Budget Esports Video Game

Jason Tse

California State University Monterey Bay

CST499, Computer Science Capstone

Eric Tao, Brian Robertson, Cassandra Humphrey

10 October, 2019

Executive Summary

The goal of this project was to produce an innovative and demonstrative video game that catered to the needs of the underserved population aspiring to become a professional involved in the Esports industry. Vital features of the final video game includes an interface for adapting controls to a player's input device and a single demonstrative, playable stage. Features of this video game were created with the consideration of the hardware limitations of an average, lower-income consumer. The video game is intended to achieve an average of 60 frames per second, on a consumer laptop that is less than $500 in value without any aftermarket modifications. Controls for the video game does not require any external hardware. While the final product was intended for the amateur market, the video game's aesthetics are polished beyond an early developmental phase and satisfactory for a proposal presentation to a game studio. These polished aesthetics include effects, sounds, textured geometries, a user interface layer, and the ability to customize controls. The video game was deployed to Itch.io and distributed for free on the Windows. Linux, and Mac platforms.

The player plays the role of a machine operator sent to an unknown planet to gather resources. Upon landing on the planet, the player finds that there are plenty of hostile creatures who are composed of the resources being sought. The player must destroy these creatures, collect the resources dropped, and safely deposit the resources back at the rendezvous point. Projectiles and the combined weight and speed of the player's machine is required to deliver enough force to destroy the creatures. More resources collected means more weight to throw around, but it also means more time to build up speed and more strain on the machine. As the player progresses, the player must also manage the strain on their machine. Every action puts strain on

the machine which will recover slowly over time, but taking too much damage from monsters or straining the machine while overheating can cause permanent damage. How much resources can the player accumulate before the machine breaks down?

Table of Contents

Introduction

Project Description

This capstone project is titled “Budget Esports Video Game” and the purpose of this project was to create a video game that will have a minimal financial impact for aspiring Esports players and game developers. Success of this project was achieved by creating a game using free software and having that game run smoothly on cheap, non-enthusiast, consumer-grade, off-the-shelf, and unmodified hardware. This project did not aim to become a triple-A Esports title, but rather a stepping stone into the world of video games with the minimum of hardware requirements. The beneficiaries of this project were aspiring Esports athletes who lacked the funding to compete at professional standards and students interested in pursuing game development but lacked the knowledge or resources to start. With both target audiences, there was a need for accessibility. There existed underserved populations where school districts could not afford extracurricular activities that would propel their students into Science, Technology, Engineering, and Math (STEM) fields. Despite the seemingly contradictory entertainment value of video games, video game development is a great academic example of the culmination of the STEM fields. Video game development involves knowledge of computer science, physics, computer operation, problem solving, vector math, organization, psychology, critical thinking, and other valuable skill for students to acquire. While this project was not a panacea for enabling educational growth or becoming the future of Esports gaming, this low budget video game was meant to promote entry into the video game industry without exorbitant costs.

Problem and Issue in Technology

Video games are big. In 2018, the video game industry generated 134.9 billion dollars in

revenue globally; a 10.9% increase from the previous year (Batchelor, 2018). The Esports market alone is expected to surpass 1 billion dollars in revenue and garner a massive 453.8 million viewers (Pannekeet 2019). As the game industry grew increasingly larger each year, the amount of aspiring professional Esports athletes also increased. These Esports athletes invested greatly into hardware to acquire a competitive edge that will allow them to stand out amongst the growing competition and claim the top prize. Much like the athletes of traditional sports such as football or baseball, Esports athletes had to invest in the hardware, or their equipment, to be able to enter and participate in the sport. Hardware manufacturers have responded to demands with premium lineups of "gaming" oriented hardware. The costs of these specialty hardware rivaled and sometimes surpassed that of enthusiast-grade hardware.

Computer hardware defined much of an Esports athlete's competitive ability and the trend was that an increased monetary investment correlated to increased performance. Key factors affecting a player's performance included frame rate, resolution, and response times which depended on the hardware platform. The quality of peripherals such as the monitor, mouse, keyboard, headphones, and controllers greatly impacted a player's response times, so cheaper peripherals had potentially bottlenecked the amount of information a player could have sent and received per millisecond. Other bottlenecks included internal hardware components such as CPU, GPU, and RAM which worked in unison to reduce latencies and determine the amount of frames processed and rendered. Newer GPUs also enabled minute details to be rendered such as more accurate shadows or reflections. Unfortunately, high-performance internal hardware components also required an equally high-performance motherboard and cooling solution. As an added cost, gaming oriented hardware were often marketed with superfluous

aesthetic features that did not contribute to performance. At the time of writing this report, the cost of an Esports oriented gaming computer could have easily surpassed 2,000 US dollars. How are the underserved populations able to financially justify any ventures into the Esports world, or the more general video game industry, when the entry cost to obtain an industry experience is so costly?

Solution to the Problem and Issue in Technology

To enable more accessibility into the gaming world, either the cost of hardware would have to decrease or the demands on the hardware by video games would have to decrease. Manipulation of the premium gaming hardware market would have required immense investments or illegal activities, so the most viable and reasonable approach would be to tackle the issue from the consumer end rather than the manufacturer end. The solution already existed prior to the conception of this project, but the issue was the lack of awareness. By creating a modern video game that was made with free software and ran on cheap hardware, this project became a promotional tool to encourage more underserved populations to engage in Esports, game development, and STEM fields.

Projects Goals and Objectives

Beyond this project, the long term goal was to create an example for demonstrating that video games are viable to create, download, and play on a relatively low budget. The objective was to promote accessibility to lower income populations and especially tightly-budgeted educational institutions. This project also encouraged more to enter the video game industry. To fully achieve this goal, the resulting video game from this project was created using open-source

software or freeware, had a small footprint for portability, and did not require any enthusiast-market hardware to run. Based on these goals, the vital objectives were as follows:

- Game development involved the creation of a workflow from conception to deployment using open-source or free software.
- The development of assets was done with the final file size in mind. Final file size was kept optimally around 20 megabytes but strictly under 100 megabytes.
- Target platform for the final game was primarily a device with the Windows operating system, a dual-core CPU, integrated graphics, 4 gigabytes of RAM, a non-mechanical keyboard with about 88 keys, a touchpad or a 3-button mouse, and a 60 Hertz screen.
- The game achieved a steady 60 frames per second on the device described above.
- Players were able to customize the controls of the video game's character using an in-game key mapping user interface, and the default keymap would be exclusively limited to keyboard inputs.
- Players was able to control the video game's character to navigate the map and fire projectiles to achieve an end game result.
- Players was able to gauge the game's progression through a health bar.
- The video game had an element of replayability.
- The video game was deployed as an executable on the Windows platform through the Itch.io website.

Secondary objectives, which were mostly aesthetics, was implemented following successful implementation of the vital objectives. These secondary objectives included creating a textured models for the geometries, adding visual effects and sounds, and deployment to Linux and Mac

platforms. Should all the above objectives were fulfilled, remaining time was allocated towards creating new gameplay elements.

Community and Stakeholders

As briefly mentioned previously, the main beneficiaries for this project were the underserved populations interested in careers with Esports or seeking a pathway towards the video game industry or other STEM fields. Those considering pursuing Esports professionally are able to use the resulting video game from this project as a freely available resource to determine if the career route is a fit for their lifestyle. Students are able to use this project as a hands-on, interactive, and demonstrative resource to learn more about the video game development process and gauge their interest in pursuing higher education in video game development and STEM fields. This project also demonstrated the applications of arts and design fields within technological industries, so this project appealed to a broad scope of students. While there are other video games available that may fulfill such premises mentioned above, little to none of those other video games consider the burden of expenses towards underserved populations.

The greater community impacted by this project was the game development community. Within the community, there were common industry standards being promoted such as the use of commercial game engines like Unity or Unreal, and the use of C# for scripting. While those commercial engines had free free versions available, the promotion of these resources had created a negative perception of completely free and open-source alternatives within the gaming community. The perception was further enforced by the expansive portfolios of commercial engines and their AAA game titles. It was perceived that quality games could only be the product

of industry standard software and their implementations. This project challenged those industry standards by using completely free and mostly open-source resources. Hopefully, this project attracted more attention and funding towards the future development of open-source software, and inspired game developers to suggest such software as viable alternatives to commercial software.

Stakeholders and community did not face any losses from this project, as this video game was fulfilling a purpose that already needed more exposure. There were no compromises either. This project was a single video game in the vast, growing independent game development community, and the gains were likely minute. This project was not being developed for the mainstream and did generate any revenue directly. The main benefit of this project to the community was as representative product demonstrating the potential viability of game development without the need of premium hardware or software.

Evidence that the Project is Needed

With the proliferation of online streamers who showcase themselves playing games to large audiences easily numbering in the thousands, it is possible to forget that the digital divide still exists. Even in one of the most developed countries in the world, there exists communities with limited to no access to internet or even a computer. In a study by the Pew Research Center, it is reported that “24% of teens whose annual family income is less than $30,000 say the lack of a dependable computer or internet connection often or sometimes prohibits them from finishing their homework” (Anderson & Perrin, 2018). Shockingly, the study also shows that 9% of students who come from families earning at least $75,000. This drastic state of availability of computers for low-income families has two major implications. First, of the low-income

population that do have computers, it is likely that the computer owned by these families are not enthusiast-level systems. Second, investment towards video games are understandably unlikely because there are much more vital necessities to purchase. A lack of access to video games that will perform smoothly on non-enthusiast hardware, limits the future potential of prospective Esports players, game developers, and STEM students because they lack the information and experience of the field. This project aims to lessen the burden on those families and expand the opportunities afforded to low-income communities by giving them access to an experience they could otherwise not afford to invest into.

Feasibility Discussion

There were two major aspects to compare when comparing this project to other video games; the game's intention and the mechanics of the video game. It was difficult to determine if there existed a video game similar to this project due to the sheer amount of games in the market. There are many independent video games being released everyday. In 2017, the popular online video game distribution platform, Steam, hosted over 7,600 new video games (Wright, 2018). On average, Steam released more than 20 games every day and Steam is not the only platform that host games. While this project offered some unique sounding features, it is likely that the components of this project already existed in other possibly obscure video games. It was simply impossible to examine each and every AAA title, let alone the ever-growing number of independent games.

One of the main intentions of this project was demonstrating that video game development could have been done with little to no budget. When people think of video games, the big AAA titles with equally big budgets immediately came to most minds. Certainly, large

professional studios have numerous expenses and large overhead costs that gets considered into a game's budget, but even smaller independent studios have budgets starting in the hundreds of thousands per game (Pichlmair, 2011). The components of a stereotypical, modern video game required a breadth of professional fields that alluded to the necessity of developing video games in teams which implied necessary cost. There was not much information that could be found to support the development of a budget-free video game. An online search generally yielded a variety of free software and assets for making a video games, but not the development process. When searching for specific technologies, only then would one have found tutorials for video game development, but these tutorials would often require some beforehand knowledge that could potentially intimidate beginners. So while the resources for learning game development does exist for free, it was difficult to find a simple "yes" to the question, "can video games be developed without a budget?".

Another intention of this project was to develop a Esports-like game that catered to the technology owned by low-income individuals. Esports titles come in many genres, but the common factors would be multiplayer functionality and a competitive element. Of today's popular Esports titles like Fortnite, Apex Legends, and League of Legends, it would be difficult to source a laptop under $300 that meet the minimum specifications to run these games. Even if the device met the minimum requirements, the game settings would have to be set far from competitive levels. At the time of writing this report, one of the highest performance to cost laptops under $300 was Lenovo's 2019 IdeaPad 130 which retails for $299.99 on Amazon.com. This laptop had hardware indicative of the upper-end of the sub-$300 range. The IdeaPad 130 had a dual-core CPU with a base clock speed of 3.1 GHz, 4Gb of RAM, a 128Gb solid-state

drive, and AMD Radeon R5 integrated graphics. According to benchmarks from pcgamebenchmark.com, well-known ESports titles that could run at minimal settings on this device included Counter-Strike: Global Offensive, Dota 2, and the outdated Unreal Tournament 2004. Unfortunately, another bottleneck for low-income families were the large file sizes of these games. Both Counter-Strike: Global Offensive and Dota 2 boast file sizes of at least 6Gb which can be difficult to download with an unreliable internet connection.

Gameplay mechanics are limited by the hardware limitations of the controller, or input device, so innovations in gameplay mechanics are usually derivatives of existing mechanics and not completely new. For example, the player's movement in this project differed from the conventional first-person video game controls. Usually the w, a, s, and d keys are used for player movement while the mouse controls the point of view. This project featured a locked point of view and player movement was controlled by six keys controlling the forward, backwards, and sideways motion of the left and right sides of the character. The project's controls were more akin to flight simulators that maneuvered an aircraft by controlling individual wing flaps. This seemingly awkward control convention, gave players more control over the character and made piloting a skill vital to gameplay. Another differentiating mechanic of this game was the single energy and health indicator. It is common for players to have single health indicator where the indicator would decrement when damage is taken, increase when a recovery event is triggered, and potentially grow as the game progressed. Energy indicators would have similar traits but were often represented as a separate user interface element, such as the stamina bar from the Monster Hunter franchise or gas meters when riding in vehicles in many first-person shooter games. For this game, the player's health was equivalent to the energy remaining. Every action a

player took decremented the indicator, time incremented the indicator, and the indicator shrunk when damage was incurred. The use of a single indicator bar for both health and stamina helped to simplify the interface and not overwhelm players who were new to these types of gameplay.

Design Requirements

Functional Decomposition of Project

Major criterias that this project had to consider were hardware limitations and software design. The resulting game had to operate smoothly on specific hardware. In terms of external peripherals, the hardware limitations included a 60 hertz monitor, a non-mechanical keyboard, the possibility of having only a trackpad and no mouse or game controller. Crucial internal hardware limitations included consideration of a dual-core CPU, an integrated GPU, four gigabytes of RAM, and potentially low storage space. Other hardware limitations to consider were little or no access to internet and device form factors. Software design pertained to what game elements were necessary for the resulting game to be potentially considered as an Esports title. These game elements included a sense of repeatability, a method of gauging in-game progress and victory, an element that requires variable amount of human skill, a competitive element, and graphical elements. The graphical elements specifically must have been able to achieve a peak frame rate of 60 frames per second in tandem with the minimal hardware requirements. Other software design considerations included limiting the file size to potentially 20 megabytes but absolutely less than 100 megabytes.

Selection of Design Criterion

Laptops availed among other form factors. Desktop computers lacked the portability and convenience that appeals to most non-enthusiast consumers. Mobile devices were not ideal for

most academic uses and often had less processing power. To determine the type of laptop to target for this project, research was done on various online retail stores to determine what was considered a low price range and what specifications were common to devices in this price range (see Table 1). Operating systems with low market share, resale, and promotional pricing were not considered. In sampling devices, the most popular devices were considered if the retailer featured such designation.

Table 1

Survey of Budget Laptops from Online Retailers (September 26, 2019)

Device/Model	CPU*	GPU	RAM	Storage	Price	Source
Ideapad 130S	2 x 1.1 GHz	Integrated	4GB	64GB	$162.00	Amazon
Ideapad 130-15AST	2 x 3.1 GHz	Integrated	4GB	128GB	$299.99	Amazon
Aspire 5	2 x 2.6 GHz	Integrated	4GB	128GB	$349.99	Amazon
Ideapad S145	2 x 2.6 GHz	Integrated	4GB	500GB	$279.99	BestBuy
Ideapad L340-15API	2 x 2.6 GHz	Integrated	8GB	1TB	$379.99	BestBuy
14-CF0012DX	2 x 2.3 GHz	Integrated	4GB	128GB	$239.99	BestBuy
L402WA-EH21	2 x 1.5 GHz	Integrated	4GB	32GB	$229.99	Newegg
Vivobook E203MA	2 x 1.1 GHz	Integrated	2GB	32GB	$199.99	Newegg
Inspiron 11 3000	2 x 1.8 GHz	Integrated	4GB	500GB	$349.00	bhphoto
NX.H0UAA.005	2 x 1.1 GHz	Integrated	4GB	64GB	$269.00	bhphoto
Aspire 3	2 x 1.1 GHz	Integrated	4GB	500GB	$299.00	bhphoto
13.3” Notebook Flash	4 x 1.1 GHz	Integrated	4GB	64GB	$319.00	bhphoto

*Base clock speeds were recorded.

From Table 1, the most consistent trend for laptops were between the $200 and $300 range. The average clock speed was 1.95 gigahertz on a dual-core processor. Dedicated graphic cards were

not found within this price range. 4GB was a common amount of RAM. Storage varied greatly, but lower capacity devices often offered some sort of cloud storage solution. Screen refresh rate was not a specification with most product listing, but 60 hertz is a common frequency for today's non-enthusiast devices.

There are many artistic liberties in designing a game. One of the challenges of this project was balancing aesthetics and performance. 2D games are generally less compute intensive than 3D games, but the great majority of today's Esports titles are 3D. 3D games also enables more competitive gameplay mechanics seen in Esport titles, so 3D graphics was necessary for this project. The frame rate for this project's game was locked to 60 frames per second. While higher frame rates could allow for smoother animations, 60 frames per second matches the refresh rate of common monitors and locking the frame rate allows for fair gameplay against players who may have higher refresh rate monitors. The default controls for the game is mapped to keys on a tenkeyless, QWERTY keyboard seen on many laptops. To accommodate for individuals with disabilities, the game's controls are also customizable with external controllers. The customization also allows for overlapping keys which can simplify controls. Actual in-game mechanics were determined by the amount of assets needed to be generated in order to achieve gameplay resembling an Esports title. Due to time limitations, a single map with reusable assets was the most viable design. Arcade-style fighting games required too many animations to exhibit techniques. Real-time strategy games required too many assets for elaborate strategies. Racing simulation game controls were too different for a keyboard experience. Tournament-style action games appeared to be the most viable style of game for a minimal amount of animations and assets. While the lack of assets simplified gameplay, emphasizing the game's controls gave some

depth and difficulty to gameplay. Artificial players were added to simulate multiplayer to offset actual networking capabilities in case such features could not be implemented during the duration of this project.

Final Deliverables

The final product of this project was a playable game demonstration. Although not considered a fully complete game, the game implemented all major elements and mechanics for players to understand the nature of the game. The runnable files were hosted on an itch.io webpage for Windows, OSX, and Linux platforms. Along with the game, there was a development blog. This blog was separate from the course blog and detailed the completion of tasks and problems encountered during the development of the video game. To complement the blog, there was a public Github repository where all assets for the video game could be downloaded and inspected. As there was no specific client for this project, a client's approval was not part of the final deliverables.

Approach/Methodology

The primary software to create this game was intended to be a combination of Blender and the Armory3D engine. Blender is an open-source 3D modeling and animation software with various built-in media production features. The Armory3D engine was a game engine built to work inside of Blender while utilizing Blender's node-based graphic pipeline for constructing game logic and the Haxe-powered multimedia framework of Kha. Unfortunately, Armory3D was early in development and stopped functioning a couple weeks into the project. Godot Engine was used in place of Armory3D. Godot is a more mature engine that is standalone from Blender and, like Blender, is open-source. Other prominent software used for creating game assets included

Audacity, Bfxr, MediBang, and ArmorPaint. All software used for this project were either open-source or can be legally obtained for free.

In the initial developmental phases, Blender's default primitives was used as placeholders for staging models in the game. Concurrently, concept art for the final models was developed for accurate scaling purposes. The goal of this phase was to implement character movement using the default keymap and model basic terrain obstacle. Most of the time during this phase was allocated to adjusting the character's movement speed and maneuverability to suit gameplay. Godot's kinematic bodies was used for collisions. While rigid bodies would have relied more on the physics engine, reducing the amount of explicit code written, the behavior of kinematic bodies were more predictable. This phase also implemented the spawning of projectiles but not projectile collisions. No actual gameplay mechanics were implemented during this phase.

The following phase was the development of the key mapping user interface. A main menu interface was also created to act as a directory for the entire game. Upon a player's selection of the configuration option in the main menu, the player is taken to a new screen where they can customize the key mapping. On the screen, the character model is visible and responsive to the current key mapping. Simultaneously, the players are able to see a listing of corresponding keys to actions. Players are able to select an action and subsequently press the key to be reassigned to that action. Controls are then saved to Godot's dictionary object and outputted to an external file for subsequent use. Different implementations were needed for mapping keyboard and controller input as the input signals spawned unique child objects of Godot's InputEvent class. A function dedicated to converting input events was used to abstract the differences.

In the third phase, game dependent variables were created for player, enemy, projectiles, and the level itself. All coding was done in Godot's internal editor in the engine own language, GDScript. It was important that every character had its own script containing attributes and functions related only to itself. The only shared, global-like variable was gravity. It was important to keep only relevant code attached to the objects it represented because it allowed for better organization and modularity of the game. The main game loop started taking form at this point and a specification for end game conditions was formulated. Some early staging of the scene also occurred.

For the final vital phase, all spawning and interactive events was scripted. A user interface for the player's energy gauge and a set of temporary tests for gameplay behavior was created. Major stretch goals were implementing local area networking gameplay, replacing primitives with textured models, and implementing more aesthetic effects. This was the longest development phase which required hours of refinement. Deployment to the Windows platform, and other platforms such as Mac and Linux, was done using Godot engine and uploaded to the itch.io website.

Legal Considerations

All software used for this project was obtained legally and attribution was not required as per the license and user agreements. The resulting video game was also distributed for free and not intended for business purposes. Assets for the video game was personally designed, created, and did not utilize, reference, or modify any copyrighted material. When deployed on Itch.io, specified non-attribution licenses allowed for the free use and modification of the final product

without attribution or prior permission of the original developer. The game does not contain explicit materials and adhered to the guidelines of the Itch.io community.

Ethical Considerations

Video games have been the subject of many studies regarding health and have also been associated with unhealthy lifestyles. In 2018, the World Health Organization declared an addiction to video games as a mental disorder under the International Classification of Diseases (World Health Organization, 2018). More recently, mass shootings in the United States have been attributed to violent video games by some politicians. The events prompted the removal of certain video game titles and its promotional materials from one of the largest retail stores while still selling more imminently life-threatening firearms (Chapman et al., 2019). Does this make video games more dangerous than guns? What about countries that have a larger consumption rate of video games but drastically lower gun violence like South Korea or Japan? Despite conflicting concerns and statistics, consumers are wary of dangers that can prompt such violence as mass shootings. A major concern for this project was its availability. The main audience of this project is the underserved community who would not otherwise have access to video games. By enabling video games to reach a greater audience, this project could be seen as a vehicle for spreading violence and enabling unhealthy lifestyles.

This project would be considered a violent video game since the goal and mechanics are geared towards the use of force. Players are often submerged into a video game's world, and there are concerns that players would project the virtual world onto the real world and reenact violence as if they were in-game. To mitigate ethical concerns, the setting and characters are fictional, there is no gore, and low polygonal models abstract forms to reduce connotations to

real-world relationships. The controls are also more complex and different compared to common vehicles. By implementing mechanics that are hard to reproduce in the real world, players should find it difficult to reenact the game's mechanics in reality. To combat possible implications of causing unhealthy lifestyles, this project also considered the potential for addiction. The final product has no saveable data beyond keyboard and controller configurations. No competitive scoring are recorded beyond a game instance. Player character customization, which was an early concept feature, was not implemented so there are no in-game goals beyond an instanced end-game score. While limiting such features would usually negatively impact a video game's entertainment, these limitations were aligned with this project's goal of being an entry-level video game for academic and general learning purposes. Players of this game were not meant to play this game for a long period.

Timeline/Budget

The initial timeline was as outlined in Table 2, but non-development items related to presentations and such were removed as it was not relevant to the game element of the project.

Table 2

Project Timeline for Budget Esports Video Game Development

Task	Estimated Time Allotment
Week 0	
Setup development environment	6 Hours
Preliminary information and tutorials	4 Days
Concept designs	1 Day
Week 1	
Rough modeling of assets with rigging	2 Days

Develop character controls with animations	2 Days
Design main menu UI	1 Day
Week 2	
Create controller setup interface	3 Days
Structure character, enemy, and game objects	1 Day
Design main game UI	1 Day
Week 3	
Create map loader	1 Day
Code gameplay logic	4 Days
Develop enemy AIs	2 Day
Week 4	
Overflow	3 Days
Setup Itch.io	1 Day
Incorporate sounds	2 Days
Week 5	
Deploy game (continuous)	1 Day
Polish game	6 Days
Week 6	
Project testing	3 Days
Bug fixing and/or game polishing	2 Days
Week 7	
Presentation preparation	5 Days
Polish game	1 Day
Week 8	
Capstone presentation	1 Day

There were two major milestones and both milestones were met on-time and under budget. The first milestone was the completion of the controller customization interface. Despite later revisions for implementing joystick support, major functionalities for keyboard and joypad button input were completed on schedule by the end of week 2. The second milestone was the completion of the main game stage where players can play, interact, and get a grasp of the game's mechanics. While development of the second milestone was still in progress during the "overflow" period of week 4, development was still well within budgeted time. A third milestone which was not expected to be reached due to time constraints was the implementation of multiplayer gameplay over local area networks. This third milestone was also achieved by early week 6. The reason why the third milestone was achieved was due to the allocation of time for failure at a rate of double the original time budgeted.

Usability Testing/Evaluation

The testing of the game simply entailed having testers play the game. Many tasks were not explicitly outlined because part of the evaluation examined the intuitiveness of the game's structure and design. For example, the testers were not guided towards the control customization menu because the unorthodox controls should force the players to seek out the menu to familiarize themselves with the controls. Even the main requirement to play a complete round of the single player stage was not explicit because there is only one repeatable stage in the game to play. To ensure feedback was given for certain game elements, testing concluded with a survey (see Appendix A for testing plan and link to survey). The survey questioned the tester about the game's performance, the tester's device, and the responsiveness of the controls. A frame rate counter was exposed during gameplay and referenced in the survey to facilitate the report of

performance metrics. Questions in the survey also catered the underlying project by asking for opinions of the current state of accessibility to games and game development. As a final question, an open-ended question for general feedback was used. While the ultimate goal was not to get specific feedback pertaining to this game's mechanics, as Esport games span many genres, the open-ended question was meant to garner feedback that may have been applicable across game genres and alluded to more accessible design practices.

Evaluation of the game usability depended on the answers of three general questions.

1. Can a five minute round be played to completion without any fatal errors or glitches that may halt gameplay?
2. Is the game's objective understandable or intuitive enough to discover through visual cues?
3. Does the unorthodox controls distract from gameplay such that the game is less enjoyable or understandable?

The first question addresses the premise of this project which is the viability of creating an Esports game with a low budget. Failure to achieve a complete game might have indicated the necessity for proprietary software that offered more support or toolsets to overcome performance issues. The second question examines the game's visual design and layout. A negative response to this question is not necessarily detrimental to the project. A lack of intuitive design could be an issue similar to the first question where proprietary software is needed, or the negative feedback may highlight an element where better design could help improve accessibility. The third question gauges the difficulty of the controls because the controls were so unorthodox from any other game. This question helped to determine if the gameplay was balanced. There should

have been a learning curve to adapt to the controls, but the controls should not have been so complicated that it distracts the player from the game's objective or prevented gameplay completely.

Final Implementation

The game's development began with the development of some graphic assets. In figure 1, the different stages of creating a 3D model is shown.

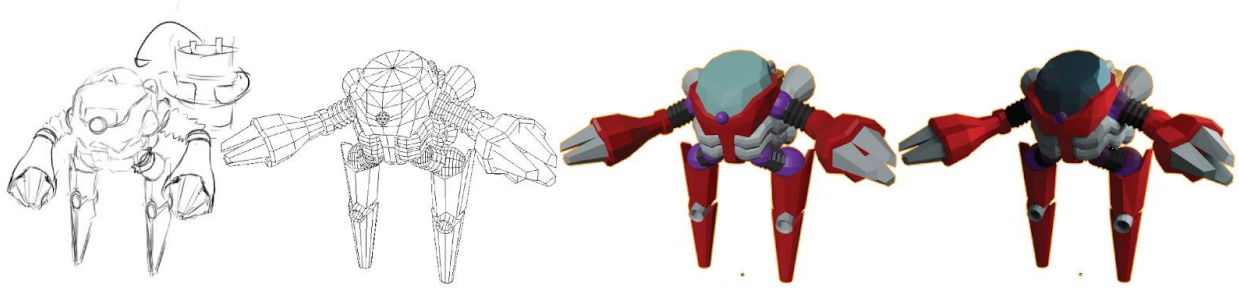


Figure 1. The stages in the creation process of 3D assets.

3D assets begin with a conceptual sketch, then the sketch is modeled with 3D polygons, base color is then applied, and finally lighting is baked so the engine does not have to calculate some of the lighting and shadows. After the model's appearance is set, the model is assigned as a child of a skeletal rig for convenient manipulation. Figure 2 shows the animation process where movements are "keyed" or saved into a timeline to produce an animation much like stop-motion animation. Blender interpolates between keyed frames for smoother animations.

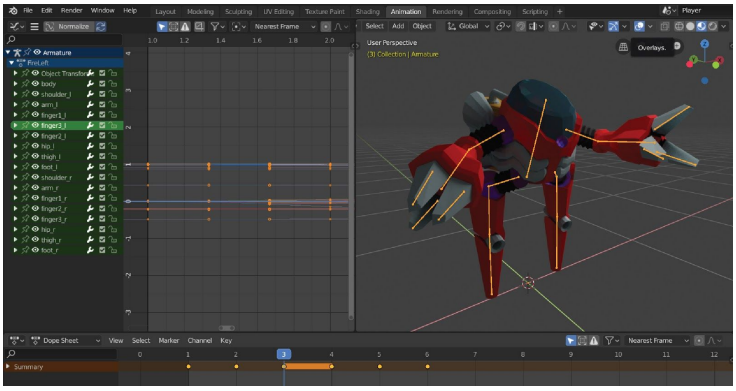
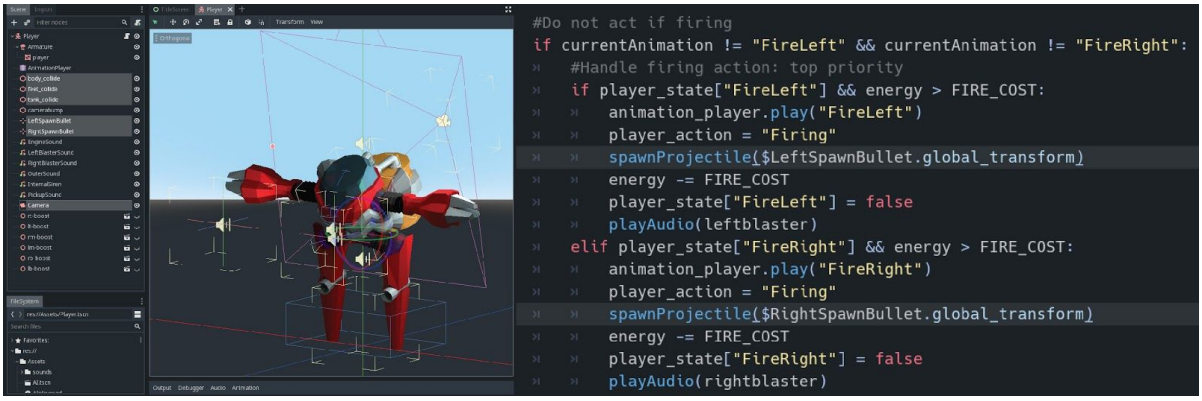


Figure 2. The Blender workspace for animating.

The entire model is then exported from Blender and imported into the Godot Engine. The format used for this project was the JSON encoded GL Transmission Format (gtlf format) which allowed for the export of the model, material, and animation in a single file. Figure 3 shows how a 3D model would have to then be outfitted with cameras, sensors, and targets for in-game interactions.



```
#Do not act if firing
if currentAnimation != "FireLeft" && currentAnimation != "FireRight":
    #Handle firing action: top priority
    if player_state["FireLeft"] && energy > FIRE_COST:
        animation_player.play("FireLeft")
        player_action = "Firing"
        spawnProjectile($LeftSpawnBullet.global_transform)
        energy -= FIRE_COST
        player_state["FireLeft"] = false
        playAudio(leftblaster)
    elif player_state["FireRight"] && energy > FIRE_COST:
        animation_player.play("FireRight")
        player_action = "Firing"
        spawnProjectile($RightSpawnBullet.global_transform)
        energy -= FIRE_COST
        player_state["FireRight"] = false
        playAudio(rightblaster)
```

Figure 3. The character's component setup.

The code on the right of the image exhibits some of the GDScript used to link the character to interactive components. Many user interface components are also setup in a similar manner. As shown in figure 4, modular buttons in the main menu and controller customization menu are triggered by events that executes scripts attached to nodes.

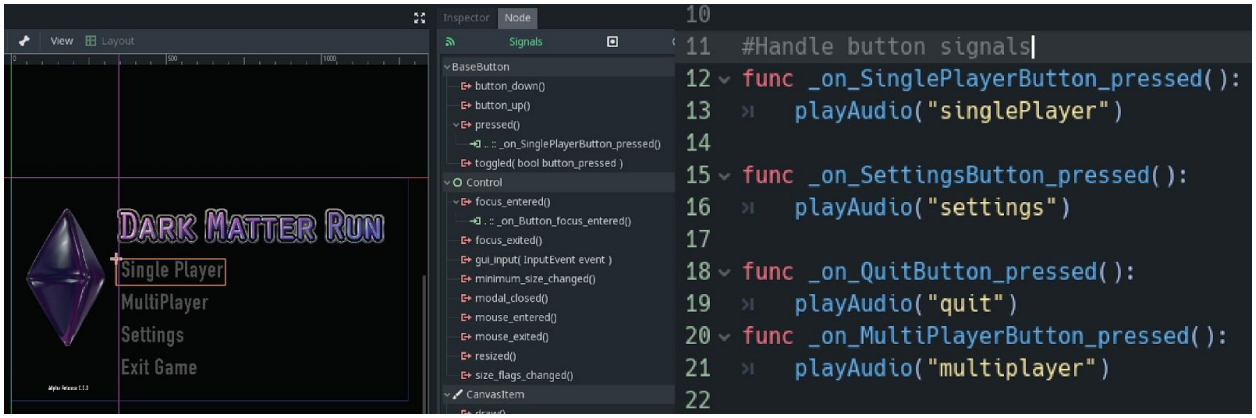


Figure 4. The main menu and its associated script for calling a globally defined function for changing the game state.

One of the difficulties of this project was allowing for various types of controllers. In the controller customization menu, various types of inputs are translated into a format that can be better accepted by the game. While Godot does have a built-in key mapping menu, their menu is not very flexible for changing key maps in-game. Figure 5 shows how different types of key presses are translated.

```
#The default key map
var default_controls = {
    #Format is ["event class", global enum value, axis orientation (-1, 0 for N/A, or 1)
    "LeftForwardKey" : ["InputEventKey", KEY_E, 0],
    "LeftBackwardKey" : ["InputEventKey", KEY_D, 0],
    "LeftStrafeKey" : ["InputEventKey", KEY_S, 0],
    "LeftFireKey" : ["InputEventKey", KEY_F, 0],
    "RightForwardKey" : ["InputEventKey", KEY_I, 0],
    "RightBackwardKey" : ["InputEventKey", KEY_K, 0],
    "RightStrafeKey" : ["InputEventKey", KEY_L, 0],
    "RightFireKey" : ["InputEventKey", KEY_J, 0]
}
#Changes input code to string
func get_input_name(inputValueArray):
    if inputValueArray[0] == "InputEventKey":
        return OS.get_scancode_string(inputValueArray[1])
    if inputValueArray[0] == "InputEventJoypadButton":
        return Input.get_joy_button_string(inputValueArray[1])
    if inputValueArray[0] == "InputEventJoypadMotion":
        if inputValueArray[2] > 0:
            return Input.get_joy_axis_string(inputValueArray[1])
        else:
            return str("-" + Input.get_joy_axis_string(inputValueArray[1]))
```

Figure 5. Key encoding for game event translation.

For the first and only stage of the playable game, the stage's initial layout has player models in their starting positions and established areas that generate instances of enemy characters. Things like projectiles and reward items are instanced by the object that logically owns the item. Once instanced, those same items are essentially their own individual entity that is free to react to its environment. Figure 6 highlights the code for a projectile used by the player and the functions that dictates its lifecycle.

```
extends KinematicBody

var SPARKS = preload("res://Assets/Sparks.tscn")
var SPEED = 2


# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta):
    var col_info = move_and_collide(get_transform().basis.xform(Vector3(0, 0, SPEED)))

    #check for collision
    if col_info:
        #damage colliding object
        if col_info.collider.has_method("damage"):
            col_info.collider.damage()

        #create sparks and remove
        var sparks = SPARKS.instance()
        sparks.transform = self.transform
        get_parent().add_child(sparks)
        queue_free()
```

Figure 6. The life of a fired laser.

Any in-game object also holds all variables that define the state of the object. This is especially of importance for the player. Figure 7 shows code attached to the player's character for interacting with other objects.

```
        #COLLISION DETECTION
        if col_info:
            #if collider can be knocked back, knock it back
            if col_info.collider.has_method("knockback"):
                col_info.collider.knockback(mass + 1000, speed, col_info.position - self.transform.origin)
            #if collider can be picked up, pick it up and add to mass
            if col_info.collider.has_method("pickup") && mass < MAX_MASS:
                col_info.collider.pickup()
                pickup.play()
                mass += randi()%51+50
                if mass > MAX_MASS:
                    mass = MAX_MASS

```

```
#adds damage taken to total damage taken
func damage(amount = 10):
    playerUI.flicker()
    damaged += amount
    #lose cargo when damaged
    mass -= int(amount)
    #check min mass
    if mass < 0:
        mass = 0


#knocks character back and does damage
func knockback(weight, velocity, normal):
    playAudio(outer)
    var force = weight * velocity * 3    #adjust last number to change knock back intensity
    var knock_vector = Vector3(normal.x * force, 0, normal.z * force)
    move_and_slide(get_transform().basis.xform(knock_vector))
    damage(int(force/30))

```

Figure 7. Functions that allow player to interact.

With every frame of the game, the player's character reports its state to the user interface for displaying. When the in-game timer expires, the user interface polls the player and automated characters for the amount of rewards obtained. The amount returned represents the players' final score and it is recorded globally for in the next scene (figure 8).

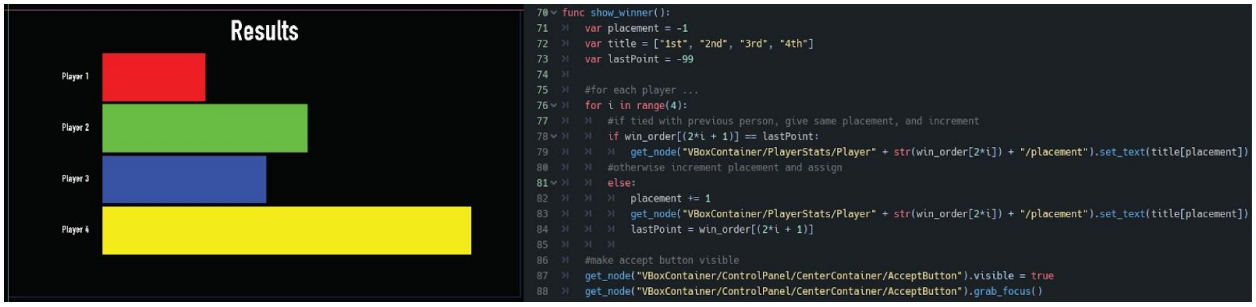


Figure 8. The results scene.

For a more in-depth examination of the code used in this project, the gdscript files, noted by their ".gd" extension, are available for public viewing at the game's repository (Appendix C).

Discussion

Many elements in the original concepts of the game had been altered or changed to meet the time constraints of this project. The original game concept was a tower defense game where players gathered resources, store resources at towers located where they started, and protect the tower from other players. After early testing of the game, the return-to-starting-point element was removed because the unorthodox controls made navigation and positioning difficult for anyone new to the game. It created a high learning curve. To alleviate some of the difficulties of navigation, left and right maneuverability was added via the strafing controls. Additionally, omitting features that aided the player's location awareness such as a stage map, 3D models for towers, and unique landscape features help to decrease impact on performance. In the original concept, there was also the ability for the player to customize the appearance and performance of

their character. Customization required an immense amount of 3D asset design and creation which was not a skill meant to be highlighted for this project, and highly unrealistic for a short time frame in a solo project. Other omission and changes include the changing of first-person perspective to third-person perspective and the removal of the weight gauge. The change in perspective allowed the player to have more visual feedback with navigation controls and the weight gauge was removed because it was a distracting element that was not informative. Unfortunately, the final product became so featureless that the lack of features could have contributed to the lack of interest that made obtaining feedback difficult.

There were difficulties with the game engine chosen at the start of the project. Armory3D was the engine used early in the project, and the engine appeared to be the perfect fit for the heavily open-source workflow because it was embedded into the 3D modeling software, Blender. Being embedded into the asset creation software eliminated the need to translate meshes, materials, and animations into separate file formats. After getting as far as creating the game's main menu, Armory3D properties were no longer visible in Blender. The cause is still unknown and reinstalling Blender and Armory3D had no effect. Priority switched to the immediate adaption of the more mature Godot Engine. All knowledge gathered in preparation of using Armory3D did not translate to Godot Engine as the structure, user interface, and underlying language were completely different. The workflow also had an extra step added for exporting assets from Blender to Godot Engine which did not always translate assets faithfully.

As previously mentioned, there was difficulty in obtaining feedback for this game. The game was released publicly five weeks after the official start of the project, allowing for two full weeks of feedback. In the first week, the game's Itch.io page was viewed at least 30 times but

only downloaded 8 times. The game was promoted on Itch.io community forums, Twitter, and in a couple of relevant Reddit communities. Initially, the game was to be tested in-person with a local video game interest group, but contact with the coordinator was unreliable at best. Similarly, no communications could be established with a local college video game development club. While the survey was also promoted with the game, early feedback did not result in a survey response. A particular struggle to advertising the game was trying to make a promotional video for the game. There was a promotional video made for the capstone project, but the intended audience was not the same. The struggle was with making the game look attractive despite the lack of features and time to refine aesthetics. Ultimately, the aid of the project's advisors was obtained to help source some much appreciated testers.

Conclusion

With the current gaming industry catered towards prosumers devices, the limitations of low-end consumer hardware are disregarded in the development of popular Esports titles. Creating a low budget, Esports-like video game that caters to the technological limitations of less fortunate communities demonstrates the viable ease of accessibility to videos games and game development resources for aspiring game developers and Esports athletes. Furthermore, using free and open-source software encourages the proliferation of independent developers and lessens the stigma of games made with non-proprietary software in the video game industry. The game made for this project specifically made use of Blender and the Godot Engine. Mechanics of the game resembled tournament-style shooter games, but unorthodox controls for piloting the player's character was introduced to differentiate from similar games and cater to those who may only own a keyboard as their only input peripheral. Careful consideration was taken to keep file

sizes at a minimum for communities without access to high-speed internet connections. A multiplayer functionality that relies only on the local area network further allows for Esport-like competition gameplay without reliance on a stable internet connection to an external server. While this game is not a polished, finished product, the potential to expand into a full game is a task that requires more time and not large, expensive software.

This project was a lesson of frugality in game development. With limitations on file size, there are more conscious design choices with assets and game structure. There is a want to be thrifty and try to recycle resources thanks to the modularity of the Godot Engine. At the same time, there is a demand from players for a certain quality that developers must balance with a certain aesthetic. Good graphics these days are equivalent to how much realism the graphics can portray. An alternative to realism is having a super stylistic, but still cohesive, art style. This game's art did not have cohesiveness and did not lean far enough in a realistic or stylized aesthetic. The game also lacked many quality mechanics that gives games depth. With how expectedly unorthodox this game's mechanics has and will become, working on this project as a solo developer allowed for a much needed flexibility in decision making and pacing. Future development for this version of the game will be discontinued. Instead, this game will benefit from being redesigned from scratch while adhering closer to the original concept. Asset design will incur the greatest benefit since art was not a priority for this project. Even with limitations loosened for a remake, the resulting game will hopefully be suitable for lower-end hardware and inspire future developers and Esports athletes who would otherwise not have involve themselves with the video game industry.

References

Anderson, M., & Perrin, A. (2018, October 26). Nearly one-in-five teens can't always finish their homework because of the digital divide. Retrieved September 12, 2019, from https://www.pewresearch.org/fact-tank/2018/10/26/nearly-one-in-five-teens-cant-always-finish-their-homework-because-of-the-digital-divide/

Batchelor, J. (2018, December 17). GamesIndustry.biz presents... The Year In Numbers 2018. Retrieved July 27, 2019, from https://www.gamesindustry.biz/articles/2018-12-17-gamesindustry-biz-presents-the-year-in-numbers-2018

Chapman, M., Anderson, M., & Pasani, J. (2019, August 9). Walmart pulls violent game displays but will still sell guns. Retrieved September 12, 2019, from https://www.apnews.com/26c7aeb48fda4a168232241714e8d775

Pannekeet, J. (2019, February 12). Newzoo: Global Esports Economy Will Top $1 Billion for the First Time in 2019. Retrieved July 27, 2019, from https://newzoo.com/insights/articles/newzoo-global-esports-economy-will-top-1-billion-for-the-first-time-in-2019/

Pichlmair, M. (2011, May 2). Opinion: Indie Project Budgets. Retrieved from https://www.gamasutra.com/view/news/124674/Opinion_Indie_Project_Budgets.php

World Health Organization. (2018, September 11). Management of substance abuse - Gaming abuse. Retrieved August 11, 2019, from https://www.who.int/substance_abuse/activities/gaming_disorders/en/

Wright, S. T. (2018, September 18). There are too many video games. What now? Retrieved August 4, 2019, from https://www.polygon.com/2018/9/28/17911372/there-are-too-many-video-games-what-now-indiepocalypse

Appendix A

Usability/Evaluation Test Plan

The following table outlines the original survey questions to be included. Many game-specific questions were ultimately excluded because this project considers all genre of Esports titles.

Subject	Criteria	Rating Scale
Player Survey	Gaming habits description	Text
	Device used to play this game	Multiple choice
	Operating System	Multiple choice
	Dedicated graphics card	True/False
	Graphics Card	Text
	Retail price of device	Text
Game Performance	Game loads quickly	1 - 5, Agreement level
	Able to achieve 60+ FPS consistently	True/False
	Where was there lag, if any?	Text
	Controls are responsive	1 - 5, Agreement level
Game Assets	Quality of assets' design	1 - 5, Satisfactory level
	Any defective/unsatisfactory elements?	Text
	Effectiveness of UI	1 - 5, Satisfactory level
Game Logic	Quality of gameplay	1 - 5, Satisfactory level
	Difficulty of gameplay	1 - 5, Difficulty level
	Quality of customizable controls	1 - 5, Satisfactory level
	Was it difficult to learn gameplay?	Text
Player's Opinion	Can this game be an Esports title?	Text
	Is there a need for more accessible gaming?	Text

	Is it easy to get started in game development?	Text
	Gameplay video	URL, optional
	Other comments	Text, optional
The final survey can be found here: *https://docs.google.com/forms/d/e/1FAIpQLSdz8sKZYNaZ-2s6Y-qwJ7bQoeB6AubyrAZLh4s5wHKAbV2Iog/viewform?usp=sf_link*		

Appendix B

Team Roles, Responsibilities, and Division of Work

This capstone was completed by the sole member, Jason Tse. Jason was responsible for scheduling, conceptualizing game design, visual and audio asset creation, programming, initial testing, and deployment of the game.

Appendix C

Relevant Links

The game is hosted on the Itch.io game distribution platform: https://jatse.itch.io/dark-matter-run

Source code and assets can be found on the Github repository: https://github.com/jatse/Capstone