3D Tank Simulator

By

Khizar Hayat

A report submitted to Quaid-i-Azam University Islamabad
as a partial fulfillment of the requirements for the
Degree of M.Sc. in Computer Science

September 2004

QUAID-I-AZAM UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Dated:

FINAL APPROVAL

This is to certify that we have read the project report submitted by Mr. Khizar Hayat and it is our judgment that this report is of sufficient standard to warrant its acceptance by the Quaid-i-Azam University, Islamabad for the degree of Master of Science in Computer Science.

COMMITTEE:

1. External Examiner

   [illegible]

   Dr. A. F. M. Ishaq
   Director
   COMSATS Institute of Information Technology
   Sector H-8/1,
   Islamabad

2. Supervisor/Chairperson

   [illegible]

   Professor Dr. Farhana Shah
   Department of Computer Science
   Quaid-i-Azam University
   Islamabad.

To my loving Parents.

Project Brief

Project Title:	3D Tank Simulator
Undertaken By:	Khizar Hayat
Supervised By:	Dr. Farhana Shah (Chairperson)
	Department of Computer Science
	Quaid-i-Azam University, Islamabad
Starting Date:	February 2004
Completion Date:	September 2004
Tool Used:	Microsoft Visual C++ .NET
Operating System:	Windows XP
System Used:	Intel® Pentium® 4 CPU 1.70GHz.

Abstract

3D Tank Simulator is a 3D game engine. Initially 3D Tank Simulator is designed and implemented for research purposed. It is component of Complete Game Project. This is era of IT and Computer Based Training (CBT) is going to be used in almost every field of science. Learning through computer is becoming popular. So, MVRDE aimed to introduce CBT for training purpose. 3D Tank Simulator is first step towards Computer Based Training CBT in MVRDE organization.

Initially, I restrict the problem for simulation Tank in 3D virtual world and simulate the firing effects. I designed a model of Tank in 3D Studio Max which is exported as .x file. That .x file is used to load the tank in virtual world. To create virtual world I studied different techniques to create terrain from field of Computer Graphics. To bring the theme into reality I used Microsoft DirectX 9.0 APIs in MS Visual C++.Net environment.

Preface

This report represents description of Analysis, Design, Development, Implementation and Testing of "3D Tank Simulator". The entire work is summarized in chapters and some Appendixes.

Chapter-1	This chapter contains the material about the problem domain.
Chapter-2	This chapter describes problem to be solved, scope objective and feasibility of problem to be solved.
Chapter-3	This chapter concerns about requirement engineering, project plan resources to be used.
Chapter-4	This chapter contains items regarding analysis of system.
Chapter-5	This chapter has items regarding the design of system.
Chapter-6	This chapter includes documentation about implementation of system.
Chapter-7	This chapter has documentation about testing of the system.
Appendix-A	It contains the description about DirectX 9.0.
Appendix-B	This appendix contains Gantt Chart.
References	Reference of books used in this documentation.
Webliography	URL's of web sites, those were useful in development process.
Glossary	Glossary of terms used in this documentation.

Acknowledgement

I have no words at my command to express my deepest sense of gratitude to Almighty **ALLAH** who enabled me to complete my project against all odds and adversities.

Would that I have words to pay tribute to my loving parents whose invaluable prayers, productive advice and bolster attitude kept my spirit alive to strive for knowledge and integrity, which enabled me to reach this milestone.

It is with genuine pleasure that I acknowledge my debt to my teachers and internal supervisor **Dr Farhana Shah** who provided me personal stimulation, support and guidance. Her experience, searching questions and arguments have contribution in designing and developing system and writing this report
This task was not possible without the unforgettable and great co-operation and company of my class fellows, **Rebellions**. I will never forget all of them. I am especially thankful to M.Afzaal, Hassan Sadiq, and my room mate Nasir Hussain.

And last but not least, I would like to acknowledge the support of my family, especially my parents, my elder brother and sister Umer Hayat and Sidra and my younger brother M.Hayat, and my friend Asif Jamal, Taj Khan, Javeed khan, Nauman, M.Imran and Atif Jamal who love and perpetually pray for my success in my life and hereafter, and these prayers played a vital role in my success. I love them all.

In the end, once again I would like to thank **Dr Farhana Shah** for her nice co-oporation.

July 6, 2004. Khizar Hayat

Contents Page Number

Overview of System

1.1 Introduction to Organization

The project is defined by **Military Vehicles Research and Development Establishment (MVRDE)**. The major aim of the establishment of this organization is to develop the entire project related to Army, Air force and Navy for computer based training (CBT) and major confidential project. The software that I am going to develop is also a part of this activity. This would be implemented in enhanced form in major confidential projects of **MVRDE**.

1.2 Existing Systems

The already existing system is manual and due to lack of visual effects the exact system can not be easily understood by trainees. So there is need to simulate

- ➢ The simulation of movement of tank gunner.
- ➢ The control of movement of tank through arrow keys.
- ➢ The firing effects with keys as well with mouse.
- ➢ The sound effects of the scene have to display.
- ➢ Implementation of synthetic camera

1.3 Problem Definition

The task assigned is to simulate a tank gunner performing the firing effects while moving in 3D virtual world. In 3D virtual world, I have to simulate following requirements:

- ➢ The simulation of movement of tank.
- ➢ The control of movement of tank through arrow keys.
- ➢ The firing effects with keys as well with mouse.
- ➢ The sound effects of the scene have to display.
- ➢ Implementation of synthetic camera.

1.4 Scope

There are three major scenarios to discuss the scope of 3D Tank Simulator. Those are discussed below.

1.4.1 Context

3D Tank Simulator is a part of a comprehensive simulator that is being implemented in **MVRDE**. They are working in an environment in which the acting and frequently used operating system is Windows XP. 3D Tank Simulator must be built fit into a larger system. So the constraint imposed as a result of context on 3D Tank Simulator is that, 3D Tank Simulator must be compatible with Windows XP operating systems.

1.4.2 Information Objectives

In information objective the data objects produced as outputs of system and data objects, which are required as inputs of system are discussed.

1.4.2.1 Input

All inputs of the system are control inputs. The control inputs can be performed with arrow keys, function keys and alphabetic keys as well as with mouse. The movement of tank will be controlled with arrow keys and firing effects with alphabetic keys and mouse.

1.4.2.2 Output

Outputs of the system are movement of tank, firing effects and sound effects.

1.4.3 Functions and Performance

Functions are those tasks, which will be performed by our system in order to transform the input data to output. General functions that can be performed with our Simulator are following.

- Implementation of synthetic camera
- Firing effects
- Sound effects

1.5 Objectives

The objectives are those functionalities, which are required from the system or in other words those features in the system, which are required to fulfill the problem definition. In order to fulfill requirements of the Simulator we have the following objectives to achieve.

- The simulation of movement of tank.
- The control of movement of tank through arrow keys.
- The firing effects with keys as well with mouse.
- The sound effects of the scene have to display.
- Implementation of synthetic camera.

1.6 Recourses Identification

There is need of resources for doing any kind of job. Following resources are required to accomplish the development of this system.

1.6.1 Human Resources

The teams working on Simulator in MVRDE has maintained some software development metrics. These metrics indicates size, complexity and feasibility of my module. According to these metrics, complexity and size of the project indicates that effort required for this project is 5 person months.

1.6.2 Hardware Resources

Hardware resources required for the development are as follow.

Resource Name	**Minimal**	**Recommended**	**Description**
Processor	450 MHz	850 MHz	To run software required for development
RAM*	128 MB	256 MB	To run software required for development
Hard Drive	5GB	10GB	To run software required for development and store data.
Printer	Dot Matrix	HP DeskJet 600	To get hard copy of

			documentation for end user.
Keyboard	Standard	Standard	To give input to the computer
Mouse	Standard	Standard	To give input to the computer
Color Monitor	With resolution 800x600	With resolution 1024x768	To view out put.

1.6.3 Software Resources

Software resources required for the development are as follow.

Resource Name	Minimal	Recommended	Description
Operating System	MS Windows XP	MS Windows XP professional	To run the hardware.
Development Tool	Visual C++ .NET	Visual C++ .NET	To develop the system.
Documentation Tool	MS Office 97	MS Office XP or Star Office TM 5.2	To prepare the documentation.
Management Tool	MS Project 98	MS Project 2000	For analysis & Design
CASE Tool	Rational Rose 2.0	Rational Rose 5.0	For analysis & Design
Help Tool	MSDN library	MSDN library	For development help

1.7 Feasibility Study

I study the feasibility of the system in following four ways:

(1) Technical Feasibility

(2) Economical Feasibility

(3) Operational Feasibility

(4) Legal Feasibility

1.7.1 Technical Feasibility

A software project may be regarded as technically feasible or 'practical' if the organization has the necessary expertise and infrastructure to develop, install, operate and maintain the proposed system. System requires no other technical hardware than simple IBM Compatible system and from development tool point of view technical help is available from organization.

1.7.2 Economical Feasibility

Development of a system may be regarded as economically feasible or 'good value' to the organization if its anticipated benefits outweigh its estimated costs.

The cost of the application can be determined in term of following:

Software Cost

- Cost of Windows XP Professional
- Cost of Microsoft Visual C++.NET

Hardware Cost

- Cost of machines and resources identified in section

Development Cost

- It involves cost of developers and cost of training the developer.

As it is individual project so no cost involve in searching and hiring the developers. Organization has already Windows XP for which the system will be developed and they already have Microsoft Visual C++ .net as their development tool. The organization already has the required hardware components and machines. So, the project is economically feasible for the organization as no extra investment or resources are required for the development or implementation of this software project.

1.7.3 Operational Feasibility

A system development project is likely to be operationally feasible if it meets the 'needs' and expectations of the organization.

The proposed system is operationally feasible because development of the system will be done with collaboration of the organization (customer of the system) and final product will be according to the requirement of the organization. System will be user friendly and easy to understand, so average person would be able to understand and use this system easily. Thus the operational feasibility of the system is enough.

1.7.4 Legally Feasibility

Since the tools used for development of this project are purchased under license from Microsoft Corporation so it is also legally feasible

1.8 Paradigm Selection

The factors involved in the selection of software paradigm are project size, complexity, identification of requirements and time. Keeping in mind all these factors development has decided to follow **The Incremental Model**. It is very suitable when developing research project because team can analyze, design, code and test step by step, which is an excellent path to explore new domains.

Requirement Engineering

The hardest single part of building a software system is deciding what to build. No other part of the work so cripples the resulting system if done wrong.

2.1 Requirement Engineering

It is the set of activities that leads to the production of requirements definitions and specifications of system.

System requirement include the following sub headings

- Requirements Definitions
    a. Functional Requirements
    b. Non Functional Requirements
- Requirements Specifications.

2.1.1 Requirement Definition

Requirements definitions of Simulator are, what services the system is expected to provide and the constraints under which it must operate. Some of them are functional and some are non-functional which are defined separately in the subsequent sections.

2.1.1.1 Functional Requirements

- Creation of tank object in a 3D virtual world.
- Movement of tank with arrow keys
- Implementation of synthetic camera
- Firing effects
- Sound effects

2.1.1.2 Non-Functional Requirements

- The system should provide an attractive user interface for doing tasks.
- Response time of the system must be real time.
- System should be memory efficient.

2.1.2 Requirement Specification

In it we have specified each and every functional requirement defined above in a precise way using *From Based Method**.

Function:	Creation of tank object in a 3D virtual world
Description:	The tank object will be created in 3D studio max and will be exported in DirectX.
Inputs:	Color, size and dimension of tank object will be specified in 3D studio max.
Source:	The information will be entered by user.
Outputs:	A desired tank object will be created ready for loading.
Destination:	The tank object is then converted in .X file format and after creation it can be stored at HDD for further use.
Requires:	Nil
Pre-conditions:	Creation of 3D virtual world.
Post-conditions:	Creation of tank object.

Function:	Movement of tank with arrow keys.
Description:	Tank will move towards right with right arrow key, move towards left with left arrow key, move forward with up arrow key, and move back with down arrow key.
Inputs:	The user presses the left, right, up or down arrow key.
Source:	keyboard
Outputs:	A tank will move forward, backward, right, or left.
Destination:	Nil
Requires:	It requires tank object.
Pre-conditions:	Tank object should be loaded first.
Post-conditions:	Nil

Function:	Implementation of synthetic camera
Description:	With the help of synthetic camera we can see different views of scene.
Inputs:	The user moves the mouse.
Source:	mouse
Outputs:	A desired output will be different views of the scene.
Destination:	Nil.
Requires:	Nil
Pre-conditions:	Creation of 3D virtual world.
Post-conditions:	Nil

Function:	Sound Effects
Description:	After the bullets or bomb hit the target sound will be produce.
Inputs:	Nil
Source:	DirectX
Outputs:	A desired sound will be produce.
Destination:	Nil.
Requires:	Nil
Pre-conditions:	Nil
Post-conditions:	Nil

Function:	Firing Effects
Description:	After the tank is loaded user can fire bullet or bomb.
Inputs:	The user selects the gun and fire with the help of keyboard.
Source:	keyboard
Outputs:	A desired output will be bullets or bomb is seen hitting the target.
Destination:	Nil.
Requires:	It requires the tank.
Pre-conditions:	Tank should be loaded first.
Post-conditions:	Nil

System Analysis

For Analysis and Design purposes the Unified Modeling Language (UML) is used. In UML, a system is represented using five different "views" that describe the system from distinctly different perspective. Each view is defined by a set of diagrams. The following views are present in UML.

User Model View

The use-case is the modeling approach of choice for the user model view.

Structural model view

Data and functionality are viewed from inside the system. That is, static structure (classes, objects and relationship) is modeled.

Behavioral Model

This model represents the dynamic or behavioral aspects of the system. It also depicts the interactions or collaborations between various structural elements described in the user model and structural models view.

3.1 User Model View

3.1.1 Identification of Actor

Actor of this systems are (1) Game player

3.1.2 Use Case Identification

Use cases are represented graphically in a use case diagram to allow the analyst to visualize each use case in the context of other use cases in the system or subsystem to show its relationship with actors and other use cases. Use case names are text strings that contain letters, numbers and most punctuation marks except for colon, which is used to separate use case names from the name of packages, and it is good idea to keep them short. Use case names are normally made up of an active verb and a noun or noun phrase that concisely describe the behavior of the system that you are modeling. There is only one actor of the system, which is a game player.

3.1.3 Use Case Diagram

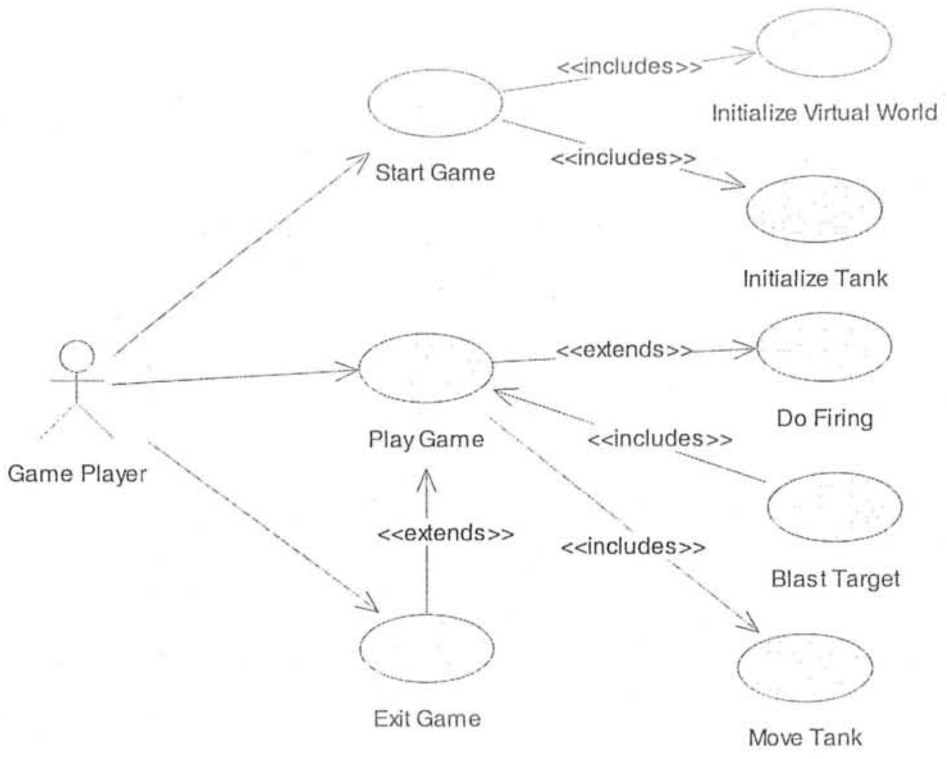
In use case diagram use case are drawn as an ellipse, the name of the use case usually written inside the ellipse, but can be placed beneath it. Do not mix these two styles in the same model.

3.1.4 Use Case

Following are the possible identified use cases of the system.

- Start game
- Initialize virtual world
- Initialize tank
- Play game
- Move tank
- Do firing
- Blast target
- Exit game

3.1.5 Use Diagram



(Figure 3.1)Use Case Diagram

3.1.6 Use Case Description (Use-Case Template)

Here is the description of the few use cases identified above.

3.1.6.1 Start Game

Use case:	Start game
Actors	Game Player
Trigger	User select the "start button"
Flow of events	
1 User will click the start button from main menu of application.	

3.1.6.2 Play game

Use case:	**Play game**
Actors	Game player
Trigger	User select the "Train ANN" option from menu
Flow of events	
1	User will click the train ANN option from main menu of application.
2	A dialog box will appear for taking inputs of Name of ANN for saving it to HDD.
3	An ANN will be trained and will be ready for recognizing.

3.1.6.3 Initialize virtual world

Use case:	**Initialize virtual world**
Actors	Game player
Trigger	User select the "start button" option from menu
Flow of events	
1	User will click the start button option from main menu of application.
2	Virtual world is created.

3.1.6.4 Initialize tank

Use case:	**Initialize tank**
Actors	Game player
Trigger	User selects the "start button" option from main menu of application.
Flow of events	
1	User will click the start button option from main menu of application.
2	.x file is loaded in the DirectX.

3.1.6.5 Do Firing

Use Case: Do Firing	
Actors	Game player
Trigger	User selects the “fire button” from keyboard
Flow of events	User will click fire button from keyboard.

3.1.6.6Move Tank

Use Case: Move Tank	
Actors	Game player
Trigger	User selects the arrow keys from the keyboard
Flow of events	1. With up arrow key tank will move forward 2. With down arrow key tank will move backward 3. With right arrow key tank will move right 4. With left arrow key tank will move left

3.2 Problem Definition

The task assigned is to simulate a tank gunner performing the firing effects while moving in 3D virtual world. In 3D virtual world, I have to simulate following requirements:

- The simulation of movement of tank.
- The control of movement of tank through arrow keys.
- The firing effects with keys as well with mouse.
- The sound effects of the scene have to display.
- Implementation of synthetic camera.

3.3 Static Model

Object oriented methodology provides information hiding, abstraction and modularity, by viewing a software system as a set of interconnecting data objects. These data objects have their own private status represented by a set of attributes (data members) and a set of member functions to change their state. The data objects communicate with one another through messages and by calling each other's member functions directly.

3.3.1 Identification of Classes

The most important classes of this system are as follow:

- CBase
- CGame
- CTerrian
- CMesh
- CFont
- CSound
- CPanel

3.3.1.1 Representation of Classes

A CRC model is used for the organized representation of the classes. The CRC cards are divided in three sections. Along the top of the card the name of the class is specified. In the body of the card responsibilities of class are listed on the left and the collaborators on the right.

Class Name: CBase	
Class Type: Property Class	
Class Characteristic:	
Responsibilities	**Collaboration**
• It generates the log file containing all information about the object.	

Class Name: CTerrian	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
It generates the terrain.	CBase

Class Name: CMesh	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
Load the tank.	CBase

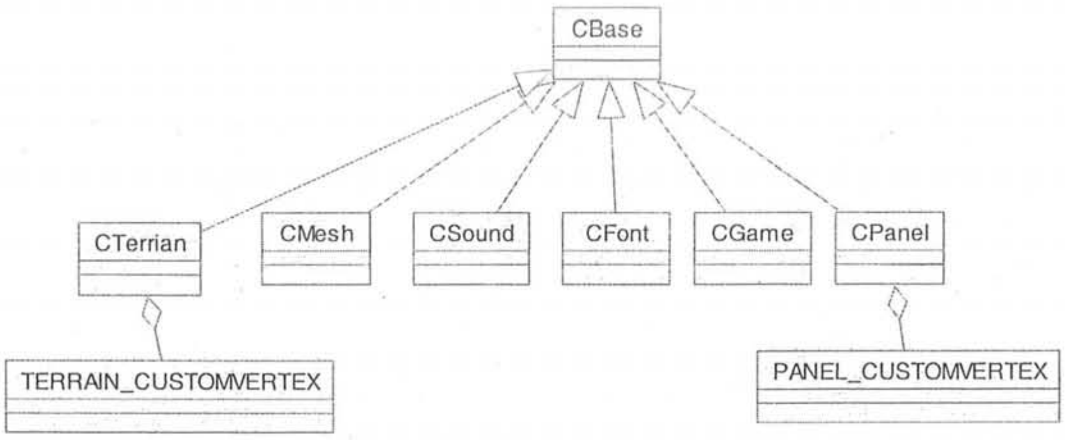
Class Name: CSound	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
Produce a sound.	CBase

Class Name: CFont	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
Displays the information.	CBase

Class Name: CPanel	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
	CBase

Class Name: CGame	
Class Type: Property Class	
Class Characteristic: Aggregate, Concurrent	
Responsibilities	**Collaboration**
All work is done in this class.	CBase

1.3.1.2 Class Diagram



(Figure 3.2) Class Diagram

3.3.1.3 Identification of Attributes and Methods

The following paragraphs briefly describe the specifications of the above-mentioned classes:

CBase

Class:	CBase	
Parent Class:	None	
Operations:	GetMemoryUsage()	DWORD (return type)
	GetTriangeNormal(D3DXVECTOR3* vVertex1, D3DXVECTOR3* vVertex2, D3DXVECTOR3* vVertex3)	D3DXVECTOR3 (return type)
	LogError(char *lpszText, ...)	void (return type)
	LogInfo(char *lpszText, ...)	void (return type)
	LogMemoryUsage()	void (return type)
	LogWarning(char *lpszText, ...)	void (return type)
	StartLogging()	void (return type)
	StopLogging()	void (return type)

CTerrian

Class:	CTerrian	
Parent Class:	CBase	
Attributes:	m_pD3DDevice	LPDIRECT3DDEVICE8(attribute type)
	m_pVertexBuffer	LPDIRECT3DVERTEXBUFFER8(attribute type)
	m_pTexture	LPDIRECT3DTEXTURE8(attribute type)
	m_matMaterial	D3DMATERIAL8(attribute type)

	m_pIndexBuffer	LPDIRECT3DINDEXBUFFER8(attribute type)
	m_wRows	WORD(attribute type)
	m_wCols	WORD(attribute type)
	m_wMaxHeight	WORD(attribute type)
	m_rTileSize	float (attribute type)
	m_dwNumOfVertices	WORD(attribute type)
	m_dwNumOfIndices	WORD(attribute type)
	m_dwNumOfPolygons	WORD(attribute type)
Operations:	CreateIndexBuffer()	bool (return type)
	CreateVertexBuffer()	bool (return type)
	CTerrain(LPDIRECT3DDEVICE8 pD3DDevice, WORD wRows, WORD wCols, float rTileSize, WORD wMaxHeight)	
	~CTerrain()	
	Render()	DWORD (return type)
	SetMaterial(D3DCOLORVALUE rgbaDiffuse, D3DCOLORVALUE rgbaAmbient, D3DCOLORVALUE rgbaSpecular, D3DCOLORVALUE rgbaEmissive, float rPower)	bool (return type)
	UpdateVertices()	bool (return type)

Class:	CMesh	
Parent Class:	CBase	
Attributes:	m_pD3DDevice	LPDIRECT3DDEVICE8 (attribute type)
	m_dwNumMaterials	DWORD (attribute type)
	m_pMesh	LPD3DXMESH (attribute type)
	m_pMeshMaterials	D3DMATERIAL8* (attribute type)
	m_pMeshTextures	LPDIRECT3DTEXTURE8*(attribute type)
Operations:	CMesh(LPDIRECT3DDEVICE8 pD3DDevice, LPSTR pFilename)	
	~CMesh()	
	Render()	void(return type)

CGame

Class:	CGame	
Parent Class:	CBase	
Attributes:	m_pD3D	LPDIRECT3D8 (attribute type)
	m_pD3DDevice	LPDIRECT3DDEVICE8 (attribute type)
	m_dwFrames	DWORD (attribute type)
	m_dwTotalPolygons	DWORD (attribute type)
	m_dwEndTime	DWORD (attribute type)
	m_dwStartTime	DWORD (attribute type)
	m_pMesh	CMesh* (attribute type)
Operations:	CGame()	
	~CGame()	

	CheckDisplayMode(UINT nWidth, UINT nHeight, UINT nDepth)	D3DFORMAT(return type)
	GameLoop()	void(return type)
	GetDevice()	LPDIRECT3DDEVICE8(return type)
	GetDevice()InitialiseD3D(HWND hWnd, UINT nWidth, UINT nHeight)	bool(return type)
	InitialiseGame()	bool(return type)
	Initialise(HWND hWnd, UINT nWidth, UINT nHeight)	Bool(return type)
	InitialiseLights()	bool(return type)
	Render()	Void(return type)
	SetupCamera()	void(return type)

CFont

Class:	CFont	
Parent Class:	CBase	
Attributes:	m_pD3DDevice	LPDIRECT3DDEVICE8(attribute type)
	m_pFont	LPD3DXFONT(attribute type)
Operations:	CFont(LPDIRECT3DDEVICE8 pD3DDevice, LPSTR pFontFace, int nHeight, bool fBold, bool fItalic, bool fUnderlined)	
	~CFont()	
	DrawText(LPSTR pText, int x, int y, D3DCOLOR rgbFontColour)	void(return type)

CSound

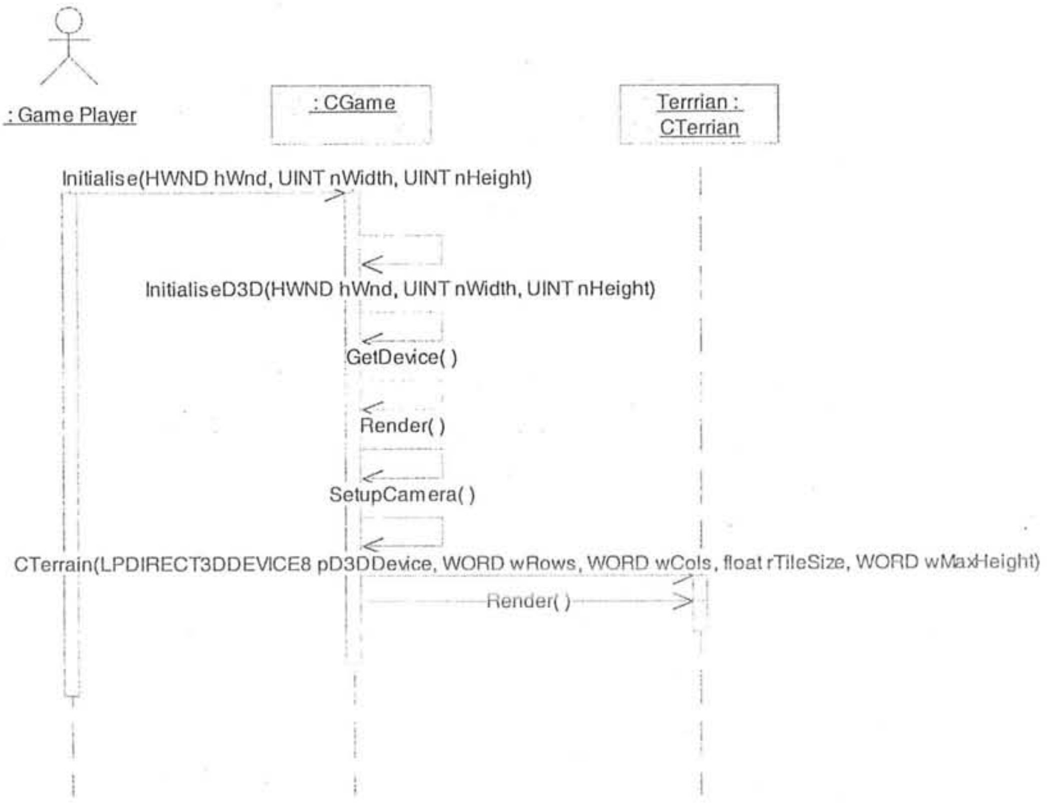
Class:	CSound	
Parent Class:	CBase	
Attributes:	m_pSegment	IDirectMusicSegment8*(attribute type)
	m_pDirectAudioPerformance	IDirectMusicPerformance8*(attribute type)
	m_pDirectAudioLoader	IDirectMusicLoader8*(attribute type)
	m_pGraph	IGraphBuilder*(attribute type)
	m_pMediaControl	IMediaControl*(attribute type)
	m_pMediaPosition	IMediaPosition*(attribute type)
	m_enumFormat	Format(attribute type)
Operations:	CSound()	
	~CSound()	
	InitialiseForMP3()	Void (return type)
	InitialiseForWavMidi(IDirectMusicPerformance8* pDirectAudioPerformance, IDirectMusicLoader8* pDirectAudioLoader)	void(return type)
	IsPlaying()	bool(return type)
	LoadSound(const char* szSoundFileName)	bool(return type)
	Play(DWORD dwNumOfRepeats)	bool (return type)
	Stop()	bool(return type)

CPanel

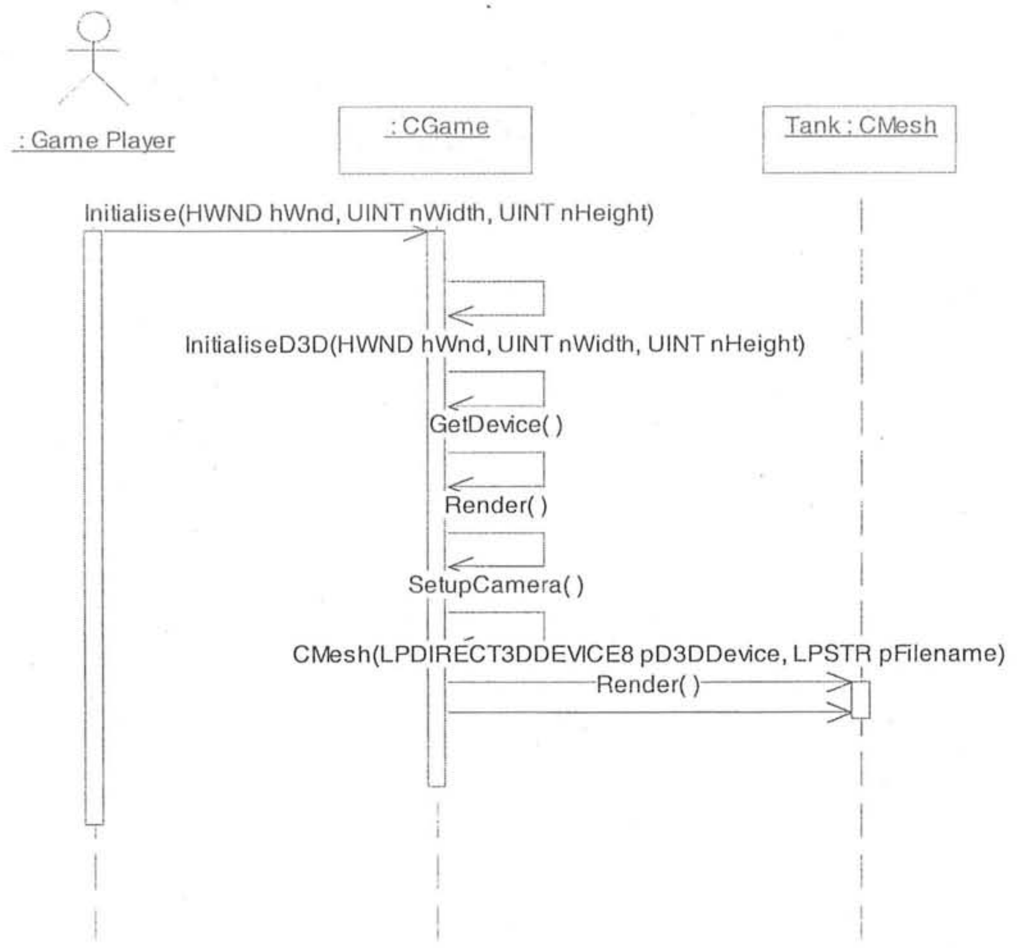
Class:	CPanel	
Parent Class:	CBase	
Attributes:	m_pD3DDevice	LPDIRECT3DDEVICE8(attribute type)
	m_pVertexBuffer	LPDIRECT3DVERTEXBUFFER8(attribute type)
	m_pTexture	LPDIRECT3DTEXTURE8 (attribute type)
	m_nWidth	Integer (attribute type)
	m_nHeight	Integer (attribute type)
	m_nScreenWidth	Integer (attribute type)
	m_nScreenHeight	Integer (attribute type)
	m_nX	Integer (attribute type)
	m_nY	Integer (attribute type)
Operations:	CPanel(LPDIRECT3DDEVICE8 pD3DDevice, int nWidth, int nHeight, int nScreenWidth, int nScreenHeight, DWORD dwColour)	
	~CPanel()	
	CreateVertexBuffer()	bool (return type)
	IsPointInsidePanel(int x, int y)	bool(return type)
	MoveTo(int x, int y)	void (return type)
	Render()	DWORD (return type)
	SetTexture(const char *szTextureFilePath, DWORD dwKeyColour) {	bool (return type)
	UpdateVertices()	bool (return type)

3.4 Behavioral Model

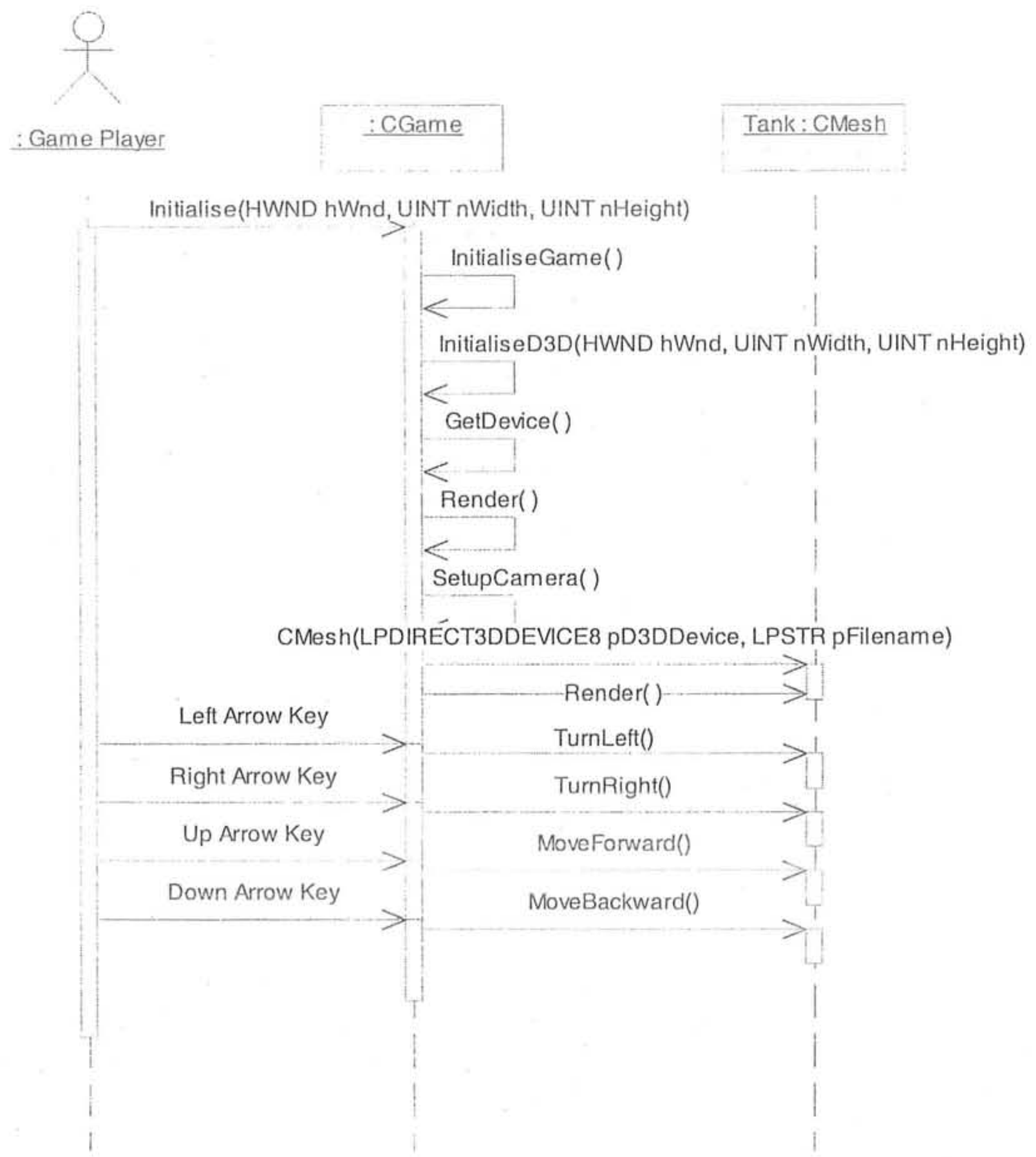
This model represents the dynamic or behavioral aspects of the system. It also depicts the interactions or collaborations between various structural elements described in the user model and structural models view.


: Game Player
: CGame
Terrrian : CTerrian
Initialise(HWND hWnd, UINT nWidth, UINT nHeight)
InitialiseD3D(HWND hWnd, UINT nWidth, UINT nHeight)
GetDevice( )
Render( )
SetupCamera( )
CTerrain(LPDIRECT3DDEVICE8 pD3DDevice, WORD wRows, WORD wCols, float rTileSize, WORD wMaxHeight)
Render( )

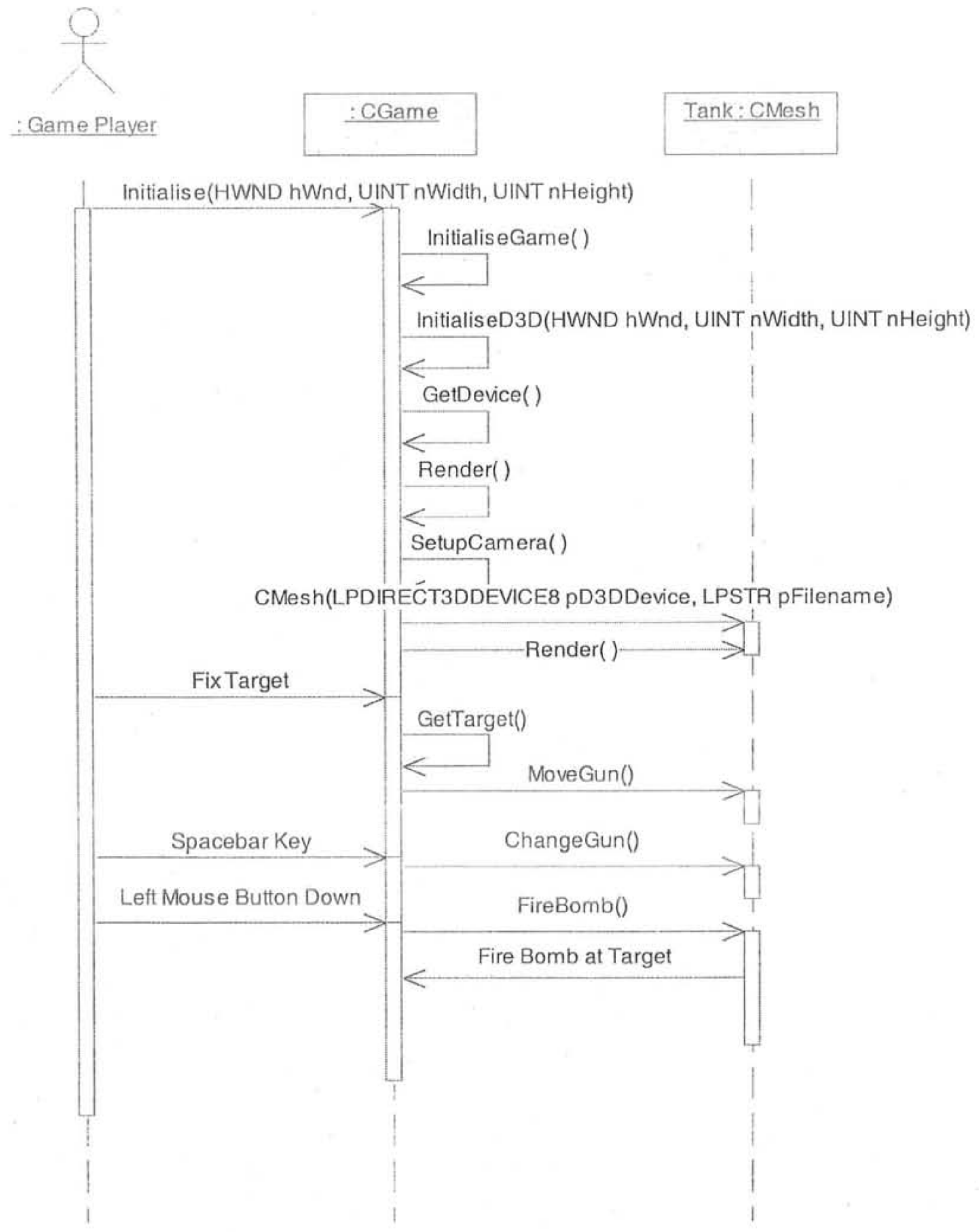
(Figure 3.3) Initialize Virtual World Sequence


: Game Player
: CGame
Tank : CMesh
Initialise(HWND hWnd, UINT nWidth, UINT nHeight)
InitialiseD3D(HWND hWnd, UINT nWidth, UINT nHeight)
GetDevice( )
Render( )
SetupCamera( )
CMesh(LPDIRECT3DDEVICE8 pD3DDevice, LPSTR pFilename)
Render( )

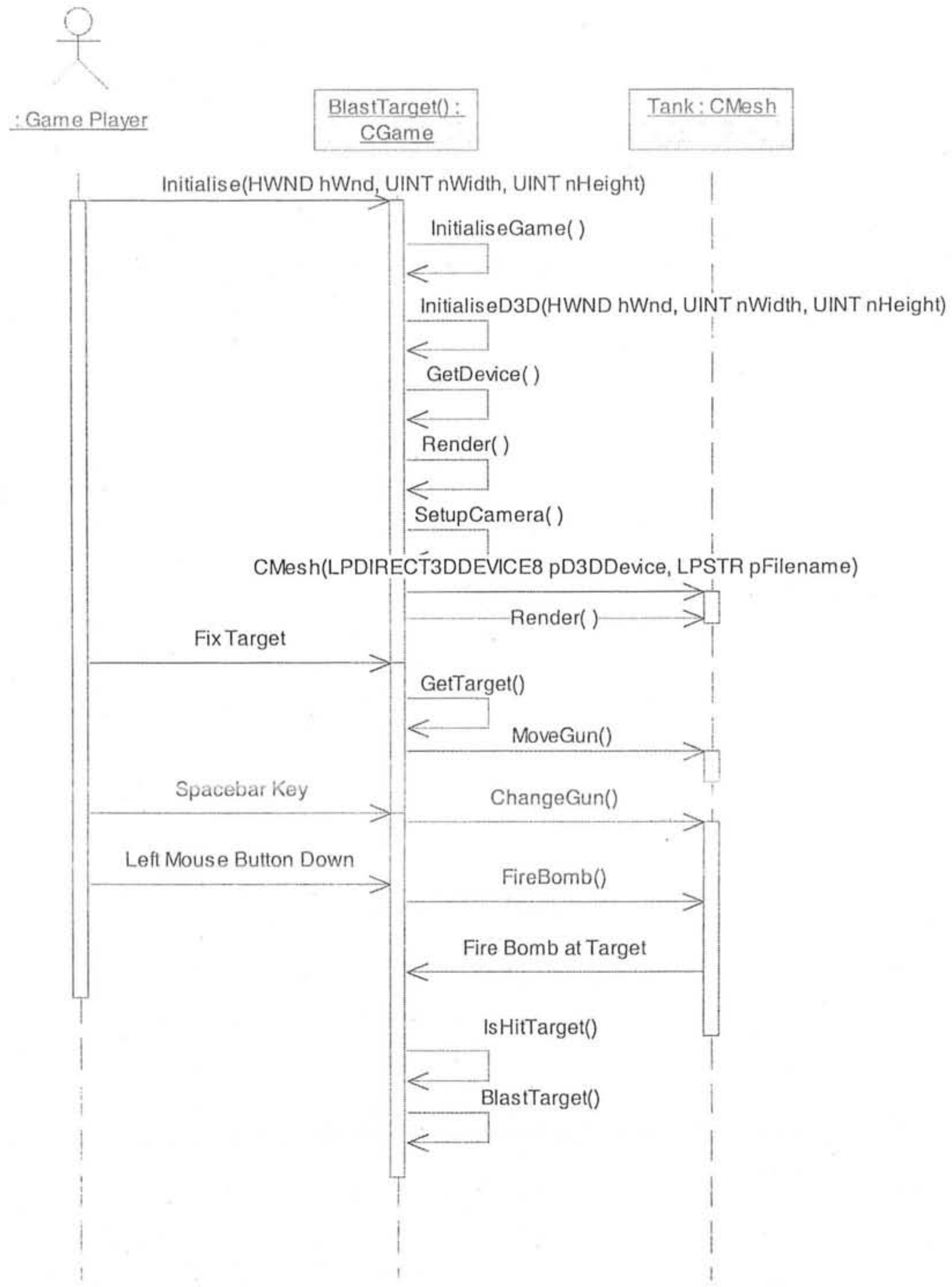
(Figure 3.4) Initialize Tank Sequence



(Figure 3.5) Move Tank Sequence



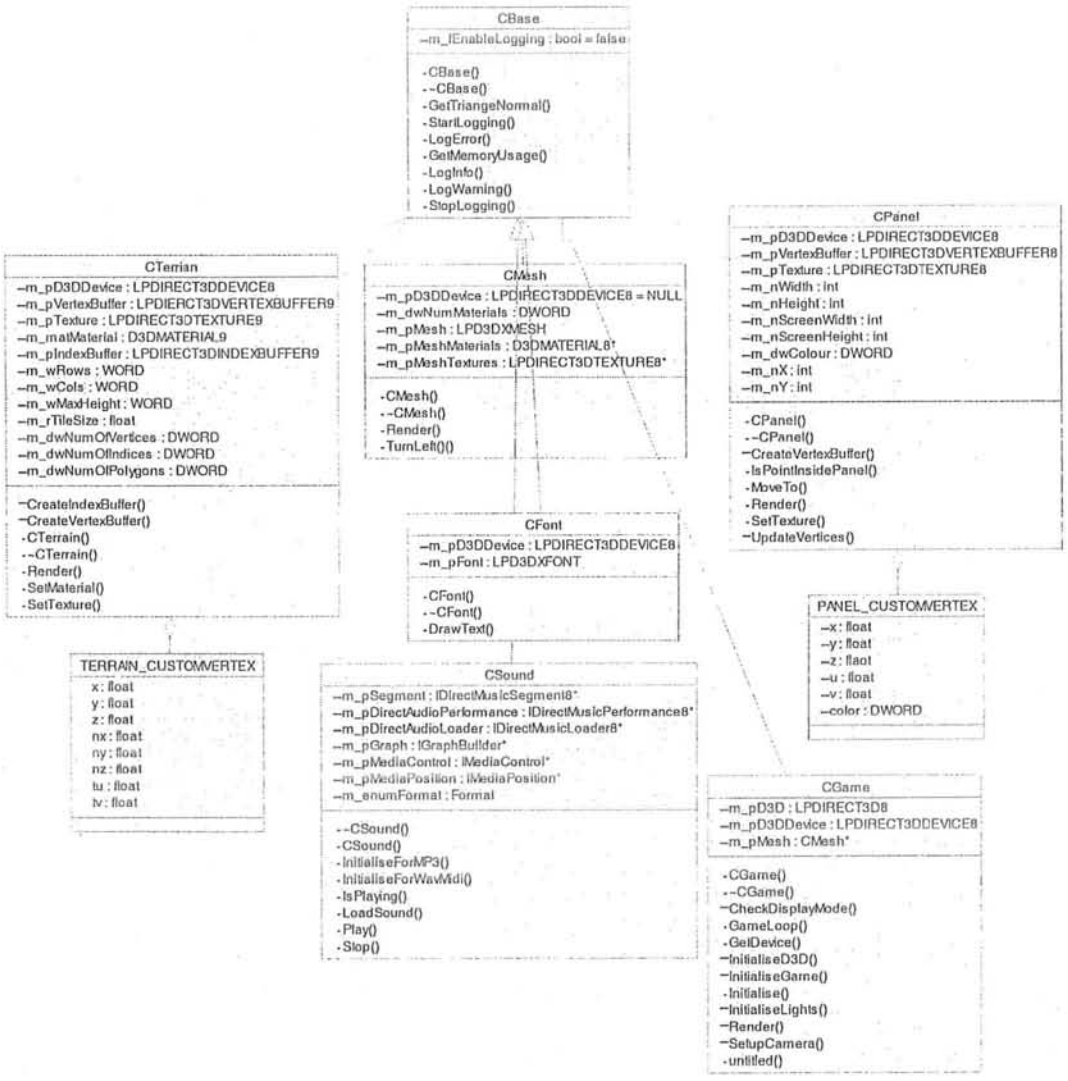
(Figure 3.6) Do Firing Sequence



(Figure 3.6) Blast Target Sequence

System Design

4.1 Detailed Class Diagram



(Figure 4.1)Detailed Class Diagram

4.2 DirectX Components

- **DirectX Graphics**

DirectX Graphics combines the Microsoft DirectDraw® and Microsoft Direct3D® components of previous DirectX versions into a single application programming interface (API) that you can use for all graphics programming. The component includes the Direct3D extensions (D3DX) utility library, which simplifies many graphics programming tasks. The interfaces use in developing this application are

- IDirect3D9
- IDirect3DDevice9
- IDirect3DTexture9
- IDirect3DVertexBuffer9

- **Microsoft DirectInput**

Microsoft DirectInput provides support for a variety of input devices, including full support for force-feedback technology. The interfaces use in developing this application are

- IDirectInput8
- IDirectInputDevice8
- IDirectInputEffect8

- **Microsoft DirectSound**

Microsoft DirectSound can be used in the development of high-performance audio applications that play and capture waveform audio. The interfaces use in developing this application are

- IDirectSound8
- IDirectSound3DBuffer8
- IDirect3DSoundListener8
- IDirectSoundCapture8

- **Microsoft DirectShow**

Microsoft DirectShow provides for high-quality capture and playback of multimedia streams.

- **Microsoft DirectMusic**

Microsoft DirectMusic provides a complete solution for both musical and non-musical soundtracks based on waveforms, MIDI sounds, or dynamic content authored in DirectMusic Producer.

System Implementation

5.1 Platform, Programming language and Tool Selection

Platform, programming language and related tools are selected according to nature and requirement of this project.

5.1.1 Platform Selection

Platform chosen for development of this system is Microsoft windows XP, the objectives and reasons selected as development platform as it is organization's standard and later the system will be implemented on the same operating system. So, it is better to use it at development time for easy enhancement and later compatibility.

5.1.2 Programming Language Selection

Language selection is very difficult job and it depends upon the requirements of the system to be developed. Visual languages provide a visual interface and many other facilities, which help the developer making an application, which are more users friendly and attractive. Programming language chosen for development of this project is VC++.NET and DirectX 9.0 This language is ultimate choice for this type of operating system related project. The selection is based on number of reasons.

- Visual C++.NET provides simple way to build user interface.
- Many constructs and utility classes used in System are already available in Visual C++.NET.
- It is requirement of the MVRDE to follow the Microsoft standard because they are solution provider of Microsoft products; Visual C++.NET supports Microsoft standards so it was selected as implementation tool.
- Visual C++.NET is object oriented language. As object oriented approach is used during analysis and design phase of software development so it was necessary to select an object oriented language.

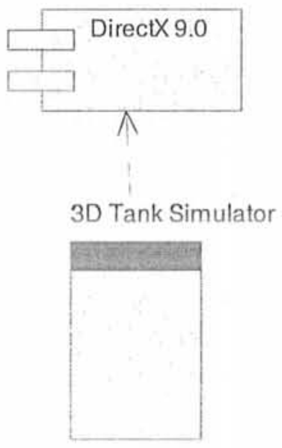
5.1.3 Code Documentation Standard

Code of project has been documented well with coding standard as told by external project in charge. These standards are described below,

- Each Class name begin with capital letter C and first letter of class name is also capitalized e.g. CGame class.
- Fully object oriented design is followed in implementation of this system. Some Classes are pure virtual classes that are only used as base classes for others e.g. CBase class.
- A general Design structure is used for the system, this design generalization will enable future enhancements, and other types of network interfaces can be easily incorporated in current system without much difficulty.
- Each Class is well documented in form of class documentation. Documentation of these classes is provided already.

5.2 Component Diagram

A component represents a physical piece of program code, either as source code or DLL.This application uses DirectX COM component (Direct Draw, Direct Sound)



(Figure 5.1) Component Diagram

System Testing

6.1 Objectives of Testing

Testing of the application is done to achieve following objectives.

- Execution of the program is done with intent to find the errors in the program.
- Test Cases are designed so that these have high probability of finding an as-yet-undiscovered error.

6.2 Boundary Value Analysis

Boundary value analysis is a test case design technique that complements equivalence partitioning. Rather than selecting any element of an equivalence class, BVA leads to the selection of test cases at the "edge" of the class.

The reason to use this technique is that a greater number of errors tend to occur at the boundaries of the input domain rather than in the "center". The test cases are designed so that these test cases satisfy the requirements of BVA also.

6.3 Test Cases

Test cases are designed so that these maps with following functional requirements of the system.

R.1	Creation of tank in a 3D virtual world.
R.2	Movement of tank with arrow keys
R.3	Implementation of synthetic camera
R.4	Firing effects
R.5	Sound effects

Req. No.	R.1
Date of Test	29/07/2004
Program	TankSimulator.dsp
Description	Creation of virtual world in which the tank is loaded to view game player.
Input	Selecting the Start Game Button creates the 3D virtual world.
Expected Output	3D virtual world will be created showing the earth and a tank

	on the earth.
Actual Output	3D virtual world is created showing the earth and a tank on the earth.
Test Conductor	Khizar Hayat

Result of Test: Test showed that desired result is being shown.

Req. No.	R.2
Date of Test	01/08/2004
Program	TankSimulator.dsp
Description	Tank will move with arrow keys. With up arrow key the tank will move forward, with down arrow key tank will move backward, and with left arrow key tank will move left, with right arrow key tank will move right .
Input	Selection of arrow key.
Expected Output	A tank will be seen moving.
Actual Output	A tank is seen moving.
Test Conductor	Khizar Hayat

Result of Test: Test showed that desired result is being shown.

Req. No.	R.3
Date of Test	01/08/2004
Program	TankSimulator.dsp
Description	Implementation of synthetic camera, with the help of camera we can see different views.
Input	Movement of mouse.
Expected Output	A view will be changed.
Actual Output	A view is seen changed.
Test Conductor	Khizar Hayat

Result of Test: Test showed that desired result is being shown.

Req. No.	R.4
Date of Test	02/08/2004
Program	TankSimulator.dsp
Description	Firing is done by the user of application.
Input	Selection of gun with space bar and pressing the left mouse button.
Expected Output	Bullets or bomb will be seen moving and hitting the target.
Actual Output	Bullets or bomb is seen moving and hitting the target.
Test Conductor	Khizar Hayat

Result of Test: Test showed that desired result is being attained

Req. No.	R.5
Date of Test	02/08/2004
Program	TankSimulator.dsp
Description	Sound effects are produced when the bullet or bomb hit the target.
Input	Bullets or bomb hitting the target.
Expected Output	Sound will be produced.
Actual Output	Sound is produced.
Test Conductor	Khizar Hayat

Result of Test: Test showed that desired result is being attained

Appendix-A

COM

What is COM? Well, the Component Object Model is basically a library of methods. You can create COM objects in your program and then call the methods that they expose to you. Methods are grouped together in collections of related methods. These collections are known as Interfaces. You could think of a COM object as a library of functions arranged by subject. DirectX provides a whole host of these libraries that will enable you to create 3D games. The best part is that DirectX takes care of a lot of the hard stuff for you.

Page Flipping

What is Page Flipping? Well, think of a flipbook. This is a number of pages with a slightly different drawing on each page. Then, when you hold the corner and "flip" the pages, it looks like the picture is moving. This is how DirectX Graphics works. You draw all of your objects onto a hidden page, known as the "Back Buffer". Then when you have finished, flip it to the Front Buffer and repeat the process. As the user is looking at the new front buffer, your program will be drawing onto the back buffer. What would happen without Page Flipping? Without Page Flipping, the user would see each object appear as it was drawn, which isn't what you want at all.

Vertex Buffers

A Vertex Buffer is a memory buffer for storing vertices. A vertex buffer can store vertices of any format. Once your vertices are stored in a vertex buffer you can perform operations such as: rendering, transforming and clipping.

Backface Culling

What is Backface Culling? Backface Culling is a pretty simple concept. Basically, it is a process where all of the polygons that are "facing" away from the user are not rendered.

For example, I have created a square with one side red and the other blue. Let's say that I defined the polygons so that the red side was "facing" the user and then started rotating the square. With Backface Culling enabled, the user would only see the red face and would never see the blue face.

Why is this useful? Well, if we are creating a closed 3D object (like a cube), we do not need to render the inside faces because they are never seen anyway. How do I specify which face is "facing" the user and which face to cull (not render)? It's all in the order that you specify your vertices. Below are two diagrams showing the order in which to define a "Clockwise" polygon. If you created a polygon in this way, the polygon would be rendered as shown, but if you were to flip the polygon (rotate), it would not be rendered. You can define which faces are culled, clockwise or anti-clockwise. By default, DirectX will cull anti-clockwise polygons.

Depth Buffers

Using a Depth Buffer (also called a z-buffer) ensures that polygons are rendered correctly based on their depth (distance from the camera). Let's say, for example, that in your scene you have two squares - one blue and one green. The blue one has a z value of 10 and the green square has a z value of 20 (the camera is at the origin). This means that the blue square is in front of the green one. A depth buffer is used to make sure that where one object is in front of another, the correct one is rendered. DirectX will test a pixel on the screen against an object to see how close it is to the camera. It stores this value in the depth buffer. It will then test the same pixel against the next object and compares it's distance with the value held in the depth buffer. If it is shorter, it'll overwrite the old value with the new one, otherwise it will be ignored (there is something in front of it). This will determine what colour the pixel will be, blue or green. Fig 4.1 below, shows this for a given pixel on the rendering surface.

What is a Texture?

Texture in 3D graphics is a 2D bitmap that can be applied to a polygon (or a number of polygons) to increase realism. For example, let's say that you want to have a brick wall in your scene. You could create a square object in front of the camera and colour it red. Hmm... That looks more like a red square than a brick wall, what you really want to see are the bricks and maybe a window. You can do these using textures. All you need is an object (your square) and a wall texture. You can create your texture in any art package that will save .bmp files, I use Adobe Photoshop but you can use any software you like, even Microsoft Paint that comes with Windows.

Appendix-B

WinMain

This is the applications entry point. Code execution will start here. This is where we register, create and show our window. Once that is complete, we initialise Direct3D and enter our game loop.

WinProc

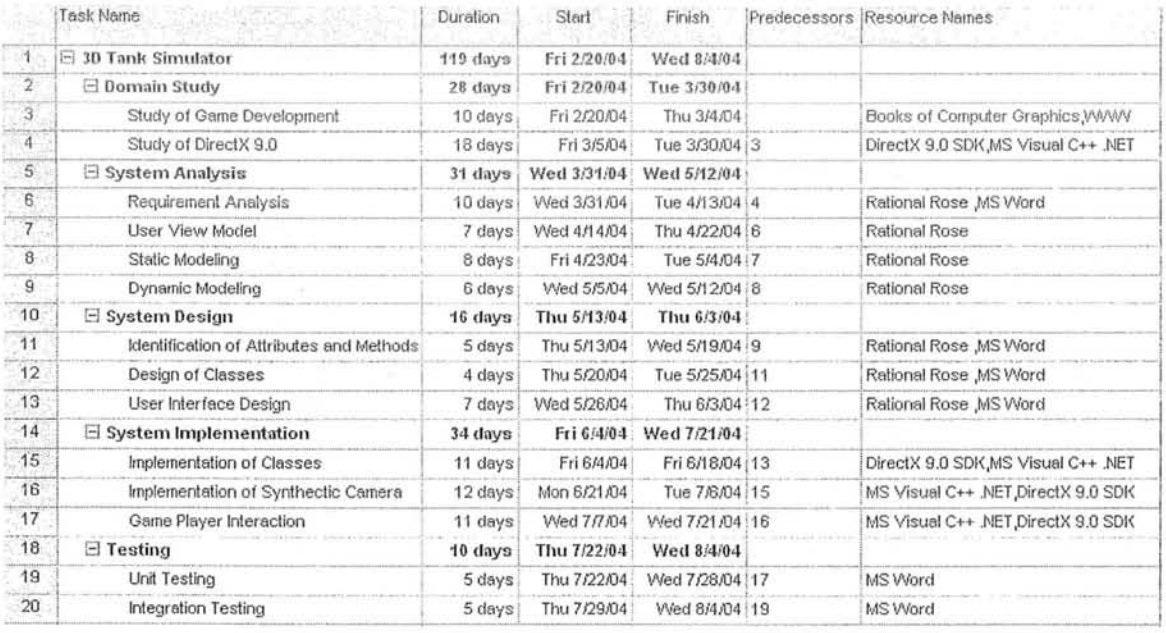
This is the applications message handler. Whenever Windows sends a message to our application, it will be handled by this function. Notice that there are two messages that our application will handle: WM_DESTROY and WM_KEYUP, all other messages are passed to DefWindowProc for default message processing.

Flexible Vertex Format (FVF)

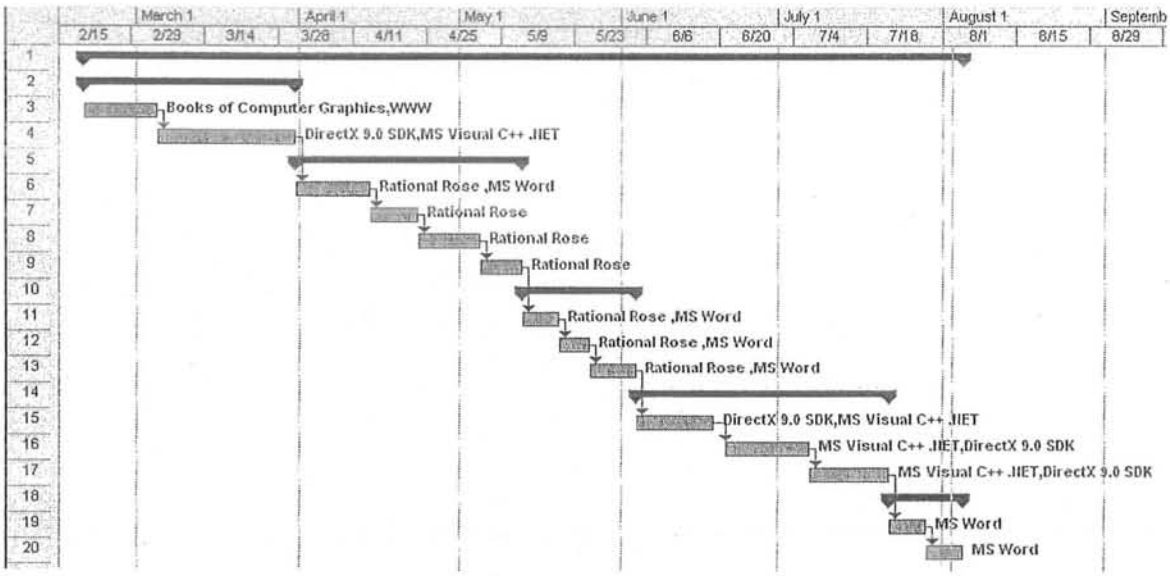
A Flexible Vertex Format or FVF is a format for describing attributes of a vertex. We've already seen three attributes: x value, y value and z value. There are other attributes that we can specify for a vertex, such as, color and shininess.

Using FVFs we can configure which attributes we want specify for a vertex. When you specify a polygon in Direct3D, the polygon can be filled based on attributes of the vertices. The fill of the polygon is interpolated (blended) between vertices. In our example below, you will see that the three vertices of our polygon are all different colors: red, green and blue. These colors are blended together across the polygon can satisfy certain constraints, which make them useful computational tools.

1 Project Gantt chart



	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	⊟ **3D Tank Simulator**	**119 days**	**Fri 2/20/04**	**Wed 8/4/04**		
2	⊟ **Domain Study**	**28 days**	**Fri 2/20/04**	**Tue 3/30/04**		
3	Study of Game Development	10 days	Fri 2/20/04	Thu 3/4/04		Books of Computer Graphics,WWW
4	Study of DirectX 9.0	18 days	Fri 3/5/04	Tue 3/30/04	3	DirectX 9.0 SDK,MS Visual C++ .NET
5	⊟ **System Analysis**	**31 days**	**Wed 3/31/04**	**Wed 5/12/04**		
6	Requirement Analysis	10 days	Wed 3/31/04	Tue 4/13/04	4	Rational Rose ,MS Word
7	User View Model	7 days	Wed 4/14/04	Thu 4/22/04	6	Rational Rose
8	Static Modeling	8 days	Fri 4/23/04	Tue 5/4/04	7	Rational Rose
9	Dynamic Modeling	6 days	Wed 5/5/04	Wed 5/12/04	8	Rational Rose
10	⊟ **System Design**	**16 days**	**Thu 5/13/04**	**Thu 6/3/04**		
11	Identification of Attributes and Methods	5 days	Thu 5/13/04	Wed 5/19/04	9	Rational Rose ,MS Word
12	Design of Classes	4 days	Thu 5/20/04	Tue 5/25/04	11	Rational Rose ,MS Word
13	User Interface Design	7 days	Wed 5/26/04	Thu 6/3/04	12	Rational Rose ,MS Word
14	⊟ **System Implementation**	**34 days**	**Fri 6/4/04**	**Wed 7/21/04**		
15	Implementation of Classes	11 days	Fri 6/4/04	Fri 6/18/04	13	DirectX 9.0 SDK,MS Visual C++ .NET
16	Implementation of Synthectic Camera	12 days	Mon 6/21/04	Tue 7/6/04	15	MS Visual C++ .NET,DirectX 9.0 SDK
17	Game Player Interaction	11 days	Wed 7/7/04	Wed 7/21/04	16	MS Visual C++ .NET,DirectX 9.0 SDK
18	⊟ **Testing**	**10 days**	**Thu 7/22/04**	**Wed 8/4/04**		
19	Unit Testing	5 days	Thu 7/22/04	Wed 7/28/04	17	MS Word
20	Integration Testing	5 days	Thu 7/29/04	Wed 8/4/04	19	MS Word



References

[ROG00] Pressman Roger S., "Software Engineering, A practitioner's Approach", IEEE Software, 2000.

[IAN98] Sommervillie Ian, "Software Engineering", Addison Wesley, 1998.

[BER99] Oestereich Bernd, "Developing Software with UML", Addison Wesley, 1999.

[CHA98] Chapman Davis, "Teach Yourself Visual C++ 6 in 21 Days", SAMS, 1998.

[TOT98] Toth Viktor, "Programming Windows 98/NT Unleashed", SAMS, 1998.

[STR99] Eric Stroo, "Desktop Applications with Microsoft Visual C++ .NET", Microsoft Press, 1999.

Webliography

http://msdn.microsoft.com/

http://www.andypike.com/

http://codeproject.com

http://cs.felk.cvut.cz/~neurony/neocog/en

http://ct.radiology.uiowa.edu/~jiangm/courses/dip/html/

http://ct.radiology.uiowa.edu/~jiangm/courses/mm-cv-ip/

http://www.citeseer.org

http://32bit.com/

Terms	Explanation
ARGB	Alpha, red, green, and blue components of a pixel.
DLS	Downloadable sounds. A standard for synthesizing wave sounds from digital samples stored in software. The DLS level 1 and level 2 standards are published by the MIDI Manufacturers Association
Front Buffer	Rectangle of memory that is translated by the graphics adapter and displayed on the monitor or other output device.
Mesh	Set of faces, each of which is described by a simple polygon
Scene	Entire set of objects that make up a virtual environment, including visible objects, sounds, lights, and frames. In Direct3D, the entire set of objects is contained in a root frame
SDK	Acronym for Software Development Kit
sRGB	Color space defined by Microsoft.
UML	Unified Modeling Language, These are notation used to describe system models of OOA & OOD etc.
Use Case	A scenario based methodology for election of requirements.
DLL	Dynamic Linked Library, used to wrap some functionality of code, which is dynamically linked during execution of software.
Test Cases	Test Cases are a set of inputs, which are used to check the system for its reliability and verification of code and validation of requirements, is done with the help of these.
Texture	Rectangular array of pixels that is applied to a visual object in Direct3D.