Pages of History

By

Maj Raja Ehsaan Elahi

Capt Muhammad Rizwan

Capt Adeel Shahid

Supervisor:

Lt Col Dr. Naveed Rao

Submitted to the Faculty of Computer Science

National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

June 2014

CERTIFICATE

Certified that the contents and form of project report entitled **“Pages of History”** submitted by 1) Capt Muhammad Rizwan, 2)CaptAdeelShahid, and 3)Maj Raja EhsaanElahi have been found satisfactory for the requirement of the degree.

Supervisor: _____

Lt Col Dr. Naveed Rao

ABSTRACT

The goal of this project was to develop an industrial standard game encompassing true events of history. This product would be a beginning of subsequent versions which will be developed in a later time frame. With a view to create a product fueled by the actions of bravery and valor of Muslim heroes, so as to counter all the propagandas of crusaders.

The product comprises of a strategy based scenario as baseline with prescribed target/task to be achieved (which will determine victory or defeat condition) vis-a-vis certain constraints in terms of resources, time and difficulty of the task to be performed. Hierarchical pawns (objects e.g. Horses, trebuchets, swordsmen etc.)Will be placed over, which will be either user or computer controlled. Game balance will be controlled through mathematical models. Object physics will also be controlled in real time by the game physics engine. On top of it HUD will act as the interface between user and game functionalities. HUD will also provide a dynamic feedback of the map and game play constituents.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To Our Parents and Wives for their prayers and support

and

To Google, which made it all possible

ACKNOWLEDGMENTS

We are grateful to our parents, families for their support and prayers. Their faith kept us going.

We are extremely grateful to our project supervisor Lt Col Dr.NaveedRao for his support and guidance without which we could not have moved on with the research. We are thankful to our teachers and instructors here at Computer science department, MCS, all of them have guided us have made it possible for us to complete both this degree and the project.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

a. Background

The era of information technology brings with it a plethora of entertainment packages, the greatest of which is the video game industry. This multi-billion dollar industry, has not only engulfed the entire globe &brought to life all scenarios, it is an avenue worth marching on. Over a passage of time the contents have been molded and shaped on depicting western society's culture and views. It is a great opportunity for a nation to not only put her share in this industry but to mold the tide to its own direction so as to bring forward their own historic events that actually happened in the history through time.

Keeping in light the above factors, It was strongly felt and needed to build and develop our own strategy based game, that could not only be used as a bastion of guidance for upcoming generations, but be an instrument having potentials to enhance their interest in Muslim history.

b. Problem Statement

Development of an industrial standard game encompassing true events of history. This product would be a beginning of subsequent versions which will be developed in a later time frame. With a view to create a product fueled by the actions of bravery and valor of Muslim heroes, so as to counter all the propagandas of crusaders. Keeping in mind the above and the fact that textual information is very sensitive and can cause damage to reputation, the team is working on obfuscating this information on the go. This application shall be installed on a web server and provide obfuscation at runtime.

c. Objectives

In pursuit of all above mentioned scenarios, the main goals our team wishes to accomplish are:

- A first of its kind commercial video game from Pakistan.
- A complete entertainment package for the youth.
- Cost Effective solution for teaching history.
- Counter for the western propaganda.

d. Deliverables

1st Progress Report: including SRS Document

2nd Progress Report: including System Design

Final Report: including complete documentation and demonstration

e. Technological Requirements

In order to run the game with optimal performance following specifications must be met following:

Recommended Requirements

- Standard Microsoft Windows 7 Operating System
- 3 GB free disk space recommended
- Graphics card recommended NVIDIA GeForce GTX 660
- 3 GB RAM

Minimum Requirements

- Standard Microsoft Windows Operating System (XP / Vista / 7 / 8)
- 2 GB free disk space recommended
- Graphics card recommended NVIDIA GeForce GTX 240
- 2 GB RAM

Chapter 2

Literature Review

a. Previous Work

Games like Crusaders and rise of nation are available. These games are also of same kind which are RTS (real time strategy).These games shows the western and biased views no doubt these are source of entertainment but still not good for the our new generation because in these games heroes are basically the westerns and in this way our new generation won't be able to recognize the our own heroes. Hero's from subcontinent and Islamicheroes so aim was to develop a game which can counter that stuff and showcase our own heroes.

b. Shortcomings

As of today in gaming industry, an extreme competition is seen in the market with regards to variety, entertainment, artificial intelligence and virtual reality. Although it provides a refreshing enjoyment yet beneath the line conveys a hidden message by the developers. This message derails our youth as they start idealizing for artificial and fake heroes. On a detailed study, it has been found as fully programmed propaganda by the enemies whereby generations are poisoned and thwarted from their actual heroes of the history to the ones created in the games by entertainment industry.

c. Issues solved by Pages of History

The aim of this project is to develop and demonstrate an industrial standard game encompassing true events of history. This game would be a beginning of subsequent versions which will be developed in the later time frame. With a view to create a parallel product teamed with the actions of bravery and valor by Muslim heroes of the history, so as to counter all the propagandas of crusaders.

The main objectives of the project include a real time strategy game that engulfs the player in an overhead map scenario. Main focus of these scenarios will be the events leading to fall of Mysore, (India 1776) revolving around the forces of Tipu Sultan and the British East Indian Allies. Endeavors will be made to construct a unique ambience pertinent to the subcontinent never before attempted.

In pursuit of all above mentioned scenarios, the main goals our team wishes to accomplish are:

- A first of its kind commercial video game from Pakistan.
- A complete entertainment package for the youth.
- Cost Effective solution for teaching history.
- Counter for the western propaganda.

Chapter 3

Design and Development

a. Introduction

This part of the document provides a detailed description of the system. The system shall allow a player to play a strategically video game.

b. Scope

The core deliverable of this project comprises of a game package that contains all third party and middleware solutions embedded. The package is a fully installable setup program that works on Microsoft Windows Environment and supports Windows XP/Vista/7/8. The main objectives of the project include a real time strategy game that engulfs the player in an overhead map scenario. Main focus of these scenarios will be the events leading to fall of Mysore, (India 1776) revolving around the forces of Tipu Sultan and the British East Indian Allies. Endeavors will be made to construct a unique ambience pertinent to the subcontinent never before attempted.

c. Product Perspective

The goal of this project is to develop an industrial standard game encompassing true events of history. This product would be a beginning of subsequent versions which will be developed in a later time frame. With a view to create a product fueled by the actions of bravery and valor of Muslim heroes, so as to counter all the propagandas of crusaders.

d. Product Functions

Real-Time Gameplay

Brief Description

The gameplay should mimic that of modern RTS games. Fully 3d environments adorned with key and background pawns should present an eye candy to the viewer. The animations should be blended by physics so as to minimize the distance between simulation and reality. Foliage and architectural details need to be captured and presented in most efficient manner; large scale exteriors should not be overloading the graphics hardware.

Normal Course

- User will start the game
- System will load the game
- User will select the campaign from the main menu
- System will load the user selected campaign
- User will select the option of start game
- System will load a gameplay menu for the adjustments of the control for the user
- User will select the controls according to his choice
- Game engine will make these control available for the user and will start the real time game play

Alternative Course

- If the user selects an inappropriate control for some campaign system will prompt and will redirect him to control menu.
- If the not available ground is selected system will redirect the user to its last selected place.
- If the exit button is pressed than game engine will direct the user to main menu.

Scenario-based Story Telling

Brief Description

At the start of every campaign scenario a high-fidelity narrative of the situation is to be presented to the user, elucidating the events and conditions that led to the given scenario. Actual timeframes and key events are to be used for this purpose; however the follow –up gameplay may be fictionalized.

Normal Course

- User will start the game
- System will load the game
- User will select the campaign from the main menu
- System will load the user selected campaign
- User will select the option of background story
- System will load the background story of the selected campaign

Alternate Course

- If the user wants to listen the background story of different campaign he has to exit this campaign and go back to main menu and select the other one.
- If the user wants skip the story he can press skip button at any time to move forward.

Strategy Elements with Player Inputs

Brief Description

The gameplay outcomes and experiences should be crafted as per the players input. Basic orders like attack, guard, move, etc. need to be easily coordinated using mouse (or other pointing mechanisms for further expansions). Units and key figures should be naturally differentiated for the player by means of their responses

Normal Course

- User will open the gameplay menu from the HUD of the GUI
- System will load the option related to the selected menu
- User will select the strategy elements from the menu
- Game engine will apply the changes on the game
- User will select the further options of the gameplay elements
- System will change the gameplay elements according to the user selection And those change will be displayed on the GUI and game as well

Alternate Course

- If the user selects an inappropriate control for some campaign system will prompt and will redirect him to control menu.
- User will get the prompt on wrong selection of the game play elements.
- If system on which game is running gets crashed next time game will be loaded from the point where user saved its progress last time.
- If the exit button is pressed than game engine will direct the user to main menu.

Selection of Difficulty level

Brief Description

It is basically the level of difficulty with which user wants to play. Game engine will provide him 3 different levels of difficulty Amateur, Hard and Professional.

Normal Course

- User will start the game
- System will load the game along all the required features.
- User will click the difficulty level button
- Game engine will display the options available in this menu
- User will select the difficulty level from this menu
- System will apply difficulty which is selected by the user

Alternate Course

- If user will not select any option by default Amateur level will be selected
- User cannot change the difficulty level during the campaign he has to restart the campaign.
- If the Back button is pressed than game engine will direct the user to main menu.

Placement of Troops on selected place

Brief Description

When a game user makes a strategy and places his army troops on some place. The complete process of selection of troops and the selection of the area as well

Basic Flow of Events

- User will on the selection mode from the HUD
- System will on the selection mode for the selection of events with mouse cursor
- With the help of Mouse cursor user will select the troops
- System will highlight them with dotted circle and mark them as selected troops
- User will move the cursor to the place where he wants to shift them and will click at that position
- System will move the selected troops to the selected position

Alternative Flows

- If user will not ON the selection mode system will not allow him to select the troops.
- If the user selects the location which is not available system will deselect the location.
- If the user stops the selection mode all the selections will be automatically deselected.

e. Assumptions and Dependencies

In the due course of development, it is assumed that

- UDK (the core software of construction in our project) will have the same price and features throughout the development phase.
- All open source soft wares included in our the development cycle will remain open for commercial use

- Close nip nap between PC and other consoles will remain the same and the demand for PC games will not be altered in the due course of time. Moreover new console rift will not be emerged in this timeframe.

f. Quality Attributes

Functionality

The product comprises of a strategy based scenario as baseline with prescribed target/task to be achieved (which will determine victory or defeat condition) vis-a-vis certain constraints in terms of resources, time and difficulty of the task to be performed. Hierarchical pawns (objects e.g. Horses, trebuchets, swordsmen etc.)Will be placed over, which will be either user or computer controlled. Game balance will be controlled through mathematical models. Object physics will also be controlled in real time by the game physics engine. On top of it HUD will act as the interface between user and game functionalities. HUD will also provide a dynamic feedback of the map and game play constituents.

Performance

The Product's smooth working is directly proportional to the system it is installed on, how much ram does it have, how many cores of CPU, memory etc.

Availability

The product shall be available as long as the system it is installed on is running.

Portability

This product is initially designed to run on windows environment.Android version of the game will be launched in future

Reusability

- The game will be expanded in future releases on other platforms as well, therefore all the core assets should be reusable in the future.
- Further expansions are also planned that will take the effort towards other Muslim conquests. All assets should be modifiable for such usage.

Testability

Different quality test can be performed on to the product so as to ensure that product is performing well and without faults.

g. Architectural model

System Block Diagram

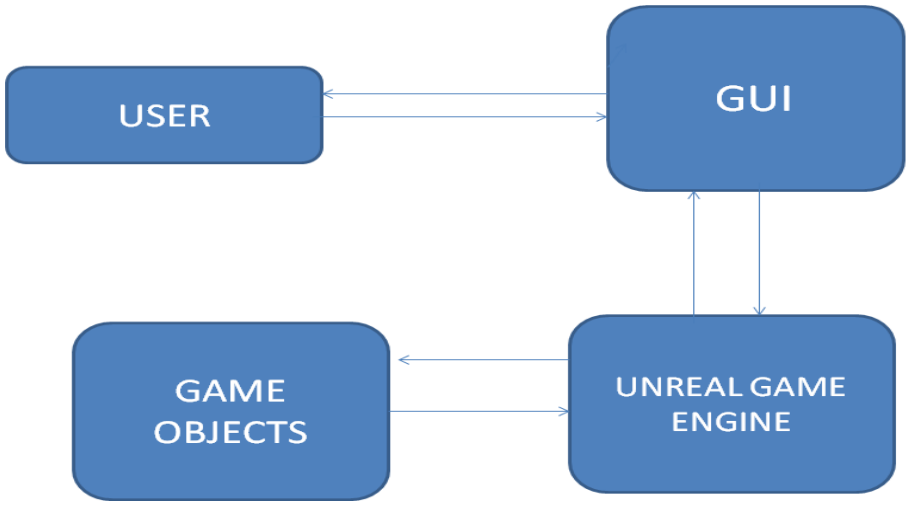

Figure 1:System Block diagram

Software Components

Our project contains following software components:

- Standard Microsoft Windows 7 Operating System
- UDK ENGINE DEVELOPMENT HOME
- Auto desk 3DS
- Maya
- NVIDIA Physics
- Epic Games
- Adobe Photoshop
- Adobe sound booth

Hardware Components

- 3 GB free disk space recommended

- Graphics card recommended NVIDIA GeForce GTX 660
- 3 GB RAM

Architectural Style

Architecture of Pages of History can be modeled using **Model-View-Controller** .Interface of the system is different from the Game Logic. Interface forms the View of the architecture. Model has the data .Controller controls the coordination between the view and the model. There is no direct communication between model and the view all the communication is directed using the controller. The system is divided into modules as per the main functionality of the system. These functionalities can be used separately as off-the shelf components .Interface of this system has no application logic embedded in it so the system architecture can easily be made using Model View Controller approach.

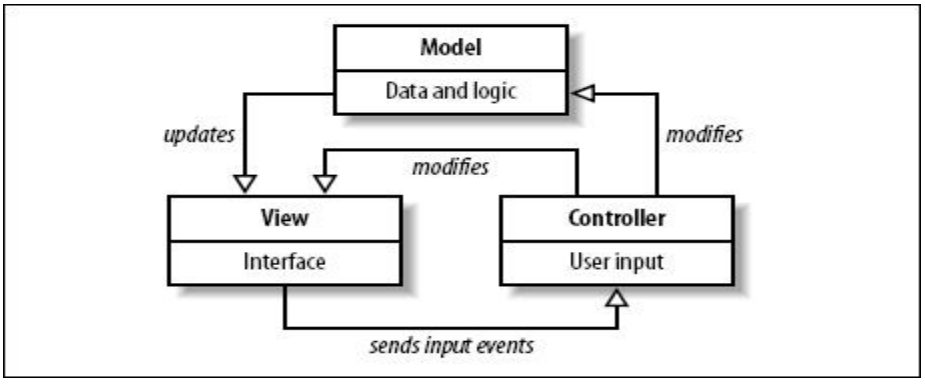

Figure 2:MVC

The reasons to use **MVC**are as follows:

- Since basic working of product is based on different modules which will work in collaboration with each other so MVC was best choice to use.
- All three major modules Player controls, HUD and game objects have their complete functionality for efficient working.

h. Logical View

Logical view contains class diagram and use case diagram. It describes the static behavior of system.

Use case Diagram

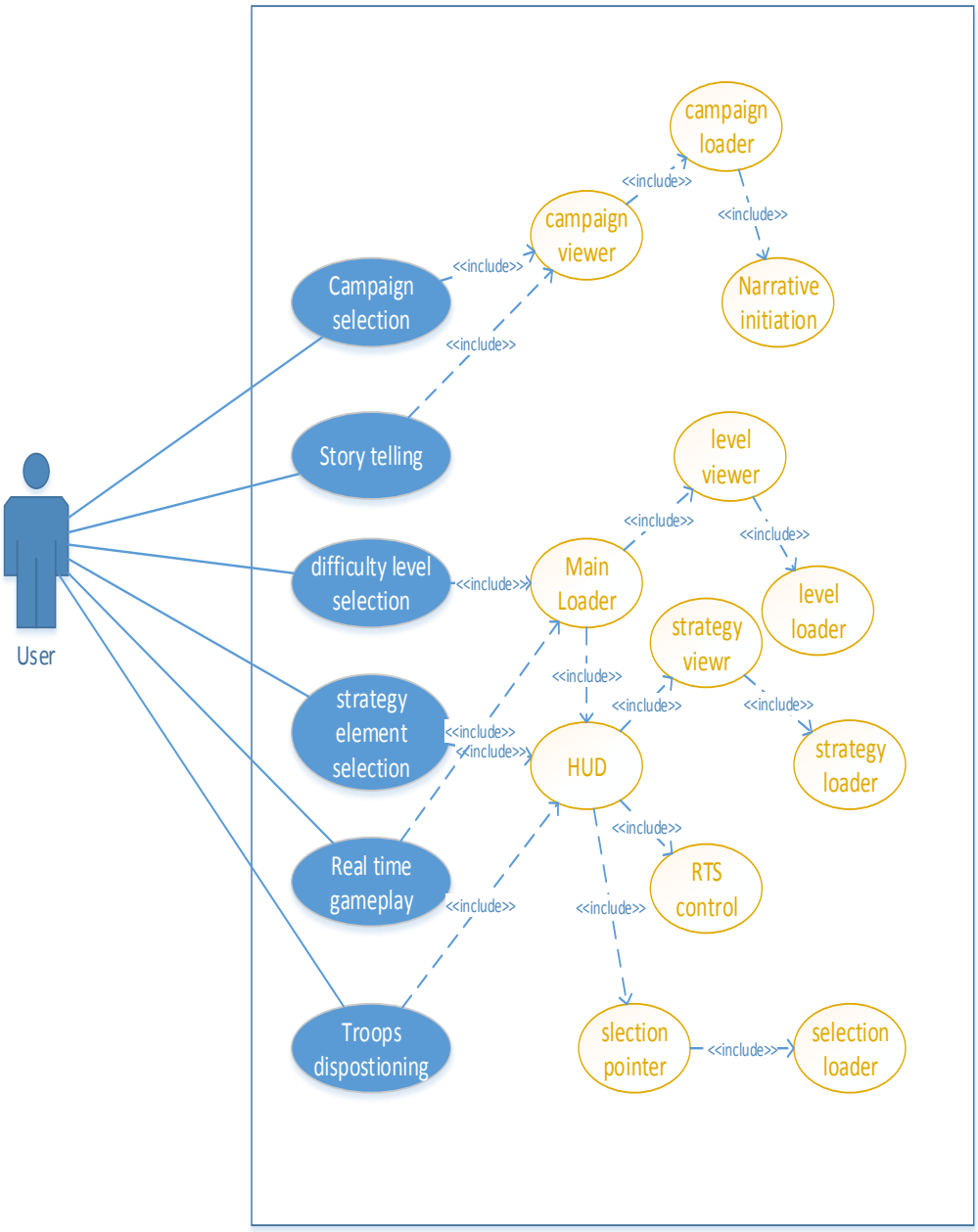

Figure 3:Main Use Case

Real –Time GamePlay

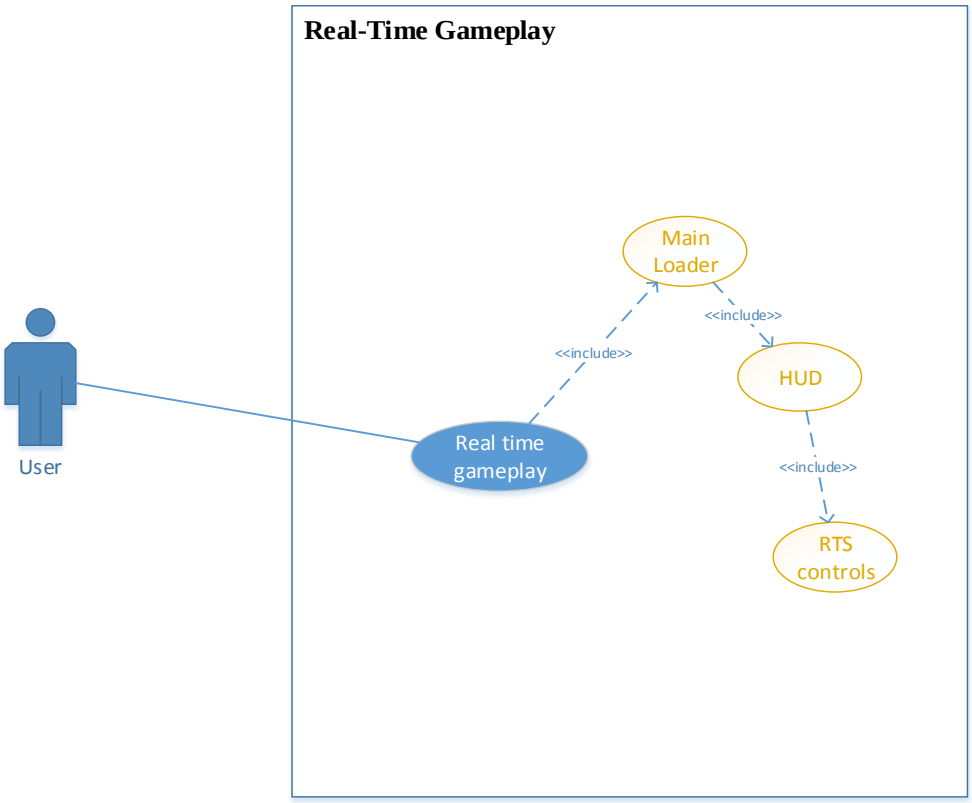

Figure 4:RTG

Scenario-based Story Telling

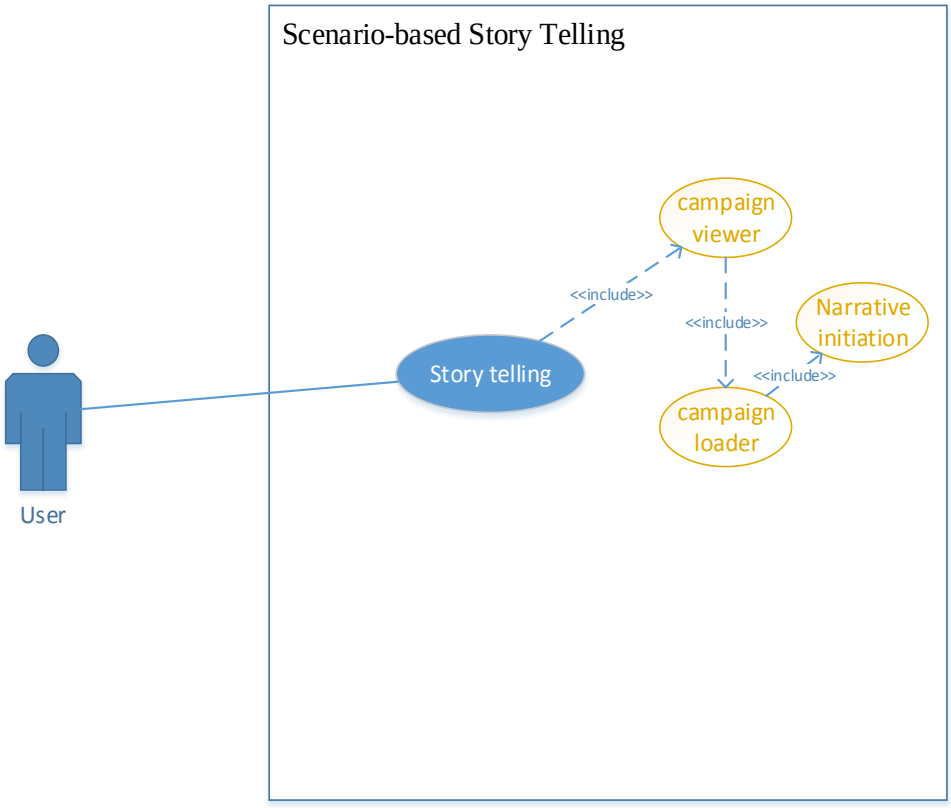

Figure 5:Story telling

Strategy Elements with Player Inputs

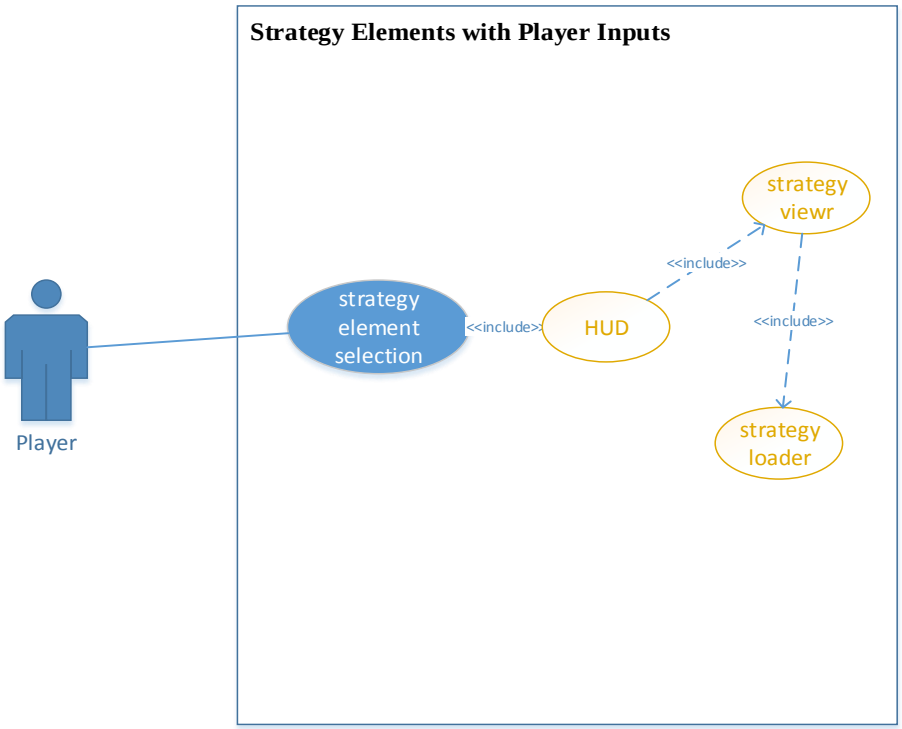

Figure 6: Selection Elements

Selection of Difficulty Level:

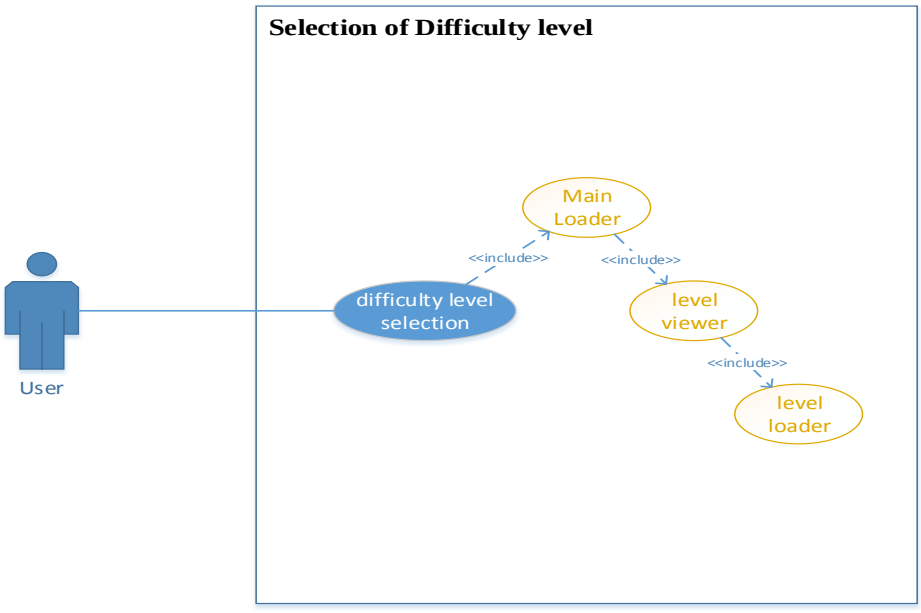

Figure 7:Difficult level

Placement of Troops on selected place:

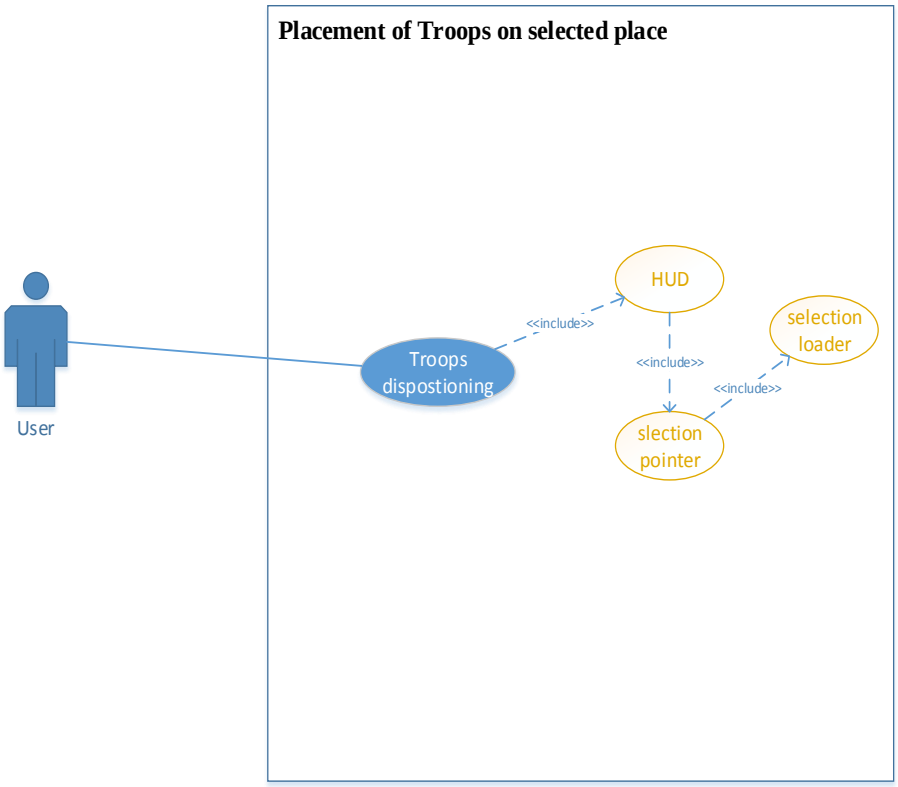

Figure 8:Troops placement

Class Diagram

i. Basic Flow

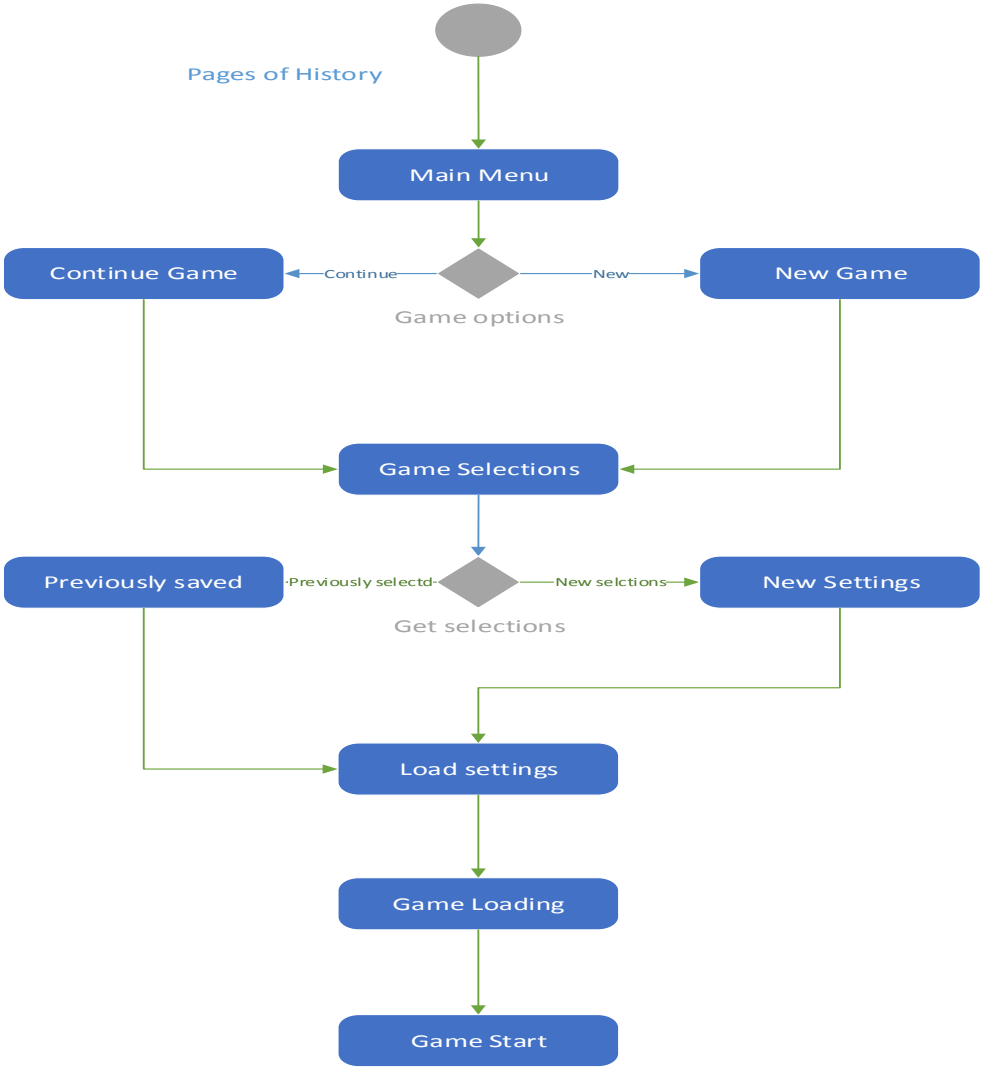

Figure 9: Flow Diagram

j. Dynamic View

Scenario-based Story Telling

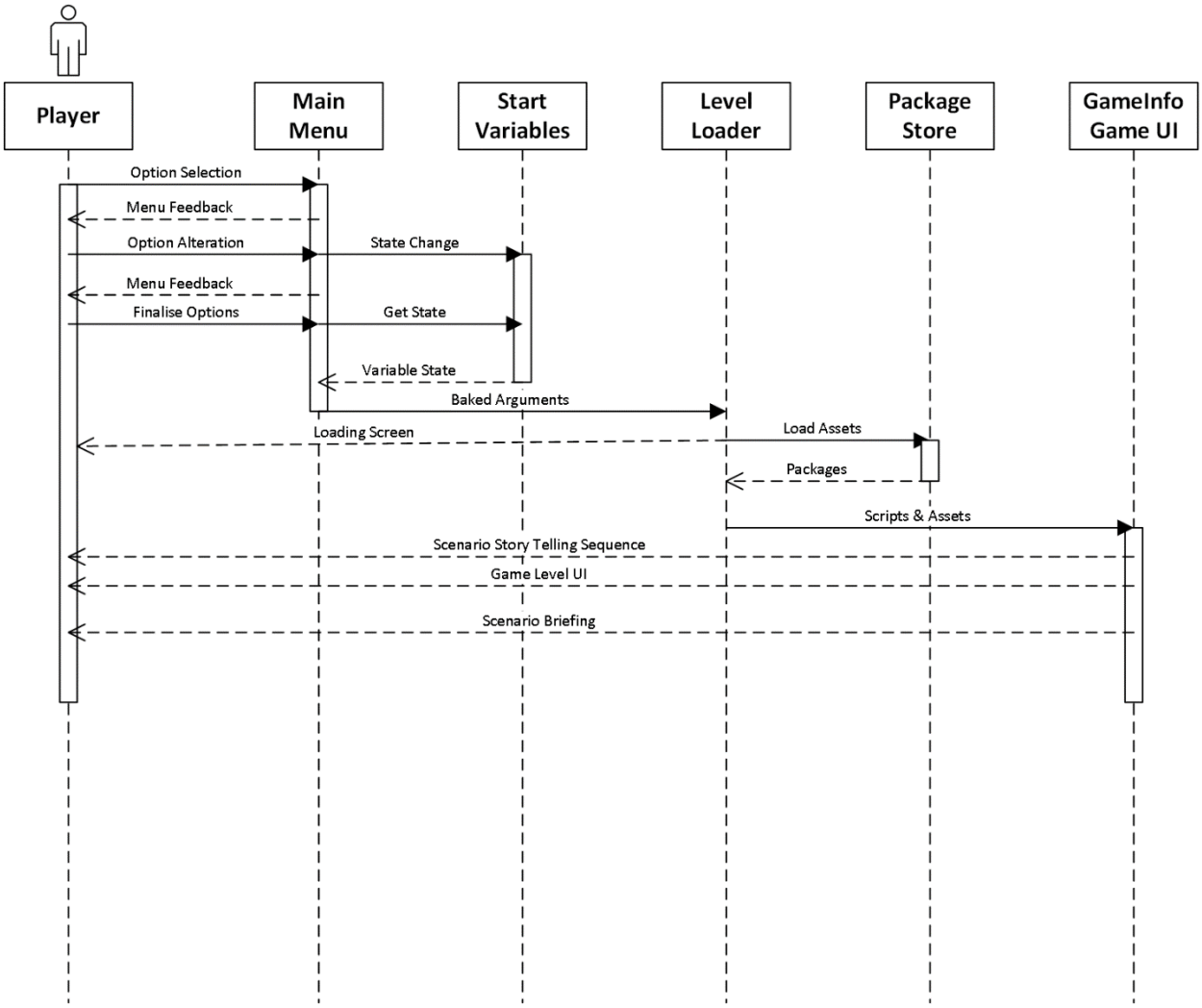

Figure 10:Sequence Diagram 1

Placement of Troops on selected place

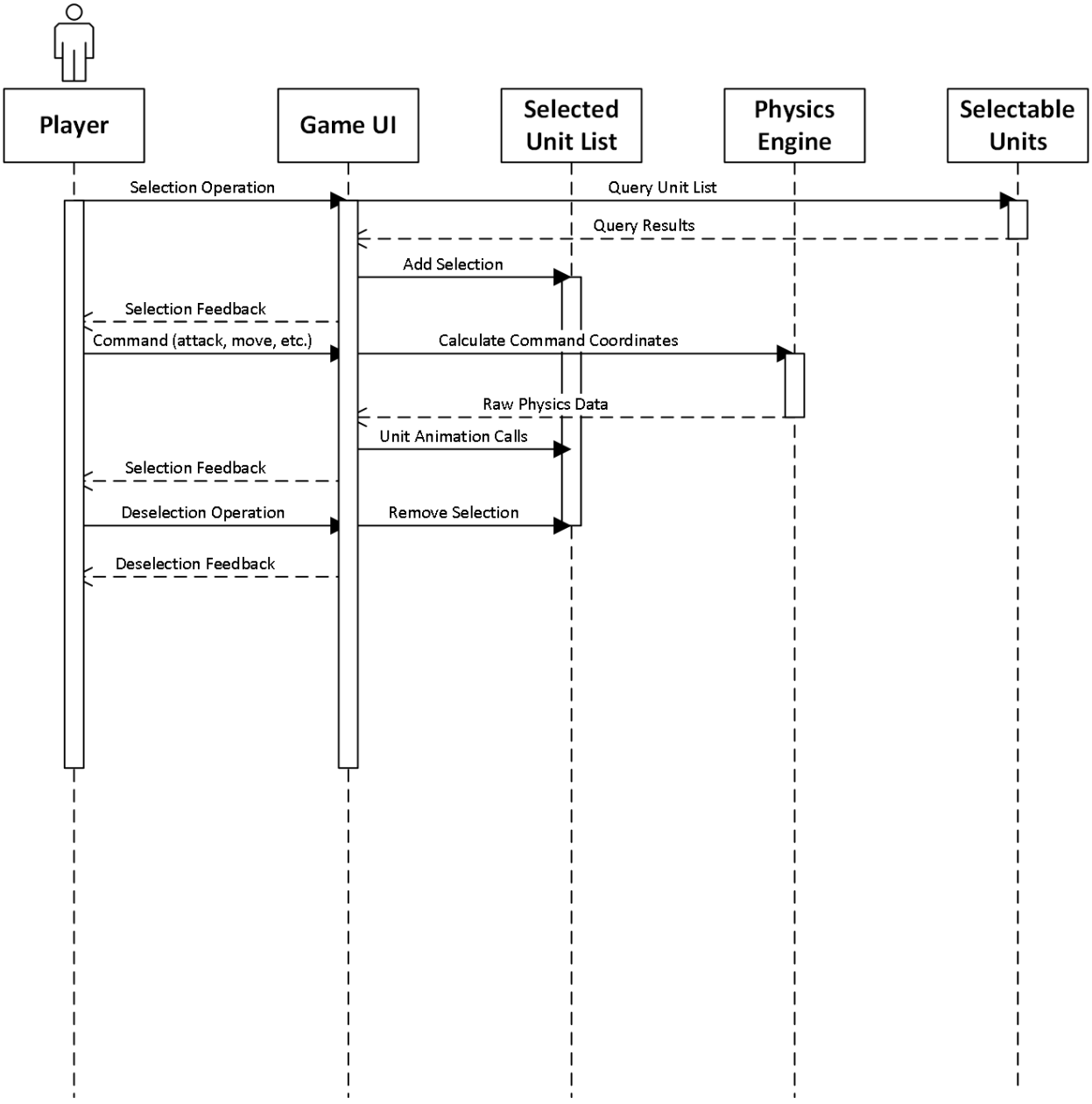

Figure 11:Seq diagram2

Real-Time Gameplay

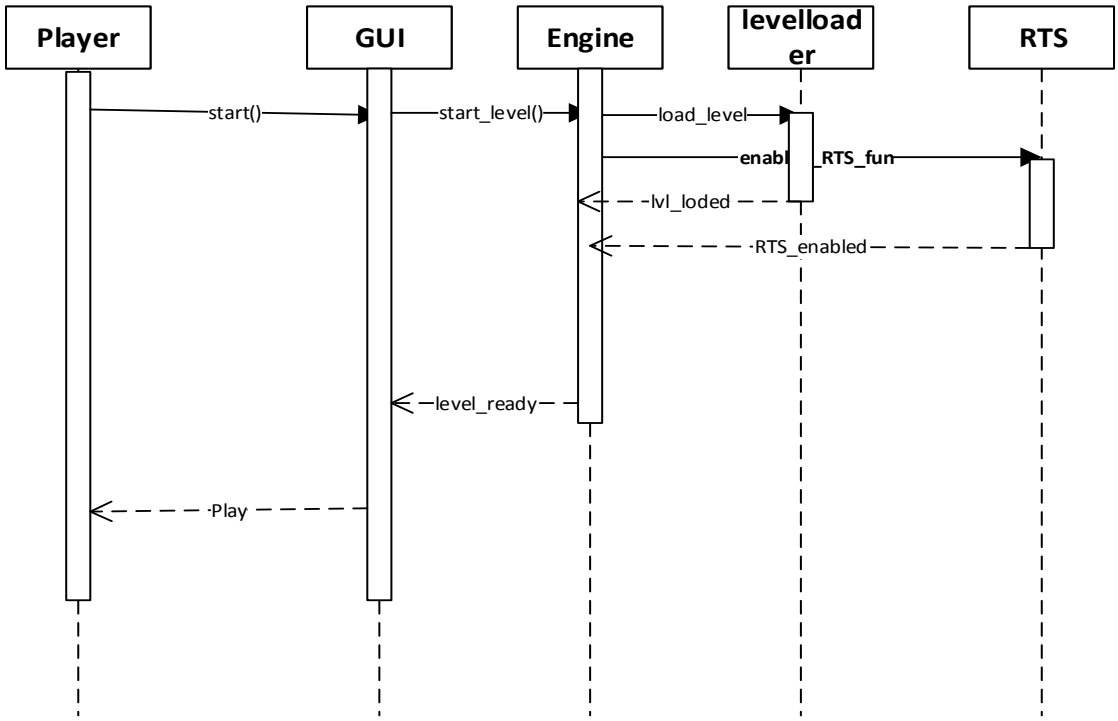

Figure 12: Sequence Diagram 3

Strategy Elements with Player Inputs

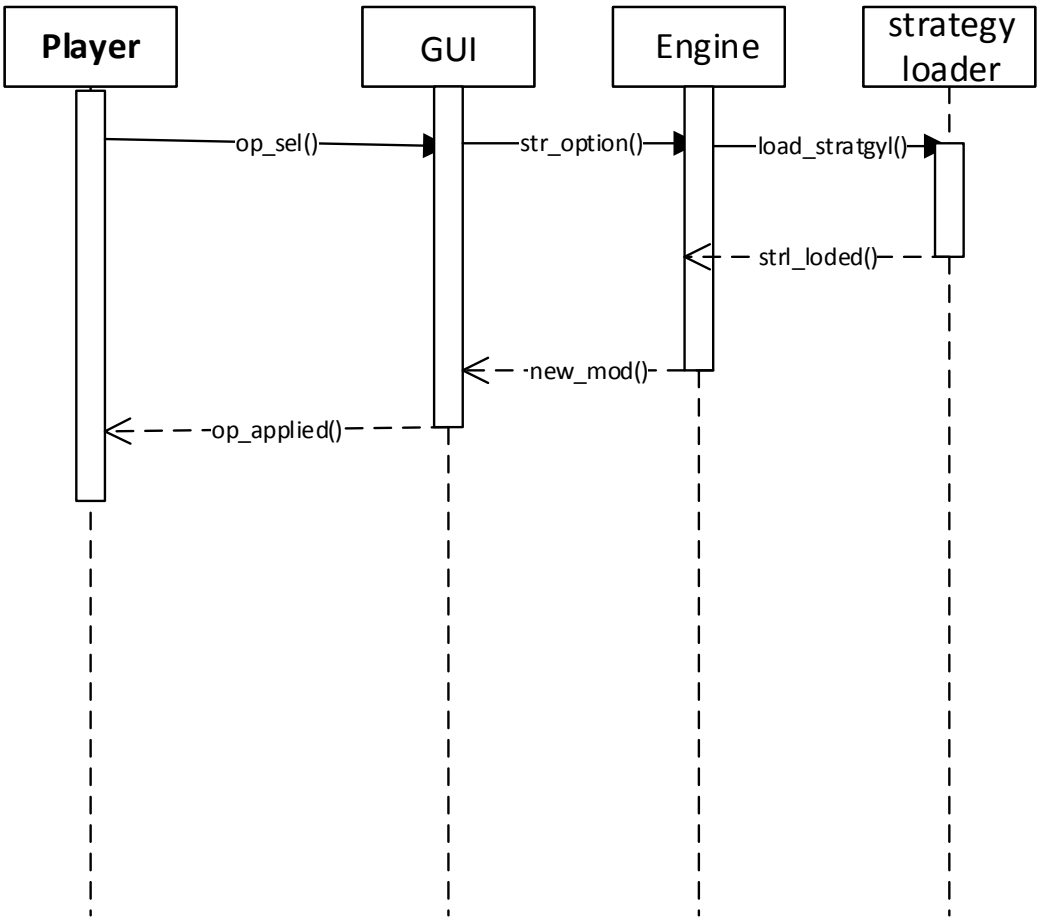

Figure 13: Sequence Diagram 4

Selection of Difficulty level

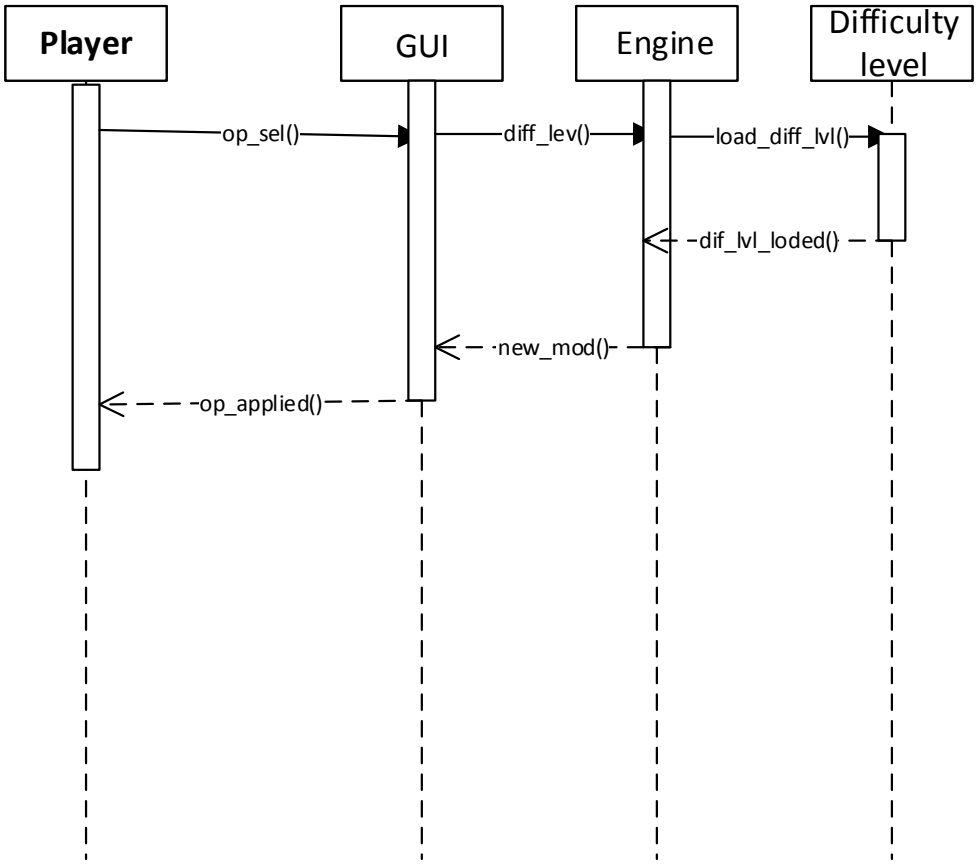

Figure 14: Sequence Diagram 5

Chapter 4

System Implementation Tools and Technologies

Photoshop: Adobe Photoshop has been used to integrate and rasterize 2D images. It is utilized in making reference images for hit box and icons design. Not only has this but props for 3D moving objects been using alpha images.

Maya: Autodesk Maya has been used to use in development of number of objects . It stands as a backbone and core development tool for all kinds of meshes (static and moving). Props imported from 2D images are mounted on to 3D skeletons to form realistic objects. Also material design for static meshes has been created in to it .

MudBox: Mud box is advance 3D object development tool, which is integrated with UDK. 3D objects are further modified and fine-tunead to form polygons, edges and collusion surfaces. After defining these edges, Meshes are exported in the format easily understandable to development engine.

UDK: The Unreal Engine is a game engine developed by Epic Games. Although primarily developed for first-person shooters, it has been successfully used in a variety of other genres, including stealth, MMORPGs, and other RPGs. With its code written in C++, the Unreal Engine features a high degree of portability and is a tool used by many game developers today and so by us. Current version being utilized is UE4 whose main features are real-time global illumination, eliminating pre-computed lighting, reduce iteration time and allow direct updating of C++ code. Along with it developer can then jump to the source code and edit it. Elements in the game can be clicked on directly to more easily change the game world. This also ultimately results in less of a divide between technical

artist, a designer, and a programmer. The result is a reduced time to compile code and allows game creators to tweak settings in real time. Portable with many plateform and integration with many other tool like MUDBOX and MAYA , makes it a sound equipment to be handled.

1. Software Implementation

The software has been developed by using concepts of OOPs and step3 visual programing. All the functionality is implemented in related classes and intercommunication between the classes is performed by passing variables to appropriate functions or notifying to the registered events with related data so that concerned class can deal with the content accordingly.

a. Game Initialization

In this our game engine will be initialized there are number of packages and files that are needed to be loaded for the engine initialization. After this all the files, packages and pipelines which are required by our product will be loaded which are given below:-

- Platform API: In this UDK engine will load all its packages, files and all the necessary items which it required to run the product smoothly without any hindrance.
- Packages, levels and resources of the project will be loaded after Platform API of the engine which the Pages of history required to run for its working.
- Main loop: will be started, main loop basically gets all the calls from complete working environment of the game and will further process them and respond them. Some primary classes and their derived classes required for the running of the game level will be loaded in this, Primary classes are:
    - Main _Menu

- Camera class basically will use to give the view to the player this will also be initialized.

```
#include "CameraActor.generated.h"
UCLASS(ClassGroup=Common, hideCategories=(Input, Rendering), MinimalAPI, Blueprintable)
classACameraActor : public AActor
{
        GENERATED_UCLASS_BODY()
public:
        // The camera component for this camera
        UPROPERTY(Category=CameraActor, VisibleAnywhere, BlueprintReadOnly)
        TSubobjectPtr<class UCameraComponent>CameraComponent;
private:
        UPROPERTY()
        uint32 bConstrainAspectRatio_DEPRECATED:1;
        UPROPERTY()
        floatAspectRatio_DEPRECATED;
        UPROPERTY()
        floatFOVAngle_DEPRECATED;
        UPROPERTY()
        floatPostProcessBlendWeight_DEPRECATED;
UPROPERTY()
```

```
    structFPostProcessSettingsPostProcessSettings_DEPRECATED;


public:

    // Begin UObject interface

    virtual void Serialize(FArchive&Ar) OVERRIDE;

    ENGINE_API virtual void PostLoadSubobjects(FObjectInstancingGraph*
OuterInstanceGraph) OVERRIDE;

    // End UObject interface

};
```

  - Player control
- Pipelines initiation is the next thing that will be initialize in our product for the communication of different components of the system these pipelines are:
  - Sound pipelining: In this a channel will be establish in between engine and sound card. This channel will pass on the sound calls to the sound card from engine side.
  - Graphic Pipelining: In this a communication channel will be establish between game engine and graphic card for the delivery of graphic calls to the graphic card.
  - I/O pipelining: establishment of this is to create a link between game engine and I/O.

b. Running the Product

In this phase all the primary and their derived classes will be instantiated. From Main_menu class player will select the level and type in result of this following will be loaded:-

- Maps will be loaded through the terrain class and terrain class will load the buildings class and all the derived classes of the building will also be loaded according to the level of the game.

```
#include "PlayerStart.generated.h"

UCLASS(ClassGroup=Common, hidecategories=Collision,MinimalAPI)
classAPlayerStart : public ANavigationObjectBase
{
        GENERATED_UCLASS_BODY()

        /** Used when searching for which playerstart to use. */
        UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Object)
        FNamePlayerStartTag;

#if WITH_EDITORONLY_DATA
        UPROPERTY()
        TSubobjectPtr<class UArrowComponent>ArrowComponent;
#endif

        ENGINE_API virtual void PostInitializeComponents() OVERRIDE;

        ENGINE_API virtual void PostUnregisterAllComponents() OVERRIDE;
```

- HUD Class will be initialized that will be required for the loading of the HUD for the game play.

```
class ENGINE_API FHUDHitBox
{
public:

        FHUDHitBox( FVector2D InCoords, FVector2D InSize, constFName&InName,
boolbInConsumesInput, int32 InPriority );

        /**
         * Are the given coordinates within this hitbox.
```

```
	 * @param	InCoords			Coordinates to check.
	 * @returns true if coordinates are within this hitbox.
	 */
	bool Contains( FVector2D InCoords ) const;

	/**
	 * Debug render for this hitbox.
	 * @param	InCanvas			Canvas on which to render.
	 * @param	InColor				Color to render the box.
	 */
	void Draw( class FCanvas* InCanvas, constFLinearColor&InColor );

	/** Get the name of this hitbox. */
	constFName&GetName() const { return Name;};

	/** Should other boxes be processed if this box is clicked. */
	constboolConsumesInput() const { return bConsumesInput; };

	/** Get the priority of this hitbox. */
	const int32 GetPriority() const { return Priority; };

private:
	/** Coordinates of top left of hitbox. */
	FVector2D	Coords;

	/** Size of this hitbox. */
	FVector2D	Size;

	/** The name of this hitbox. */
	FName		Name;

	/** Wether or not this hitbox should prevent hit checks to other hitboxes. */
	boolbConsumesInput;

	/** The priority of this hitbox. Higher boxes are given priority. */
	int32 Priority;
};
```

- Team class will be loaded and all its derived classes as well :-
    - Squad
    - Animal
    - Weapon
    - Person
    - Vehicle

Level will be started and following will also loaded of a particular level:-

- All the variables will initialized
- Setup of Game_state class it is the main class which will control all the moves and calls will get the response from other classes will passes them to the required classes
- Setup of selected level: Game_State will load the setup of selected level which will include the terrain class, building class and all its derived classes.
- HUD will be Loaded: which will display all the updated values related to game play including buildings, squads, weapons and health
- Setup of Player controls: these are the controls which player will use in order to control the game
- Setup of user interaction through this user will be able to interact with the game. It will pass the calls to Game_state class and this class will further transfer them to the required class

c. Player Request

Whenever there is player input either click or press of some key Player_controller class will get the call and will transfer it to the Game_State class will identify that what player requires, if this click is on HUD than it will have different response if on game play area than it will have different response.

In case of HUD: it will execute the command via blue print.

In case of Game play area: Class will get the id of the actor and will apply the action on it which player have applied on it. After this it will get the response from of the map from it will apply it on the map and will pass these values to the HUD. HUD will update the values according to calls which it will receive from Game_State class on the user request.

d. Response of the Game on user Input

It is the response of the game on the input of the player . On player Input Game_state will perform few task:-

- Id of the actor involved is passed to the Game_state it will perfrom required operation on it and passed it back to the class from where this call was generated and also updated on the HUD.
- Calculation of the path an actor has followed on the command and also save it for future and also pass it to all the concerning classes for their future use
- Calculation of the offset based on terrain information this will also change the values of loc in the Person class and to all those classes which required its location for future, it also pass its value to the HUD as well for updation and for the player knowledge

```
enumELandscapeSetupErrors
{
        LSE_None,
        // No Landscape Info available
        LSE_NoLandscapeInfo,
        // There was already component with same X,Y
        LSE_CollsionXY,
        // No Layer Info, need to add proper layers
        LSE_NoLayerInfo,
        LSE_MAX,
};
```

```
UCLASS(dependson=ULightComponent, HeaderGroup=Terrain, Placeable,
hidecategories=LandscapeProxy, showcategories=(Display, Movement, Collision,
Lighting, LOD, Input), MinimalAPI)
classALandscape : public ALandscapeProxy, public INavRelevantActorInterface
{
        GENERATED_UCLASS_BODY()

        UPROPERTY()
        classULandscapeSplinesComponent* SplineComponent;

        // Make a key for XYtoComponentMap
        staticFIntPointMakeKey( int32 X, int32 Y ) { return FIntPoint(X, Y); }
        static void UnpackKey( FIntPoint Key, int32&OutX, int32&OutY ) { OutX = Key.X;
OutY = Key.Y; }

        // Begin AActor Interface
#if WITH_EDITOR
        virtual void CheckForErrors() OVERRIDE;
        virtual void Destroyed() OVERRIDE;
#endif
        // End AActor Interface

        // Begin INavRelevantActorInterface Interface
        virtualboolDoesSupplyPerComponentNavigationCollision() const OVERRIDE {
return true; }
```

- Listen to the collisions it means if actor is a soldier it should not be able to cross through the wall. For this all the building meshes are so arranged that these have a collision level against any such kind of move of an actor
- Animation Playing on the player action some animation is required on the character that animation will be applied through player control class. This class will apply the animation on to the character which was required

```
class ENGINE_API APlayerController : public AController
{
        GENERATED_UCLASS_BODY()
```

```
/** UPlayer associated with this PlayerController.  Could be a local player or a net connection. */
UPROPERTY()
classUPlayer* Player;

/** when true, reduces connect timeout from InitialConnectionTimeOut to ConnectionTimeout.
        Set once initial level load is complete (client may be unresponsive during level loading). */
uint32 bShortConnectTimeOut:1;

/** Used in net games so client can acknowledge it possessed a specific pawn. */
UPROPERTY()
classAPawn* AcknowledgedPawn;

/** Director track that's currently possessing this player controller, or none if not possessed. */
UPROPERTY(transient)
classUInterpTrackInstDirector*  ControllingDirTrackInst;

/** last used FOV based multiplier to distance to an object when determining if it exceeds the object's cull distance
 * @note: only valid for local player
 */
floatLocalPlayerCachedLODDistanceFactor;

/** Heads up display associated with this PlayerController. */
UPROPERTY()
class AHUD* MyHUD;

// ********************************************************************************
// Camera/view related variables

/** Camera manager associated with this Player Controller. */
UPROPERTY(EditInline)
classAPlayerCameraManager*  PlayerCameraManager;
```

```
/** PlayerCamera class should be set for each game, otherwise Engine.PlayerCamera is used */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=PlayerController)
TSubclassOf<class APlayerCameraManager>PlayerCameraManagerClass;

/**
 * True to allow this player controller to manage the camera target for you,
 * typically by using the possessed pawn as the camera target. Set to false
 * if you want to manually control the camera target.
 */
UPROPERTY(EditAnywhere, Category=PlayerController)
boolbAutoManageActiveCameraTarget;

/** Used to replicate the view rotation of targets not owned/possessed by this PlayerController. */
UPROPERTY(replicated)
FRotatorTargetViewRotation;

/** Smoothed version of TargetViewRotation to remove jerkiness from intermittent replication updates. */
FRotatorBlendedTargetViewRotation;

/** The actors which the camera shouldn't see - e.g. used to hide actors which the camera penetrates */
UPROPERTY()
TArray<class AActor*>HiddenActors;

/** Used to make sure the client is kept synchronized when in a spectator state */
UPROPERTY()
floatLastSpectatorStateSynchTime;
```

- Play of sound according to the situation , whenever some action has been taken by the player there will be related sound with it which will played on that particular action Player_controler will be responsible for this as well

Chapter 5

Project Analysis and Evaluation

a. Testing

To ensure quality of the product, testing is conducted. Accuracy and efficiency of tasks performed by our systemhad to be tested to analyze the system and verify and validate it. Software testing techniques and results obtained are discussed in the coming sections.

b. Testing Levels

Separate modules were developed to provide different functionalities of the system. All of these modules were tested at different levels during development and after integration. Different levels of testing and results have been described here:

I. Unit Testing

Each module was designed, developed and tested individually. Each functionality was also tested separately.

a) Static Mesh Testing: All the static Meshes were designed and then were tested. These are like Buildings, Vegetation's, bridges and stone blocks were tested in it.

b) Animated Mesh Testing: All the animated Mesh designed and were tested animated meshes like Person and Horse

c) HUD(Head Up Display) Testing it was tested for updation of values and for display

d) Camera Testing was carried out for the player view at different angles

Test Case ID	1
Unit to Test	Static Meshes
Assumptions	1. Title of itself 2. Description 3. Button 1 and Button 2 4. Trespassing 5. Sound
Test Data	1. Game play Mode mouse Click 2. Game play Mode move of an actor
Steps to be Executed	1. Turn On the Game Play Mode 2. Click on the particular mesh and check the response 3. click on the Button 1 and Button 2 and check their functionality 4. Moved a person through it for its trespassing check
Expected Result	Sound will turn ON all the Buttons will work properly , trespassing is meeting its criteria and passing the required information
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	2
Unit to Test	Animated Mesh testing
Assumptions	On click 1. will pass its info 2. it will animate according to given coordinates 3. perform all the applied animations
Test Data	1. Mouse Click 2. Animate to the selected place
Steps to be Executed	1. Game is in Game play mode 2. Player will click the cursor of the on to the animated mesh 3. it will pass the required information 4. player will select the animated mesh and place it on some other place on the map 5. Selected mesh will animate to that particular place
Expected Result	Animated mesh has passed the information , sound was played and moved to the selected place
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	3
Unit to Test	HUD Testing
Assumptions	1. HUD has placed on selected positions on the screen 2. HUD is updating the values 3. Selections made from HUD are working on the actual scenario as well
Test Data	1. selection of different objects from the map (person, buildings ,weapons) 2. changing the mode of army from defense to attack from HUD
Steps to be Executed	1. Turn on game play mode 2. change the values of person, buildings and weapons 3. HUD should also change the values of selected items 4. Now change of the mode from HUD 5. On Game playing area mode of the force will also be changed
Expected Result	Updation of different objects on HUD Change of Mode on MAP of the force
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	4
Unit to Test	Camera testing
Assumptions	Different Camera views are available
Test Data	Changing the view of the camera with the help of mouse
Steps to be Executed	1. Game is in Game play mode 2. Player will change the view through mouse from all the directions 3. Player will also apply Zoom on different objects
Expected Result	Change of view should be available Zoom on different objects should be possible
Actual Result	As Expected
Pass/Fail	Pass

II. Integration Testing

- Initially static meshes with camera were tested for integration when there integration was finalized than we move ahead
- Then animated meshes with camera were tested for integration
- After that static and animated meshes were tested with camera for integration
- HUD will be integrated with all these static meshes, animated meshes and camera at the end.

III. System Testing

System testing was performed at the end of development. Complete system was tested by running the game and by applying different options on it for complete testing

c. Results

The results of the tests were in the acceptable range. There were very less meshes and HUD updates which were not working according to the designed patterns (almost none). Although the performance does depend on the system used, there is still margin of improvement.

d. Analysis

The results tell us that the application is truly implementing standards and concepts of 3D graphics and video gaming. But it definitely need minimum system specifications for efficient working, this means that the dependency of performance is directly proportional to the kind of processor, number of cores, available memory. Usability of static and animated meshes makes them easy to be re-used and implemented /integrated in other.

This application implements a new idea that is emerging and a source of learning and entertainment for the youth.

Conclusion and Future Work

The aim of this project was to develop and demonstrate an industrial standard game encompassing true events of history. The project was chosen after careful selection mainly because very less work has been done on this and also because it was interesting. The team had set an aim to change themindset of the students in order to work in this field as well. The goal was set and achieved as per planned timeline.

During the course of development, the team encountered some difficulties, especially when it came to "Animated Meshes" module. The team had some experienced in Maya and MudBox but it needed polishing. There were no animation trees available for animated meshes. This meant developing / planning / tweaking everything from scratch. Also UDK was needed to be learnt. The team also required in-depth knowledge of Photoshop, adobe sound booth and NVidiaPhysics.

The goal (thanks to Allah Almighty) has been achieved till now, although the team shall continue its work on this application in their own capacity to make it more efficient and also implement this idea into other applications. The team hopes that this project brings good name to Military College Of Signals, NUST and the Armed Forces of Pakistan. "Amin!"

Appendix A: Glossary

Activity diagram: An analysis model that shows a dynamic view of a system by depicting the flow from one activity to another.

Availability: Present and ready for use; at hand; accessible.

Class: A description of a set of objects having common properties and behaviors, which typically correspond to real-world items (persons, places, or things) in the business or problem domain.

Constraint: A restriction that is imposed on the choices available to the developer for the design and construction of a product.

API (application programming interface): specifies how some software components should interact with each other. In addition to accessing databases or computer hardware, such as hard disk drives or video cards, an API can be used to ease the work of programming graphical user interface components.

OS: Operating System

Data flow diagram: An analysis model that depicts the processes, data collections, and flows among them that characterize the behavior of a business process or of a software system.

External interface requirement: A description of an interface between a software system and a user, another software system, or a hardware device.

Feature: A set of logically related functional requirements that provides a capability to the user and enables the satisfaction of a business objective.

Flowchart: An analysis model that shows the processing steps and decision points in the logic of a process or of a program.

Functional requirement: A statement of a piece of required functionality or a behavior that a system will exhibit under specific conditions.

Hardware: A computer and the associated physical equipment directly involved in the performance of data-processing or communications functions.

Hardware Interface: The logical and physical characteristics of each interface between the software product and the hardware components of the system.

Implementation: Execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

Interface: A point where two systems, subjects, organizations, etc., meet and interact.

Nonfunctional requirement: A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior.

Operating Environment: The circumstances surrounding and potentially affecting something that is operating.

Operating System: A collection of software that manages computer hardware resources and provides common services for computer programs.

Procedure: A written description of a course of action to be taken to perform a given activity, describing how the activity is to be accomplished.

Process: A sequence of activities performed for a given purpose. A process description is a documented definition of those activities.

References: List of any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.

Response: A reaction, as that of an organism or a mechanism, to a specific stimulus.

Requirement: A statement of a customer need or objective, or of a condition or capability that a product must possess to satisfy such a need or objective. A property that a product must have to provide value to a stakeholder.

Requirements specification: See software requirement specification and specification, requirement.

Scope: The portion of the ultimate product vision that the current project will address. The scope draws the boundary between what's in and what's out for the project.

Software requirements specification: A collection of the functional and nonfunctional requirements for a software product.

Specification, requirements: The process of documenting a system's requirements in a structured, shareable, and manageable form. Also, the product from this process.

Stimulus: Something causing or regarded as causing a response.

Supplementary Information: Something added to complete the information.

System requirement: A top-level requirement for a product that contains multiple subsystems, which could be all-software or software and hardware.

Usability: Fit for use; convenient to use.

Use case: A description of an interaction between an actor and a system that results in an outcome that provides value to the actor.

Use case diagram: An analysis model that identifies the actors who can interact with a system to accomplish valuable goals and the various use cases that each actor will perform.

User: A customer who will interact with a system either directly or indirectly (for example, using outputs from the system but not generating those outputs personally). Also called end user.

User class: A group of users for a system who have similar characteristics and requirements for the system.

User Interface: the logical characteristics of each interface between the software product and the users.

User requirement: User goals or tasks that users must be able to perform with a system, or statements of the user's expectations of system quality.

Validation: The process of evaluating a work product to determine whether it satisfies customer requirements.

Verification: The process of evaluating a work product to determine whether it satisfies the specifications and conditions imposed on it at the beginning of the development phase during which it was created.

PC: Personal computer

Call log: display of missed, received and dialed calls

Modifiability: The effort required to make changes in the software

Portability: The effort required to move the software to a different target platform

Reliability: dependable, how often the software fails.

Appendix B: References

www.unrealengine.com/udk/

www.unrealengine.com/

www.unrealengine.com/udk/documentation

http://www.autodesk.com/products/

http://www.autodesk.com/products/autodesk-maya/overview

http://epicgames.com

http://www.unrealengine.com/en/news/epic_games_launches_unreal_engine_4_integrated_partners_program

http://www.adobe.com/products/photoshopfamily.html

http://www.photoshop.com/

http://tv.adobe.com/en/show/learn-soundbooth-cs4

http://www.adobe.com/products/audition.html