

OCR GCE A COMPUTER SCIENCE PROJECT H446-03

Name : Cameron Noakes

Candidate Number: 9453

The Fitzwimarc School: 16217

Title of Project: Exhumation Anthropology

H446-03 – PROJECT CONTENTS

TABLE OF CONTENTS

A. Analysis	5
Research:.....	5
Summary of what I will/will not be using:	8
Reason:	9
Computational Methods (5):	10
Stakeholders:.....	12
Success Criteria and requirements.....	17
Features of the proposed solution	18
Limitations:	18
The Controls.....	20
Hardware and software.....	20
game states.....	21
scoring	21
How to win or lose	21
Additional Points on proposed solution.....	22
B. Design	23
Systems diagram	23
Proposed Screen Design and usability features	30
Scoring	31
Menu	32
Peripheral Instructions	34
Pause menu	35
Setting/ background.....	35
Username input	37

sprite sheet	38
Summary and structure	38
Classes	40
Algorithms.....	41
how the complete solution will work.....	46
Testing self-contained systems with DRy data	47
SDLC Process	50
Beta TESTING	50
Assignment of Variables	56
Proposal Sign off	63
Menu Summary	63
Username Screen Summary	63
Main Game Summary	64
Pause Menu Summary	64
End-Game Screen Summary.....	65
C. Developing the coded solution (“The development story”).....	65
Task #1 – Sprite making and Initializing.....	65
Prototype Making.....	67
Prototype #1 - Drawing Sprites	67
Prototype #2 – Menu	69
Task #2 – The peripheral controls	70
Task #3 – Collision detection.....	73
Task #3B – Objects Class – collison detection.....	76
Task #4 – Player Class	77
Task #6 – Setting the background	79
Task #7 – Menu.....	80
Task #8 – Username input page	85

Background Music	89
Username validation:.....	91
How each section links together to form the overall solution	93
Main Game Code	93
Missile-Bullet Code	99
Stakeholders input at key development features	100
Overview of Most Issues and Fixes.....	101
Overall Development Summary	104
D. Evaluation	104
Testing Tables	105
Overall Menu Testing.....	105
Overall Username Page.....	105
<i>Running Game State Testing</i>	106
Rendering to the screen.....	106
Sound effects.....	107
User Sign off agreement	108
Signature of approval.....	108
Evaluation of Solution – usability features	108
Does the solution meet the requirements?.....	110
Missed Features.....	116
Bugs to fix before pushing the code to production	118
Upgrades	118
Project Appendixes.....	119
Appendix A – Code Listing	119

A. ANALYSIS

Overview

Exhumation Anthropology will be a 2-dimensional top down design with infinite waves of zombies based around 1 or maximum of 2 levels. You play as a survivor, the aim is to survive as long as you can against the infinite waves of zombies, the game will have an infinite loop and continues forever, continuously spewing out more zombies than the previous round.

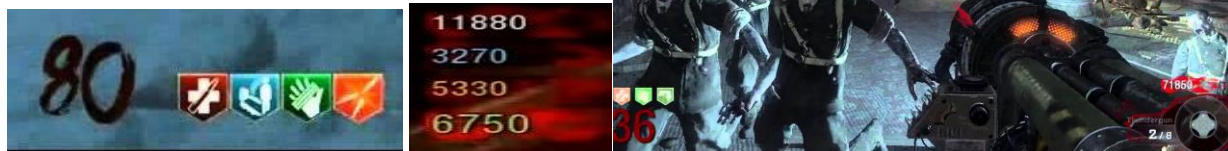
I may even decide to have just one big level which would reduce the need to have multiple ones my game will be targeting desktop computers with windows to run my game. Exhumation Anthropology means, digging up corpses [exhumation] to study human development and run tests/ experiments [Anthropology] I called it this because it has connotations of testing experiments on humans, it goes wrong, burying them and them coming back as zombies. I feel this name is a great example of literature and the way you can combine words to describe ideas, my idea being my game.

This can be abbreviated to 'ExAn'. EA is a worldwide gaming company and if I was to call my game that abbreviation I could run into legal issues. My game is based around two games that have been very successful over the years, these games are 'Hotline Miami' and 'Call of Duty Black Ops', both are analysed below in 'Research'.

The reason for the camera angle of top-down two dimensional is because I feel that this suits the topic of retro, classical games from years back where there is an odd use of the camera angle and doesn't match main games (as main games are first person or third person), it also allows the player to see more of the screen and appreciate the rendering as well as plan out routes ahead of time (thinking ahead) in order to stay one step ahead of the zombies.

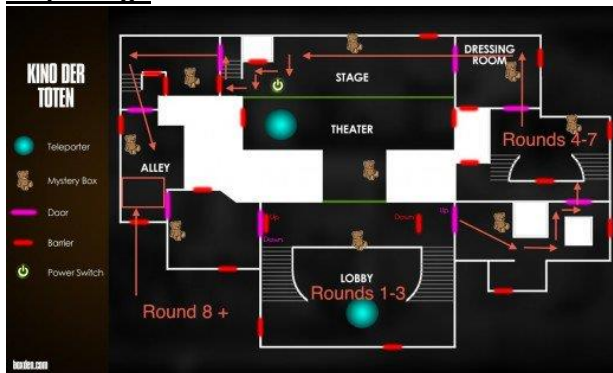
RESEARCH:

Game 1: Call of Duty Black Ops 1, COD BO1 is a first-person shooter video game, developed by Treyarch and published by Activision, from this I will be focusing on the zombie levels rather than the campaign, which is about war or the dead ops arcade minigame. I like the idea of infinite waves of zombies continuously trying to kill you, so I will implement that. I do enjoy the feel of a points-based system but may include a virtual currency instead to make my design more original and coherent. There is also perks (pickups) that I might consider including as well. I won't be adding first person or anywhere near the realistic graphics in Black Ops as the limitation is too big, more on that later.





Map Design



The Black Ops zombie map design is interesting and unusual, making it original and out of the ordinary, it is a theatre, literally the last place for anyone to think a zombie game would take place. My map design will be a random place, much to the same effect as Kino Der Toten in Black Ops, somewhere that hasn't been recreated in films or popular games, a farm or somewhere with wide open areas easy to maneuver around the zombies. I will create blueprints like above for my map design to make it easier to acknowledge and understand.

However, I will be including the features of infinite waves of zombies as well as 1 map design per level (Black Ops has multiple maps but when you die you restart the same map, this is what I will be including). The weapons in Black Ops is a wide variety and based on the first stakeholders' feedback below I will try my best to include this variety. The audio ideas are good, and I will try to incorporate my own versions of them such as zombie noises or music (from Hotline Miami or Carpenter Brut, there is a one-hour version of his music). The gun weaponry in Black Ops is bought from walls containing a chalk outline of the type of gun, I will be adapting this feature and instead, will have an armoury where you can buy better weapons in return will cost points, points will be given for every successful bullet landing in the enemy. I will be taking the idea of perks possibly depending on limitations like how hard it is to implement and how time consuming it will be, alternatively I could have the weapons have special powers with them such as a flamethrower with fire grenades.

The Black Ops pause menu is very simple, it pauses the state of the game and has 3 options, resume, settings and quit. I will be including a simplistic menu like this also. I also enjoy the idea of having to open pathways to progress through the level more and to different areas of the map and costing points as payment, this will be a decent sized part of my project.

The controls for PC are W,A,S,D as well as the mouse clicks, very similar to other types of games as follows basic gaming constructs.

Candidate Name: Cameron Noakes

Candidate Number: 9453

Black Ops Zombies Perks, this game has a feature called 'perks', perks allow you to gain a better advantage over the zombies, can be the form of an extra life, more hits before you die or faster reloading. I unfortunately will not be including perks but might consider having special weapons as a 'lucky charm'.

Camera Angle: The camera angle is First person as it is an FPS (first person shooter) game. This allows the game to be more realistic as it is shot by an everyday human angle. My game is 2D and has a top-down view so the graphics will be much more simplistic and will use sprites.

Game 2: Hotline Miami, this is a retro, top down 2D game based around clearing out houses of 'bad guys' with brutality and logical thinking. You, much to the same effect as Black Ops, also have a points-based system.

Hotline Miami has starting weapons and I will adopt this feature too. This allows the user to start off with a selective weapon for the level but can be switched out over the course of the level for other weapons. In Hotline Miami you choose these weapons by selecting a certain mask in which comes with the weapon.

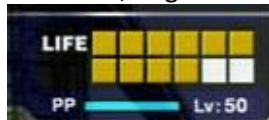
The game has multiple levels (around 16) and each one becomes progressively harder, making it crucial to think logically about the strategy. Additionally, you only have one life, making critical thinking even more crucial for completion of the level. As stated before, the camera angle is top-down 2D making it easy to see a lot of the map/level, I will be also using this angle as well as having 1 life but health slots and after they disappear, you die. I will have one map design as well.

This is a very fast-paced game which requires skill on tackling the levels, this gave me the idea of multiple levels but was later overruled by the levels in Black Ops. On the other hand, I did take the idea of top down 2D from this game. I will also have music over the top of my game to create suspense (just like in hotline Miami). However, I will not be including all the weapons they have in the game as there are too many, the same with the masks. I hope to create the project with similar graphics to Hotline Miami. I may even include some sprites from the game. Hotline Miami also looks neon-like as well as Synthwave and will try to include this to help the user feel nostalgia.





I will be taking multiple ideas from Hotline Miami like the variety of weapons (just like from Black ops), the score system (also the same with Black Ops), I will be taking the same, or similar music from Hotline Miami as well as the sprite ideas such as the blood splatter and dead bodies as well as the objects like sofas, chairs, plants, fences, tables and so on. I do enjoy the map design and how when you move the character the map also moves and is dynamic, helping to show actual progression through the level, allowing me to consider this, even if it is abruptly switching when the player goes off screen, although I would like it to be a fluent transition. The health system in Hotline Miami is a one-life per level and you restart if you die, I like this idea, This can be considered too hard and after a while it gets frustrating, I therefore, might decide to have it as dead risings health block bars to signify health like below.



Dead rising's health bar

In hotline Miami the controls are W,A,S,D and right and left mouse click. The left mouse button is to shoot, and right mouse click is to throw the weapon. I was inspired by this game to have similar controls in my game, however, may decide to add in a reload button as a form of suspense building. The simple controls add to the feel of a retro game as it is not overcomplicated.

SUMMARY OF WHAT I WILL/WILL NOT BE USING:

Black Ops implementing:

- ❖ I will be using one big level for the game and not needing any more than maximum 2 if I do end up deciding to have more than one but is unlikely. This makes it easier to get the plyer familiar with the level and map design.
- ❖ Infinite waves of zombies: The survivor has to keep fighting them off. This feature is needed in order to create suspense and for the game to have a purpose/solution, without it there wouldn't be a game.
- ❖ Points-based system/currency: in order to keep track of how successful the user is and to handle calculations for buying weapons. This is needed in order to show how well you are doing in the game and helps to progress further.
- ❖ I won't be including First person shooter (FPS) as my game is 2D
- ❖ Reduced graphics: limitation due to processing power as well as my game isn't mean to have amazing graphics.
- ❖ Variety of weapons: I will be including a variety of weapons but not to the scale as Black ops or Hotline Miami. This feature will be used to mix up the gameplay a bit more and keep it interesting
- ❖ Audio features: Will be including noises such as zombies and gun shots and a hurt noise when the player is attacked. This feature will help to immerse the user in the game more.
- ❖ I will be implementing audio music just like in Hotline Miami.

- ❖ Buying guns with points/currency: I will adapt the idea of wall guns from Black ops to an armoury.
- ❖ Might decide to include weapons with special powers to make the game more interesting.
- ❖ Pause menu: I will adopt the idea of a simplistic pause menu just like in Black Ops. This feature is needed to pause the state of the game and to have breaks.

Hotline Miami implementing:

- ❖ The 2D aspect: I will be including this feature as helps to make the game have a retro feel and it's the way I want to design my game. This will be implemented to add the retro classical feel to it.
 - ❖ I won't be including the masks like the ones in Hotline Miami as unnecessary and not a feature I am very passionate about and won't be able to see them from top-down.
 - ❖ Similar graphics: This is because of the processing power as well as helps to add a retro feel to the game without over complicating it with high-end graphics.
 - ❖ Same or similar sprite-type: This makes it easier to have a retro game as you are basing your sprites from a successful retro game.
 - ❖ Colours: I will use similar colours in order to capture the synth and retro feeling.
 - ❖ Choice of objects: I will be including objects such as chairs and fences to make the game more realistic.
 - ❖ I will have a similar map design to Hotline Miami ones but maybe will be based outside.
 - ❖ Blood splatter: I do like this feature in Hotline Miami and will include this for sure, in which appeals to my age audience.
 - ❖ Dead bodies: Dead bodies are to link more to the age audience as well, it is more violent appealing to an older audience (teenagers).
 - ❖ Health: I dislike the idea of having one life so will implement a feature from 'dead rising' where you have a health bar.
-

REASON:

The reason why I decided to make this my primary project goal is because there is a gap in the market to pursue retro top down 2D game development even though these, in my opinion, are some of the greatest. They represent classical games and give a sense of nostalgia directed towards the user, constantly reminding them of games such as 'Fallout Retrospective' (a top down 2.5D game in a wasteland – released in 2008). My aim is to bring back the feelings of nostalgia by implementing aspects of classic retro games into my project, especially top down 2D ones for users to relive memories of their past. Another one of my aims is for my stakeholders to have a short break from work and enjoy my game as a form of light gaming entertainment to help relieve stress.

Tracking metrics: The main metrics I will track to make sure my project is on target is: Creating the project in as less lines of code as possible, making the execution and code more efficient and making sure I am also efficient with my time and do not use any unnecessary code (similar to the first tracking metric), build a project with using mainly subroutines to help with debugging and keeping all code self-contained as well as also making sure my game doesn't throw any errors to the user unless it is in try and except blocks that don't crash the game.

Success Criteria:

Success metrics: My main success metric will be whether or not the user is happy with playing my game and enjoys the concept and understands the amount of effort I have put into creating it. Another

success metric of mine will be if the user feels any nostalgia towards the game (I won't be monitoring brain activity; I will just ask them). This, alongside the tracking metrics, will be my success criteria on how I perceive my game to be successful.

COMPUTATIONAL METHODS (5):

Computer games all use a series of thinking methods, these methods are called computational methods and help to solve problems and tasks that need completing. There are 5 main strands of computational methods and they are listed below; each one should be thought of when creating the project in turn.

- **Thinking Ahead:** This method is used to be able to plan out what needs to come next after the current task, this then allows easy identification of problems in the future and how to solve them in the current situation, it helps be more organised and makes tasks easier to complete.

My game will be using thinking ahead by working through the project and simultaneously thinking of what comes next, allowing me to set brief requirements for that next task. For example, after developing the menu display, I can then think of how the user will navigate to the buttons. Another will be when programming the bullet collision with the target what else should it need in order to work.

My inputs for my game will be the arrow keys on a keyboard and the mouse, I will also implement a reload feature in which will be on the 'r' button.

My outputs on the VDU screen will be player movement, enemy movement, possibly the map moving around the orientation of the player, health, powerups, inventory and the level theme.

I am a little concerned on how I will make the map move around when the player moves as well as the camera to follow, however, I am less concerned with the camera as the camera shot can be static and the player can be always in a set location on the screen.

- **Thinking Logically:** Mainly refers to the programming construct of selection where decisions can be made. Have a conditional sentence and rewrite it in logical expressions and reduce the word count. It makes it easier to see where decisions will be made based on what conditions.

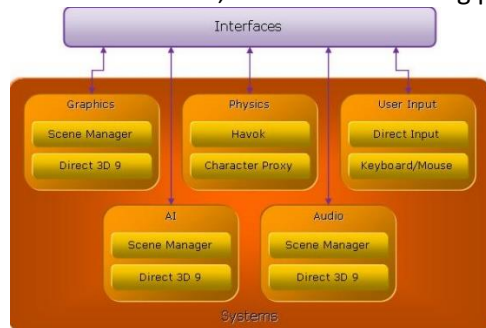
With logical thinking I will be able to use abstract selection down into basic branches and expand them later, this will make it easier to focus on the main parts instead of selective, conditional subroutines. An example is how the bullet, after fired, will be deduced from the magazine and how health will be lost from the bullet. Logically, if you throw (or fire) something you do not have it anymore which is why it is important to reduce the clip size for the magazine.

- **Thinking Procedurally:** A disciplined method of a combination of thinking in sequence (one after the other) and logically (to make decisions after a set number of steps). Thinking Procedurally is all about breaking up problems into smaller chunks making it easier to find a solution to them. Thinking this way will help get my head around how to code this up and the tasks I need to

consider. Using subroutines can help as they are self-contained and split a large program into smaller chunks, SWE companies do this for multiple engineers to work on different parts without affecting others work. I will adopt this approach.

An example can be when I start to code the physics, what should happen first when bullet is fired, deduct bullet from magazine or health from target? In this case it would be deduct the bullet from the magazine first as the bullet is fired then if the bullet hits the enemy the health will decrease.

The diagram below helps to explain how a problem can be broken down through decomposition into smaller tasks, this is what thinking procedurally allows you to do.



Game States: The game states I will include are if the game is running or not, therefore, the menu game state, the state of the player (if they're still alive) and the game state on whether or not a new weapon is bought in the game. Each game state will be tested for the Boolean value, if the state is true this means it is running, otherwise it is false. Each one will make sure that an aspect of my project is running and at the correct time. The game state will be sequential going from the 'menu' game state to the 'play' game state in which in turn will go through each game state and test when a change is needed.

- **Thinking Abstractly:** Removing unnecessary information in order to break up tasks into fundamental steps to make it easier to code up. Simplifies physics, graphics and objects as well as scenery to make it easier to work with. Detail is added later when the fundamental components are functioning. I will also have a pause menu; this will pause the state of the game in which cannot be done in real life hence a form of abstraction of real life.

My graphics, at the start, will be simplistic and the objects will be basic shapes (squares, circles etc.) to make sure I have the foundation down before anything else. Graphics can be added at a later date. Player will be a blue dot and the enemies will be red dots. I won't include items such as backgrounds like sofas, chairs, textures, blood splatter etc as these aren't necessary in order to test the state of the game to see if it works.

- **Thinking Concurrently:** Tackling multiple problems at once to create an overall efficiency. Known for several things happening at once.

Will be used for zombies moving while the player moves or does any other action (Zombies aren't dependant on the players movement or actions except for the pathfinding).

I will be using concurrent processing for the actions of the player and when the game is running to have the music play in the background. Concurrent processing is used widely in my project.

Target audience

The target audience for my game is the age of teenagers (roughly around the ages of 15-18) this is because they need to be old enough to understand how to play the game and use the controls and also has a minimum age of 15 because this game will include gore and blood which can be distasteful for some underage users. I also chose teenagers because they are the ones that play the most games in their free time and carry an interest in them the most.

The target audience of teenagers has also been chosen due to the fact that most teenagers have deadlines and assignments due in, whether that be A levels or College coursework, this game can act as a platform to de-stress them as they can hop onto the game to help them relax and get away from all the stress for a little bit of time, this is because my game does not have a storyline therefore can leave the game whenever the user feels like it, allowing it to be a quick time filler, allowing them to play it when they have a quick time window.

My platform is windows 10 on PC, this is handy as most teenagers will have access to a PC more than a console as consoles are more expensive as consoles are a set price whereas PC's have different and cheaper prices. The age is capped at 18 because above ages are considered to have more important things to do such as University and jobs whereas most people up to the age of 18 are still in education. Having the target audience of teenagers allows me to add in more complex features that don't throw off the users because they are too young to understand them, which can help to improve my game but not so complex that it is too hard to understand and play because this would make my game unsuccessful.

STAKEHOLDERS:

Stakeholders are people who are integrated into the development of the game, giving feedback, ideas and to test the game at specific states/ prototypes. They allow user feedback in which is crucial to rapid application development (the methodology I will be using) as the feedback will be put into the next prototype. They specify what the user could want. I have two stakeholders, Jake Parker and Jai Alden.

The stakeholders I have spoken to are disappointed with there not being many retro games (that are also 2D) and feel the same way I do, to tackle this problem I am creating a new retro game based on other successful retro games of the past.

Jake Parker

Jake is 17 years old and over the many years has played a lot of games, we used to go round each other's house and play video games together of zombie survival mainly with Black Ops, we have bonded over these games and discussed many concerns and information about these types of games and this was a main reason why I wanted him to be one of my stakeholders as we both enjoy the same type of genre but on different consoles allowing variety of sources for the games we play. Jake also doesn't like the idea how technology has become so vast that we use them to over complicate games and wishes he

Candidate Name: Cameron Noakes

Candidate Number: 9453

could go back and relive the moments of retro games more where there were just simple rules without complexity. I did an interview for him to share the ideas on my video game and provide alternative options to the development of it. The interview is given below:

Interview with Jake Parker

Q. What experience have you had with gaming?

I have played a lot of games from 8 years old, Xbox classic starting then moved to PlayStation and have played a lot of shooter games.

Q. Console (if so specify) or PC?

PlayStation now and retro games on old Xbox

Q. What is your favourite genre of games?

Shooting, zombie survival

Q. Have you played any retro games (original, classic games like arcade games e.g pacman etc.)?

Dead ops arcade, pacman, snakes and ladders, hotline Miami

Q. What do you do in spare time?

Enjoy coding in python, play online games, research into development of games

Q. What are your ideas on how retro games are now not as common in 2019

It's a Shame, a lot of successful games are retro, now they are too complex and too much tech is involved, retro games are more enjoyable, shooting games that are basic are more fun because they are less complex.

Q. What is your favourite retro game (examples: hotline Miami, pacman, BO1 arcade, space invaders, Tetris, donkey Kong, Sim city)

Dead ops arcade

Q. Any ideas you would recommend recreating from games (map/ graphics design, weapons, points, currency, enemies etc.)

Variety of weapons, as rounds progress more weapons, rounds are good as competitive, infinite zombies is a good idea as goes off more on skill.

Q. What is your opinion on my game design

1 map is a good idea, maybe 2 or 3. Round 1-3 on first map then 3-5 on second map, different maps to make the game more enjoyable. But one overall is good not to overwhelm the user.

Q. How many options on the menu screen?

4

Q. How many powerups should there be?

Start with 2 powerups until round 5 then increment it more as it gets harder.

Q. How many hits on an enemy till they die (round 1-4)?

2 hits, one to weaken them another to finish them off

Q. How many enemies for the first round?

10-15 enemies on first round, enough for the player to get used to the game and enjoy it without instantly dying.

Q. What should be the starting weapon?

Start with a handgun, maybe have a knife that is a one hit kill as it forces the player to risk their life in a swarm of zombies when out of ammo.

Q. How many points should be given for each kill?

100 pts (points) as it is easy to see and it's easy to use in calculations.

Q. What age would you rate my game?

13-18 years old as zombies can be a bit scary for younger.

Q. To prevent infinite level playing (leaving 1 enemy alive), should the game have a level countdown to complete until the next level starts?

3-4 minutes which gives the player enough time to kill off the zombies, so they don't end up hiding away from them.

Q. How many lives should the player have?

One. It creates more of an impact and more realistic.

From this interview I have gathered a lot more information about Jake and his past with video games, he has given me valuable opinions and ideas to include. I'm glad Jake is familiar with Hotline Miami and parts of the zombies in Black Ops as this will make it easier to talk about and

discuss ideas on both games, however, the part of Black Ops Jake has played is similar to the actual zombies but is closer to Hotline Miami as it is top-down 2D, therefore, I can easily talk to him about Dead Ops Arcade and compare it to Hotline Miami as well. He also does enjoy the main zombie section of the game; I know this as we used to play it together all the time. I do like the idea of the rounds getting harder. Jake is disappointed that retro games aren't as common now and does like zombie survival games, I feel that from this aspect alone that my game design will be very successful for Jake as he has first-hand experience with both and combining the two seems like an elaborate idea.

Jai Alden

Jai is a good friend of mine and is my second stakeholder. I have chosen him as he enjoys retro games and likes the nostalgia feeling in games. I have known Jai for around a year now, we talk a lot about life in general and occasionally games, I came to Jai to be one of my stakeholders as I feel that my game could benefit from him and from his opinions. Jai is 17 and plays games quite often but isn't considered a full time or massive gamer. The interview process is below. I interviewed my stakeholders to make sure that they know what they need to help with and that this project is serious and very important to me, I wanted to rule out anyone that would not be helpful and not take it seriously.

Interview with Jai

Q. What experience have you had with gaming?

Wouldn't class myself as a massive gamer, I indulge in them quite often

Q. Console (if so specify) or PC?

PC

Q. What is your favourite genre of games?

Survival

Q. Have you played any retro games (original, classic games like arcade games e.g Pacman etc.)?

Mario, Sonic, Pacman, Dead Ops

Q. What do you do in spare time?

Swimming or gaming

Q. What are your ideas on how retro games are now not as common in 2019

Drastically dying out, which is a shame, we should pay amends to retro games as they all started from there and give Nostalgia.

Q. What is your favourite retro game (examples: hotline Miami, pacman, BO1 arcade, space invaders, Tetris, donkey Kong, Sim city)

Tetris, pacman

Q. Any ideas you would recommend recreating from games (map/ graphics design, weapons, points, currency, enemies etc.)

I would recommend, special enemies with special abilities and increasingly harder as the game goes on.

Q. What is your opinion on my game design

Seems pretty good, from the top of my head there isn't anything I would change.

Q. How should the leaderboards be counted (points, time survived)?

I feel that if its points based it is more rewarding for the player.

Q. Menu music and game music (should they be the same)?

Have different ones, make it so it is calm on the menu screen and hardcore music when you start up the game

Q. Sounds of weapons (all the same, or different)?

Different for each gun, it creates a better sense of realism in the game.

Q. Any specific themes you would like to see?

I think a suburban setting would be a good theme because I think it would appeal to the masses, it's something everyone can visualize.

The main purpose of my game is to act as a place for someone to relax and play something that will de-stress them and help them have an enjoyable experience to be calmer afterwards and more relaxed. My game could also act as a quick time filler as you can play it whenever you have free time, this allows a quick break to enjoy a gaming experience away from deadlines and work which is how my stakeholders will make use of my game.

Jai does like classic retro games a lot and thinks it's a shame that they have disappeared off of the face of the earth. I will try my best to bring back the retro gaming community by the game I am creating. Jai mentioned Nostalgia, which is just by sheer coincidence because that same emotion is what I am trying to provoke in my project so it is good that we are on the same table when regarding this aspect which means we can discuss future features that can help provoke it.

SUCCESS CRITERIA AND REQUIREMENTS

Requirements	Justification
Only one map	As stated by Jake (page 11) the game should have 1 map design 'to make the game more enjoyable' and concise to help keep the player interested and continue to play the game.
Menu becomes fully functioning and has buttons that bring the user to different pages	The user shouldn't just jump straight into the game, the menu allows information to be given in order for a successful completion of the game and so the user understands how to play
Menu has music in the background (soundtrack)	This was not a specific requirement from the stakeholders however, I feel that this would have create a more interesting game experience which is overall what both stakeholders want
Soundtrack will change when entering the main game	This would allow a more unique and interesting experience as also said above.
Main game will have a backing track of rock/hardcore music (maybe Carpenter Brut)	Overall improves the experience of my game on the end user and makes it more enjoyable
More than one weapon	Jake requested on page 11 to have a variety of weapons to keep the game interesting
Menu screen options of 4	On page 12, Jake decided that 4 options on the menu screen is enough to help give the information each user needs for when they play the game.
Have powerups in the main game	Jake brainstormed that having the chance of 2 powerups between level 1-5 then after level 5 the amount of powerups goes up, may increment it by one every 5 levels, however, will cap at 8 otherwise it will be too easy to survive. This does not mean 8 powerups per level, it means the chance is now 8/10 to get a powerup each level. (page 12)
Rounds/ levels 1-4 will have enemies that die after 2 hits	Jake proposed that between rounds 1 to 4 the enemies will die after 2 hits, one to weaken them and another to finish them off. This makes the health system easy to implement as an enemy's health can be 100 and the damage of the start gun can be 50.

The first level will have 13 enemies	Jake stated in page 12 that having 10-15 enemies in the first level will make the game easy to focus and enjoyable as new users can grasp the controls.
The starting weapon will be a handgun	A handgun is basic and low damage, Jake also recommended to have a knife that is a one hit kill (100 damage) to make the player risk their life more (page 12)
100 points per kill	Jake suggested having 100pts per kill as it is simple to use in calculations and also it is very easy for the user to see what they can afford. (page 12)
15 year old rating in order to play the game	Jake said that my game would be aimed for 13-18 year olds, so I decided to go in the near middle with age rating of 15 due to the gore and effects I will implement (page 12 is Jakes summary on why)
3-4 minutes for the killing of every enemy in the level until the next level starts	This makes sure the player is always on their feet about killing the enemies and isn't hiding, as the longer you wait, the more enemies will come. Jake references this in page 12.
Leader boards will be based on points	Jai has stated that a point based system for the leader boards will be more rewarding (page 14)
The game and menu will have different music	Jai requested to have different music for the menu and game, and I see this working very well (page 14 references this)
Each weapon will have a different firing sound	Jai explained how having a different sound for each gun will create more sense or realism (page 14)
Player will have 1 life or a few hits until death	Jake agreed this will make my game more realistic and harder. (page 12)

FEATURES OF THE PROPOSED SOLUTION

This section will explicitly state what will and will not be implemented in the development process, it will allow coherent and concise understanding on why specific features will not be included as well as reasons behind the ones that are.

Limitations refer to the features that won't be included due to them potentially being time consuming or other related factors, the below section will help to easily see the features I will not be implementing.

LIMITATIONS:

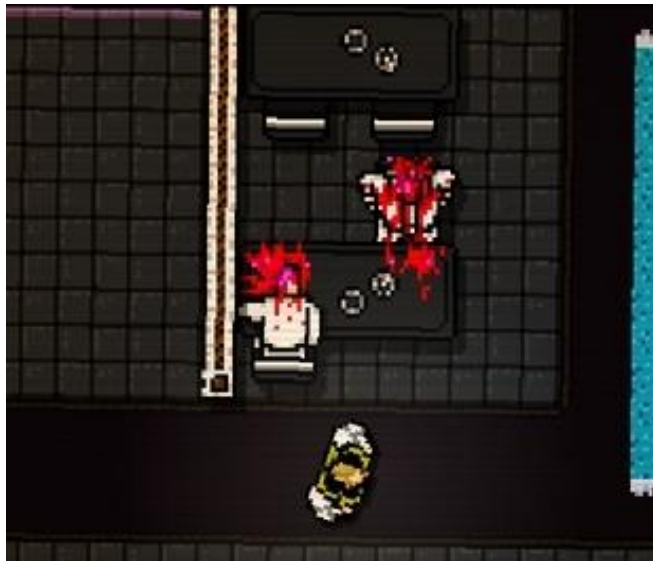
Between the two games I have chosen I have decided to leave certain parts out because of different factors. I will only be having one level as making many levels is very time consuming and doesn't add any more complexity to the completion of the game, one level is enough especially if it's a large one the user can navigate around, I have a limited time frame to make this project and prefer to spend more time on more important factors like the game design instead of map designs.

There is a lot of weapons in Black Ops and Hotline Miami, I will not be including all of them or even the number of weapons they have because I would be spending more time doing small scale graphic design than actually making the game and writing the code, as well as I am also not trained as a graphic designer, only as a computer scientist and programmer.

Feeding from the graphic designing, my sprites won't be as good as other mainstream games due to the sheer processing power I have and the experience as well as I do not have a whole developer team behind me to back me up and help with the designing of graphics.

I will leave out text speech (like in Hotline Miami) because it doesn't feel necessary and as important in my version of the game because you are a zombie survivor, although, I might decide to include a right-side chat box for the character to say some pre-existing lines of speech. This depends on the time allocated to each aspect of the project and if I have any left over to improve and add additional features.

I will be including blood splatter as my audience is high-aged teenagers (15-19) so having something to make the game more realistic about violence will help to make my game more of a success and more relatable to real life. This will also make the game more fun and interesting to play as it can be used to release stress and relax the users. I will try to make the violence match the game Hotline Miami's violence, however, this is a game a company made with a developer team, I do not have this but will try my best to make blood, organs and dead bodies look the same or similar to the ones in Hotline Miami like as seen below:



Overall, these limitations are because of time constraints, unforeseen consequences, efficiency, need ability or by me not having the correct expertise to implement them. If I feel that it won't be a good idea to include an aspect of the game/s then I won't, this will also help to make my game original and differentiate between games alike.

Hotline Miami does not have a reload feature, but Black ops does, and I might add this in if I have the time as I feel that it is a great feature to create suspense.

THE CONTROLS

The controls are a key important feature that will need to be implemented in order to have any type of interactive game in which allows the user to move and navigate around the game. I will use the typical A,W,S,D keys for left, up, down and right, allowing the character to move around the screen, this is because these are the most common keys to use for PC gaming. I will also have alternate keys to use, these keys will be the arrow keys to give the user alternate movement buttons depending on what they prefer.

Left hand mouse click will be for shooting as it feels easier to use than the right-hand mouse clicks, I have decided to also use the 'r' key for the reload feature in my game to reload the gun as it is the closest to the normal A,W,S,D keys for movement.

The 'Q' button will be used to switch the gun, again, this is next to A,W,S,D keys to allow easy transition between these set of keys to make sure it doesn't hinder the players actions too much in the game. If the switch weapon button was 'N' for example, this would create issues on finding the correct switch weapon button due to it being far away from the normal controls for movement, possibly making the player lose health to the enemies.



HARDWARE AND SOFTWARE

Platform/deployment: The game will be ran offline, as a Python file for Windows desktop computers. This is because this is the type of operating system that supports python easily and doesn't need a terminal to run.

There is specific software and hardware requirements in order for my game to function correctly and run, compile etc, however, some are listed below in order to allow my game to run for the end user:

Software: The software needed in order to run this game will be Python and Pygame installed and setup, the game is being made in Python and will need a 32-bit or 64-bit version in order to make it, having Python installed is needed as well as pygame, this is the only software needed in order to play the game, you will need some free space on the secondary storage to hold the game file, this is needed in order to run the game and load the files.

The development of the game will require an Operating System and an Integrated development environment in order to code the game and also debug.

Operating System: Windows XP/7/8/10 – supports Python more than other Operating Systems

Python programming – needed to run the code and play the game

Python libraries – Additional features to help improve my game overall, some examples of the libraries I will be using are: Pygame and time (for delays).

Hardware:

CPU 2GHz+ : Needed in order to run my game and render the graphics

Memory 1GB RAM: 1GB of RAM is needed at the minimum in order to store my game in the Ram when it is running.

Hard Drive 250MB+: My game won't be any more than 250MB to store on the hard drive.

Keyboard: Needed to input commands in order to play the game and to navigate the player.

Mouse: to use the other controls like shooting

Speaker: To allow an output of sound for the user so they can hear the sounds I have programmed.

Screen: to display the game so the user can play

GAME STATES

Game states are what will be in the code that will determine what sections to run and when to run them, I will have a menu state, a main game state and an end game screen state, when the user is on one of these pages the state for that page will be equal to 'True' and the others will be set as 'False'. This ensures that no unnecessary programs or code are running in the background that could potentially hinder the processing power.

SCORING

Players will get 100pts (points) for each successful kill of a zombie, these points will accumulate over time allowing the user to buy new weapons with them, but will have to be smart on how many they spend as these points will be the ranking factor for the scoreboard

The special zombies will be spawned very rarely but will have double health and damage but in return give 500pts to the player instead of the generic 100pts.

HOW TO WIN OR LOSE

My game solution has no way to win as you continuously play with infinite waves of zombies trying to attack you and you try to last as long as possible to reach higher on the scoreboards. Moving up the scoreboards will become harder and harder with each increment of time due to every wave containing more zombies than the next.

With each successful kill you will be awarded points, these points are what rank you high in the scoreboards page, you can use these points to buy new weapons that will help you last longer, which in the end, have a better return on investment (ROI) in terms of points.

An example of this process would be if you killed 10 zombies on round 2 and now have 1000 points and you spend 700 of the points on a new gun that makes you last up to round 11, if the zombie count is static at 10 per round you will now have a total of 9000 points (8000 more than before).

The losing factor of my game only appeals when the in-game player is attacked by a zombie and the health bar of the player reaches zero, this signifies death in my game and when this happens the user will be brought to the end game screen page where it will display the score boards and the players points. The score board will host the top 5 players and if the user is not up there it will display their points at the bottom allowing the user to determine how many they were off in order to be on the score board.

ADDITIONAL POINTS ON PROPOSED SOLUTION

Levels: The level count will be kept short to maybe one or two maps so I can then focus on more important features. More levels can be added at a later date if necessary or have spare time. This helps to allow the player to master each level individually without getting easily confused over a dozen maps for practice.

What happens when the game loads? When the user loads up the game, they will be greeted to a simple menu which will have explicit text on the buttons shown to easily navigate them. There will be on average 3 buttons, 'Play Game/Start', 'Scoreboards' and 'Quit'. 'Play Game/Start' will create a new session from scratch and this will be done every time this option is selected. My game won't save the state of the game when exited, e.g exit on round 20 and reload, you will start from round 1. Pausing the game will save the state. The menu will be designed to fit my game, with a background related to my game as well as some calm introduction music, when the game loads the music will change to hardcore 'Carpenter brut' music to get the mood of tension from the initial.

Collectables: I will have items that you will be able to collect in my game, for special zombies, when they die they will drop a random item in which can help to fight off the zombies, these items could range from another weapon to a special power against the zombies (like a nuke), this idea was taken from Black Ops zombies as certain zombies tend to drop items (nuke, max ammo, carpenter, double points and instant kill). Another form of collectable can be buying new guns at the armoury to help fend off from the infinite waves of zombies trying to eat you alive.

Theme: The level will have a unique theme to it, just like how in Black Ops zombies there's a theatre theme to one of the maps. Possible themes could include: A farm, A boat, a desert map.

I have these ideas so far because in modern culture, zombie films and games all base around urban areas like big cities and malls, I want to switch this and try a rural place like an isolated desert, rundown subway, farm or even boat as it is so remote no one has likely thought about that venue before, however I would have to set spawn locations on how the infinite zombies will appear if it is on a remote boat, maybe climb up from the sea?

Spawn Locations of enemies: Spawn locations are very important to get your head around key features include:

- Make sure the enemy doesn't spawn on the players head
- The player doesn't often (or at all) see them spawn in (ruins the realism)
- They do not get stuck after spawning
- Don't make spawn locations obvious (otherwise the player will spawn camp)

Spawn locations are needed in order to create new enemies when old ones die off, simulating an infinite number of zombies, ways to help with the spawn locations is to block off certain areas of the map from the player in order to spawn the zombies peacefully without the player interfering (like done in Black Ops zombies).

Types of system: A few examples of the types of systems I will be implementing in my game are collision detection, if the player has enough points for a function, kill count, scoreboard, physics, routing zombies to player (pathfinding) as well as the spawning system. These are a few areas in which will need to be self-contained in subroutines in order for easier debugging.

A Sequence Completion: Where in order to access certain areas or use turrets you first need to turn on (the power, turbine, add fuel etc....) this makes the gameplay more interesting and dynamic instead of running around forever you can unlock different areas with points, but some will require you to turn on a facility before you can buy access to certain areas

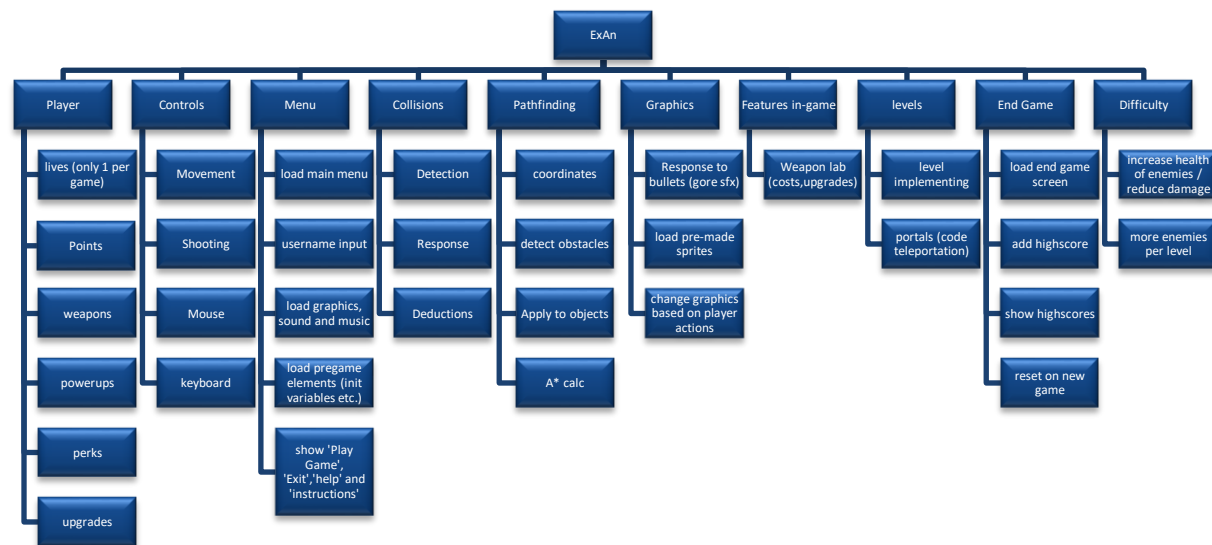
B. DESIGN

SYSTEMS DIAGRAM

The below step-wise refinement diagram helps to understand the sub procedures that need to be programmed in my game, I have split ExAn (my game) into multiple sub tasks in which in turn also have more sub tasks, this helps me to abstractly break down a problem into tangible solutions that can be coded up in turn.

Thinking ahead in terms of my project will help accurately depict what processes I will need to code up and potentially the sequence I complete them in.

I am using the step-wise refinement diagram to break down my project into many sub parts as I found this the most readable and easy to follow for my project, it helps layout everything I need to code and explicitly states the processes that link up through the decomposition of my project (e.g. points linking to the player).



I have created a systems diagram to help manage the problems that I will need to code up and also helps when it comes to testing as every component is self-contained and can be tested independently, system diagrams are clear ways to represent what task links with another tasks and what order to complete the tasks in, I found this type of design the most applicable for my project.

Taking each 'box' in turn and each role

ExAn is the game title for my project, under this heading has many subtasks that are all required for my game completion. The ExAn subtasks have further subtasks which are coded to work the underlying system of how my game will make calculations and deductions.

SUBTASK 'PLAYER'

- LIVES

The first subtask of 'player' is to make calculations on the lives, this will be relatively simple, it would be based on the game state variable (running or not) and if the damage dealt is more than the HP (health points) of the player, if either one of these is true it would present an end screen. The user has one life per game so deductions of lives wouldn't be necessary however I will need to code calculation algorithms to make deductions or increase health based on pickups. I have included lives in my game to help the player feel more at risk to the enemies and make sure that the player would make intelligent decisions more often otherwise they would become closer to death.

- POINTS

The points are what my stakeholder wants to be the main priority for rising the leader boards and showing the success of every game, the leader boards will take in multiple values to display such as the points, so it is crucial to get the points algorithm to a good standard. Points will be given from every successful kill, points will then be added to the players points and display it on the screen, the players points is what will be displayed on the leader boards, will need to also make the leader boards so they are in descending from players points. I had a good idea on how I wanted the game to work, with points being given for every kill, so I decided to include a points based system to allow that.

- WEAPONS

The weapons for the player will be unique and a small variety, I will need to include different sound clips for each fire of a different weapon and make them responsive to the 'fire' button when pressed. The justification behind why I've chosen different weapons is because if the player was using only one weapon the game would get repetitive after playing for a while, I wanted to add in more weapons to mix up the gameplay a bit.

- POWERUPS

The powerups are pickups that will be dropped every so often by enemies when they die, these will help the player to survive longer. I need to include the timings of drops, what enemies will drop them, when will they drop them and possibly base then on other variables (such as dropping an ammo box when the player is low on ammo). Each pickup of a powerup will have to be parsed into the collision detection algorithm in order to know when the player moves into one. Each powerup will have a different value, so getting the correct effect for that powerup is essential. Pickups are an additional feature I decided to implement as the reasoning behind it is so that the user feels that the powerups can give them a better advantage when the waves of zombies become uncontrollable. It gives the player a glimmer of hope, thinking that they will be able to escape alive.

- PERKS

Perks are similar to powerups however perks stick with you for the rest of the game, whereas a powerup is a short burst of effect (like more ammo). A perk would have an effect like 'double damage' when bought with points, this will stick with you for the rest of the game, so you only have to buy it once. Perks will be a good way to ensure the player still feels like they have a change of surviving. I will have it so if the player buys one of these perks, it will add it to their perk inventory (most likely an array) and deduct the costing points from the player, this is good as the points are which base the players position on the leader boards therefore, the more you spend the less you will climb the leader boards however, if you buy items to help you last longer you can then save up a lot more points.

You will only be able to buy one perk per game otherwise the player will be too overpowering, there will be a fixed size for the array and if you try and buy another it won't allow you as there is not enough room in the perk inventory.

The player will need some help to survive all the waves of zombies that come at them so I have chosen to add in perks that will stay with them for the rest of the game, helping them along the way.

- UPGRADES

The stakeholders didn't specify specific upgrades to have for guns, so I get to make them up myself and choose the new damage dealt and sort out the price of points as well.

This will be used for upgrading weapons which will create more damage, the upgrades will be a cost of points and will double the damage of the weapon, therefore, for each upgraded weapon I will have a subroutine in which would update the damage for that gun overall, each damage would be double of the original damage served from the gun, upgrading a gun creates double damage. Implementation of this idea I have chosen is the waves become a breeze for a little while and then get much harder as you progress, the upgrading machine will help the player in the harder waves.

SUBTASK 'CONTROLS'

- MOVEMENT

I have coded the movement for the player at which is responsive to the specific keys pressed by the player. If the user presses down on an arrow key the new coordinates are calculated and updates to the screen, making the player move. Each key that is let go stops the players coordinates moving, making the player static and motionless. I have decided to use the classic arrow keys in order to operate the movements of my player as these are simple and generic for games alongside the A,W,S,D keys.

- SHOOTING

Shooting will be another subroutine I will need to implement, making sure every fired shot from the weapon that hits an enemy successfully deduces the enemy's health and another calculation to remove 1 or more bullets from the players weapon.

- MOUSE AND KEYBOARD

The mouse and keyboard inputs are what allows the player to move and navigate around the game, the inputs for these peripherals will be included in the 'Movement' subroutine.

SUBTASK 'MENU'

- LOAD MENU

The menu will need to be loaded and presented to the player via outputting it to their screen, I will then change the menu state to 'True' showing that the menu is running, I will then also need to queue the music to play over the top of the menu whilst navigation from the user takes place. I will then need to decide the layout of buttons, this wouldn't count as coding, however, will need to identify the order in which I present the buttons and the order I write my code for them. This covers the other subtasks of initializing variables and states as well as loading sound, music and graphics under user input in systems diagram. I have included a menu because the player needs time to get used to the game and how the controls work, jumping straight into the game when loaded up is not ideal.

- **USER INPUT**

The username input will require an input field that the user can fill out and will be of static size, making a character limit to prevent stack overflows and ridiculous names in length. This input field will be presented on the menu screen potentially after the user has clicked the button 'Play Game' as this will start the game and change the game state, allowing the user to input a username. I will then later on refer to the username inputted for other subroutines such as the leader boards. I have chosen to include a user input to make the game more dynamic for the user and feel like they have control of the outcome of the game.

- **LEADER BOARDS**

The leader boards are to display the top players who have played my game and got the most points, this will be hierarchical in terms of points (Descending), showing the top player at the top. I need to pass by reference of variables information regarding what will be displayed on the leader boards page such as username, points and time survived.

- **OTHER BUTTONS**

I will have other buttons on my menu such as Exit, help and instructions as well as possibly a brief note regarding the position the player is in, allowing an overview of how the zombies came about, however, this doesn't need to be coded. The coded parts will be the menu buttons, each one bringing the user to a different subroutine with a different feature, e.g. Play Game will start the game whereas, 'Help' displays key information on how the game works.

SUBTASK 'COLLISIONS'

- **DETECTION**

Collision detection is to make sure the player conforms to natural physics such as objects not being walked through when they are in the way, I have had issues with making the collision work before such as the player hitting the object and teleporting to the start coordinates again (this is continued in the 'Portal' subtask), opposed to stopping when it hits the object's edge. To implement this algorithm, I need to take value of the corners of the objects and compare it to my player's coordinates.

I have coded up the collision detection and it is fully functioning; coordinates of the player and object are used in the subroutine 'detectCollisions' and if the result is True then the player and objects coordinates are passed into the 'render' subroutine through the variable 'collisions'. The new players coordinates are calculated by having the current coordinates minus the players move of x and y coordinates. I am going to implement the collision detection as it is a crucial part of my game in order to make it function the way it is supposed to, otherwise the enemies have no real threat.

- **RESPONSE**

The response would be to stop the player in their tracks because something is obstructing their path, in theory this is a lot easier to understand than to implement, but the main response factor would be to stop the player and have the player move around the object and take a different route.

- **DEDUCTIONS**

Deductions refers to when the player encounters an enemy the enemy will hit the player and reduce the health of the player, this means the enemy caused the player to get damaged and HP (Health Points) to decrease.

SUBTASK 'PATHFINDING'

- **COORDINATES**

I have to create a subroutine that will, based on the current position two objects, find the fastest way to the other object, this will be implemented in enemy – player pathfinding to allow the enemies to find the quickest way to the player in order for an efficient kill. I will take the coordinates of the player and of the enemy and pass them by reference into a subroutine in which would calculate the fastest path to the player.

- **DETECT OBSTACLES**

The enemies need to detect objects in the way and these need to be accounted for when the pathfinding algorithm is in process to route around the objects. Then apply this algorithm to the enemies

- **A***

A star pathfinding will most likely be the algorithm I use to find efficient paths, implementing this will be a lot harder than ever the collision detection and will most likely research a lot about the overall process for completion. Requires passing by reference the coordinates of the two objects, regarding any objects in the way also.

SUBTASK 'GRAPHICS'

- **PRELOADING**

Need to write code to initialize the sprites rendering and being responsive. This is done at the start of the game when the menu state isn't running anymore, and the game state is.

- **UPDATING**

The graphics need to be updated at specific times such as the enemies disappearing when killed, this is the removing of sprites and the respawn of it elsewhere. Need to add extra graphics responsive to player actions such as shooting in order to make a blood splatter. I have included updating of the screen every so often to help make sure that any actions the user takes that the game will update itself and display the changes to the screen.

SUBTASK 'INGAME FEATURES'

- **ARMOURY**

Some of the in game features will need to be coded as well, such as an armoury where you can upgrade weapons. This is the weapon upgrade system, which was discussed already, this is where the deductions and calculations will be lying. I thought of a interesting idea of including a place to upgrade your health, most games do not have this. I have decided to include it as I feel that as the waves will get harder the user will find it a little easier to hold off longer.

SUBTASK 'LEVELS'

- **LEVEL IMPLEMENTING**

Making the level design and map will be graphics design and won't need to be coded however, implementing it will be, I will need to import the map design and make sure each object on the map is responsive to collision detection – Or, I will add a background to the screen and have that as my map. Overcomplicating the map design isn't necessary but will have a few additional levels to make the game feel more dynamic in the maps and allow variance to the user.

- **PORTALS**

I have experimented with the players coordinates and the collision subroutine and now know how to now teleport the player as well as speed up the player as it passes through an object. This is what the portal will do, the portal will teleport the player to a different location on screen, this is accomplished by changing the coordinates of the player in the collision subroutine when it collides with an object, make this specific to an object like a portal and you have a teleportation system. This was accomplished before I had collision built in place as I was experimenting to make collision detection work.

The players coordinates would be passed into a specific subroutine for the portal and if the player is touching the portal (through the collision detection), the players coordinates would then be updated.

- **LOAD END GAME SCREEN**

The game state is updated to stop running and to start the running of the end game screen, would need to stop the state of the game and halt player input keys to prevent the player moving when dead. Will need to load in variables into the end game subroutine like the leader boards and the numbers for it such as number of kills, time survived and points.

- **ADD USER TO LEADERBOARDS**

I will need to code up a calculation algorithm of appending the username and relevant information to the leader board and making it hierarchical. To justify the reasoning behind this is because I needed a way to represent the players scores in a unique way that allows the user to see where they came in terms of the waves survived.

- **RESET ON NEWGAME**

When the game state ends, the game will terminate, and a new game will be ready to create. This is to make sure that the player cannot keep playing the same game that they previously died in. The reason behind me resetting the game after the player has died is to allow a new user to play their own game from the start without continuing the previous users game.

SUBTASK 'DIFFICULTY'

- **INCREASE HEALTH OF ENEMIES EVERY LEVEL**

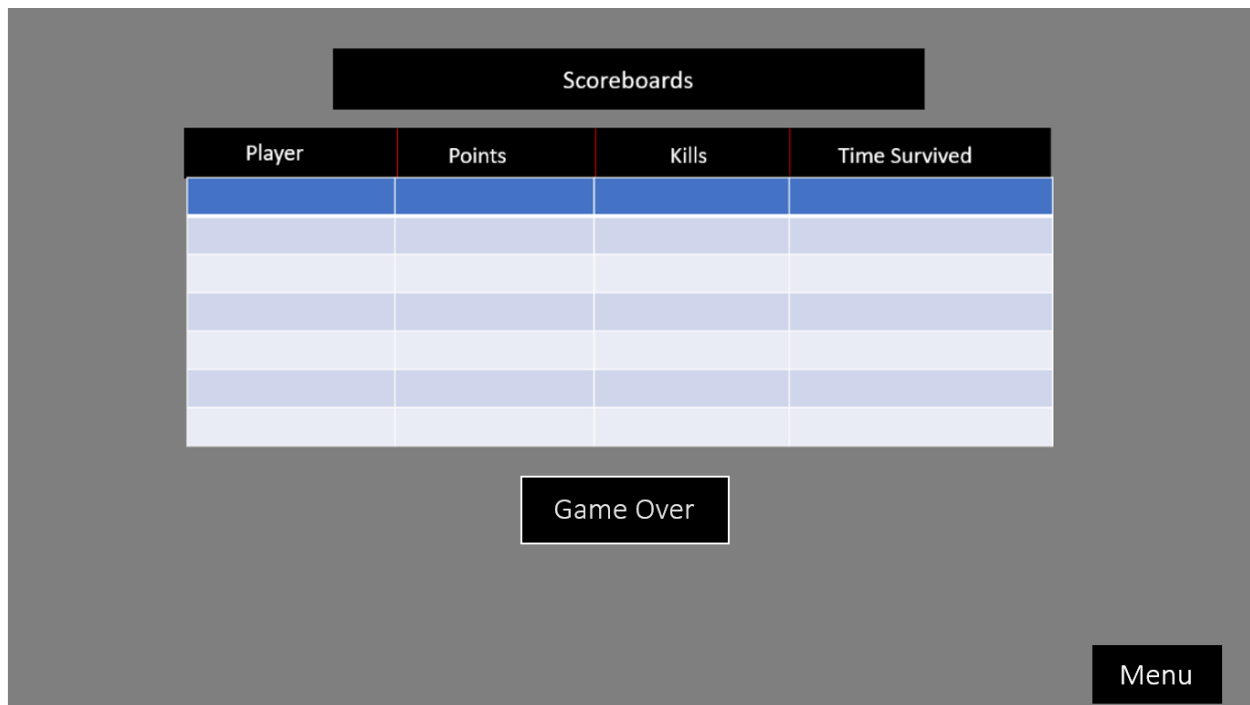
Each level will have enemies with more health, I can choose two methods of implementing this, I can either reduce the players damage over time or, I can just increase the enemies health depending on when the player gets to a specific level, I will choose the second option as the enemies health will be set in one variable and can be easily changed instead of reducing the weapons damage as this will be complicated when the player upgrades the weapon damage. To justify this process is because I need to make each wave harder than the previous in order to challenge the end game user, and I feel that this is a good way in doing so.

- **MORE ENEMIES PER LEVEL**

The first level will not start with one enemy, the first level, like stated by the stakeholder Jake, will have 10-15 enemies. Then will increment every level. Incrementing by one enemy each level is a very slow process, I will choose to spawn 2 or 3 additional enemies per increment of a level with the first level having between 10-15 enemies. Between level 5 and 15 a 'BOSS' will spawn and will have more health than the other smaller enemies. This is random between level 5-15 so the player won't be able to predict when he comes, creates more of a challenge.

PROPOSED SCREEN DESIGN AND USABILITY FEATURES

SCORING



The above image is what the scoreboard will look like for the ending screen of the game when the player dies. It will be hierarchical and have the player with the most points at the top, this suits the layout well as the players name will be at the front and will have the setting factor next to it, making it easy to identify the factor which ranks the players higher in the list.

All columns will match up and will be easy to see and have a decent design to it, allowing the user to easily read off of the scoreboard to see where they came, the players name will be the name inputted at the start of the game after clicking 'Play game'.

This is also the 'game over' screen, where the user is taken when they are deceased, the game state stops when reaching this page and the user is greeted with a 'game over' message as well as the scoreboard. There is no way to actually complete my game, my game will go on forever until you lose, fighting against an uncontrollable wave of undead.

The banner uses contrasting colours for the colour coordination of the game, making it unique to the 'dark feel' of my project. The music will change when this page is displayed to relieve the tension of the game and chill the user out. A menu button will also be displayed in one of the 4

corners, allowing the player to revert to the menu to choose their next option. The 'Gave Over' button has a white outline as it is considered more important than the menu button.

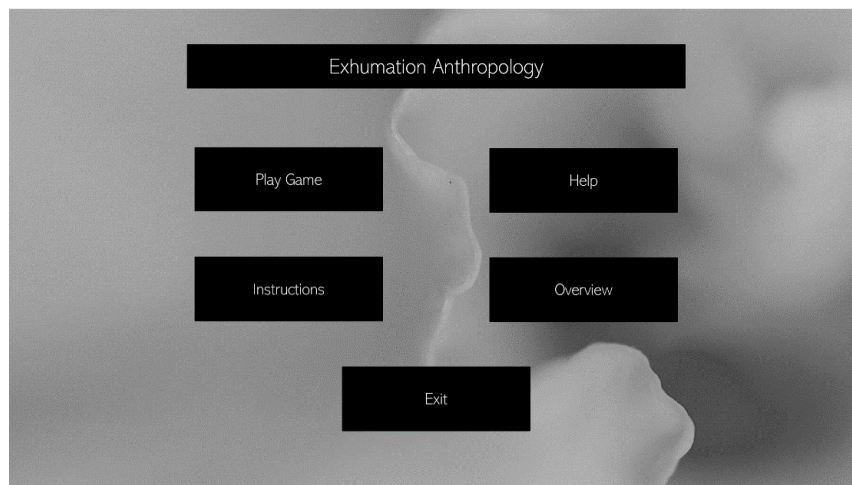
I created this simple design, so it fits well with my project and doesn't throw the user off when viewing it as it is simplistic. No stakeholders explained how they would like the end screen, so I decided to use my thoughts and come to a suitable conclusion on what would suit my game.

I may decide to change the background setting when it comes to implementing as I feel that this could be too vague and bore the user a little. I may include a background image to help set the mood of the game to the users when they die, as when they die they will be frustrated and want to have something ideal to look at, hence the simple design and possible background idea.

The scoreboard ending design will match that of the menu design, conclusively adding the same structure to the designs and similar comparisons. This is key as it allows the user to understand that the game is encapsulated in between the same designs making it look implicitly self-contained and organized.

I have created this score board at the end of the game to allow the player to see where they ranked. I feel that this is a good way to see the players rank and how well they did. This is common in video games and is a good way to portray the game and the players in it. It's an abstract way of looking at scores and helps structure the overall process of the winner and players that come after the winner be more readable.

MENU



The menu design is simplistic and contains the same variety of colours as the scoreboard design does, allowing colour coordination. The buttons will be black with white text over them explaining what each button does, like shown above. There will be 4 main options on the menu screen which are in pairs of two under each other, exit is at the end, showing least significance

when loading up the game, allowing implicit hierarchical value. My project title will be at the top, over all the buttons, there to greet the user and remind them of what the games called. The menu will have calm music being played over to add a sense of peace for 'the calm before the storm' effect as the user won't know what is going to happen next as well as also adding the effect of calmness before the zombie outbreak.

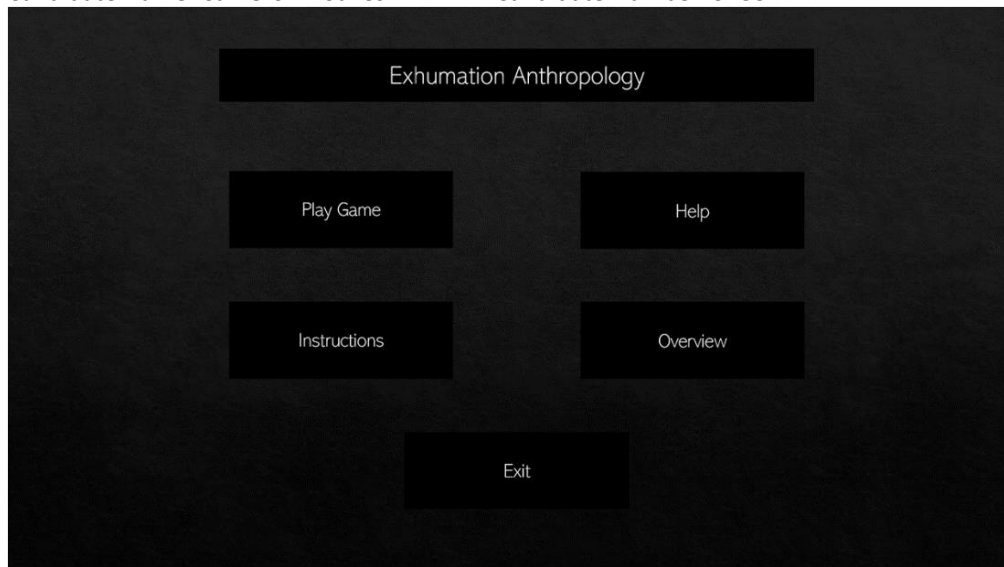
The background is of smoke, it has no resemblance to my game however, I do feel that it gives off the correct vibe for my game as its eerie and mysterious, making the user unaware of what will happen, which is creating suspense before the game has even been played.

The reason behind why I've included the menu is because the menu is an important feature in a game, it prevents the game from running until the user is fully ready to play. The menu is a screen where additional information can be displayed, allowing the user to know more about how the game operates, this is key for a successful game, where the users understand it throughout and the concept behind it.

The justification behind why I have used the colour pallet of mainly black, white and grey is to set the mood of the game when the user loads it up for the first time. I have chosen these specific colours for my game colour coordination because it represents how the world has went to rubble from the apocalypse and everything is dark as there is barely any hope left as you are one of the last surviving survivors out there.

Further justification is for how it will affect the end user, the user will see a clear and organized menu when they load the game, giving them time to get familiar with the buttons as well as get ready for starting the game. The menu is very concise in what it displays and is fully functioning and allows the user to also get an understanding of the concept of my game as well as including the controls.

I have created two menu designs, the one above is the ending one I have chosen however, I did also like the look of the one below I made originally:



PERIPHERAL INSTRUCTIONS



The peripherals are a key aspect of my project as without them the game wouldn't function. The user wouldn't be able to carry out instructions like 'move' or 'shoot'. As my game is designed for PC's the peripherals will be a mouse and keyboard. The movement commands will be A,S,W,D like normal and to shoot will be the left click of the mouse. 'P' will be to pause the game, halting all game states and allowing the user to take a break. The 'R' button is to reload the current gun you have, and 'Q' is to switch the weapon. This layout and explanation are for my instruction menu on the menu page of my game.

I have made the controls the easiest to follow as my game is for users aged 15-18 which makes sure they can understand the controls in order to play the game. This is why I have detailed the buttons and arrows pointing to what buttons do what when pressed. Keyboard and mouse is

black, matching my game colours and red for the arrows as it contrasts well and also because it suits the topic of killing zombies.

The justification behind the peripherals I have chosen is that of classic PC games, A,S,W,D are common keys to use to navigate the player so I decided to keep it simple so the end user can play the game without any hassle. 'Q' and 'R' are chosen as other inputs because they are next to the movement keys, allowing quick maneuver to different keys for different controls of the player and game. Left click for shooting is because it feels natural to the player to click a button that has more friction on the input, giving the impression of 'pulling the trigger'.

I have given the player the option on what controls they would like to use to control the player, it will be either A,S,W,D or the arrow keys, depending on what the user prefers.

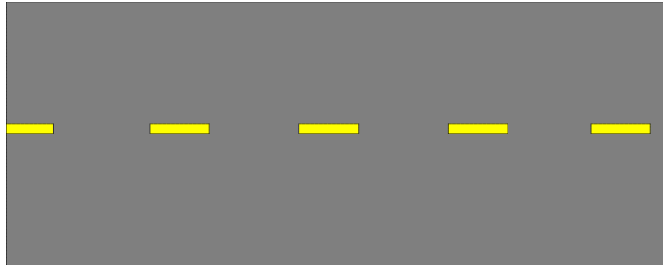
PAUSE MENU



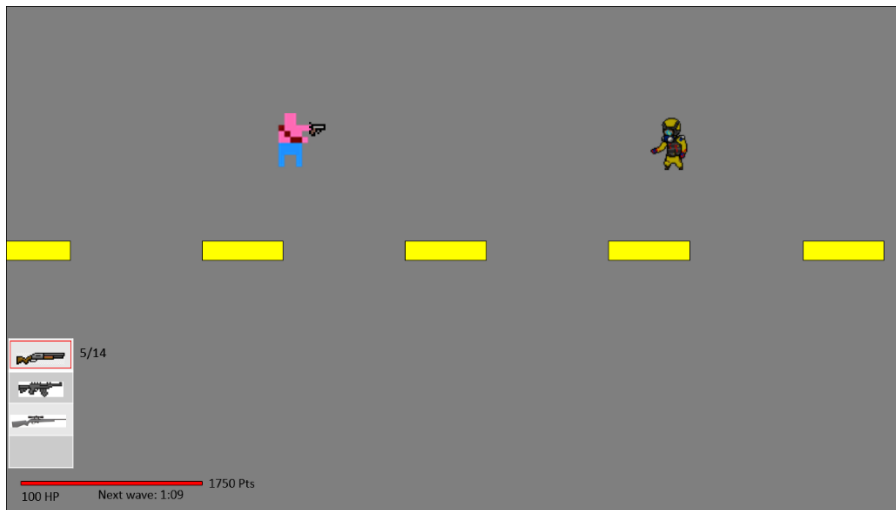
This will be the pause menu for when the player pauses the game. It's simplistic and has the same colour tone as the game, the buttons are with white outlines and are two options, one to resume and one to quit the current game. It is simplistic because I don't want to throw off the user and make it too complex that they get confused. This pause menu pauses the state of the game, allowing the player to take a break.

The reason why I have included a pause menu is because after a while the player will get eye strain and maybe temporary RSI (repetitive strain injury) in their hands and will need a short break to rest their mind or go to the toilet, the pause menu allows short or long breaks from the game, the game state is still running, just paused, so when you resume, it continues from where you left off, this allows the user to jump straight back into the action.

SETTING/ BACKGROUND



This is the screen the player will be greeted by after they have launched the game from the menu screen. The menu state stops running, and the game state starts. This is the background that the player and enemies will be seen on. Below will have the same background but with the additional features like inventory, the one above is just to show the background. The background is simple and will be easy to implement. I'm not going to spend much time doing graphics designing (mentioned in 'limitations') as my time can be spent more wisely on other functions to code. The reason behind this background is that I feel the player will be able to resonate with the background as it is a classic motorway road where you are in open space, unaware of your surroundings. This creates the illusion of enormity and unsure where the enemies will come from.



This is roughly what the game screen will look like in the end. As you can see, it isn't much more complicated than the background itself, the gun currently equipped will put a red square in the inventory (see the shotgun above for reference). The player will hold a placeholder of a gun, this design will not change when a different gun is equipped as this will be more interactive designing and this is not what I want to focus on as much.

The inventory is made up of a table in which can hold up to 4 items, when the player presses 'Q' the weapon will switch to the one below, if the last one is currently selected and the player switches the weapon it'll bring it back up to the first inventory slot.

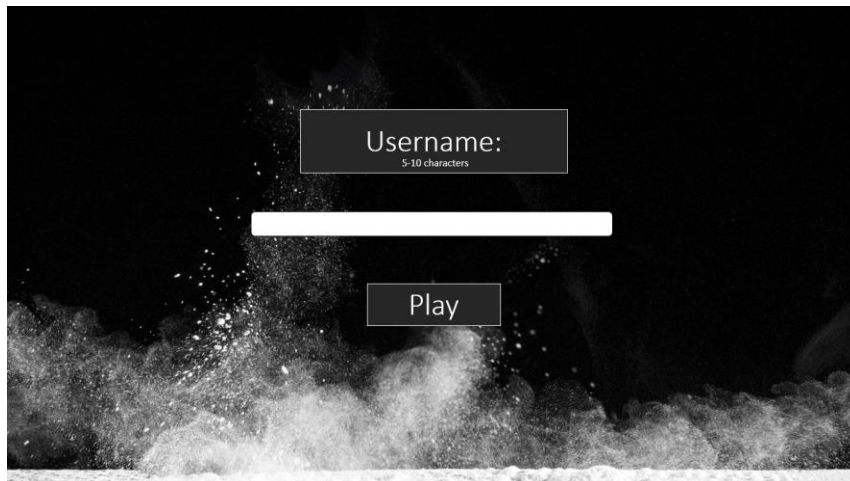
The health is represented above as a red beam that goes across the screen horizontally, this is a classic example of health in retro games (like Fallout 4, however, Fallout 4 is not a retro game, it is designed to look like one). The players health bar will go down after every successful hit of the enemy, and I also will implement a counter underneath which gives the total amount of health left until the player is dead (bottom left).

Adjacent to this is a 'next wave' counter, continuously counting down the seconds until the next wave appears. This helps give the player a time frame in which to kill the enemies, always keeping them on guard and in the 'zone' of the game. After a while there will be so many zombies that the player won't be able to keep up with the time frame and eventually die from hordes of zombies, this is why my game is non-winning and is a 'how long can you last' game.

Justification for this background setting is from my stakeholder, Jai, he wanted a suburban background as it resonates well with a zombie apocalypse, so I chose this design as it seemed suitable and prevalent. The user will find this setting interesting as the video game 'The Walking Dead' by Overkill has a trailer in which is based on a silent motorway.

Having a suburban setting will help the user relate to their surroundings as most people live in a suburban area.

USERNAME INPUT



The username input screen is what greets the user after the menu. This is a way for the user to be creative and choose the players name. This allows a more dynamic approach in the game as the players username will be presented instead of a static value like 'player1'. I will include a text bar which will allow the user to enter their chosen name, there will be no restrictions in the name however, there will be for the length, the length must be 5-10 characters long otherwise it will be a very long user name and will go off the screen.

When this page is loaded the player will know exactly what this page is for as you have a box heading of 'username' above the text input. There is a button underneath the username input, this will start the game after a username is typed. If a username is not typed in the box, then the game won't start.

The design of this page is like the pause menu, just with different buttons, this is to be consistent with the colours and designs used in the game to be able to match the type of design my game is. If the name is valid the player, then goes into the game.

The justification on why I would use this is to make my game more dynamic and allow the user to feel that the game evolves around their actions in the game, making a better overall experience.

SPRITE SHEET



These are my sprites used for the player, this allows more of an illusion of dynamic motion as depending on the user's direction it will change the sprite. If the user presses 'W' the players sprite will move up the screen and the sprite will change to the 'Up' sprite. I made these sprites myself using Pixel Art Studio.

The reason why I am including this in my project is, so it feels more real and easy to identify which way the player is going. It allows the player to see how much time I have put into this project.

SUMMARY AND STRUCTURE

This section will give an overview of the information given above as well as other processes that happen, decomposing all important factors into key bullet pointed features. These are given in order.

Player name and Menu

- Initialize the menu, buttons and the calm music
- Load the initialized
- If user clicks 'instructions', initialize instructions and load them to screen
- Or the user clicks 'help', initialize help screen and load it
- If user clicks 'Overview', load the text and Overview screen.
- If the user clicks the 'exit' button, exit game.
- Load user clicks 'play' load the username input page
- Check user input between 1-10
- If valid, proceed into game

Main Game

- Initialize the main game, variables and sprites
- Initialize the wave timer, player health and enemies
- Initialize the different music and load it into memory
- Set players score points to zero
- Load pickups (guns, powerups etc.)
- Load main game screen
- Load background/ setting
- Load sprites
- Load music and the rest
- If a bullet hits an enemy, reduce enemy health and ammo.
- If an enemy hits the player, reduce player health.
- If player pauses, initialize pause screen and load pause screen
- If player health is below threshold, stop main game
- Check the outcome of the game for any other additional actions like pickups, if outcome is positive, action is true and do necessary functions

Motion/Navigation in Main Game

- Initialize variables
- Check if player has used peripherals, if so, change sprite and carry out the direction of motion.
- If player moves the mouse the in-game cursor updates and moves it on the screen

Peripherals that aren't motion

- If player switches weapon through 'Q', update screen to new value.

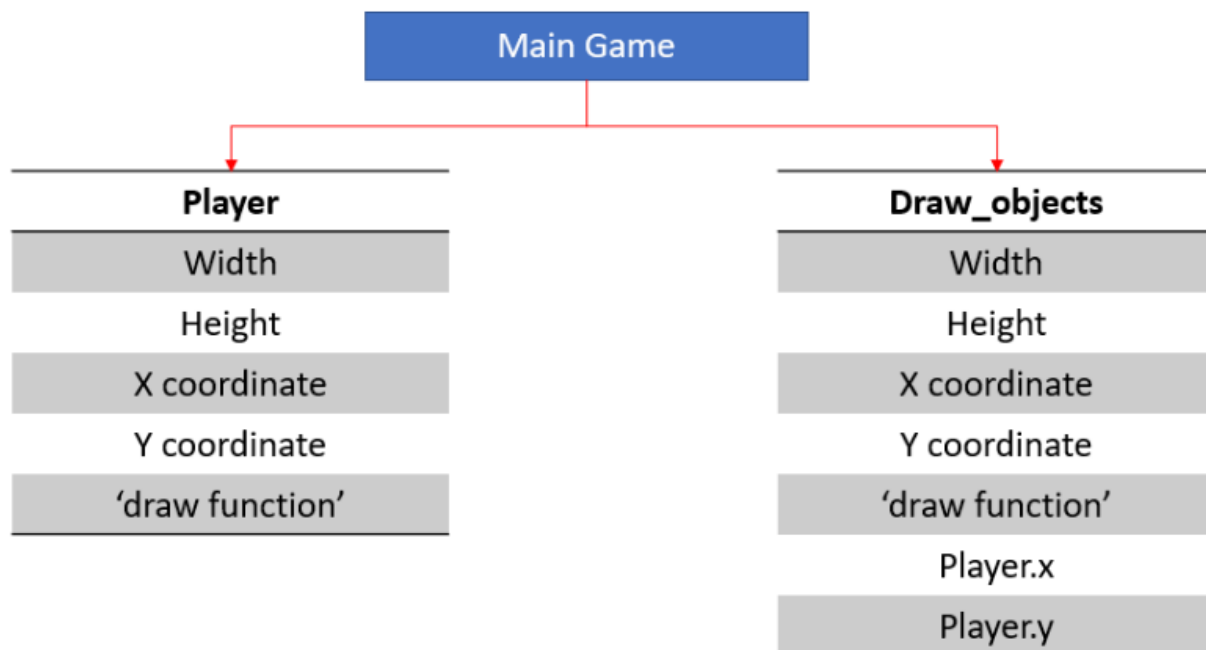
- If player hits pause with 'P', load and init (initialize) the pause menu.
- If player taps 'R', reload the current gun, reset ammo to full magazine, update screen with new rendering.

End Game Screen

- Initialize the end game screen, update scoreboard, then load end game screen with updated scoreboard
- Stop current backing track
- Stop game running and all objects in main game sequence
- Initialize and load the 'menu' button and the change in music.

CLASSES

- 'Player' Class
- 'Draw_objects' Class



Both of my classes are to help render objects to the screen, the 'Player' parent class is to allow the rendering of the player to be self-contained and separate from the other objects, I have struggled to implement this before and under the same superclass would join the player and other objects (like crates) together, using this method I found it to be most efficient and self-contained, allowing each instance of the super class to be rendered on its own, creating a easier process for rendering and debugging later on.

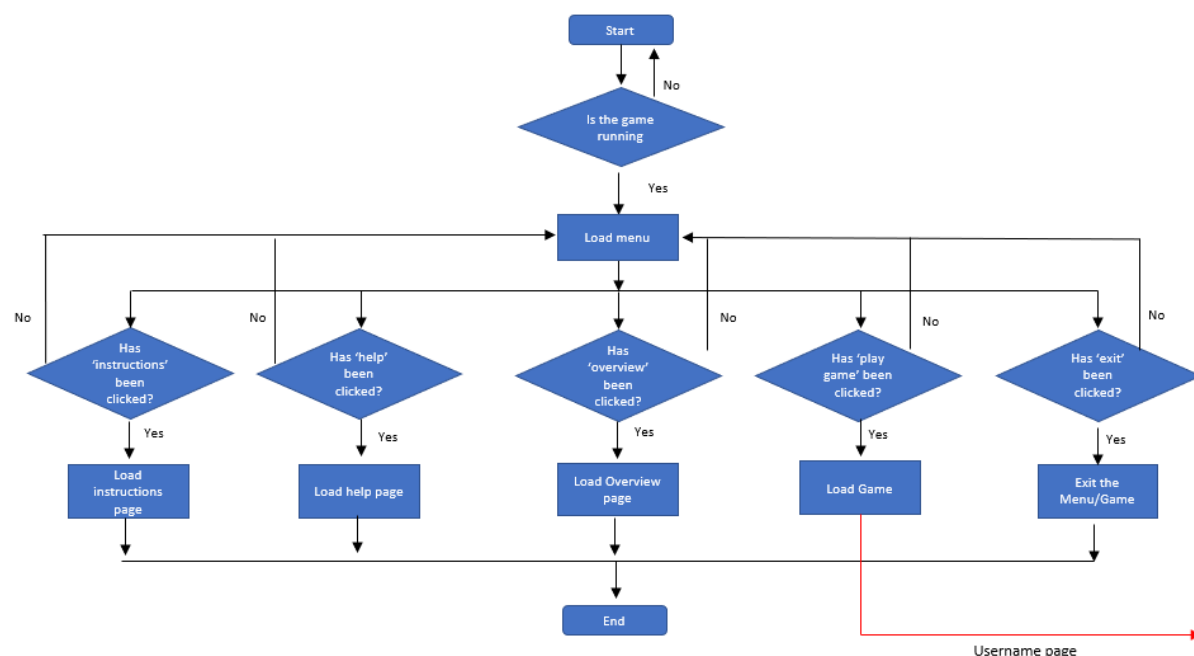
Each of the above super classes have two functions each in them, as these parent classes are self-contained, the functions inside them are both called the same, no variance of the names of `'__init__'` and `'draw'`. The init subroutine is to initialize any variables within the scope of the program and bring them into the scope of the class, making them local. These are then attributes of any instance in the class (which is only 1 in `'player'` and multiple in `'Draw_objects'`). One instance variable of player is rendered from the class `'Player'` and many instances of the class `'Draw_objects'` are rendered as I will have many objects to render to the screen for the player to see (crate, buses, perks etc.)

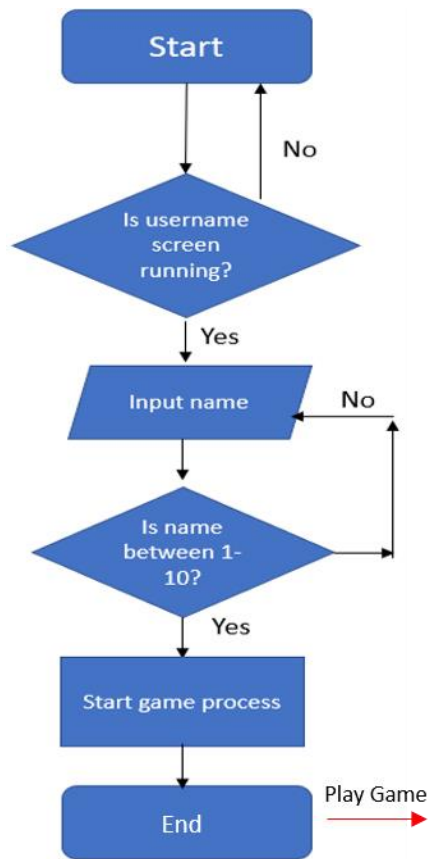
The reasoning behind why I have chosen to use two classes for this idea is to allow concise reading of my code and to remove code duplication and improve the overall integrity. This has no direct impact on the game or end user however, making my code more readable is good for when I must debug and go back through my code.

ALGORITHMS

I am going to structure my algorithms below in a flowchart, this allows easy identification to see visually how the program and algorithms will work, it is easy to see the flow other to pseudocode where that is relative to Computer Science only, flowcharts everyone knows how to interpret and allows easy readability. This is the reasoning behind why I have decided to choose flowcharts, they create a visual flow of an algorithm.

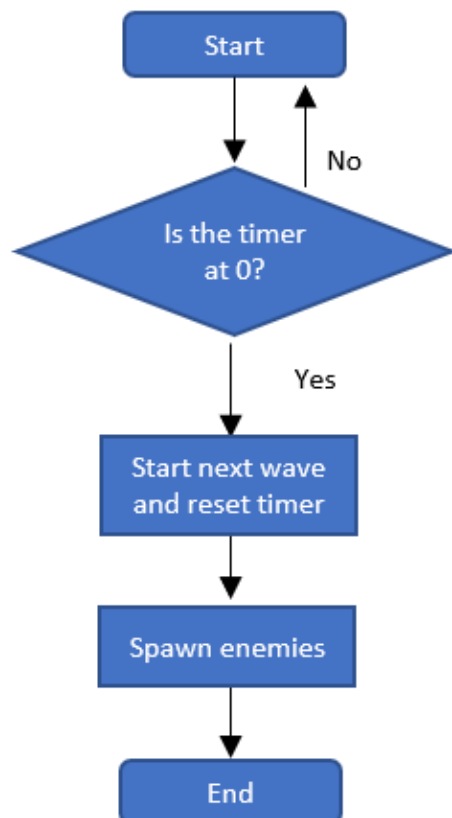
Menu Screen



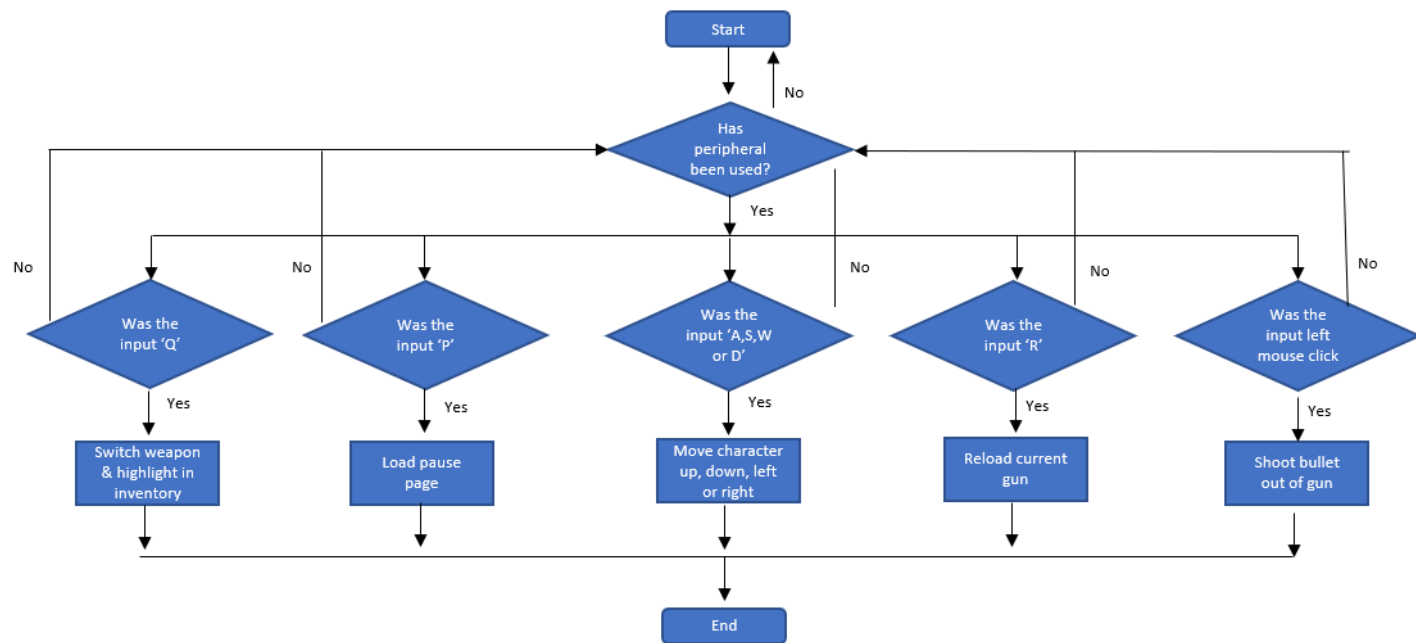
Customizable Player tag**Bullets, collisions and scoring**



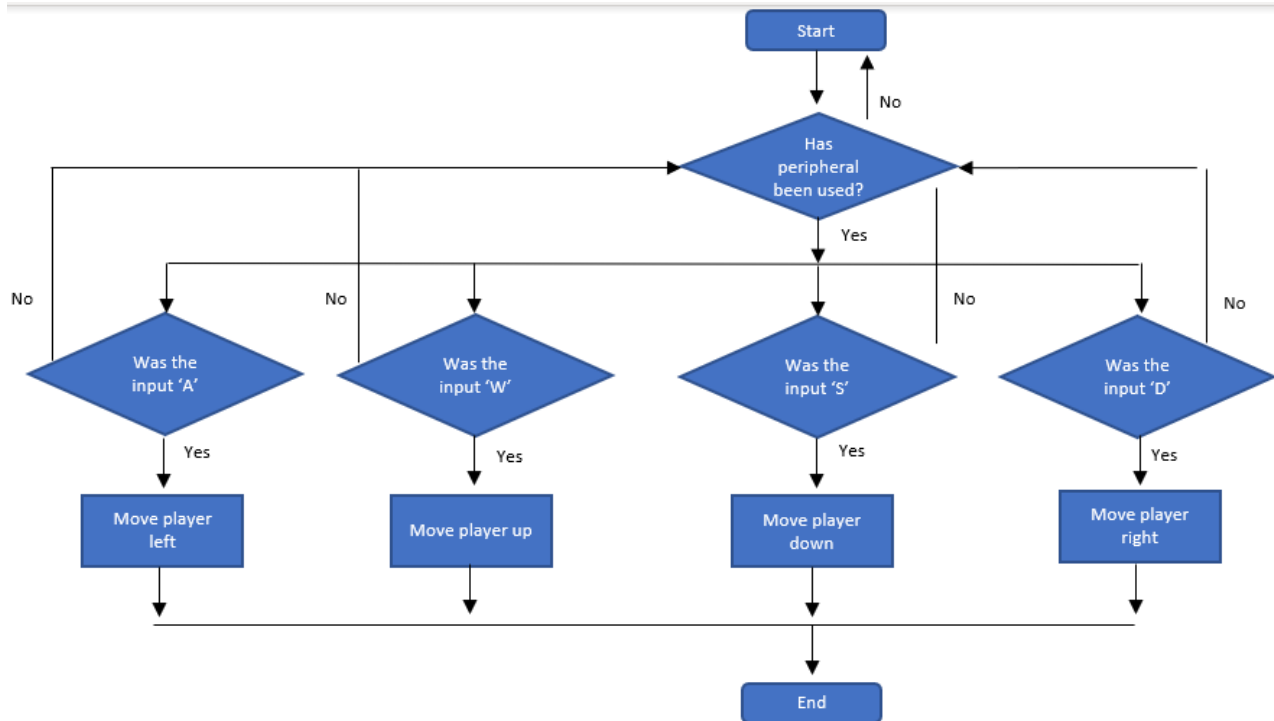
Next Wave Initialization



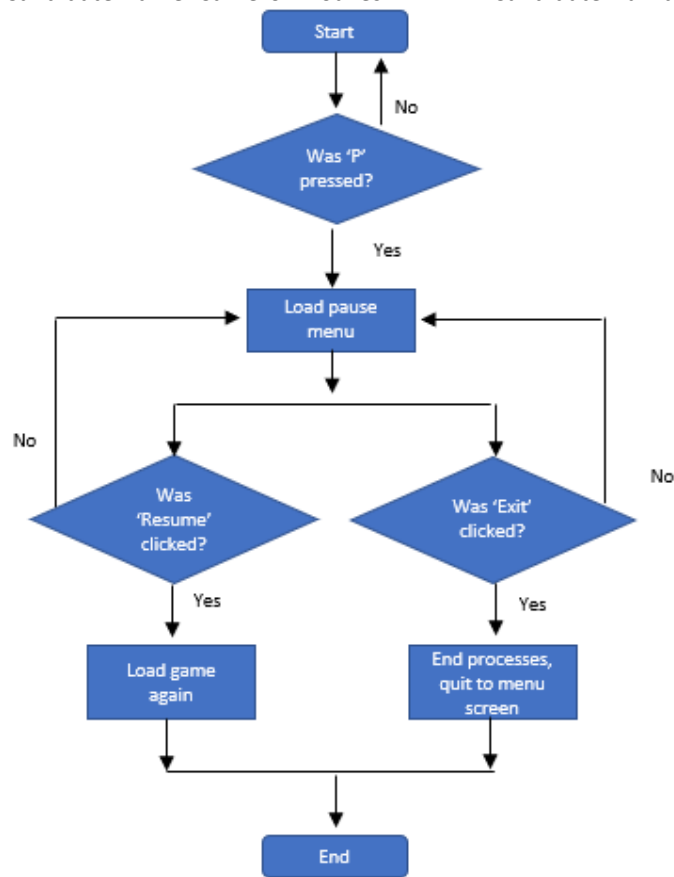
Player Peripherals



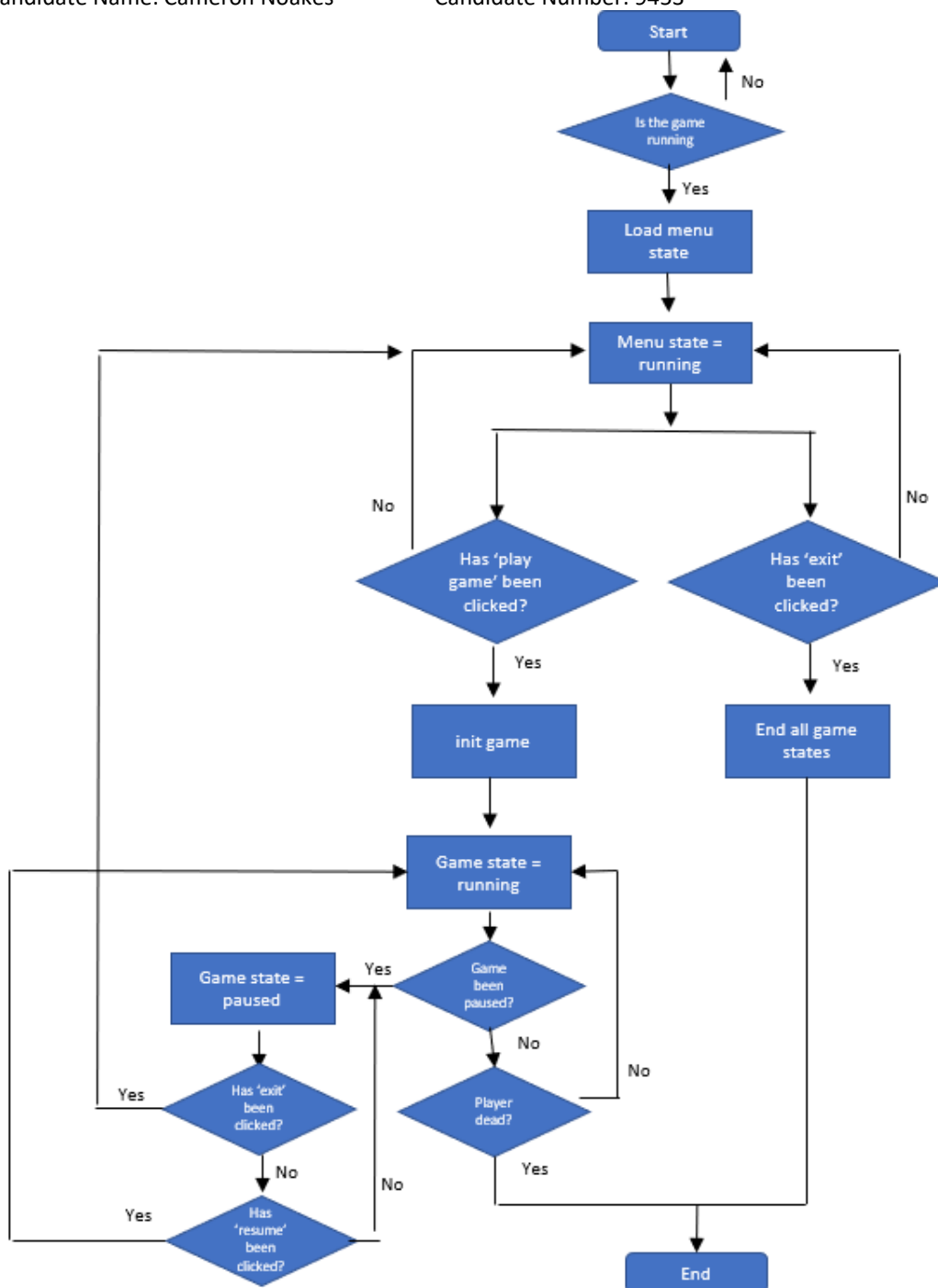
Movement (inside peripheral diagram of if input was A, S,W,D)



Pausing the game



Game States



HOW THE COMPLETE SOLUTION WILL WORK

All the algorithms above neatly fit into one another, the 'end' button process in the flowcharts is simply the end of the flowchart for that particular algorithm, in practice it would flow into the next algorithm to use. Once all these algorithms are concatenated together they will form the overall process of my complete project game and are then taken in turn to ensure that the correct algorithm is running until moving onto the next continuous one.

TESTING SELF-CONTAINED SYSTEMS WITH DRY DATA

Menu

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
A,W,S,D	Invalid	Not in main game, so there is no use to make them work
Arrow keys (left,right,up,down)	Invalid	(same answer as before)
Left click on 'play' button	Valid	This shows that the game will launch after 'play' button is clicked.
Left click on 'help' button	Valid	To check if the help page loads as it's supposed to
Left click on 'instructions' button	Valid	To check if the instructions page loads
Left click on 'exit' button	Valid	To see if my game window closes
Left click on 'Overview' button	Valid	To see if my overview page loads into memory
Right click	Invalid	No action when on menu
'R' (for reload)	Invalid	No action when on menu
'Q' (for weapon switch)	Invalid	No action when on menu

Username input page

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
Username less than 5	Invalid	This is to make sure they enter a name
Username between 5-10	Valid	Good length for a username
Username over 10	Invalid	Accurate length to allow it in leaderboard row
Username of numbers	Invalid	To make the username readable
Username of pasted picture	Invalid	Only text for the username to make it classic and original

Username of string	Valid	Username must be of string to be accepted.
--------------------	-------	--

MAIN GAME

Movement and peripherals

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
'A'	Valid	Controls to see they work in game
'W'	Valid	-
'S'	Valid	-
'D'	Valid	-
'R'	Valid	Allows a button to be pressed to allow reloading of a gun
'Q'	Valid	To allow quick switching of weapons
Left click on buttons	Valid	To allow the player to shoot the weapon the player is currently holding
spacebar	Invalid	Top down game so the spacebar for jump isn't needed.
A number	Invalid	No numbers are shortcuts
Left arrow	Valid	Movement of character
Right arrow	Valid	Movement of character
Up arrow	Valid	Movement of character
Down arrow	Valid	Movement of character

Pause Menu

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
'p'	Valid	Displays the pause menu for the user to have a break from the game.
Left click on buttons	Valid	This is the action to take if you want to navigate pages through the buttons
Left click when no buttons	Invalid	Left clicking on empty space performs no action

A number	Invalid	Numbers are still invalid
----------	---------	---------------------------

End Game Screen

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
Left click off menu button	Invalid	Left clicking on empty space doesn't do any function or action
Left click on menu button	Valid	Brings the user back to the menu screen
A,S,W,D	Invalid	When not in game these controls are vacant
A number	Invalid	Also, invalid due to the same reasoning previously stated

Draw to screen

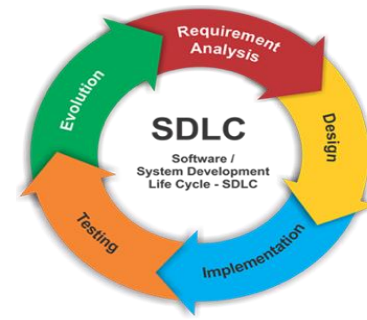
<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
Load Game	Valid	To make sure the game loads into memory
End Game	Valid	To make sure the game closes when it is supposed to
Hits enemy	Valid	Checks the collision detection algorithm to see if it works accordingly.
Spacebar	Invalid	No action

Sound Effects

<i>Test Data</i>	<i>Type</i>	<i>Reason/Justification</i>
Load Sound	Valid	To make sure the audio for my game works and sounds during the running of my game
Load other sounds over the top (sfx)	Valid	To have an impact on what the user will experience
Change music based on screen	Valid	To change up the pace and tone of my game
Stop Music	Invalid	The user can always mute their device, not going to implement a 'stop music button'

SDLC PROCESS

I will be using the spiral model as my software development lifecycle mainly because it actively has user input which leads to the next cycle, I complete one cycle and the stakeholders give me feedback on what I have made already and that feedback feeds into the next iterative cycle, constantly making sure I am on target and the stakeholders are happy with the development of my game.



I decide to not to make prototypes, however, after each cycle I have implemented I will be sure to show the stakeholders the changes I have made, more of an updated version of the original than a prototype. I would consider this a larger project than usual, as making a game has many factors to consider such as audio and visual as well as peripherals to manage, making the spiral model very suitable to the project currently undergoing.

BETA TESTING

This test data is for beta testing to make sure the game functions correctly for every piece of code I have written for the self-contained subroutines.

GAME, MENU AND USERNAME SCREEN

Test Number	What is being tested?	Input	Output/ expected outcome	Reason/ Justification
1	movement	A,W,S,D are pressed	Player moves on screen	To make sure the player can move around the screen
2	Enemy movement	Player moves	The enemies follow player	to check if the enemies can find the player on screen
3	Collision detection	Player moves into object	Player stops and must go around object	Tests the algorithm of collision detection

4	Fire bullets	Left click	Reduces ammo and if bullet hits enemy, reduction on health for that enemy is calculated.	Player must be able to shoot and kill enemies
5	Moving the players cursor	Move mouse in any direction	The cursor should update its position and move to where the mouse now points	This allows the user to see where they are aiming
6	Menu buttons work correctly	Clicking on each button on the menu screen	Brings to specific page based on what button is pressed	This makes sure the menu is fully functioning and won't break
7	Switching weapons	Q	The red highlight box moves to next weapon in inventory and applications of the new gun are used.	Fats switching ready for the user mid game
8	Username input screen	VALID: 5-10 chars as a string – INVALID: Not a string or <5 or >10 BORDERLINE: 5 letters	VALID: Enters the player into the game INVALID: Screen does nothing until correct format is used. BORDERLINE: Accept and enter the game	Makes sure that a valid username is entered and not a fake immature one is entered instead.
9	Pausing feature	VALID: P in game INVALID: P on menu or end game screen	VALID: Pauses game and loads pause screen INVALID: No action occurs	Should pause the game to give the user a break
10	Spawning enemies off screen then	No input	Enemies walk onto screen	Allows a better feel for the user as the

	move onto screen			enemies don't spawn in right in front of the player
11	On menu if the exit button presses, close the window	No input	The game should close	Makes sure the user has full control of how the game functions as an end version
12	If the player is seen on screen	Enter username and click play button	Player should be rendered to the screen and be seen	The makes sure the render functions work and the user can see their character
13	Level increment	Timer runs out	The next level starts, and more enemies appear on screen	Justifies the level/wave choices and makes sure they work
14	Timer for levels	Player starts game	The timer should count down to 0 and when it reaches 0 next level starts	To show the user how long they have left to complete the wave

POWERUPS AND WEAPON UPGRADES

Test Number	What is being tested?	Input	Output/expected outcome	Reasoning/Justification
15	If powerups/perks appear	Play game long enough	The powerups appear on screen and the player can pick them up	The user must see the perks or powerups in order to

				interact with them
16	Weapon upgrades	Player places gun in upgrade machine	The gun gets double the damage as previous damage; enemies die quicker	Is to allow the player to impact double damage
17	BOSS is spawned	BOSS spawns randomly between level 5-15	BOSS appears (larger enemy and more health)	To make the game harder for the end user. To make sure the Boss spawns.
18	Delay of enemies when game starts	Player starts new game	Slight delay before enemies start attacking player	To give the user a few seconds to prepare.
19	Points counter	Kill enemies	Points should increase at a rate of 100 per kill	Points are used for different actions in the game, need a way to pay for the items.

GAME STATE TESTING

Test Number	What is being tested?	Input	Output/ expected outcome
20	End game state at correct time	Choosing to display a different screen	Each screen should load in time and same with the game states
21	Menu to playing game state	Clicking the 'play' button on username input screen	Menu state stops and playing game state starts at correct timing
22	Exiting the game	Clicking the 'X' button in top corner or 'exit' on a menu screen	All game states stop and the game shutdown, leaving no states running in the background.

23	Playing state to end game state	Player dies	Playing state stops and the end game state starts
----	---------------------------------	-------------	---

SCORING ALGORITHM TESTING

Test Number	What is being tested?	Input	Output/ expected outcome
24	If the scoreboard is rendered to the end game screen when player dies	Player dies and goes to end game screen	The scoreboard is presented without any errors and with all the correct information
25	If the player scores less than the minimum on scoreboard they won't be seen	100 pts or something	Scoreboard is presented but player isn't on there
26	If the player scores more than the minimum points to be on score board but less than max	'10,000 pts' when lowest is 8000 and highest is 10,600	Player is presented in correct order of points on the scoreboard
27	Player reaches new highest score	'17,000 pts' when original highest was '16,000 pts'	The new player is then appended to the top of the scoreboard and the original best player is moved down the list by one row, original minimum scoreboard player is removed.
28	White box tests the scoring algorithm	If NewScore > playerScoreTop then NewScore = playerScoreTop	If the current player has more points than the current leader on the scoreboard, it is overwritten.
29	Players variables	Player scores points	I will use breakpoints and show that after every successful kill the players score variable increases by 100 each time

SOUND EFFECTS AND TIMINGS

Test Number	What is being tested?	Input	Output/ expected outcome
30	Music #1 loads when menu is loaded	Open game	When the user opens the game file and is greeted with the menu, the music track #1 will kick in
31	Music track #1 continuation	User clicks 'play game' and is greeted on username input screen	The same music track will continue to play without interruptions or it stopping
32	Music track #1 fading out	Clicking the 'play' button on the username input screen	When the actual game starts, I expect the game to fade out the music track #1
33	Music track #2 fades in	When actual game is started and music track #1 has completely faded out	Load up the actual game and see whether music #1 fades out and music #2 fades in.
34	Music track #2 fades out	When player dies	Load game up and die to see whether the music track #2 fades out
35	Load/ fade in music track #3 after music track #2 fades out	When player dies Music track #1 – happy music Music Track #2 – Hardcore Carpenter Brut music Music Track #3 – Calm peaceful music (Claire de lune - Debussy)	The 3 rd music track fades in after the music track 2 fades out when end game screen is presented.
36	Correct soundtrack plays at the correct time	Load game, enter game, end game	Each soundtrack should be changed when the game state is also changed,

			minimizing delays between them.
37	Pausing the game music	'P' is pressed	The pausing of the game will load in the menu music (music track #1)
38	Sound effects of enemies dying	When an enemy is killed	Audio track of a death noise will play over the music track #2 when a zombie is killed
39	Sound effect of death duration	Enemy dies	Duration should last around 1 second
40	Sound effect of player dying (different audio than a zombie dying)	Player gets killed by a zombie/ enemy	The device should output a specific audio track for 1 second for the players death before being greeted on the end game screen.

ASSIGNMENT OF VARIABLES

Main Game

Method	Name used	Data type	Explanation	Justification
Game state	game_state	Boolean	Main state for the game, for when the user is playing the actual game	Helps split up a large project into smaller solutions as each state is self-contained, allowing easier readability
X coordinate	moveX	Integer	The players coordinate in the x-axis	The player must have an x coordinate in order to move left and right
Y coordinate	moveY	integer	The players coordinate in the y-axis	The player must have an y coordinate in order to move up and down

Variable	screen	N/A	The pop-up window pygame creates, this is the window that will be stored in the variable 'screen'	Making sure to use similar names to the actual object allows it to be coherent and easy to identify, the pygame window must be in a variable in order to blit (render) to screen
Variable object	'crate','enemy' etc.	Char	In regarding to any object that is in question. Refers to any instance of a method or class I might decide to use.	Objects are just other variables which will be rendered to the screen that the player can interact with.
Variable player	player	Char	this is the variable used for what the user will interact with and control	This is the variable that the player will be able to control the coordinates to, this is the actual player where its characteristics can change based on other actions
procedure	detectCollisions()	N/A	This procedure will perform calculation to see if an object is in contact with another object.	Collision detection is needed in order to represent basic physics to make the game look more realistic and work correctly
Variable	X	Integers	Another variable used for coordination	It is a variable which isn't the ending coordinates, the

				end coordinates are based on moveX and moveY, these x and y's are just for calculations which lead to the ending values
Variable	y	integers	Another variable used for coordination	Same justification as above
Variable	clock	N/A	This is the frame rate of the game from the built-in clock to pygame	This variable allows easy manipulation and querying of the frame rate without typing the command in over and over every time I need to
variable	Collisions	N/A	Holds the result of the collision detection	This allows the coder to check if there is a collision or not easily
variable	Player_health	Integer	Players current health will be stored in a variable ready for deductions	This is to keep everything clear in the code and easy to use in calculations
variable	Weapon_damage	Integer	Shows how much damage each weapon does	This will be specific for every weapon and a variable allows quick switching and increases to the damage when the weapon is upgraded
Variable	Player_weapon	N/A	The current weapon will be kept in this variable	Easy to follow what weapon the

			allowing easy identification	player currently has equipped
variable	Weapon_ammo	Integer	Keeps track of the weapon ammo	Calculations will now become easier as you can deduct one bullet for every time the gun is shot
variable	Enemy_health	Integer	How much health an enemy will have	This variable will keep track of enemy health that can easily be used in deductions
Variable	Enemy_damage	Integer	How much damage an enemy gives	This variable will be useful in calculations for player health deductions
variable	powerup	N/A	Powerups will be different variables but the generic one used is 'powerup' e.g. 'powerup_health_increase'	The powerup variable can be called at any time randomly, allowing it easier to call
Variable	timer	float	The timer variable will be the countdown for every level until the next one starts	This is what will be presented on the screen, storing it in a variable will make it easier to call further down the code.
variable	black	N/A	This is the variable of the abstract background until I sort out the image. Something to draw the sprites and objects onto in the test program.	Saves me writing out the RGB value of (0,0,0) every time I want to use the colour black.
variable	Self.i1	image	Loads the sprite image into the program and then renders it through the render function	This allows the user to control a sprite I designed instead of just a

				rectangle, makes it more realistic
function	Render()	N/A	This is what renders and blits the sprites and objects to the screen	The reason for the render function subroutine is to keep all rendering self-contained just in case I need to debug why an object is not rendering.
function	__init__()	N/A	Initializes variables for the players width, height and X, Y coordinates	This allows any sprite I make to have the characteristics of the variables inside this functions, saving me time writing out all them again for each sprite or object
variable	Width	Integer	This will initialize the width of the players sprite	Sets the sprites width and saves it to the variable for saving time and later calling
variable	Height	Integer	This will initialize the height of the players sprite, ready for rendering	Sets the sprites height and saves it to the variable for saving time and later calling
Variable	Audio_track_2	N/A	This is the backing track when playing the game	Needs to have a variable to be called into the program otherwise won't work.
Variable	Time survived	float	Keeps track of how long the player stays alive for	This is so it can be used for the leader boards

Menu and Pausing

Menu State	menu_state	Boolean	The menu state will run when the user is on the menu screen and will stop when they come off it	This is to allow a clear idea of how the game will be split up into dedicated sections regarding the states
Game_state	paused	Boolean	Game state is paused when on pause page	This is so all processes stop so the game doesn't continue in the background
Procedure 'resume'	Resume()	N/A	Resumes the game and game state is set back to True/Running	This allows the player to keep player from where they paused it
Procedure 'exit'	Exit()	N/A	Game state ends and menu state starts running, user is brought back to the menu page	If the user is bored or has other things to do and wants to quit the game, there should be a button for it which stops all processes of the game state. Needed for a successful exit of the playable game
variable	Audio_track_1	N/A	The first audio backing track the user hears when they load up their game	This is kept in a variable, so it is easy to switch between audio tracks when the game states change

Scoring

variable	points	Integer	This variable will keep track of the points the player has scored	This is so the player can be seen on the score board if they have a high enough amount of points
Variable	Highest_score, second_score, third_score etc.	integer	Top 5 players will be kept in separate variables under their title on the score board	This helps to find out where the player sits in terms of points and if they are seen on the leader boards or not
variable	Audio_track_3	N/A	This variable is loaded when the player dies and is brought to the score board screen/ ending screen	This track is more similar to the menu one and is loaded in order to stop the game state music and have a more peaceful track playing so the user can focus on where they came on the leader board
variable	hit	integer	If an enemy is hit, then the points will increase.	easy way to identify a hit in collision detection
procedure	Endgame()	N/A	Switches the game to end screen allowing the score board to be seen	To stop the game from continuing after the player has died

PROPOSAL SIGN OFF

MENU SUMMARY

<u>What is deemed successful</u>	<u>Any improvements</u>
Aesthetically pleasing	N/A
Page changes dependent on which button has been pressed	N/A
Easy to use and has a basic design	Could add in a 'tip' section with more info on the menu page although it is quite self-explanatory.
Colours are matching and pairing and have a specific theme	N/A
The boxes/ buttons don't look out of place	I could always decide to add a drop shadow on each button to have an illusion of depth.
All screens on the menu are easy to follow and are concise in information	N/A
Easy to navigate the menu page	N/A
Menu music loads into memory on time	N/A
All buttons work and bring user to different pages	

USERNAME SCREEN SUMMARY

<u>What is deemed successful</u>	<u>Any improvements</u>
Matching colours of the theme	N/A
Easy user input	N/A
Be abstract to show the user the reason for the page	N/A
Other features	Have a back button on the username screen
User understands the limitations of the input box	N/A

Input limitations - >1 but <15 and no special characters	Could add more special characters to blacklist.
--	---

MAIN GAME SUMMARY

<u>What is deemed successful</u>	<u>Any improvements</u>
Main game changes the theme and looks natural	Could add a gradient filter over it to make it look more retro style.
On screen items (health bar, inventory etc.) are easy to see and are clear for what they are used for	N/A
User understands where they are and can easily orientate themselves.	Maybe after the username page, fade screen to black and have text saying 'California 2033' then fades into the game – better flow
User gets familiar with controls	N/A
Crosshairs are clear	Make bigger otherwise.
User finds the game interesting and enjoys playing it	Add in more features if this becomes an issue.
User understands the aim of the game	N/A
User find it difficult and tension builds	If it doesn't, then make the waves harder and the enemies stronger.
The layout of the game and the theme	N/A
Music changes on time after menu music	N/A

PAUSE MENU SUMMARY

<u>What is deemed successful</u>	<u>Any improvements</u>
Layout of pause menu	N/A
Similar theme to other pages (menu, username screen etc.)	N/A
Easy controls to bring up the pause menu	N/A
Easy to navigate the pause menu	N/A

END-GAME SCREEN SUMMARY

<u>What is deemed successful</u>	<u>Any improvements</u>
Layout of end game menu	N/A
Similar theme to menu screen	N/A
Does it show without delays?	N/A
Easy to navigate the end game screen	N/A
Ease of use	N/A

C. DEVELOPING THE CODED SOLUTION ("THE DEVELOPMENT STORY")

My coded solution will be the development of my game, this code will perform all functionality on my game and will be self-contained and standalone compared to other games where an internet connection must be applied.

Initially I was overwhelmed of how I will be able to construct a game from a simple interface such as pygame in Python, I noted all ideas I wanted for my game on a piece of paper in which I then converted to code when I was happy with every idea on the paper. Initially, I started with the game as this is the main feature, the main game, in my opinion, was the most important to focus on first, so I decided to start with some simple tutorials on how to setup pygame in my chosen IDE of Thonny.

The testing for my solution is done through the test tables in each section as well as video evidence in the 'evidence' folder, this allows easy identifying of whether or not the tests are successful and if they aren't then the measurements taken to improve the actual outcome.

TASK #1 – SPRITE MAKING AND INITIALIZING

Originally I had the idea to create the sprites, I used pixel art studio to create my sprites. After this I went through multiple tutorials on YouTube on how to init pygame, I got my bearings with how it all worked and I managed to create a 'skeleton code' program with the basic pygame functions in it such as the title of the window, updating and key starting variables like the initialization of the screen.



My sprite design for the user to interact with and control, this will be referred to as the player in the future and may even change the sprite further down the line.

```

1 import pygame
2
3 # This is it made with Classes and it works well
4
5
6 pygame.init()
7 screen_width = 850
8 screen_height = 550
9
10 screen = pygame.display.set_mode((screen_width,screen_height))
11 pygame.display.set_caption("Collision Detection")
12
13 #Init variables for movement
14 moveX,moveY = 0,0
15 x,y = 0,0
16
17

```

Import pygame module

Comment for me when referring to my code

Init pygame module

Sets dimensions of pygame window

Init screen variable and set to dimensions above.

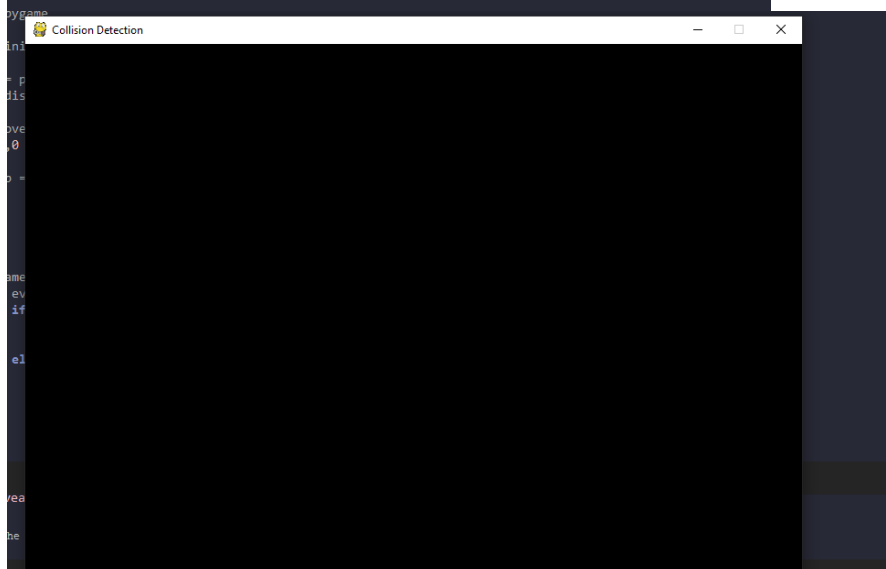
Title bar name

```

#Inits the frame rate and the cycle clock
FPS = 70
clock = pygame.time.Clock()
clock.tick(FPS)

# Updates the screen while True in game_state
pygame.display.flip()
pygame.quit()

```



Candidate Name: Cameron Noakes

Candidate Number: 9453

This code is only for initialising pygame and creating the window, hence why the black screen is empty and no objects or players can be seen on it, this process is to help me get familiar with pygame and understand the underlying nature of how pygame operates and what I need to declare.

I will annotate using the built in commenting feature of my IDE, this is shown above and will save me time drawing arrows and text boxes. For code reference for each part be sure to see the yellow hashtags as comments, I will also decide to do some annotation of the code for additional context.

PROTOTYPE MAKING

This section will identify all prototypes I have made for my game, including any demos or test code. This will help to further show my development process, instead of showing only the ending code I choose to also include my early programs to see where I have come from.

PROTOTYPE #1 - DRAWING SPRITES

This code prototype was made using Object Orientated Programming and this was one of my first prototypes that I coded as it was crucial to make sure I can render sprites.

```
drawing_sprites (OOP).py * x
1  import pygame
2  pygame.init()
3
4  screen = pygame.display.set_mode((850,650))
5  pygame.display.set_caption("Test game")
6
7  #players moove x and y
8  moveX,moveY = 0,0
9  x,y = 0,0
10
11  game_state_play = True
12
13  #player_sprite_LEFT = pygame.image.load("sprites/sprite1_LEFT.png")
14  #player_sprite_RIGHT = pygame.image.load("sprites/sprite1_RIGHT.png")
15  #player_sprite_UP = pygame.image.load("sprites/sprite1_UP.png")
16  #player_sprite_DOWN = pygame.image.load("sprites/sprite1_DOWN.png")
17
18
19  #Class for creating sprites
20  class Sprite:
21      def __init__(self,x,y):
22          self.x = x
23          self.y = y
24          self.width = 20
25          self.height = 20
26
27          self.i1 = pygame.image.load("sprites/sprite1_RIGHT.png")
28
```



I initialized pygame just like the rest of the programs I coded up, you can see comments of code, this was where I thought it would be a good idea to change the facing direction the sprite of the player depending on the direction of the player movement, this became harder to implement so I decided to leave it out. Below is a screenshot of my game code for instances of the class, you can find my sprites and objects. This was where the code wasn't great as the objects moved with the player as they were all in one class, so I separated the player and objects by putting them in different classes, now works well. Below is a developed version of the sprite making prototype:

```
white = (255,255,255)
black = (0,0,0)
player = Player(100,50,100,100)
#crate = draw_objects(800,100,50,50)
jeep = draw_objects(800,350,130,120)

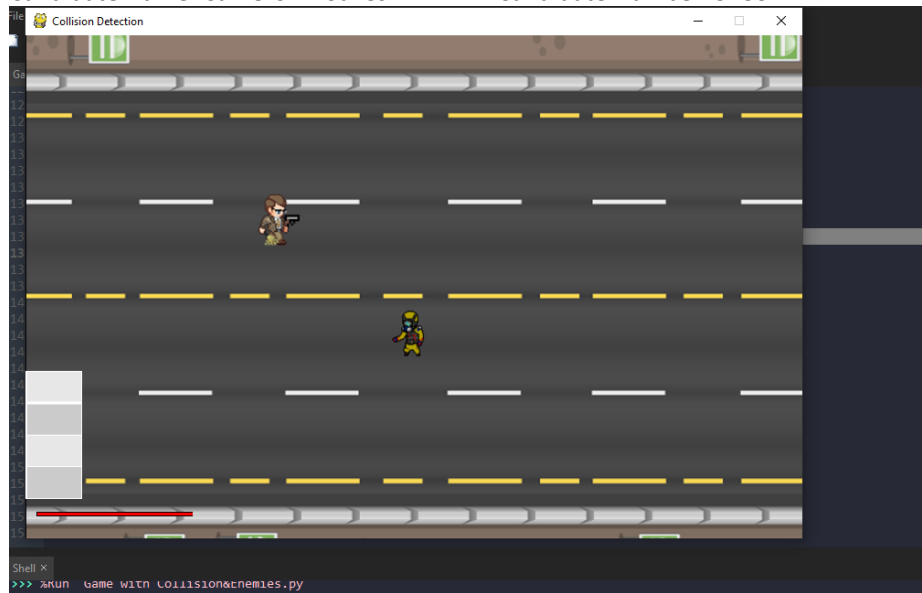
game_state = True # inits the game state

while game_state:

    # inits the events the user could choose
    for event in pygame.event.get():
        if (event.type==pygame.QUIT):
            game_state = False

    # if the user uses a keydown (presses a button) it
    # will go through this elif below:

    # Arrow keys or A,S,W,D first
    elif (event.type==pygame.KEYDOWN):
```



The sprite is made in the 'Player' class and then checks the input of the keyboard, this allows the player to move on the screen.

In the first prototype I made it, so the screen background is black, this checks if my movement code works and if the player renders to the screen correctly.

PROTOTYPE #2 – MENU

My prototype of my menu is just how it would say the button name when hovered over the button, it is not too complex, the prototype code is not too efficient and could be more readable, but I just left my prototype and focused on making the ending menu prototype readable and efficient like it is now.

```
# Mouse coordinates
event = pygame.event.poll()
if event.type == pygame.QUIT:
    exit()
elif event.type == pygame.MOUSEMOTION:
    x, y = event.pos
    print(x, y)
    # (x) top left, (x) top right, (y) top left, (y) bottom right
    if (x) > 186 and (x) < 373 and (y) > 158 and (y) < 230:
        print("play game")

    elif (x) > 480 and (x) < 664 and (y) > 160 and (y) < 233:
        print("Help")

    elif (x) > 186 and (x) < 371 and (y) > 284 and (y) < 356:
        print("Instructions")

    elif (x) > 479 and (x) < 664 and (y) > 284 and (y) < 356:
        bg = pygame.image.load("Designs/overview_page.png").convert()
        bg = pygame.transform.scale(bg, (screen_width, screen_height))
        screen.blit(bg, (0,0))

    elif (x) > 332 and (x) < 518 and (y) > 408 and (y) < 480:
        print("Exit")

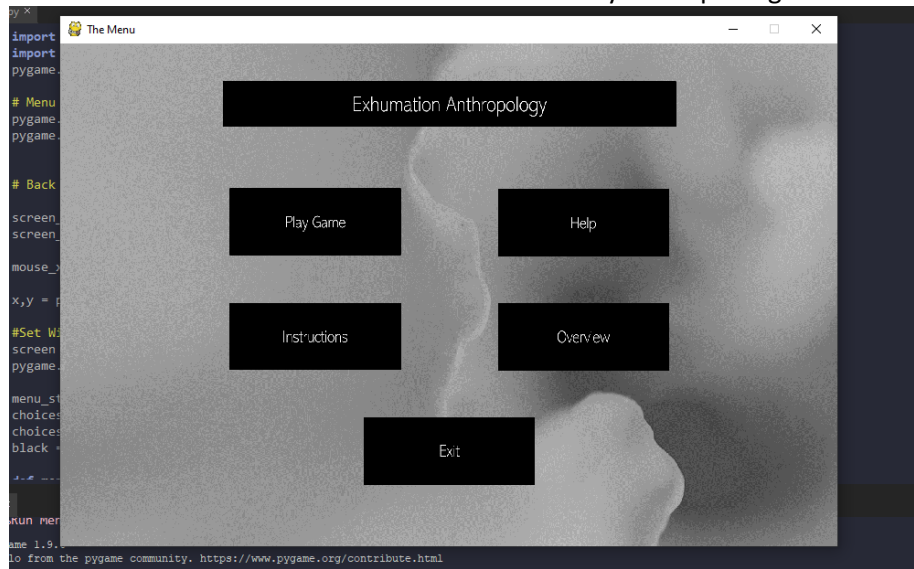
pygame.display.update()
```

This small menu prototype has the same code as the ending menu. In this prototype there is no subroutines for the individual buttons, this is partly why it is less efficient but didn't feel necessary to include as I don't need to reuse any components from the menu in my main game for which the user

Candidate Name: Cameron Noakes

Candidate Number: 9453

would be playing. An end-user view can be seen below which is the official end of the menu process and this is the menu that the user would see when they load up the game.



TASK #2 – THE PERIPHERAL CONTROLS

Next, I focused on how I would render my sprite I created to the screen and how the player will be controlled by the user. I looked on a few websites on how to test for user processes for the peripherals like the keyboard. The reason for implementing the peripheral controls is to allow the user to control the player on the screen in a full dynamic approach. After about 2 days I managed to get the hang of it and my code for it was concise and readable, it is annotated below.

```
game_state = True # inits the game state

while game_state:

    # inits the events the user could choose
    for event in pygame.event.get():
        if (event.type==pygame.QUIT):
            game_state = False

    # if the user uses a keydown (presses a button) it
    # will go through this elif below:

    # Arrow keys or A,S,W,D first
    elif (event.type==pygame.KEYDOWN):
        if (event.key==pygame.K_LEFT or event.key== ord('a')):
            moveX = -22

        elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
            moveX = 22
            player.draw(collisions)
        elif (event.key==pygame.K_UP or event.key== ord('w')):
            moveY = -22

        elif (event.key==pygame.K_DOWN or event.key== ord('s')):
            moveY = 22
```

Part A

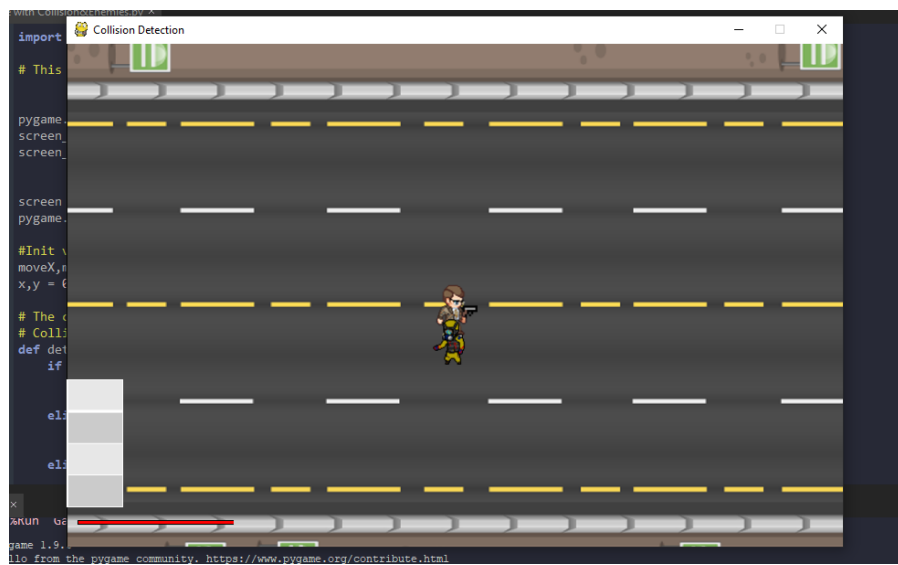
```
# Checks to see if the user has let go of the key
elif (event.type==pygame.KEYUP):
    if (event.key==pygame.K_LEFT or event.key== ord('a')):
        moveX = 0
    elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
        moveX = 0
    elif (event.key==pygame.K_UP or event.key== ord('w')):
        moveY = 0

    elif (event.key==pygame.K_DOWN or event.key== ord('s')):
        moveY = 0
```

Part B

This section of code fits into the main game I have created, this allows user input at the peripheral controls to allow the movement of the player in the game, this section can go anywhere in the main game code file as long as it is under the while statement for the game state of the main game.

Game in action and player moving:



Dry data/ Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
W,A,S,D	To test the peripheral controls of the keyboard	To allow the player to move on the screen	To move the player the direction the user wants	Works as expected	N/A
Arrow Keys	Determines if the player will be responsible to the	An alternate method of moving the player on the screen	To move the player the direction the user wants	Works as expected	N/A

	peripheral keys of the arrow keys				
Lifting keys up	Stops player movement after keys have stopped being pressed	This is to allow full manipulation of the player on the screen, allowing the player to stop	Player stops immediately when the button has stopped being pressed	Player didn't stop	Altered the code to update movement to 0 – now works.
Overall experience	Does the player get updated every time it renders it to the screen per frame?	Overall test to check the other peripherals are working correctly.	Player moves without lag	Player didn't move	Moved update code to the bottom and after this little tweak it works.

TASK #3 – COLLISION DETECTION

Implementing the collision detection was interesting as it was the first time I have made collisions for a game, it allowed me to get an insight of how these would work and once I understood there are multiple ways to implement it, I became familiar with a chosen method. The method chosen for collision detection is to take each x and y values of an object and pass it into a collision detection function, I chose this method instead of (for example) the collision box method as I found this one was more suited to how my game will function and I understood it more.

I took the coordinate values of x and y of two objects then passed them into the subroutine of 'collisions'. It then took me a while to then understand how to compare the coordinates of both objects and to see if one object was inside the other. I managed to complete this subroutine after around a week of researching it and practicing trial and error.

The justification behind why I chose to implement collision detection was due to the player being able to successfully interact with objects in the game and on the screen for more of a dynamic gameplay and to allow deductions of health and bullets through collisions.

Collision detection would fit into the main game library code I have created as the player must be able to interact with objects on the screen. This section of code only needs to be added into the main game code library as it isn't needed on the menu or end game screen as the player is not rendered at these points.

```
# Collision detection - takes two objects, (x,y), width and height
def detectCollisions(x1,y1,w1,h1,x2,y2,w2,h2):
    if (x2+w2>=x1>=x2 and y2+h2>=y1>=y2):
        return True

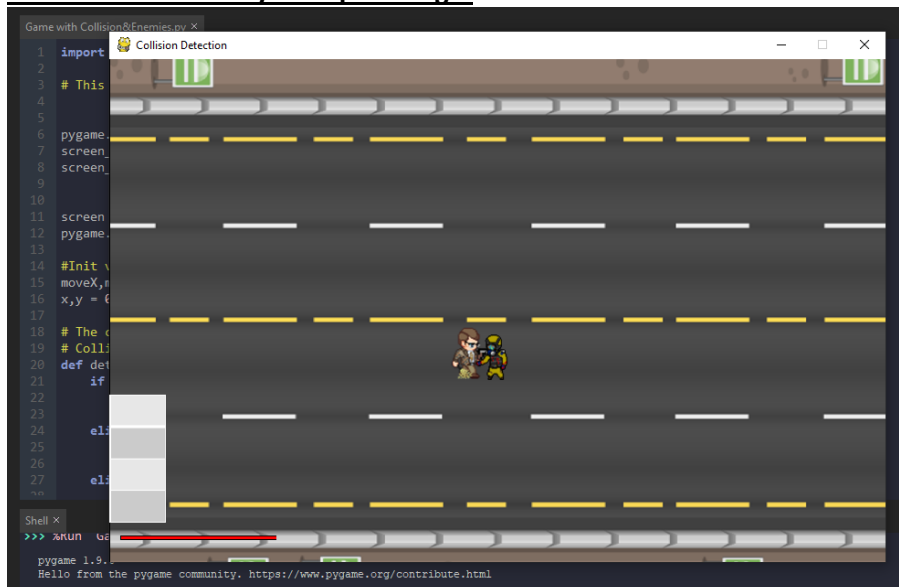
    elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1>=y2):
        return True

    elif (x2+w2>=x1>=x2 and y2+h2>=y1+h1>=y2):
        return True

    elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1+h1>=y2):
        return True

    else:
        return False
```

Game in action – Player stops at edge:



Values passed with '1' in the variable (e.g x1,w1 etc) refer to the first object in question and variables with '2' refer to the second objects x and y coordinate values. This code will determine whether or not if the first object is within the limits of the second object, causing a collision to be true and returning it. This function is called by the code below.

```
#test for collisions
collisions = detectCollisions(player.x,player.y,player.width,player.height,crate.x,crate.y,crate.width,crate.height)

#parse values into the 'Player' and 'draw_objects' Class
Player.draw(player,collisions)
draw_objects.draw(crate,collisions)
crate.draw(False)
player.draw(collisions)
```

The code underneath the second comment is parsing variables into two classes I have made, these classes have a subroutine inside that help render the sprites and objects to the screen and stop movement if a collision is True.

It is dependent on the value returned from the 'collisions' function, if the function returns True then the subroutine in the class will prevent the player moving any more due to an object being in the way.

Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Player hits enemy from x axis left to right	Checks if a collision was made on the left side of enemy horizontally	This checks to see if the player has hit the enemy horizontally on left side	Player stops at edge of object	Player passes straight through object	Altered code for collision detection – Works now.
Player hits enemy from x axis right to left	Checks to see if a collision happened on back of enemy	This one doesn't really have to be here as enemies would attack from the front but included it anyways to cover the all-round collisions	Player stops at edge of object	Works as expected	N/A
Player hits enemy from y axis above to below	To stop player t edge of object	Easy fix, change the dimensions of the enemy y axis of sprite	Player stops at edge of object	Works as expected	N/A
Player hits enemy from y axis below to above	Player sprite stops at bottom of enemy sprite, as a collision has been detected	This is to stop the player passing through the enemy, not triggering a collision.	Player stops at edge of object	Works as expected	N/A

TASK #3B – OBJECTS CLASS – COLLISION DETECTION

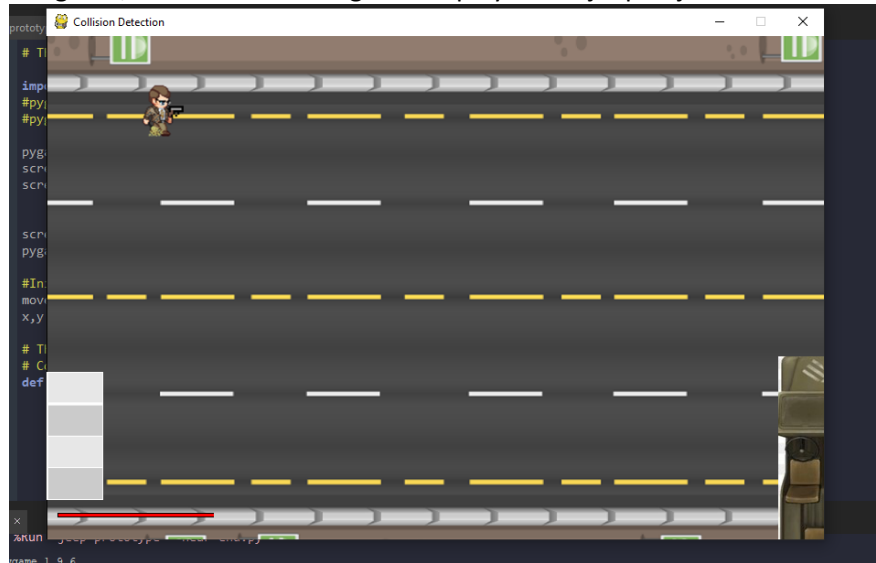
This is the second class in my program that I have coded, it is very similar to the 'Player' class however, these attributes refer to instances that are only objects, not the player itself as I did combine these classes before and the objects were linked to that of the player, moving the player and objects as one. This meant that the objects (instances of the class 'draw_objects') would move with the players input, they should be static. I decided to make them two separate, self-contained classes as this would stop this issue, now the user can move the player as they wish without worrying about having the background objects linking and following. The code for this second class is below and is not annotated because the comments in the code state the meaning behind it all.

The reasoning behind the object class is to allow a consistent design of my code to follow on from the player OOP class – to continue to use Object orientated. Objects needs to be distinguished from the player (which is also an object) this is done to allow different objects to behave in different ways, the player will be moveable, the crates and other objects will not. This object class is needed to render different instances of the objects to the screen.

```
Game_Classes.py * x
59 # Class 2
60 class draw_objects:
61     #initialize all attributes of each object
62     def __init__(self,x,y,width,height):
63         self.x = x
64         self.y = y
65         self.width = 25
66         self.height = 25
67
68         #load image of object into memory
69         self.i2 = pygame.image.load("images/crate1.png")
70
71
72     #testing for a collision
73     def draw(self,collision):
74         if (collision == True):
75             player.x -= moveX
76             player.y -= moveY
77
78         #render object to the screen
79         screen.blit(self.i2,(self.x,self.y))
80
81
82 white = (255,255,255)
83 black = (0,0,0)
84 player = Player(100,50,100,100)
85 crate = draw_objects(200,350,50,50)
```

The "crate" variable at the bottom shows how an object will have the attributes of the class. Just above the "player" variable you have some RGB values sealed as a tuple, these are for easy referencing of colours as I can reference colours by their name instead of their RGB value. I later changed the 'crate' object to a jeep, this is to make it more post-apocalyptic and give a better feel for

the game, see below for the game display of the jeep object:



Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Object renders to screen	Uses a crate as an example to test if objects render to screen and collision detection	This is because I will need to render the enemies into the game	Object should be rendered to the screen at coordinates x,y	Works as expected	N/A
Collision detection works	Collision detection is needed to stop player at objects vertex'	This stops players walking through objects or enemies	Player stops at start of edge of object (jeep)	Passes through object and off screen	Improved the collision detection

TASK #4 – PLAYER CLASS

The 'Player' class is a self-contained class that contains the attributes of the player, this allows easier coding for the player as there is one player and having it as a class makes the collision detection work better. Inside the class there is an 'init' subroutine in order to setup the attributes of the instance of the player, then another to render the player to the screen, this second subroutine is (like stated above) used to detect collisions based on the value returned from the collision detection subroutine.

The player class justification is to allow concise code that will help make my program more efficient and readable, using classes improves the readability of code and minimises the duplication of code as a class can have many instances with the same attributes. My 'Player' class is detailed below.

The player class is an Object orientated approach to showing the player on the screen and updating the players movements. Justification of this would be to allow an efficient OOP method of development of the player to allow the user to control a character on the screen.

```
# Class 1
class Player:
    def __init__(self,x,y,width,height):
        self.x = x
        self.y = y
        self.width = 20
        self.height = 25

        # Player is now the medium sprite size to match the background size
        self.i1 = pygame.image.load("sprites/sprite2.png")

    def draw(self,collision):
        if (collision == True):
            player.x -= moveX
            player.y -= moveY

        screen.blit(self.i1,(self.x,self.y))
```

Variables passed into `__init__` subroutine

Attributes of each instance

Load the sprite image as the player

Detects collisions based on collision outcome

Stops player movement

Renders it all to the screen

Game in action for the rendering of the player using OOP:



Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Player loads to screen	The player sprite should load to the screen without any delays	The user needs to see where their player is in order to protect it from enemies	Player sprite rendered to the screen	Didn't render the player	Had to reference what I wanted to draw sprite on (Screen) – works now.
Player has fully functionality	User can have dynamic change to the player	Movement, firing etc. needed to play the game	User can control the player sprite	Works as expected	N/A

TASK #6 – SETTING THE BACKGROUND

```
# "Designs/setting_smaller1.png" was the original background, changed to setting2
bg = pygame.image.load("Designs/setting2(features).png").convert()
bg = pygame.transform.scale(bg, (screen_width, screen_height))
screen.blit(bg, (0,0))
```

At the moment the user is displayed on a black background, I have used a built in feature of pygame to allow an image to be set as a background, took only a few lines of code. I felt whilst I was trying to implement classes that I was not making any progress at the start so I did a visual change, added a sprite instead of a square, added a background and made the window more pleasing to the eye, this then helped me focus at the task at hand which was the classes, not long after this change I had the classes complete.

The explanation behind setting the background is to allow a more of an interesting game to the end user as a black or white screen is abstract and doesn't hold the users attention much, I wish to implement a background that will improve the overall design features of the game to allow a more enjoyable experience. Implementing this background has no other effects on the logic or dynamic of the game.

The background code section will go at the bottom of my main game code as it needs to be rendered at the end when all other processes have been executed such as rendering the player to the screen.

Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
If the Background (BG) loads	The background must load for the player to be loaded onto it	Without a background it would just be a black screen with a player moving on it	Background loads and the player and other objects load onto it	Background loads but dimensions mess up the pygame screen	Used a scale where I declared the dimensions of the background image to that

					of the screen – works fine now.
BG loads without delays	Background is loaded without any latency due to my high FPS	This saves the player waiting around for the game to load as loading screens aren't fun			
BG is brought into memory after username page	After username is entered and play button is pressed the background then loads due to importing a python library that I made	This is to stop the menu processes and start the main game process, stopping lag from one massive game file.			

Refer to any video evidence as they all show the background being loaded.

TASK #7 – MENU

This section will cover all of my menu and how I managed to create one. At first, I struggled to get my head around how I would implement and code a menu and how I would link it to my main game. After thinking of it abstractly I had a clearer understanding of how this could be possible.

I decided to copy the code of my main game into a new file and delete the main game code and name the program 'Menu', now I had the base code for the initializing of pygame. I then created a new window and loaded my menu image into memory to render it to the screen.

After this I then was unsure on how to make it interactive as, after all, it was just an image rendered to the screen. After researching a bit, I found out that pygame has no built in features of default libraries to allow button clicking. This made me even more unsure on the process of the menu, so I just decided to code up a button coordinator method myself in order to complete this process. I did it this way in order for my game to stand out and make the menu easier to use with a mouse instead of using numbers corresponding to the buttons as this takes the feeling away from being a better overall game.

I took the mouse coordinates when it was on the menu screen and outputted every motion to the terminal behind it, this way I could see every x,y coordinate of the mouse whenever it moves. This didn't take long, and then I managed to make it, so the coordinates only outputted to the terminal when a left

click was present. I opened a notepad and took coordinates of every vertex of each button, I left clicked on every buttons vertex and then the coordinates were shown in the terminal, I copied them over to a notepad and with a key for which vertex it was (TR = Top Right). This data is given below:

(x , y)

186,158 TL of 'play game'

373,161 TR of 'play game'

185,230 BL of 'play game'

373,230 BR of 'play game'

480,160 TL of 'Help'

664,159 TR of 'Help'

480,231 BL of 'Help'

664,233 BR of 'Help'

186,284 TL of 'Instructions'

371,285 TR of 'Instructions'

185,357 BL of 'Instructions'

371,356 BR of 'Instructions'

479,284 TL of 'Overview'

664,285 TR of 'Overview'

479,357 BL of 'Overview'

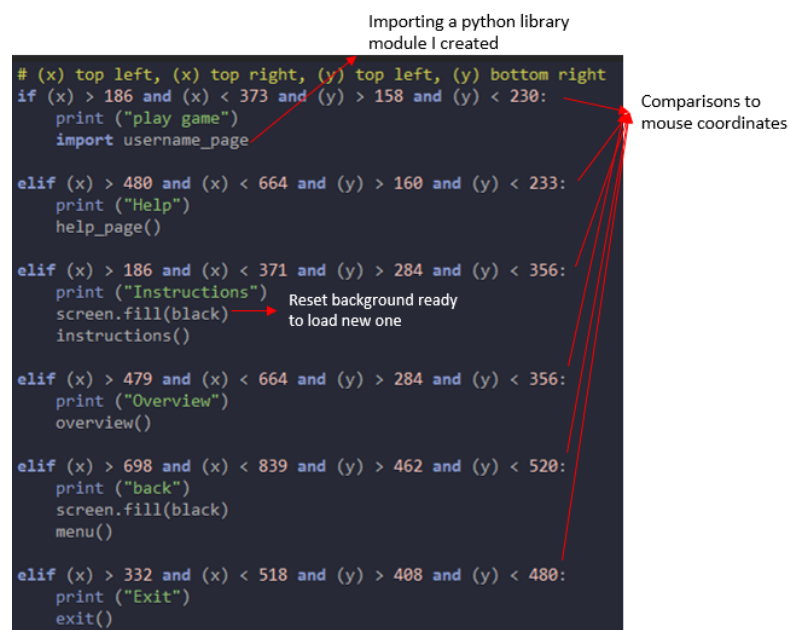
664,356 BR of 'Overview'

332,408 TL of 'Exit'

518,408 TR of 'Exit'

332,483 BL of 'Exit'

518,480 BR of 'Exit'



```
# (x) top left, (x) top right, (y) top left, (y) bottom right
if (x) > 186 and (x) < 373 and (y) > 158 and (y) < 230:
    print ("play game")
    import username_page

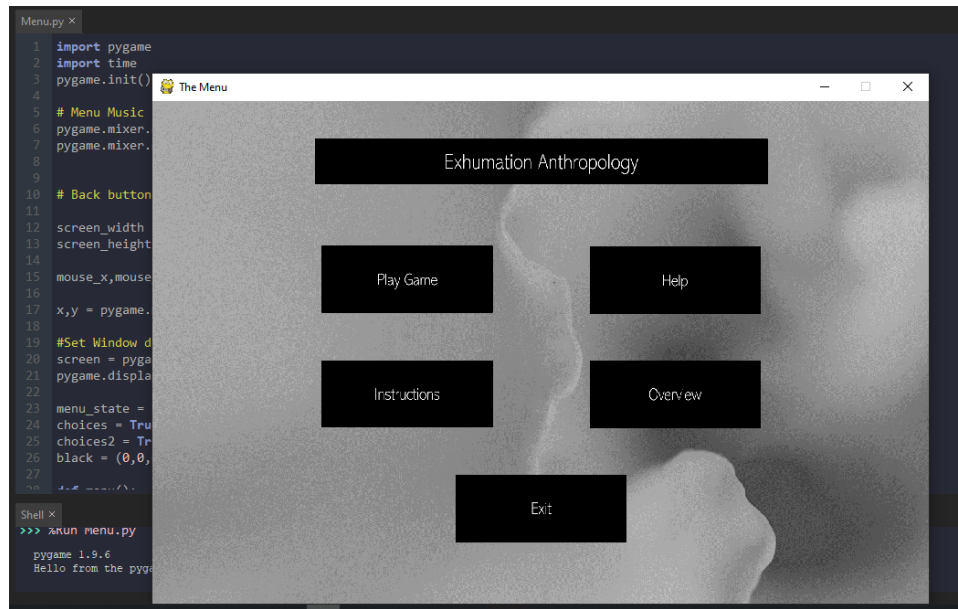
elif (x) > 480 and (x) < 664 and (y) > 160 and (y) < 233:
    print ("Help")
    help_page()

elif (x) > 186 and (x) < 371 and (y) > 284 and (y) < 356:
    print ("Instructions")
    screen.fill(black)
    instructions()

elif (x) > 479 and (x) < 664 and (y) > 284 and (y) < 356:
    print ("Overview")
    overview()

elif (x) > 698 and (x) < 839 and (y) > 462 and (y) < 520:
    print ("back")
    screen.fill(black)
    menu()

elif (x) > 332 and (x) < 518 and (y) > 408 and (y) < 480:
    print ("Exit")
    exit()
```



I then compared the current mouse coordinates to that of each coordinate in the data referenced above:

(x) top left, (x) top right, (y) top left, (y) bottom right

These were the holders of the coordinates I needed for the buttons, in turn, I took each button and its needed coordinates and compared it to the current position of the mouse on the screen. I tested this through hovering over the button and if the button was within specific limits it would print out in the terminal what it said on the button, this showed that the buttons correctly worked and only now needed to be linked to the other pages and the main game.

I then coded it so instead of hovering over the button and displaying the button name, it now only displays the button name in the background terminal when the button is left clicked.

Most candidates would just label the buttons with numbers and have the user press the button specified with the button (e.g 1. play). I chose not to do this as I found it more of a challenge to take the mouse coordinates instead, it also allowed my game to be more professional as the user will see that I haven't abstracted any processes for them and is easier for the user to click the button they want instead of typing its corresponding number.

Now that all the buttons now worked, I needed to link each new page that I designed.

This was simple enough, I just had to set the background to the new image. As the only interactivity on the 'Overview', 'instructions' and the 'help' page is a back button, I just implemented it on every page including the menu, the back button is hidden on the main menu page so if the user clicks where the back button is hidden, it will do nothing as you are already on the main menu, making it look like there is not back button. The back buttons are only on the pages provided above.

I had an overall issue with how the new pages loaded into the pygame window as one was already loaded, this would make it so both load at the same time, to stop this from happening I reset the

background to a black one before loading in the new page into memory and then also put the new background code into a while loop to prevent it from interfering with the original menu background, after this the pages loaded how they were supposed to and all functionality of the menu and additional pages worked. The additional while loop to help load the new pages is given below:

```
while menu_state:
    menu()
    while choices:
        # Mouse coordinates
        event = pygame.event.poll()

        if event.type == pygame.QUIT:
            exit()
```

Each additional page on the menu has a specific subroutine, the callings can be seen on the above image, this allows concise code and makes it much cleaner and readable, see below for the individual subroutines that get called:

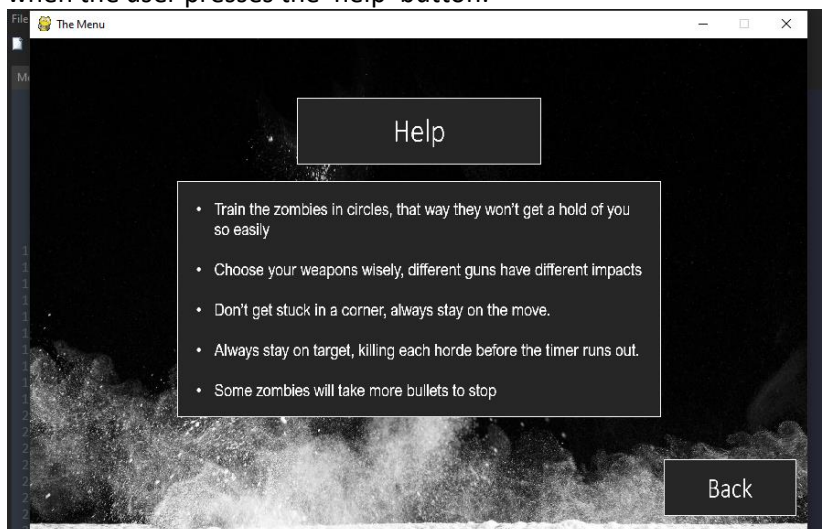
```
def menu():
    bg = pygame.image.load("Designs/Menu_design.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def instructions():
    bg = pygame.image.load("Designs/peripheral1.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def help_page():
    bg = pygame.image.load("Designs/help_page.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def overview():
    bg = pygame.image.load("Designs/overview.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))
```

An example of one of the pages on the menu can be seen below, this is the page that will be displayed when the user presses the 'help' button:



The justification behind why I decided to code up a menu screen is to not make the user jump straight into the game without any prior knowledge, I have made the menu page to allow the user to cope with the controls and the overall premise of the game so they have a higher chance of understanding what the game is about and how to play it. The menu is an abstract way to view information about the game that could help the user during gameplay, my menu doesn't need much prior knowledge to know the overall workings, you click the buttons and different pages are displayed depending on the button clicked, this is a good way to display exposition as well as overviews and controls within the game.

Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Clicking overview page	Shows the 'overview' button is linked and working	This shows that my menu is fully functioning	The code links correctly to the overview code to load the page	Works as expected	N/A
Loading the overview page	When button is clicked, the overview page loads	This shows the overview page for the user to understand the aim of the game	The overview page loads without latency or omissions	Got an error and didn't show page	Had to reset the background, corrected by setting background to a black screen then loading in the new screen for 'overview' page
Clicking 'exit' button and exiting the game	Easy leave of the game/menu	Make the user have more dynamic change to the game and a fast close	Exits game immediately	Works as expected	N/A
Clicking 'Help' button and	Loads the page that could help the user	New users will need to understand	Help page loads with button	Error again, same as	Reset the background to black then

loading the page	succeed in the game	how the game works	clicked on menu	'overview' page one	load in the new one
Clicking and loading the 'instructions' page	Instructions page is loaded and put onto the screen	This is to allow the player to get a grasp of the controls of the game	Loads the specific page needed	Same error as 'overview' and 'help' page	Reset the background to black before loading in the next page.
Clicking anywhere else	Nothing would happen and the screen wouldn't change as the user has not clicked on a button	The user only has the menu options when loading the game at the start	No action	Works as expected	N/A
Clicking 'play game' button	Loads username library I created for the next step	This must be a button in order to allow the user to play the game	Brings user to username page	Works as expected	N/A

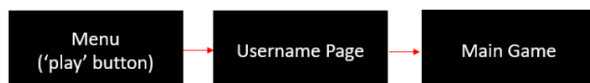
TASK #8 – USERNAME INPUT PAGE

This page is what is shown after the user clicks the play button on the menu page, this is where the end user will add in their player name/ gamertag, allowing a small dynamic change to the game, allowing the user to feel that they have more control of the outcome of the game. This was relatively simple, I added this in after I made the menu link to my main game, I added it in between (see below for a visual)

From



To



There is a text bar that allows a user input of a string only, based on the input, if valid, it will take the user into the game, if not the screen will stay the same. There is no back button on this page as this is where I feel the game will actually start, showing no turning back and to make the user prepare for the game on the next screen.

For this page I made sure to have the same dimensions as the previous page (the menu), this creates the illusion of only the screen changing.

A while loop is used to then call a subroutine that will load the new username input screen into memory and display it on the screen to the user. I then used the same button method comparisons like on the menu for the 'play' button on the username page, I got the coordinates of the button from the same method before for the buttons on the menu page, applied it to the 'play' button and it was all complete.

The 'play' button then imports my main game module that then loads the main game and is a smooth transition between the two.

The reason sitting behind why I decided to choose to add in an input page for the username is to allow more of a dynamic approach when it comes to the player, I wish to give the end user an experience that they find 'dynamic', changing the way the game functions, even if that is just with a username the user can enter as it can give them a sense of achievement once they reach the leader board.

```
.py x  username_page.py * x  Game_Classes.py x
import pygame
import time
#import sys
pygame.init()

screen_width = 850
screen_height = 550

mouse_x,mouse_y = pygame.mouse.get_pos()

x,y = pygame.mouse.get_rel()

#Set Window dimensions and title
screen = pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("Username Input")

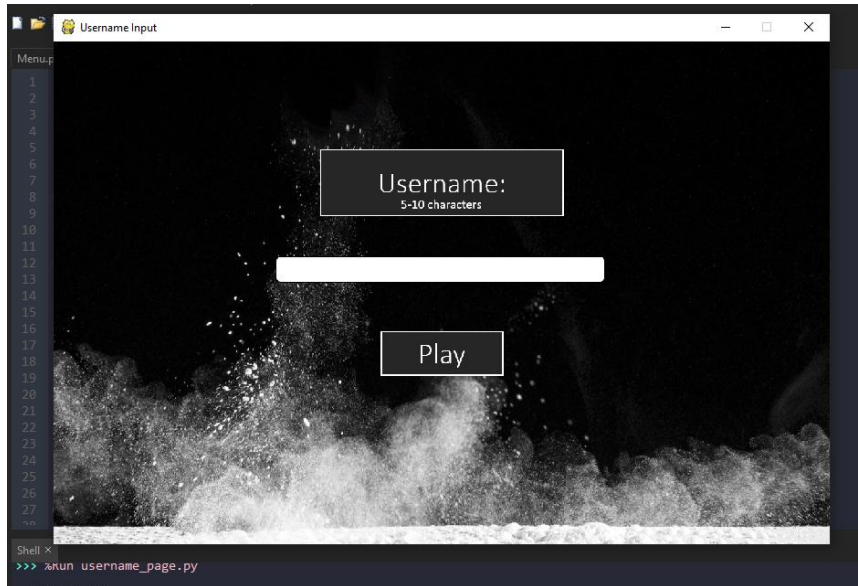
#Variables for while loops
user = True
username_screen = True

def usr_scrn():
    bg = pygame.image.load("Designs/username_screen.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))
```

```

6 # Enters a loop for module of username page
7 while user:
8
9     usr_scrn()
10
11     while username_screen:
12
13         # Mouse coordinates
14         event = pygame.event.poll()
15
16
17         #exit game
18         if event.type == pygame.QUIT:
19             exit()
20
21         # debugging test
22         elif event.type == pygame.MOUSEBUTTONDOWN:
23             x,y = event.pos
24
25
26
27         #Username input field clicked - enter username in terminal.
28
29         # (x) top left, (x) top right, (y) top left, (y) bottom right
30         if (x) > 245 and (x) < 600 and (y) > 238 and (y) < 262:
31
32             valid = False
33
34
35             while valid == False:
36
37                 username = input("Username: ")
38
39                 #array stores invalid chars
40                 array = ["@", "#", "|", "<", ">", "?", "[", "]"]
41
42                 #length check
43                 if len(username) > 1 and len(username) < 15:
44
45                     #takes each letter in turn - checks if invalid
46                     for i in array:
47                         if i in username:
48                             print ("invalid username - must be a string.")
49                             valid = False
50                             break
51
52                         if i not in username:
53                             print ("Valid Username.")
54                             valid = True
55                             break
56
57                     else:
58                         print ("Username must be between 1 and 15 chars.")
59
60                 # if 'Play' button is pressed
61                 elif (x) > 359 and (x) < 489 and (y) > 318 and (y) < 365:
62                     print ("play")
63                     pygame.mixer.music.fadeout(1500)
64
65                     import Game_Classes
66
67
68             pygame.display.update()
69
70             clock = pygame.time.Clock()
71             clock.tick(70)
72
73             pygame.display.flip()
74             pygame.quit()

```



Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Username can be inputted	Allows a dynamic gameplay for user	A name to show on leaderboards	Allows user to start to play the main game	Didn't allow user input on pygame window	Allowed it to be inputted in the terminal instead
Cannot use blacklisted items	This is to secure my game and not make it hackable/changeable	Could cause other issues in my game e.g. unrecognized symbols	issue caught and custom error is given	Error given to tell user what the issue is	N/A
Sensible/accepted username	A username that doesn't violate any of the rules I have made	Username that is allowed and what will be shown on scoreboard	Takes them into the game when 'play' button clicked	Works as expected Username accepted and stored in variable	N/A
'Play' button clicked	This is the button you press after a username is inputted	To allow the player to get fully ready before beginning the game,	Takes user into game and main game	Works as expected	N/A

		need a button to start the game	loop starts		
Length check 1-15	Length check to make it a suitable username	To not cause errors on the scoreboard by strolling off of the screen	If less than 1, custom error given (same if over 15) otherwise suitable	Works as expected	N/A

BACKGROUND MUSIC

I, at a later date, also added in background music for my menu, the reason for this is to greet the user when they first open up the game, the music is 'Heaven Falls' or another calm soundtrack, and is very uplifting and a light, calm type of music, I have intentionally used this type of music to make it hard for the user to predict what they will be up against when the actual game loads and they control a character, it adds a sense of disillusionment to the game and vagueness as well as suspicion also given from the title 'Exhumation Anthropology'.

I have then extended the soundtrack for the username page as well and at you click 'Play' button on the username page the music will stop and the actual gameplay music will load, which will be Carpenter Brut, which is very menacing and 'hardcore'.

Justification on why I decided to finally add in background music was to improve the overall user experience when the user plays my game, it would allow the user to enjoy the game more as they wouldn't be playing the game in silence, the music in the background could help them when they are in 'flow state', when the user is fully emerged in the game.

```

Menu.py ×  username_page.py ×
1  import pygame
2  import time
3  pygame.init()
4
5  # Menu Music
6  pygame.mixer.music.load('audio\HeavenFalls.mp3')
7  pygame.mixer.music.play(0)

```

In the username page I also faded out the music of the menu screen and on the actual game library changed the music to the music of 'Carpenter Brut'

Username fadeout:

```
# if 'Play' button is pressed
elif (x) > 359 and (x) < 489 and (y) > 318 and (y) < 365:
    print ("play")
    pygame.mixer.music.fadeout(1500)

import Game_Classes
```

Main Game music load:

```
import pygame
pygame.mixer.music.load('audio\CarpenterBrut.mp3')
pygame.mixer.music.play(0)
```

Type of Test	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
Plays menu music when game is opened	Tests to see if music loads when game startup	To make sure music is meant to start where it is supposed to, and no errors arise	Menu music plays when game is loaded up	Works as expected	N/A
Latency on music	Sometimes the music can lag into the game which would explain missed starting audio	I didn't want this so tested it by opening the game a few times and didn't set a time delay to start it.	No latency on the music	Works as expected	N/A
Fades out	To make sure it doesn't abruptly end and startles the user	This is to add a better feel for the game as it is more of a smooth transition	Menu music fades out when main game starts	Works as expected – due to fading timer	N/A
When on main game	Different tracks for different	To change it up as you	To stop menu	Error occurred	Had to edit code to allow

main music soundtrack loads	sections of the game	enter the main game	music and load the other	as of overlay of music	one track to stop and start the other
--	---------------------------------	--------------------------------	---	---------------------------------------	--

USERNAME VALIDATION:

Username validation was done on the sole purpose to secure my game to prevent any potential vulnerabilities and attacks against my game. I want my game to be fun and enjoyable and not to be taken apart and exploited so I decided to add in a blacklist of characters to make sure that no unusual characters could be potentially used to hack my game.

```
while valid == False:

    username = input("Username: ")

    #array stores invalid chars
    array = ["@", "#", "|", "<", ">", "?", "[", "]"]

    #length check
    if len(username) > 1 and len(username) <15:

        #takes each letter in turn - checks if invalid
        for i in array:
            if i in username:
                print ("invalid username - must be a string.")
                valid = False
                break

            if i not in username:
                print ("Valid Username.")
                valid = True
                break
        break

    else:
        print ("Username must be between 1 and 15 chars.")
```



Type of Test	Dry data	Explanation	Justification	Expected Outcome	Actual Outcome	Improvements
String and integers only	Jack117	User must enter a sensible name tag for their player	A string must be entered to not make data type errors when at scoreboard	Username accepted	Works as expected	N/A
Special characters	j@ck	Usernames will be of strings only of text and numbers	Could make it harder to display the scoreboard	Not accepted and error given	Works as expected	N/A
Insertion of images or links	N/A	The user could paste an image file or link in text field	Cannot be done as I haven't included any special libraries to make this possible	No way for images to be pasted	Works as expected	N/A
Username presence check	""	A username must be entered in order to continue. 'if username >1'	This is for the players name to be displayed on scoreboard. You cannot play as no one.	Error message occurs for length	Works as expected	N/A
Username length check	20 "1's"	A check to make sure the username length is >1 but less than 15.	This is to stop an overflow or underflow error. Prevents overflow attacks.	Error message occurs for length check (same message as presence check)	Works as expected	N/A

HOW EACH SECTION LINKS TOGETHER TO FORM THE OVERALL SOLUTION

At the end of each section I have briefly described where the sections of code will go and justification behind why they will go at these specific places, traversing more in depth in how this overall solution will be finalised will show how each section will concatenate together to help link all sections together.

The menu section is a standalone library that I have created, once the 'play game' button has been clicked a new library function that I have coded will run, this section is the username input page and allows the user to enter a username for their player which allows my game to be more dynamic, relatable to the user and enjoyable. Once the user clicks the 'play' button on the username page the menu state will end and will then load in a new library code module I have implemented to take the user to the main game, the main game state starts running.

All classes will go at the top of the main game code library and underneath will have the variables such as the player (instance of the player class) as well as the enemies (instances of the enemy class). I then have the code for the while loop for the main game state. Inside the main game state, I have the code for the bullet mechanics and physics which is followed by the peripheral controls of the user and then the border control. Finally, I then have the updating of the background as well as the collision detection system.

MAIN GAME CODE

My whole entire coding process for my project is separated into different python files, I chose this method as it allows them to be concise and self-contained for easier debugging and understanding what parts are being executed when, my reasoning behind this is to allow it to become easier to debug and will make the overall process and development of my game more readable and concise.

I coded specific code sections (e.g collision detection, main game, menu etc.) in separate python files like stated above, and then saved them in the same folder as the main game, this then allowed me to import python files within my original python code, see below.

```
# (x) top left, (x) top right, (y) top left, (y) bottom right
if (x) > 359 and (x) < 489 and (y) > 318 and (y) < 365:
    print ("play")
    import Game_Classes
```

This coding section is my main game code that the user will interact with the most. This is the code for the actual game and contains collision detection, bullet firing, rendering, initialization, player creation, movement, enemy creation and more, conducted through the method of Object Orientated Programming (OOP). My main game code can be seen below and have further analysis and information after the code dumps:

```
Game_Classes.py * x
1  # This is it made with Classes and it works well
2
3  import pygame
4  pygame.mixer.music.load('audio\CarpenterBrut.mp3')
5  pygame.mixer.music.play(0)
6
7  pygame.init()
8  screen_width = 850
9  screen_height = 550
10
11
12  screen = pygame.display.set_mode((screen_width,screen_height))
13  pygame.display.set_caption("EXHUMATION ANTHROPOLOGY")
14
15  #Init variables for movement
16  moveX,moveY = 0,0
17  x,y = 0,0
18
19  # The collision detection stops working atm, won't take long to sort out
20  # Collision detection - takes two objects,(x,y), width and height
21  def detectCollisions(x1,y1,w1,h1,x2,y2,w2,h2):
22      if (x2+w2>=x1>=x2 and y2+h2>=y1>=y2):
23          return True
24
25      elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1>=y2):
26          return True
27
28      elif (x2+w2>=x1>=x2 and y2+h2>=y1+h1>=y2):
29          return True
30
31      elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1+h1>=y2):
32          return True
33
34      else:
35          return False
36
37
38  # Class 1 - Class for creating player through OOP
39  # and links to collisons for the collision detection
40  # All classes follow the same structure of init then rendering
41  class Player:
42      def __init__(self,x,y,width,height):
43          self.x = x
44          self.y = y
45          self.width = 15
46          self.height = 15
47
48          # Player is now the medium sprite size to match the background size
49          self.i1 = pygame.image.load("sprites/Sprite2.png")
50
51      # Draw the sprites to the screen
52      def draw(self,collision):
```

```

53
54     # if there is a collision stop the player at the edge of the object.
55     if (collision == True):
56         player.x -= moveX
57         player.y -= moveY
58
59     # render the player to the screen with the x,y coordinates
60     screen.blit(self.i1,(self.x,self.y))
61
62
63
64
65
66
67
68 #Class 1b - Creation of enemies to attack the player.
69 class enemies:
70
71     #initialize variables to be used in OOP process
72     def __init__(self,x,y,width,height):
73         self.x = x
74         self.y = y
75         self.width = 20
76         self.height = 20
77
78     # Player is now the medium sprite size to match the background size
79     self.i2 = pygame.image.load("Designs/zombie1.png")
80
81
82
83
84
85
86
87
88
89
90
91 #Class 3 - Creation of missiles/ bullets.
92 class Missile:
93
94     def __init__(self,x,y,width, height):
95         super().__init__()
96         self.x = x
97         self.y = y
98         self.width = 15
99         self.height = 15
100        speed = 0
101
102        #load bullet image into memory through memory location
103        self.i3 = pygame.image.load('images/laser.png')
104
105    def draw(self,collision):
106
107        speed +=10
108
109        # render the bullet to the screen
110        screen.blit(self.i3,(self.x,self.y))
111
112
113 # All instances of the classes
114 # used in the OOP of the game
115 bullet = Missile(5,5,moveX,moveY)
116 white = (255,255,255)
117 black = (0,0,0)
118 player = Player(100,50,100,100)
119 zombie = enemies(400,300,100,100)
120 points = 0
121 perks = []
122 powerups = []
123
124
125
126 game_state = True
127
128 while game_state:
129
130     # Every process is an event which can change the outcome of te game
131     for event in pygame.event.get():

```

```

132     if (event.type==pygame.QUIT):
133         game_state = False
134
135     elif event.type == pygame.MOUSEBUTTONDOWN:
136         # links to the 'Missile' class
137         Missile(5,5,moveX,moveY)
138         print ("fire.")
139
140
141     elif (event.type==pygame.KEYDOWN):
142
143         if (event.key==pygame.K_LEFT or event.key== ord('a')):
144             moveX = -30
145             print ("player.x: ",player.x,"player.y:",player.y)
146
147         elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
148             moveX = 30
149             print ("player.x: ",player.x,"player.y:",player.y)
150
151         elif (event.key==pygame.K_UP or event.key== ord('w')):
152             moveY = -30
153             print ("player.x: ",player.x,"player.y:",player.y)
154
155
156         elif (event.key==pygame.K_DOWN or event.key== ord('s')):
157             moveY = 30
158             print ("player.x: ",player.x,"player.y:",player.y)
159
160
161     # Program stops movement of player when keys stop being pressed.
162     elif (event.type==pygame.KEYUP):
163         if (event.key==pygame.K_LEFT or event.key== ord('a')):
164             moveX = 0
165         elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
166             moveX = 0
167         elif (event.key==pygame.K_UP or event.key== ord('w')):
168             moveY = 0
169         elif (event.key==pygame.K_DOWN or event.key== ord('s')):
170             moveY = 0
171
172
173     # Loads background into memory so the player and enemy are rendered on something.
174     bg = pygame.image.load("Designs/setting2(features).png").convert()
175     bg = pygame.transform.scale(bg, (screen_width, screen_height))
176     screen.blit(bg, (0,0))
177
178     pygame.display.update()
179
180     # update players movement
181     player.x += moveX
182     player.y += moveY
183
184
185     # Border Control - stops player wandering off of screen.
186     if player.x > 790 or player.x < 0:
187         player.x -= moveX
188
189     elif player.y > 480 or player.y < 0:
190         player.y -= moveY
191
192
193     # Test for collisions of the player and an object or enemy
194     collisions = detectCollisions(player.x,player.y,player.width,player.height,zombie.x,zombie.y,zombie.width,zombie.height)
195
196     #Links and goes through 'Player' and 'collisions' class
197     Player.draw(player,collisions)
198     enemies.draw(zombie,collisions)
199     zombie.draw(False)
200
201
202     #Inits the frame rate and the cycle clock
203     FPS = 150
204     clock = pygame.time.Clock()
205     clock.tick(FPS)
206
207     # Updates the screen while True in game_state
208     pygame.display.flip()
209     pygame.quit()

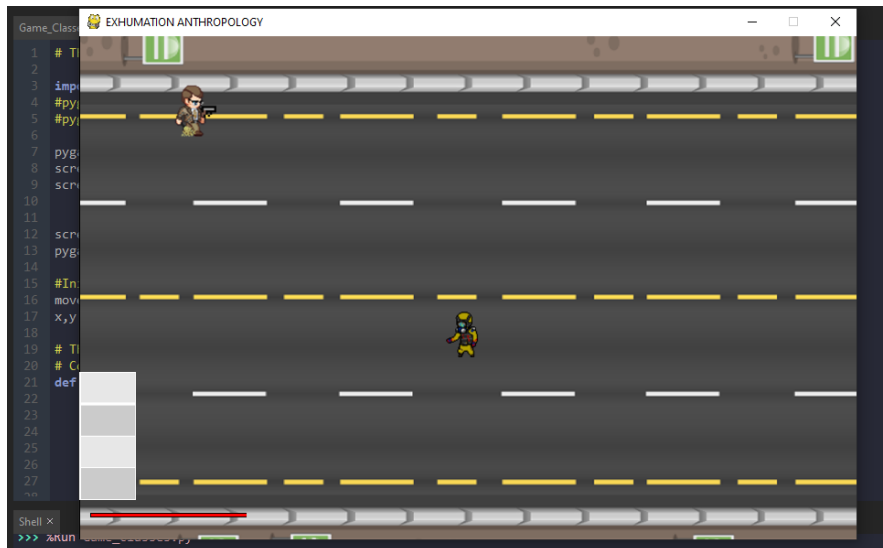
```

Below I have concise analysis of each section of the main game code from above, doing this will make it easier to understand by going through each segment step by step and explaining the importance of it. Some of this code I have explained further up (at the start of the Development section), the rest can be found here.

Candidate Name: Cameron Noakes

Candidate Number: 9453

I have also now improved the background setting for my game, I have implemented a health bar (static) as well as an inventory slot on the left side, this is to give a better feel for the game and make it more enjoyable for the end user as they have to be more careful when their health drops, I have also made my sprite hold a gun. See below:



The main aspects of my actual gameplay code was referenced before where the testing was done.

I continued with my actual main game and the features of it, on the username input field I had an issue with not being able to add in a text box easily for a username input, which is one of my main features that is unique and I liked the idea of. After a break from trying to implement it I realised I could just have a normal Python variable and assign it to a user input, `username = input("")`

This allowed the user to type in an input in the terminal and save it in the main game under a variable name. I then assigned it to only work when the username input box was clicked, to initialise it ready for the input. This can be seen below.

```
#Username input field clicked - enter username in terminal.

# (x) top left, (x) top right, (y) top left, (y) bottom right
if (x) > 245 and (x) < 600 and (y) > 238 and (y) < 262:
    username = input("Enter your username: ")
    print (username,"successfully registered.")
```

I did this through the previous method of coordinates of the buttons and it seemed to work pretty well, now the user can input their name in the terminal and it can be a part of the main game. Alternatively, if I didn't get past this issue, I would've had to have the user choose a player to play with because I wouldn't have been able to implement the username feature.

I have am now focusing on how I will implement the enemies (zombies) into my game, I have created an enemy class at which I render them and compare coordinates of their own to the coordinated of the player. This allows me to pass in the zombie attributes into the collision detection parameters to check if

a collision is true. Then I rendered the player and zombie to the screen, the collision detection at the moment is a bit off but other than the player glitching out a little it is working correctly again.

The reason for me reinstating the collision detection module, even though I finished it a while ago is because when I implemented the classes into my code to make it more efficient a accidentally broke the collision detection and have now made it again but just need to touch it up a little. My enemy game class can be seen below in its entirety.

```
#Class 1b:
class enemies:
    def __init__(self,x,y,width,height):
        self.x = x
        self.y = y
        self.width = 15
        self.height = 15

        # Player is now the medium sprite size to match the background size
        self.i2 = pygame.image.load("Designs/zombie1.png")

    def draw(self,collision):
        if (collision == True):
            player.x -= moveX
            player.y -= moveY

        #render object to the screen
        screen.blit(self.i2,(self.x,self.y))

# Class 2
```

I have also at this point resized the zombie, so it fits with the other size of the players character, this makes my game seem more realistic and not daunting on the player of oversized zombies.

I have made sure to include comments in the code to help other programmers understand what my code does and how to debug it if it is needed. Comments in the code help to make code more readable and concise and overall be more coherent, allowing easy reading of it all and what each section of code does what.

For the perks and powerups I have an array that will be able to store a static amount of items to help the player, this is on my first column of my systems diagram, I felt having an array was a good idea as it can store multiple different items which can be perks or powerups. See below:

```
jeep = draw_objects(800,350,130,120)
points = 0
perks = []
powerups = []
```

Points stored I chose to be an integer that would increment as the player kills enemies or survives a wave, as points will be constantly changing and would only be incrementing it was correct to have it to store a number only. These points would be displayed when the player dies in the game.

MISSILE-BULLET CODE

I created a new class for the bullets and called it 'Missiles' this initialises the variables and is linked to the input of the user for the left mouse button click. When the left mouse button is clicked it will print a 'fire.' Terminal message for easier debugging and calls the 'Missiles' class, this creates the bullet and updates its position when moving across the screen. This class is to allow the player to shoot bullets at the enemies, and then update its position so the bullet continues until either it hits an enemy or travels off of the screen window.

```
class Missile(pygame.sprite.Sprite):
    #create missile
    def __init__(self, image, x, y):
        super().__init__()
        self.image = image
        self.image = pygame.transform.scale(image, (25, 38))
        self.rect = self.image.get_rect()
        self.rect.centerx = x
        self.rect.bottom = y
        self.speedy = -10

    def update(self):
        screen.blit(missile_img, moveX, moveY)
        #updating position
        self.rect.y += self.speedy
        if self.rect.bottom < 35:
            self.kill()
```

I later decided to improve my OOP approach when it came to the 'Missile' class as it looked messy and overcomplicated so I made it neater and easier to read as well as took out unnecessary code that didn't change the overall outcome of the OOP class of 'Missiles'. A newer version of this class can be seen below:

```
#Class 3 - Creation of missiles/ bullets.
class Missile:

    def __init__(self,x,y,width, height):
        super().__init__()
        self.x = x
        self.y = y
        self.width = 15
        self.height = 15
        speed = 0

        #load bullet image into memory through memory location
        self.i3 = pygame.image.load('images/laser.png')

    def draw(self,collision):

        speed +=10

        # render the bullet to the screen
        screen.blit(self.i3,(self.x,self.y))
```

I decided to implement it this way as it was the easiest and most efficient way to do so, allowing the use of Object Orientation it allowed me to continuously call this class every time the left mouse button is clicked by the user. Saves duplication of code later on for each iteration of a bullet.

```
while game_state:

    # inits the events the user could choose
    for event in pygame.event.get():
        if (event.type==pygame.QUIT):
            game_state = False

        elif event.type == pygame.MOUSEBUTTONDOWN:
            Missile(missile_img,moveX,moveY)
            print ("fire.")
```

Above is the event that is triggered when the mouse button is clicked, this 'fire' message is shown in the terminal to show that the game has registered a shooting.

The justification on why I have made sure to include bullets is due to the fact that deductions will take place in order for the enemies to lose health and eventually, in turn, die off. Bullets are the pointers that will point to deductions when collisions are detected.

STAKEHOLDERS INPUT AT KEY DEVELOPMENT FEATURES

In this section I reached out to my stakeholders again to help aim in the development process to make sure they are happy with the current and continuous solution of the actual coding of the game, below is some questions I asked both of them in regards to how the development process is going at the

moment and if they wish to amend and features or change any in any way. I gave my game solution to my stakeholders for useful feedback, below shows the feedback between both stakeholders Jake and Jai.

The best response was taken from the best of the two stakeholders for each question.

Q. Are you happy with the designs of the sprites (pixel art, downloaded, self-made etc.)

Yes, the sprites have been made yourself and not downloaded, it's unique to your game and pixel art is a good idea for it as they are a good design for retro games.

Q. Does the menu appeal to your designs, if not, any feedback or changes?

The design for the menu is good and can't see anything that has been missed out, maybe add a 'hint' button to help the user get straight into the game.

Q. Any feedback on peripheral controls and do they work?

The controls are good and match what I said in the first interview, so I am happy with them, I'm glad there is alternate movement controls.

Q. Does the collision detection work?

When I control the player, it stops at the edge of the enemy on the screen, yes it works, could get closer to the enemy by a few pixels maybe.

Q. Any additional features overall to add?

Seems to work fine and looks good, all parts seem to work correctly and couldn't find any errors.

As a result of this feedback I have improved how close the player can get to the enemy, I altered the pixel space taken up by the enemy, allowing the player to get closer.

OVERVIEW OF MOST ISSUES AND FIXES

Collision Detection – 11th October 2019

I had a real struggle with implementing a theoretical approach to the collision detection, I found a tutorial online which covered the basics, however, I still didn't understand, later on I found a solution to make it so if an object is in one of the corners of another object, the collision detection is true. The video I watched for this changed the object to red when inside the other object, I didn't want to do this, alternatively, I wanted to stop the player at the edge of the object (due to physics laws), I couldn't find a way of changing the x and y coordinates of

the player to allow this, I have 2 variables, x and y then moveX and moveY, the x and y variables are localized to then manipulate the ending moveX and moveY variables, these ending variables are the actual ones that respond to the game and move the player. I tried setting them to zero when a collision occurs, however, this just reset the player to coordinates (0,0). I then tried deducting the objects width and height from the players x and y coordinates, but this only made the player teleport down to those coordinates (this might be handy later for the teleportation machine). I finally got it working after I changed the players (x,y) coordinates to minus the moveX and moveY, stopping the player at the edge of the object.

Player flickering and loading background – 14th November

Originally, I came across a graphics error when importing the background, it was lagging the player movement, the player movement was much smaller in motion and the player sprite kept flickering. I changed the background, increased the frame rate of `clock.tick(150)` and still didn't work, I came across a stack overflow error page for why this could be, someone said it could be the aspect ratio of width and height of the background that didn't match the window size in pygame, proceeded to spend 10 minutes on converting pixels to inches and saving a new screen size, ran my code and didn't change. I then looked at my code again and debugged it and found out that I have put two `'display.update.flip()'` this was the cause of the lag and flickering as it kept switching between each state of the player, it worked fine after that.

OOP for Player and Objects – 15th November

I was stuck a long time on using OOP for sprite and objects, I made it so the class of 'Sprite' has two values, the player and an object, the player was only drawn to the screen, then after a while of trying to fix the issue I got the player and the other object on the screen but you controlled the object as well. I did not want this, I want a controllable player and an object in the way, today I decided to give it another go as I cannot go forward if I cannot render player and object to screen and control only the player sprite. I checked out one of the very few tutorials I had left at my disposal for OOP for sprites and objects this gave me an advantage and a deeper insight to how classes work, I had an overview to begin with, however, I did not know how to debug them, watching this video allowed me to pause it half way through and create two different classes, a spark initiated which allowed me to see exactly what I had to do, one for the sprite with attributes and another for drawing the objects. This was a big breakthrough

in my project. At the minute I have broken the collision detection however, I have now got OOP. A follow up on the collision break will be soon.

The Menu – Mouse Coordinates - 22nd November

In the morning of today I managed to (in one hour) load the menu, reduce and abstract my main game, transforming it into the menu. I now have two main programs, the main game and the menu. These are separate for easier debugging; I will then concatenate them together once I have them perfect and working. I thought of all the ways how a mouse's coordinates could 'hit' a button, as my background is just a background, it doesn't show the coordinates of the buttons on the screen (as it is just a picture and self-contained), I began to focus mainly on getting the x,y coordinates of the mouse instead and would come back to the button coordinates.

I used a bunch of resources (given in resource section) and eventually found good documentation on the process which helped me to code mine up, using MOUSEMOTION I managed to grab the coordinates and print them out in the terminal when they hover over the pygame window, I tested this by hovering my mouse over and seeing a continuous scroll of x,y coordinates. This then hit me... I could use these coordinates of the mouse to find the coordinates of the buttons on the background, I recorded them by a notepad and putting my mouse pointer to every corner (example of one button given below):

TL = Top Left

(186,158) TL of 'play game'

(373 161) TR of 'play game'

(185 229) BL of 'play game'

(372 230) BR of 'play game'

```
# Mouse coordinates
event = pygame.event.poll()
if event.type == pygame.QUIT:
    running = 0
elif event.type == pygame.MOUSEMOTION:
    x, y = event.pos
    print(x,y)
    if (x) > 186 and (x) < 373 and (y) > 158 and (y) < 230:
        print("play game")
```

```
Shell x
186 229
187 229
play game
186 228
83 235
0 341
```

The Menu- Changing to different pages – 26th November

I have been stuck on the switching the screen state for quite a few days now, I have managed to fix it just now though, it would switch to the 'overview page' when that button is clicked when in the menu but then flicker back to the menu screen and alter the graphics into a mesh of both screens, I debugged it and found out that both pages were loaded correctly but it kept calling the menu screen based on the menu state of being true. I created a subroutine for the menu and overview page which allowed me to call the menu function before the menu while loop, then inside the loop, call the overview function when that button is pressed.

OVERALL DEVELOPMENT SUMMARY

The development of my game took the most amount of time but with each time increment it made me enjoy the process a lot more as the solution to the problem became more coherent and more finished as I continued with the process. I spent a lot of time coding and researching and a bit more on fixing omissions, some of the omissions were simple fixes like changing variable names but some were harder, for example the decision to redesign my code for the structure of Object Orientated for efficiency and readability improvements which would also make my code more concise, having many instances of the class instead of coding the same segment of the solution a number of times for each enemy or bullets.

Overall, the development cycle and segment of this project was challenging at some parts, and others were straight forward, the errors that occurred allowed me to identify new pieces of information regarding pygame and how the game library operates.

D. EVALUATION

This testing is to help show the overall performance of my solution, this will result in a better written report of my processes and how well they work separately as well as when concatenated together in one program for my ending game.

Testing allows me to specifically state what went well and what needs to be improved upon as well as the overall analysis stage of my code. It also allows me to identify potential issues in my code alongside seeing if I can make my code more efficient for the overall process of my game.

TESTING TABLES

OVERALL MENU TESTING

TESTING	EXPECTED RESULT	ACTUAL RESULT	IMPROVEMENTS IF ISSUE	VIDEO EVIDENCE
A,W,S,D	No action on the menu	This is a successful test as no action takes place when these buttons are pressed.	Could give an error saying these are invalid command	1.3
Arrow keys (left,right,up,down)	No action on the menu	When on menu the game is irresponsive to the Arrow keys	Could give a custom error message saying invalid command	1.3
Check all menu buttons work	Show the relevant pages for the buttons clicked	Yes, executes the new pages that should be loaded	N/A	1.3
Exit button	Exits the game immediately	Successful testing, no user kept waiting, interrupt is prioritized.	Could add a keyboard shortcut to close key to the game e.g. "F5" to close immediately	1.3

OVERALL USERNAME PAGE

TESTING	EXPECTED RESULT	ACTUAL RESULT	IMPROVEMENTS IF ISSUE	VIDEO EVIDENCE
Username less than 5	Not allow the username to be registered and provide a suitable error message.	Performs what it should do.	Created a custom error message for length of username	1.4
Username between 5-10	Allows the user to continue to play the game	Username is accepted and the test is successful.	N/A	1.4
Username over 10	The same error message as <5 is repeated until a valid username is entered	Provides an error message as successful testing.	Created a custom error message for length of username	1.4

<i>Username of numbers</i>	<i>Allows this request</i>	<i>Numbers are accepted in username</i>	<i>N/A</i>	<i>1.4</i>
<i>Username of pasted picture</i>	<i>Not allow this input</i>	<i>Doesn't allow this input as it is a string of characters not through pasting.</i>	<i>N/A</i>	<i>1.4</i>
<i>Username of string</i>	<i>This is the only type of valid username input with integers, allows this username</i>	<i>Username accepted as the input is of string value</i>	<i>N/A</i>	<i>1.4</i>

RUNNING GAME STATE TESTING

TESTING	EXPECTED RESULT	ACTUAL RESULT	IMPROVEMENTS IF ISSUE	VIDEO EVIDENCE
<i>Player Movement</i>	<i>player moves in direction of motion</i>	<i>Works as expected. alternate movement via arrow keys as well as A,W,S,D.- Works as expected</i>	<i>N/A</i>	<i>1.5</i>
<i>Firing</i>	<i>Register the firing of bullets and render them moving to the screen</i>	<i>Registers the bullet firing however does not render any image to the screen and move it in x or y direction. This is a rendering issue.</i>	<i>Would need to alter the arrangement of the code to allow the bullets to be rendered, this is a bug within the 'Bullet' class that I cannot figure out at this point in time.</i>	<i>1.6</i>
<i>Collisions</i>	<i>Stop the player when a collision is met.</i>	<i>The player stops at the edge of the object, making a successful collision detection. Works as expected</i>	<i>N/A</i>	<i>1.1</i>

RENDERING TO THE SCREEN

TESTING	EXPECTED RESULT	ACTUAL RESULT	IMPROVEMENTS IF ISSUE	VIDEO EVIDENCE
Player rendering in the main game	Player should be rendered to the screen	The user can now control the player on the screen.- Works as expected	N/A	1.2
Rendering of enemies	Enemies are then rendered to the screen	Works as expected	N/A	1.2
Rendering objects	Objects should be rendered to the screen in the main game	This is a successful test as all objects are rendered to the screen upon startup.	N/A	1.2 and 1.5
Rendering Bullets	Render the bullets moving across the screen	Bullets are registered as fired but no bullets fly across the screen	Need to alter the code and implement a rendering system for the bullets to be seen on the screen	1.6

SOUND EFFECTS

TESTING	EXPECTED RESULT	ACTUAL RESULT	IMPROVEMENTS IF ISSUE	VIDEO EVIDENCE
Load Sound	The sound should be loaded when the initial game is started, and the menu is opened.	Works as expected.	N/A	1.3
Load other sounds over the top (sfx)	Additional audio for overlapping the main game for a dynamic noise approach	No additional audio has been used	Downloaded a free sample of firing and added code to play audio of a gun fire every time the peripheral of the fire button is clicked.	1.7
Change music based on screen	The music should change from the menu music to the main game music after a valid username is entered.	All testing for this process is successful. Works as expected	N/A	1.8
Switching tracks	Stop one track to play another	Yes, the music track fades out then after it goes the new track is loaded in.	N/A	1.8

USER SIGN OFF AGREEMENT

Stakeholder interview

Q. Are you happy with the current progression of my game?

Jake: Yes, a lot of time has been spent on it and looks like an arcade game that once finished could be on the PC store.

Q. Would you like if the game was continued to be developed?

Jake: Yes, because there are a few features left out that kind of need to be completed before being finished based on my original interview statements such as powerups.

Q. If so, what additional features would you like to see?

Jake: I would like to see the powerups completed and working, allowing the user to pick them up and use them but also would like to see the teleportation machine as it seems like a cool idea.

SIGNATURE OF APPROVAL

Please sign below if you are happy with the overall project progression and is an appropriate solution and are happy with the features outlined in the meeting during design.

.....

Jake Parker

EVALUATION OF SOLUTION – USABILITY FEATURES

Written Evaluation/ Usability features

- Game Controls

As agreed with Jake and Jai (my stakeholders) I am going to keep my controls similar to what they wanted to have, they wanted a retro style 2D game, which, in the past, had very easy controls so I have

adapted this to my playable game, they did not explicitly state to have easy use controls, however, I felt this would be more suitable to the types of games they were referring to. I have A,W,S,D as normal controls as well as an alternate for selective users of the arrow keys, depending whichever one suits the user more. Overall, the controls are basic and simple allowing the player to be directed around the screen by the user.

For this game I have created as a solution to a problem, the user will control a player, this player will have the ability to move Up, Down, Left and Right, in turn, will allow the user to have full dynamic control of the player character/survivor of my game.

- Powerups/ Perks

Jai wanted to see perks or powerups in the game, so did Jake as Jake plays a lot of Call Of Duty Zombies and they have them in there, this is one of my main factors in which I decided to create the idea as it would be deemed more successful to my stakeholders if I structure my game around that of their favourites, however, I was unable to implement these at this time. The time allocation for such a minor feature, in my opinion, wouldn't be worth it, considering I still have more important features to include that have been delayed or halted due to time limitations.

- Collision Detection

Collision detection is successful in my solution and it is fully finished, the player can now collide with objects on the screen. This was a main part of my game as the player must be able to collide with the enemies.

The user can control the player and the player can move around the screen but cannot go through objects, this symbolizes basic physics and mechanics on how the game should function.

- Enemies

I have started to implement the enemies but is still not yet finished. I can render the enemy and player to the screen but haven't implemented pathfinding and updating the screen for the movement of the enemy but have done for the player. The player can move around the enemy and collision detection is present, allowing the enemies to be actual entities.

- Menu

There was no specific requirements from the stakeholders for the menu, so I decided to use my initiative and implement different pages and features to help the end user play the game and understand it. Some of these pages is a username page, controls page, help and overview page. The menu is fully complete and is not missing any features that I wanted it to have. The menu is loaded when the game starts and the user has a variety of buttons to press to return different pages, the buttons are coherent and are easy to understand their use.

- Customisable gamer tag

I have made a username input page that is dynamic and allows the user to input a screen display name to add more of a feel to the dynamic approach to my solution. I have also added parameters to appeal to cyber security by blacklisting special characters for the username input, to make sure my game

cannot be exploited through Cross site scripting (XSS) or another feature that would allow my game to not function as it is supposed to. The user will enter a custom gamer tag and if accepted based on the rules of validation it will allow the user to continue into the main game

- Bullets

I have the code for bullets to work as well as a separate class for it to be self-contained, however, I have not managed to fully implement the firing system to allow firing of bullets and rendering them to the screen. The player can left click and it registers as a fire however no bullets appear on the screen. The user has the option for the player to fire bullets, firing is done through clicking the left mouse button which triggers the 'Bullet' class to be called as well as a firing noise is heard.

- Rendering the player

I have completed all code to allow the rendering of the player to the screen and updating it every time there is a change in the game state. The rendering of the player and objects is finished.

- Peripherals

All peripherals are working and up to date with no omissions or errors, the player can move, and any movement is updated to the screen. The inputs have been linked to the correct features the user can access like movement or firing. This section is fully completed.

- Limitations

There are limitations within my solution as I have not completed the development, however, these limitations are based around the modules that I have completed. The limitations were due to time frames at which I was not able to either finish developing the solution or start doing them, this, as a result, has made my game slightly unfinished, however, do wish to continue developing it in the future until it is fully fledged game. The processes and blocks of code that I haven't been implemented are as follows:

- Rendering bullets to the screen (and moving them)
- Rendering and the moving of enemies
- Powerups/ Perks and other small features
- Deductions class
- Pathfinding class
- End game screen and leader boards/ writing to external file

DOES THE SOLUTION MEET THE REQUIREMENTS?

Success Criteria from Analysis section is given below with additional columns on the end to show if they were changed in the final solution or any features have been amended.

Requirements	Justification	Does the final solution suit the criteria?	Future additional Improvements
Only one map	As stated by Jake (page 11) the game should have 1 map design 'to make the game more enjoyable'	Yes, it does, I have only got one map for simplicity	Could add more maps later on as an update

	and concise to help keep the player interested and continue to play the game.		
Menu becomes fully functioning and has buttons that bring the user to different pages	The user shouldn't just jump straight into the game, the menu allows information to be given in order for a successful completion of the game and so the user understands how to play	Yes, works as expected and matches the criteria I had set out for it	N/A
Menu has music in the background (soundtrack)	This was not a specific requirement from the stakeholders however, I feel that this would have create a more interesting game experience which is overall what both stakeholders want	The final solution matches the success criteria for backing tracks	Could loop the menu music so it continuously plays
Soundtrack will change when entering the main game	This would allow a more unique and interesting experience as also said above.	Final solution meets this criteria	N/A
Main game will have a backing track of rock/hardcore music (maybe Carpenter Brut)	Overall improves the experience of my game on the end user and makes it more enjoyable	Yes, Carpenter Brut music plays for gameplay music, works as expected.	N/A
More than one weapon	Jake requested on page 11 to have a variety of weapons to keep the game interesting	The final solution currently only has one weapon available and is the starting handgun.	When After publishing the game solution I will add this feature in
Menu screen options of 4	On page 12, Jake decided that 4 options on the menu screen is enough to help give the information each user needs for when they play the game.	Menu solution matches this criteria if you do not count the 'exit' button as this button has minimal effect	N/A
Have powerups in the main game	Jake brainstormed that having the chance of 2 powerups between level 1-5 then after level 5 the	Unfortunately, I did not get around to implementing powerups for my	The second update will have powerups implemented that would allow the user to pick up and

	amount of powerups goes up, may increment it by one every 5 levels, however, will cap at 8 otherwise it will be too easy to survive. This does not mean 8 powerups per level, it means the chance is now 8/10 to get a powerup each level. (page 12)	game due to time limitations	use, this update would happen after publishing the game.
Rounds/ levels 1-4 will have enemies that die after 2 hits	Jake proposed that between rounds 1 to 4 the enemies will die after 2 hits, one to weaken them and another to finish them off. This makes the health system easy to implement as an enemy's health can be 100 and the damage of the start gun can be 50.	Final Solution has changed a little based on this success criteria, the player has around 5 hits until they die and there is no rounds, there is only one continuous round that plays.	An update in the future could introduce a hardcore mode where you only have one life.
The first level will have 13 enemies	Jake stated in page 12 that having 10-15 enemies in the first level will make the game easy to focus and enjoyable as new users can grasp the controls.	There is a continuous round so there is no need to disclose how many will spawn at once.	The pathfinding for enemies would be completed before publishing then the enemies would move correctly.
The starting weapon will be a handgun	A handgun is basic and low damage, Jake also recommended to have a knife that is a one hit kill (100 damage) to make the player risk their life more (page 12)	The final solution for this piece of criteria is the same, the user starts with the handgun like stated and works well and gives off a gun noise when fired.	N/A
100 points per kill	Jake suggested having 100pts per kill as it is simple to use in calculations and also it is very easy for the user to see what they can afford. (page 12)	The points system hasn't been implemented just yet so is hard to evaluate this based on nothing, final	Before publishing the points system would have to be working correctly, giving points and have a deduction system at hand.

		solution does not meet this criteria	
15 year old rating in order to play the game	Jake said that my game would be aimed for 13-18 year olds, so I decided to go in the near middle with age rating of 15 due to the gore and effects I will implement (page 12 is Jakes summary on why)	This will still be the same as this game could be graphic one the blood splatters have been implemented but due to time restraints this isn't possible at this point in time.	Implement this feature either before publishing or as an additional update for experience and not performance.
3-4 minutes for the killing of every enemy in the level until the next level starts	This makes sure the player is always on their feet about killing the enemies and isn't hiding, as the longer you wait, the more enemies will come. Jake references this in page 12.	A timer has not been implemented due to time limitations; final solution does not match the success criteria for this part.	Add in timer before publishing
Leader boards will be based on points	Jai has stated that a point based system for the leader boards will be more rewarding (page 14)	Leader boards are based on points, success criteria is matching with final solution	N/A
The game and menu will have different music	Jai requested to have different music for the menu and game, and I see this working very well (page 14 references this)	Final solution meets this criteria	N/A
Each weapon will have a different firing sound	Jai explained how having a different sound for each gun will create more sense or realism (page 14)	There is currently only one weapon in the game but if I had more time I would include more weapons and different sounds for each	Before publishing be sure to add more weapons
Player will have 1 life or a few hits until death	Jake agreed this will make my game more realistic and harder. (page 12)	Final solution meets this criteria	N/A

Does the solution of the menu meet the success criteria and requirements?

- Outlined in the 'Analysis' section (the same section of success criteria and requirements table) the menu needs to be fully functioning – After fully completing the menu I can say that the menu works in its entirety and so do all of the buttons, each button brings the user to a different page allowing the design for it to be acceptable to the solution and to be accurate to what I depicted for it.
- There is 5 options on my menu page, 4 are primary buttons and the other is just the exit button. So overall I have 4 main option buttons which is the amount specified in the success criteria and requirements table
- In the success criteria it was stated that the menu will have some sort of background soundtrack music to supplement the game solution, I have implemented this so consider this piece of the success criteria met.
- The background music will change once off of the menu page and in the main game, the menu music will fade out on the username page once the 'play' button is clicked then will load in the new soundtrack for the actual main game. This is successful and fulfils the requirements stated in 'Analysis'.

Does the solution of the main game meet the success criteria and requirements?

- Only one map has been created allowing the solution to be easy to navigate and for the user to get familiar with the design of the level, allowing them to stay alive longer, one map was easier overall as it didn't require unnecessary designing of another map that wouldn't hinder the end game. In conclusion, the success criteria was met.
- Another piece of success criteria was to have multiple weapons to allow more variation in the game, however, I didn't manage to get around to implementing this due to time limitations being the only reason, if I had more time I would implement this through pressing a key on the keyboard which would allow the switching to the next available weapon.
- Powerups were an additional item I was going to add into the game after everything was finished and I had some spare time, unfortunately, I did not have this additional time and if I did I would've spent this time on more important factors like pathfinding implementation and coding a deduction system for bullets and health.

- Although it was not part of the success criteria, I would still like to mention the enemy pathfinding algorithm, if I had more time as well as better knowledge on A* pathfinding algorithm I would've coded it up for the final solution to the problem.
- A deduction system would also be reasonably simple but due to the same continuous reason above I was unable to find the time to complete this self-request, if I had more time I could implement this.
- Rounds were not important in the end as it would just be one wave of endless hordes of zombies.
- The starting weapon is a handgun, this is because it was referenced in the success criteria and requirements and decided to implement it as it didn't take much time to code and get working, the gun sprite spawns in the players hand when the main game starts, this success criteria was met.
- As I haven't coded up a deduction system through OOP classes, I haven't managed to get around to giving points to the player for every successful kill, this would've only been useful for the leader boards as well as the in game shop to buy new weapons with the collected points. This has not been completed which was outlined in the success criteria due to time restraints.
- In the end a timer was not needed, the timer would countdown until the next round of zombies spawned but as there is only one continuous round it was deemed unnecessary and thus was removed.
- The leader boards have not been started due to all other aspects being delayed and postponed due to the time constraints and due to the prioritization of the sections, this section would have been near the end of the additional time if I was ever given a new deadline.
- Movement has the same basic controls as outlined in the design phase which would make the movement of the player easy to navigate around the screen

Have any of the menu features been changed from its original design?

In the design stage I have a clear and coherent view of what my menu would look like and the features that it would contain, therefore, I can say that no features were changed upon development of the

menu, the criteria stated in the design phase is in alignment with that of the implementation of the menu and no features have been missed out or removed necessarily.

Have any of the main game features been changed from its original design?

Yes, features have been altered, added as well as deleted in the implementation and development stage, this mainly was due to the fact of time limitations limiting my coding and ability to add in these features. The features that don't completely reflect the stated ones in design are given below.

- Levels were taken out due to a different change in the game I wanted to make, I wanted to have one continuous wave of enemies and not to have waves of them as they are too generic and are in most zombie games. This feature was deleted to give a more overall unique experience to the solution.
- The leader boards were not in the development phase due to me running out of time to implement them, I only designed a rough abstract idea of the leader boards page so the time on the designing of it wasn't greatly wasted as if I had more time I would base it off of the original design I had in mind for it.
- Difficulty was taken out as for each increment of time the enemies health would continuously grow with each new spawn as there are no levels and it is just one continuous wave.
- Powerups and points were on hold due to them being one of the last things to be implemented as there is higher priority tasks I have put off due to time constraints.
- Upgrades and weapons – Upgrades have been taken out as the map is small enough as it already is and do not need it to become overcrowded with unnecessary items and features. Weapons would have been implemented more other than the starting handgun but due to running out of time I have had to put this feature also on hold.

MISSED FEATURES

Due to time restraints I was unable to complete or start specific features of my game, if I was given more time these are the features (in sequence) that I would work on to make my solution overall much better and collective.

A* Pathfinding Algorithm

I was unable to start the pathfinding algorithm. The algorithm I would've used would have been A* pathfinding as it seems the most applicable to my solution and easier to code which could have saved

time when implementing other features that have been missed out. The pathfinding would have been for the enemies to find where the player is on the screen so they can follow him, opposing a constant threat towards the player and add tension to the game, I would have created another class that would have been called each second the screen is re-rendered to see if the player has moved and if the enemies should move closer.

End game screen/Leader boards

This feature would have been the end screen the user would see when the in game player dies, this end game screen would display relevant information based on the player and previous players of the game through a leader board, this leader board would be from a text file that is written to each time a game is played, this would allow constant updating of new winners. This would be less crucial than the pathfinding but more crucial than powerups as the powerups would have been an additional feature added at the end if I had spare time.

Deductions

A class would have been created that would see if a deduction was to be made, this would alter the overall logic of my game, if one bullet is fired remove this bullet from the gun. It is deemed necessary due to the fact of the user not having infinite number of bullets and health, this class would also account for the overall health of the enemies, with each successful hit the enemy should lose health, this is the same for the player as well, this class for deductions would make that the game continues to function as it is supposed to. I would go about implementing this by having an OOP class that would pass in values of the players bullets, health and enemy health ready for deductions to take place based on the collision detection system.

Powerups and other additional features

Another class would be made to help keep the code readable and follow the same format as previous code for the game and would randomly render and spawn powerups around the screen for the player to pick up and interact with, other features would include a teleportation system to teleport the player to different parts of the map and a armoury to buy new weapons. These features are very low level in terms of hierarchical and would be implemented last if any time was spare, these do not hinder my game from working but would add more of an experience to the game overall. The justification on why these features have been left out are because they are the last to be implemented and is due to them being unnecessary for the overall function of my game and do not hinder the results or the solution in any way.

Rendering

There is somewhat a small issue with rendering fired bullets to the screen and moving them across but due to the few other features that have been left out I have decided to not focus on this minor rendering issue either due to the fact that the game identifies the bullets being fired and just doesn't show them, showing that my game logic is fully functioning when regarding the bullet and firing system.

Pausing

The pausing feature has been pending for some time now due to the fact of having other features to implement first, the pausing feature is relatively simple to include, when 'p' is pressed the game state is changed to false and the pause screen pops up, this is how I would implement it. The reason behind doing it this way is because it is the simplest and doesn't need harder solution to be implemented for this to work.

BUGS TO FIX BEFORE PUSHING THE CODE TO PRODUCTIONRendering of bullets

The issue with bullets being rendered to the screen would need to be fixed, this is a minor issue and could be figured out not long after this process of fixing has started, the coordinates of each fired bullet would have to be incremented in the X or Y direction depending on the direction the bullet has been fired.

Changing the facing direction of Player

This is not a bug, but the player could face the opposite way when moving backwards, different player sprites could be loaded depending on which direction the player is moving.

Collision detection is not pixel perfect

When the player collides with enemies it stops near the edge of the enemy, this could be amended to make it pixel perfect by stopping at the exact pixel before the enemy pixels begin, creating a more realistic game when it comes to the collisions.

How the data will be stored

The data and information of the game must be secure so much so that it cannot be modified by end users, this ensures my game cannot be broken, reverse engineered or manipulated so it is only used to play my game the way it is supposed to be played. Python is interpreted therefore cannot simply just be compiled to an executable for the security to be finished, encryption would have to be used on the points system if ever implemented as well as the username and code as well to stop the end user reading these and changing the values.

UPGRADES

- Upgrades could allow features to be implemented at a later date when more time and resources are discovered. Powerups are not currently in my game but could be as an update, updates keep the game fresh in people's minds and creates overall a better performance and experience as the game is dynamic and keeps changing.

- More map designs could be added to change the scenery of the game to add variety. As my game is offline the game source code must be uploaded to a server each time there is a new update and then pushed to the internet where new versions can be downloaded due to the fact that my game is offline therefore cannot simply just update the game over the internet.
- At a later date as an update there could be a feature that allows you to pick styles for the player you control, this would allow a more dynamic and interesting approach to my game and hopefully make the overall experience much better due to it being unique to the end user.

PROJECT APPENDIXES

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

- Complete Code Listing (ESSENTIAL)
- Interview Transcripts
- Meeting notes
- Observation notes or questionnaires

APPENDIX A – CODE LISTING

In order for the successful screen code dumps without the issue of resizing and cropping I decided to upload my project code to my GitHub privately and then the format is changed into code format that I can then easily paste into this document.

I have coded using separate libraries that I have made through decomposition and for coherent-ness for them all to be self-contained for easier debugging. The 'menu' calls 'username page' and the 'username page' calls 'Game Classes' library.

Menu code dump is below:

```
import
pygame
import time
pygame.init()
```

```
# Menu Music
pygame.mixer.music.load('audio\HeavenFalls.mp3')
pygame.mixer.music.play(0)

# Back button works but glitches the screen again

screen_width = 850
screen_height = 550

mouse_x,mouse_y = pygame.mouse.get_pos()

x,y = pygame.mouse.get_rel()

#Set Window dimensions and title
screen = pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("The Menu")

menu_state = True
choices = True
choices2 = True
black = (0,0,0)

def menu():
    bg = pygame.image.load("Designs/Menu_design.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def instructions():
    bg = pygame.image.load("Designs/peripheral1.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def help_page():
    bg = pygame.image.load("Designs/help_page.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))

def overview():
    bg = pygame.image.load("Designs/overview.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))
```



```
while menu_state:

    menu()

while choices:

    # Mouse coordinates
    event = pygame.event.poll()

    if event.type == pygame.QUIT:
        exit()
    elif event.type == pygame.MOUSEBUTTONDOWN:
        x,y = event.pos
        print (x,y)

        # (x) top left, (x) top right, (y) top left, (y) bottom right
        if (x) > 186 and (x) < 373 and (y) > 158 and (y) < 230:
            print ("play game")
            import username_page

        elif (x) > 480 and (x) < 664 and (y) > 160 and (y) < 233:
            print ("Help")
            help_page()

        elif (x) > 186 and (x) < 371 and (y) > 284 and (y) < 356:
            print ("Instructions")
            screen.fill(black)
            instructions()

        elif (x) > 479 and (x) < 664 and (y) > 284 and (y) < 356:
            print ("Overview")
            overview()

        elif (x) > 698 and (x) < 839 and (y) > 462 and (y) < 520:
            print ("back")
            screen.fill(black)
            menu()

        elif (x) > 332 and (x) < 518 and (y) > 408 and (y) < 480:
```

```
        print ("Exit")
        exit()

    pygame.display.update()

    clock = pygame.time.Clock()
    clock.tick(70)

    pygame.display.flip()
pygame.quit()
```

My 'username page' code dump is given below:

```
import
pygame

import time
pygame.init()

screen_width = 850
screen_height = 550

mouse_x,mouse_y = pygame.mouse.get_pos()

x,y = pygame.mouse.get_rel()

#Set Window dimensions and title
screen = pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("Username Input")

#Variables for while loops
user = True
username_screen = True

def usr_scrn():
    #loads bg onto window screen
    bg = pygame.image.load("Designs/username_screen.png").convert()
    bg = pygame.transform.scale(bg, (screen_width, screen_height))
    screen.blit(bg, (0,0))
```

```
# Enters a loop for module of username page
while user:

    usr_scrn()

while username_screen:

    # Mouse coordinates
    event = pygame.event.poll()

    #exit game
    if event.type == pygame.QUIT:
        exit()

    # debugging test
    elif event.type == pygame.MOUSEBUTTONDOWN:
        x,y = event.pos

    #Username input field clicked - enter username in terminal.

    # (x) top left, (x) top right, (y) top left, (y) bottom right
    if (x) > 245 and (x) < 600 and (y) > 238 and (y) < 262:

        valid = False

        while valid == False:

            username = input("Username: ")

            #array stores invalid chars
            array = ["@", "#", "|", "<", ">", "?", "[", "]"]

            #length check
            if len(username) > 1 and len(username) <15:

                #takes each letter in turn - checks if invalid
                for i in array:
                    if i in username:
                        print ("invalid username - must be a string.")
```

Candidate Name: Cameron Noakes

Candidate Number: 9453

```
        valid = False
        break

    if i not in username:
        print ("Valid Username.")
        valid = True
        break
    break

else:
    print ("Username must be between 1 and 15 chars.")

# if 'Play' button is pressed
elif (x) > 359 and (x) < 489 and (y) > 318 and (y) < 365:
    print ("play")
    pygame.mixer.music.fadeout(1500)

import Game_Classes

pygame.display.update()

clock = pygame.time.Clock()
clock.tick(70)

pygame.display.flip()
pygame.quit()
```

My Main Game code dump is given below:

```
import pygame
pygame.mixer.music.load('audio\CarpenterBrut.mp3')
pygame.mixer.music.play(0)
```

```
pygame.init()
screen_width = 850
screen_height = 550

screen = pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("Collision Detection")

#Init variables for movement
moveX,moveY = 0,0
x,y = 0,0

# Collision detection - takes two objects,(x,y), width and height
def detectCollisions(x1,y1,w1,h1,x2,y2,w2,h2):
    if (x2+w2>=x1>=x2 and y2+h2>=y1>=y2):
        return True

    elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1>=y2):
        return True

    elif (x2+w2>=x1>=x2 and y2+h2>=y1+h1>=y2):
        return True

    elif (x2+w2>=x1+w1>=x2 and y2+h2>=y1+h1>=y2):
        return True

    else:
        return False

# Class 1
class Player:
    def __init__(self,x,y,width,height):
        self.x = x
        self.y = y
        self.width = 15
        self.height = 15

    # Player is now the medium sprite size to match the background size
    self.i1 = pygame.image.load("sprites/Sprite2.png")
```

```
def draw(self,collision):

    if (collision == True):
        player.x -= moveX
        player.y -= moveY

    screen.blit(self.i1,(self.x,self.y))

#Class 1b:
class enemies:
    def __init__(self,x,y,width,height):
        self.x = x
        self.y = y
        self.width = 20
        self.height = 20

    # Player is now the medium sprite size to match the background size
    self.i2 = pygame.image.load("Designs/zombie1.png")

    def draw(self,collision):
        if (collision == True):
            player.x -= moveX
            player.y -= moveY

    #render object to the screen
    screen.blit(self.i2,(self.x,self.y))

#Class 3
class Missile:
    #create missile
    def __init__(self,x,y,width, height):
        super().__init__()
        self.x = x
        self.y = y
        self.width = 15
        self.height = 15
        speed = 0
        self.i3 = pygame.image.load('images/laser.png')
```

```
def draw(self,collision):

    speed +=10
    #updating position
    screen.blit(self.i3,(self.x,self.y))

bullet = Missile(5,5,moveX,moveY)
white = (255,255,255)
black = (0,0,0)
player = Player(100,50,100,100)
#crate = draw_objects(800,100,50,50)
#jeep = draw_objects(800,350,130,120)
zombie = enemies(400,300,100,100)
points = 0
perks = []
powerups = []

game_state = True # inits the game state

while game_state:

    # inits the events the user could choose
    for event in pygame.event.get():
        if (event.type==pygame.QUIT):
            game_state = False

        elif event.type == pygame.MOUSEBUTTONDOWN:
            Missile(5,5,moveX,moveY)
            print ("fire.")
            pygame.mixer.music.load('audio/fire_noise.mp3')
            pygame.mixer.music.play(0)

    # if the user uses a keydown (presses a button) it
    # will go through this elif below:

    # Arrow keys or A,S,W,D first
    elif (event.type==pygame.KEYDOWN):
```

```
if (event.key==pygame.K_LEFT or event.key== ord('a')):
    moveX = -30
    print ("player.x: ",player.x,"player.y:",player.y)

elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
    moveX = 30
    print ("player.x: ",player.x,"player.y:",player.y)

elif (event.key==pygame.K_UP or event.key== ord('w')):
    moveY = -30
    print ("player.x: ",player.x,"player.y:",player.y)

elif (event.key==pygame.K_DOWN or event.key== ord('s')):
    moveY = 30
    print ("player.x: ",player.x,"player.y:",player.y)

# Checks to see if the user has let go of the key
elif (event.type==pygame.KEYUP):
    if (event.key==pygame.K_LEFT or event.key== ord('a')):
        moveX = 0
    elif (event.key==pygame.K_RIGHT or event.key== ord('d')):
        moveX = 0
    elif (event.key==pygame.K_UP or event.key== ord('w')):
        moveY = 0
    elif (event.key==pygame.K_DOWN or event.key== ord('s')):
        moveY = 0

#loads background into memory and blits (renders) it to the screen

# "Designs/setting_smaller1.png" was the original background, changed to
setting2
bg = pygame.image.load("Designs/setting2(features).png").convert()
bg = pygame.transform.scale(bg, (screen_width, screen_height))
screen.blit(bg, (0,0))
```



```
pygame.display.update()

# update players movement
player.x += moveX
player.y += moveY

# Border Control
if player.x > 790 or player.x < 0:
    player.x -= moveX

elif player.y > 480 or player.y < 0:
    player.y -= moveY

# Test for collisions
collisions =
detectCollisions(player.x,player.y,player.width,player.height,zombie.x,zombie.y,zomb
ie.width,zombie.height)

#parse values into the 'Player' and 'draw_objects' Class
Player.draw(player,collisions)
enemies.draw(zombie,collisions)
zombie.draw(False)

#The jeep
# draw_objects.draw(jeep,collisions)
# jeep.draw(False)

#Inits the frame rate and the cycle clock
FPS = 150
clock = pygame.time.Clock()
clock.tick(FPS)

# Updates the screen while True in game_state
pygame.display.flip()
pygame.quit()
```

RESOURCES

https://www.youtube.com/watch?v=idKYGwVEaK8&list=PLzkbUH_IjWupcK_IWgCj3_snt-eK5PGmy

http://programarcadegames.com/index.php?lang=en&chapter=example_code_longer_examples

https://www.reddit.com/r/Unity2D/comments/de9wej/i_made_dualshock4_pixelart_for_ui_use_it_if_you/

<https://www.pixilart.com/art/pixel-pistol-176f681b08c642e>

<https://www.vectorstock.com/royalty-free-vectors/gun-pixel-weapon-vectors>

<https://www.hiclipart.com/free-transparent-background-png-clipart-dclxm>

<http://pixelartmaker.com/art/8685ad854868138>

<https://pythonprogramming.net/pygame-start-menu-tutorial/>

<https://lorenzod8n.wordpress.com/2007/05/30/pygame-tutorial-3-mouse-events/>

<https://www.google.com/search?client=avast&q=cocode+for+pygame+mouse+movement>

<https://www.programcreek.com/python/example/7655/pygame.MOUSEMOTION>

<http://www.poketcode.com/en/pygame/mouse/index.html>

[https://www.google.com/search?client=avast&q=pygame.mouse.get_pos\(\)](https://www.google.com/search?client=avast&q=pygame.mouse.get_pos())

<https://stackoverflow.com/questions/39626018/getting-position-of-user-click-in-pygame>

<https://stackoverflow.com/questions/23841128/pygame-how-to-check-mouse-coordinates>

https://www.pygame.org/docs/ref/mouse.html#pygame.mouse.get_pos

<https://www.google.com/search?client=avast&q=pygame+how+to+get+cursor+x+and+y+coorindates>

<https://stackoverflow.com/questions/36653519/how-do-i-get-the-size-width-x-height-of-my-pygame-window>

<https://pixelcalculator.com/index.php?lang=en&dpi1=&FS=15>

<https://stackoverflow.com/questions/20337502/pygame-collisions-with-floor-walls>

<https://inventwithpython.com/invent4thed/chapter19.html>

<https://www.youtube.com/watch?v=1aGuhUFwvXA>

<https://stackoverflow.com/questions/29640685/how-do-i-detect-collision-in-pygame>

<https://www.youtube.com/watch?v=dGwmmBBMIKs>

<https://stackoverflow.com/questions/28005641/how-to-add-a-background-image-into-pygame>

<https://nerdparadise.com/programming/pygame/part2>

http://programarcadegames.com/python_examples/show_file.php?file=game_class_example.py

<https://opensource.com/article/17/12/game-python-add-a-player>

<https://stackoverflow.com/questions/35642629/using-classes-in-pygame>

<https://nerdparadise.com/programming/pygame/part3>

<https://pythonprogramming.net/adding-sounds-music-pygame/>

<https://github.com/pygame/pygame/issues/322>

<https://pythonprogramming.net/pygame-start-menu-tutorial/>

<https://techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/projectiles/>

<https://stackoverflow.com/questions/21567250/how-to-create-bullets-in-pygame>

<https://gamedev.stackexchange.com/questions/55806/pygame-top-down-shooter-firing-bullets>

<https://stackoverflow.com/questions/23841128/pygame-how-to-check-mouse-coordinates>

https://programtalk.com/python-examples/pygame.mouse.get_pressed/

<https://stackoverflow.com/questions/23003459/python-pygame-input-controls>

<https://realpython.com/lessons/user-input/>

<https://stackoverflow.com/questions/29314987/difference-between-pygame-display-update-and-pygame-display-flip>

<https://www.pygame.org/docs/ref/display.html?highlight=flip>

<https://stackoverflow.com/questions/21257865/how-to-clear-up-screen-in-pygame>

<https://gamedev.stackexchange.com/questions/137839/clearing-screen-in-pygame-so-content-on-the-screen-is-not-lost-but-updated>

<https://sites.cs.ucsb.edu/~pconrad/cs5nm/topics/pygame/drawing/>

<https://www.youtube.com/watch?v=UdsNBIZsmII>

<https://stackoverflow.com/questions/54273814/how-do-i-collide-and-reduce-player-health-in-pygame>

<https://www.youtube.com/watch?v=FfWpgLFMI7w>

<https://techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-tutorial-movement/>

<https://kite.com/python/docs/pygame.Surface.blit>

https://www.reddit.com/r/pygame/comments/2y60lh/pygamedrawing_to_a_surface_and_then_blitting_that/

<https://pythonprogramming.net/displaying-images-pygame/>

Candidate Name: Cameron Noakes Candidate Number: 9453

<https://www.youtube.com/watch?v=IT-Mi75B6hA>

<https://stackoverflow.com/questions/20002242/how-to-scale-images-to-screen-size-in-pygame>

https://en.wikipedia.org/wiki/Hotline_Miami

<https://www.engadget.com/2015-05-08-hotline-miami-making-of-documentary.html>