**Computer Science – NEA**

**Draft**

Student: Dmitry Veshchikov – 19005199

Supervisor: Mr Abbas

1. **Analysis**
   1. **Introduction**

**Background**

I have been fascinated with programming a complex binary machine since a young age, seeing all the almost “magical” things that could be achieved with them. However, the most interesting were video games, not any regular games, but games that were competitive, engaging, intense, skill-based games that I had to grind in order to master and become good at. This is the overall goal of my NEA project.

**Game Idea – Bioforce2D**

I will be creating a fast-paced, skill-based, 2D platformer shooter game, that has multiplayer networking, arena/wave fighting genre, as well as mission-based campaign. [Ideally, I would want to also implement a level editor so players can create their own levels and play them with their friends, but I don’t know the time it would take for that, so I will not include it in this document just now] The game will take much inspiration from other legendary 2D games, such as: Super Meat Boy, Broforce and Shovel Knight.

The game will be set in a sci-fi, post-apocalyptic ruins of the great civilisations that humanity has erected, during year 2087 [my Minecraft username from about 7 years ago was Legoma87]. The player is part of an elite military squad named Bioforce, who has been sent on a 1-way mission to infiltrate and eliminate the heart of the alien invasion. 

The main objective of the game would depend on the game-mode being played. All game modes can be tackled solo, but player has the option to play with others, thanks to networking “magic”. The wave-mode would consist of brutal infinite wave challenges, set in a double screen-sized map. Every wave would get harder and have mini-boss battles. The campaign-mode would consist of a map, crawling with enemies, objectives given to complete the mission and main-boss battles. However, probably the most fun personally for me will be arena-mode, where players will show off their acquired skills, and try rip apart, explode and ionise each other into fundamental particles found in the standard model. <https://www.youtube.com/watch?v=NNPAl84I1rY>

[Link game teaser of an old version, 5.5.20]

The “rules” & difficulty of modes will be set before play begins, during prep-phase (transition period between menu and game-time). Difficulty setting is a basic setting, that changes number of enemies, their health, damage dealt to player. However, “rules” is a more interesting mechanic where player will have the ability to limit what they can use throughout the game-time, e.g. only melee weapons. Making each game feel slightly different and more exciting, (adding large hours of replay-ability without adding more maps, missions, etc…)

**Project Scope**

I will be using a popular, pre-made game engine – Unity, to create the game. A major question that I need to ask whilst making this project in Unity is; “should I implement this tool on my own, or use the tool set already created by Unity team themselves?” It would certainly be a lot easier to implement complex features like, networking, using Unity’s own networking systems. However, in order to show my skill and nerves, I will implement features like networking, mainly through CSharp networking libraries. (This will greatly increase the time spent implementing these features)

**Timeline**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task ID** | **Task Name** | **Time for completion (hours)** | **Date Completed** |
| 1.1 | Analysis – Introduction | 3 | x |
| 1.2 | Analysis – Investigation | 4 | x |
| 1.3 | Analysis – Prototyping | 20 | x |
| 1.4 | Analysis – Constraints | 1 | x |
| 1.5 | Analysis – Limitations | 0.25 | x |
| 1.6 | Analysis – Objectives | 2 | x |
| 1.7 | Analysis – Solutions | 1 | x |
| ∑ 1 |  | 31.25 |  |

**Target Audience**

The main audience of Bioforce 2D will be teenagers or adults (16-34) [Realistically 14] who are bored or have free time on their hands, and have access to a smartphone, with or without an internet connection. The average player will be an above average shooter gamer, who likes to show off their skills to others. Casual gamers will be welcomed, however might have trouble with the multiplayer arena-modes, but there is still a ton of variety and gameplay without it.

The game will be available on smartphone devices, so a touch-interface will not be so alien to the players who are picking up the game. However, I still will include an optional tutorial level for less experienced players.

**NEA Supervisor**

The supervisor for this NEA project is Mr Abbas. With whom I will be in contact with, regarding progress, advice and reality-checks.

* 1. **Investigation**

**Game Engine**

I have been fascinated with building complex systems from the ground up and took up making a window based Connect 4 game with adjustable board size, counters in a row to win as well as player number, using GTKSharp. However, building my own game engine for making of Bioforce 2D would be incredibly difficult, and would be a NEA project on its, so to focus my energy and nerves I will using a pre-made game engine.

1. **Unity**

Unity is an object oriented, game-engine that utilises higher-level CSharp as its programming language. I have been programming in CSharp since the beginning of year 12 (can’t stand interpreted languages like python… For some reason it is easier for me use something where I have to know how most of the pieces fit, rather than having it done in the background for me.), so the programming Unity environment would be natural for me. Furthermore, there is an ocean’s worth of documentation, tutorials and lessons on how to utilise Unity. (especially now that the Unity’s own lessons became free for everyone’s use, used to have to pay for them. During pandemic they made it free) Unity is a gameObject driven engine, which is very similar container/organising system, that is extremely intuitive to object-oriented programmers. The higher-level CSharp language allows for quick testing, optimisation and debugging, due to its light-speed compilation time, in comparison to the lower-level game-engines like Unreal Engine. Therefore, a great candidate as an engine.

1. **Unreal Engine**

Unreal Engine is also an object oriented, game-engine. However, it utilises a lower-level programming language C++. C++ is similar to CSharp, however I have not programmed in it, so therefore it would be a greater hurdle to learn a new programming language and make Bioforce 2D at the same time. In addition, C++ lower-level nature is more difficult to manage and comprehend. However, it has the ability to be optimised more than Unity, making it a good game-engine for big computer taxing games, which Bioforce 2D won’t be. The slow compilation times make it harder to test and optimise the game. Therefore, a more challenging candidate.

1. **Godot**

Godot is an open-source, smaller sized game-engine that utilises its own scripting language GDScript. However, it supports other languages: CSharp, C++. The open-source nature is good for veteran game/software developers that need the freedom to optimise and change things for their visions. I, on the other hand, not a veteran developer, therefore, don’t require that freedom, also Godot requires other software and tools to be installed and learnt to use, in order to compile for mobile devices (not as easy as Unity/Unreal Engine). The GDScript language used by Godot is similar to Lua and Python, which I have not to mastered as much as CSharp, therefore would require even more time to adapt to. Therefore, a good candidate if I have been making games for a longer period of time.

**Chosen – Unity**

**Existing Games**

1. **Broforce**

Broforce is the most patriotic, action filled, heavy metal game that screams America. Bringing all the action and experience we feel at the cinema at every single action movie. Making you experience something that is very familiar, but improved and unique, allowing you not only to be along for the ride, watching the action, but driving the ride, playing and determining the action.

Broforce further brings familiarity from the best, well known action movies through its wide variety of playable characters, making you experience being that character. Some of my favourite are Rambro with his M60 machine gun (inspired from Rambo), Bronan the Brobarian wielding the Atlantean sword (inspired from Conan the Barbarian) and Bro Lee using his insane martial art skills (inspired from Bruce Lee).

This clever character design allows you to pick the best characters of all time and make it more familiar to the player, giving a better player experience with a wide variety of characters. This mixed in with the mechanic of switching the player character every time they die, gives the player a more interesting and an engaging experience for a prolonged amount of time.

Broforce shines on its own as a single-player game, however everyone knows that multi-player adds excitement, challenge and experience, due to the ability to share your fun experience with others. This adds a layer of replay-ability and keeps the player engaged, because they sense a feeling of being in a community and working on a common goal, driving the player to be better and continue playing.



Last major lesson from Broforce is knowing that the player will finish the whole game you made and get bored of playing the same maps over and over again, no matter the number of levels and maps. Broforce tackled this by adding a level editor easily accessed in the game, that allows the player to have a creative outlet, building their own levels and maps, further adding enjoyment and hours played. More games should allow this and promote this behaviour, in order to make the game fun months after completing the main campaign and exploring every nook and cranny of the game.

Broforce uses pay-to-play to generate revenue, at a price of £11.99 (price on steam in the UK market). However, Broforce is available on Windows, MacOS, Linux, Nintendo Switch and PlayStation4. Missing the huge market of mobile gaming, which is growing every day.

1. **Shovel knight**

Shovel knight brings back memories and fires up nostalgia from simpler times, bringing back the pixelated graphics, the arcade like sound effects as well as the classic dialog boxes used to communicate with the player. This makes the player feel the nostalgia and fill the wide, deep hole of memories that the player has been missing from the "good-old-days".

In addition, this game’s core player mechanics are filled with inspiration from other old school games, bringing back more nostalgia than ever. The blue armour and rivalry between a similar antagonist character, reminds you of Mega man on the NES. The jumping and falling attacks are clearly inspired by ducktales on the NES. The level picking is designed so you pick levels from the map, with a simple map where you traverse to find obstacles and level icons, which is a popular design choice in Super Mario Bros.

The game was inspired, and picked mechanics from the best games of earlier, nostalgic times, creating a nostalgia factory. The mechanics are not simply copies, but improvements, and the creator’s own spin offs. Making the player relive those 8-bit moments, whilst keeping an authentic and distinct line between the shovel knight and the games it was inspired from.



Also shovel knight fixes some of the frustrating, death mechanics of the older games, in which if you were to die, you would have to start all the way from the beginning. Instead you lose some of your gold where you died, and restart either from the main spawn point or checkpoint, similar to the Dark Souls series. Shovel knight is a modern game without the physical and software restrictions of the past, so it takes inspiration from newer games as well. Bringing the best of both worlds, the best of the past and the best of the new.

The key lessons from this game is to be inspired from the best games that are already around, rather than reinventing the wheel on your own. Whilst also bringing some new mechanics and improvements from the games that inspired you, in order to make the game be authentic and distinct from them.

Shovel knight made revenue from pay-to-play, where the player pays an initial fee of £30.99 (price on steam in the UK market). However, no more revenue is made after the game is bought. This hefty price tag may deter many players, because some players may see shovel knight as an old school game that isn’t worth the money. Also, it is available on most platforms, Desktop, Consoles, (even amazon fire TV), but not on smartphones. This greatly decreases the availability and outreach to mobile gamers that would love this game. Therefore, shovel knight loses a lot of possible profits from the smartphone market.

**Key notes**

Bioforce will take lessons from pros of these games: Making the game feel familiar, yet unique in its own way to drive nostalgia. Not being afraid to take inspiration from popular games. Multi-player is a must, because it adds competition, excitement and fun. Allow the community to create their own decisions of how the want to play the game, will use the “rules” mechanic to achieve this. (perhaps a level editor)

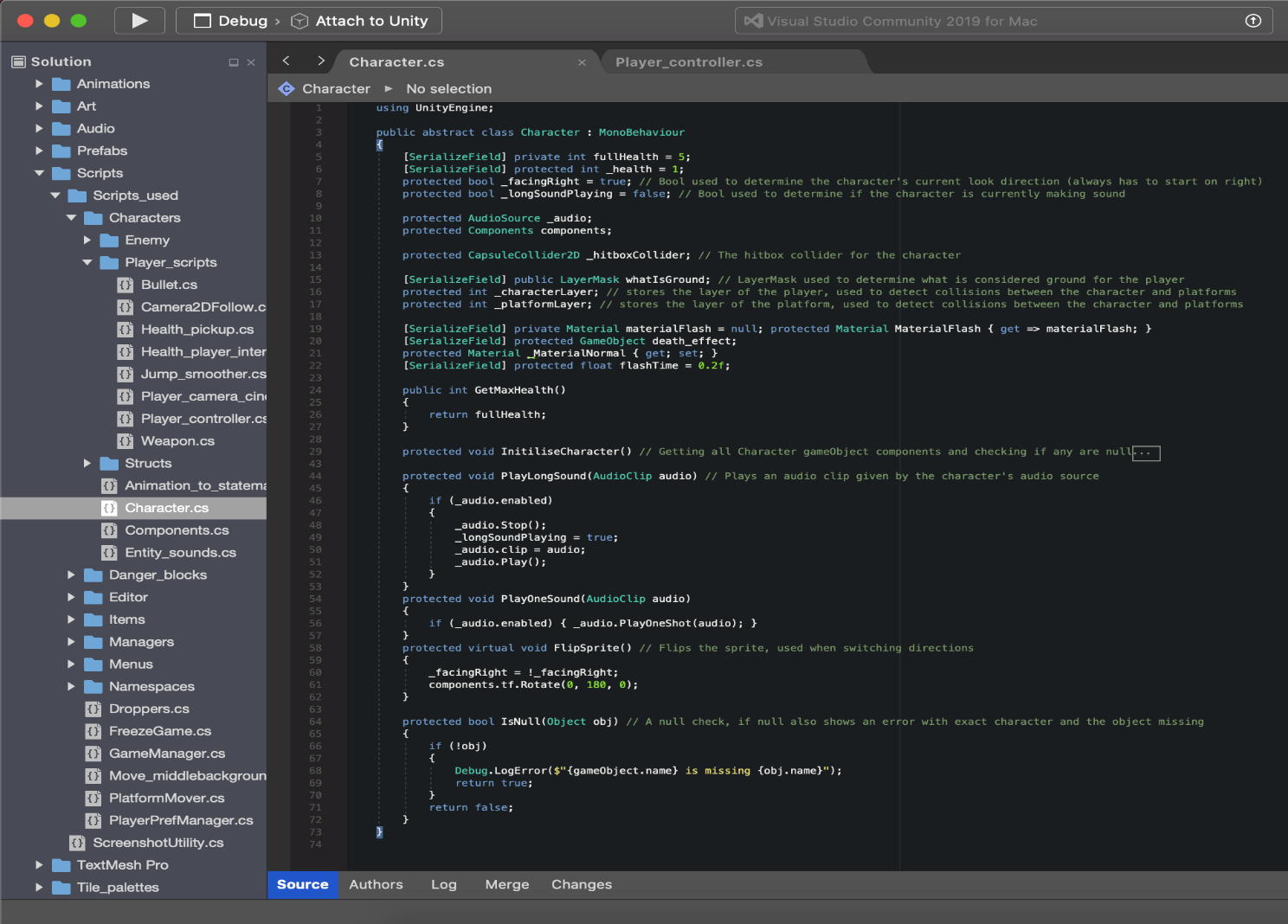
**Surveys of features – Need to perform these, ask Mr Abbas how to do the ones that I replied to last year**

* 1. **Prototyping**

<https://github.com/DmitryVesh/Bioforce-2D-unity>

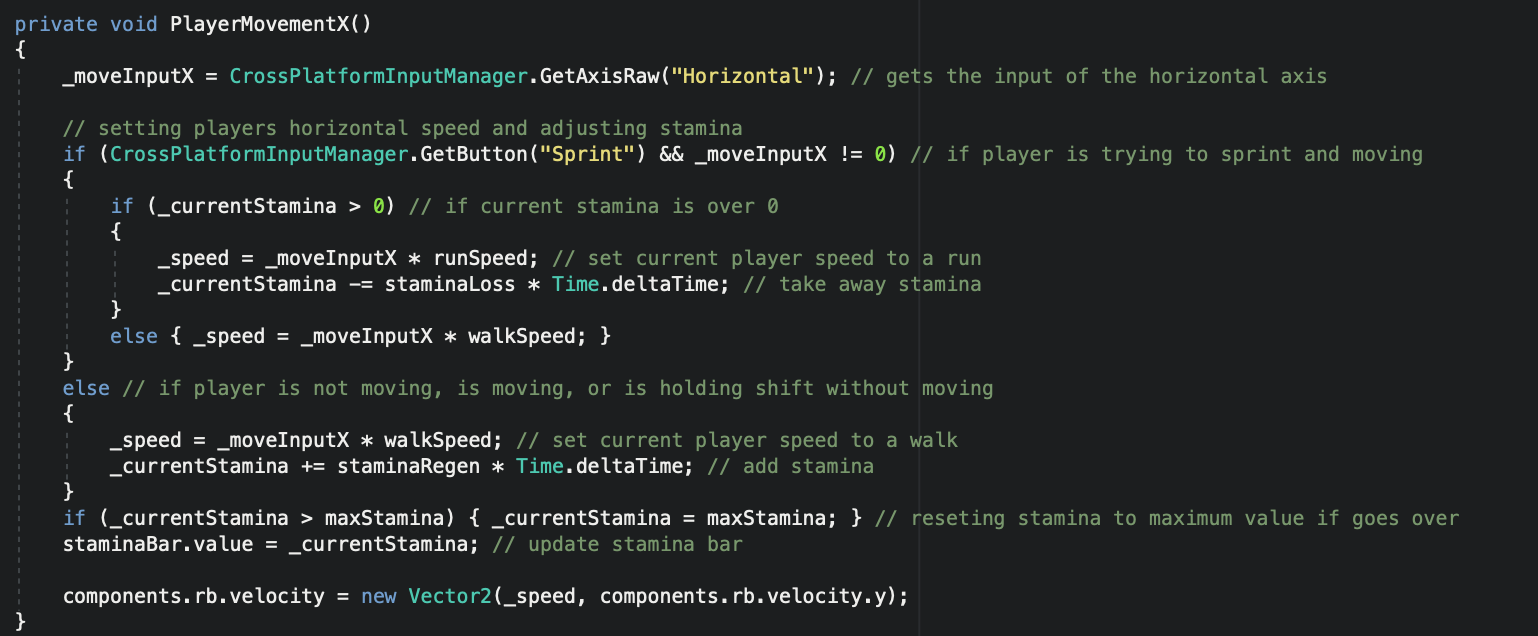
**Player Scripts**

The player script that allows the user to control the player inherits from a “Character” script, which holds methods and data found in both players and enemies. Holds the integer value for max and current health, as well as which way the sprite is facing.

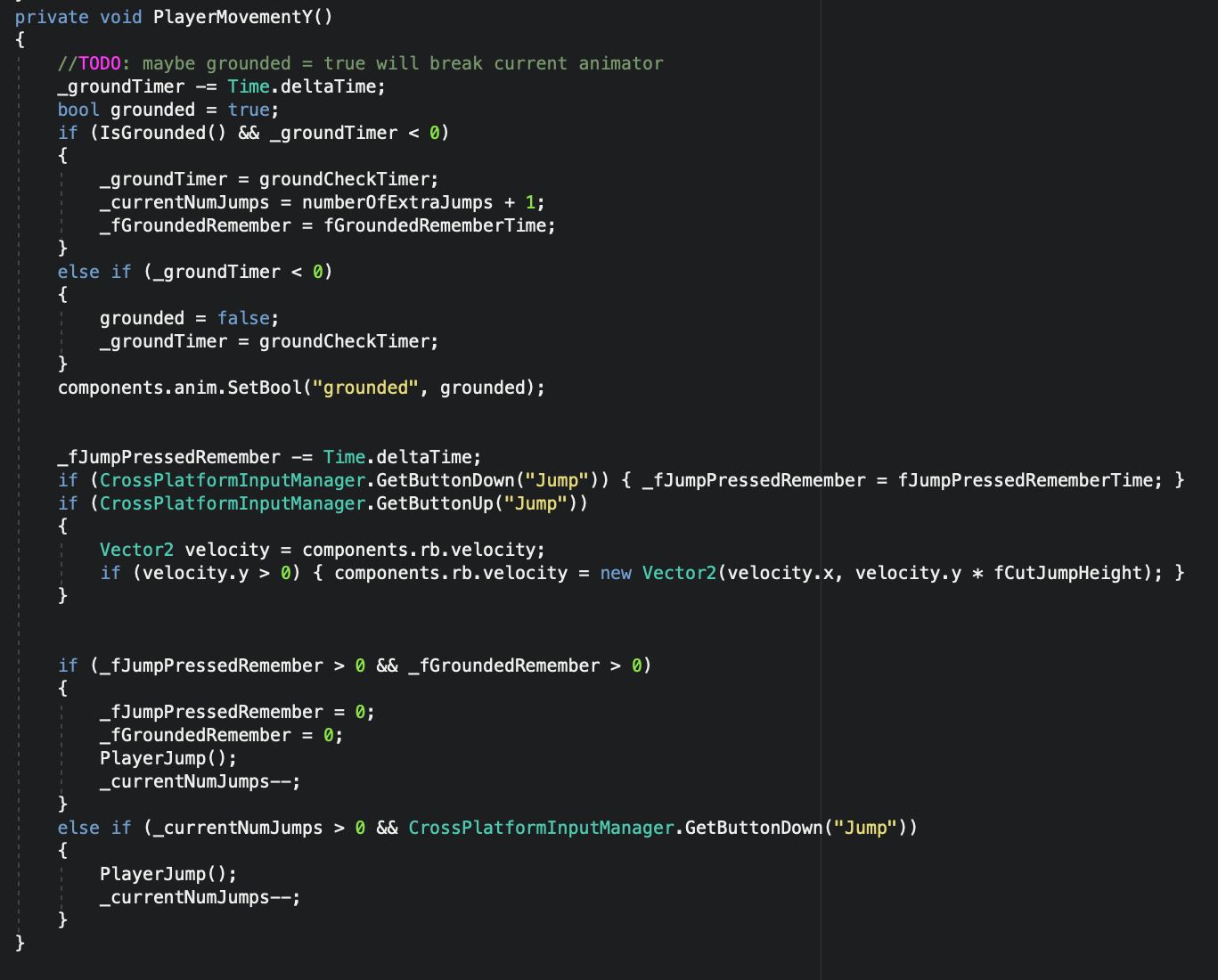


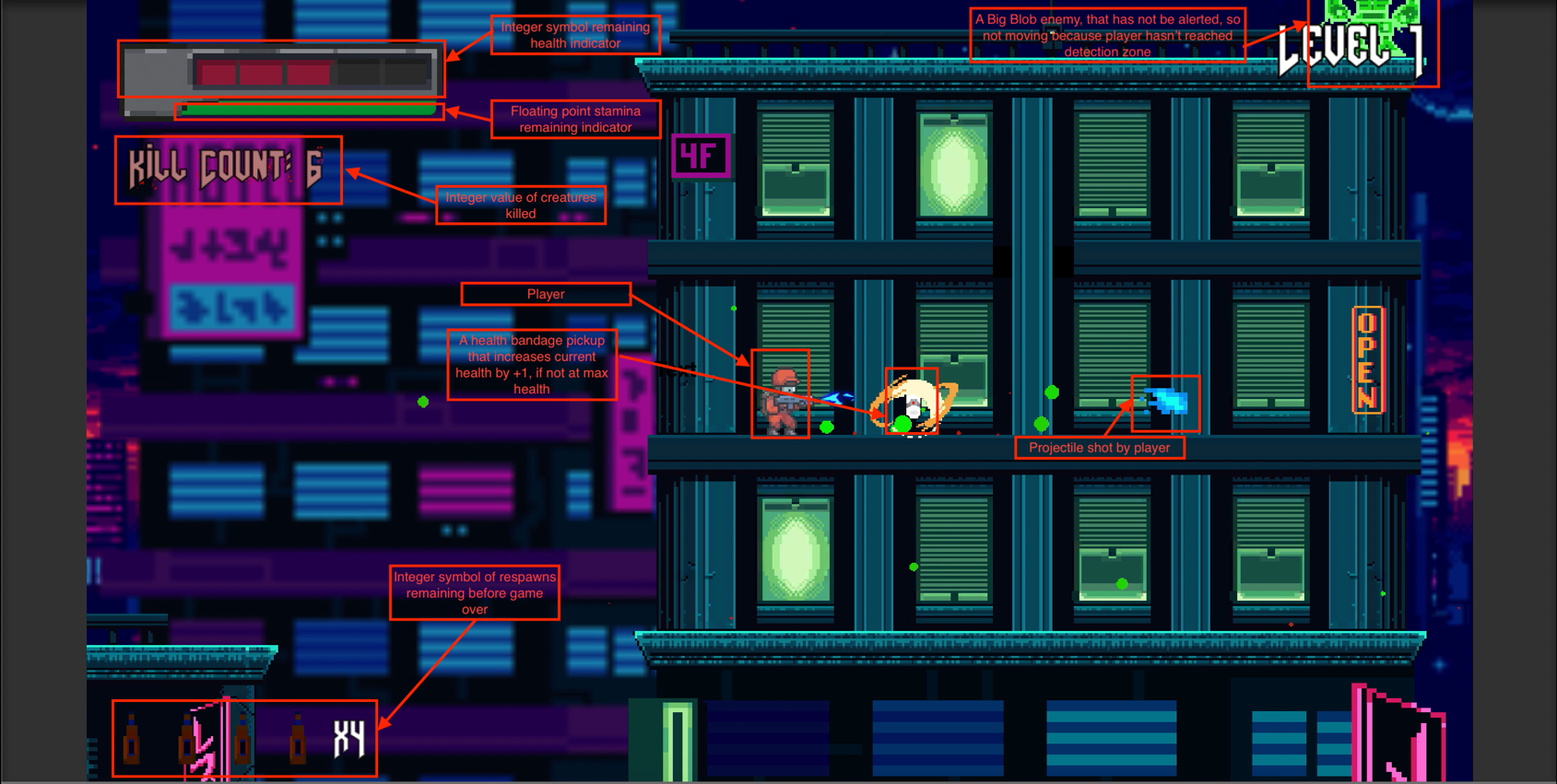
The player script itself has code for controlling movement along both directions, x and y.

PlayerMovementX code. \_moveInputX represents the value of input returned from keys/sticks/touch. The player also has ability to sprint for a short burst of time, using up stamina in the process.



PlayerMovementY code. Checks if player is able to jump/ double jump, using a ground check and integer number value to store number of jumps remaining and replenish them after touching the ground.





* 1. **Constraints**

**Hardware**

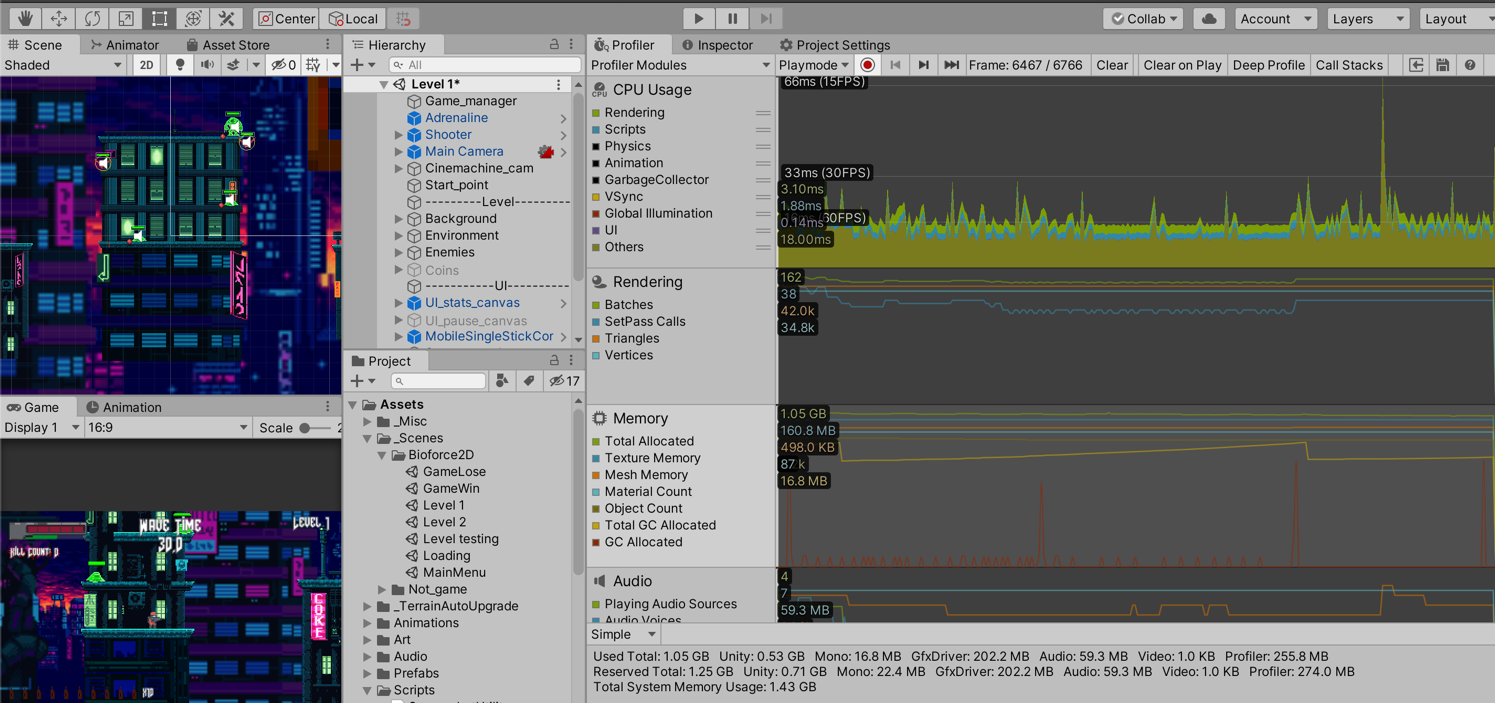
The mobile nature of the game would reduce the available processing power of more powerful machines like PC’s and Laptops, so therefore the graphical details and number of effects will have to be reduced in order to run the game smoothly.

The input controls would consist of touch-screen display integrated within the device, so control sticks, fire button, aim button and shift switch must be carefully arranged in order to provide best user experience. However, I will be adding a feature of customisation of the control layout, so players are able to adapt their needs and varied screen sizes. All pause buttons and menus will be accessed via touch, and very intuitive on mobile devices.

The file size of a windows build is >200 MB, which therefore suggests to me that the full project will be around 300MB – 400MB, when fully optimised.

**Software**

In prototype, unoptimized version of Bioforce 2D is using total of 1.05GB of RAM, whilst in editor mode, so not in the build version. Therefore, an optimised, compiled build version will not be using as much as 1GB, however assuming it will equal to 1GB, smartphones come with well over 1GB of RAM (1-2GB of RAM in smartphones in 2017), which would be enough to run the game smoothly, upon optimisation. (Attached article)



<https://mobiforge.com/news-comment/what-are-the-most-common-ram-specs-for-smartphones>

* 1. **Limitations**

A further great feature that adds an unlimited amount of replay-ability is the ability to generate maps randomly, so the player faces uncharted territories and mission objectives differently every time. However, due to need for creation of other features, I won’t be implementing this feature.

Another great in game feature would be a fully customisable weapon of choice for every player. However, this feature would require a lot of work in designing and artistic time, that it would prove very challenging.

* 1. **Objectives**

**General Objectives**

1. Produce a fast-paced, skill-based, 2D platformer shooter
2. Implement unique, yet familiar/nostalgic 2D platformer mechanics
3. Create a variety of maps for to be played on
4. Implement a playable character
5. Implement enemies that attack and kill the player
6. Implement a multi-player mode for campaign-mode
7. Implement wave-mode (endless wave mode on a single map)
8. Implement arena-mode (multi-player deathmatch)

**Specific Objectives**

1. Produce a fast-paced, skill-based, 2D platformer shooter
2. Selected Unity game-engine
3. Players play solo or with others to kill AI enemies
4. Players are pushed towards being more skilled
5. Implement unique, yet familiar/nostalgic 2D platformer mechanics
6. Select mechanics from previous 2D platformer games that have done well and re-implement them in my own style
7. Think and mind map ideas for “rule” mechanics
8. Select and implement art assets that are older styled, 16/8bit style
9. Select and implement music assets that fit into the art style of the game
10. Create a variety of maps for to be played on
11. Group complementary art and music assets into maps
12. Using grouped assets, make a short platformer map
13. Test maps to see if there are any bugs or possible improvements possible
14. Implement a playable character
15. Using character assets, choose a player character
16. Add movement, attack and interactive behaviours to the character
17. Animate the character
18. Implement power-ups and items that can be picked up
19. Implement enemies that attack and kill the player
20. Using character assets, choose a few enemies and boss enemies
21. Add AI behaviour to the enemies, sleep/wait, alerted, move, attack, seek
22. Animate the enemies
23. Implement a multi-player mode for campaign-mode
24. Using CSharp Networking libraries, implement a server that runs on a player’s machine
25. Implement that players are able to join the server
26. Implement server updates, that players are able to download from /upload to
27. Limit/control the possible moves a player can perform
28. Conjoin 2 players to be on the same team and be able to play to complete a map
29. Add a local scoreboard that shows each player’s status: health, lives, kills…
30. Implement wave-mode (endless wave mode on a single map)
31. Select few unique places in maps, that can be used as a map for wave-mode gameplay
32. Implement spawners of enemies
33. Implement waves
34. Implement rising difficulty of waves
35. Using previously multi-player networking, implement multi-player
36. Implement arena-mode (multi-player deathmatch)
37. Select few unique places in maps, that can be used as a map for arena-mode gameplay
38. Implement enemy AI that controls playable characters
39. Using multi-player networking, implement multi-player
40. Implement that players are able to damage each other
41. Implement teams that players are able to join
42. Implement a timer that will end the session
43. Re-adjust the local scoreboard used in multi-player campaign to work as a leader-board in arena-mode
    1. **Solutions**

I will be using CSharp programming language, due to its vast capabilities in game making, networking used for multiplayer, as well as due to the enormous amount of documentation and support it has. I will be using Visual Studio as a code editor, due to its convenient debugging analyser, syntax correction and code auto-completions.

I will be using Unity game engine, due to its developer friendly environment, as well as the documentation available. Unity will allow me to focus more on the complex programming implementations of the game mechanics, integration with networking for multi-player.