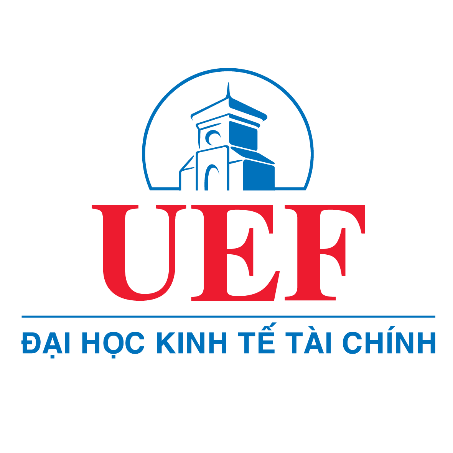
**UNIVERSITY OF ECONOMY - FINANCE HO CHI MINH CITY**

**FACULTY OF INFORMATION TECHNOLOGY**

**\*\*\*\*\*\*\*\*\*\*\*\***



**FINAL PROJECT REPORT**

**PROJECT:**

**BATTLE UNIVERSE GAME**

**Lecturer** : NGUYEN VAN TAN

**Source** : DESKTOP APPLICATION DEVELOPMENT

**Class** : : 221.ITE1264E.B04E

**Group** : 2

*Tran Cao Minh Thang 205051813*

*Ho Lam Gia Khanh 20505180**5*

*Ho Chi Minh city, December 2022*

**TEACHER COMMENT**

|  |
| --- |
| ***Ho Chi Minh city, ……. December 2022***  **Lecturers** |

**TABLE OF CONTENTS**

[**PREFACE** 4](#_Toc118057844)

[**CHAPTER I: INTRODUCTION TO TOPIC** 5](#_Toc123260969)

[1.1. Reason for choosing the topic: 5](#_Toc123260970)

[1.2. Game Overview 5](#_Toc123260971)

[**CHAPTER II: TOPIC REQUIREMENTS** 6](#_Toc123260972)

[2.1. Project Problems. 6](#_Toc123260973)

[2.2. Gathering Requirements. 6](#_Toc123260974)

[2.3. Analysic of Requirements. 7](#_Toc123260975)

[**CHAPTER III: DESIGN** 8](#_Toc123260976)

[3.1. Model. 8](#_Toc123260977)

[3.2. Design. 11](#_Toc123260978)

[**CHAPTER IV: CODING - IMPLEMENTATION** 14](#_Toc123260979)

[5.1. Create a project. 15](#_Toc123260980)

[5.2. Data Storage Design and Class Library Design. 15](#_Toc123260981)

[5.3. Improved interface and functions in the interface. 17](#_Toc123260982)

[5.4. Build Controls Character & Enemy & Bullet. 27](#_Toc123260983)

[5.5. Handling Game Situations 35](#_Toc123260984)

[**CHAPTER V: TESTING** 39](#_Toc123260985)

[6.1. Test Menu. 39](#_Toc123260986)

[6.2. Test Option. 39](#_Toc123260987)

[6.3. Test Store. 43](#_Toc123260988)

[6.4. Test Game. 47](#_Toc123260989)

[**CHAPTER VI: CONCLUSION** 50](#_Toc123260990)

[6.2. Achievement 50](#_Toc123260991)

[6.3. Development direction 50](#_Toc123260992)

[**CHAPTER VII: SUMMARY** 51](#_Toc123260993)

[**REFERENCES** 52](#_Toc123260994)

**PREFACE**

In today's society, technology is an integral part of human society. Video games are starting to use more artificial intelligence to calculate the most profitable moves and increase the difficulty to make the game experience great. After a while of researching and refering to video game genres, we realized that the combination between space battles and video games are a very interesting idea. The universe is something very new, and there are many mysteries behind it. So we believe aliens are real. After much thought, we decided to make a game about spaceships attacking each other. We have combined current technology, and beautiful, novel technology graphics to create this video game that you can play anytime even when alone and without an internet connection . We'll cover the making of this game in our report below.

***Ho Chi Minh city, 30 December 2022***

***Student group made***

*Tran Cao Minh Thang*

*Ho Lam Gia Khanh*

# **CHAPTER I: INTRODUCTION TO TOPIC**

## Reason for choosing the topic:

The universe is a topic that many people care about and love because of its mystery and attraction. Among those who care and love are also us. Therefore, we have relied on that to build the in-game context in which the player will play the role of the defender of the Earth and attack those who are hunting the Earth with bad intentions.

## Game Overview

* **Prepare**

To prepare for the game, players need to equip themselves with a spaceship, and the type of ammunition used for the battlefield.

For enemies there will be a default one spaceship and can use a variety of beautiful neon.

Besides, we prepare a special space for the battlefield.

* **Rule :**

The player will control the character with the main buttons that we implement in the code, the movement buttons we will provide for the W button is to go up, S is to go down, A is to go left, D is to go right. And to shoot bullets, the player will click the left mouse button.

The gameplay is very simple, players just need to move their spaceship safely and attack the enemy without being hit by enemy bullets. The game will not have a win mode, the game is for the player to earn enough coins and buy all the items, meaning the player has won and does not need to continue playing.

# **CHAPTER II: TOPIC REQUIREMENTS**

## Project Problems.

* Program a simple game program.
* Code language: C-Sharp.
* Code Tool : Visual Studio 2022.

## Gathering Requirements.

* + Play games with computers(PVE Easy).
  + Game interface design, layout, layout, distribution of functions.
  + Handling control situations.
  + Handling the action mechanism (motion, animation, ...).
  + Menu, Options, Store interface.
  + The store includes product lists: Gun Ammo, Spaceship.
  + Stores a list of item ownership.
  + Store coins to buy items.
  + Game interface displays full information (Points, Coins received).

## Analysic of Requirements.

* Function menu includes: Play game (Go to the interface to start playing the game), Option (Select game equipment), Store (Buy items), Help (How to play).
* Game interface: Player's Health (When the game is over), Earn Coins, Points, Replay the game (When the game is over, the player can press the play button again), Return to Menu (Return to the menu to be able to perform the procedures). Options, Store).
* Game mode: Create a spaceship for the player to control, and the enemy's spaceship moves and attacks the player, the game ends when the player runs out of health, at the end the player is added the number of coins earned in the process. play program.
* For the Option interface, we will add optional bullet and spaceship functions. The interface will have information about coins, spaceships, bullets that the player owns, display the correct spaceship that the player is using and include buttons to select and switch the interface. In those buttons, there will be 2 buttons to switch the interface, the first is to return to the menu, the second is the bullet selection interface.
* Regarding the bullet selection interface, only shows all the bullets you own, and the button returns to Option.
* The shop interface, we need to display all the items, if the item has an owner, it will be "owned", otherwise it will show the selling price, price and name of each type of spaceship, we will process as data, just read our data file to output the price. And the same goes for bullets. And we will make a table showing successful purchases or failed purchases sent to players. In this interface, there will be an additional buy button and interface switch button.

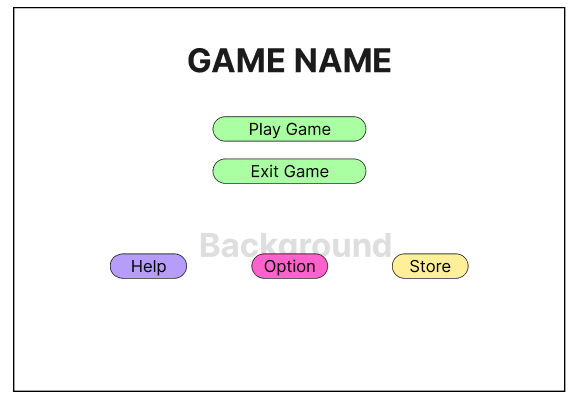
# **CHAPTER III: DESIGN**

## Model.

**- Menu Interface.**

According to the analysis above, this interface we will need a Background and include buttons to handle interface transitions. We will have :

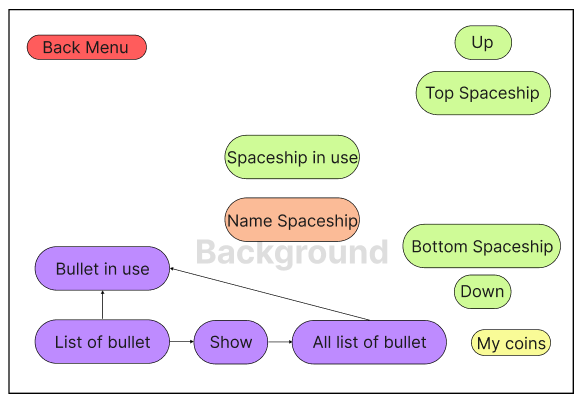
* Play Game will switch to the player interface to start playing the game.
* Option will give players the option to equip their items.
* Store to buy unowned items.
* Help for instructions on how to play the game.
* Exit to exit the game.



**- Option Interface.**

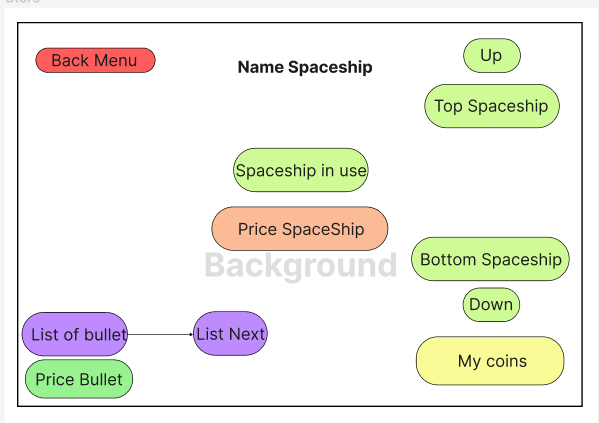
In the Options interface, first we need to have an area showing the spaceship we own, and we will name it SpaceShip in use. Along with that, the name of the spaceship will appear below. For TopSpaceShip, BottomSpaceShip will show 2 spaceships the player owns, Button Up and Down to see the next owned spaceship.

Bullet is similar, there will be an area to display the selected bullet, along with a bullet list, because this list will not be able to display all, so we will add a Show button, it will change the interface. so that OptionBullet shows all ammo the player owns. There is also an area that shows the player's coins and a button to return to the menu in the upper left corner.



**- Store Interface.**

At the Store interface will be similar to Options but will add buttons that show prices and players can click on them to buy. For the bullet display button, it will not switch to the OptionBullet interface, but will display the next list of all bullets, each displayed list will have 4 bullets appearing.



## Design.

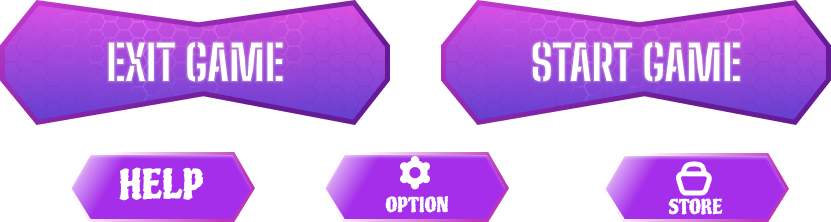
Based on the above model analysis, we have determined and proceed to design the interface.

**- Menu Interface.**

* **Background.**



* **Button.**



**- Option Interface.**

* **Background.**

A picture containing shape

Description automatically generated

* **Button.**

A blue sign with white text

Description automatically generated with low confidence

**- Store Interface.**

* **Background.**

A picture containing icon

Description automatically generated

* **Button.**

A picture containing graphical user interface

Description automatically generated

# **CHAPTER IV: CODING - IMPLEMENTATION**

## Create a project.

To start executing the code, we first have to create a project and define the project that we need to work on in the Visual Studio 2022 tool. After thinking for a while, we decided to get going.

Windows Form application because it is easy to use and suitable for simple games. To create this project do the following:

- Step 1 : Open Visual Studio 2022 tools

- Step 2 : Select Create a new project.

- Step 3 : Select Windows Forms App(.Net Framework) and Click Next .

Graphical user interface, text

Description automatically generated

- Step 4 : Enter a name for the project and select the path to save the project and then Create.

## 5.2. Data Storage Design and Class Library Design.

In our project there are shops, and select items, to have those functions we must first have a database to store the player's item ownership information. So we need to determine how to handle the data. After some thought, we decided to use the database using the INI file. And then because of the requirement to use DLLs, we decided to execute the code to read and write ini files using DLLs.

The data we need is: Data (Save player coins), SpaceShip (Save information about SpaceShip ownership, Name SpaceShip), and Bullet (Save player ammo ownership). And we save at the path : debug\\Data\\[ name file.ini].

Graphical user interface, application

Description automatically generated

To create a DLL file, we first have to create one more project, which is the Class Library (.Net Framework) project, then create a DataIni class. We execute the code in the Class Library as follows:

Text

Description automatically generated

Next we Rebuild Solution, the project will automatically output the DataIni.dll file in the debug folder of the project. Then we include the library in the game project we are building, to use the library we do the following:

* **READ**(path, List of data, name data) : For example, I want to get the Coin data at the DataCoins directory in the path debug\\Data\\Data.ini. In the Data.ini file there will be the following data :

Graphical user interface, application

Description automatically generated

**CODE** : READ("Data\\Data.ini", " DataCoins", "Coins")

**Answer** : 2

* The same, similar : WRITE(path, List of data, name data, data) CODE : WRITE("Data\\Data.ini", " DataCoins", " Coins", "10"); Data.ini will update at Data Coins, Coins will be equal to 10.

## 5.3. Improved interface and functions in the interface.

Although we already have graphics for the interfaces, there is still something that has to be built in code for the interface to be complete.

**- Option :**

After we received the graphic data from the designer, we proceeded to put it into the Form, the first thing I want to present is the Option interface. This is the demo interface that we have included: background and buttons, Label, PictureBox. In which the red box boxes that I marked are the areas where the code needs to be executed to execute.

Graphical user interface

Description automatically generated

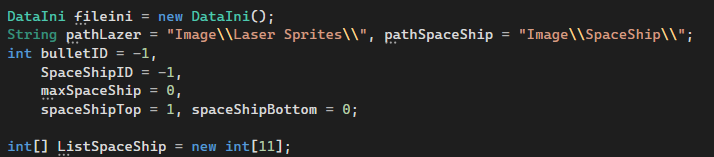
At the Option button, we handle the situation for it to return to the Menu. This is quite simple in code generation, we just need to create a new form from the Menu, and show it.

Text

Description automatically generated

Next in the large frame in the middle is the spaceship display frame that the player is using, the 2 left panes are showing the list of the spacecraft that the player owns, along with 2 Upper and Lower buttons, these 2 buttons will change the display of the list of available spaceships. And just below the big middle frame is a small frame, which will display the name of the spaceship.

First, we will set the necessary user variables, we will set all the relevant variables in the Form Option.



Ok, now we will start working with the 2 boxes on the right, the frame showing the spaceship we own and the spaceship name frame will rely on these 2 frames to display. Along with that is the handling of 2 buttons up and down.

To display the list of spaceships we own, we need to collect all the spaceships we own, for that we will have to rely on data, we have shown how to read and write in use data file ini above. If you notice in the Settings I called new DataIni is the fileini, now we will rely on it to get the data out. Then put the list in ListSpaceShip.

Text

Description automatically generated

Ok, now we have the list, and next we need to display into the 2 frames we mentioned. To display is quite simple, we rely on the list that we have saved, then display the frame according to the list. First, we set the bottom frame as position 0 in the list and the top as position 1 in the list, along with that we based on the information in the data, give the name and the spacecraft that the player is choosing in the middle frame.

Text

Description automatically generated

Text

Description automatically generated

Next we will handle 2 buttons up and down to display the spaceship in the list. This is quite simple, we just need to increase the position and check if the position exists or not. Here's how we do it:

Text

Description automatically generated

We then perform action when clicking on the right panes, they will update the selected spacecraft information.

Text

Description automatically generated

Next we will deal with the red frames below, most of the frames below just need to be displayed, and click the same way as the spaceship selection process above, so we won't repeat it because it does the same thing. on one's own. Now we only have to show the bullet list, and to display it we proceed to create a new Form, which is OptionBullet.

- **OptionBullet** **:**

Initially we have the following interface:

A picture containing graphical user interface

Description automatically generated

The display of the bullet list in possession is quite simple, based on the available graphics, we define to create a 2-dimensional array to display the colored neon bullet list for each cell. First we will save the list of bullets already owned.

Text

Description automatically generated

We then go through the list and display all the ammo that the player has.

Text

Description automatically generated

With that, we've added a frame to identify players who clicked the bullet.

Text

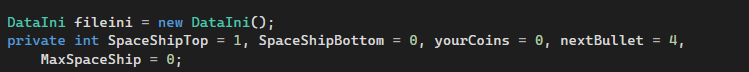
Description automatically generated

**- Store :**

In Store, it is quite similar to Option, and easier to handle because there is no need to get a complicated list of players that will display all the items and players just need to click buy. Already owned items, we will update the data with FileINI of 1 and unowned items of -1.

We're not using arrays like Optional, we'll need to display the entire list of spaceships from the system.

First we will install the necessary variables.



We then execute the code for the up and down button events.

Text

Description automatically generated

Then we process click on the starship list on the right, and display in the middle box, the middle box will show the player information about this ship, the player can click the buy button, if they already own it. owned will show already owned.

Text

Description automatically generated

- At btnNext Bullet to display the next Bullet list, we do the following:

Text

Description automatically generated

Text

Description automatically generated

## 5.4. Build Controls Character & Enemy & Bullet.

**- Character control :**

* **Setup Event :**

To control the character, we need to trigger the following event in the game form: KeyDown (When the player presses the button) and KeyUp (When the player releases the button).

A screenshot of a computer

Description automatically generated with medium confidence

* **Setup Timer :**

Timer Control in C# plays an important role in developing both Client-side, Server-side applications as well as in Windows Services. When using Timer Control, we can repeat events for a specific period of time without the interaction of another thread (Threading). Timer Control in the toolbox has an icon as shown below:

A screenshot of a computer screen

Description automatically generated with medium confidence

After we set the timer, we set the name to gameTimer, set Interval to 20 and Enabled is true.

A screenshot of a computer

Description automatically generated with medium confidence

* **Setup MouseDown :**

MouseDown happens when player clicks, we use this MouseDown for player to shoot bullets.

The installation method is similar to the above, you scroll down to find the Mouse item, you will see MouseDown, then double click on it.



Then we execute the code that spawns bullets here.

Text

Description automatically generated

With the ShootBullet function we will explain below.

* **Build Code Move :**

In the code interface, we set the move state for the character of data type bool : goLeft, goRight, goUp, goDown.



We click on KeyDown, and KeyUp. At KeyDown we set the variable names just set above to true, and KeyUp is the opposite. We use the W button for goUp, S for goDown, A for goLeft and D for goRight.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

At Timer, we will rely on bool data types to determine the character to move, if it is true, it will execute the move, until the player releases the button, it will return false, then the move will also stop.

Text

Description automatically generated

- **Enemy initialization.**

To initialize the enemy, we create an additional class called Enemy.cs .

To create the enemies, we used a PictureBox to create, and added a Timer to handle auto-move for the enemy.

And here is how we execute the code in the Enemy class.

Text

Description automatically generated

We handle Enemy's migration automatically through the Timer as follows:

Text

Description automatically generated

We move the Enemy based on speed, when the Enemy hits the Shape boundary, the speed will switch between positive and negative numbers. And like that it will go back and forth in the position we have set.

For us to use in the game, we call the Enemy class with the following method:

Text

Description automatically generated

**- Bullet initialization.**

We created the same Bullet class as Enemy, but just as a reference, we have 2 targets for bullet spawn, the first is the character that the player controls, the second is the Enemy that automatically controls. In addition, we also have to determine the type of ammo the player uses, so we need to have references to Form, bool Player, and Bullet ID.

Here's how we execute the code:

Text

Description automatically generated

Text

Description automatically generated

As you can see in the image above, we have 2 use cases for Timer, the first is for the character, the second is for Enemy. For the character we adjust the bullet will shoot from bottom to top, and for Enemy it will be from top to bottom. Our code execution is as follows:

Text

Description automatically generated

For us to use in the game, we call the Enemy class with the following method:

Text

Description automatically generated

## 5.5. Handling Game Situations

- **Characters Shooting Enemy.**

After we have handled the control for the character, created the enemy, and finished the bullets, then we proceed to handle the situation where the character shoots bullets at the enemy.

To handle this situation, let's go back to see how we initialize the enemy and the bullet. Here we have added tags for cases to differentiate, we will rely on each object's tag to handle this scenario. Assuming the bullet belongs to the player, the bullet tag is "bulletPlayer" and the enemy has the tag "enemy".

We'll run an endless loop looking for all the pictureBoxes we've created, and check if the tag is "bulletPlayer" and "enemy". When the test is correct, we use cmd IntersectsWith to check if these two objects collide or not.

Here's how we execute the code:

Text

Description automatically generated

- **Enemy Shooting Characters & End Game.**

Similar to the Character Shooting the enemy above, the code implementation is the same. But with this situation we have to handle in end game mode, when the player is attacked with out of blood, it will perform the end game here and will display the end game result along with other buttons.

Text

Description automatically generated

Text

Description automatically generated

- **Drop Coins.**

As in the above code, we have added the function DropCoins(), this is a function that will drop coins when an enemy dies, this coin is used for players to buy items, at the end of the game you will see a paragraph code to save coin to data.

Creating a coin is quite simple, just create a PictureBox and Add it to the form.

Text

Description automatically generated

Then we use a Stopwatch to make it fall from the top, players just need to touch the coin to have it added to their data.

Text

Description automatically generated

# **CHAPTER V: TESTING**

## Test Menu.

Our menu interface is pretty straightforward, and it works as I expected.

A picture containing text

Description automatically generated

## 6.2. Test Option.

When we launch the option, it will show all my information, it shows the spaceship we own, along with the number of coins, bullets.

Graphical user interface, application

Description automatically generated

To see more spaceships I own, I just press the up or down button on the right sidebar, I will try with the up button and at the same time I will click on any spaceship I want to equip it's in the game.

Graphical user interface, application

Description automatically generated

Next we will test selecting bullets, assuming I choose 1 of the 4 default bullets that are showing.

Graphical user interface, application

Description automatically generated

The bullet selection box has also changed, next we will open the entire list of bullets that we own, and choose any type of bullet.

A screenshot of a computer

Description automatically generated with medium confidence

Yes it worked great, after we finished selecting the bullet and went back to the Option interface, it also automatically changed my selected bullet.

A screenshot of a video game

Description automatically generated

## 6.3. Test Store.

We continue to test the Store interface, now we will open its interface.

Graphical user interface, application

Description automatically generated

That's perfect, now we're going to pick up any spaceships and ammo that we don't own and try to buy it and see if it works.

Graphical user interface, application

Description automatically generated

If the player does not have enough coins to buy, it will show a message of failure to buy, and if there are enough coins, it will display a successful purchase.

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

## 6.4. Test Game.

After the player completes his equipment and enters the game, the initial game interface will be:

A screenshot of a video game

Description automatically generated with medium confidence

Players can freely move and attack enemies.

A screenshot of a video game

Description automatically generated with medium confidence

# 

# 

# **CHAPTER VI:** [**CONCLUSION**](#_heading=h.1t3h5sf)

## Achievement

After learning and developing the game "Battle Univese", we have created a simple shooting game. We designed more features for the game. To make the game more interesting, we also created many other features.

* Players can buy other spacecraft, different types of ammunition to experience the game more vivid and interesting.
* Interface Menu, Option, Store, Play easy to see, easy to use.
* Easy character control, adapted to today's users.
* But limited time as well as software support, so we can't develop all other features of the game yet.

## Development direction

* In the future we will increase the graphics, integrate the game to use the Unity engine.
* Improve enemies, making it more difficult for players to play the game.
* Added additional health when the player is low health.
* Displays a list of the highest 3 scores ever played.
* Update added graphics like bullet, spaceship.

# **CHAPTER VII: SUMMARY**

*Sincere thanks to teachers Nguyen Van Tan for enthusiastically teaching, equipping us with useful and valuable knowledge during the implementation of the thesis. As well as creating conditions for us to challenge to implement this interesting topic.*

*Thank you to my classmates for their support, help and encouragement in the process of working on the thesis. Those are great sources of encouragement and encouragement to help us have more energy to complete this thesis well.*

*In a short time, although we have tried and made a lot of efforts, the topic must still have many shortcomings. We are looking forward to receiving your comments and suggestions for our topic to be more complete.*

***We sincerely thank…!***

***Student group made***

*Tran Cao Minh Thang,*

*Ho Lam Gia Khanh*

# **REFERENCES**

1. **https://learn.microsoft.com/vi-vn/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8**