Game Development in C# Programming Language, using Microsoft XNA

Andrei ŞCHIOPU, Mihail KULEV
Technical University of Moldova
andrew.schiopu@gmail.com , mkmk54@mail.ru

***Abstract* — The program was developed that would be able to initialize textures, fonts and sound effects, and that would be able to draw and update them while the program will be running, based on the inputs that would be read from keyboard, thus handling events.**

***Index Terms* — algorithm flowchart, class, frames, layer, object-oriented programming.**

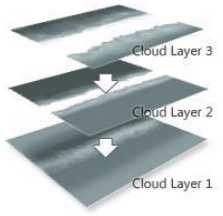
I. INTRODUCTION

The game was elaborated in C# Language. There were multiple data types used, which are mostly defined in XNA library. These data types are helpful in reading input from keyboard, initializing textures and sounds. The program is complex and object-oriented one, thus consists of 6 classes: Main, Animation, Background, Player, Enemy, Laser. The game will also need its content, which basically consists of different images, fonts and sounds that will be used in the game.

II. GENERAL OVERVIEW OF THE PROGRAM

The goal was to create a simple but pretty shooting game. For this to happen we have combined programming skills with drawing skills. The game by itself represents a set of textures, fonts, animations and sound effects, and we had to create a program which combined them efficiently.

Firstly, we start with the player. The player is represented by an object, which describes his behavior, thus, the health, the position on the screen, etc. Our screen is 800x480 pixels, so we had to make sure that the player doesn't go outside these borders.



Another thing that had to understand is that it is not the player which is moving in the sky, it is the sky that is moving, and the player moves only alongside within the screen; therefore, we had to have an object which had to describe the moving background so that it would create the illusion of flying player. So, the thing we needed was to draw the image of the sky in 3 layers and then save them apart.

Of course, we needed enemies, and, to simplify, those were made as just rockets flying towards the player. Moreover, these will be accompanied by sound effects.

Another thing is the animation. The animation is in fact a set of images that are drawn in order, as frames, merged in one image, so, in other words, in order to play the animation we will need a rectangle that will jump from one frame to another, in this manner creating an illusion of animation.

Fig.1. Set of Frames for Explosion Animation

III. PROGRAM FUNCTIONS AND VISUALIZATION

The functions that we implemented in different classes are almost the same, (except for the Main class) and they reduce to functions: Initialize, Update, Draw. In case of Main class we also have LoadContent, which is called to load all of the textures, fonts, and sounds in the game, and, consequently, functions for adding and updating player, laser, collisions and explosions. Now, how do the main 3 functions work? First, we call the Initialize function, which initializes the content that will be used at first, thus the default textures, first frame of the animation, position of the player, etc. The Update function technically does the following: It changes the value of the health if needed and jumps from one frame to another in case of animation. The updating is usually done once every 60-th part of second. The last, but not the least, is the Draw method. What it does, is it draws all of the textures, enemies, player, background, etc. Well, the algorithm of the application can be easily explained by the following flowchart:

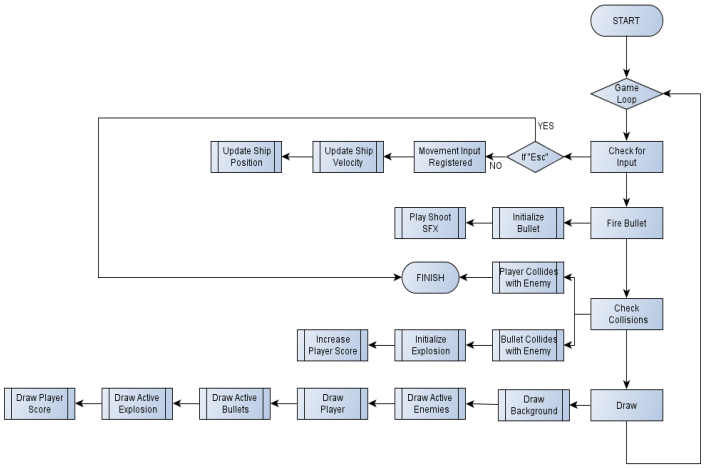


Fig.2. Algorithm Flowchart for the Game

In the end, the result is a game with nice audio and video characteristics:

Fig.3. Game Screenshots

IV. CONCLUSION

Implementation of the program was not an easy task, but when you get in the topic, you can see that many things look just alike. The first thing when doing a program, a game especially, is thinking and analyzing the algorithm you are going to use. The best way to do it is to draw a generalizing flowchart. It is also important to get a hold of all the content that will be needed in the program and better have more than you will need. C# is a very good language for implementing the algorithm for game programming as it has XNA Library at your disposal that will help you a lot, because it has multiple data types for reading input, initializing sound effects and textures.

ACKNOWLEDGMENTS

The textures of the game were drawn by Ilie Ciorbă.

REFERENCES

[1] « Microsoft XNA Game Studio 4.0 » Rob Miles
[2] « Beginning C++ Through Game Programming, Third Edition » Michael Dawson
[3] XNA Game Studio 4.0 Available: http://msdn.microsoft.com/en-us/library/bb200104.aspx