnt and it should be databaseScreen instead of window.

All this allows me to do is make a window for the database.

Testing The problem is that I could spam open many windows very easily. This was easy to fix as shown above. I did not need the tkinter command as I had already imported all the Tkinter functions

I also put all the code for the database into another file, so it would not interfere with my game. I now need to work out how to enter my name and start to get it checked. After checking a website[9] I tried using the .get command to get the contents from the user when they press a button. **Testing** I did forget to pack the button, but it did not work. This is because I am calling something which is not global. Even when I did call it, the program still did not take my name which I had inputted. So, at the moment I cannot add a username. How I might get around this is to add it to a csv file and then have another file which sorts out the username. It is a bit complicated, but I think it could work. This would also allow me to have all the SQL in the same file as adding the username. I could get this to work so you can add a username and the program can check it. Now I need to save the scored to a csv file send it over and then delete what is in that csv file so then I can add it all to the database.

Before doing all of that I am going to get the program to check through the whole database and see if there is already a person with that username. **Testing** The code, I wrote returned this error:



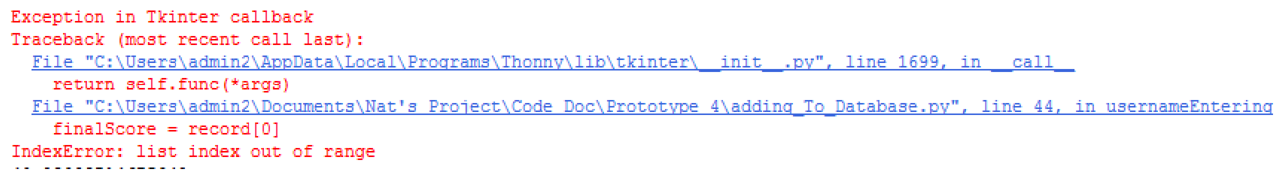
```
File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype
4\adding_To_Database.py", line 12, in checker
  isUserNameThere = curser.execute('SELECT UserName FROM
Highscores WHERE UserName == newUsername')
OperationalError: no such column: newUsername
```

so for some reason it thinks newUsername is a column. When I added some quote marks around it, the code seemed to work. It should now all be working. Now I need to get it to print it all out which worked with a .fetchall. Now I need to get it to add data to the database but firstly I need to make sure the program can see that it is all empty.

I have now started to program it that the code checks if the username is there and see what happens if it does. **Testing** Firstly the label was not appearing, but this is because I was forgetting to pack it to the screen. When I added Nat, everything seemed to work so I now need to get it to add my score then using csv files for it all to work. This happened because the code for the score and for the database is in different files. I am using a csv file as it can be overwritten very easily but also be read easily and use the csv to send data across the different program files.

[9] http://effbot.org/tkinterbook/entry.htm In Bibliography

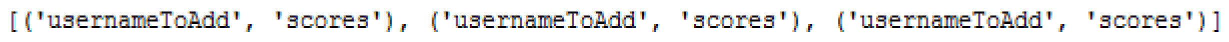
I have now written code to add your score to a file and then send it to the program which checks for username in the database. **Testing** When I ran the code I got an error this error:



```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\admin2\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1699, in __call__
    return self.func(*args)
  File "C:\Users\admin2\Documents\Nat's Project\Code Doc\Prototype 4\adding_To_Database.py", line 44, in usernameEntering
    finalScore = record[0]
IndexError: list index out of range
```

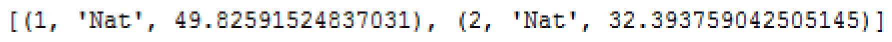
Testing So for some reason the score does not seem to be being added to the record. I ran the code again and itdid not seem to work. I used a debugger on the code and still it did not seem to work. However, when I added the number I was forgetting to add a "/n" so I added this and checked what happened, but I still got the same error! However, I got an error as there was always another row afterwards. So, to counteract this I added an if statement saying if the array was empty it should just pass and not do anything. When I ran this code, I did not get any error codes, so I am assuming it works.

Now I need to add these values to the database and then just get the database to be displayed in order. To add the values into the database I used. Everything seemed to work but I am now going to check the code worked by displaying it. **Testing** When I displayed the database this was in it:



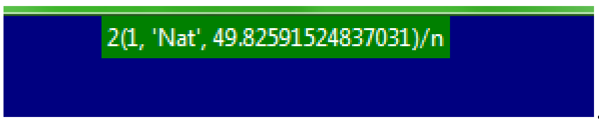
```
[('usernameToAdd', 'scores'), ('usernameToAdd', 'scores'), ('usernameToAdd', 'scores')]
```

This is slightly wrong! As well using my username as the Primary Key does not seem to work. I was able to get around this problem by using the command executemany which meant I added a tuple. How I was checking the database was not working either, so I decided to change my code. The code which I rewrote might have crashed Python. **Testing** So the checker did work but it does not like to change the label. So instead of having unique usernames I will have a unique number. Therefore, I need to reset my database which I did with no problem. I then changed my checker to work out the largest ID number and after a few spelling errors I still got an error. The error I got was because when I created the database I had a comma between the column name and what it will be. After running the code twice everything worked. **Testing** This is what the database looks like:



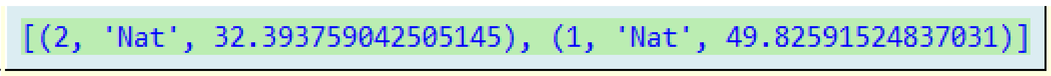
```
[(1, 'Nat', 49.82591524837031), (2, 'Nat', 32.393759042505145)]
```

The next thing I need to do is to make it sort this all out and display it via Tkinter and then we will have everything done. Firstly, I checked to see what the sort algorithm for Python is like and the timsort is very good with a big O notation of O(nlogn) which is not bad at all. So, I will use this to sort my database. However, it is hard to say which element to sort by, so I will actually use Quick Sort (I have code for this from when I have written it at school[10]) as it is the fastest sorting algorithm and is not hard to implement looking at a specific element. As the database is also unsorted it will have a Big O notation of O(nlogn). **Testing** I had a few errors because I was not using the right data type. When I had fixed this my whole table did not output to the screen:



```
2(1, 'Nat', 49.82591524837031)/n
```

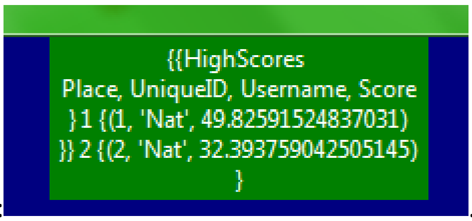
. However, a more pressing concernt was that it sorted the smallest item first



```
[(2, 'Nat', 32.393759042505145), (1, 'Nat', 49.82591524837031)]
```

This was from my debugger. To edit this in my sorting code I changed a < to a >. My other problem was that I forgot to

[10] https://repl.it/@NathanielLowis/Quick-Sort In Bibliography

have a tableDisplay + tableDisplay. This is now what my table looks like: 

```
{{HighScores
Place, UniqueID, Username, Score
}1 {(1, 'Nat', 49.82591524837031)
}} 2 {(2, 'Nat', 32.393759042505145)
}
```

. This has meant once I have added it to the main table I have finished my 4th prototype.

Testing I have also noted that you could earn more points by killing yourself by running into the enemy and earn more points than if you completed the level still. To counteract this, you earn an extra 100 points for completing the level. I also got one of my stakeholders (Max) to check and see how the colour scheme was. He said he could see everything correctly.

Finally, all I am doing is putting all the needed files in the same folder and renaming some folders. **Testing** When I was playing my game I found that the game would not finish if I touched the endpoint. I quickly took out my print statement for the scoring and it all worked (I changed a few variable names which did crash the game briefly before I solved the problem). I have now written all my code with correct variable and module names. All I have to now do is comment and to add my final docstrings and then I have completed all the code. I also named the game Bullet Shot. I also changed some of the variable names along with some methods names (I had a few coordinate methods for different classes, so I have changed them to coord_Bullet and coord_Player) and finally have changed what a few comments and docstrings say. All of these changes were just to make it clearer about what the code is doing. The variable name changes were to make sure each variable is unique.

Review

I have now finished! All the code works for the iterative testing and I have now a working UI.

Prototype 4 code

I am not adding code here as the only difference between this prototype and the final one is me just changing some names of variables and not much else.

Evaluation

Testing to inform development.

Provided annotated evidence of post development testing for function and robustness.

Videos: What they show

To find the evidence for each test check the notes which will specify the video and where it is.

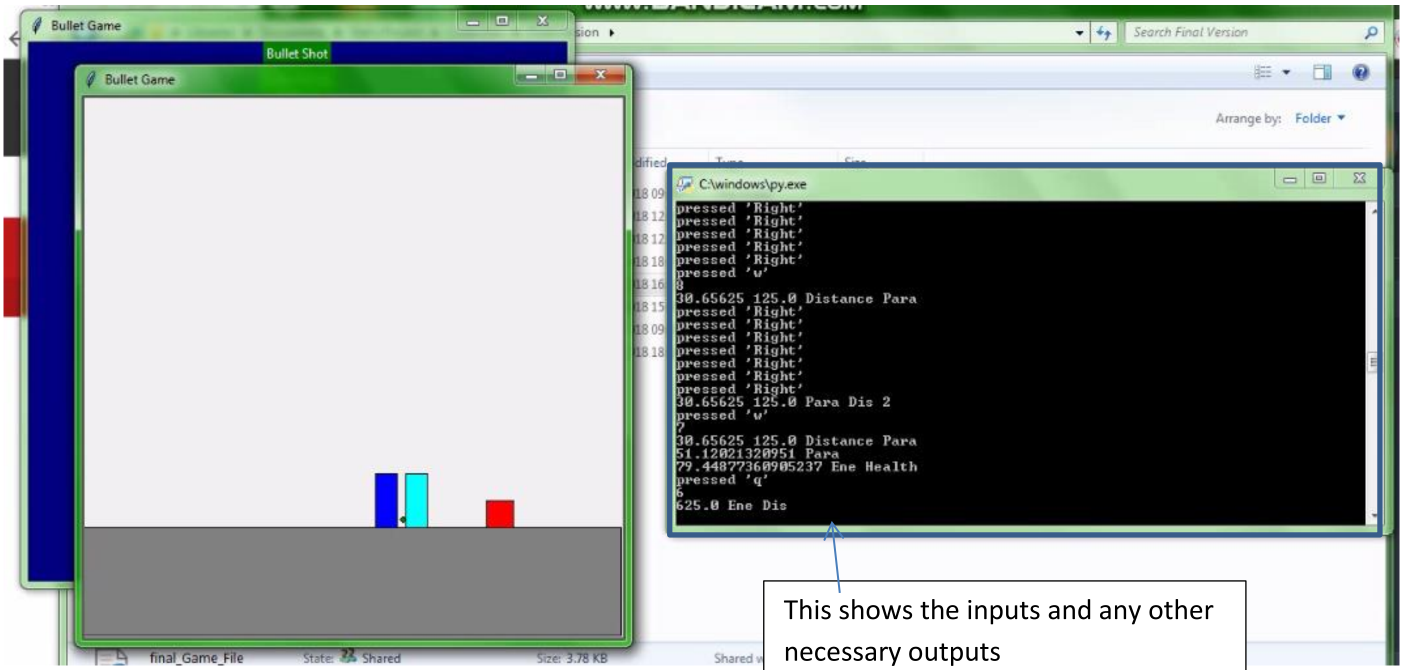
Video 1 – This firstly shows the menu and the instructions screen. It then shows the game running with emphasis on moving to start with. You then see me get shot at by the enemy when I press one of the fire buttons. and the output of when it hits the user's character. It then shows the User's return fire after the bullet from the enemy. This is then repeated. Then you see me fire my parabolic bullet twice and you can see what happens when that is outputted. The single number after the pressed 'w/q' shows how many bullets I have remaining. You then see me 'kill' the enemy and end the game.

Video 2 – You see the menu then the game. You see the bullets being shot with me killing the enemy (you can see how much health is left and how many bullets are left). Then you see the game end because of an error with a variable name (which is what the next 10 seconds shows me finding). You then see me restart the game and kill the enemy and finish the game. Next, you see me enter a user name and try and restart the game. After that you see me restart the game and die by running into the enemy 4 times and enter a username. You can see the database of scores. You see me shoot a parabolic bullet and then input different buttons to show the game does not break when thw wrong keys are pressed. Finally, you see me kill the enemy and finish the level and enter another username.

Video 3 – Here you see the database being created.

How I tested the game.

In every video will be a black console. This shows the input from the keyboard of what the user presses and also the numbers for each bullet when shot. The user will not see this; however, it is used here allowing me to see what button I have pressed thus knowing the input and you can see what the output is.



```
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'w'
8
30.65625 125.0 Distance Para
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
pressed 'Right'
30.65625 125.0 Para Dis 2
pressed 'w'
7
30.65625 125.0 Distance Para
51.12021320951 Para
79.44877360905237 Ene Health
pressed 'q'
6
625.0 Ene Dis
```

This shows the inputs and any other necessary outputs

Actual Output explanation

True – It means that the expected output did happen.

False – It means the expected output did not happen.

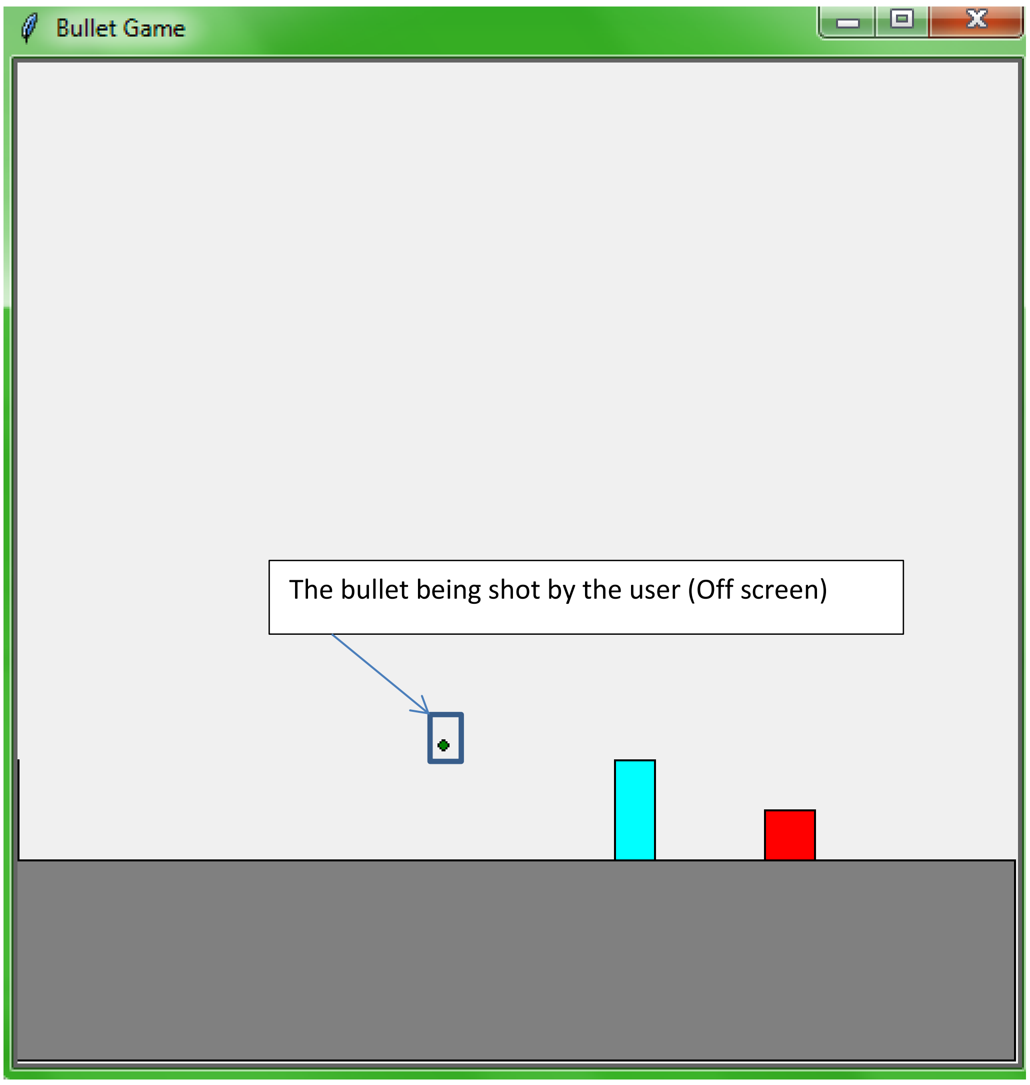
Test Number	Statement	Justification	Input	Expected Output	Actual Output	Notes
1	The program will work. The player will be able to play a game which is realistic and is a shooting 2D platform game with an enemy which you need to defeat in it. The program will be robust enough to not break if a wrong input is entered.	I do not want there to be any game breaking bugs or glitches so I will want to have identified them before I have finished. I want the game to be finished and that the user feels that it is polished and finished.	-	The game does not unexpectedly die or anything like that	FALSE	See all 3 videos. Can still output multiple bullets
2	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character, and this will be the way to actually interact with the level.	Valid - Right Key	The character moves to the right	TRUE	See video1 19 – 20 seconds
3	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character, and this will be the way to actually interact with the level.	Valid - Left Key	The character moves to the Left	TRUE	See video1 20 -21 seconds
4	The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character, and this will be the way to actually interact with the level.	Valid - Up key	The character moves up then down landing on the floor.	TRUE	See video1 22 – 23 seconds
5	The program will have an enemy	This is so there is some difficulty in the game.	-	There is more than one	TRUE	See video1 16 seconds

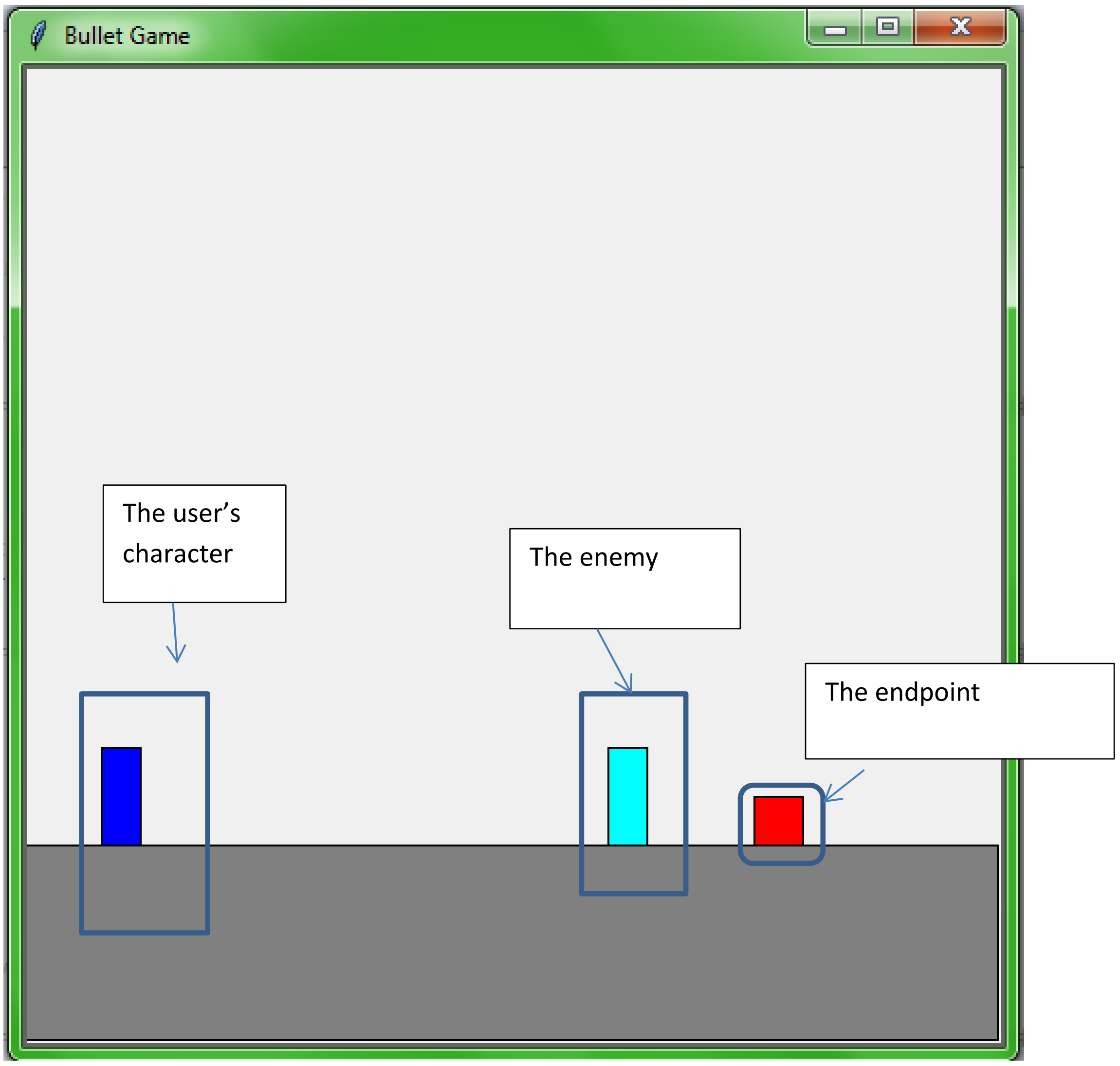
				character on the level		(Cyan Block)
6	The program will end when the user's character reaches an endpoint.	This means the level will end at one point and will not go on forever.	Valid - Reach the endpoint	The Game ends showing the next screen	TRUE	See video1 1:36
7	The program will allow the user to change which bullet to use	This means they can use both bullets.	Valid – Press q, e, q	This should shoot 2 bullets	FALSE	See video1 34 – 45 seconds. Instead have two buttons to shoot each bullet
8	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid -Press q	The enemy should shoot a bullet. Hits the user's character and disappears	TRUE	See video1 40 – 45 seconds
9	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid - Press q	Once the enemy's bullet disappears the user will fire a bullet and when it hits the enemy, this too will be deleted	TRUE	See video1 40 – 35 seconds
10	The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Valid -Press w	The hero will fire a bullet and when it hits the enemy it will disappear	TRUE	See video1 55 – 1:05. Cannot see bullet as the user's character is so close
11	The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Valid -Press q	When bullet hits the hero, it will output the power of it and the health remaining	TRUE	See video1 32 – 36 seconds
12	The program will take away	This makes it realistic and also means you	Valid -Press q	When the bullet hits	TRUE	See video1 36

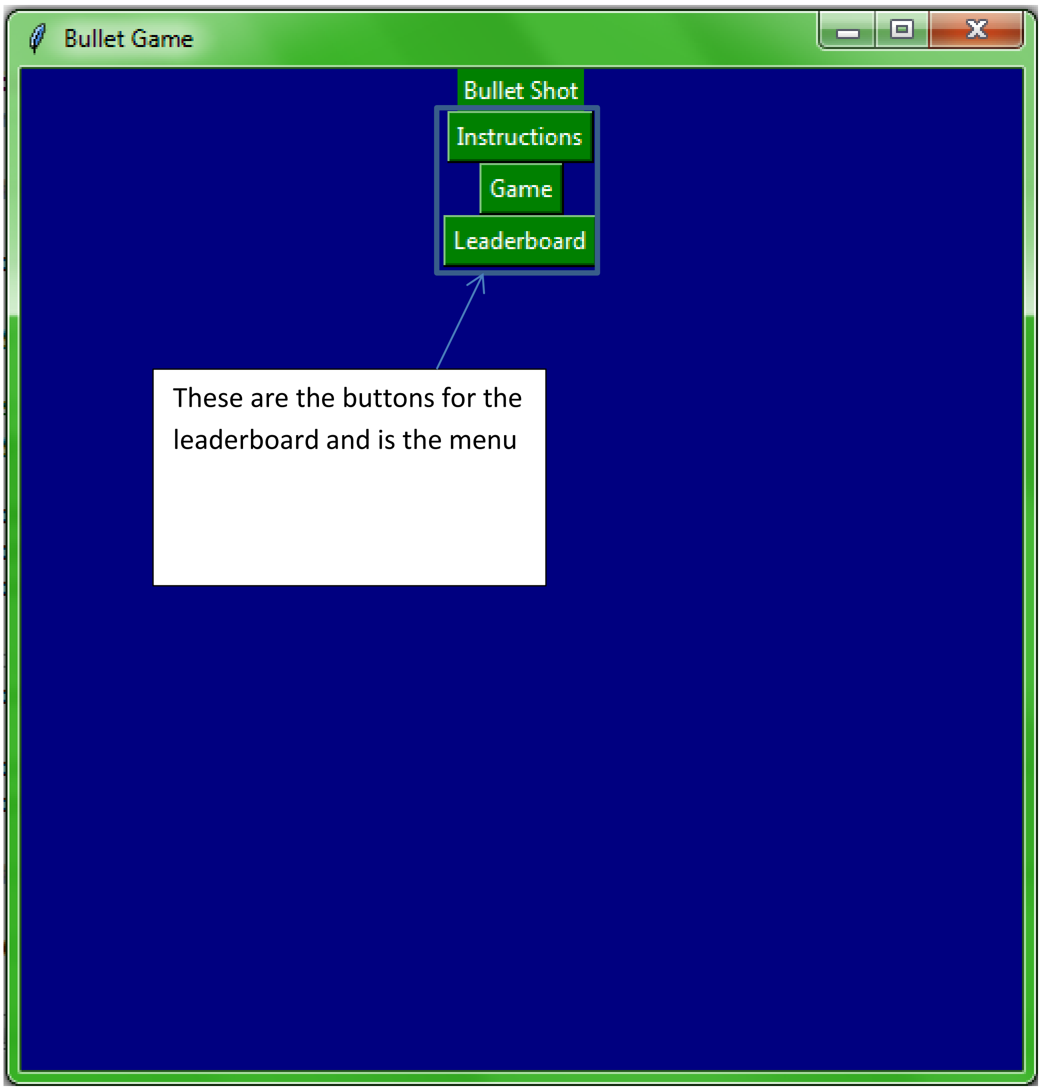
	health from a character when a bullet hits it using the power equation.	can destroy the enemy/character.		the enemy, it will output the power and the health remaining		– 38 seconds
13	The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Valid -Press w	When the bullet hits the enemy, it will output the power and the health remaining	TRUE	See video1 55 – 1:05. Cannot see bullet as the user's character is so close
14	The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Valid – Press q	The program will output how far the enemy bullet is going (Will be a float)	TRUE	See video1 30 – 35 seconds
15	The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Valid – Press q	The program will output how far the hero's bullet is going (Will be a float)	TRUE	See video1 35 seconds
16	The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Valid – Press w	The program will output how far the bullet is going (Will be a float)	TRUE	See video1 55 seconds – 1:00
17	The program will have two bullets a parabolic one and a horizontal one.	This allows the user to have more than one weapon	-	There are two bullets the user can use	TRUE	See video1 2 Bullets being shot
18	The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.	q	The number of bullets left decreases by 1	TRUE	See video1 41 seconds
19	The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.	w	The number of bullets left decreases by 1	TRUE	See video1 54 – 57 seconds
20	The program will have a timer which allows it	This is to be used in the total points and allows the user to try	-	There is a time at the end which is	FALSE	See video1 1:36.

	to work out how long the user has taken.	and make it quicker		similar to the time I will take on my phone		Fixed in video2 2:12 (Time on Phone was 15 seconds)
21	The program will work out a total.	This is to give a point for the game	-	At the end the program will work out a score	TRUE	See video2 2:28
22	The program will allow the user to see how well they have done	This will allow the user to see how well they have done	-	The score is outputted	TRUE	See video2 2:28
23	The program will have a database of high scores	This is so all scores can be entered and also, so the user can see how well they are doing.	Running database_Creator	highscore.db is made in same file	TRUE	See video3 (Whole)
24	The program will put the score in a database	This is so the user can see how they compare to other people.	-	The score is added into the database (We can see it on the high scores page)	TRUE	See video2 2:40 – 2:45
25	The program will allow the user to enter a username.	This is so it can go into the database along with their score.	Valid: 10low1	This is accepted	TRUE	See video2 1:30 – 1:40
26	The program will only allow usernames which no one else has used before.	This is so it is easy for the User to see their score in the database	Invalid: 10low1	This is not accepted	FALSE	See video2 2:25 – 2:30. Username accepted
27	The program will allow the user to choose whether to see the high scores or to play the game	This will be a menu and allow the user to see different parts of the program.	-	There is a menu showing different options	TRUE	See video1 Start
28	The program will display a table of high scores.	This allows the user to see how well they are doing.	Press Leaderboard	There is a leaderboard	TRUE	See video2 2:40 – 2:45
29	The program will make sure all characters are on the ground when jumping	This will then make sure it looks realistic and also means you can see how high they are.	Invalid: Press up button multiple times	Only jumps once until it reaches the ground	TRUE	See video2 3:20 – 3:25

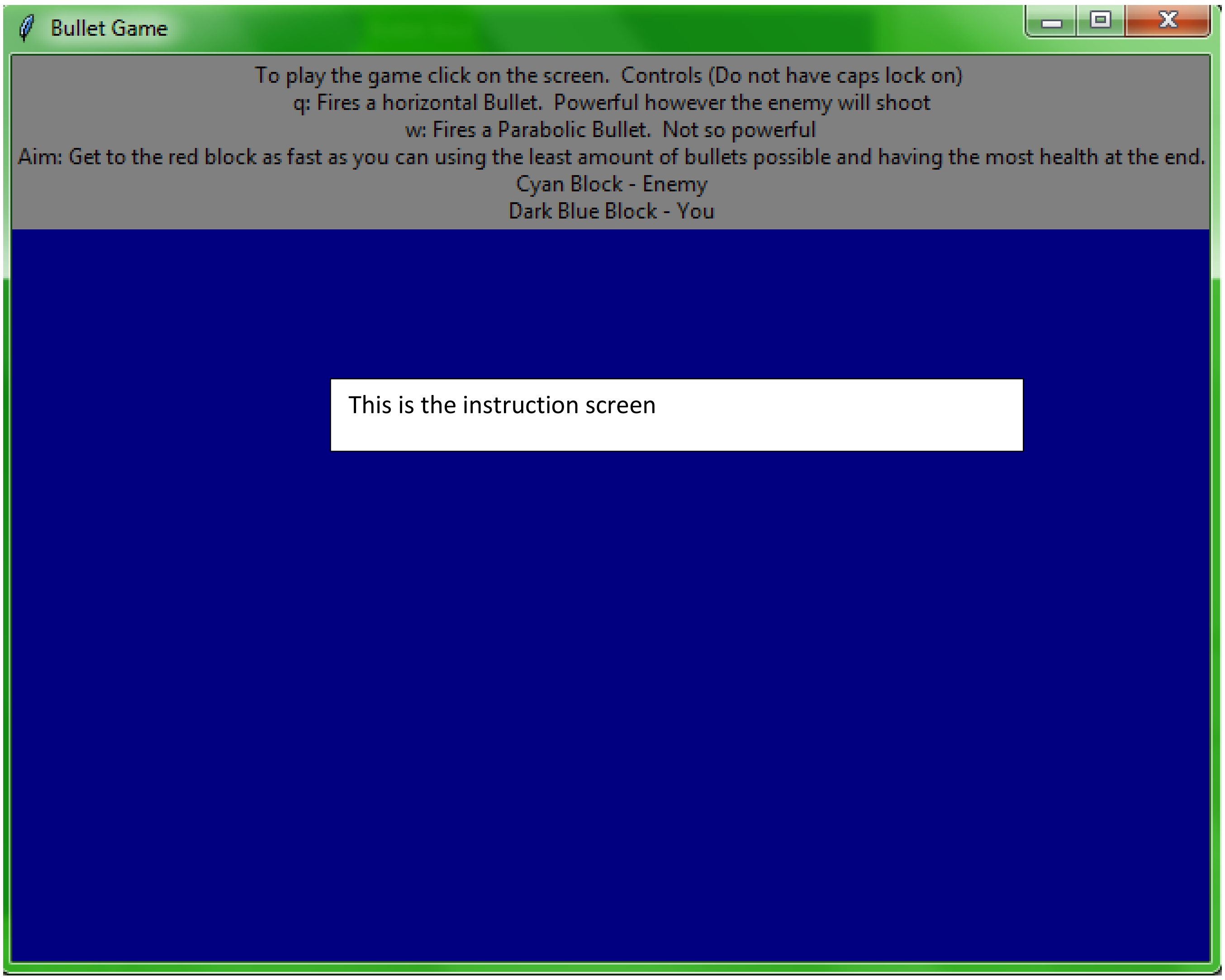
30	The program will end if the user's character hits 0.	This is so there is a risk if you are hit by a bullet	Touch enemy 4 or more times	The program will open the page asking for the username	TRUE	See video2 2:00 – 2:12
31	The program will have a bullet being shot by the enemy every 5 seconds	This is so there is a continuous risk to the user.	-	There is a bullet shooting every 5 seconds	FALSE	Never happens. Instead the enemy fires when the user fires a horizontal bullet
32	The program will allow any characters in the username	This means it's easy for the user to have any name	Valid: hiTh1s @h!	This is accepted	TRUE	See video2 3:35 – 3:50
33	The program will have an instructions screen	This will allow the user to know how to play the game	Press Instructions in menu	Instructions are outputted	TRUE	See video1 0:05 – 0:15
34	The program will have a menu	Allows the user to go to different parts of the program	-	There is a menu	TRUE	See video1 Start
35	The program will have a leaderboard screen	This will allow the user to see the current best scores	Press Leaderboard	This will output the leaderboard	TRUE	See video2 2:40 – 2:45
36	The program will output the leaderboard from the largest score (Best) to the smallest score (worst)	This will make it easy for the user to see the best and worst scores	-	Out of the multiple scores largest is top smallest bottom	TRUE	See video2 2:40 – 2:45
37	The program will take away health if the user's character touches the enemy	This means there is some danger to touching the enemy and means you cannot just run straight to the endpoint	Touch enemy	Hero will lose health	TRUE	See video2 2:10 – 2:12
38	The game will do nothing if you press buttons which are not q,w or the arrow keys.	This means the game cannot break if you enter an accidental input.	Press g	Nothing	TRUE	See video2 3:10 – 3:22


Bullet Game
The bullet being shot by the user (Off screen)


Bullet Game
The user's character
The enemy
The endpoint

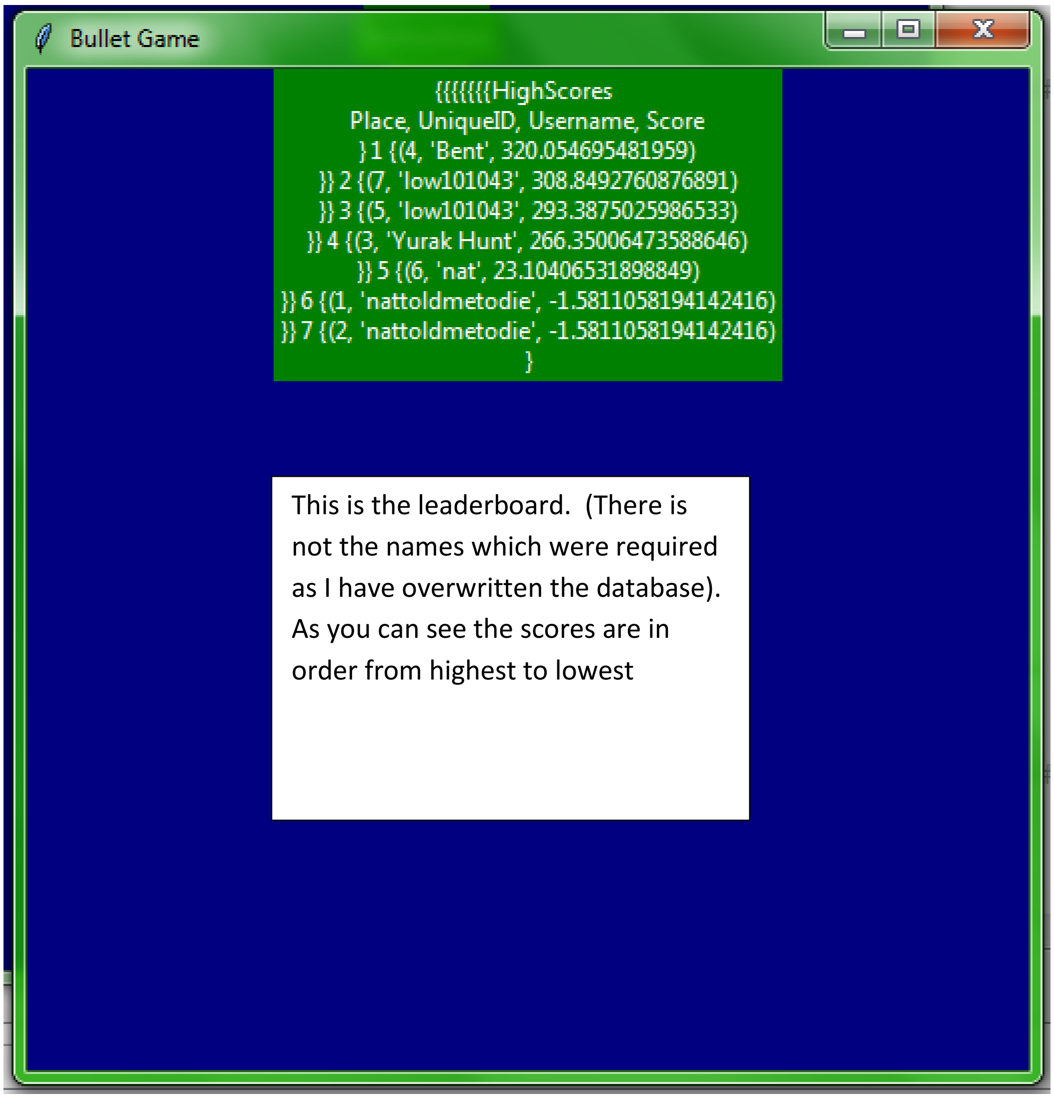


These are the buttons for the leaderboard and is the menu



To play the game click on the screen. Controls (Do not have caps lock on)
q: Fires a horizontal Bullet. Powerful however the enemy will shoot
w: Fires a Parabolic Bullet. Not so powerful
Aim: Get to the red block as fast as you can using the least amount of bullets possible and having the most health at the end.
Cyan Block - Enemy
Dark Blue Block - You

This is the instruction screen



```
{{{{{{{HighScores
Place, UniqueID, Username, Score
}1 {(4, 'Bent', 320.054695481959)
}} 2 {(7, 'low101043', 308.8492760876891)
}} 3 {(5, 'low101043', 293.3875025986533)
}} 4 {(3, 'Yurak Hunt', 266.35006473588646)
}} 5 {(6, 'nat', 23.10406531898849)
}} 6 {(1, 'nattoldmetodie', -1.5811058194142416)
}} 7 {(2, 'nattoldmetodie', -1.5811058194142416)
}
```

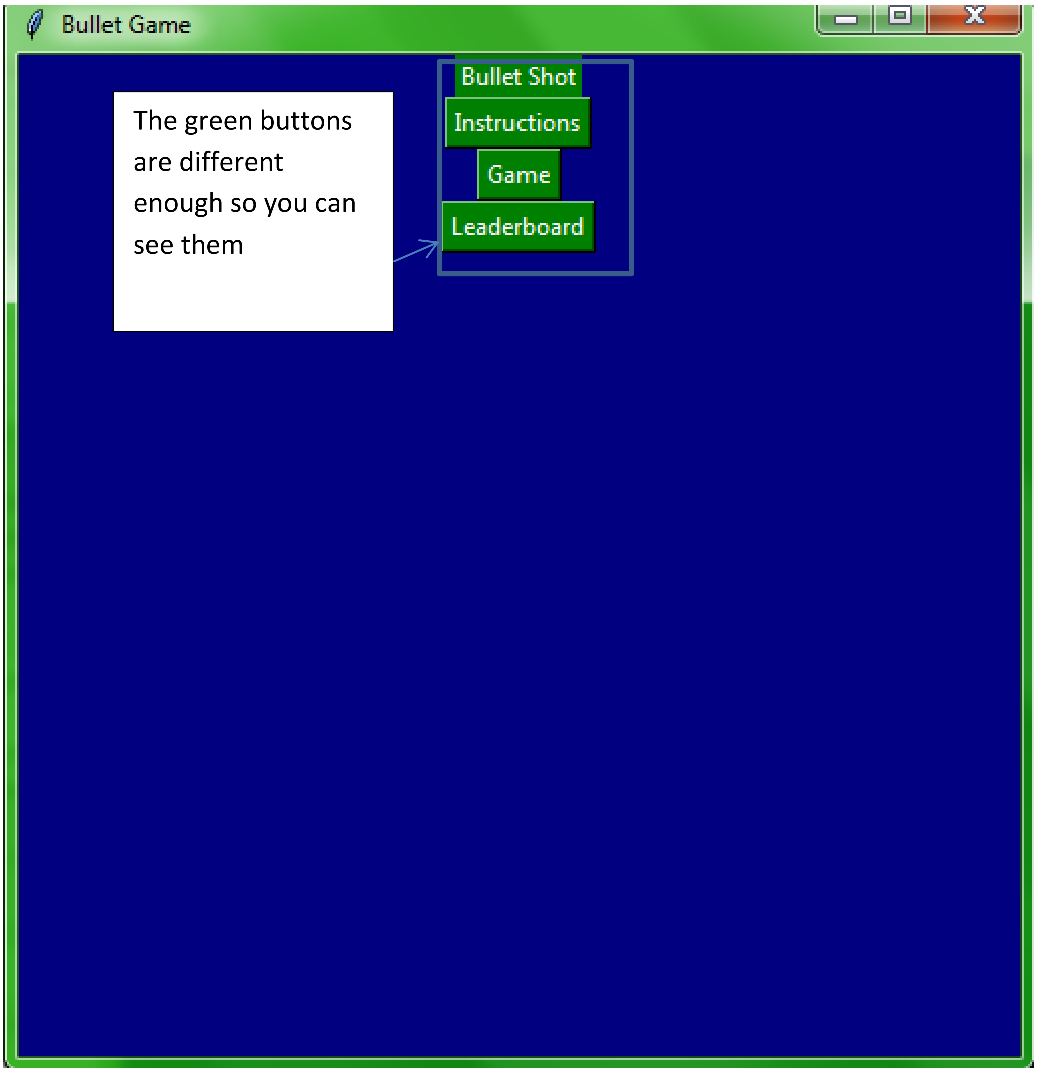
This is the leaderboard. (There is not the names which were required as I have overwritten the database). As you can see the scores are in order from highest to lowest

Provided annotated evidence for usability testing.

Usability Feature	Explanation	Justification	Output	Notes
Different colours for the different objects	All the different objects made will be different colours.	This is done to make the user able to identify the different objects on the screen.	Colours which can be seen by one of my stakeholders	See video1 and video2. See Screenshot 2
High contrast colours	All the different colours will be very different	This makes it easier for mildly colour-blind users see the level	Colours which can be seen by one of my stakeholders	See video1 and video2 and screenshot 1
Instructions	There will be a choice for the instructions to make it easier for the user to know what is happening.	Means the user can know what is happening and helps the user understand how to play the game.	TRUE	See video1 Start. See Screenshot 3
The buttons are close to each other.	This means that I am using buttons which are close to each other so q and w for shooting	This makes it easier for people who have limited mobility meaning they can still play the game.	TRUE All keys are near to each other as all the user will use is w,q and the arrow keys	

See Screenshots below for evidence

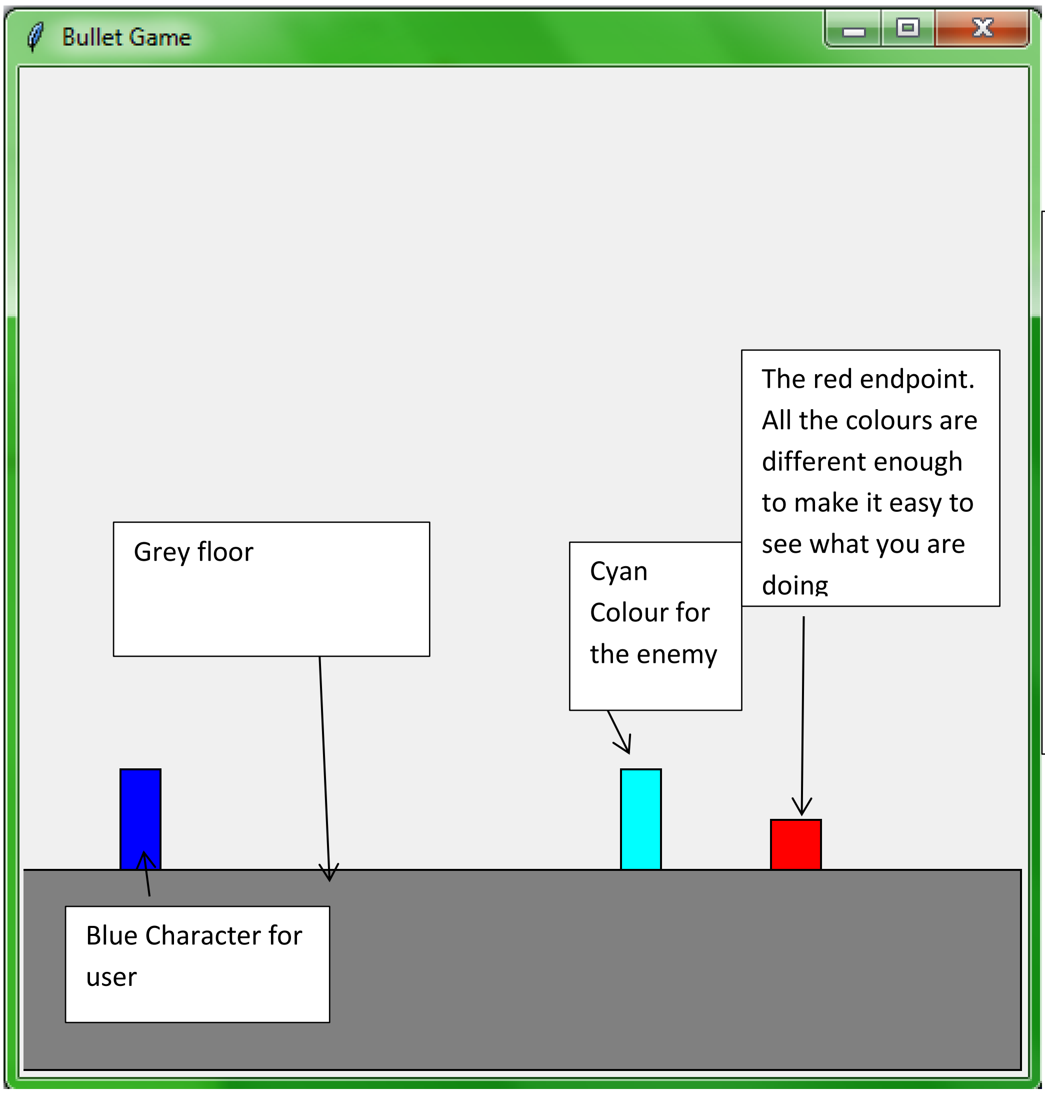
Screenshot 1



The green buttons are different enough so you can see them

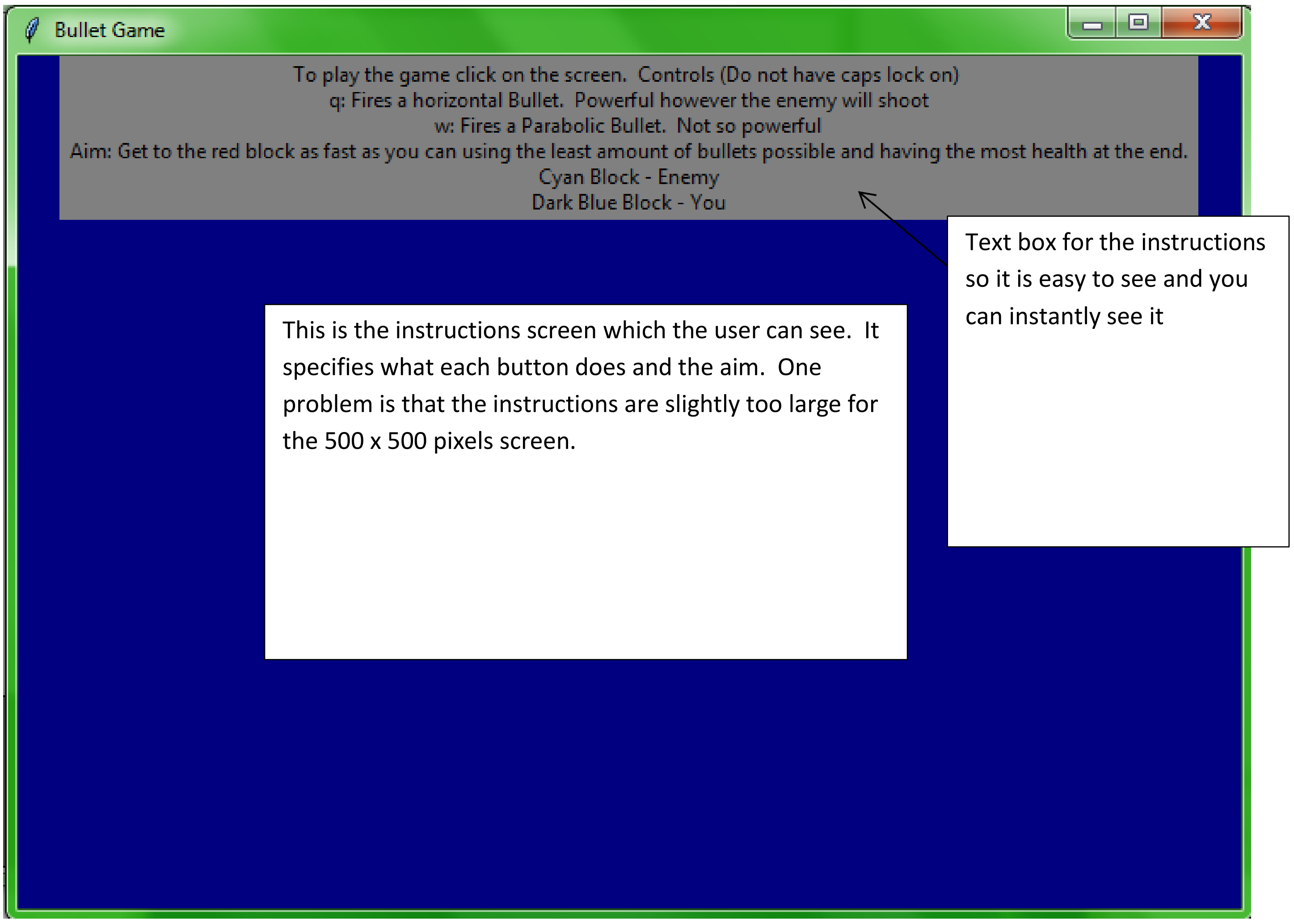
This uses both blue and green and is used for all screens (apart from the game)

Screenshot 2


Bullet Game
Grey floor
Cyan Colour for the enemy
The red endpoint. All the colours are different enough to make it easy to see what you are doing
Blue Character for user

These colours for the objects on the screen have been used as they are all different enough that users can differentiate between them

Screenshot 3



To play the game click on the screen. Controls (Do not have caps lock on)
q: Fires a horizontal Bullet. Powerful however the enemy will shoot
w: Fires a Parabolic Bullet. Not so powerful
Aim: Get to the red block as fast as you can using the least amount of bullets possible and having the most health at the end.
Cyan Block - Enemy
Dark Blue Block - You

Text box for the instructions so it is easy to see and you can instantly see it

This is the instructions screen which the user can see. It specifies what each button does and the aim. One problem is that the instructions are slightly too large for the 500 x 500 pixels screen.

Beta Testing

After my testing, I also allowed my stakeholders and other classmates to play the game as 'beta' testing. I then have recorded some of their reactions to the game.

One Stakeholder (Ben Lowis) commented how he did not like the game because of the graphics thus not making it look professional enough. He commented that the game does not look good compared to games like Fortnite or even games you could find on websites like Coolmaths Games.

Quite a few people mentioned that the leader board was messy and did not look very good. I believe this will be from how I got the data out of the database, thus each record came in its own tuple.

Cameron mentioned that the game was fun and served its purpose. As a fellow classmate in Computer Science, he was impressed (along with the rest of my classmates) by how the game was written purely in Python and not using any game engine.

Finally, all my testers said that the game was fun and enjoyable should be better by having more bullets, enemies, difficulty and graphics!

Use the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria have been fully, partially or not met in each case.

Success Criteria

Statement	Justification	Met	Comment
The program will work. The player will be able to play a game which is realistic and is a shooting 2D platform game with an enemy which you need to defeat in it. The program will be robust enough to not break if a wrong input is entered.	I do not want there to be any game breaking bugs or glitches so I will want to have identified them before I have finished. I want the game to be finished and that the user feels that it is polished and finished.	Partially (Test Number: 1)	The program does work however you can spam the bullets and also cannot go to a different screen from the menu twice (Apart from instructions) because how I am calling the files. Also, the program does not delete the window which you have just left
The program will allow the user to move the character side to side and also to jump.	Again self-explanatory. The user will only be able to use one character, and this will be the way to actually interact with the level.	Met (Test Number: 2, 3, 4)	The character can move left, right and jump (Though jumping does stop everything else!)
The program will have an enemy	This is so there is some difficulty in the game.	Met (Test Number: 5)	There is an enemy which you have to defeat
The program will end when the user's character reaches an endpoint.	This means the level will end at one point and will not go on forever.	Met (Test Number: 6)	The game will end when you reach the point
The program will detect when the bullet hits either a character or the wall.	This will make the bullet more realistic and also mean the bullet does not go for infinity.	Met (Test Number: 8, 9, 10)	There are no walls to hit. Might not always notice the enemy on the parabolic bullet
The program will take away health from a character when a bullet hits it using the power equation.	This makes it realistic and also means you can destroy the enemy/character.	Met (Test Number: 11, 12, 13)	This all works
The program will make the bullet go the correct distance using SUVAT.	This make the game realistic	Met (Test Number 14, 15, 16)	All works
The program will have two bullets a parabolic one and a horizontal one.	This allows the user to have more than one weapon	Met (Test Number: 17)	This works
The program will allow the user to change which bullet to use	This means they can use both bullets.	Partially (Test Number: 7)	How I initially planned that you would use the same button did not happen however now the user just

			have to press one of two buttons
The program will take away bullets from the total amount of bullets when one is fired	This makes it more difficult to the user and gives the game a level of difficulty.	Met (Test Number: 18, 19)	This works however both bullets take off the same number of bullets instead of different amounts
The program will have a timer which allows it to work out how long the user has taken.	This is to be used in the total points and allows the user to try and make it quicker	Met (Test Number: 20)	This works (Might not work on another OS from windows as the time library is different for different Operating Systems
The program will work out a total.	This is to give a point for the game	Met (Test Number: 21)	Works
The program will allow the user to see how well they have done	This will allow the user to see how well they have done	Met (Test Number: 22)	This works
The program will put the score in a database	This is so the user can see how they compare to other people.	Met (Test Number: 24)	This works
The program will have a database of high scores	This is so all scores can be entered and also, so the user can see how well they are doing.	Met (Test Number: 23)	This works
The program will allow the user to enter a username.	This is so it can go into the database along with their score.	Met (Test Number: 25)	This works
The program will allow the user to choose whether to see the high scores or to play the game	This will be a menu and allow the user to see different parts of the program.	Met (Test Number: 27)	Also, can see the instructions
The program will display a table of high scores.	This allows the user to see how well they are doing.	Met (Test Number: 28)	Works
The program will make sure all characters are on the ground when jumping.	This will then make sure it looks realistic and also means you can see how high they are.	Met (Test Number: 29)	This works. However, you can spam the jump button to stay above bullet (Intentional) and stop all bullets
The program will end if the user's character hits 0.	This is so there is a risk if you are hit by a bullet	Met (Test Number: 30)	This works
The program will have a bullet being shot by the enemy every 5 seconds	This is so there is a continuous risk to the user.	Not Met (Test Number: 31)	This was changed to shooting a bullet whenever the user fires a horizontal bullet. This change was made because of how for anything to happen, it requires a user input.
The program will allow any characters in the username	This means it's easy for the user to have any name	Met (Test Number: 32)	This works
The program will only allow usernames which no one else has used before.	This is so it is easy for the User to see their score in the database	Not Met (Test Number: 26)	I could not program this to work. Instead have a unique number for each user

The program will have an instructions screen	This will allow the user to know how to play the game	Met (Test Number: 33)	Instructions could always be expanded
The program will have a menu	Allows the user to go to different parts of the program	Met (Test Number: 34)	This works
The program will have a leaderboard screen	This will allow the user to see the current best scores	Met (Test Number: 35)	However, the leaderboard is messy with curly brackets
The program will output the leaderboard from the largest score (Best) to the smallest score (worst)	This will make it easy for the user to see the best and worst scores	Met (Test Number: 36)	This works
The program will take away health if the user's character touches the enemy	This means there is some danger to touching the enemy and means you cannot just run straight to the endpoint	Met (Test Number: 37)	This works
The game will do nothing if you press buttons which are not q,w or the arrow keys.	This means the game cannot break if you enter an accidental input.	Met (Test Number: 38)	Works

Provide comments on how any partially or unmet criteria could be addressed in further development.

The program will have a bullet being shot by the enemy every 5 seconds	This is so there is a continuous risk to the user.	Not Met (Test Number: 31)	This was changed to shooting a bullet whenever the user fires a horizontal bullet. This change was made because of how for anything to happen, it requires a user input.

This could be changed in further development to try and get a bullet shooting every 5 seconds. I did not do this in my development as it would slow the game even more than it does. Maybe using a different language which is not Python as Python does not handle different threads. Using an actual game engine might also help with this.

The program will only allow usernames which no one else has used before.	This is so it is easy for the User to see their score in the database	Not Met (Test Number: 26)	I could not program this to work. Instead have a unique number for each user

To do this I would have to be able to get better at SQLite3. My inexperience with it meant I could not get the username out in the correct format to look at. How I have it at the moment (Using a unique number), I would keep but I would see if I could get unique usernames as well.

The program will work. The player will be able to play a game which is realistic and is a shooting 2D platform game with an enemy which you need to defeat in it. The program will be robust enough to not break if a wrong input is entered.	I do not want there to be any game breaking bugs or glitches so I will want to have identified them before I have finished. I want the game to be finished and that the user feels that it is polished and finished.	Partially (Test Number: 1)	The program does work however you can spam the bullets and also cannot go to a different screen from the menu twice (Apart from instructions) because how I am calling the files. Also, the program does not delete the window which you have just left

This is down to the time I had to program the code and how easy it is to get small bugs into my code. To improve, I would need more time to iron out some of the bugs. However, none was too game-breaking, and the game still works at all times even if you enter a wrong input.

The program will allow the user to change which bullet to use	This means they can use both bullets.	Partially (Test Number: 7)	How I initially planned that you would use the same button did not happen however now the user just have to press one of two buttons

My actual solution to this problem is better than using the same button with it switching with another as it is faster for the user to use two different buttons.

Provide evidence of the usability features justifying their success, partial success or failure as effective usability features.

The evidence is in the section of ‘Providing annotated evidence for usablilty testing’

Usability Feature	Explanation	Justification	Output	Justifying success or not as effective usability features
Different colours for the different objects	All the different objects made will be different colours.	This is done to make the user able to identify the different objects on the screen.	Colours which can be seen by one of my stakeholders	This was a success as all the colours could be seen but all were different enough that anyone could
High contrast colours	All the different colours will be very different	This makes it easier for mildly colour-blind users see the level	Colours which can be seen by one of my stakeholders	Success as one of my stakeholders could see the different colours in both the menus and game
Instructions	There will be a choice for the instructions to make it easier for the user to know what is happening.	Means the user can know what is happening and helps the user understand how to play the game.	TRUE	This is a partial success. Could see it but probably not detailed enough
The buttons are close to each other.	This means that I am using buttons which are close to each other so q and w for shooting	This makes it easier for people who have limited mobility meaning they can still play the game.	TRUE All keys are near to each other as all the user will use is w,q and the arrow keys	Success. All buttons are either in top left of keyboard (q, w) or the arrow keys

Provided comments on how any issues with partially or unmet usability features could be addressed in further development.

Usability Feature	Explanation	Justification	How to improve
High contrast colours	All the different colours will be very different	This makes it easier for mildly colour-blind users see the level	Maybe make the colours a bit more vivid and use colours with more association (So red for enemy, grey for the bullet) to make it easier for the user
Instructions	There will be a choice for the instructions to make it easier for the user to know what is happening.	Means the user can know what is happening and helps the user understand how to play the game.	To improve this, I could use the colours in the game to tell which character is which (So red for enemy as red equals bad). Also, probably mention more of the complex parts of my project (So you can move whilst jumping)
The buttons are close to each other.	This means that I am using buttons which are close to each other so q and w for shooting	This makes it easier for people who have limited mobility meaning they can still play the game.	The problem with this is that on a larger keyboard you lose a lot of the accessibility you had

Considered maintenance issues and limitations of the solution.

Maintenance

- The length of the code – This is a problem. As the code is over 1200 lines over 4 or 5 different files maintaining the code will be a difficult. One change can trickle down the whole program.
- Use of Global Variables – I have used too many global variables than reasonable. The main reason for this is because of how Tkinter works and that it is very hard to send some variables to the right function when you press a button. To improve this, I would consider using a different GUI (More on this later).
- The 'flow' of the code – The code does not run that smoothly. This is easiest to see when firing a bullet (There are 3 functions/methods for each bullet to be shot) meaning it is quite hard to see where the code should go. This will make it hard to maintain as you might not entirely know where to go
- How the different files are imported– How the files ared imported is not very good. When a file is imported, it runs the whole code making it hard for the next programmer to know entirely where the code goes next.
- Later updates – If Tkinter is updated and you lose some of the commands it would mean that the whole code might not work. As well, if you try and update the code to a later version of Python you might find it hard if Python or Tkinter has been significantly changed. Finally, you would have to test the code on all different OS systems you use just in case Tkinter or Python does not work on it.

Limitations

- Length of code – Again this is a problem. At 900 lines my largest program makes Thonny very slow. This means that it will take a while to actually program and hard for it to be completely readable
- Python and Tkinter – The limitations here are firstly Tkinter cannot produce very good-looking graphics (Blocks do not look very good) secondly Python has no way to do threading. It has meant that I could not get the enemy character to fire a bullet automatically as the user would not be able to move. To improve this, I should have used PyGame or an actual game engine
- Threading – A consequence of there being no threading in the game is that whenever there is a user input, all other objects on the screen freezes (Most notable when you shoot 2 bullets – One of them stops).
- How hard the game is – If the user knows how to beat the game (IE when you jump it will stop the enemy bullet and you can go over it, Get close to the enemy and spam the parabolic bullets will kill it quickly) the game is incredibly easy. The game could use more than one enemy
- Parabolic arc – As I did the SUVAT equations using time as a constant it has meant that the parabolic arc is very hard to use. As it will barely hit the enemy the user either needs to be right next to the enemy or in the perfect spot to be able to use it. This would need to be improved next time.
- Importing of files which I have created – When the main program calls code from another file which I have written, the program wil import it then start running that files code immediately. This then means that the main program cannot use the file it have just imported again (So the user could not play two games when you run the program once).

Describe how the program could be developed to deal with limitations and potential improvements / changes.

A lot of my problems with the program, is down to the language I have used and also not using a game engine. Even using something like PyGame would have improved the game hugely as this will allow me to have better graphics and would probably help with the bullets moving. However, using a game engine would help even more. The reason I did not use a game engine was I might accidently take advantage of the physics engine, however the benefits of using a collision detector would outweigh the negatives. Also, I could then use more realistic physics as the engine would be able to do the maths more easily. It might also mean I would not have to have different files with different code

Another problem was how my bullet classes were written. As I did not know how to override the constructor method the parent class Bullet had all the attributes. Now I know how to override the constructor class I would not have all attributes in the parent class and have it more specific.

Another thing I would improve on would be my database. At the moment when outputted there are loads of curly brackets which I would try and take out and also, I would try and get the user to have a unique username instead of a unique ID. As it was an idea I was unsure I would have enough time to program, I am happy enough of how it ended. As well, as it was not in the 'main' part of the project (the Game), it did not have any usability testing. This would be an immediate change if I was going to improve the project.

If I kept on using Tkinter I would do more research to learn how to delete windows when you go onto a new window. This would make it look more professional. I would also learn how to make Tkinter look less like there were just batch colours and more professional.

Another huge improvement would be how I do the importing of different files. As mentioned in my limitations you can only import a file once, meaning you can only run the game once when you run the program. This yet again would be something I would like to change by looking to have all the code in one file or to improve how the program calls a different file.

Another problem was how I wrote my code. A lot of my code basically is just reusing different parts just changing it ever so slightly. There is also repeated code. To improve this, I would put a few more into their own functions/procedures and also see if I could generalise a few more functions. An example of this would be refactoring the code by combining the functions which fire the horizontal bullets for the user and the enemy by making it more general. This is so it is smaller and easier to read through.

Finally, I would improve on would be trying to find a better GUI than Tkinter. As I am using Tkinter, I have a lot of limitations with me having to have a lot of 'try except' statements (which do not help the flow of the code) and global variables (Which are hard to use to know what is global and what is not). If I could use a better GUI (like if it comes pre-packaged with a game engine/physics engine) that would help as it means the code is easier to read. Another problem from my inexperience with Tkinter was that the user has to manually shut down a screen even if it should be completely deleted already. This would be something I would try to improve to make it easier for the user to run the game.

Final Code

adding_To_Database.py

```
from tkinter import *   #These commands will import external libraries which are needed for this file .  Tkinter - GUI

import sqlite3   #See above . sqlite3 - Database

import csv   #See above csv -holds score from other file

########################################################################################################################################################################################################################################################

def checker():

   """Made By Nathaniel Lowis First Edit: 17/10/18 Latest Edit: 18/10/18
     Inputs - None
     Outputs - largest ID (Integer)
     This will work out what the latest Unique ID is in the Database"""

   connNew = sqlite3.connect("highscores.db") #This will connect to the database and set it up
   curser = connNew.cursor()

   largest = 0   #This will be the largest unique ID we have
   curser.execute('SELECT UniqueID FROM Highscores')   #gets all the Primary keys from the table (Will be in a list of tuples)
   uniqueID = curser.fetchall()  #Lets me use the primary keys

   for idRecord in uniqueID:   #Checks every primary key
      if idRecord[0] > largest:  #If the primary key is larger than the largest at the moment
         largest = idRecord[0]  #We will get largest to equal it

   connNew.close()  #Shuts the database

   return largest  #Returns it to the main program



   #print(curser.fetchall())  #Testing

########################################################################################################################################################################################################################################################

def username_Entering():

   """Made By Nathaniel Lowis.  First Edit: 17/10/18 Latest Edit: 18/10/18
     Inputs - None (Do get score (Float) and username (string) from csv file and gui though)
     Outputs - None
     This will allow the username to be entered and for the score to be taken out of the csv file and added to the database"""
   #finalScore = 5
   with open("Scores.csv", mode= "rt", encoding = "utf-8") as readingScores:   #Opens the csv file as a reader

      reader = csv.reader(readingScores)  #Sets up a reader

      for recordToBeRead in reader: #Checks all records in the csv file (should only be 1)

         if recordToBeRead == []:  #If the record is empty (EG the 2nd one) the code should do nothing
            pass

         else:   #If the record has something in it
            finalScoreDatabaseFile = recordToBeRead[0]  #We allow the program to access it so it can be used

   username = addingUserName.get()  #This will get the username from the screen
   uniqueIDFunction = checker()  #This will get the largest uniqueID so far in the database

   newID = uniqueIDFunction + 1   #This will add 1 to the last UniqueID to make the primary key for the code

   labelChange.configure(text = "Your score is being added to the database.  Your score was
{}".format(finalScoreDatabaseFile))  #This will output the score and saying everything is being added to the database
   adding_To_Database(newID, username, finalScoreDatabaseFile)  #Adds everything to the database

##################################################################################################
##################################################################################################
#########################################

def adding_To_Database(uniqueIDToUse, usernameToAdd, scores):

   """Made By Nathaniel Lowis  First Edit: 18/10/18  Latest Edit: 18/10/18
     Inputs - uniqueIDToUse (Integer), usernameToAdd (string), scores (Float)
     Outputs - None
     This will add the user's username and score to the database"""

   connectionNew = sqlite3.connect("highscores.db")  #Opens the database and sets it up to be used
   curserNew = connectionNew.cursor()

   toAdd = [(uniqueIDToUse, usernameToAdd, scores)]  #This is all the data to be added to the database in a way
which the database will allow

   curserNew.executemany('INSERT INTO Highscores VALUES (?,?, ?)', toAdd)  #Adds the data into the database.
Done like this so it adds everything using variable names
   connectionNew.commit()  #Commits (Saves) it to the database
   connectionNew.close()  #Closes the database

##################################################################################################
##################################################################################################
#########################################

#main
databaseScreen = Tk()  #Makes a screen

databaseScreen.title("Bullet Game")  #The name of the screen
databaseScreen.geometry("500x500")  #The size

#menu.wm_iconbitmap("favicon.ico")  #If I wanted to change the image for the file
```

```
databaseScreen.configure(background = ("navy"))  #Sets the background

addingUserName = Entry(databaseScreen)  #Adds an entry which the user will enter their username
addingUserName.pack()  #Sends to the screen

labelChange = Label(databaseScreen, text = "Enter username", bg = "green", fg = "white")  #Sets a label which will
say what is going on
labelChange.pack()  #Sends to the screen

buttonAskingForUserName = Button(databaseScreen, text = "Enter Username!", bg = "green", fg = "white",
command = username_Entering)  #The button which will send the username off
buttonAskingForUserName.pack()  #Sends to the screen

databaseScreen.mainloop()  #Infinite loop needed to allow GUI to work.
```

database_Creator.py

```
import sqlite3  #Imports an external library which will allow me to work with databases

##########################################################################################################
##########################################################################################################
#########################################

def create_Table(name):

   """Made By Nathaniel Lowis.  Edited from: https://repl.it/@NathanielLowis/Database-in-python  First Edit:
17/10/18.  Latest Edit: 18/10/18
      Inputs - Name of the filename (String)
      Outputs - None
      This will set up a database"""

   connection = sqlite3.connect(name)  #Connects to a database and allow you to edit it
   cursor = connection.cursor()

   cursor.execute('CREATE TABLE Highscores ("UniqueID" integer,"UserName" text, "Score" real)')  #This will make all
the columns for the database

   connection.commit()  #Saves it and closes the database
   connection.close()

##########################################################################################################
##########################################################################################################
#########################################

#main
create_Table("highscores.db")  #This creates a database
```

displaying_Database.py

```
import sqlite3   #Imports all the libraries needed for this file

from tkinter import *  #Imports everything from the library Tkinter

##########################################################################################################
##########################################################################################################
#########################################

def display_Database():
   """Made By Nathaniel Lowis  Edited from: https://repl.it/@NathanielLowis/Database-in-python  1st Edit: 
18/10/18.  Latest Edit: 18/10/18
      Inputs - None
      Outputs - A sorted table of high scores (list of tuples)
      Gets data from database and sorts it and sends it to the main program"""

   connectionDisplay = sqlite3.connect("highscores.db")  #Connects to the database and sets it up to be accessed
   cursorDisplay = connectionDisplay.cursor()

   cursorDisplay.execute('SELECT * FROM Highscores')  #This will get all records from the database
   table = cursorDisplay.fetchall()  #Allows the program to access it

   sortedTable = quick_Sort(table)  #This will sort out the database



   connectionDisplay.close()  #Closes the database
   return(sortedTable)  #Returns the sorted database to the main program

##########################################################################################################
##########################################################################################################
############################################

def quick_Sort(dataArray):

   """Made By Nathaniel Lowis  Edited from: https://repl.it/@NathanielLowis/Quick-Sort  First Edit: 18/10/18  Latest 
Edit: 18/10/18
      Inputs - dataArray (List of tuples)
      Outputs - dataArray (List of tuples)
      Sorts the data out on order of the scores."""

   #print("running quick sort\n")
   if len(dataArray) == 1 or len(dataArray) == 0:                    # base case for the recursive call
      return dataArray
   else:
      pivot = dataArray[0][2]         # using first value as the pivot value
      i = 0

      for j in range(len(dataArray)-1):                               # rearranging values around  pivot
         if dataArray[j+1][2] > pivot:
            dataArray[j+1],dataArray[i+1] = dataArray[i+1], dataArray[j+1]
            i = i + 1
      dataArray[0],dataArray[i] = dataArray[i],dataArray[0]

      #print("dataArray sorted either side of pivot", dataArray,"pivot", pivot)

      first_part = quick_Sort(dataArray[:i])                          # recursive calls on either side of pivot
      second_part = quick_Sort(dataArray[i+1:])
      first_part.append(dataArray[i])                                 # put pivot in correct position

      #print("first part",first_part, "second part", second_part,"dataArray", dataArray)
      return first_part + second_part
```

```
##########################################################################################################################################################################################################################################

def make_Text():

    """Made By Nathaniel Lowis 1st Edit: 18/10/18  Latest Edit: 18/10/18
      Inputs - None
      Outputs - Text to display (String)
      This creates the table needed to be displayed"""

    tableToDisplay = display_Database()  #Gets the database table

    tableDisplay = "HighScores \n Place, UniqueID, Username, Score \n"  #What the user will see
    place = 0  #The position in the table#

    for recordToDisplay in tableToDisplay:  #Takes each record in the array

        place = place + 1  #Increments place to the correct position
        placeString = str(place)  #Turns it into a string

        recordToDisplayStr = str(recordToDisplay) #Turns the tuple into a string so it can be added to the final string

        tableDisplay = tableDisplay, placeString,  recordToDisplayStr + "\n"  #Adds all the data to the main table so all
the scores will be on it

    return tableDisplay

##########################################################################################################################################################################################################################################

#main
databaseShowScreen = Tk()  #Makes Window
databaseShowScreen.title("Bullet Game")  #Names Window
databaseShowScreen.geometry("500x500")  #The size of the window

#window.wm_iconbitmap("favicon.ico")
databaseShowScreen.configure(background = ("navy"))  #The background colour

displayTable = make_Text()  #Makes the table to be displayed

labelWithTable = Label(databaseShowScreen, text = displayTable, bg = "green", fg = "white")  #Adds the label with
the scores on it
labelWithTable.pack()  #Packs label to screen

databaseShowScreen.mainloop()  #Infinite loop
```

final_Game_File.py

```
# Main Game

from tkinter import *     #This means I can use Tkinter
#import gamePrototype4
```

```
##########################################################################################################################################

def instructions():

    """Made By Nathaniel Lowis.  1st Edit: 17/10/18.  Latest Edit: 17/10/18
      Inputs - None
      Outputs - None
      Displays the instructions for how to play the game"""

    instructionScreen = Tk()   #Makes a window
    instructionScreen.title("Bullet Game")  #Name of the window
    instructionScreen.geometry("500x500")  #Size of the window

    #menu.wm_iconbitmap("favicon.ico")
    instructionScreen.configure(background = ("navy"))  #Background of the window

    instructionsToDo= Label(instructionScreen, text = "To play the game click on the screen.  Controls (Do not have caps lock on) \n q: Fires a horizontal Bullet.  Powerful however the enemy will shoot \n w: Fires a Parabolic Bullet.  Not so powerful \n Aim: Get to the red block as fast as you can using the least amount of bullets possible and having the most health at the end. \nCyan Block - Enemy\nDark Blue Block - You", bg = "grey", fg = "black")  #This will output all the instructions onto the window so the user can see how to play the game
    instructionsToDo.pack()  #Sends the label to the screen

##########################################################################################################################################

def play_Game():

    """Made By Nathaniel Lowis 1st Edit: 17/10/18  Latest Edit: 17/10/18
      Inputs - None
      Outputs - None
      Plays the game"""

    import main_Game  #Goes to file named gamePrototype4 and runs it.  This will run the game
    #exec(open("gamePrototype4.py").read())  #Testing

##########################################################################################################################################

def database_To_Display():

    """Made By Nathaniel Lowis 1st Edit: 18/10/18  Latest Edit: 18/10/18
      Inputs - None
      Outputs - None
      Will go to the file which ouputs the leaderboard"""

    import displaying_Database  #Goes to file called displaying_Database and runs the code

##########################################################################################################################################
```

```
#main
window = Tk()  #Makes a window for the GUI

window.title("Bullet Game") #Names the window
window.geometry("500x500")  #The size of the window

#window.wm_iconbitmap("favicon.ico")
window.configure(background = ("navy"))  #Sets the background colour


labelWelcome =Label(window, text = "Bullet Shot", bg = "green", fg = "white")  #This is the welcome label saying the
name of the game (Bullet Shot)
labelWelcome.pack()  #Sends it to the main screen


buttonInstructions = Button(window, text = "Instructions", bg = "green", fg = "white", command = instructions )
#This is the button to Go to the instructions
buttonInstructions.pack()  #Sends button to window

buttonGame = Button(window, text = "Game", bg = "green", fg = "white", command = play_Game)  #This is the
button to play the game
buttonGame.pack()  #Sends button to the screen

buttonLeaderboard = Button(window, text = "Leaderboard", bg = "green", fg = "white", command =
database_To_Display)  #This is the button to go to the leaderboard
buttonLeaderboard.pack()  #Sends button to the screen

window.mainloop()  #Infinte loop.
```

main_Game.py

```
from tkinter import *     #This means I can use Tkinter

import time   #Allows me to use time and measure how long bullets

import csv  #This allows me to use the library csv

##################################################################################################
##################################################################################################
########################################

class BulletsShot():

    """Made by Nathaniel Lowis 1st Edit: 17/10/18 Latest Edit:17/10/18
      This will work out how many bullets you have in a game"""

##################################################################################################
##################################################################################################
########################################

    def __init__(self):

        """Made by Nathaniel Lowis 1st Edit: 17/9/18  Latest Edit: 17/10/18
          Inputs - None
          Outputs - None
          Sets up the class"""

        self.howManyShot = 10  #How many shots the person will have

####################################################################################################
####################################################################################################
##########################################

    def bullets_Sub(self):

        """Made by Nathaniel Lowis 1st Edit: 17/9/18  Latest Edit: 17/10/18
          Inputs - number of bullets left (Integer)
          Outputs - None
          Takes bullets off when you shoot one"""

        self.howManyShot = self.howManyShot - 1  #Will take away a bullet when you shoot one

####################################################################################################
####################################################################################################
##########################################

    def bullets_Show(self):

        """Made by Nathaniel Lowis 1st Edit: 17/9/18  Latest Edit: 17/10/18
          Inputs - None
          Outputs - How many bullets you have (integer)
          Shows how many bullets you have"""

        return self.howManyShot  #Returns how many bullets you have to the main program

####################################################################################################
####################################################################################################
##########################################
####################################################################################################
####################################################################################################
##########################################

class Bullet():

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 3/10/18
      This is the superclass which will allow me to model how the bullet moves and output it to the screen.  Parts of
code has been adapted from:https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas
      """

####################################################################################################
####################################################################################################
##########################################

    def __init__(self, x1BulletGiven, y1BulletGiven, x2BulletGiven, y2BulletGiven, canvasToUse):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 3/10/18
          Inputs - the canvas (Tkinter Object), coordinates (Integers)
          Outputs - None
          Sets up the class when initialised"""
        self.intialSpeedHori = 50.0  #These will set up the intial speed, acceleration for use in the rest of the code for
both the Parabolic bullet and Horizontal Bullet
        self.intialSpeedHoriPara = 50.0

        self.finalSpeedHori = 0.0
        self.finalSpeedVertical = 50.0
        self.finalSpeed = 0.0

        self.distanceHori = 0.0
        self.distanceVertical = 0.0

        self.horizontalAcceleration = 30.0
        self.verticalAccelerationUp = -9.81
        self.verticalAccelerationDown = 9.81

        self.x1Bullet = x1BulletGiven                    #These 4 co-ordinates are where each corner should be placed
        self.y1Bullet = y1BulletGiven
        self.x2Bullet = x2BulletGiven
        self.y2Bullet = y2BulletGiven

        self.canvasBullet = canvasToUse  #Sets up the code

        self.ball = canvasToUse.create_oval(self.x1Bullet, self.y1Bullet, self.x2Bullet, self.y2Bullet, fill="green")  #Outputs
to screen
        self.time = 5       #More constants made.
        self.mass = 0.008
        self.energy = 0.0

####################################################################################################
####################################################################################################
##########################################

    def delete_Bullet(self):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - The class
          Outputs - Deletes the bullet from the screen
          Deletes the bullet from the screen"""

        self.canvasBullet.delete(self.ball)  #Deletes bullet

####################################################################################################
####################################################################################################
##########################################

    def energy_In_Bullet(self):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Class
          Outputs - The energy of the bullet (float)
          Works out the amount of energy in the bullet"""

        self.energy = .5 * self.mass * ( self.finalSpeed ** 2)   #Uses E = 1/2 mv^2
        ##print(self.energy)    #testing
```

```
##################################################################################################
##################################################################################################
#########################################

    def power_At_Point(self, timeTaken):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs - Time (float), class
          Outputs - The amount of power from the bullet (float)
          Works out the amount of power in the bullet"""

        power = self.energy / timeTaken  #Uses p = E/t

        ##print(power)   #Testing

        return power

##################################################################################################
##################################################################################################
#########################################

    def coord_Bullet(self):

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 12/9/18
          Inputs  - The class
          Outputs - Where the bullet is (Array of floats)
          This will show where the bullet is"""

        coordinatesBullet = self.canvasBullet.coords(self.ball)  #This gives the coordinates for the bullet
        return coordinatesBullet  #Returns it to the main program

##################################################################################################
##################################################################################################
#########################################
##################################################################################################
##################################################################################################
#########################################

class HoriBullet(Bullet):

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      This is a subclass of bullet which calculates and output it to the screen."""

##################################################################################################
##################################################################################################
#########################################

    def speed_At_Any_Point(self, distanceGone):

        """Made by Nathaniel Lowis 1st Edit: 12/9/18  Latest Edit: 12/9/18
          Inputs -  Distance Gone (Float), class
          Outputs - The final speed of bullet (Float)
          This works out the final speed of the bullet"""
        self.finalSpeedHori = ((self.intialSpeedHori ** 2) + (2 * self.horizontalAcceleration * distanceGone) )** .5   #Uses 
V^2 = u^2 + (2as)
        ##print(self.FinalSpeed)   #Testing

        self.finalSpeed = self.finalSpeedHori  #This means the program can use it later

        return self.finalSpeed  #Returns the speed

##########################################################################################################################################

    def move_Bullet_Hori(self, xMovement):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Input - How much to move by (Float)
          Output - Moving on the screen (Tkinter Object)
          This will move the bullet on the screen"""

        self.canvasBullet.move(self.ball, xMovement, 0)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen

##########################################################################################################################################

    def distance_To_Work_Out(self):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - class
          Outputs - The distance to go (Float)
          This will work out how far the bullet has to go"""

        self.distanceHori = (self.intialSpeedHori * self.time) + (.5 * self.horizontalAcceleration *(self.time **2)) #Uses S = 
ut + 1/2at^2
        return self.distanceHori

##########################################################################################################################################
##########################################################################################################################################

class ParabolicBullet(Bullet):

    """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
      This is a subclass of Bullet which works out the stuff for a parabolic Bullet"""

##########################################################################################################################################

    def distance_Worker_Up(self):
        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Vertical Distance (Float), Horizontal Distance (Float)
          This will work out how far the bullet will have to move when the bullet moves upwards"""

        self.distanceVertical = (-.5 * self.verticalAccelerationUp * (self.time /2)**2)  # Uses S = vt - (1/2)at^2
        self.distanceHori = (self.intialSpeedHoriPara * (self.time /2))  #Uses S = ut
        ##print(self.distanceVertical, self.distanceHori)  #Testing
        return self.distanceVertical, self.distanceHori

#####################################################################################################
#####################################################################################################
##########################################

    def distance_Worker_Down(self):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Vertical Distance (Float), Horizontal Distance (Float)
          This will work out how far the bullet will have to move when the bullet moves downwards"""

        self.distanceVertical = (.5 * self.verticalAccelerationDown * (self.time /2)**2)    #Uses S = ut + (1/2)at^2
        self.distanceHori = (self.intialSpeedHoriPara * (self.time / 2))    #Uses S = ut
        ##print(self.distanceVertical, self.distanceHori)  #testing
        return self.distanceVertical, self.distanceHori

#####################################################################################################
#####################################################################################################
##########################################

    def speed_Vert_Up(self, distanceVertUp, timeTakenSoFar):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class, Distance Gone Vertically Up (Float), time (Float)
          Outputs - The final Vertical speed (Float)
          This will work out the final vertical speed of the bullet when the bullet goes upwards"""

        self.finalSpeedVertical = (distanceVertUp + (.5 * self.verticalAccelerationUp * (timeTakenSoFar ** 2)))/
timeTakenSoFar  #Uses S = vt - .5at^2

        if self.finalSpeedVertical < 0:  #If the speed is worked to be less than 0 it sets the final speed to 0
            self.finalSpeedVertical = 0.0

        ##print(self.finalSpeedVertical)  #Testing
        return self.finalSpeedVertical

#####################################################################################################
#####################################################################################################
##########################################

    def speed_Vert_Down(self, timeTaken):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class, time (Float)
          Outputs - Final Vertical Speed (Float)
          This will work out the final vertical speed of the bullet when the bullet goes upwards"""

        self.finalSpeedVertical = self.verticalAccelerationDown * timeTaken  #Uses v = u + at and u is assumed to be 0
        ##print(self.finalSpeedVertical) #Testing
        return self.finalSpeedVertical

####################################################################################################
####################################################################################################
##########################################

    def final_Speed_Parabolic(self):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 26/9/18
          Inputs - Class
          Outputs - Final Overall Speed (Float)
          This uses vector working to work out the final speed of a parabolic bullet"""

        self.finalSpeed = ((self.finalSpeedVertical  **2) + (self.intialSpeedHoriPara **2))**.5  #Uses Pythagorous
Theorem to resolve into 1 vector
        ##print(self.finalSpeed)  #Testing
        return self.finalSpeed

####################################################################################################
####################################################################################################
##########################################

    def move_Ball_Para_Up(self, yMovement, xMovement):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 27/9/18
          Input - yMovement (Float), xMovement (Float)
          Output - Moving on the screen (Tkinter Object)
          Moves the bullet on the screen"""

        self.canvasBullet.move(self.ball, xMovement, -yMovement)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen

####################################################################################################
####################################################################################################
##########################################

    def move_Ball_Para_Down(self, yMovement2, xMovement2):

        """Made by Nathaniel Lowis 1st Edit: 26/9/18  Latest Edit: 27/9/18
          Input - How much to move by
          Output - Moving on the screen
          Works out how far the bullet has to be moved """

        self.canvasBullet.move(self.ball, xMovement2, yMovement2)   #Moves the bullet
        self.canvasBullet.after(100)  #Waits 100 ms until it updates screen
        self.canvasBullet.update()  #Updates the screen
```

```
################################################################################################
################################################################################################
########################################
################################################################################################
################################################################################################
########################################

class Character():

  """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 17/10/18
    This class will make the people on the screen and will allow the computer/user use the character"""

################################################################################################
################################################################################################
########################################

  def __init__(self, canvasCharacter, x0Given, x1Given, y0Given, y1Given, colour):

    """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 31/08/18
      Inputs - The canvas (Tkinter Object), and coordinates (Integers)
      Outputs - Character on the screen
      This will initialise the character meaning it can be used"""

    self.characterOnScreen = canvasCharacter.create_rectangle(x0Given, y0Given, x1Given, y1Given, fill= colour)
#Creates the character
    self.canvasForCharacter = canvasCharacter  #Allows programmer to still use the screen

    self.health = 200 #Sets the health for the character

################################################################################################
################################################################################################
########################################

  def health_Left(self):

    """Made By Nathaniel Lowis 1st Edit: 17/10/18  Latest Edit: 17/10/18
      Inputs - Class
      Outputs - health (Float)
      This will output the health, so the program can use it for scoring"""

    return self.health  #Returns the health

################################################################################################
################################################################################################
########################################

  def jump(self):

    """Made By Nathaniel Lowis 1st Edit: 25/9/18 Latest Edit: 3/10/18
      Inputs - Class
      Outputs - None
      This will allow the character to jump and it be shown on the screen"""

    height = 0  #Sets how high the character is
```

```
      while height <= 50:  #Whilst the character has not reached the maximum height (50)

         self.canvasForCharacter.move(self.characterOnScreen, 0, -1)#Moves the character up
         self.canvasForCharacter.update()  #This will update the screen, so the user can see it
         self.canvasForCharacter.after(10)  #Waits 10 ms until running the program.  Allows the user to move here I
think
         height = height + 1  #Increments height by 1

      downHeight = 0  #Sets how high the character  Needed to get the character down

      while downHeight <=50:  #Whilst the character has not reached the floor (Need to go down 50 pixels)

         self.canvasForCharacter.move(self.characterOnScreen, 0, 1)#Moves the character
         self.canvasForCharacter.update()  #This will update the screen, so the user can see it
         self.canvasForCharacter.after(10)  #Waits 10ms until it does the anything
         downHeight = downHeight + 1  #Increments downHeight by 1

##########################################################################################################
##########################################################################################################
###########################################

   def delete_Character(self):

      """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
        Input - Character Class
        Outputs - Deletes the character
        Deletes character if they are killed"""

      self.canvasForCharacter.delete(self.characterOnScreen)  #Deletes the character

##########################################################################################################
##########################################################################################################
###########################################

   def lose_Health(self, healthToLose):

      """Made by Nathaniel Lowis 1st Edit: 11/9/18  Latest Edit: 12/9/18
        Inputs - The health to lose (Float)
        Outputs - Health left (Float)
        Makes the character lose health"""

      self.health = self.health - healthToLose  #Lose health
      return self.health

##########################################################################################################
##########################################################################################################
###########################################

   def move_Left(self, amount):

      """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 26/9/18
        Inputs - The class
        Outputs - Move the Character left
        This will move the character left by 'amount' pixels"""
        self.canvasForCharacter.move(self.characterOnScreen, -amount, 0)#Moves the character
        self.canvasForCharacter.update()  #This will update the screen so the user can see it

##########################################################################################################
##########################################################################################################
############################################

    def move_Right(self):

        """Made By Nathaniel Lowis 1st Edit: 25/08/18.  Latest Edit: 11/9/18
          Inputs - The class
          Outputs - Move the Character Right
          This will move the character right by 2 pixels"""

        self.canvasForCharacter.move(self.characterOnScreen, 2, 0)  #Moves the character
        self.canvasForCharacter.update()  #This will update the screen so the user can see it

##########################################################################################################
##########################################################################################################
############################################

    def coord_Player(self):

        """Made by Nathaniel Lowis 1st Edit: 31/8/18.  Latest Edit: 31/8/18
          Inputs  - The character
          Outputs - Where the character is
          This will show where the character is"""

        coordinatesPlayer = self.canvasForCharacter.coords(self.characterOnScreen)  #This gives the coordinates for the
character
        return coordinatesPlayer #Returns it to the main program

##########################################################################################################
##########################################################################################################
############################################

    def fire_Hori_Bullet_Class(self, coordinateInFireBulletHori):

        """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
          Inputs - Coordinates of character in an array
          Outputs - Shooting bullet
          This controls the bullet"""


        bulletShootingHori = HoriBullet(coordinateInFireBulletHori[2], coordinateInFireBulletHori[1],
coordinateInFireBulletHori[2] + 5, coordinateInFireBulletHori[1]+5, self.canvasForCharacter)  #Makes an instant of
the HoriBullet Class
        fire_Hori_Bullet(bulletShootingHori)   #Controls the bullet
        bulletShootingHori.delete_Bullet()  #These will delete the bullet afterwards
        del(bulletShootingHori)

##########################################################################################################
##########################################################################################################
############################################
```

```
    def fire_Hori_Bullet_Class_Enemy(self, coordinateInFireBulletHoriEne):

        """Made by Nathaniel Lowis 1st Edit: 2/10/18 Latest Edit: 2/10/18
          Inputs - Coordinates of enemy character in an array
          Outputs - Shooting bullet
          This controls the bullet for the enemy"""


        bulletShootingHoriEne = HoriBullet(coordinateInFireBulletHoriEne[2], coordinateInFireBulletHoriEne[1] + 45,
coordinateInFireBulletHoriEne[2] +5 , coordinateInFireBulletHoriEne[1] + 40 , self.canvasForCharacter)  #Makes an
instant of the HoriBullet Class
        fire_Hori_Bullet_Ene(bulletShootingHoriEne)   #Controls the bullet
        bulletShootingHoriEne.delete_Bullet()  #These will delete the bullet afterwards
        del(bulletShootingHoriEne)

##############################################################################################
##############################################################################################
#######################################
    def fire_Para_Bullet_Class(self, coordinateInFireBulletPara):

        """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 28/9/18
          Inputs - Coordinates of character in an array
          Outputs - Shooting bullet
          This controls the Parabolic bullet """


        bulletShootingPara = ParabolicBullet(coordinateInFireBulletPara[2], coordinateInFireBulletPara[1],
coordinateInFireBulletPara[2] + 5, coordinateInFireBulletPara[1]+5, self.canvasForCharacter)  #Makes an instant of
the ParabolicBullet Class
        fire_Para_Bullet(bulletShootingPara)   #Controls the bullet
        bulletShootingPara.delete_Bullet()  #These will delete the bullet afterwards
        del(bulletShootingPara)

##############################################################################################
##############################################################################################
#######################################
##############################################################################################
##############################################################################################
#######################################

def set_Up():

    """Made by Nathaniel Lowis 1st Edit: 25/8/18 Latest Edit: 25/9/18
      Inputs - None
      Outputs - A working screen with the floor and endpoint and sent to main program
      This sets up the level the game"""

    rootSetUp = Tk()    #This makes the screen
    rootSetUp.title("Bullet Game")  #Gives game a name
    rootSetUp.resizable(True, True)    #This means the user can make it full screen
    canvasSetUp = Canvas(rootSetUp, width = 500, height = 500)  #Sets up the
    canvasSetUp.pack()  #Outputs screen
    floorSetUp = canvasSetUp.create_rectangle(500, 500, 0, 400, fill="grey")  #Sets up the floor
    endPointSetUp = canvasSetUp.create_rectangle(375, 375, 400, 400, fill="red")  #Sets up the endpoint
    heroSetUp = Character(canvasSetUp, 50, 70, 350, 400, "Blue")  #This will make the user's character
    enemySetUp = Character(canvasSetUp, 300, 320, 350 ,400, "Cyan")  #This will make the enemy class

    bulletsLeftSetUp = BulletsShot()



    return rootSetUp, canvasSetUp, floorSetUp, endPointSetUp, heroSetUp, enemySetUp, bulletsLeftSetUp #Sends
everything back to main program.

##############################################################################################
##############################################################################################
########################################

def fire_Hori_Bullet_Ene(bulletFiringEne):

    """Made by Nathaniel Lowis 1st Edit: 2/10/18 Latest: 3/10/18
      Inputs - The Class HoriBullet
      Outputs - The bullet moving and interacting with the environment
      This will allow the bullet to interact with the environment for the enemy bullet"""

    timer1StartEne = time.time()    #Starts a timer
    distanceToGoEne = bulletFiringEne.distance_To_Work_Out()  #Works out how far the bullet goes
    #print(distanceToGoEne, "Ene Dis")  #testing
    distPer10MilliEne = distanceToGoEne / 50   #Divides the distance by 50 so they are all in equal chunks
    bulletGoneEne = 0  #Creates a variable which works out how far the bullet has gone
    timeDoneHoriBulletEne = 0  #The time for how long the bullet has gone for

    while bulletGoneEne != distanceToGoEne and timeDoneHoriBulletEne < 5000:   #Whilst the bullet has not gone as
far as it needs to

        heroCoordinatesEne = hero.coord_Player()  #Gets hero's coordinates in an array
        #print(heroCoordinatesEne) #Testing
        bulletCoordinatesEne = bulletFiringEne.coord_Bullet()  #Gets the bullets coordinates in an array
        #print(bulletCoordinatesEne) #Testing

        #try:  #The program will go down this route when there is an enemy class #Not needed anymore
        #print("TRY") #Testing
        if bulletCoordinatesEne[0] <= heroCoordinatesEne[2] and bulletCoordinatesEne[2] >= heroCoordinatesEne[0]
and bulletCoordinatesEne[1] <= heroCoordinatesEne[3]: #If the bullet has hit the hero

            #print("HIT") #Testing
            timer1FinishEne = time.time()  #Stops timer
            timer1Ene = timer1FinishEne - timer1StartEne   #Works out length of time the  bullet has gone for

            ##print(bulletGone) #testing
            #print(timer1Ene)    #testing

            bulletFiringEne.speed_At_Any_Point(bulletGoneEne)   #Works out the speed of the bullet
            bulletFiringEne.energy_In_Bullet()    #Works out the energy of the bullet
            bulletPowerEne = bulletFiringEne.power_At_Point(timer1Ene)   #Works out the power of the bullet

            #print(bulletPowerEne, "Power") #testing

            bulletGoneEne = distanceToGoEne    #Means the while loop can stop
            hero_Lose_Health(bulletPowerEne) #Makes the hero lose health
    else:

        negativeDistPer10MilliEne = distPer10MilliEne * -1  #This will allow the bullet to move towards the hero
        bulletFiringEne.move_Bullet_Hori(negativeDistPer10MilliEne)   #Moves the ball the amount it needs to
        bulletGoneEne = bulletGoneEne + distPer10MilliEne  #Added to  bulletGone
        timeDoneHoriBulletEne = timeDoneHoriBulletEne + 100


##      except:  #This is when there is no enemy     #All not needed anymore
##
##          print("Except")
##
##          negativeDistPer10MilliEne = distPer10MilliEne * -1
##          bulletFiringEne.move_ball(negativeDistPer10MilliEne)   #Moves the ball the amount it needs to
##          bulletGoneEne = bulletGoneEne + distPer10MilliEne #Added to BulletGone
##          timeDoneHoriBulletEne = timeDoneHoriBulletEne + 100

##########################################################################################################################################################################################################################################

def fire_Hori_Bullet(bulletFiring):

    """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 12/9/18
      Inputs - The Class HoriBullet
      Outputs - The bullet moving and interacting with the environment
      This will allow the bullet to interact with the environment"""

    timer1Start = time.time()    #Starts a timer
    distanceToGo = bulletFiring.distance_To_Work_Out()  #Works out how far the bullet goes
    #print(distanceToGo, "Distance Hori")  #testing
    distPer10Milli = distanceToGo / 50   #Divides the distance by 50 so they are all in equal chunks
    bulletGone = 0  #Creates a variable which works out how far the bullet has gone
    timeDoneHoriBullet = 0  #How long the bullet has gone for

    while bulletGone != distanceToGo and timeDoneHoriBullet < 5000:   #Whilst the bullet has not gone as far as it
needs to

        enemyCoordinates = enemy.coord_Player()  #Gets enemy's coordinates in an array
        bulletCoordinates = bulletFiring.coord_Bullet()  #Gets the bullets coordinates in an array

        try:  #The program will go down this route when there is an enemy class

            if bulletCoordinates[0] > enemyCoordinates[0] and bulletCoordinates[1] >= enemyCoordinates[1]:  #If the
bullet is past the enemy
                timer1Finish = time.time()  #Stops timer
                timer1 = timer1Finish - timer1Start   #Works out length of time the  bullet has gone for

                ##print(bulletGone) #testing
                ##print(timer1)    #testing

                bulletFiring.speed_At_Any_Point(bulletGone)   #Works out the speed of the bullet
                bulletFiring.energy_In_Bullet()    #Works out the energy of the bullet
                bulletPower = bulletFiring.power_At_Point(timer1)   #Works out the power of the bullet

                #print(bulletPower, "Power") #testing

                bulletGone = distanceToGo    #Means the while loop can stop
                enemyHealth = enemy.lose_Health(bulletPower)  #Makes the enemy lose health

                #print(enemyHealth, "EnemyHealth")  #testing

                if enemyHealth <= 0:  #If the enemy's health is below or equal to 0
                    #CULD PUT THIS IN OWN FUNCTION
                  enemy.delete_Character()  #Deletes the enemy


        else:

                bulletFiring.move_Bullet_Hori(distPer10Milli)   #Moves the ball the amount it needs to
                bulletGone = bulletGone + distPer10Milli  #Added to  bulletGone
                timeDoneHoriBullet = timeDoneHoriBullet + 100

        except:  #This is when there is no enemy

        bulletFiring.move_Bullet_Hori(distPer10Milli)   #Moves the bullet
        bulletGone = bulletGone + distPer10Milli #Added to BulletGone
        timeDoneHoriBullet = timeDoneHoriBullet + 100

####################################################################################################
####################################################################################################
##########################################

def fire_Para_Bullet(bulletFiring2):

    """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 28/9/18
      Inputs - The Class ParabolicBullet
      Outputs - The bullet moving and interacting with the environment
      This will allow the bullet to interact with the environment"""


    #This bit is for the bullet moving upwards
    timer2Start = time.time()    #Starts a timer
    distanceToGoUp, distanceLeft = bulletFiring2.distance_Worker_Up()  #Works out how far the bullet goes
    #print(distanceToGoUp, distanceLeft, "Distance Para")  #testing
    distLeft = distanceLeft / 25  #Divides the distance by 25 so they are all in equal chunks to go left
    distPer10MilliUp = distanceToGoUp / 25  #Divides the distance by 25 so they are all in equal chunks to go up
    bulletGoneUp = 0  #Creates a variable which works out how far the bullet has gone
    timeDoneParaBullet = 0  #This will be how long the bullet has moved for
    goDown = True  #This is used to say whether the bullet should go down

    while bulletGoneUp != distanceToGoUp and timeDoneParaBullet < 2500:   #Whilst the bullet has not gone as far as
it needs to and not gone for long enough

        enemyCoordinates = enemy.coord_Player()  #Gets enemy's coordinates in an array
        bulletCoordinates2 = bulletFiring2.coord_Bullet()  #Gets the bullets coordinates in an array
        try:  #The program will go down this route when there is an enemy class

            if bulletCoordinates2[0] > enemyCoordinates[0] and bulletCoordinates2[1] >= enemyCoordinates[1] -5 and
bulletCoordinates2[0] < enemyCoordinates[2]:  #If the bullet is hitting the enemy
                ##print("HIT UP") #Testing

                timer2Finish = time.time()  #Stops timer
                timer2 = timer2Finish - timer2Start   #Works out length of time the  bullet has gone for
                ##print("HIT UP", timer2)  #Testing

                ##print(bulletGone) #testing
                ##print(timer1)    #testing

                bulletFiring2.speed_Vert_Up(bulletGoneUp, timer2)   #Works out the speed of the bullet going upwards
                ##print("Working With Speed") #Testing

                bulletFiring2.final_Speed_Parabolic()  #Works out the final speed for the bullet
                ##print("Final Speed") #Testing
                bulletFiring2.energy_In_Bullet()    #Works out the energy of the bullet
                ##print("Energy")  #Testing
                bulletPowerPara = bulletFiring2.power_At_Point(timer2) #Works out the power of the bullet
                ##print("Power") #Testing

                #print(bulletPowerPara, "Para") #testing

                bulletGoneUp = distanceToGoUp    #Means the while loop can stop
                enemyHealth2 = enemy.lose_Health(bulletPowerPara)  #Makes the enemy lose health

                #print(enemyHealth2, "Ene Health")  #testing
                goDown = False  #Means the bullet does not have to go down

                if enemyHealth2 <= 0:  #If the enemy's health is below or equal to 0
                    #CULD PUT THIS IN OWN FUNCTION
                   enemy.delete_Character()  #Deletes the enemy


            else:

                bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)   #Moves the ball the amount it needs to up
and left
                bulletGoneUp = bulletGoneUp + distPer10MilliUp  #Added to  bulletGoneUp how far it went
                timeDoneParaBullet = timeDoneParaBullet + 100   #Time is updated

        except:  #This is when there is no enemy

            bulletFiring2.move_Ball_Para_Up(distPer10MilliUp, distLeft)   #Moves the ball the amount it needs to up and
left
            bulletGoneUp = bulletGoneUp + distPer10MilliUp  #Added to  bulletGone how far it went
            timeDoneParaBullet = timeDoneParaBullet + 100    #Time is updated



    if goDown == True:         #Means the bullet can go downwards
        timer3Start = time.time()    #Starts a timer
        distanceToGoDown, distanceLeft2 = bulletFiring2.distance_Worker_Down()  #Works out how far the bullet goes
        #print(distanceToGoDown, distanceLeft2, "Para Dis 2")  #testing
        distLeftToGo = distanceLeft / 25   #Divides the distance by 25 so they are all in equal chunks to go up
        distPer10MilliDown = distanceToGoDown / 25  #Divides the distance by 25 so they are all in equal chunks to go
down
        bulletGoneDown = 0  #Creates a variable which works out how far the bullet has gone
        timeDoneParaBullet2 = 0  #Creates a variable for amount of time it has been

        while bulletGoneDown != distanceToGoDown and timeDoneParaBullet2 < 2500:   #Whilst the bullet has not
gone as far as it needs to and not for long enough

            enemyCoordinates2 = enemy.coord_Player()  #Gets enemy's coordinates in an array
            bulletCoordinates3 = bulletFiring2.coord_Bullet()  #Gets the bullets coordinates in an array

            try:  #The program will go down this route when there is an enemy class

                if bulletCoordinates3[0] > enemyCoordinates2[0] and bulletCoordinates3[1] >= enemyCoordinates2[1] -5
and bulletCoordinates3[0] < enemyCoordinates2[2]:  #If the bullet is hitting the enemy

                    #print("HIT Down") #testing
                    timer3Finish = time.time()  #Stops timer
                    timer3 = timer3Finish - timer3Start   #Works out length of time the  bullet has gone for

                    #print("HIT Down", timer3) #Testing


                    ##print(bulletGone) #testing
                    ##print(timer1)    #testing

                    bulletFiring2.speed_Vert_Down(timer3)   #Works out the speed of the bullet going downwards

                    bulletFiring2.final_Speed_Parabolic()  #Works out the final speed of the bullet
                    bulletFiring2.energy_In_Bullet()    #Works out the energy of the bullet
                    bulletPowerPara2 = bulletFiring2.power_At_Point(timer3)   #Works out the power of the bullet

                    #print(bulletPowerPara2, "Power") #testing

                    bulletGoneDown = distanceToGoDown    #Means the while loop can stop
                    enemyHealth3 = enemy.lose_Health(bulletPowerPara2)  #Makes the enemy lose health
                    #print("Health: ", enemyHealth) #Testing

                    if enemyHealth3 <= 0:  #If the enemy's health is below or equal to 0
                        #CULD PUT THIS IN OWN FUNCTION
                       enemy.delete_Character()  #Deletes the enemy


                else:

                    bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)   #Moves the ball the amount it needs
to move left and down
                    bulletGoneDown = bulletGoneDown + distPer10MilliDown #Added to  bulletGoneDown
                    timeDoneParaBullet2 = timeDoneParaBullet2 + 100  #Updates time

            except:  #This is when there is no enemy

                bulletFiring2.move_Ball_Para_Down(distPer10MilliDown, distLeft)   #Moves the ball the amount it needs to
            bulletGoneDown = bulletGoneDown + distPer10MilliDown #Added to  bulletGoneDown
            timeDoneParaBullet2 = timeDoneParaBullet2 + 100  #Updates Time

################################################################################################################################################################################################################################################

#def key(event):

#    """Used as testing """

#    print("pressed", repr(event.char))  #TESTING

################################################################################################################################################################################################################################################

def enemy_Shooting():

    """Made By Nathaniel Lowis 1st Edit: 2/10/18,  Latest Edit: 2/10/18
      Inputs - None
      Outputs - None
      This allows the enemy to shoot"""

    characterPosistionForEnemy = enemy.coord_Player() #Gets the coordinates for the enemy's character
    enemy.fire_Hori_Bullet_Class_Enemy(characterPosistionForEnemy)  #Fires bullet

################################################################################################################################################################################################################################################

def hero_Touch_Enemy():

    """Made By Nathaniel Lowis 1st Edit: 28/9/18, Latest Edit: 3/10/18
      Inputs - None
      Outputs - None
      This will mean the user's character is hurt if it touches the enemy"""

    heroCurrentCoordinates = hero.coord_Player()  #Gets the hero's coordinates
    enemyCurrentCoordinates = enemy.coord_Player() #Gets the Enemy's coordinates

    try:  #If there is an enemy

        if heroCurrentCoordinates[2] >= enemyCurrentCoordinates[0] and heroCurrentCoordinates[3] >=
enemyCurrentCoordinates[1]:  #If the hero touches the enemy

            hero_Lose_Health(50) #Hero Loses health
            hero.move_Left(20)  #Hero pushed back

    except:  #If there is no enemy nothing should happen
        pass

################################################################################################################################################################################################################################################
```

```
def scoring(timeDoneIn, levelScore):

   """Made By Nathaniel Lowis  1st Edit: 17/10/18 Latest Edit: 18/10/18
     Inputs - the time the level has been done in (Float), the score for the level (integer)
     Outputs - The final Score (Float)"""

   healthForHero = hero.health_Left()  #Gets how much health the hero has
   howManyBulletsLeft = bulletsLeft.bullets_Show()  #Gets how many bullets the user has
   finalScore = (1 / timeDoneIn) + (howManyBulletsLeft * 5) + healthForHero + levelScore  #Works outs the score for
the level

   with open("Scores.csv", mode = "wt", encoding = "utf-8") as addingScores:  #OPens and wipes a csv file which will
hold the score which will be added to the database

      writingARow = csv.writer(addingScores)  #Allows to write into the csv
      arrayWithScoreIn = [finalScore, "\n"]  #What needs to be entered into the csv
      writingARow.writerow(arrayWithScoreIn)  #Writes in the data to the csv

   import adding_To_Database  #Runs the code to add data to a database




   return finalScore  #Returns score to the main program

##########################################################################################
##########################################################################################
#####################################

def hero_Lose_Health(healthToLoseFunction):

   """Made by Nathaniel Lowis 1st Edit: 28/9/18, Latest Edit: 17/10/18
     Inputs - Health to lose (Float)
     Outputs - None
     This will take away health if the hero is hit and stop the game if the hero is dead"""

   heroHealth = hero.lose_Health(healthToLoseFunction)  #Gets Hero to lose health
   #print(heroHealth, "Health Hero")  #Testing

   if heroHealth <= 0:  #If the hero is 'dead' it should delete everything
     # print("DEAD")
      finalTimerEnd2 = time.time()  #Gets the latest time
      frame.destroy()  #Deletes everything


      finalTime2 = finalTimerEnd2 - finalTimerStart  #Works out how long it took to complete the level
      #print(finalTimerEnd2, "Time")   #testing
      scoring(finalTime2, 0)  #Calls the function to work out the school

##########################################################################################
##########################################################################################
#####################################

def callback(event):
   """Editied from: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm  Made by Nathaniel Lowis  1st 
Edit: 25/8/18  1st Edit: 31/8/18
   Inputs - What the user did
   Output- Allows user to move left and right
   Means we can press on the screen and use the buttons"""

   frame.focus_set()   #This will mean we can use the left and right button as you have to press the window
   ##print("clicked at", event.x, event.y)  Used as testing

################################################################################################################################################################################################################################################

def hero_Move_Left(event):

   """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 28/9/18
   Inputs - What the user did
   Output- Allows user to move left
   Makes the hero move left and checks if the game has finished"""

   #print("pressed 'Left'")  #testing
   hero.move_Left(2)#Calls this method to move character
   hero_Touch_Enemy()  #This will check if you have touched the enemy
   finish_Game()  #Checks to see whether you have finished the game

################################################################################################################################################################################################################################################

def hero_Move_Right(event):

   """Made by Nathaniel Lowis  1st Edit: 31/8/18  1st Edit: 28/9/18
   Inputs - What the user did
   Output- Allows user to move Right
   Makes the hero move Right and checks if the game has finished"""

   #print("pressed 'Right'")  #testing
   hero.move_Right() #Calls this method to move character##
   hero_Touch_Enemy()  #Will check if you are touching the enemu
   finish_Game()  #Checks to see if the user has finished the game

################################################################################################################################################################################################################################################

def hero_Jump(event):

   """Made by Nathaniel Lowis  1st Edit: 25/9/18.  Latest Edit: 3/10/18
   Inputs - What the user did
   Output- Allows user to move Jump
   Makes the hero move Right and checks if the game has finished"""

   #print("pressed 'Up'")  #testing
   heroCoordinatesJumpFunction = hero.coord_Player()
```

```
   if heroCoordinatesJumpFunction[3] != 400:   #This is used to check if the user is touching the floor. If not, it will
pass
       pass
   else:
       hero.jump() #Calls this method to move character##

##############################################################################################################
##############################################################################################################
############################################

def finish_Game():

   """Made by Nathaniel Lowis  1st Edit: 31/8/18  Latest Edit: 17/10/18
     Inputs - Nothing
     Outputs - Checks if game has finished
     This will check if you have passed the ending point and if you have deletes the screen"""


   ##print("TRYING")  Used for testing
   x0Endpoint = 375  #These are the known coordinates for the endpoint
   x1Endpoint = 400
   y0Endpoint = 375
   y1Endpoint = 400

   characterPosistion = hero.coord_Player()  #Works out where the user's character is

   if characterPosistion[2] >= x0Endpoint and characterPosistion[3] >= y0Endpoint and characterPosistion[3] <=
y1Endpoint and characterPosistion[2] <= x1Endpoint:   #Checks if the character has passed the endpoint

       finalTimerEnd = time.time()   #Gets the latest time
       frame.destroy()  #If they have it will delete all objects on the canvas
       finalTime = finalTimerEnd - finalTimerStart  #Work out how long it took the user to play the level
       #print(finalTime, "time") #testing
       scoring(finalTime, 100)  #Works out the score for the user

##############################################################################################################
##############################################################################################################
############################################

def fire_Bullet(letter):

   """Made by Nathaniel Lowis 1st Edit: 10/9/18  Latest Edit: 17/10/18
     Inputs - The q which must (for some reason) be included
     Outputs - None
     This will start the process of firing a Horizontal bullet and get the enemy to shoot first"""

   #print("pressed 'q'")  #testing
   bulletLeftHori = bulletsLeft.bullets_Show()  #Checks out how many bullets the user has
   #print(bulletLeftHori)  #testing
   if bulletLeftHori > 0: # If you have bullets it will start the shooting process

       bulletsLeft.bullets_Sub()  #Takes away a bullet from your score

       try:  #If there is an enemy it should start the shooting process
```

```
            enemy_Shooting()  #Allows the enemy to shoot

        except: #If there is no enemy it should do nothing
            pass

        characterPosistion = hero.coord_Player() #Gets the coordinates for the user's character
        hero.fire_Hori_Bullet_Class(characterPosistion)  #Fires bullet

################################################################################################
################################################################################################
#########################################

def fire_Bullet_Para(letter):

    """Made by Nathaniel Lowis 1st Edit: 27/9/18  Latest Edit: 17/10/18
       Inputs - The w which must (for some reason) be included
       Outputs - None
       This will start the process of firing a Parabolic bullet"""

    #print("pressed 'w'")  #testing

    bulletLeftPara = bulletsLeft.bullets_Show()  #Checks to see if you have bullets left
    #print(bulletLeftPara)  #testing

    if bulletLeftPara > 0:  #If you do it will start shooting

        bulletsLeft.bullets_Sub()  #Takes away a bullet

        characterPosistionPara = hero.coord_Player() #Gets the coordinates for the user's character
        hero.fire_Para_Bullet_Class(characterPosistionPara)  #Fires bullet

################################################################################################
################################################################################################
#########################################

#main
finalTimerStart = time.time()  #Gets time so it can work out how long it took the user

root, frame, floor, endPoint, hero, enemy, bulletsLeft= set_Up()  #Sets up game



frame.bind("<Left>", hero_Move_Left)  #Moves Character left if left button pressed
frame.bind("<Right>", hero_Move_Right)  #Moves Character Right if right button pressed
frame.bind("<Up>", hero_Jump)      #Allows the user to jump
frame.bind("<Button-1>", callback)  #Means user can press screen
frame.bind("<q>", fire_Bullet)     #Allows the user can shoot a horizontal Bullet
frame.bind("<w>", fire_Bullet_Para)   #Allows the user to shoot a parabolic bullet

#frame.bind("<Key>", key) # testing
frame.pack() #Sends it to the screen



frame.mainloop() #Infinte loop used
```

Bibliography

New Mexico Tech (Last Accessed on 12/7/18) Tkinter 8.5 reference: a GUI for Python URL: http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/canvas-methods.html

Stack Overflow (Last Accessed on 12/7/18) URL: https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas

Effbot (Last Accessed on 12/7/18) URL: http://effbot.org/tkinterbook/canvas.htm

Repl.it @LevMckinney (27th June 2018) A simulator for calculating the flight of a projectile in the atmosphere Last Accessed 17/7/18 URL https://repl.it/talk/share/-simulator-for-calculating-the-flight-of-a-projectile-in-the-atmosphere/4767

New Mexico Tech Last Accessed 10/9/18. URL: http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/universal.html

Google Search Last Accessed 11/9/19 URL: https://www.google.co.uk/search?q=weight+of+bullet&rlz=1C1GGRV_enGB764GB764&oq=weight+of+bullet&aqs=chrome..69i57.2707j0j7&sourceid=chrome&ie=UTF-8

Wikipedia (Last Accessed 4/10/18) Python programming Languages URL: https://en.wikipedia.org/wiki/Python_(programming_language)

New Mexico Tech (Last Accessed 25/8/18) Tkinter 8.5 reference: a GUI for Python. URL: http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_rectangle.html

Effbot (Last Accessed 11/10/18) URL: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

Stack Overflow (Last Accessed 11/10/18) Moving Balls in Tkinter URL: https://stackoverflow.com/questions/25430786/moving-balls-in-tkinter-canvas

The Physics Factbook (Last Accessed 11/10/18) URL:https://hypertextbook.com/facts/2003/MichaelTse.shtml

evatotuts+ (Last Accessed 7/9/18) URL: https://code.tutsplus.com/articles/introduction-to-parallel-and-concurrent-programming-in-python--cms-28612

REPL.it @NathanielLowis (Last Accessed 20/10/18) URL: https://repl.it/@NathanielLowis/Database-in-python

Stack Overflow (Last Accessed 7/11/10) URL: https://stackoverflow.com/questions/7974849/how-can-i-make-one-python-file-run-another

Effbot (Last Accessed 7/10/18) URL: http://effbot.org/tkinterbook/entry.htm

Wikipedia (Last Accessed 7/10/18) TimSort URL: https://en.wikipedia.org/wiki/Timsort

REPL.it @NathanielLowis (Last Accessed 7/10/18) URL: https://repl.it/@NathanielLowis/Quick-Sort