Analysis

Project definition

# The Pitch

I am creating a 3D first person multiplayer arena shooter. It will resemble the classic PC shooter games such as the original Unreal Tournament, but with the added twist of more modernized graphics, user interfaces and multiplayer features. The game will include a simple roster of arenas, which can each be selected and used to host several game types. The player will be able to create a multiplayer lobby, and other players will be able to join their game through either the internet or through the host’s local network. The host will be able to set the game type from a list of several. This would include team-based deathmatch, where the aim is for one team to hit a target number of eliminations before the other. Additionally, there will be a ‘capture the flag’ mode, where the aim is for one team to capture the opponent team’s flag from their base and return it to their own, winning the round. There will also be an all-out deathmatch, where every player in the match is competing against one another to rack up the highest number of eliminations within the time limit. A modern addition I will introduce is a ‘battle royale’ style mode, in which each player only has one life, and the winner is the last player alive in an all vs all deathmatch.

Each gamemode will feature a variety of weapons fulfilling different combat archetypes such as a sniper, an assault rifle, a shotgun, and several other special purpose weapons. The weapons will spawn randomly in different spots around the arenas. However, some arenas will have locations that spawn specific weapons every time, making them highly contested positions. As well as weapons, there will be item pickups spawning around the arenas. This will include stat-boosting items to make the player more powerful, and additional health / ammunition to aid the player. Some arenas will feature dynamic objects such as teleporters, elevators and moving platforms in order to make the gameplay more interesting.

# Audience & Target market

The game will have two distinct groups of appeal: older players seeking an experience similar to the games they played in their youth, and curious younger players looking for a multiplayer experience that is different to what the ‘AAA’ game studios are currently offering. This will land the project in a large age bracket of approximately ages 15-40. While the game may have appeal outside of this group, this will be my focus going forward. The simple arena shooter genre is very sparsely populated in the current videogame market, which is why I think that this project solves the issue of that lack in game variety. In order to maximise market appeal, and additionally to keep the game populated with players, it will be completely free. This will include all of the game modes and content, allowing for anyone with a computer and some free drive space to download and play with their friends. This will make the game very appealing to groups of friends looking to pick up a quick and entertaining multiplayer game that everyone will be able to join in on. The aforementioned Unreal Tournament will be my main study into the genre and, having set most of the groundwork all those years ago, I will be looking to build on the experiences offered by it.

# Stakeholders

In this instance, the stakeholder would be any player interested in multiplayer first-person experience, which is a very wide demographic. However, as I have mentioned previously, my main target will be Unreal Tournament / Quake veterans as well as younger players seeking a new experience.

[insert stakeholder 1 here]

[insert stakeholder 2 here]

# Control scheme

The game will feature an array of re-assignable keyboard and mouse inputs to allow for precise and accurate control of the player character. Many of these inputs will be based upon the standard set of video game controls shared between most first-person shooters and third-person shooters. This is to maximize playability and to make the game feel familiar to those not yet acquainted to it. Player movement will be defaulted to the keys ‘W, A, S, D’ as well as the arrow keys as an alternative. This is the standard method of movement seen in games. The player’s camera will be aimed with the mouse’s X and Y axial movement. This allows for precise aiming and natural control of the camera, as well as being a standard. The left mouse button will be mapped to fire the player’s current weapon, in addition to the right mouse button – which will be assigned to the weapon’s alternative fire mode. The scroll wheel will be the primary method of switching between weapons, alongside the option to use the number row of the keyboard for more precise switching. Weapons are reloaded manually with the ‘R’ key. Jumping and climbing ladders will be assigned to the space bar, as is commonly seen in games. Sliding and crouching will be mapped to ‘CTRL’ and, additionally, the ‘E’ key will be used to interact with items as it is easy to reach from the left hand. Some additional controls will include accessing the scoreboard and the text chats. These will be defaulted to ‘Q’ and ‘T’ respectively. The in-game options menu will be accessible with ‘ESC’ and will be the only non-bindable control. This is to avoid the player becoming stuck and unable to change their controls if they accidentally unbind it, as the control remapping menu is accessed through this interface.

Solving the problem computationally

# Thinking abstractly

The movement, weapons and environments featured in-game will be heavily abstracted from reality. This is because realism is not a requirement nor a focus, and computational resources such as CPU and GPU time as well as memory can be better spent than on mirroring the real world.

The following aspects of the game are examples of abstraction:

* Movement: Movement in the game is abstracted down to simple directional inputs and jumping. The game does not require precise emulation of walking, sitting, crouching, lying down and the like. Only movement functions necessary for gameplay will be programmed in.
* Visuals: While the game will have a high level of visual fidelity in lighting and animation, the surface textures and user interfaces feature a retro aesthetic that is a far cry from realism. The very nature of pixilation is a form of abstraction, as unnecessary resolution is disregarded in favour of more performant low resolution generalisation.
* Environments: The environments in the game will be stripped down to the bare essential architecture and props needed to create a believable scene. Abstractions in this regard include clouds being baked into the skybox texture, and billboard sprites being used to represent vegetation and other insignificant details.
* Weaponry: In the heat of the action, players do not want to worry about having to reach down into their suits to find ammo cartridges, nor do they expect to have to operate the weapons realistically. The weapons in the game will be reloaded at the click of a button and will fire based on mouse input. This is far from reality yet is all that is necessary to create fun and mechanically diverse weapon systems in the game.

# Thinking ahead

Before I begin working on the project, I have to identify all of the inputs and outputs necessary to complete the game. This is the process of thinking ahead, and it is vital for a project of this level of complexity. Inputs will include all of the ways that the player interacts with the game. This would include the keyboard and mouse-based functions. The raw inputs from these devices will be all of the mouse buttons and keyboard keys, as well as the scroll wheel and mouse X and Y movement. Additionally, I must consider the outputs of the project. This will include how the game is displayed on the screen, as well as the positions of network-connected players and how their actions affect the player (such as dealing damage or rendering on the screen). Additionally, the scoreboard, in-game text chat and match results could also be considered outputs, so they will be planned accordingly.

# Thinking procedurally & decomposing the problems

In order to make my project more manageable, I will break the project down into the separate systems necessary for the game to function. Generally, the code will be separated into script files which each dictate an aspect of the game’s logic or dictate the activation of other scripts. This process of decomposing the problem is called thinking procedurally.

Below are some of the systems I will break the project down into:

* Player: Each player gameobject have a player script that will store its health damage multiplier, location, current weapon and username. The health value will then be checked by the server against other player’s damage outputs to prevent players from cheating and becoming invincible by modifying their health value.
* Weapons: Each weapon will have its own respective script that is called on by the master weapon script when a weapon is selected or picked up from a pickup location. This master script will instantiate the weapon gameobject, which in turn contains the weapon script that controls functions of the weapon such as its fire modes, particle effects, reloading animation and sound effects. Each individual weapon script will also contain a reference to the weapon stats database.
* Weapon stats database: This database will contain the statistics for each weapon, including the damage amount, reloading time, fire rate, spray amount and magazine capacity. This allows for values to be adjusted without having to go delving into the scripts for each weapon individually.
* Items: Some special items like damage boosters, health boosters and health replenishers will contain scripts that modify the players’ health and damage output upon activation. These will be controlled by scripts attached to the items when they are spawned within the maps.
* Networking: This will be broken down into several sub-systems such as a network manager, matchmaker, movement & damage networking scripts, scoreboard system and the text chat.
* Inputs: The game will have an input manager system which maps the desired inputs to player actions, as well as controlling player motion, aiming and player physics.
* Game modes: As the game will have multiple playable gamemodes, there will be a master game mode script, which will control which game mode script is loaded, as well as instantiating the selected map for it to be played on.
* Environment systems: Each map will contain a variety of spawn locations for players and items. These locations, like the weapon stats, will be loaded from a database detailing which spawn locations are on each map and their coordinates, as well as item spawn rates. This allows for manual tweaking of each map’s spawn locations and rates without having to open up the level in the unity editor or hardcoding them.
* Special systems: Some maps will contain special items such as opening doors and moving platforms. These will each have their own attached behaviour script, which will be loaded alongside their associated maps.
* Interfaces: The game’s user interface will contain many subsystems for different menus. This will include the main menu, gamemode selection menu, matchmaking lobby, options menu and pause menu, heads-up-display, scoreboard and chat visuals. The last

# Thinking Logically

[To-do]

# Thinking Concurrently

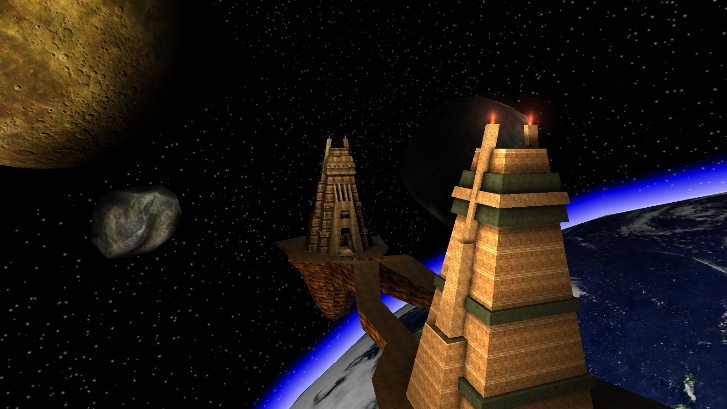
As a complex game, there will be many aspects of it that have to be implemented concurrently. Networking will have to run concurrently with movement / player and weapon systems, which will have to also run concurrently with the user interface and heads-up-display. The use of the Unity game engine will simplify this somewhat, as it easily allows for the execution of separate connected scripts concurrently. These systems will still have to be constructed in a way that allows for continual flow of data between them, as the real-time nature of the game will require every system to represent the most up-to-date values. This is especially important considering the delay that will already be introduced by the network.

Research

# Existing solution: Unreal Tournament 99

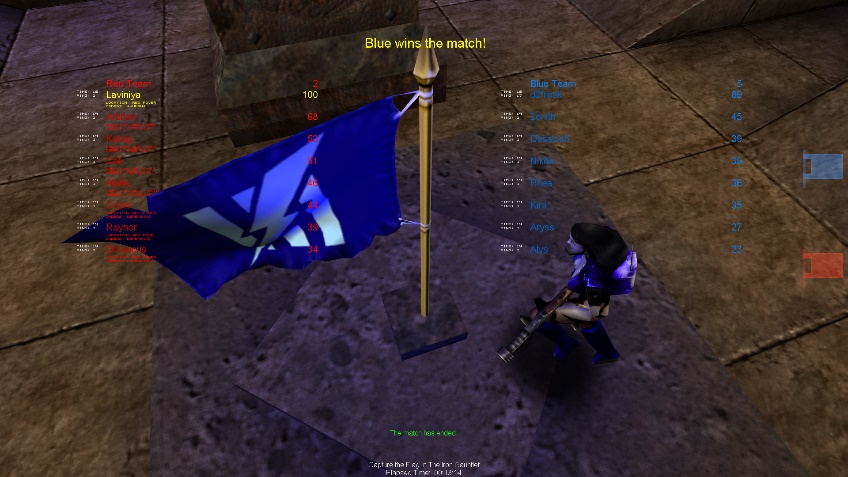
Unreal Tournament (or UT99) will be my prime research focus. As a pioneer in both graphics and first-person multiplayer shooters, it will be the main inspiration for my gameplay and style. UT99 features a wide array of varied and fun weapons, as well as a variety of maps and gamemodes to choose from. I will take cues from UT99 in terms of control style, weapon variety, and the simple yet effective gameplay formula it is famous for.

The maps: The game’s maps sport sprawling and detailed architecture (for the time) that wowed many with visual fidelity and complex gameplay possibility. Each map has a unique theme and layout, which creates unique gameplay on every map.

Some maps are iconic in their design. An example of this is ‘Facing Worlds’, which features two towers on floating platforms, separated by two mirrored bridges. The key takeaway from this design is complexity in simplicity. The fact that the same maps are played over and over allows for skilful players to use their knowledge of the map to outplay new players. This has the effect on new players of inspiring them to learn the map to improve.

The weapons: Each weapon in UT99 has a unique function, with an alternative fire mode to add some complexity. Some weapons can perform combination moves. For instance, the Shock Rifle (top left) can shoot a thin streak of energy that instantly damages anything it contacts. On the alternate fire, it shoots a ball of lightning that explodes, dealing damage. After shooting the alternate fire, you can shoot the ball of lightning with the primary fire to create a much larger explosion. However, this takes a lot of skill to do correctly, lending itself to the more competitive side of the game.



The gamemodes: The gamemodes offer variety and replay value to UT. Some of the gamemodes are responsible for the main appeal of the game as a whole, such as the exceedingly popular CTF (Capture the Flag) and Deathmatch. Deathmatch is a simple game of eliminating the enemy players until you reach a target number of kills. CTF involves two teams, each with a mirrored base at either side of the map, battling to obtain the flag from their opponents’ base and run it back to their own. Whichever team manages to capture the most flags within the time constraints is the winner.

The gamemodes are one of the strongest aspects of UT99, so it will be essential to include at least these two classic gamemodes in order to meet the appeal of my target audience.

The User Interface: While the UT99’s primitive 3D graphics may be telling of it’s age, a more noticeable indicator of this is the outdated graphical user interface (GUI). This is an aspect of the UT experience that I am looking to improve upon for my project. As it stands, UT99’s menus are clunky, confusing and have tiny buttons due to it being designed for older, lower resolution monitors. As such, I will aim to create a flexible and intuitive user interface that not only scales well with resolution but also conveys in-game information in a concise manner. However, an aspect of the UT99 GUI that is done well is the distilled, content-focused design. While the endless cascades of windows are not the easiest to navigate nor the prettiest, they clearly prioritize function over form. This is not often seen in modern games, which feature flashy and gimmicky GUIs that are usually designed more like marketing pages then functional interfaces. I will be following the UT99 design philosophy when it comes to the GUI, yet with a modernized layout and feature set.

# Existing solution: Counter-Strike: Global Offensive

Counter-Strike: Global Offensive (or CSGO) Is an interesting research case. Despite being a fairly recent release in the online FPS genre, it foregoes many of the now commonplace features of the genre. ‘Health packs’, regenerating health, ‘emotes’ and scores plastered over the screen are all but absent in this game. CSGO brings modern features to the genre, but with a refreshingly classic execution. It is not the game that I will be designing around, as it draws a slightly different type of audience then Unreal Tournament. However, it is an example of a modern GUI done right.

A recent addition to CSGO, the panorama UI, is an example of clean, modern design that is both simple and functional (above) (below).

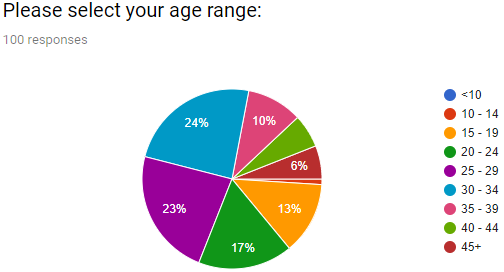


The shortcomings: CSGO, despite its classic gameplay elements, still has some features that are, in my case, unnecessary and would detract from the experience I am aiming to create. For example, CSGO’s weapon roster is unnecessarily bloated. For example, there are 7 different pistols available to the counter-terrorist team. This works fine in CSGO. However, in a simpler and more action-oriented shooter, a reduced weapon roster is desirable. CSGO also features a far less abstract set of weapons than UT, with each one mirroring a real-life equivalent in form and function. My project will instead have more archetypal weapons, such as a ‘sniper’ and a ‘shotgun’.

# Player research: UT99

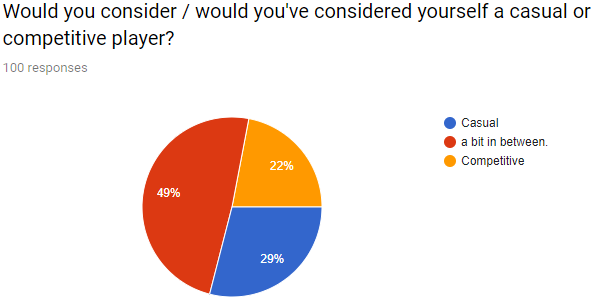
In order to make my game most suited to my audience, I decided to gather data by conducting an online survey. I shared the survey in a range of UT99-related platforms, to get a good representation of the wider age range. I shared the survey on ut99.org, which is a forum maintained by mostly veteran UT players. Additionally, I posted the survey in several Unreal or Unreal Tournament-centric Discord servers. In these servers, I would have a more likely chance of obtaining information from the younger age ranges, which is the prevalent demographic on the Discord platform. For a middle-grounds, I posted the survey in the Unreal Tournament subreddit, which is a community with a very wide age range, as well as a broader reach than some of the other sites. In this survey, I analysed which aspects of UT99 appealed to the players, as well as their age and skill level in the game. Additionally, I left several open-ended questions inquiring for more detailed written answers in order to gain a detailed insight into exactly what drew players in. This data will give me an idea of how exactly to best serve each category of my audience, and which aspects I should prioritize for my success criteria. The survey itself consists of 11 questions. Below is an in-depth analysis of each one, as well as the effects on my design direction they will have.

### Age range:



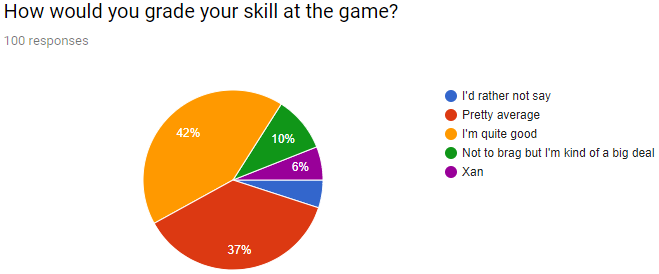
I collected the participants’ approximate ages in order to validate my target age demographic and adjust it if necessary for my specification. As expected, there are nearly no participants in the lower two age brackets. Clearly, games of this type have no appeal to children or young teenagers. That can probably be attributed to the game’s age mostly. However, that would not justify the overwhelmingly large band of 15-19s. Therefore, there must be an appeal to this game in the modern gaming landscape. As I expected. The majority of results cover the teenager/young adult and ‘90s vet’ demographics. These results will probably not pose any significant alterations to my specification, as it appears that the original target audience will be a suitable base.

### Casual or Competitive?



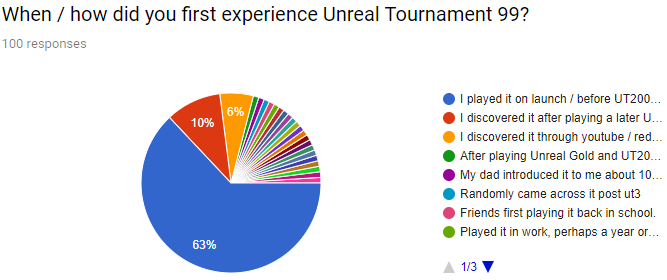
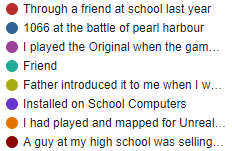
The results of this question surprised me somewhat. I would’ve predicted that the competitive category would outweigh the casual category by a significant amount. Evidently, there is far more to the enjoyment of this game than the high-skill competitive action. Most of the responses are in the central category, which suggests that many players prefer a mix of casual and competitive gameplay. The takeaway here is that even the more competitive side of the community still look for fun in their gameplay alongside competition.

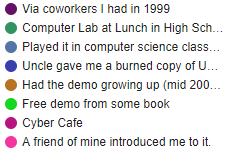
### Skill:



The main reason for inquiring participants’ perceived skill levels is to compare it to their respective age ranges and whether they consider themselves a casual or competitive player.

### Initial impressions:





A surprising majority of the participants had played the game when it released almost 20 years ago. This shows that the game needs to appeal mostly to veterans of the original, as they clearly have the most interest.

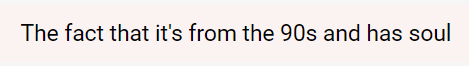
### Favourite overall aspects:

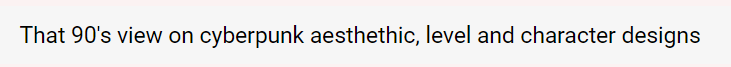
I asked people what their favourite overall aspects of UT99 are. They are into the 5 general aspects of the game that each comment represents, as there are far too many responses to analyse individually.

1. The weapons.



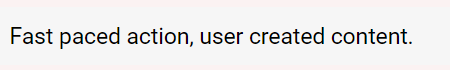
Many people expressed their appreciation for the weapons as their highlight of Unreal Tournament. As such, the weapon systems will be a priority throughout development.

1. The 90s aesthetic.





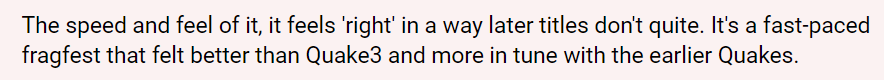
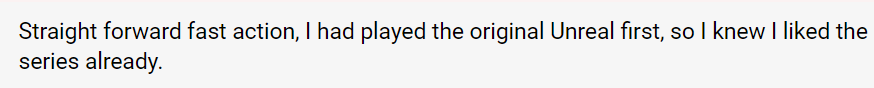
A major attraction of UT99 was the graphics and art direction. As well as having a great soundtrack, the visuals were gorgeous for the time. I will try to recreate a nostalgic style in the graphics, but with a modern twist of Physically Based Rendering offered by modern game engines to create realistic shader effects like reflections, ambient occlusion and varying roughness in surfaces. This blend will aim to draw in the nostalgic UT99 veterans, while not excluding modern players that expect a certain level of graphical fidelity. This balance will be crucial in maximizing the breadth of the game’s appeal.

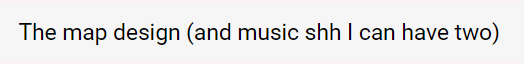
1. The fast-paced gameplay.



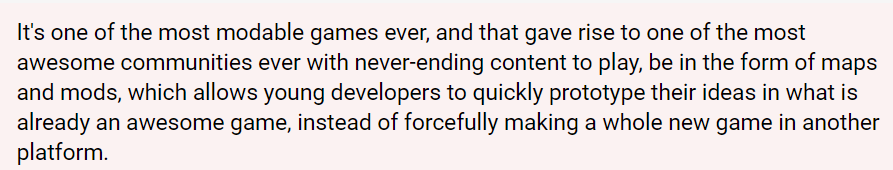




Clearly a shining pillar of the experience, one of the most popular opinions from the participants rung clear: the gameplay is key. I will make fast pacing and fluid gameplay a priority in development, even if it comes at a cost of lowering the graphical fidelity when needed.

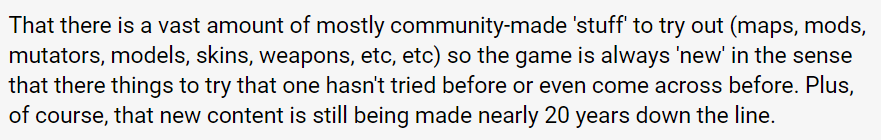
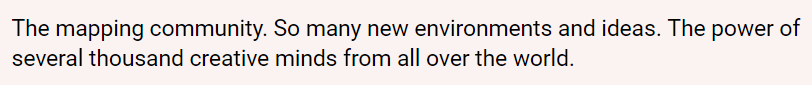
1. The level design.

Another frequently mentioned favourite feature is the level design and map variety. I will try to include at least two different playable maps for each game mode, with a variety in theme and layout. This will be crucial to getting the gameplay feel right, as you can’t have good gameplay with bad level design.

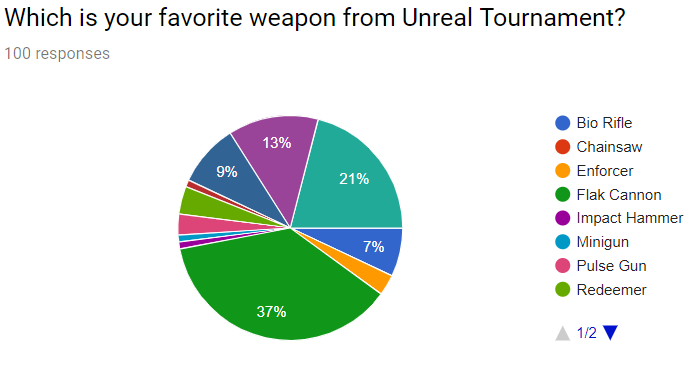
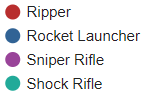
1. The mods and player-generated content.





The one key aspect of UT99 that keeps it relevant to this day is the level of customizability and mod-ability in the game. User generated content is key for keeping players playing the game years down the line. This will be the most challenging part of the experience for me to replicate, as modern game engines such as the one I am using (Unity) have a lot more limited support for external mods. However, there are some advantages to this. There are several modern tools for sharing and downloading user created maps such as the Steam workshop. However, I will need to pay a fee of £100 to list my game on the Steam store in order to use this feature so it may not be a possibility. If I have time, I will create an in-game level editor in order to address the community’s expectation of custom maps. Unfortunately, modding may be out of the realm of possibility.

### Favourite weapon?

As shown by the results, there are a few weapons that are standout favourites. This would be the Flak Cannon (37%), Shock Rifle (21%), Sniper Rifle (13%), Rocket Launcher (9%) and Bio Rifle (7%). As a result of this, since I won’t be creating a weapon roster nearly as large as that of UT99, I will prioritize creating weapons that fill the roles of these player favourites.

# Requirements

### System requirements:

In order to run my game, the user will be required to have a desktop PC or laptop that meets the game’s system requirements. As I have not yet finished the project, these requirements will be an estimation based on the requirements of the game engine, and the system requirements of games with similar levels of graphical fidelity.

#### Unity system requirements:

* Operating system: Windows 7 SP1 or later, macOS 10.11 or later, Ubuntu 12.04 or later
* Graphics card: Supports DirectX 10 & shader model 4.0
* CPU: SSE2 instruction set support

#### Estimated system requirements:

* Operating system: Windows 7 SP1 or later, macOS 10.11 or later, Ubuntu 12.04 or later
* Graphics card: Nvidia GeForce GTX 680 or better, AMD Radeon HD7970 or better
* RAM: 8GB or more
* Storage: TBD
* CPU: Intel core i5 680 or better, AMD quad-core equivalent or better

#### Stakeholder requirements:

##### Aesthetics / graphics:

|  |  |
| --- | --- |
| **Requirement:** | **Explanation:** |
| Sci-fi/fantasy art style | This is to stay faithful to the look and feel of UT99 in order to please the majority of the stakeholders who are nostalgic to the style of UT99 |
| Low fidelity textures and models | Again, this is to keep in the style of UT99 for the nostalgia factor. Additionally, this will speed up the modelling and texturing process and make the game run faster on people’s computers of varying specifications. |
| High quality shaders and image effects | This is to appeal to players used to the realism and graphical fidelity of modern games. This will set my game apart from other ‘retro’ styled games, as the combination of low-resolution assets and high-quality rendering will create a unique and interesting look |

##### User Interface:

|  |  |
| --- | --- |
| **Requirement:** | **Explanation:** |
| Modern layout | Taking cues from CS: GO’s slick modern UI, this is an important improvement to make over UT99. This was also widely complained about as an aspect of UT99 that needs to be updated by survey participants. |
| Modern yet retro-inspired visual design | In order to stay consistent with the modern/old design mashup of the game’s visuals, the UI will have a modern composition but sporting stylized pixelated GUI elements. This is again to appeal to the nostalgia of UT99. |

##### Weapons:

|  |  |
| --- | --- |
| **Requirement:** | **Explanation:** |
| Favourites of the UT99 community | I will be implementing weapons based on the weapon archetypes of the top 5 favourites from the UT99 community that participated in my survey. This would be the Flak Cannon (shotgun), Shock Rifle (rifle), Sniper Rifle, Rocket Launcher and Bio Rifle (grenade launcher?). |
| Varied set of weapons | This is an important pillar of the UT experience. I will achieve this by following the weapon archetypes of UT99. |

##### Gameplay:

|  |  |
| --- | --- |
| **Requirements:** | **Explanation:** |
| Variety of gamemodes | The UT99 player base have a split opinion between the game modes, so it will be essential to have a variety of modes to choose from |
| Smooth movement and networking | This will be crucial to the gameplay. Without a solid player controller system and netcode, the game as an experience will be unpleasant. |

##### Maps / Level design:

|  |  |
| --- | --- |
| **Requirements:** | **Explanation:** |
| Multiple maps for each gamemode | In order to keep the gameplay fresh, each gamemode has to have at least 2 maps as a minimum. |
| Variety in style | UT99 has a map roster featuring varying artistic and environmental styles. In order to meet the players’ expectations, I will create a variety of textures and models necessary to depict a variety of environments. |
| Variety in layout | Many of the survey participants expressed their love for the unique and varied maps for UT99. In response to this, I will take inspiration from the UT99 map pool to create some new and unique layouts. |

# Success Criteria

|  |  |
| --- | --- |
| **Criteria** | **How to evidence** |
| Main menu that gives you access to local and internet multiplayer as well as an options menu and the ability to host your own game. | Screenshot or screen recording of a working main menu with the described features. |
| Server browser that allows you to search for and connect to different matches, as well as an automatic matchmaker that puts you into a game lobby. | Screenshot or screen recording of a working server browser with the described features. |
| Game creation screen that allows you to select a game mode, map and game attributes. | Screenshot or screen recording of a working game creation screen with the described features. |
| Options menu screen with the ability to adjust audio, graphics and control settings. | Screenshot or screen recording of a working options menu with the described features. |
| A working capture the flag multiplayer match, showing multiple players on two teams and the gameplay of obtaining the flag. | Screen recording of an in-game match showing capture the flag working. |
| A working deathmatch multiplayer match, showing multiple players fighting free-for-all and one person winning by getting the most kills. | Screen recording of an in-game match showing deathmatch working. |
| A working demonstration of wide-area-network multiplayer showing two or more players on different local networks playing in the same online match in real time. | Video showing two people on separate networks in the same online game in real time. |
| A working demonstration of a shotgun-type weapon, dealing large damage close up and low damage at a distance. | Screen recording of a working shotgun-type weapon with its features demonstrated. |
| A working demonstration of a rifle-type weapon, dealing long distance damage when aimed precisely. | Screen recording of a working rifle-type weapon with its features demonstrated. |
| A working demonstration of a sniper rifle-type weapon, dealing very high damage when aimed at the body or head and a working zooming scope. | Screen recording of a working sniper rifle-type weapon with its features demonstrated. |
| A working demonstration of a rocket launcher-type weapon, shooting slow moving but high impact projectiles. | Screen recording of a working rocket launcher-type weapon with its features demonstrated. |
| Multiple maps for each game mode. | Screenshots of different maps. |
| Retro-inspired yet modern rendered visual style. | Screenshots showing the art style of the textures and models |