Development of video game design using BLENDER and UNITY software

Mr.Azhagiri M[1],K.Srinivas[2],K.Vishal[3],SK.Mujasir[4]

SRM Institute of Science and Technology,
Ramapuram, Chennai- 600089,Tamil Nadu, India.

ABSTRACT

Using unity and blender software we are designing a 3D game model. For this game we are developing 3D interaction techniques for a robot shooting game. Our game is based on a category called as FPS(First Person Shooter). First-person shooter (FPS) is a video game genre centered around 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 robot models from the blender software and importing it into unity software were we are actually designing the game. For the game we are bringing c# script into the field. With all these introduction we are developing the game. For the overview of the game, the concept is , a robot will have to shoot his enemy robotsbefore his own life will come to an end. As in for our multiplayer game, we has to create servers. For the multiplayer game, there is a server, in that we have CLIENT. This client is the main part of the game. with this we are going to create a game.

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

I.INTRODUCTION

The game Shooter is a contemporary video game, and its objective is to offer the player a challenging and enjoyable experience in a robot shooting against game-controlled robot. Although the player's goal within the game is to win against the enemy robot, the AI techniques adopted in the game are primarily designed to give the player an enjoyable time on the shooter robot. In other words, the objective of winning by either side is not given the highest priority.This paper proceeds as follows. Section 2 provides some background literature relevant to the work. Section 3 presents the design of the software system of Shooter. Section 4 presents implementation of the game software system using some AI techniques. Section 5 gives a sample execution of the game system. Section 6 presents some discussion on the AI techniques adopted in shooting and its performance, and Section 7 includes the conclusion and some ideas on future directions of the work.

Students are introduced to theory behind video game design .The course will introduce theories of the serious game initiative, which covers education games and interactive simulations. Topics will include game prototyping, game testing, and game balance. With all these introduction we are going to bring our game on the field.

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 robot bot must make continuous adjustments to its robot's direction as opposed to simply shooting in a straight line and turning only when a curve in the arena approaches. To ensure the robot bot can mimic human behavior in this manner, some AI techniques were used for implementing the bots in Robots, which are explained in detail in the following section.C# for the programming language, unity software where we worked for the game , auto desking 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 coding part in the unity software.

NETOWORKING 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 lag 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 time step is 15ms, so 66.666... ticks per second are simulated, but mods can specify their own tick rate. 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 tick rate 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 tick rate with the –tick rate command line parameter, though tick rate changes done this way are not recommended because the mod may not work as designed if its tick rate is changed. *Note:* The tick rate 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 update rate (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 min rate and sv max rate (both in bytes/second). Also the snapshot rate can be restricted with sv min updaterate and sv max update rate (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 cmd rate. 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 full update 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 tick rate, 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 I

OVERVIEW OF THE PROJECT

This is the detailed part of the project. Basically our project name is BATTLE WHEELS. The name is proposed so because the game depends on the shooting. The FPS means first person shooter. First-person shooters are a type of three-dimensional shooter game, featuring a first-person point of view with which the player sees the action through the eyes of the player character. They are unlike third-person shooters, in which the player can see (usually from behind) the character they are controlling. The primary design element is combat, mainly involving firearms.

First person-shooter games are also often categorized as being distinct from light gun shooters, a similar genre with a first-person perspective which use light gun peripherals, in contrast to first-person shooters which use conventional input devices for movement.[6] A more important key difference is that first-person light-gun shooters like Virtual Cop often feature "on-rails" movement, whereas first-person shooters like Doom give the player more freedom to roam.

We are introducing multiplayer FPS. For this we have to develop our own servers. Multiplayer video game is a video game in which more than one person can play in the same game environment at the same time. Video games are often single-player activities, putting the player against preprogrammed challenges or AI-controlled opponents (which lack the flexibility of human thought). Multiplayer games allow players interaction with other individuals in partnership, competition or rivalry, providing them with social communication absent from single-player games. In

multiplayer games, players may compete against two (or more) human contestants, work cooperatively with a human partner to achieve a common goal, supervise other players' activity, co-op, and objective-based modes assaulting (or defending) a control point. Multiplayer games typically require players to share the resources of a single game system or use networking technology to play together over a greater distance.

For this multiplayer FPS 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:2.1

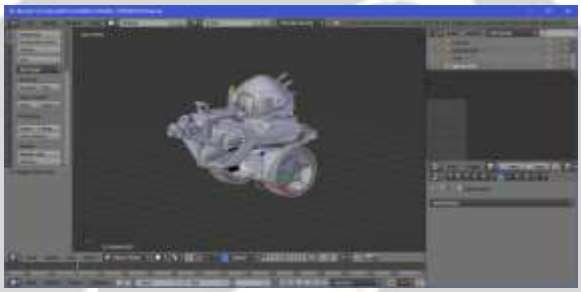

fig:2.2

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 modelling , 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 texturesand then we will upload colours for the models in substance painter. Our proposed model will look like:

fig:2.3

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:

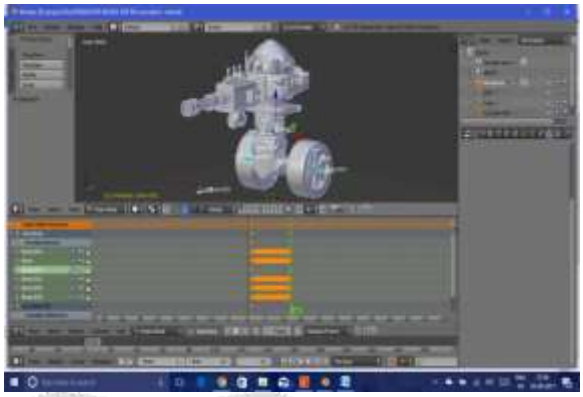

fig:2.4

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#.

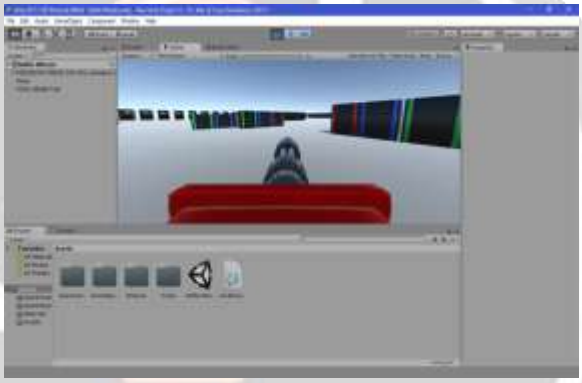

fig:2.5

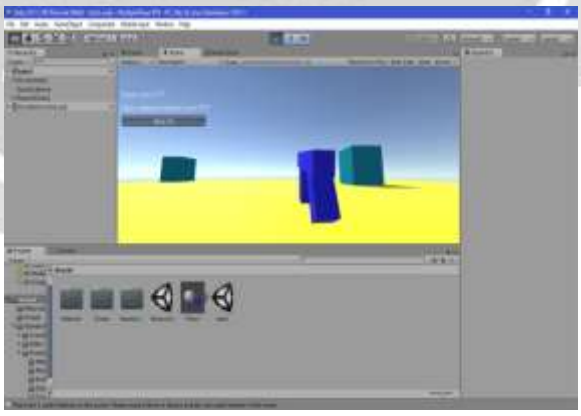

fig:2.6

For 3D games, Unity allows specification of texture compression and resolution settings for each platform that the game engine supports,[7] 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 robot 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 robot 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. This all has been introduced together in the unity software. And finally we will bring the game in a particular software where other players can join the game and collides with the robots (no. of players joined). In future, this game we will import this game as an android application so that all over world players can play this game. As it is a multiplayer game so we can add numerous players.

III.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 code craft. 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.

IV.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 FPS , this is the first person shooter. This is introduced in maximum games. The unique part in our game is the character as it is not a human model.

V.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 robots 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.

VI.CONCLUSION

The concept that video games are only fun and entertaining is to miss the creative and technical aspects of VGD necessaryto develop and play games. 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.

REFERENCES:-

[1] M. Cantu, E. Espinoza, R. Guo and J. Quarles, "Game cane: An assistive 3DUI for rehabilitation games," 2014 IEEE Symposium on 3D User Interfaces (3DUI),2014, pp. 43-46.

[2] Y. Zhu, Y. Wang, G. Forman and H. Wei, "Mining large-scale connectivity Refinement of Road Maps," in *The Computer Journal*, vol. 58, no. 9, pp. 2109-2119, Sep. 2015.

[3] P.Das, A. K. Sadhu, A. Konar, A. Lekova and A.K.Nagar, "Type 2 fuzzy induced person identification using Kinect sensor," *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Istanbul, 2015, pp. 1-8.

[4] V.B. Vasudevamurt and A. Uskov, "Serious games," 2015 I.E.E information Technology (EIT), Dekalb, IL, 2015, pp. 440-445.

[5] P. J. Diefenbach, "Practical Game Design andPedagogy,"in *IEEE Computer Graphics and Applications*, vol. 31, no.3,pp.84-88,May-June2011.

[6]"Asphalt Xtreme will be going off-road soon for Android racing fans". Android Authority. Retrieved February 5, 2015.

[7]"SuperCell's Clash of Clans finally hits the Google Play Store". Android Authority. Retrieved October 8, 2013.

[8]"Clash Royale: Now available EVERYWHERE!". Clash Royale. Retrieved October 8, 2013.

[9]"Dead Trigger". CNET. Retrieved October 8, 2013.

[10]"Dead Trigger 2 Hands-On Preview". Slide to Play. Retrieved March 25, 2013.