

**TRIBHUVAN UNIVERSITY**  
**INSTITUTE OF ENGINEERING**  
**NATIONAL COLLEGE OF ENGINEERING**



**A**  
**MINOR PROJECT REPORT**  
**ON**  
**“A GAME ON Q-LEARNING”**

**SUBMITTED BY:**  
**DIPESH JOSHI (071 BCT 117)**  
**BADAL ARYAL (071 BCT 120)**  
**ANIL KHANAL (071 BCT 133)**  
**BIKASH KHANAL (071 BCT 118)**

**LALITPUR, NEPAL**

**JULY 26, 2017**

**TRIBHUVAN UNIVERSITY**  
**INSTITUTE OF ENGINEERING**  
**NATIONAL COLLEGE OF ENGINEERING**



A

**MINOR PROJECT REPORT**  
**ON**  
**“A GAME ON Q-LEARNING”**

**SUBMITTED BY:**

DIPESH JOSHI(071 BCT 117)  
BADAL ARYAL (071 BCT 120)  
ANIL KHANAL (071 BCT 133)  
BIKASH KHANAL(071 BCT 118)

**SUBMITTED TO:**

DEPARTMENT OF ELECTRONICS AND COMPUTER  
ENGINEERING IN FULFILLMENT OF THE REQUIREMENT FOR  
BACHERLOR’S DEGREE IN ELECTRONICS & COMPUTER  
ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER  
ENGINEERING  
LALITPUR, NEPAL  
JULY 26, 2017

## PAGE OF APPROVAL

This is to certify that the work carried out by **Mr. Dipesh Joshi, Mr. Badal Aryal, Mr. Anil Khanal, and Mr. Bikash Khanal** for the project entitled “**A GAME ON Q-LEARNING**” for the award of the degree of Bachelor of Computer Engineering of the Institute of Engineering is based upon the authentic work. We have the pleasure in forwarding their project. The project was carried out under our supervision and all the materials included as well as the software product is the result of their yearlong authentic work-effort.

---

Department of Electronics  
And Computer Engineering  
Institute of Engineering  
Pulchowk campus  
(**External Examiner**)

---

**Er. Pradeep Adhikari**  
Department of Electronics  
and Computer Engineering  
National College of Engineering,  
Talchhikhel, Lalitpur  
(**Head of Department**)

---

**Er. Mohan Maharjan**  
Department of Electronics  
and Computer Engineering  
National College of Engineering  
Talchikhel, Lalitpur  
(**Project Coordinator**)

---

**Er. Anup Shrestha**  
Department of Electronics  
and Computer Engineering  
National College of Engineering,  
Talchhikhel, Lalitpur  
(**Project Supervisor**)

## **COPYRIGHT**

The author has agreed that the Library, Department of Electronics and Computer Engineering, National College of Engineering, may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisor who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, National College of Engineering in any use of the material of this project report. Copying or publication of the other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, National College of Engineering and author's written permission is prohibited. Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

---

**Er. Pradeep Adhikari**

Department of Electronics and Computer Engineering

National College of Engineering,

Talchikhel, Lalitpur

Nepal

## ABSTRACT

“Space Shooter” is a name of the game that was developed for our third-year project in Computer Engineering Department at National College of Engineering (NCE). The game is a 2D single player BIRDS VIEW SHOOTER (BVS) created using C# in Unity-Game Engine. The aim of this document is to describe how the problem was analyzed, development methods, development processes carried out for creating a working product and provides information on what went right and wrong about the project and the lessons learned from experiences that have been gained during a semester project period and our discussions.

System is a simple application of q-learning technique. The game does not have advanced graphics but it surely implements the q-learning algorithm. The system is developed for desktop and any other platforms. It runs in Windows, Mac, Linux or any other operating system.

The game is able to provide a lot of entertainment to its user. In the game, Graphics and UI are made as easy to understand as possible for any group of users. It's not difficult to program in unity for Game Objects as the programming language is similar to natural language and only few keywords are required. In addition to this Unity has a lot of engines such as AI engine, graphics engine, sound engine and physics engine built in it. Here the engines are simply the libraries and dependency to be used in the game.

The unity supports two of the widely accepted programming language i.e. C# and JavaScript. Therefore, programming in unity is easy. In addition to this, unity has a large community in which we can share our work and get support from them.

This project doesn't use any dataset and therefore we don't need any database for it. The only data used are player score and the enemy score. In addition to this, the user can enter his name and give the username in the game. And we have also tried to track the position of the moving object in the game. At last, in the game we have tried to show the path followed by the player.

## ACKNOWLEDGEMENT

The project “**A GAME ON Q-LEARNING**” is prepared as per requirement of the course designed by Tribhuvan University for Bachelor’s Degree in Electronics and Computer Engineering as minor project for third year students.

Firstly, we would sincerely like to acknowledge to our college “**National College of Engineering**” for providing us with this great opportunity, technical and financial helps to lead this project towards its completion.

We would like to thank **Er. Pradeep Adhikari** (Head of Department of Electronics and Computer Engineering) for providing guidance in our minor project. We would like to express our gratitude to our project supervisor: **Er. Anup Shrestha** for providing guidelines in our minor project. We are equally indebted to our teachers: **Er. Om Prakash Mahato, Er. Raju Shrestha, Er. Mohan Maharjan, Er. Subash Pandey**, for providing guidelines in our minor project.

At last, but not the least, we would like to thank all the teachers, students and others who help us in bringing this idea in our major project.

## LIST OF FIGURES:

FIGURES	PAGE NO
FLOWCHART	<b>25</b>
USE CASE DIAGRAM	<b>29</b>
CLASS DIAGRAM	<b>30</b>
SEQUENTIAL DIAGRAM	<b>31</b>
ITERATIVE MODEL	<b>16</b>
SNAPSHOTS	<b>32</b>

## LIST OF ABBREVIATIONS

GDD	Game Design Document
BEVS	Birds Eye View Shooter
RL	Reinforcement Learning
UI	User Interface
IDE	Integrated Development Environment



## Contents

PAGE OF APPROVAL.....	3
COPYRIGHT.....	4
ABSTRACT.....	5
ACKNOWLEDGEMENT.....	6
LIST OF FIGURES:.....	7
LIST OF ABBREVIATIONS .....	8
INTRODUCTION.....	11
Background .....	11
Knowledge representation .....	11
Problem Statement.....	12
Project Objectives & Deliverables.....	13
Project Organization .....	13
LITERATURE REVIEW .....	14
TECHNICAL SUMMARIES.....	15
System to be developed .....	15
Intended Users.....	15
Development Tools and Equipment .....	15
METHODOLOGY .....	16
Proposed Development Process .....	16
Development Process .....	16
RESULTS .....	17
ALGORITHM .....	18
DISCUSSION.....	21
What Went Right: .....	21
What Went Wrong.....	22
CONCLUSIONS.....	24
APPENDIX.....	25
FLOWCHART.....	25
Use case diagram .....	29
Class diagram .....	30
Sequential diagram .....	31
Snapshots.....	32
Gantt Chart .....	33
REFERENCE.....	34



## INTRODUCTION

“Space Shooter” is a 2D arcade game is created as a third-year project for Computer Engineering department at National College of Engineering (NCE). The aim of this document is to describe the development process and the results of this project.

### Background

On the process of computer improvement and increase in computing capability, computers are slowly entering the field of artificial intelligence. This means computers can act like human beings. Development in AI can be very useful to human beings or detrimental in other side.

Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving". Artificial intelligence is the hot topic nowadays. The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects.

### Knowledge representation

- ❖ Planning
- ❖ Learning
- ❖ Natural language processing
- ❖ Perception
- ❖ Motion and manipulation
- ❖ Social intelligence
- ❖ Creativity
- ❖ General intelligence

Machine learning is the subfield of computer science that, according to Arthur Samuel in 1959, gives "computers the ability to learn without being explicitly programmed. Machine learning helps computers to program themselves and act autonomously.

Machine learning mimics the way humans learn or a new born baby learns from its surroundings. A cycle of rewards and punishment helps a child to distinguish what is right and wrong, what should be reacted in future and what not.

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in

itself (discovering hidden patterns in data) or a means towards an end (feature learning).

- **Reinforcement learning:** A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

### Problem Statement

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

The basic reinforcement learning model consists of:

- ✚ a set of environment and agent states  $S$ ;
- ✚ a set of actions  $A$  of the agent;
- ✚ policies of transitioning from states to actions;
- ✚ rules that determine the scalar immediate reward of a transition; and
- ✚ Rules that describe what the agent observes.

Q-learning is a model-free reinforcement learning technique. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Additionally, Q-learning can handle problems with stochastic transitions and rewards, without requiring any adaptations. It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable.

Q learning is used in many applications such as games (Atari 2600) and any sort of learning processes.

The idea of developing game is not new for us. It was one of the options for our project of third year from first year. However, our team consisted of four students did not want to take such a risk. Because, **unity** was a new technology and everyone was familiar with it. Thus, we had to waive a chance to implement our ideas on unity because it provides us with an environment for the development of interactive 2D and 2D content including a rendering and physics engine, a scripting interface to program interactive content, a content exporter for many platforms (desktop, web, mobile) and a growing knowledge sharing community.

The primary goal of using Q Learning in this game is to implement Reinforcement learning in game so that game will be more fun to play. As we know the primary goal of using AI is to search from all the possible move. So, this is purpose of AI in this game also.

### **Project Objectives & Deliverables**

The purpose of the project is to design and implement a 2-dimensional game written in C# using Unity – Game Engine. The project includes a complete level of game with documentation. The level will include everything that should be available in a FPS game. The game will be a single-player. The team members don't take aim at developing an instructive game; instead the aim is action, action and more action. Project Deliverables:

- **Project Plan**
- **Game Design Document**
- **Test Plan**
- **Test Report**
- **Final Report**
- **Product**
- **User Manual**

### **Project Organization**

Each member takes responsibilities of each role in each phase to guarantee that the project can continue. Please refer to section 4.1 in Project Plan [1] for role specific responsibilities and refer to Gantt Chart document [2] for detailed work distribution of the project.

## LITERATURE REVIEW

Reinforcement Learning (RL) is an exciting new field of machine learning, in which bots learn by playing games. The biggest breakthrough was made by Deepmind, who programmed an AI to play Breakout. It is different from traditional machine learning (supervised or unsupervised) in that there are no training samples with expected outputs. In RL, the bots are thrown into a computer game (and gaming is a field they are most extensively tested in), and then trained to learn by *observing their actions and rewards*.

This is the most radical thing about RL: We, as programmers, only tell the algorithm what we expect, like finish the game with at least 100 score. The algorithm then randomly tries different combinations, learning as it goes, until it reaches our desired goal (which can take hours or days).

RL is hot! You may have noticed that computers can now automatically learn to play ATARI games (from raw game pixels!), they are beating world champions at Go, simulated quadrupeds are learning to run and leap, and robots are learning how to perform complex manipulation tasks that defy explicit programming. It turns out that all of these advances fall under the umbrella of RL research. We also became interested in RL myself over the last year: We worked through Richard Sutton's book, read through David Silver's course, watched John Schulman's lectures. A group at DeepMind is working in the DeepRL group, and most recently they pitched in a little with the design/development of OpenAI Gym, a new RL bench-marking toolkit.

Many game run on what is called a Game Engine. The Game Engine is what tells components in a computer how to Render Graphics or Play Audio. Game such as Half-Life 2, Counter Strike and Portal run on the same engine i.e. **Source** Engine which is why they have many things in common when it comes to physics, graphics quality and game play. Previously Game Studios have to spend huge amount of work in creating engine from the scratch or purchase one from pre-existing one from another studio which was too much expensive. But these days, a lot of independent engine has been made sole purpose of people using them to build their games on top. One such engine is **Unity**. For implementing this game, we use Unity which is a Game Engine itself.

## TECHNICAL SUMMARIES

### System to be developed

“Space Shooter” is an action based 2D first-person shooter game where the player takes role of the main character to destroy enemies. The game is inspired from a science fiction movie “Star Wars” by George Lucas, which tells the story which begins with the title “A long time ago in a galaxy far, far away...” in which there is a battle between the galaxies. New and crowded community afraid of these wars and is trying to destroy them. The game starts exactly at the point where the two main characters of the movie realizes that the only way to protect from them is to kill all of them.

The gameplay is action-based with no strategic or role-playing elements; instead, the game entirely provides a rush of adrenaline. The only goal is to kill all enemies until even one is not left before they kill our character and to collect both the gems.

The version in this project consists of one level. We will add upper levels in our future works. The game starts in milky way galaxy, where the end begins. In addition, the game involves no checkpoints; the player will start the game at level entries.

The player plays these levels from a first person view in a 2D environment. The player experiences the game world through the eyes of a main character. This makes one feel one is a part of the game. The player controls the main character named Dr. Hero, who is one from ours galaxy. The abilities of our character as a player is flying in all four directions, using-reloading-changing weapons, and collecting the pickups.

The game has a design with only single-player mode. Depends on sub-modes, the number of enemies and pick-ups will vary. During the game, one has the ability to pick up more Gems. Along the way one face off against certain number of enemies which depends on the type of sub-mode. By default, the player has 0 score. The attack of enemies reduces the health primarily. The effect of damage varies depending on the type of attack. The enemies can have the ability to use to shoot the player. However, the player can pick up the power-ups that increase health. The player has only one life. When his/her health reduces to zero or lower, one will die. When our main character dies, the player loses the game. If the player kills all the enemies and collect gems before the health reduce to zero or lower, one wins the game.

Please refer to Game Design Document for detailed information about the game.

### Intended Users

Because of no violent activities, the system/game is mainly intended for all including child (age 6+) who also are referred to as players.

Players Intended users for the game

Project Examiner Responsible for project evaluation

Team Members Responsible for system development

### Development Tools and Equipment

Here is the list of tools and equipment used to develop the proposed system.

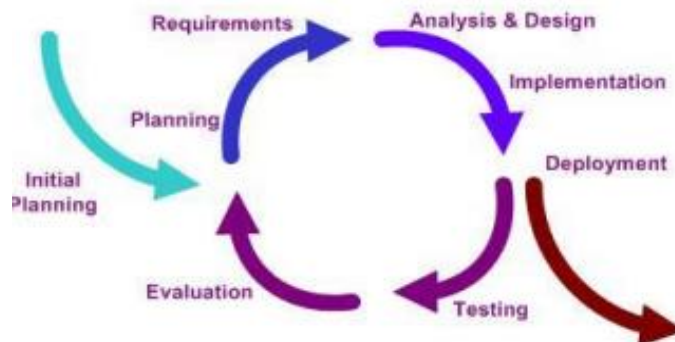
- **Unity 5.1**
- **Monodevelop IDE for unity**
- **Microsoft Visual C# 2008 Express Edition**

## METHODOLOGY

### Proposed Development Process

Even if one can prepare a well-thought-out design document, some features do not give the same effect in gameplay as on the paper. During implementation phase many features are added, removed or modified. Accordingly, one needs to make some modifications on game design and requirement analysis. Thus, one of the most suitable development methodologies for game development is iterative development process. Most of game developers have the same idea about the advantages of iterative development process for game developers. As Ian [6] said, “The more times you iterate, the better your final game will be”.

We prefer the iterative development process with specified milestones and deliverables for our project.



### Development Process

We planned the project over a period of 2 months and divided it into four iterations. We planned the first iteration for planning, second iteration for game design, third iteration for coding and the final iteration for testing and finalizing the product.

In the first iteration, we focused on Project Plan determined as the first planned milestone of the project as we mentioned in Project Plan document. Planning is essential for starting of upcoming milestones and delivering a finished project on time. Successful completion of a project is heavily dependent on effective planning. Iteration 1 was set to approximately one week (8 days) to figure out the detailed activities in order to minimize risk to the final result and delivery.

The second iteration started by brainstorming among group members on what the game would be. Each group member denoted the attributes or properties of the game that one dreamed to implement. We gathered suggestions together and chose the ones that was possible to be



implemented within a semester of project time. As soon as the game concept became clear, we made some early decisions on basic requirements of the project in order to more easily reach the development goals. GDD was also the main deliverable in this iteration. Game design document was meant to be a living document. In other words, throughout the production process the document was updated, if needed.

As we had GDD in hands, it was used as baseline to start implementation. In the third iteration coding in C# using **Unity-Game Engine** was under way. None of us had experience in game programming. Therefore, most of time in this iteration was dedicated for internal training sessions. In this iteration, we needed to achieve four milestones each was dependent on the previous one.

GDD was also in favor of Test Plan. Last iteration was planned for testing and finalizing the product. The testing process is an iterative process. We performed the testing process in four iterations. The successful testing process of software requires a good plan. Therefore, after the requirements of the project are confirmed, the future testing of the system and the code were planned. The test plan provided information on how and when the testing will be executed. In the second iteration, test cases were designed for the planned tests. In iteration three, the designed test cases were executed alongside the module testing and usability testing. During the last iteration, according to the result of the tests, the test reports were documented properly and the bugs were reported after the testing is completed.

## RESULTS

Concerning the managerial aspects, the members acquired experience in how to apply an iterative process for game development. Due to the fact that the group consists of two members, both have the responsibilities of all the roles for continuity of the project. Therefore, both team members participated in producing each document. The responsibility of steering the group and keeping the project on track is on both members' hands. The result of the first iteration of the process is the documentation about project planning that is necessary for the successful progress. Project Plan has not been updated and has been used throughout the process. Because of lack of experience in game development, we faced with some difficulties while producing the game design document. All members of the group must have the overall idea of the game concept and mechanics clearly to do implementation. At the beginning, we could not estimate which properties could be implemented within project time. On this account, the requirements as presented in the initial version of GDD are based on early decisions. In the upcoming stages, GDD was updated and iterated throughout the development process. Due to time shortage, we had to put 2D graphics design and creation out of project scope. Therefore, we needed to find eligible free 2D models from the websites where the 2D graphics are sold. Luckily, we easily found the models in a short time and integrate them into our gameplay seamlessly. The testing process of this project went mostly as we had planned; however, the test plan schedule was pretty much followed. Because the implementation took longer time as we expected, we had to spend less time on testing process. Due to time issues, the testing process that we planned to perform with test tool was left out. After testing finished, it was time to write the test protocols which contains the detailed information about the results of the tests and comments for each test if necessary. The end result of the implementation is a game that almost fulfills all of its functional requirements. The only part that has not been implemented is the mini map. For a start, the mini map was not essential part of the game but more an addition to the esthetics. In our upcoming works, we can improve the galaxy background, accordingly add a mini map and make our game as bigger as the games in the market. Please refer to Test Protocols for the detailed information about the parts that has problems. The final stage was used to create the

compulsory final report. We established a document template according to the **IOE** standards and fill out the information about the progress.

## ALGORITHM

We suppose each state as a room no. and the agent as a bot.

1.

START

2.

Initialize Q-matrix as :

$Q =$  , where columns represent states and rows are actions.

3.

Assign reward matrix as:

$R =$  , where columns represent states and r rows represent actions.

4.

Current room=4(suppose)

5.

Time delay for 2 seconds such that the bot moves in certain distance.

6.

Find out in which quadrant the enemy lies:

- If x and y both coordinates are positive then 1<sup>st</sup> quadrant.
- If x is negative and y is positive then it lies in 2<sup>nd</sup> quadrant.
- If x and y are both negative then it lies in 3<sup>rd</sup> quadrant.
- If x is positive and y is negative then it lies in 4<sup>th</sup> quadrant.

7.

If there are multiple divisions inside each quadrant then calculate the x and y difference range.

Suppose its 


 1<sup>st</sup> quadrant having 4 divisions and each division represents room no. Now to find exactly which room no the enemy lies ,we calculate the x and y difference range , assuming that bottom left corner is the origin.

8.

We have calculated the next state (suppose room no=3).

Also update: current room =3;

9.

Check another set of arrays to find out which states can we further move to after reaching that state. For eg. To find out what are the possible room nos. we can move after reaching room no 3 (in this case), we retrieve information from the array of room no. 3:

Room3:

5	2	1	0	0
---	---	---	---	---

This information says that the bot can move only to room nos. 5,2,1 after reaching to room no. 3.

10.

Therefore in this case ;

State=4

Action=3 and Next state=3

11.

Store the result R[4][3] into variable “E”

12.

Declare another array (suppose array “C”) of size 10.

13.

Calculate

while(room3[i] != 0)

{

C[i]=Q[Next state, Room3[i]]

} wend;

This function stores value of (next state ,all actions) in new array “C”.

14.

We have our major formula on which iterations is to be done:

$Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \gamma * \text{Max}[Q(\text{next state}, \text{all actions})]$

15.

- We have stored value  $R(\text{state}, \text{action})$  in variable E.
- We have stored value of (next state, all actions) in array C.
- We set gamma parameter to constant=0.8

16.

We compare each value stored in array C and calculate the maximum value .  
Suppose this value is stored in variable “max”

17.

Calculate:

$Q[\text{state}, \text{action}] = (E + \text{max} * 0.8)$  such that  $Q(4,3)$  stores the value  $(E + \text{max} * 0.8)$  and finally updates Q matrix.

18.

Set counter variable  $\text{count} = \text{count} + 1$ , (initially  $\text{count} = 0$ );

19.

Check if  $\text{count} = 36$  (or desired iterations);

if ( $\text{count} == 36$ )

{

then player loses the game and finally the bot learns the optimal shortest path.

Shortest path is traced by following the room no. with maximum q value.

}

Else

{

Goto step 3;

20. STOP

## **DISCUSSION**

We accomplished our final game project by facing different milestones. There were many obstacles such as account administration, license validation and many other operational difficulties. Also in the good part, we had a very good game engine - Unity because of which our work was more effortless. We categorized our work into what went wrong and what went right.

What Went Right:

### **Good pre-planning and resource management**

We had a clear concept of what we were exactly doing in our project. We had a clear vision of future accomplishments. We had very well planned for the resources required to develop our game. We had chosen unity engine which could full fill our major project requirements.

### **Good teamwork**

All our team members were highly familiar with the game idea. We co-operated well with each other and supported individual suggestions and completed each task at specified time. All members were obedient and thus we had a great team work. We made a work and time division schedule for all of us.

### **2D Models**

Because of limited time, the design and creation of 2D models are out of scope of the project. We planned to use free ready-to-use 2D models in the project. However, at the beginning this was a nightmare for us. If we did not find suitable models, making a game would be a fantasy and accordingly the project would fail. Fortunately, we found the suitable models in unexpected short time. We selected models from free models on the web sites selling models. At the end of the project, we realized that while we think finding 2D models as a nightmare, it becomes a stage that gives rapid results and makes us very happy

#### **5.1.4 Physics Engine**

We used Unity physics engine which is one of the favored open-source physics engines for physics and collision detection. Using physics in Unity was a one of the best decisions about the project. We did not encounter any problems about the engine although it was our first time to use physics engine and we had some doubts because the engine is in its version. The engine allowed us to focus more on gameplay, instead of spending time on creating our own physic library.


#### **5.1.5 Graphical User Interface (GUI)**

The first scene of the game starts like this. It consists of 3 buttons which

Control the interactions with game, another is a checkbox which control the music in the game with it the user can control volume as well and lastly there is a username box with `placeholder Username in which player puts his /her name to be used in the game.

Second Scene of game is playing arena in which there is a box for score and for tracking the position of player as well as enemies

And lastly there is a scene telling the player if they win or lose the game or if the game is over.



## What Went Wrong

### Debugging

Error detection was easier but error correction was more difficult. We had no prior knowledge of unity game engine and c-sharp programming language. So, error correction was difficult.

### Time management

Our time was limited and we had to complete all at assigned time. For this we divided work with time but still due to several environmental factors, we had a tough time completing the work. We could not add other special effects to the game due to limited time.

### Limited Time

The main problem for the project was limited time. Although the project is small, we think that 63 days are not enough for implementation. Due to the fact that we had no game development experience, we had to do a considerable amount of research during implementation. This did lead to slow down the progress of implementation of the game. Fortunately, we did not come up against the case of unfinished implementation work. This situation resulted in just putting increasing stress on us and acting as if we had panic attacks during the entire implementation phase. We felt the impacts of limited

time during not only implementation stage but also testing stage. We did not use test tools because of lack of time combining with the case of being unfamiliar with test tool as we defined as a risk in Test Plan. The lesson learned is that the game development needs more than 20 weeks.

#### **Late Feature Addition**

After alpha phase was end, we realized that we left one important point out of account while documenting the design of game. When all the weapons that the player had were out of bullets and collectable ammo packs and weapon packs were not left, the player did not have any equipment to defend himself and was waiting to be killed despairingly. Unfortunately, this was more than enough to raze charm of the game.

The easiest solution was to add a weapon that did not need bullets. First, we tried to add missiles. However, we could not integrate free 2D missile model that we found into our game world. Thus, we had to give up this option because we did not have enough time to waste and then we started to think about new options. Finally, we decided to use bullet instead. After integrating into the game, we accordingly updated our design document and test cases. We learned how right ones who suggest iterative development process for game development are.

## CONCLUSIONS

Despite the fact that we have involved in many projects up until now, this project is our first game development project. Both of us were very inexperienced in this field. Although we both have a little bit of fear because of our naivety, this was an unquestionably important experience for us. In the following section, we mentioned about the lessons that we have learned during this project process.

Having regard to the suggestions of the authors who wrote the articles about game development process, we decided to apply iterative development methodology to our project. Especially because of two reasons, we realized that this is a very fortune decision. We faced with the first reason causing us think like that during the preparation of GDD. At the beginning, we had a very clear vision about what we wanted to implement. Our game would roughly be a single player FPS game with a known back story inspired from a well-known movie . However, due to lack of experience we were not able to forecast how many percentages of the game functional requirements could be implemented within a prescribed project period. We established the final version of the GDD after a few updates. The second was during the testing process. As we know, the testing is an iterative process. Once the bugs are fixed, the testing has to be done repeatedly. In addition, while checking the test cases we realized that there existed some test cases missing. Accordingly, we went back and made some modifications and additions on Test Plan. Shortly, the chosen development methodology is the most suitable for our project without a shadow of a doubt.

At the end of the project, the other lesson that we learned and think to apply to our future projects is making a good project plan, remaining true to this plan as possible and documenting everything we do. By doing this, whenever we have a problem we can go back and see what we did the last time we had a similar problem.

Throughout the project, the biggest problem was limited time. Many times, during the project we had the devil of a time because of time shortage. The fact that we had examinations addition to our thesis work caused us to take breaks, albeit short ones. After these breaks, we had some difficulties in adapting to the thesis again.

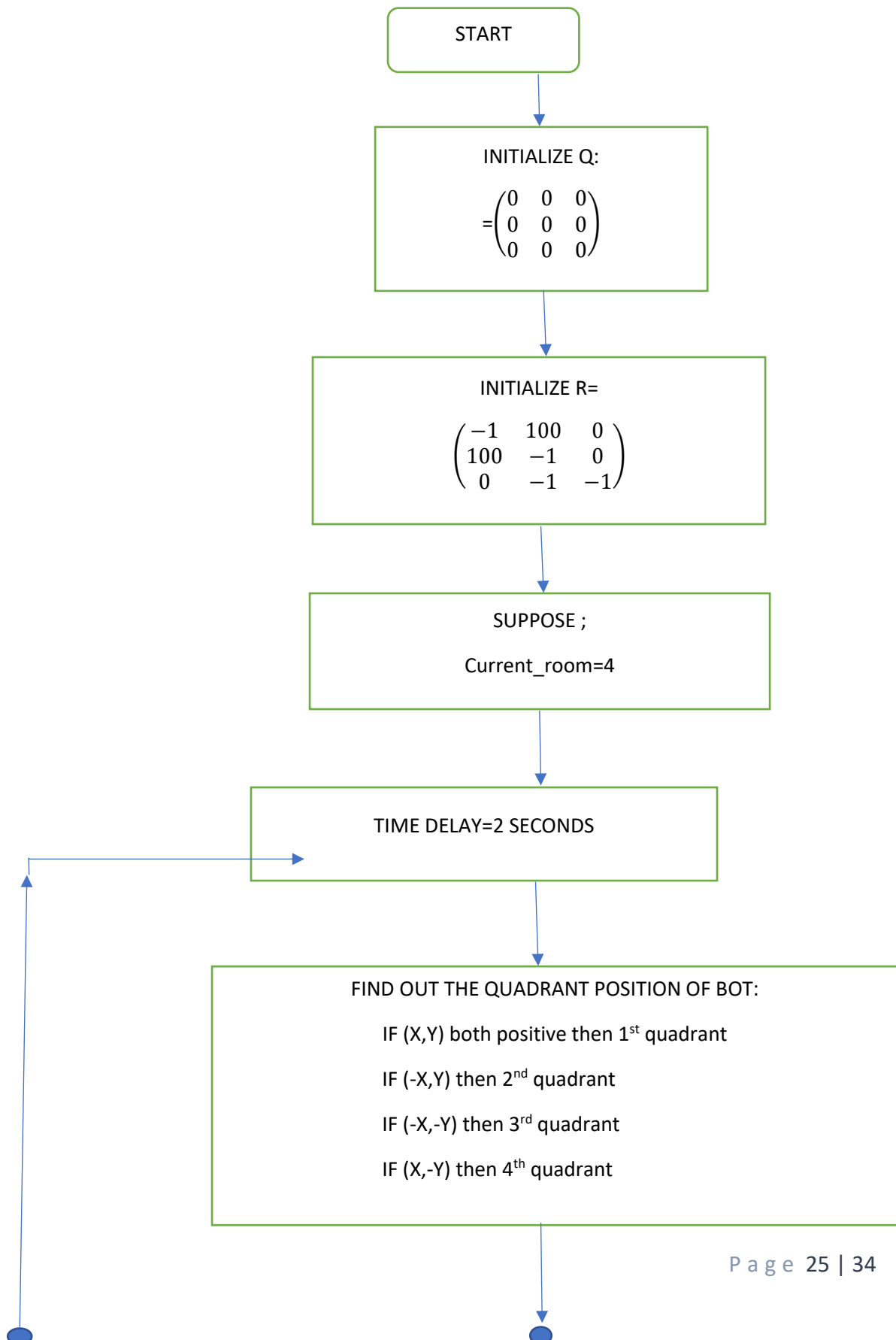
We, who developed a game for the first time, think that the decision on game genre is one of the most suitable decisions that we have made throughout the project. There exist so many alternatives to examine in the game market because of intense interest of the people in FPS games. Therefore, it was a good start for us. Our thought is “the more examples one has near at hands, the easier the implementation will be.”

As a result, 2 months were full of new great experiences and these experiences that we have gained will help us in our upcoming projects. Finally, the idea of developing game that was a dream for us since our bachelor education became real and we have experienced the pride of doing such a thing at the end of such an education.



## APPENDIX

### FLOWCHART:



```
graph TD; Start(( )) --> Step1[IF THERE ARE MULTIPLE DIVISIONS INSIDE THE QUADRANT  
,FIIND OUT EXACTLY WHICH ROOM NO. THE BOT LIES BY  
CALCULATING THE X AND Y DIFFERNECE.]; Step1 --> Step2[WE HAVE CALCULATED THE NEXT STATE FROM ABOVE  
(suppose room no.=3)  
ALSO UPDATE ; current room=3]; Step2 --> Step3[CHECK A SET OF ARRAY TO FIND OUT WHAT ARE THE POSSIBLE  
STATES AFTER THE BOT HAS MOVED TO NEXT STATE(room no=3 in  
this case)  
IN THIS CASE:  
State=4, action=3, next state=3]; Step3 --> Step4[STORE DATA OF R[4][3] IN VARIABLE "E"  
DECLARE ARRAY "C" OF SIZE 10]; Step4 --> End(( ))
```

IF THERE ARE MULTIPLE DIVISIONS INSIDE THE QUADRANT  
,FIIND OUT EXACTLY WHICH ROOM NO. THE BOT LIES BY  
CALCULATING THE X AND Y DIFFERNECE.

WE HAVE CALCULATED THE NEXT STATE FROM ABOVE  
(suppose room no.=3)  
ALSO UPDATE ; current room=3

CHECK A SET OF ARRAY TO FIND OUT WHAT ARE THE POSSIBLE  
STATES AFTER THE BOT HAS MOVED TO NEXT STATE(room no=3 in  
this case)  
IN THIS CASE:  
State=4, action=3, next state=3

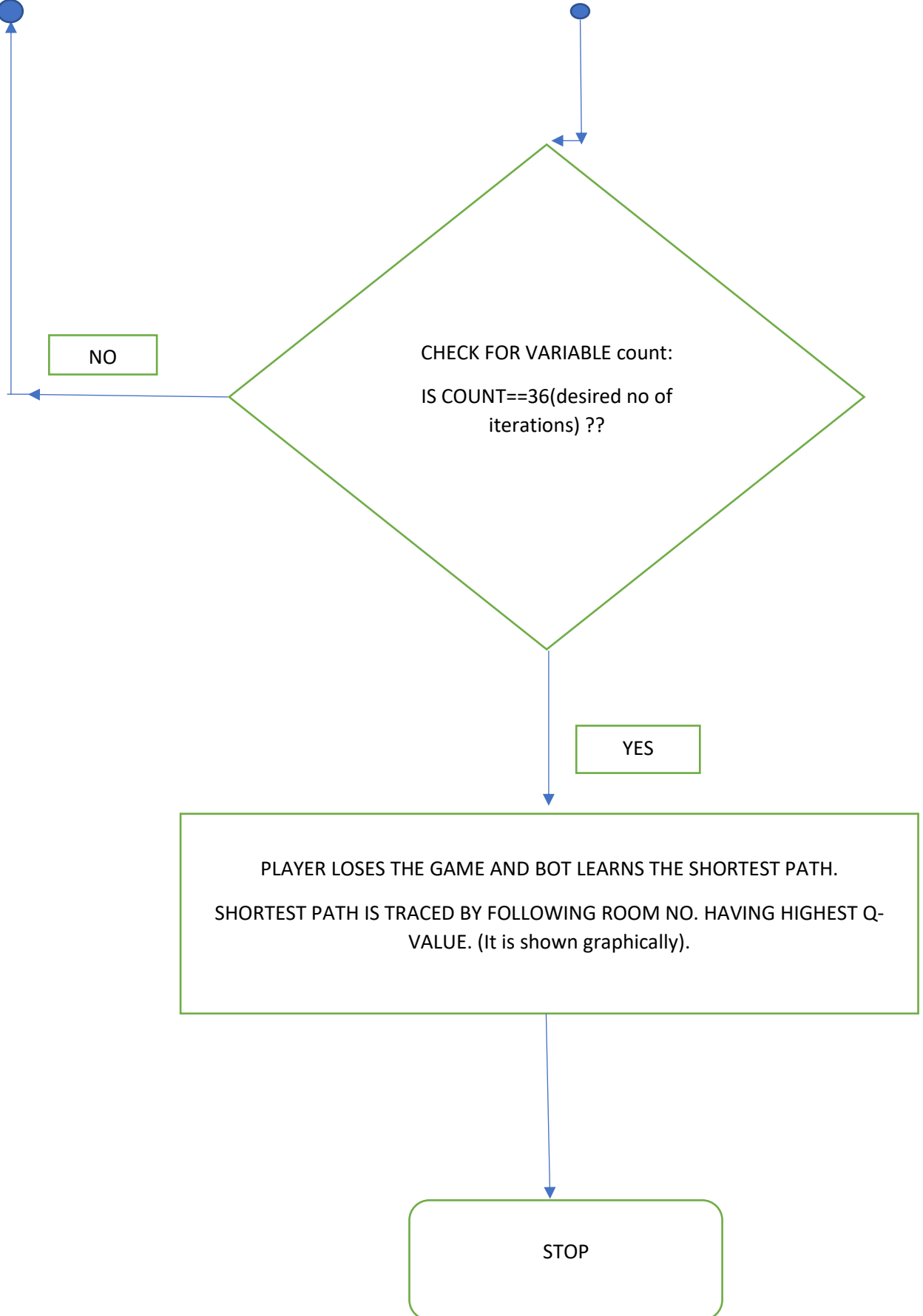
STORE DATA OF R[4][3] IN VARIABLE "E"  
DECLARE ARRAY "C" OF SIZE 10

WHILE (ROOM3[i]!=0)  
{  
C[i]=Q[next state.room3[i]];  
}wend;  
THIS FUNCTION STORES VALUE OF (next state, all actions) in array C.

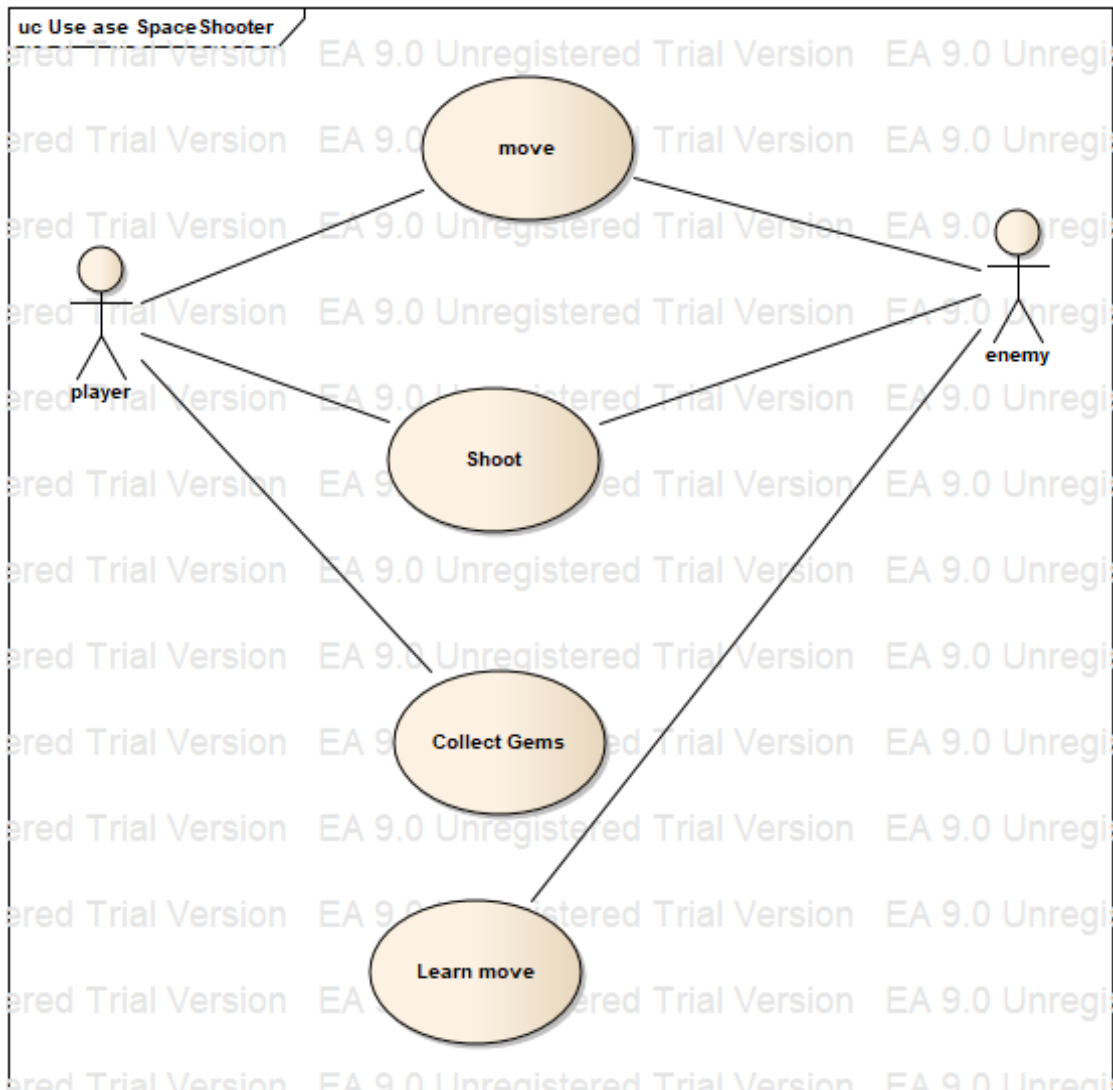
OUR MAJOR FORMULA ON WHICH ITERATION IS TO BE DONE IS :  
 $Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{gamma} * \text{MAX}[Q(\text{next state}, \text{all actions})]$  , where  
Gamma=0.8(constant)

COMPARE EACH VALUE STORED IN ARRAY "C" AND CALCULATE MAXIMUM VALUE.  
SUPPOSE THIS VALUE IS "MAX"

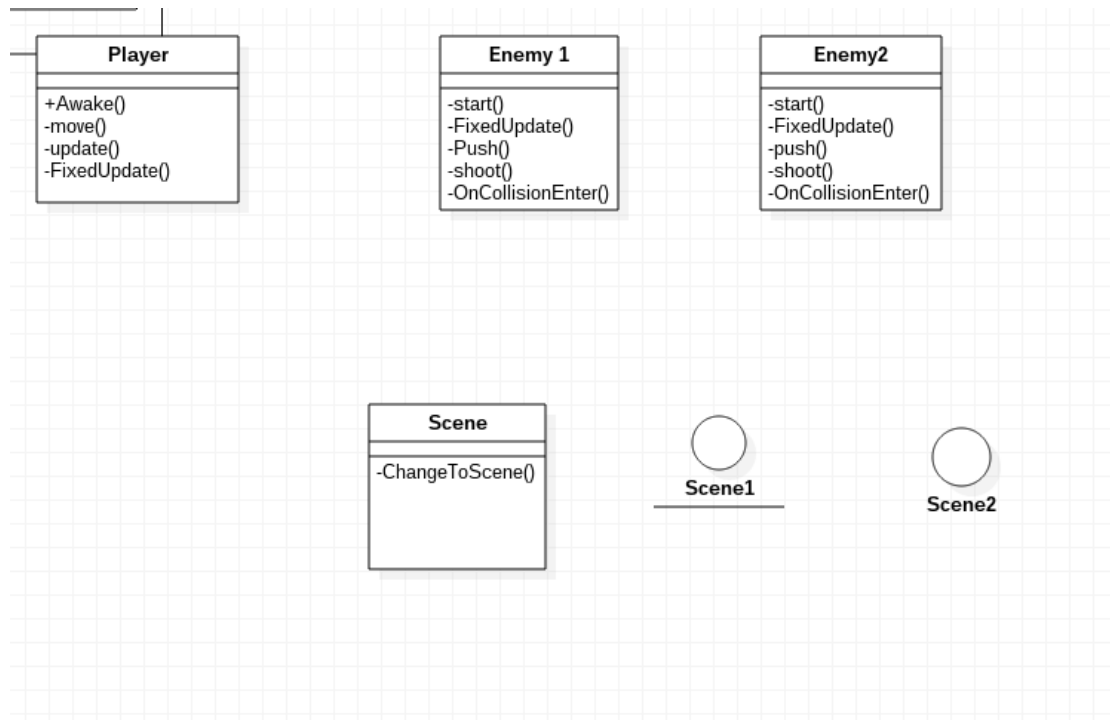
CALCULATE:  
 $Q[\text{state}, \text{action}] = (E + \text{MAX} * 0.8)$  , SUCH THAT Q(4,3) STORES THE R.H.S VALUE AND Q MATRIX  
UPDATED.  
ALSO SET:  
count =count +1 (initially count=0)



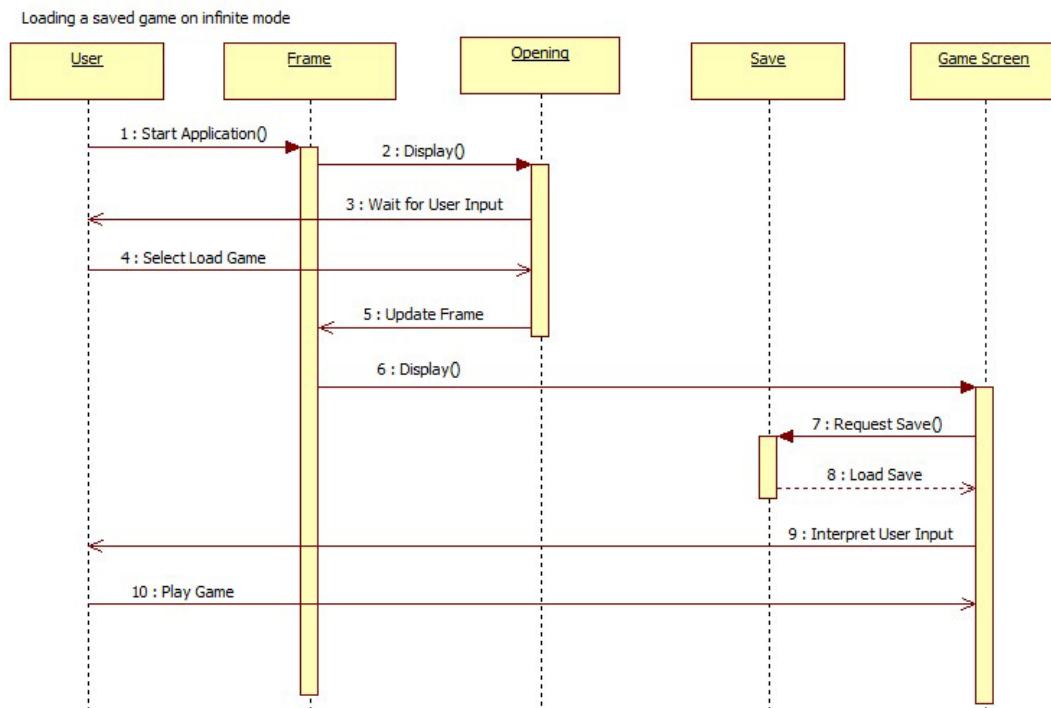
## Use case diagram



## Class diagram



## Sequential diagram



## Snapshots



## **Gantt Chart**

## **REFERENCE**

We had many online and offline references such as:

- Youtube
- Pdf files on C# programming from internet.
- Tutorials on unity game development.
- Q learning example.
- Books on C# programming language.