Implementation of Viewing and Networking Systems in a Military 3-Dimensional Environment

Firas Abdullah Thweny Al-Saedi[1], Fadi Khalid Ibrahim[2]

[1,2] *Computer Engineering Department, Al-Nahrain University, Baghdad, Iraq*

Abstract — *This paper discusses the viewing system for a 3-Dimensional (3D) military training environment. The viewing system is used to view the 3D environment from different aspects to make the user able to decide what action to take in the simulation according to the environment parameters. Also, the networking system for the environment is discussed, that is, another user can use another computer connected with a network to the first one to monitor the first user's performance. Also, the second user can choose a soldier from the user's soldiers team and participate in action along with the first user.*

Keywords — *3D, 3D camera, object follow camera, networking, multiplayer.*

I. Introduction

Before moving into the subject, the reader must know what is the Virtual Reality (VR), VR is a computer-simulated environment, whether that environment is a simulation of the real world or an imaginary world. Most current VR environments are primarily visual experiences, displayed either on a computer screen or through special or stereoscopic displays, but some simulations include additional sensory information, such as sound through speakers or headphones. Some advanced, haptic systems now include tactile information, generally known as force feedback, in medical and gaming applications. Users can interact with a virtual environment or a Virtual Artifact (VA) either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove, the Polhemus boom arm, and omni-directional treadmill. The simulated environment can be similar to the real world, for example, simulations for pilot or combat training, or it can differ significantly from reality, as in VR games. In practice, it is currently very difficult to create a high-fidelity virtual reality experience, due largely to technical limitations on processing power, image resolution and communication bandwidth. However, those limitations are expected to eventually be overcome as processor, imaging and data communication technologies become more powerful and cost-effective over time [1].

In this paper and to be cost-effective, Microsoft Visual C# 2008 [2] along with the new XNA 3.0 [3][4][5] graphics technology released by Microsoft are used. Actually, the graphics technology used is games-quality, this technology is used to generate a VR environment that is used individually or through network of two computers (this can be expanded easily). Also, the input device used is either the standard keyboard and mouse or using the new Nintendo Wii Remote (Wiimote) [6][7].

Another work on networking in VR environments can be found in [8], it reviews the techniques developed for improving networking in distributed interactive real-time applications. Also, techniques in online gaming can be found in [9].

In the next sections, the camera system along with the networking system will be discussed.

II. Camera

The camera is necessary in any 3D application, everything the user sees in the screen represents what the camera is seeing. Because of the scientific nature of the application discussed in this paper, a camera system is needed. The camera system is a system that controls how the camera will be positioned in the 3D environment, also, how the camera can avoid obstacles. Figure 1 illustrates how the camera is represented in the 3D environment.

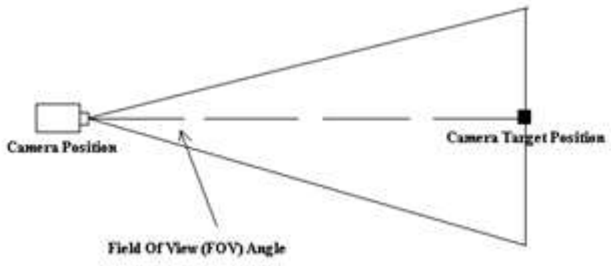

Fig. 1 The 3D camera field of view

Each of camera position and camera target position can be represented as a 3D coordinate (X,Y,Z). The Field Of View (FOV) angle can be represented in degrees or radians and its purpose is as what its name implies. For the scientific purpose of the environment, different camera views are needed. In the next subsections, each type of the camera view is discussed.

A. Free Camera

Because of the scientific nature of the environment, the user must be able to see every part of the environment and see the environment from any desired angle or height. For this purpose, the free camera is used. As what its name implies, it is a free camera that can be moved by the user to any part of environment. Figure 2 and Figure 3 show two different view of the environment taken using the free camera.

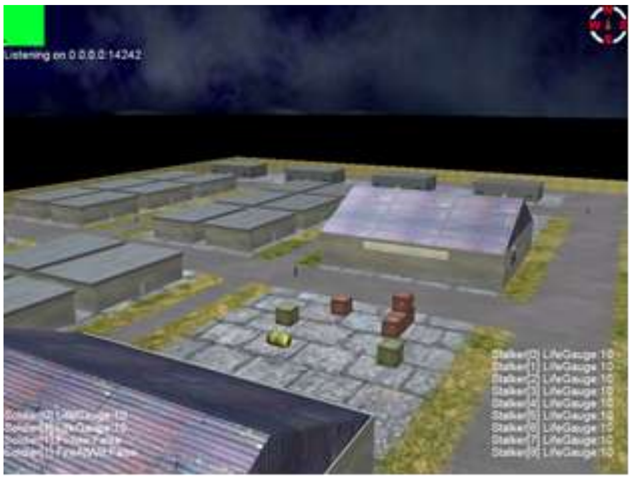

Fig. 2 A view taken using free camera

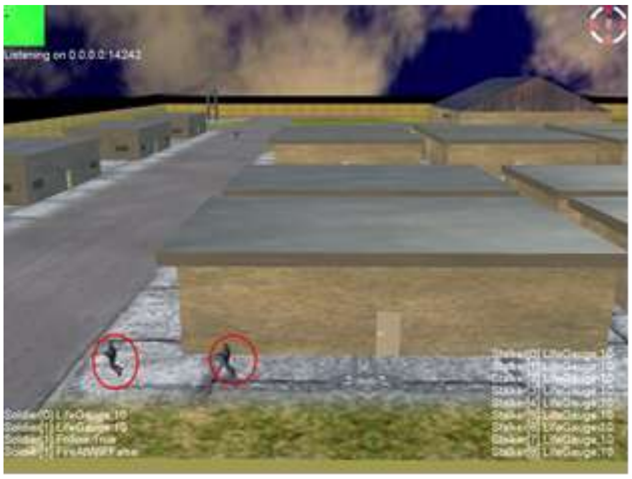

Fig. 3 A view (the user soldiers appear in action)

The free camera can be used by the user to check if there is a danger (enemy soldiers presence) in a place that he/she intends to send the team of soldiers to. For the camera system to be flexible, three actions are needed:

1. The camera must have the ability to turn left or right.
2. The camera must have the ability to look up and down.
3. The camera must have the ability to move towards the direction its now facing.

Figure 4 illustrates turn left/right and looking up/down actions.

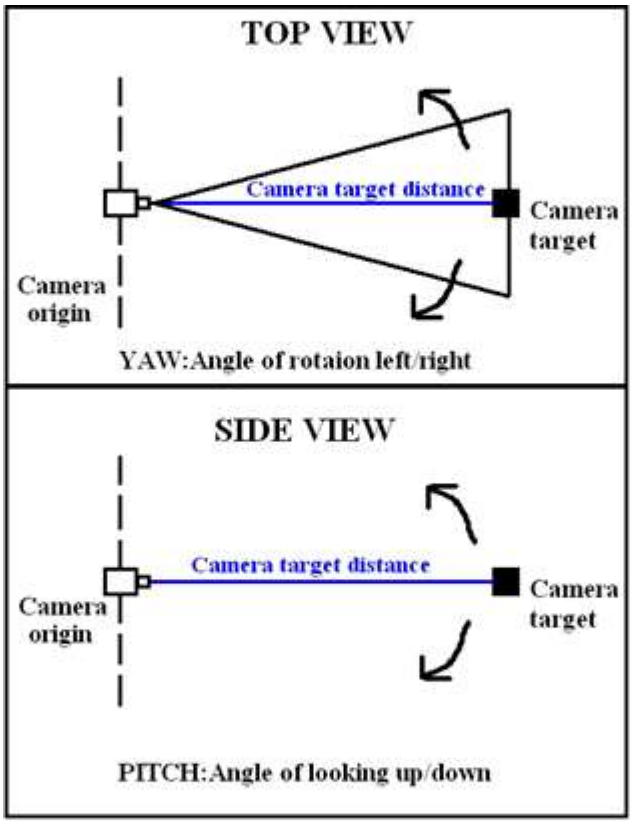

Fig. 4 The two basic actions of the free camera

The user will use the standard keyboard keys and mouse to move the camera in the environment. When the user presses the right/left key (or moving mouse left or right) the YAW angle will be updated (increases or decreases). Also, when pressing up/down keys (or moving mouse up or down) the PITCH angle will be updated (increased of decreased). After updating both angles, the new camera target position is calculated depending on the updated angles and the target moves to the new location. Also, when user presses a move forward key ('W' key on the simulation), the camera origin and target will move in the direction they are facing. Also, when pressing move backward key ('S' key on the simulation), the camera origin and target will move backward (i.e. in the opposite direction of the camera direction). Figure 5 illustrates the control system used to control the free camera movement.

The units used in the 3D environments are called the generic units. For the military simulation system, the environment is designed in a way that represents each meter by a 3.92 generic units.

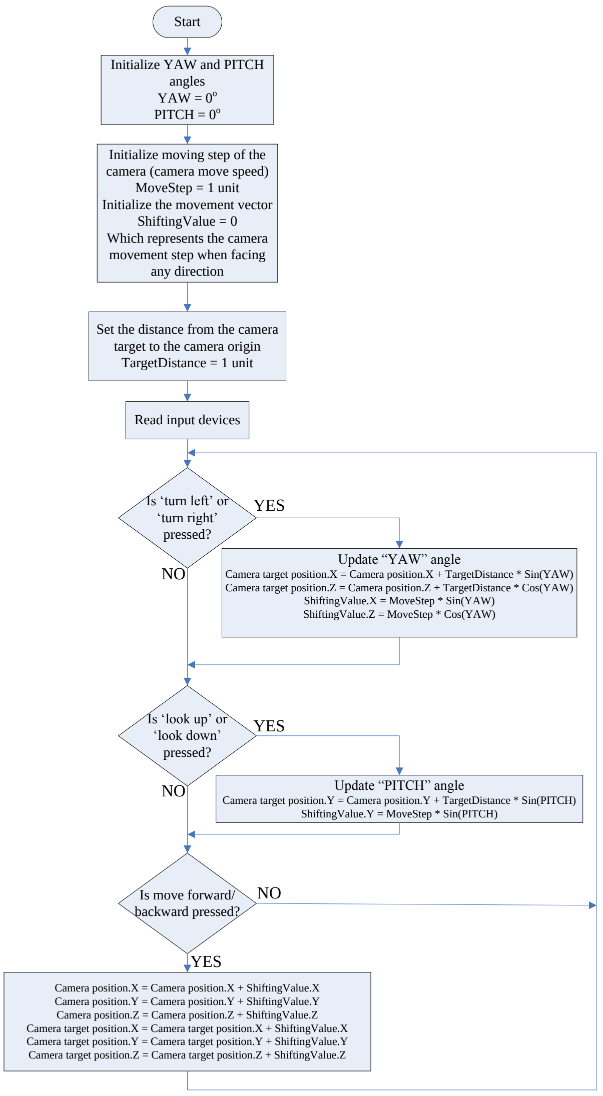
Start
Initialize YAW and PITCH angles
YAW = 0°
PITCH = 0°
Initialize moving step of the camera (camera move speed)
MoveStep = 1 unit
Initialize the movement vector
ShiftingValue = 0
Which represents the camera movement step when facing any direction
Set the distance from the camera target to the camera origin
TargetDistance = 1 unit
Read input devices
Is 'turn left' or 'turn right' pressed?
YES
NO
Update "YAW" angle
Camera target position.X = Camera position.X + TargetDistance * Sin(YAW)
Camera target position.Z = Camera position.Z + TargetDistance * Cos(YAW)
ShiftingValue.X = MoveStep * Sin(YAW)
ShiftingValue.Z = MoveStep * Cos(YAW)
Is 'look up' or 'look down' pressed?
YES
NO
Update "PITCH" angle
Camera target position.Y = Camera position.Y + TargetDistance * Sin(PITCH)
ShiftingValue.Y = MoveStep * Sin(PITCH)
Is move forward/ backward pressed?
NO
YES
Camera position.X = Camera position.X + ShiftingValue.X
Camera position.Y = Camera position.Y + ShiftingValue.Y
Camera position.Z = Camera position.Z + ShiftingValue.Z
Camera target position.X = Camera target position.X + ShiftingValue.X
Camera target position.Y = Camera target position.Y + ShiftingValue.Y
Camera target position.Z = Camera target position.Z + ShiftingValue.Z

Fig. 5 Control system of the free camera

B. Follow Camera

This camera is used when the user wants to control the soldier in action and use the shooting system. Shooting system for a user controlled soldier can be controlled from this camera mode. The reason beyond this is that the user will use the keyboard or the Wiimote to control soldier movement and use the mouse to control where to aim. Figure 6 is an illustration of this type of camera.

The camera origin will be in a point beyond the soldier always and its height will be in an appropriate height determined by testing what is the best height to make a clear view to the user. Figure 7 illustrates the follow camera mode in different views.

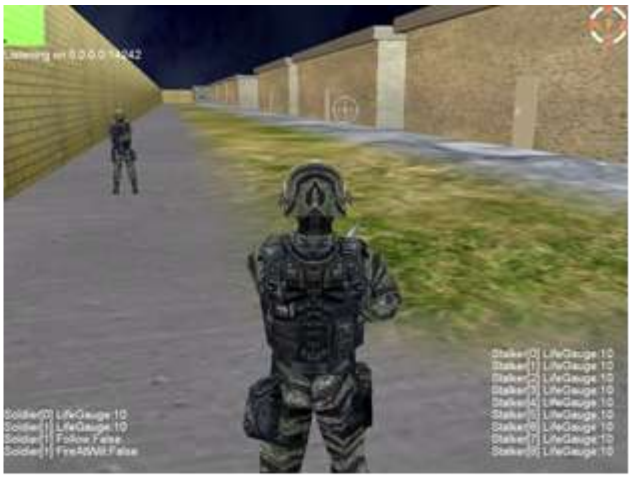

Fig. 6 Illustration of follow camera mode

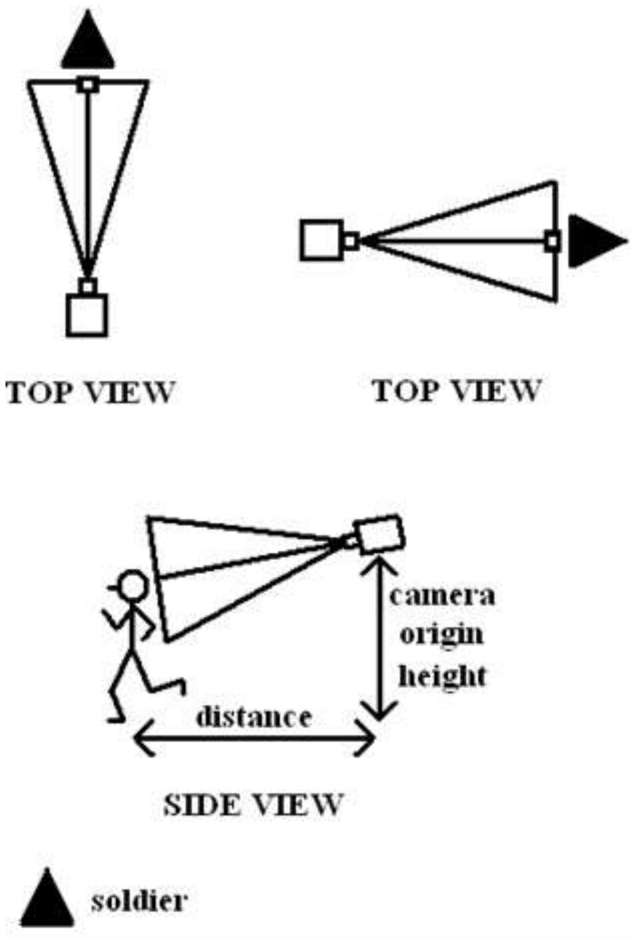

Fig. 7 Follow Camera mode in different views

As noticed, the follow camera origin is beyond the soldier and its height from the ground must equal the soldier's height plus a specific value that must be tested to see if the whole view is clear as shown in Figure 6; the soldier and the view in front of it are clear. The follow camera target must be at the soldier's X,Z position but its height must be modulated to get a clear view. Figure 8 illustrates the calculations done to make the camera always beyond the soldier.

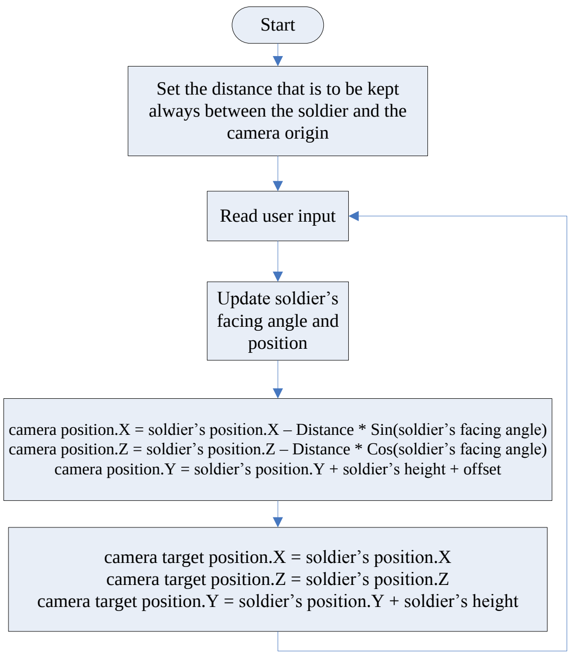
Start
Set the distance that is to be kept always between the soldier and the camera origin
Read user input
Update soldier's facing angle and position
camera position.X = soldier's position.X – Distance * Sin(soldier's facing angle)
camera position.Z = soldier's position.Z – Distance * Cos(soldier's facing angle)
camera position.Y = soldier's position.Y + soldier's height + offset
camera target position.X = soldier's position.X
camera target position.Z = soldier's position.Z
camera target position.Y = soldier's position.Y + soldier's height

Fig. 8 Follow camera system

C. Camera Collision Avoidance System

In some situations, in follow camera mode, the soldier may stand in a position that there is an obstacle beyond it. In this case, the camera origin will be inside that obstacle and its view will not show the soldier's back, instead of that, it will show what is inside the obstacle. Figure 9 shows the problem.

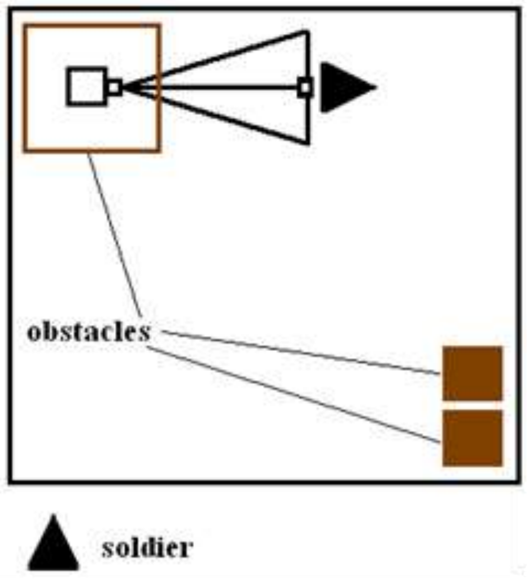

Fig. 9 Camera's origin inside an obstacle

To solve this problem, it is needed to reduce the distance between the camera origin and the soldier when the camera collides, then, getting back the original distance when no collision occurs. The camera movement during the position correction stated formerly must be very smooth (moving in a very small generic units like 0.02 unit) to keep the camera origin moving during the correction in a very smooth way. To know if the camera origin is colliding with any object, the camera origin must be bound with a bounding volume and continuously check its collision with other bounding volumes in the environment [10]. Figure 10 shows the system used to avoid the camera collision. It is important to note that this system is used only with the follow camera mode.

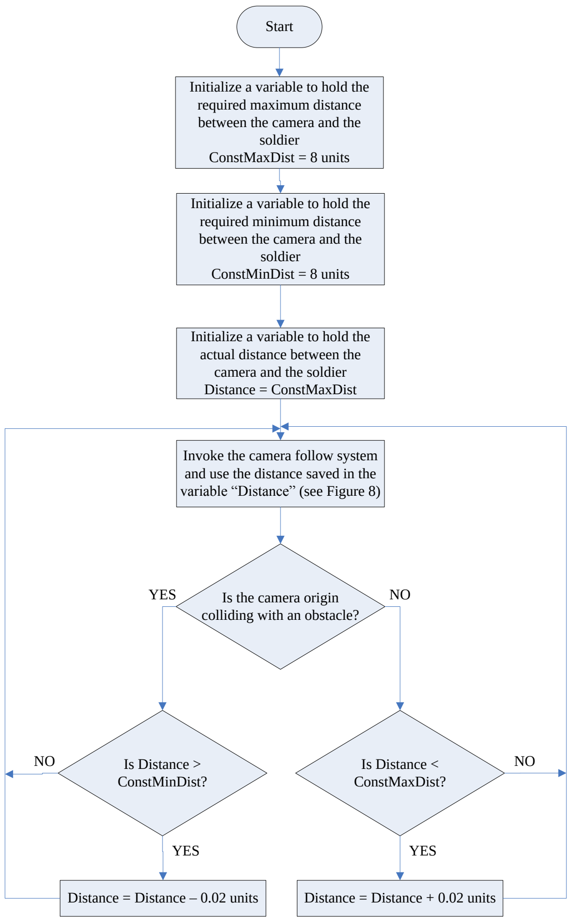
Start
Initialize a variable to hold the required maximum distance between the camera and the soldier
ConstMaxDist = 8 units
Initialize a variable to hold the required minimum distance between the camera and the soldier
ConstMinDist = 8 units
Initialize a variable to hold the actual distance between the camera and the soldier
Distance = ConstMaxDist
Invoke the camera follow system and use the distance saved in the variable "Distance" (see Figure 8)
YES
Is the camera origin colliding with an obstacle?
NO
NO
Is Distance > ConstMinDist?
YES
Is Distance < ConstMaxDist?
NO
YES
Distance = Distance – 0.02 units
Distance = Distance + 0.02 units

Fig. 10 Camera collision avoidance system

D. Eagle's Eye Camera

In order to be able to see a wider view of the environment, the user needs a top-view camera that can zoom to any part of the environment. For this the eagle's eye camera is used. It is view is similar to what a user gets from a satellite. Figure 11 illustrates the eagle's eye camera.

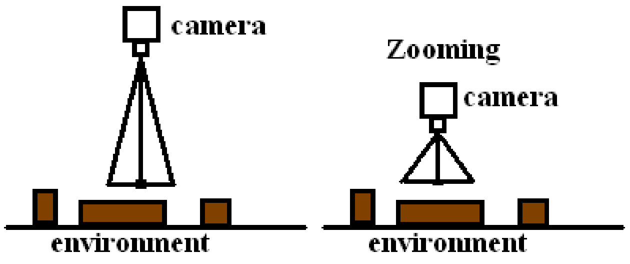

Fig. 11 Eagle's eye camera

This camera is designed to move left/right and up/down and zoom in and out to any part of the environment. For right/left move, the X-coordinate of the camera origin and target is changed while for the up/down move the Z-coordinate of the camera origin and target is changed. For zooming in and out the Y-coordinate of the camera origin only is changed as shown in Figure 11. Figure 12 and Figure 13 show two views taken using the eagle's eye camera from inside one of the simulation environments.

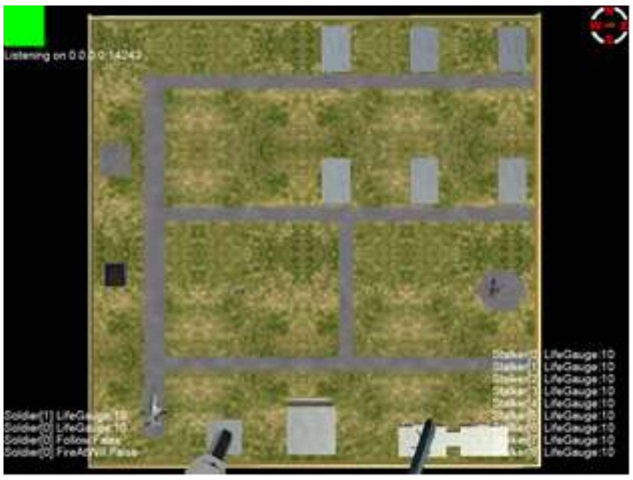

Fig. 12 eagle's eye view

Fig. 13 eagle's eye view (zooming in)

III. Networking System

As stated in the abstract that this paper is a part of a project that utilizes the 3D technology to create a 3D military environment that can be used for training. Two copies of the project have been designed: the first one, the server, can be used alone without the second one, the client, the user (trainee) can train by leading the team of two soldiers. The client version is used by connecting to the server and its purpose is to be used by the instructor to watch the performance of the trainee. Two computers must be connected using UTP cable and the server copy must be started first, then, the client copy is started. The server copy do all the calculations of collision, terrain height checking, path finding and AI, then, all the soldier's positions and states and other info are sent to the client, the client only draw the environment, set the soldiers in their received positions and states. The client does not do a lot of calculations, it only checks the input devices (keyboard and mouse) to change the camera view to let the instructor see what the trainee is doing from any desired view. The result is that the instructor can see what the trainee is doing in totally different perspective that the trainee is using, so the trainee might be using the follow camera, while the instructor is using the free camera and moving freely in the environment to evaluate the trainee performance in military tactics and squad strategies. Also, the instructor can participate in the action, and choose any of the two soldiers available in the user team at any time and even the instructor can go to the original mode of just monitoring and let the server computer AI control the other soldier that is not controlled by the trainee. In the next two subsections both network modes discussed above will be discussed in more detailed way.

A. Network Mode 0: Instructor Just Monitoring

In this mode, as discussed in Section (III), all the calculations are done by the server. The info needed by the client to draw a similar copy of the server's environment are sent via a UTP cable. The information needed to be sent can be classified into blocks and they are shown in Figure 14.

User Soldier Status Info:
-Position (X,Y,Z)
-Angle (YAW, PITCH)
-Soldier State (run, walk, dead)
-Life Gauge
-Is Alive (flag)
-Is InDoor (flag)
-FireAtWill (flag)
-FollowLeader(flag)

Enemy Soldier Status Info:
-Position (X,Y,Z)
-Angle (YAW, PITCH)
-Soldier State (run, walk, dead)
-Life Gauge
-Is Alive (flag)
-Is InDoor (flag)

Environment Status Info:
Elevator Y coordinate

Fig. 14 Sent info classified into blocks

Figure 15 illustrates the server program methodology.

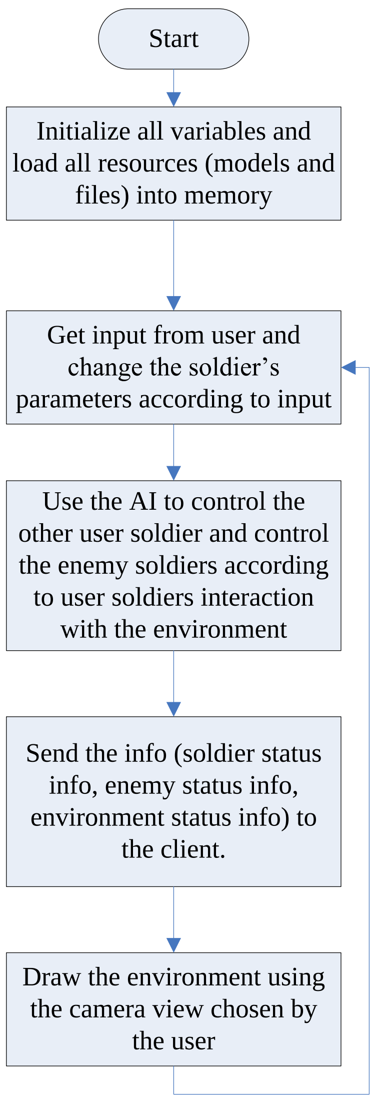
Start
Initialize all variables and load all resources (models and files) into memory
Get input from user and change the soldier's parameters according to input
Use the AI to control the other user soldier and control the enemy soldiers according to user soldiers interaction with the environment
Send the info (soldier status info, enemy status info, environment status info) to the client.
Draw the environment using the camera view chosen by the user

Fig. 15 Server program methodology (mode 0)

Figure 15 is a very simplified version of the server program and it is used just to take a top view of all the components of the project. The information about soldiers is sent in sequence, first the user soldiers info, then the environmental status info, then the enemy soldiers status info. Figure 16 shows the client program methodology.

It is noticed that in this network mode, the server only sends info and the client receives. In the next subsection, there will be a bidirectional info transfer.

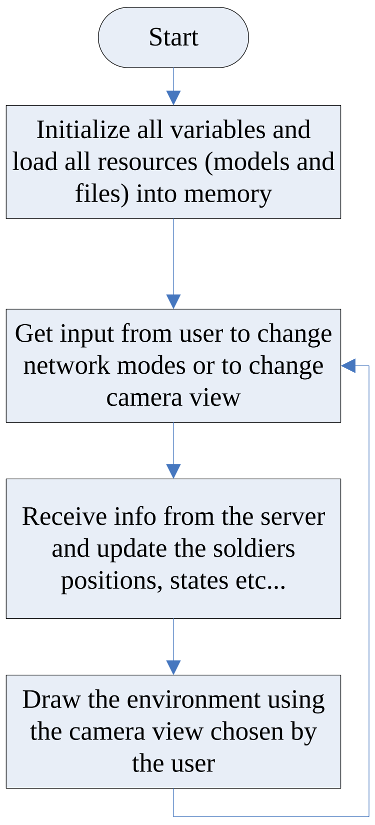
Start
Initialize all variables and load all resources (models and files) into memory
Get input from user to change network modes or to change camera view
Receive info from the server and update the soldiers positions, states etc...
Draw the environment using the camera view chosen by the user

Fig. 16 Client program methodology (mode 0)

B. Network Mode 1: Instructor and Trainee in Action

In this mode, the instructor can participate in action and control any soldier of the user team soldiers. When the instructor chooses a soldier, the trainee control will switch to the other soldier. The instructor has the priority in choosing any soldier. This was done to make the instructor able to control a soldier and show the trainee how to do a specific military tactic. Also, the instructor can switch to network mode 0, in this case, the trainee will control a soldier, and the computer AI will control the other. When network mode 1 is activated, the client sends the instructor soldier status information along with shooting ray origin and direction [11].

Those info are sent to the server, in the server, the instructor soldier status is updated based on the received status info and the received shooting ray info are checked whether it collides with an enemy to simulate the shooting process. Also, the other activities like terrain height checking and collision of the trainee soldier are calculated, and then the AI of the enemy soldiers is activated based on the moves of the user soldiers.

Then the trainee soldier status info along with environment status info and enemy soldiers status info are sent to the client to be able to draw the whole environment. Figure 17 shows the client program methodology in network mode 1.

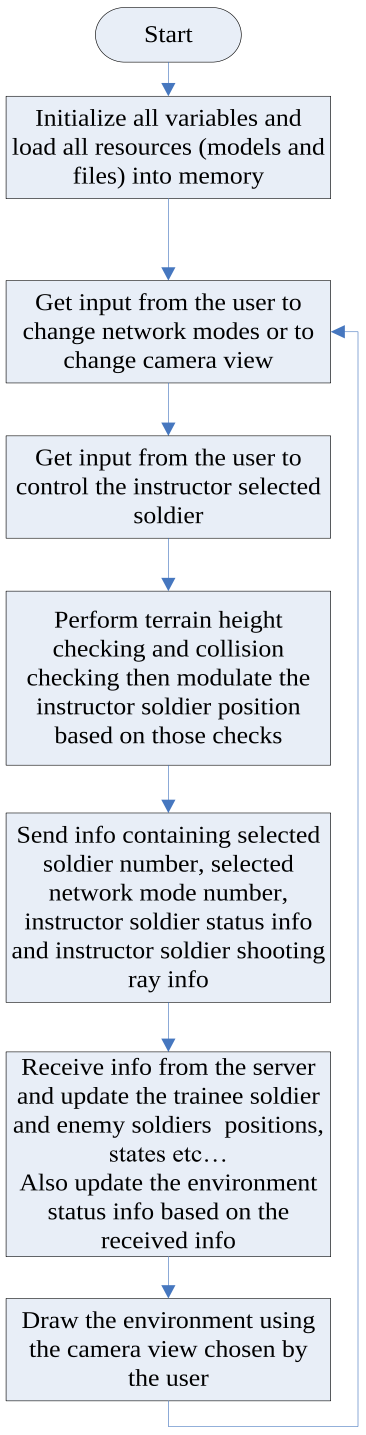
Start
Initialize all variables and load all resources (models and files) into memory
Get input from the user to change network modes or to change camera view
Get input from the user to control the instructor selected soldier
Perform terrain height checking and collision checking then modulate the instructor soldier position based on those checks
Send info containing selected soldier number, selected network mode number, instructor soldier status info and instructor soldier shooting ray info
Receive info from the server and update the trainee soldier and enemy soldiers positions, states etc…
Also update the environment status info based on the received info
Draw the environment using the camera view chosen by the user

Fig. 17 Client program methodology (mode 1)

As noticed from Figure 17, the client program in this case, performs the calculations related to the soldier selected by the instructor and sends the results to the server. Figure 18 shows the server program methodology in mode 1.

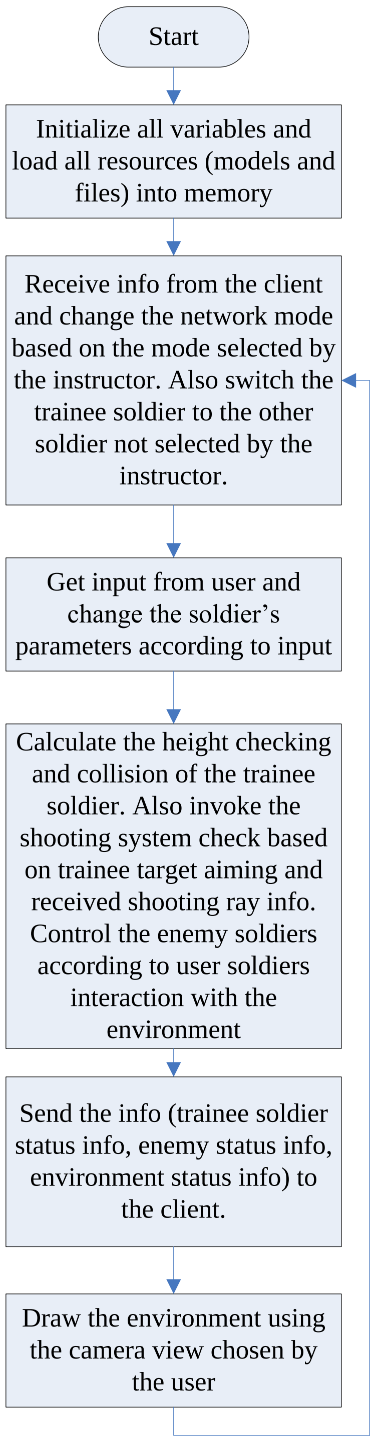
Start
Initialize all variables and load all resources (models and files) into memory
Receive info from the client and change the network mode based on the mode selected by the instructor. Also switch the trainee soldier to the other soldier not selected by the instructor.
Get input from user and change the soldier's parameters according to input
Calculate the height checking and collision of the trainee soldier. Also invoke the shooting system check based on trainee target aiming and received shooting ray info. Control the enemy soldiers according to user soldiers interaction with the environment
Send the info (trainee soldier status info, enemy status info, environment status info) to the client.
Draw the environment using the camera view chosen by the user

Fig. 18 Server program methodology (mode 1)

IV. CONCLUSIONS

The subjects discussed in this paper are parts of a project that simulates a military environment. The camera system is very important in a military environment. The user must be able to see any part of the environment to build decisions based on what he/she sees. Also, the networking system was discussed here. Its importance is in military trainings where the instructor will monitor the behavior of the trainee from totally different perspective. This can be thought as having two cameras in the same environment, one camera used by the trainee, the other camera used by the instructor.

REFERENCES

[1] en.wikipedia.org/wiki/Virtual_reality.

[2] Rob Miles, "C# Development", Department of Computer Sciences, University of HULL, October 2008.

[3] Aaron Reed, "Learning XNA 3.0", O'Reilly Media, 2009.

[4] Chad Carter. "Microsoft XNA Unleashed: Graphics and Game programming for XBOX360 and Windows", SAMS Publishing, 2008.

[5] Reimer Grootjans, "XNA 3.0 Game Programming Recipes: A Problem-Solution Approach", Apress, March 9, 2009.

[6] en.wikipedia.org/wiki/Wii.

[7] http://blogs.msdn.com/coding4fun/archive/ 2007/03/14/1879033.asp.

[8] Jouni Smed, Timo Kaukoranta and Harri Hakonen, "A Review on Networking and Multiplayer Computer Games", 2002.

[9] Marius Preda, Paulo Villegas, Franciso Morán, Gauthier Lafruit and Robert-Paul Berretty, "A model for adapting 3D graphics based on scalable coding, real-time simplification and remote rendering", 2008.

[10] Firas Abdullah Thweny, Fadi K. Ibrahim, "Implementation of Terrain Height Detection and Collision Check Systems in 3-Dimensional Environment", American Journal of Intelligent Systems, Vol. 4, No.4 , 2014.

[11] Firas Abdullah Thweny, Fadi K. Ibrahim, "Implementation of Sight and Shooting Systems with Rule-Based Artificial Intelligence in a Military 3-Dimensional Environment", IJCTT Journal Vol.29, No.2, 2015.