ONLINE ARRENA BATTLE MULTIPLAYER GAME USING UNITY NETWORKING

D.Prabhu [1] G.Sai Priya [2], SK Mojamill jan [3], Y.Vamshi Teja [4,] G Nikhil[5]

[1] *Assistant Professor(O.G.), Department of Computer Science and Engineering*
[2] *Student, Department of Computer Science and Engineering*
[3] *Student, Department of Computer Science and Engineering*
[4] *Student, Department of Computer Science and Engineering*
[5] *Student, Department of Computer Science and Engineering*

ABSTRACT

Using unity and blender software we are designing a 3D game model which includes a dinosaur and a man. For the game, developing 3D interaction techniques for a shooting game. This game is based on a category called as TPS (Third Person Shooter). Third-person shooter (TPS) is a video game genre centered on gun and other weapon-based combat in a first person perspective. If we talk about the base-structure of the game, we are introducing two dinosaurs and man models from the blender software and importing it into unity software were we are actually designing the game. For this game, we are bringing c# script into the field. With this entire introduction we are developing the game. For the overview of the game, the concept is, a man will have to shoot his enemies before his own life will come to an end. As in for our multiplayer game, we have to create servers using Unet. This client in the server is the main part of the game. With this we are going to create a game.

Keyword: - *3D user interfaces, substance painter, texturing, coding, uv mapping, characterization, UNET***.**

I. INTRODUCTION

In recent years, a significant rise in computing power and storage devices has generated enormous interest in data-driven approaches to studying all kinds of phenomena. Understanding and explaining social behavior can fundamentally alter the functioning of professional organizations, battlefield scenarios, and multi-player professional teams. The rapid advancement of data-driven techniques, combined with greater availability of data and greater capabilities in data storage have now allowed computers to better analyze and model social phenomena. THIRD PERSON SHOOTER (TPS) is a subgenre of 3D shooter games in which the player character is visible on-screen, and the gameplay consists primarily of shooting. A 3D game type that has grown to prominence in recent years, especially on consoles. It combines the shooting elements of the first-person shooter with the jumping and climbing puzzles of a 3D platformer and a simple melee fighting system from a brawler. Third-person shooter games almost always incorporate an aim-assist feature, since aiming from a third-person camera is difficult. Most also have a first-person view, which allows precise shooting and looking around at environment features that are otherwise hidden from the default camera. In most cases, the player must stand still to use first-person view, but newer titles allow the player to play like a FPS; indeed, Oddworld: Stranger's Wrath requires the player to shoot from first person, only allowing melee attacks in the chase camera views. A client is a player's computer connected to a game server.

II. BACKGROUND LITERATURE

In SHOOTING games, human players expect to control their robot by making small adjustments to the robot's direction while moving, just like how they would control a robot through remote controller in real life. Therefore, the Player model must make continuous adjustments to its Player's direction as opposed to simply shooting in a straight line and turning only when a curve in the arena approaches. To ensure the Player can mimic human behavior in this manner, some AI techniques were used for implementing the bots in Player, which are explained in detail in the following section.C# for the programming language, unity software where we worked for the game , autodesking for the background of the game . if we first talk about the blender the it is a software where we develop our own characters for the game or for the required game. Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender's API for Python scripting to customize the application and write specialized tools; often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process. Examples from many Blender-based projects are available in the showcase. Blender is cross-platform and runs equally well on Linux, Windows, and Macintosh computers. Its interface uses OpenGL to provide a consistent experience. To confirm specific compatibility, the list of supported platforms indicates those regularly tested by the development team. The secondary part we are going to talk about the texturing of the model. As we have to give some textures for the model which has to be imported into the unity. The material settings that we've seen so far produce smooth, uniform objects, but such objects aren't particularly true to reality, where uniformity tends to be uncommon and out of place. In order to deal with this unrealistic uniformity, Blender allows the user to apply textures which can modify the reflectivity, specularity, roughness and other surface qualities of a material.

The third part is all about the unity where we actually give life to our game. UNITY software is the cross-platform game engine developed by Unity Technologies , which is primarily used to develop video games and simulations for computers, consoles and mobile devices. First announced only for OS X, at Apple's Worldwide Developers Conference in 2005, it has since been extended to target 27 platforms. For introducing the character and scenes we need to go for the separate tools. All the tools has to be set according to the needs. As if it is about the introduction of a scene then we need to create the scene and then background for it. Now this scene needed the position that is set in the inspector . We have three scales x,y,z. x is for the normal position , y is for the rotation and z is for the opposite direction of x and y. so after setting up this all tools then the scene will be shown in the background . The same procedure will be followed if we are bringing a character in the unity but with different tools . This is all about the codinng part in the unity software.

III. NETWORKING AND SERVERS FOR THE MULTIPLAYER GAME

Multiplayer games based on the Source Engine use a Client-Server networking architecture. Usually a server is a dedicated host that runs the game and is authoritative about world simulation, game rules, and player input processing. A client is a player's computer connected to a game server. The client and server communicate with each other by sending small data packets at a high frequency (usually 20 to 30 packets per second). A client receives the current world state from the server and generates video and audio output based on these updates. The client also samples data from input devices (keyboard, mouse, microphone, etc.) and sends these input samples back to the server for further processing. Clients only communicate with the game server and not between each other (like in a peer-to-peer application). In contrast with a single player game, a multiplayer game has to deal with a variety of new problems caused by packet-based communication.

Network bandwidth is limited, so the server can't send a new update packet to all clients for every single world change. Instead, the server takes snapshots of the current world state at a constant rate and broadcasts these snapshots to the clients. Network packets take a certain amount of time to travel between the client and the server (i.e. half the ping time). This means that the client time is always a little bit behind the server time. Furthermore, client input packets are also delayed on their way back, so the server is processing temporally delayed user commands. In addition, each client has a different network delay which varies over time due to other background traffic and the client's framerate. These time differences between server and client causes logical problems, becoming worse with increasing network latencies. In fast-paced action games, even a delay of a few milliseconds can cause a laggy gameplay feeling and make it hard to hit other players or interact with moving objects. Besides

bandwidth limitations and network latencies, information can get lost due to network packet loss. The server simulates the game in discrete time steps called ticks. By default, the timestep is 15ms, so 66.666... ticks per second are simulated, but mods can specify their own tickrate. During each tick, the server processes incoming user commands, runs a physical simulation step, checks the game rules, and updates all object states. After simulating a tick, the server decides if any client needs a world update and takes a snapshot of the current world state if necessary. A higher tickrate increases the simulation precision, but also requires more CPU power and available bandwidth on both server and client. The server admin may override the default tickrate with the -tickrate command line parameter, though tickrate changes done this way are not recommended because the mod may not work as designed if its tickrate is changed.

Note: The -tickrate command line parameter is not available on CSS, DoD S, TF2, L4D and L4D2 because changing tickrate causes server timing issues. The tickrate is set to 66 in CSS, DoD S and TF2, and 30 in L4D and L4D2.

Clients usually have only a limited amount of available bandwidth. In the worst case, players with a modem connection can't receive more than 5 to 7 KB/sec. If the server tried to send them updates with a higher data rate, packet loss would be unavoidable. Therefore, the client has to tell the server its incoming bandwidth capacity by setting the console variable rate (in bytes/second). This is the most important network variable for clients and it has to be set correctly for an optimal gameplay experience. The client can request a certain snapshot rate by changing cl_updaterate (default 20), but the server will never send more updates than simulated ticks or exceed the requested client rate limit. Server admins can limit data rate values requested by clients with sv_minrate and sv_maxrate (both in bytes/second). Also the snapshot rate can be restricted with sv_minupdaterate and sv_maxupdaterate (both in snapshots/second).

The client creates user commands from sampling input devices with the same tick rate that the server is running with. A user command is basically a snapshot of the current keyboard and mouse state. But instead of sending a new packet to the server for each user command, the client sends command packets at a certain rate of packets per second (usually 30). This means two or more user commands are transmitted within the same packet. Clients can increase the command rate with cl_cmdrate. This will increase responsiveness but requires more outgoing bandwidth, too.

Game data is compressed using delta compression to reduce network load. That means the server doesn't send a full world snapshot each time, but rather only changes (a delta snapshot) that happened since the last acknowledged update. With each packet sent between the client and server, acknowledge numbers are attached to keep track of their data flow. Usually full (non-delta) snapshots are only sent when a game starts or a client suffers from heavy packet loss for a couple of seconds. Clients can request a full snapshot manually with the cl_fullupdate command.

Responsiveness, or the time between user input and its visible feedback in the game world, are determined by lots of factors, including the server/client CPU load, simulation tickrate, data rate and snapshot update settings, but mostly by the network packet traveling time. The time between the client sending a user command, the server responding to it, and the client receiving the server's response is called the latency or ping (or round trip time). Low latency is a significant advantage when playing a multiplayer online game. Techniques like prediction and lag compensation try to minimize that advantage and allow a fair game for players with slower connections. Tweaking networking setting can help to gain a better experience if the necessary bandwidth and CPU power is available. We recommend keeping the default settings,since improper changes may cause more negative side effects than actual benefits.

IV. OVERVIEW OF THE PROJECT

THIRD PERSON SHOOTER ARENA MULTIPLAYER GAME USING UNITY NETWORKING is the name of the project. Third-person shooter (TPS) is a subgenre of 3D shooter games in which the player character is visible on-screen, and the gameplay consists primarily of shooting. And the theme of the project is to provide fun to the players with high quality graphics, player model and attractive Sci-Fi arena. The concept of actual game is taken from HEAT(1998) the third person shooter game which is not a multiplayer game. With the same concept we externally added multiplayer using UNET. Story-line is some players were added into server where they get external point to update the levels. As per the kill feeds the player will be awarded with the external points. Thus by he can buy new weapons. The game mode is FREE FOR ALL, where each player have to shoot the other player to get those external points. The game is made using c sharp project. For this multiplayer TPS we are creating our own models in the blender software. Our model is a robots with guns. As it is shown in the figure:

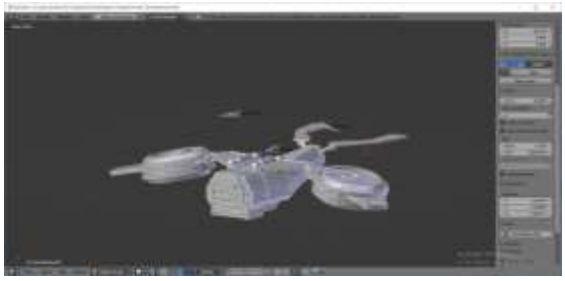

Fig-1: Character Model

Blender is a professional, free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, camera tracking, rendering, motion graphics, video editing and compositing. It further features an integrated game engine.

For this model, we are giving some textures. As after giving the textures and then we will upload colours for the models in substance painter. Our proposed model will look like:

Fig-1: Character for Texturing in Substance Painter

For this model we need to assign some resolution. For this we introduce UV mapping.

Now for the model we need to give movements which is called as rigging of the model. Rigging comprises the system of ropes, cables and chains, which support a sailing ship or sail boat's masts—standing rigging, including shrouds and stays—and which adjust the position of the vessel's sails and spars to which they are attached—the running rigging, including halyards, braces, sheets and vangs. the model will look like as shown in the figure:

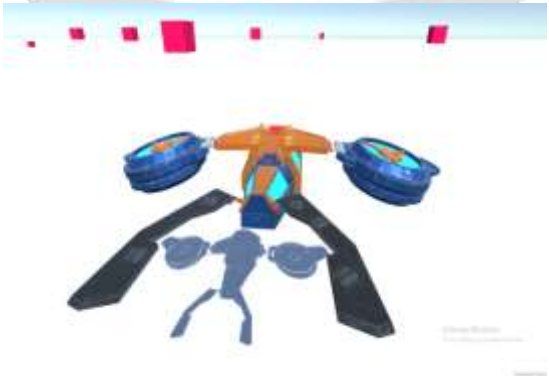

Fig-1: Model Imported to Unity Software

Finally we will import the model into unity. Where we will give life of our game. At first we will give player controllers and then we will introduce the movements through the controllers. As in unity we have to go for each and every step by changing in the inspector for the particular model and parts of the model. Coding is the main base structure of every movements of the model. Unity is an all purpose game engine that supports 2D and 3D graphics, drag and drop functionality and scripting through C#.

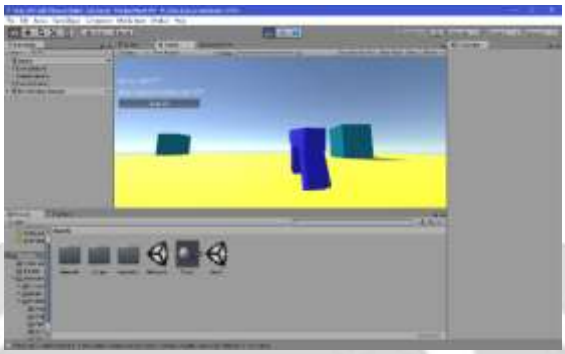

Fig-1: Reference Model in Unity

For 3D games, Unity allows specification of texture compression and resolution settings for each platform that the game engine supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects.

. For that we created ip addresses for the players. It depends upon how many players can join the game and play. So this networking is the main part in the multiplayer games. After giving the networking to no. of players we will finally bring our actual model and merge all the files to the main model of the game. Since it is a multiplayer game so separate introducing of the enemy player model is not needed. And at final our main part of the game is to give no. of levels which is included in our game. We can give multiple levels. The basic difference of the levels is, the life of the player model will be increased or boosted as the levels are introduced as well as the powers of the enemy and the robot will be increased. Even we can change the guns. And also some changes will be introduced in the model as the levels are increased. This is actual overview of the game.

V. EXISTING MODULE

- In the existing part of the project, the platform used is unity and for the coding part c sharp . But the blender software is not used for the character modelling.
- Control of the game engine will be taught with a visual state machine editor and runtime library that allows students to develop video games without any programming experience. But here we are introducing c sharp without any libraries.
- Code craft is an educational game developed at a CBU to tech computational thinking. It elicits computational thinking by requiring players. But here we are not introducing codecraft. This is also a 3D environment. As this environment is introduced by ECE students. Code craft is used for code puzzles.
- As in the video games mostly c sharp is used if it is about unity software because it is the easiest programming language.

VI. PROPOSED MODULE

- In our game, we are using blender software for the creation of the character. using the 3D sheet . The same has being done for weapons and also background scene.

- For the painting part , we are using substance painter software where we are giving separate and different colours for different parts of the character .
- Before introducing substance painter we are also introducing the vertex painting which helps us to distinguish with the parts of the character .
- In our project we are going for the c sharp coding in the unity software for accessing the character through the player and also for attacking the enemy bots. This all is done without bringing the libraries.
- Our project is totally based on TPS , this is the third person shooter. This is introduced in maximum games. The unique part in our game is the character as it is not a human model.

VII. FUTURE WORK

Our future work is to introduce the game as an android application. We are going to provide multiple levels and also upgrades of the guns as well as the robots. For the game we will have to provide actual layout from the loading bar to the completion of the game. We will also give the garage where we can upgrade the player model and the guns or simply where we can give shields to the robots by our choice as well as Secondary weapon. If we talk about the android application, Android software development is the process by which new applications are created for the Android devices operating system. Applications are usually developed in Java programming language using the Android software development kit (SDK), but other development environments are also available. The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications. Now a days people are mostly connected to the mobile games apps. As according to this new and fast generation we decided to create our game as an android application. The future work for our game is upgrades as we discussed before. This is a list of weapons that you can buy and keep updating/using for the whole game; most of them are useful in top level battles.

If you are on a budget/free player, keep clear of “EXPENSIVE” weapons, because they are not best "bang for bucks" available, and you can still have great game without buying it.

Short range (0–350 meters):

- Thunder – Heavy. Effective range 200m. Silver weapon.
- Pinata – Light. Effective range 300m. Silver weapon.
- Magnum – Light. Effective range 350m. WP weapon.
- Taran – Medium. Effective range 350m. WP weapon.
- Aphid – Light. Effective range 350m. Gold weapon. EXPENSIVE.
- Orkan – Medium. Effective range 300m. Gold weapon.
- Medium range (400–600 meters):
- Tulumbas – Medium. Effective range 500m. Silver weapon.
- Pin – Medium. Effective range 500m. Silver weapon.
- Trident – Heavy. Effective range 600m. WP weapon.
- Zeus – Heavy. Effective range 600m. Gold weapon. EXPENSIVE.

- Long range (700–1100 meters):
- Molot – Light. Effective range 800m. Silver weapon.
- Molot Mk2 – Medium. Effective range 800m. Silver weapon.
- Trebuchet – Heavy. Effective range 1100m. WP weapon.
- Gekko – Light. Effective range 1100m. Gold weapon. EXPENSIVE.

Special:

- Ancile – Heavy. Energy shield. Gold weapon. EXPENSIVE.
- Level-independent strategy
- Getting a 2nd, 3rd slot and 4th slot has priority over any robot/weapon purchases. Get them as soon as possible.
- Don't waste gold on anything (weapons/robot/WP/upgrades/paintjobs) until you have 5 slots.
- Do not buy paintjobs, WP, upgrade speedups ever.
- Don't waste too much upgrade time on robots or weapons that you won't use later in the game (Destrier, Gepard, Kang Dae).

Try to understand gameplay rules that favors you the most – is it a knife-fight (less than 400 meters), support-fight (400–800 meters), sniper (800+ meters) or a beacon run. Use appropriate bots for appropriate tasks. Do not get into a knife fight in a Cossack.

Do not mix weapon with different ranges.Know that weapon description in game often is misleading. Punishers are good for no further than 200 meters, and Molots are great in 300-700 meter range. Complete daily tasks - it gives you 60 to 80 gold per day. Do NOT complete daily tasks for WP; change it using FREE replacement points to get Gold tasks. So this is the work done by us in PHASE II.

VIII. CONCLUSION

Third person shooters are new phase in gaming world. With their advanced technology and sophisticated environment, TP Shooters gives a player thrilling experience. In every aspect it offers a great excitement with fast pace while playing. They give you the complete freedom to configure everything just the way you want it. Video Game Design programs provide students with educational skills beyond those utilized in the mechanics of developing a video game. The problem solving and critical thinking behind the design of game mechanics and game engine architecture can be used in a wide variety of fields including the game industry, financial sector, simulation engineering for data visualization, and many others. Video game design curriculums are theme oriented programs that attract students and impart engineering design knowledge to students who might not have considered engineering as a major.

IX. REFERENCES

1) Nate Garrelts, The Meaning and Culture of Grand Theft Auto: critical essays (McFarland, 2006), 159.
2) Anne-Marie Schleiner, "Does Lara Croft Wear Fake Polygons? Gender and Gender-Role Subversion in Computer Adventure Games" Leonardo Journal, Vol. 34, No. 3 (2001): 222.
3) Jonathan S. Harbour, Microsoft Visual Basic game programming with DirectX 2002
4) Rollings, Andrew; Ernest Adams (2006). Fundamentals of Game Design. Prentice Hall.
5) Geddes, Ryan, Beyond Gears of War 2, IGN, Sept 30, 2008, Accessed Apr 2, 2009
6) Blache, Fabian & Fielder, Lauren, History of Tomb Raider, GameSpot, Accessed Apr 1, 2009
7) Hutcheon, Linda, A Theory of Adaptation (CRC Press, 2006), pp. 55-56

8) Levi Buchanan (2006-11-10). "'Gears of War' is next-gen at its best". MSNBC. Retrieved 2009-03-02.
9) Ryan Donald (2002-08-27). "SOCOM: US Navy Seals (PlayStation 2)". CNET. Retrieved 2009-04-02.
10) François Dominic Laramée (2002). Game Design Perspectives. Charles River Media. ISBN 9781584500902.
11) Määttä, Aki, GDC 2002: Realistic Level Design in Max Payne, GamaSutra, May 8, 2002, Accessed Apr 6, 2009
12) "Halo Move to First-Person Shooter Confirmed". Inside Mac Games. 2001-03-15. Retrieved 2009-04-02.
13) Sal Accardo (2004-09-24). "Star Wars: Battlefront (PC)". GameSpy. Retrieved 2009-04-02.
14) Louis Bedigian (2002-11-23). "Metroid Prime Review". GameZone. Retrieved 2009-04-02.
15) Alexander R. Galloway. Gaming: essays on algorithmic culture (U of Minnesota Press, 2006), 60.
16) Jones, Steven E. (2008). The Meaning of Video Games: Gaming and Textual Strategies. Routledge. p. 83-84. ISBN 9781135902186. Clearly this early third-person shooter [Spacewar] paved the way for the FPS proper. The rockets are drawn on the screen against a 2-D backdrop of stars.
17) Voorhees, Gerald A.; Call, Joshua; Whitlock, Katie (2015). Guns, Grenades, and Grunts: First-Person Shooter Games. Bloomsbury. ISBN 9781441191441. Some of the earliest video games, such as the mainframe game Spacewar! (1962) and commercial games based on it like Galaxy Game (1971) and Computer Space (1971) also involved shooting . . . [T]hese games featured shooting from a third-person perspective.
18) Stanton, Rich (2015). A Brief History Of Video Games: From Atari to Xbox One. Little, Brown Book Group, Hachette Book Group. p. 114. ISBN 9781472118813. Radar Scope owed much to the popularity of Space Invaders and Galaxian, but nevertheless felt original thanks to its 3D third-person perspective.
19) Therrien, Carl (Dec 2015). "Inspecting Video Game Historiography Through Critical Lens: Etymology of the First-Person Shooter Genre". Game Studies. 15 (2). Retrieved October 16, 2017. [Tempest] corresponds to a third-person shooter, by contemporary standards.
20) Tube Panic at AllGame
21) Top 10 Sega Franchises That Deserve Platinum Treatment, GameZone
22) Xybots at AllGame.
23) Gerstmann, Jeff, Syphon Filter Review, GameSpot, Feb 12, 1999, Accessed Apr 1, 2009.
24) Rouse, Richard, Postmortem: The Game Design of Surreal's The Suffering, GamaSutra, June 9, 2004, Accessed Apr 1, 2009.
25) Kasavin, Greg, Max Payne Review, GameSpot, Dec 11, 2001, Accessed Apr 2, 2009.