ÇUKUROVA UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

GRADUATION THESIS

Game Development with Unreal Engine

By

2016556506

Buğra Sarıkaya

Advisor

Res. Asst. PhD Barış Ata

June 2022

ADANA

ABSTRACT

Game development with Unreal Engine via C++ language, was studied on this project. Despite lack of documentation and unofficial tutorials about Unreal Engine, game project was developed by using nearly pure C++ methods and contents of visual part, were drawn by inspiring other games. But developed game cannot be published for marketing because of copyrights of used sounds and also, game cannot compete with games of today due to low count of contents. However, analysis of this thesis, can be useful for developing advanced video games with Unreal Engine.

CONTENTS

1. INTRODUCTION

Game development which is the subject of this thesis, has been chosen for specializing in C++ language by developing popular software works.

Game industry is one of the popular software sectors. From all over the world, game developers publish their works and their productions contribute software technology because video games are basically simulations and their improvements about similarities to the real world, create qualified algorithms.

Who wants to develop a game project, also advances their abilities to solve real world software problems and for becoming an expert on C++ language, game development is a good choice because generally video games are written in C++.

One of the game engines which are written with C++, is Unreal Engine and it is chosen due to its popularity. It supports two programming methods and one of these, is using C++ methods but solutions of C++ problems, are difficult to research.

Game development does not consist only software. For improving similarities to real world, its visual and auditory parts need to design by artists. Artworks of visual design can be created by using graphic editors and Aseprite is a popular graphic editor for pixel art which is an ageless art style for video games.

In this project Unreal Engine and Aseprite, were chosen and occurred problems while developing, were resolved by researches.

Digital versions of project, were uploaded to a GitHub repository [1]. In “executable.rar” file, there is playable version of game without editor. “playthrough_video.mkv” files is a short playthrough video of the game. “thesis_report.pdf” is the current document. Update logs were written in “update_logs” folder of this repository. In this folder, there are too many benefited sources but instead of referencing all of them in this report, their important ones were referenced.

2. PROCESSES OF DEVELOPMENT

In this chapter, processes of game development, are explained.

2.1. Concepting

Developing a C++ software, was determined to become a C++ expert. A popular sector in C++ language was chosen. Chosen sector was game industry because of its raising popularity.
Retro shooter was chosen as genre of game project. Its retro part was chosen because of simplicity of old games and shooter part was chosen because FPS video games were thought most exciting games.
Mostly Viscerafest game [2] was inspired for developing this game project because its simplicity is at a level which satisfies the developer of the project.

2.2. Preparing

A research was done for detecting required technologies. Selected game engine whose name is Unreal Engine [3]. Its latest version which is 4.27.2, was installed to local computer. While developing of this project, Unreal Engine 5 was newly published but it did not be chosen because of its instability.
Unreal Engine is powerful game editor and it needs to supply high memory. To make faster building process, RAM capacity of the computer, was increased to 16 GB.
Programming language of Unreal Engine, is C++ and for a C++ IDE, Visual Studio [4] was chosen because of its detailed user interface. Visual Studio has installed previously and while developing, it was upgraded to Visual Studio Community 2022 version 17.1.6.
Visual style of this project, is pixel art and Aseprite [5] was chosen as a convenient pixel art editor for learning easy.
For editing audio sound, Audacity [6] which is a free audio editor was installed.
GitHub [7] which is an online Git provider was chosen for uploading project files and tracking code changes because it has a feature whose name is Gitignore and it ignores default game engine files for decreasing upload size.
GitHub Desktop [8] program which enables controlling Git repository on PC desktop, was

used it has installed to computer previously. Game files were uploaded to a GitHub repository of the developer. Update logs and solutions of tedious issues, were saved in “update_logs” folder of this repository.

2.3. Learning

A small research was done for learning that count of existed Unreal Engine tutorials, is sufficient and it was witnessed that there are a lot of tutorials but most of them are Blueprint tutorials. Blueprint is a visual scripting system and it is a supported programming method in Unreal engine but with reason of wanting to develop a C++ software, Blueprint was not preferred. However, a method for benefiting numerous count of Blueprint tutorials, was researched and it was found that Unreal Engine has a feature which shows C++ equivalent of Blueprint node. If developer does right click over a Blueprint node, an option whose name is “Goto Definition” appears in Blueprint page but it is disabled in default installation of Unreal Engine. To enable it “Editor symbols for debugging” option was checked for downloading 36.5 GB of Unreal Engine developer content. This feature worked great because lots of Blueprint tutorials were not wasted.
An official tutorial [9] of Unreal Engine about FPS with C++ language, was found and its instructions were implemented successfully. This tutorial was in beginner level and to develop advanced features, it was decided to examine other tutorials. While applying directives of other examined tutorials, a lack of knowledge about general concept of game development in Unreal Engine, was realized. After this awareness, it was determined to only examine tutorials without implementing their instructions. At least 37 hours of content about game developing in YouTube, was watched. YouTube channel names of most benefited video tutorials, were “Hitbox” [10], “reubs” [11], “Unreal Kafası”, [12] and “freeCodeCamp.org” [13]. Watching tutorials before starting to developing, served the purpose of knowing what doing.

2.4. Developing

Idea of game project, was actualized by using learned features about game developing. For solving unforeseen software problems, official documentation of Unreal Engine [14], forums of Unreal Engine [15], Stack Overflow [16], GitHub and Reddit [17] were benefited but they were not enough all time. For finding solutions and adding features, researches were done in source codes of the game engine. All of researches in these sources took a lot of time. Sometimes investigations took one day. Due to wasting plenty of time for researching, addition of some features, was delayed. When no solution were found, the developer created his own solutions by writing unique algorithms. Nevertheless, some uncritical problems were not be solved.

For developing visual parts of game project, various artworks were drawn on Aseprite and some special attributes of this program, were learned in posts of Aseprite Community. Drawn weapon and enemy artworks were inspired by images from DeviantArt [18], ArtStation [19] and FunnyJunk [20] but due to lack of time, count of these drawn artworks was low. Multiple textures were drawn for level designing. Project files of plane textures, whose number is 16, were packaged for reducing number of files and only one of them was used. Used plane textures were inspired by wall textures from Viscerafest. HUD and menu button textures were drawn without inspiring from anything because they were simple. All created artworks were stored in “artworks” folder of game project.

Audio files of game project, were obtained from works of artists which were not involved into the project. Unique sounds could not be created because of lack of time. Several game sounds from Reddit, GameBanana [21], SoundCloud [22] and Bandcamp [23] websites and a previously installed game whose name is Sang-Froid - Tales of Werewolves [24], were used but their copyrights made the game project unmarketable. Some obtained sound were edited by decreasing their duration on Audacity. All audio files were stored in “sounds” folder.

Font type of text in game, was available in local computer. Its name is “Press Start 2P” and it was stored in “fonts” folder.

2.5. Testing

Values sounds, spawning details, navigations, damages, speeds and other kind of features were changed for improving game experience.

After exporting game project for playing without editor, some unseen bugs in editor, occurred on executable version of game project. Fortunately, they were solved

3. USED PROGRAMS

In this section used programs are explained.

3.1. Unreal Engine

Unreal Engine is one of the popular game engines written in C++ language. It was developed by Epic Games. It has different versions and first version was published in 1998. Nowadays, Epic Games are working on fifth version.
It can be downloaded from Epic Games Launcher. Version 4.27.2, was installed and its total size was 75.0 GB. Epic Games Launcher supports downloading compressed software and compressed size of Unreal Engine, is smaller than 20.0 GB.
In default installation, some features are disabled and they can be enabled by downloading them on the launcher.
It is powerful engine to construct complex graphics and because of this it needs high memory. The game engine can works with low memory but processes will be slow.
It is a free software but if published game sells over $1 million, firm gets %5 of it for royalty [25].
It provides two programming method which are Blueprint and C++. They can be used as hybrid.

3.2 Epic Games Launcher

It is a free application of Epic Games [26]. It provides game marketing, downloading game contents and organizing game contents. Different versions of Unreal Engine, can be downloaded on this platform. Also Unreal Engine contents are published for marketing on this platform and every month, Epic Games gives some Unreal Engine contents for free.

3.3. Visual Studio

Visual Studio is an IDE which has detailed user interface. Visual Studio Community 2022 version 17.1.6, was installed to local computer.
To apply changed code, developer must build Visual Studio project. While developing an

Unreal Engine project, its building feature is sufficient but if Unreal Engine crashes, developer can use debugger feature to start the Unreal Engine editor via Visual Studio.
But its error notification system works faulty while developing an Unreal Engine project. Sometimes it shows errors at irrelevant locations and sometimes it completes building successfully, despite bug reports. To overcome such errors, developer should have experience about game developing with Unreal Engine.

3.4. Aseprite

Aseprite is a popular pixel art editor. It can be bought in Steam which is another game software platform, with price of 32.00 TL. This price is cheaper than price of its rival which is Adobe Photoshop. Price of Adobe Photoshop is $20.99 for monthly [27], and it supports advanced features for all designing types. Because Aseprite only supports pixel art designing, it is an inexpensive software and its features can be learned easily.
It has animation, sprite organization and mirroring features but its marquee tool does not work properly with mirroring feature.
Artist must resizing to a bigger size their craft while exporting otherwise image will be very small.
Its theme of user interface, is changeable.

3.5. Audacity

Audacity is a free audio editor.
Properties of imported audio file are editable on this program and audio file can be exported as a different file type.

4. BLUEPRINT AND C++

4.1. Comparison

Unreal Engine allows two programming method and they were compared in this section.
Blueprint is visual scripting system and it means it can be used without high experience of programming. It does not need to include library but for extra features, plugins are installed. Its developing process is faster but it runs slow because there is a virtual machine between Blueprint scripts and runtime functions [28]. To acquire speed, C++ methods should be used. Blueprint scripts consist of nodes and they are connected with arrows. This formation is similar to logic circuits. Due to big size of nodes, excessive amount of nodes makes system hard to trace. C++ classes can be converted to Blueprint classes and details of Blueprint classes, are visible. Their visibility feature is helpful for previewing.
C++ is other method of Unreal Engine. It needs high programming knowledge. Developing process is difficult and it takes too much time. To use functions and features of Unreal Engine, relevant libraries and modules must be included. Unreal Engine restrict usages of their libraries which means pure C++ methods, are not utilizable all time. However, it runs very fast. Developer can create unique algorithms. Developing via C++ methods, is advanced and it helps to become a coding expert.

4.2. Conclusion of Comparison

For gaining speed and improving coding abilities, C++ methods were used but Blueprints were used for previewing constructed C++ classes.

5. UNREAL ENGINE CODING COMPONENTS

In section various C++ coding components of Unreal Engine, were explained.

5.1. Classes and Structs

In source of the game engine, there are several classes and structs. They have special properties such as functions and variables. For using Unreal Engine classes and structs, their libraries are included but some of them do not need including their libraries. Their libraries can be found in online documentation of Unreal Engine.
Classes inherits other classes. Name of classes, begins with a specific letter which represent their inherited base classes. For example names of classes which inherits Actor class, begins with “A” letter. There are class names which begin with “F” letter but generally struct names begin with “F” letter. If this letter procedure is not set properly, IDE gives error.
Some classes require modules to be added into the build file of project. Module names can be found in documentation of Unreal Engine.
Majority of classes, have constructor, however, “new” keyword of C++ is not allowed to use by Unreal Engine. Classes which do not have constructors, cannot be initialized.
In the editor, user defined classes and structs can be created. Their base class can be selected. There are many number of classes but some of them, have “final” C++ keyword which disallows child classes. A created class whose base class has a “final” keyword, cannot be utilized as wanted.
C++ classes can be converted to Blueprint classes, but Blueprint classes cannot be converted to C++ classes.

5.2. Macros

5 types of macros, were used in this project and their names are “UFUNCTION”, “UPROPERTY”, “UCLASS”, “USTRUCT”, “UENUM”, “FORCEINLINE” and “UE_LOG”. The editor recognizes code compounds which are defined with these macros, except “UE_LOG”, but some code compounds cannot be used with them. For example “FTimerDelegate” cannot be used with “UPROPERTY” macro. Macros can take specifier

keywords for detailing. Some specifiers allow code compounds to appear in its Blueprint classes. In this project, specifers never be used.
"UFUNCTION" macro is for functions. Virtual functions which are inherited from base classes, do not take this macro. "UPROPERTY" macro is for variables also it provides garbage collection which is a system to delete objects which are no longer referenced. "UCLASS" macro is for classes. "USTRUCT" macro is for structs. "UENUM" macro is for enumerations. "FORCEINLINE" macro is similar to "inline" keyword of C++. It provides declaring functions in one line. "UE_LOG" is for logging.

5.3. Source Files and Header Files

After creating a C++ class in the editor, two code files are added into folder of code files. Their names are same as class name. One of them, is source file whose extension is ".cpp" and other one is header file whose extension is ".h". Both of them, begins with a comment line about copyright.
In source files, their header files and if there any other header files are included, functions are defined and written algorithms are implemented.
In header files, libraries are included. Variables, functions and if there any classes, structs and enumerations are declared.
In this project some variables which were used statically, were initialized. Reason of initialization, is finding them effortlessly because header files are simpler than source files. Declarations were sorted in order. Functions are on top and variables are on down sides. Also they were sorted in alphabetical order amongst themselves.
While developing, circular dependency problem was occurred. It happens when a code file includes a header file and this header file includes other code file. There are two solutions. One of them is using an interface code file and other one is using forward declarations. Using and interface method could not be implemented but forward declaration method which is declaring class name in header file and including header file of class in source file, was applied successfully in code files of the project.

5.4. Build File

Build file is a C# file whose extension is “.cs”. Used module names are added into this file for activating them. Function and properties of non-activated modules, do not works and IDE gives error.

5.5. Fundamental Functions

In source files, there are definitions of fundamental functions if developer declared them in header files. Name of used fundamental functions, are constructor, “BeginPlay”, “StartPlay”, “Tick”, “Init” and “EndPlay”.
Constructor is a default special function of C++. It is called after building source codes in Unreal Engine. In this function, class properties and variables are set, external objects and classes are loaded. If any bug occurs in this function, Unreal Engine editor crashes. Other ones are virtual functions. “BeginPlay” is called at the beginning of playing level. “StartPlay” is called after calling “BeginPlay” function. “Tick” is called every frame after calling “StartPlay” function. “EndPlay” is called after destroying object of class. “Init” function is special for Game Instance and it called while initializing Game Instance.

5.6. TArray

TArray is array system of Unreal Engine. It is similar to vectors of C++. Size of TArray, can be changed dynamically but its construction cost is high. Because of high cost, pass by reference method must be used instead of pass by value method for not slowing game while moving values of TArray into functions.
Unreal Engine does not allows multidimensional TArray type arrays. To utilize multidimensional arrays, an alternative solution which is using structs which have arrays, was implemented.

5.7. Logging

Logging is useful for detecting problems. “UE_LOG” macro was used while developing this project for logging. It has several parameters and according to its arguments, it writes a specific line into window of Output Log.

6. COMPONENTS OF GAME PROJECT

Components of game project, were explained in this section.

6.1. Project Settings

There is a settings section with same name in Unreal Engine editor. Some features such as maps, Game Instances, Game Modes, Pawns, HUD's, inputs, navigation details, anti-aliasing setting are set on this section. However, they are set also via C++ codes but some of them could not be set via C++ methods.
Created Game Instance and default Game Mode of Unreal Engine, were set on Project Settings and they worked properly in Play Mode of the editor. In source code of created Game Instance whose name is "game_instance", default Game Mode was set to change to created Game Mode while starting the game but after exporting the game project, it did not work as expected. Because of this problem, one of the created Game Modes, whose name is "game_mode_base_main_menu" was set on Project Settings instead of setting default Game Mode of Unreal Engine.
Other Unreal Engine projects in GitHub, were examined and it was noticed that at the beginning of all created code files, there are different comment lines about copyright. This issue was researched and "Unreal® Engine, Copyright 1998 – 2022, Epic Games, Inc. All rights reserved." sentence was added to Copyright Notice section of Project Settings. It enables to add that line at the beginning of the newly created source codes files. Unreal Engine wants developers to add that line for publishing game [29].
Editor Startup Map and Game Default Map, were set as "level_main_menu" which is the first level of the game project.
In Navigation Mesh section of Project Settings, Runtime Generation option was set to Dynamic and in Navigation System section, a Supported Agent was added.

6.2. Game Instance

Game Instance starts to run after executing game program and it stops after closing game.
A Game Instance whose name is "game_instance" was created and it was set as Game

Instance Class of game project on Project Setting. In its source files, resolution of game, was set to resolution of current computer which enables to play the game. Window mode was set to full screen. VSync which is a graphic technology for preventing screen tearing, was enabled. Screen tearing occurs when refresh rate of monitor is low than frame rate of game. Anti-Aliasing feature which smooths edges and corners, was closed because pixel art contents should be sharp. Also it increases performance.
Several codes for changing Game Mode, was written. They worked in editor but they did not work in exported game.

6.3. Game Modes

Game Mode is most important component in a game project. Pawn, HUD, Player State, Game State, Player Start and some other kind of objects are set and controlled in this component. These objects can be set in Maps & Modes section of Project Settings and also, via C++ codes. Multiple Game Modes are allowed in Unreal Engine and in this project, there are two Game Modes. In source codes of Unreal Engine, there are two Game Mode classes whose names are "GameModeBase" and "GameMode". Because "GameModeBase" is parent class of "GameMode", types of created Game Modes, were set as "GameModeBase".

6.3.1. Game Mode of Main Menu Level

"game_mode_base_level_main_menu" is the Game Mode of first level whose name is "level_main_menu". It loads HUD of main menu in its constructor.

6.3.2. Game Mode of Arena Level

"game_mode_base_level_arena" is the Game Mode of second level whose name is "level_arena".
In its constructor, it loads combat HUD, Pawn, Player State, Game State and Player Start classes are. Paper Player is set as Default Pawn. It loads object of texture materials. Loaded materials are imported material pack structs and materials pack structs are imported face structs of planes which is spawned in "BeginPlay" function.
Material textures were inspired wall textures of Viscerafest game. Also a lot of textures were drawn but only textures of type 14 pack, were used for level designing of arena level.
In "BeginPlay" function of "game_mode_base_level_arena", structures are spawned and they are designed with loaded materials, by using written functions whose names are "set_num",

“spawn_box”, “spawn_stair”, “design_prism” and “design_stair”. In arena level there are 4 kind of plane structures. Count of room structure is 1, count of floor structures is 10, count of junction floor structures is 13, count of stair structures is 8 and count of pillar structures is 4. Plane structures are consists of planes and totally 18476 planes is spawned. If there is a default Player Start in level, it is destroyed for implementing created Player Start. Existed Nav Mesh Bound Volume, is moved and scaled properly for setting navigation system.

In “set_num” function, arrays of boxes, are resized.

In “spawn_box” function, boxes are spawned according to their array size, directions and locations. Boxes are basically plane structures which have 6 plane faces. Box properties are important to build level parametrically. Because boxes were not be arranged by using drag and drop method in editor, they are spawned via C++ codes. Spawning was not be arranged blindly. All properties of every boxes are connected each other box properties which are size, location, and direction. After spawn a box or stair, with “spawn_stair” function, algorithm takes last plane element of box and next box was spawned according to this last plane. Parametric spawning system is that if developer wants to change a size of array, sizes and locations of other arrays were changed by algorithm. For example if height of room box is increased, all heights of columns, are increased.

“spawn_stair” function is similar to “spawn_box” function and in this function, settings of scaling and decreasing count of side plans, are applied to spawn stairs. A stair is a box whose side planes are not spawned.

In “design_prism” and “design_stair” functions, loaded materials are assigned to spawned objects.

In “ChoosePlayerStart_Implementation” function, “player_start” is spawned on a certain location.

In “StartPlay” function beginning time is saved for calculating elapsed time.

In “Tick” function current time is saved. By subtracting beginning time from current time, elapsed time is calculated. After several time conversions, a system is built for spawning enemy every seconds. Spawning locations are junction floors. There are 13 locations and an algorithm was written to prevent spawning if player is near the locations.

6.4. Game State

Game State is generally used in multiplayer projects. On this system, game information, such as high scores, are tracked and controlled. In this project, it was used for developing game components separately.
In constructor function of created “games_state_base” class, Save Game class is loaded.
In “BeginPlay” function, “load_game” function is called.
In “load_game” function, high score informations are loaded from save slot in local computer. If there is no save slot, a warning appears in Output Log window of Unreal Engine editor but it does not crash game.
In “compare_scores”, current score which was obtained from Player State, and loaded high scores are compared. If current score is higher than high score or current score is equal to high score but elapsed time is lower than loaded elapsed time, current score information which is score and elapsed time, is set as high score information and it is saved to a save slot whose name is “slot_0”.
Location of save slot is “Saved\SaveGames” in game project folder and for exported game, its location is “C:\Users\<UserName>\AppData\Local\<UE4ProjectName>\Saved\SaveGames”.

6.5. Player State

Player State is generally used in multiplayer projects. In Player State, Player information is tracked and controlled. In this project, it was used for developing game components separately.
In “BeginPlay” function, start time of player is obtained from player pawn.
In “Tick” function, current time and player health information which is obtained from player pawn, is tracked. Elapsed time is calculated by subtracting start time from current time. If there is no longer a player pawn, it destroys itself.
“EndPlay” function is called while destroying Player State. Score information is moved to Game State and “compare_scores” function of “game_state”, is called. Combat HUD is changed to ending HUD.

6.6. Save Game

Save Game system, is used for saving and loading game information in local computer.

In header file of created “save_game” class, there are some score variables and functions. They are used by Game State.

6.7. Build File

At the start of developing, “game_project.Build.cs” file is created by Unreal Engine. To active modules, module name are added into this file. Names of all added modules, are “Core", "CoreUObject", "Engine", "InputCore", "AIModule", "Chaos" and "RenderCore", "NavigationSystem".

6.8. Input Settings

Input settings can be set in Project Settings but coding method was preferred.
A C++ class whose name is “input_settings”, was created and in this function, firstly existed inputs were deleted and then, new inputs were added. This deletion is important to prevent running system unexpectedly due to unused inputs.
Inputs are mouse inputs and keyboard inputs. Left mouse button was set to fire weapon. Space bar was set to jump. Escape was set to open ending menu. It destroys player pawn for ending level. “W”, “A”, “S” and “D” buttons were set to move player. Mouse movement inputs were set to change looking view.

6.9. HUD

HUD is abbreviation of Heads-up Display and it is basically status indicator system.
There are 3 created HUDs in this project and their names are “hud_main_menu”, “hud_ending” and “hud_combat”. Menus are UI which is abbreviation of User Interface but in this project menus were created as HUDs.

6.9.1. Main Menu HUD

“hud_main_menu”, is the HUD of main menu. After opening “level_main_menu”, this HUD is implemented. It has 2 buttons. Start button is for starting arena level and quit button is for quitting from game. In this menu, a music is played and buttons have sounds.
In the constructor of “hud_main_menu”, an audio component is created and music sound object is loaded to this component. There are 3 sound objects and they are main menu music, clicking button sound and cursor over button sound. Button sounds is not loaded to created audio component. Button texture objects and font object is loaded. Game Mode class of arena

level, is loaded for using it for opening arena level.

Played song is “Karl Casey – Broken” and it was obtained from BandCamp [30] by using an online downloader. It was chosen because it was similar to combat sounds. Button sounds were obtained from menu sounds of Half-Life 2 game [31]. Sound pack of Half-Life 2 was obtained from a webpage of Reddit [32]. Font name is “Press Start 2P” and it was obtained from local computer. Its size is 25 and size was set on Unreal Engine editor.

In “BeginPlay” function, cursor and click events are enabled for interacting with buttons. Music sound is set to play.

“DrawHUD” function is similar to “Tick” function. It is called every frame for displaying HUD. At the beginning of this function, screen resolution values is obtained for positioning visual HUD elements. Element details which are locations and size, can be changeable according to screen resolution. Background, button details and text details, are set. Background color is RGB(18, 18, 18) and button text colors are black. Font sizes of button texts, are 25. 2 Hit Boxes which are rectangles to inform events of mouse click and cursor over, are created in this function. If player clicks a button, cursor location values are locked.

In “NotifyHitBoxClick” function, click notifications are processed. If player clicks a button, cursor is locked and click sound is spawned. If start button is clicked, arena level is started and if quit button is clicked, game closes itself. Events are applied after ending click sound for playing sounds until their end. For starting next level, “start_level” function is called and for quitting game, “quit_game” function is called. After clicking event, cursor location values are saved for locking cursor in “DrawHud” function.

In “start_level” function, arena level is opened and it changes current Game Mode to “game_mode_base_level_arena”.

In “quit_game” function, quitting from game, is implemented.

“NotifyHitBoxBeginCursorOver” function is called if cursor is over a button. In this function, button color is changed to its lighter version and a cursor over sound is spawned.

“NotifyHitBoxEndCursorOver” function is called while ending cursor over event. In this function, light button color is changed to its normal version.

In “EndPlay” function, music is stopped.

6.9.2. Ending HUD

“hud_ending”, is the HUD of ending. After death of player, this HUD is implemented. On top, a text which is “GAME OVER”, is appeared. On the bottom of this text, high score and current score, are displayed. It has 2 buttons. Restart button is for restarting arena level and quit button is for quitting from game. In this menu, a music is played and buttons have sounds.

In the constructor of “hud_ending”, an audio component is created and music sound object is loaded to this component. There are 3 sound objects and they are ending music, clicking button sound and cursor over button sound. Button sounds is not loaded to created audio component. Button texture objects and font objects are loaded. Game Mode class of arena level, was loaded for using it for restarting arena level.

Played song is “End of Level”. It is the ending mission music of DOOM Eternal game [33] and it was obtained from SoundCloud [34] by using an online downloader. Button sounds were obtained from menu sounds of Half-Life 2 game. Font name is “Press Start 2P” and it was obtained from local computer. There are 2 sizes of this font. They are 25 and 50. Sizes were set on Unreal Engine editor

In “BeginPlay” function, cursor and click events are enabled for interacting with buttons. High score and current score information is obtained from Game State and they are converted to FString which is the string class of Unreal Engine, after using “calculate_time”. Music sound is set to play.

“calculate_time” function, converts elapsed time to time units as a stopwatch.

At the beginning of “DrawHUD” function, screen resolution values are obtained for positioning visual HUD elements. Element details which are locations and size, can be changeable according to screen resolution. Background, button details and text details, were set. Background color is RGB(18, 18, 18). Font size of game over text, is 50 and its color is RGB(220, 0, 0). Font sizes of high score and current score texts, are 25 and their colors are RGB(220, 0, 0). 2 Hit Boxes are created in this function. If player clicks a button, cursor location values are locked.

In “NotifyHitBoxClick” function, click notifications are processed. If player clicks a button, cursor is locked and click sound is played. If restart button is clicked, arena level is restarted and if quit button is clicked, game closes itself. Events are applied after ending click sound for playing sounds until their end. For restarting arena level, “restart_level” function is called and for quitting game, “quit_game” function is called. After clicking event, cursor location values are saved for locking it in “DrawHud” function.

In “restart_level” function, arena level is opened again and it changes current Game Mode to “game_mode_base_level_arena”.

In “quit_game” function, quitting is implemented.

“NotifyHitBoxBeginCursorOver” function is called if cursor is over a button. In this function, button color is changed to its lighter version and a cursor over sound is spawned.

“NotifyHitBoxEndCursorOver” function is called while ending cursor over event. In this function, light button color is changed to its normal version.

6.9.3. Combat HUD

“hud_combat”, is the HUD of combat. While playing arena level, as a living pawn, this HUD is implemented. On top middle, there is a score text. On the top right corner, there is a stopwatch text. Health indicator is on the bottom left corner. Drawn crosshair is at the middle of the HUD. If health is below and equal to 25, an alert sound is played. In this HUD, 2 music sounds are played in different situations. If player health is higher than 50, a normal combat sound is played and if health is below and equal to 50, heavier version of combat sound, is played.

Played normal combat song is “Usuper Gore” and played heavy combat song is “The Super Gore Nest”. They are level music sounds of Super Gore Nest mission in DOOM Eternal game. They were obtained from SoundCloud [35] [36] by using an online downloader. Low health alert sound is the low health alert sound in DOOM Eternal game. It was obtained from GameBanana [37]. Button sounds were obtained from menu sounds of Half-Life 2 game. Font name is “Press Start 2P” and it was obtained from local computer. There are 2 sizes of this font. They are 20 and 30.

In the constructor of “hud_combat”, audio components are created and sound objects are loaded to this components. There are 3 sound objects and they are normal combat music, heavy combat music and low health alert. Crosshair, health indicator, font and sound objects are loaded. Sounds are imported to audio components.

In “BeginPlay” function, music sound was set to play.

At the beginning of “DrawHUD” function, screen resolution values are obtained for positioning visual HUD elements. Element details which are locations and size, can be changeable according to screen resolution. Font size of score text which is at the top middle of

HUD, is 20 and its color is RGB(255, 199, 0). Font size of stopwatch text which is at the top right corner of HUD, is 20 and its color is RGB(255, 199, 0). Health indicator location is bottom left corner of HUD. Font size of health indicator text, is 30 and its color is RGB(220, 0, 0). Health, score and elapsed time information is obtained from Player State. Elapsed time is converted to time units by “calculate_time” function. There is an audio system in this function. If player health is below and equal to 50, normal combat sound is changed to heavy combat sound. If player health is below and equal to 25, low health alert sound is played.
“calculate_time” function, converts elapsed time to time units as a stopwatch.
In “EndPlay” function, music is stopped.

6.10. Static Mesh

Static meshes are basically 3D objects in Unreal Engine.
“static_mesh” class was created and then, “static_mesh_plane” class whose parent class is “static_mesh”, was created. Two static mesh classes were created because it was thought static meshes may be different than planes meshes but in this project only plane meshes were used. Plane mesh is a default mesh of Unreal Engine. It is just a square plane whose width is “100.0f” and only one face is visible. For avoiding changing game engine contents, plane mesh objet was copied into folders of project.
In source file of “static_mesh_plane”, copied plane mesh is loaded and its shadowing is closed for making a simple game.

6.11. User Defined Structs

Structs are practical for fitting numerous variables into one struct variable.
4 user defined structs were defined in this project. Structs whose names are “plane_struct_field” and “plane_inner_struct_field”, were created for solving a problem which is disallowed multidimensional arrays. A research was done about this problem and according to the results of research [38] [39], multidimensional arrays can be used by defining structs which has arrays. By this method, 3 dimensional arrays was created with header files of “plane_struct_field” and “plane_inner_struct_field”. “plane_struct_field” has a TArray variable and a struct variable whose names “plane_inner_struct_field” and this inner struct has another TArray variable. Structs whose names are “struct_field_material” and “struct_field_material_pack”, were created for collecting material objects into less count of variables. These structs was decreased count of function parameters.

6.12. Player Start

Player Start is a state for spawning player pawn. For not using all default components of Unreal Engine, “player_start” class was created but it does not have any differences from the default Player Start.

6.13. Pawn

Pawn is the main character who is controlled by player.
“paper_player” class was created for implementing a Pawn. “paper” word is come from the default 2D components names of Unreal Engine, which begins with “Paper” word.
In its header file, player pawn properties is set. Player health is set to 100. Its movement speed was not set in header file because it has default movement speed which is 600.
In constructor, its capsule component is set. Capsule component is a collision component which affects physical interaction with other actors. Camera component is added for perspective. Flipbook component which is a 2D animation component, is set. Drawn pixel art weapon objects are loaded. There is only one weapon. Flipbook component is attached to camera component and camera component is attached to capsule component. Created projectile class is loaded. A sound object for firing weapon, is loaded. Shadowing is disabled.
Drawn weapon was inspired from EMG Mark V pistol [40] in DOOM (2016) game [41]. Hand position was inspired from an image in Reddit [42]. Fire sound was obtained from Plasma Defender sounds in a webpage of Fandom [43] about Fallout: New Vegas game [44]. Hand color was inspired from hand color of main character in Viscerafest game.
In “BeginPlay” function, beginning time is saved for setting stopwatch. Two code lines written to ignore inputs for a short time. A problem about improper perspective of player, was noticed and after detecting its reason which is movement of mouse at the beginning of level, ignoring mouse inputs solution was applied.
In “Tick” function of “paper_player”, firstly current time is saved to calculate elapsed time for stopwatch. Two written function whose name are “slide_weapon” and oscillate_walking”, are called.
Animation of weapon is controlled in “Tick” function. If weapon is fired, its flipbook is changed to fire animation. Also there is auto firing feature in this function. If player pressed fire button, “fire” function is called again if fire animation was ended. Health of player, is checked every frame and if it is below 0, pawn is destroyed.

“slide_weapon” function generates weapon location values for implementing weapon sliding feature. The feature slides weapon in the opposite direction of movement. Sliding feature depends world rotation of player pawn but world rotation changes according to perspective of player pawn. To obtain a constant rotation value, vector of world rotation, is rotated according to the perspective. Process of rotation, is implemented by default functions of Unreal Engine but return values of these functions, were not precise. For example, expected result is 0 but functions return “0.000001” number. Because this number is signed instead of a zero number, sliding feature works faulty. A simple tolerance system was added to prevent unexpected consequences.

In “oscillate_walking” function, values of weapon and camera movement, were produced for implementing a walking effect. This feature was inspired by a walking effect from Project Warlock game [45]. Game was examined and it was observed that weapon moves like infinity symbol while walking. After examination, an oscillation algorithm whose name is “oscillate_walking”, was written to make a movement in Y and Z axes for simulating that infinity symbol movement. Also this oscillation was added to camera.

In “SetupPlayerInputComponent” function, inputs are binding to relevant functions whose names are “start_jump”, “stop_jump”, “move_forward”, “move_backward”, “move_right”, ”move_left”, “look_right” and “look_up”.

“start_jump” and “stop_jump” functions implement jumping.

“move_forward”, “move_backward”, ”move_right” and “move_left” functions implement pawn feature of movement. “slide_weapon” and “oscillate_walking” are called in these functions.

In “fire” function, object of “projectile” class is spawned. Movement details of projectile were set in “projectile” class. Projectile moves through the crosshair. Also a fire sound was played after firing.

6.14. Projectile

Projectile is an object which is fired from weapon.

In Unreal Engine, “projectile” class was created for implementations of projectile.

In header file, its velocity is set to “4500.0f” and its life span is set to 4 seconds. Damage amount is set to “25.0f”.

In its constructor function, their collision properties is set to block all object in level. Blocking

is a feature which achieves hitting actions. Its collision component is disabled for navigation system to avoid enemies to escape from projectiles. Gravity of projectile, is set to zero. Drawn projectile object is loaded.
In “Tick” function, there is a simple rotation system to set rotation of projectile for facing to player pawn.
In “FireInDirection” function, projectile is fired according to its velocity.
In “on_hit” function, projectile is destroyed if it hits an object. If hit object is an enemy, projectile deals damage.
In “deal_damage” function, projectile damages enemy. Damage amount is 25. 4 shots is enough to kill enemy because enemy health is 100. After dealing damage, “play_sound_pain” function of enemy, is called for playing a pain sound.

6.15. Enemy

There are enemies in level for player to challenge. But type count of enemies, is 1.
In its header file, enemy properties is set. Enemy health is set to 100. Movement speed of enemy, is set to “640.0f” which is greater than player speed to ensure enemy to catch player.
In constructor function, collision component is set. There two collision components. One of them, capsule component and it blocks every objects. Other collision component is attack component. It is a sphere collision component. If it overlaps player, enemy starts to attack. Sense configuration is set. Enemy only has sight sense. If it sees the player pawn, it starts to chase player. Audio objects are loaded to created audio components and flipbook objects are loaded to created flipbook component.
Drawn enemy artwork was inspired from monster enemies in DOOM Eternal game. Parts of head, body and arm, were inspired from Icon of Sin [46] which is the final level enemy. Hands were inspired from Hell Knight [47] and legs were inspired from Baron of Hell [46]. Their images were obtained from a Funnyjunk post [48]. Idle, growl, chase sounds were obtained from Antilon Guard sounds and attack hit sound was obtained from zombie sounds in Half-Life 2 game. Death and pain sounds were obtained from werewolves sounds in local folders of Sang-Froid - Tales of Werewolves game. Sounds volumes and directions are related location of enemy and player locations.
In “Tick” function, there is a rotation system to set rotation of enemy for facing to player pawn. Flipbook and sounds is controlled in this function. If enemy sees the player, its idle

animation is changed to ready animation and a growl sound is played and if enemy chases the player, its animation changed to chase animation and chase sound is played. Dealing damage is implemented in specific frames of attack animation. If enemy health is below zero, its animation is changed to dead animation and player score is increased by 10 points but it is not destroyed for keeping its corpse. Abilities of dead enemy is restricted.

In “change_flipbook” function has several function to change flipbook.

In “on_sight_sensed” function, implementation of chasing ability is started. After flipbook and sound changings, “chase” function is called.

In “chase” function, enemy is set to chase player according to navigation system. If enemy is close to player and player falls from a high ground, navigation system is not used for falling down with player.

“on_begin_overlap” function is called if attack collision starts to overlap player. Flipbook is changed to attack flipbook and movement speed is decreased in this function. Reason of decreasing speed, is making game easier.

“on_end_overlap” function is called if overlapping of attack collision ends. Flipbook is changed chase flipbook and movement speed is increased to normal in this function.

“play_sound_pain” is callable in “projectile” class. Chase sound is stopped and pain sound is played in this function.

6.16. Navigation System

Nav Mesh Bounds Volume object implements navigation system. Its size must be bigger enough to cover level. In Navigation Mesh section of Project Settings, Runtime Generation option was set to dynamic for generating possible paths dynamically because Nav Mesh Bounds Volume was added into arena level before spawning static objects which are plane constructions. In scope of Nav Mesh Bounds Volume, possible paths are displayed in green color.

In Navigation System section of Project Settings, a supported agent was added. It has properties about possible path settings.

6.17. Levels

Only 2 levels were created in the game project and their names are “level_main_menu” and “level_arena”.

“level_main_menu” is a blank level. “level_arena” has only Nav Mesh Bounds Volume and RecastNavMesh-Default actors. These actors could not be spawned via C++ codes.

6.18. Materials

Materials are assigned to meshes in Unreal Engine. Material properties are set by Blueprint scripts. A method for setting material properties via C++ codes, was research but it could not be found.
A material whose name is “plane_material” was created. Its properties were set via Blueprint. Shading Model of created material, was set to unlit for disabling lighting and shadowing. A texture node was added and its “RBG” output was multiplied by emission strength parameter. Changing this parameter simulates lighting effect but it was not used. Result of multiplying, was assigned to “Emissive Color” property of the material. These settings were made for creating simple visual objects. Material instances were created from “plane_material” and used textures were assigned to them. There are totally 77 material objects.

6.19. Font

There is only one font is used in this project and its name is “Press Start 2P”. It was imported as a Font Face object, from local computer.
To use different font sizes, Font objects were created from Font Face object in Unreal Engine editor.

6.20. Sound Cues and Sound Attenuation

Sound cues are editable audio object in Unreal Engine editor. Their properties are set by Blueprint scripts. A method for setting sound cue properties via C++ codes, was research but it could not be found.
Blueprints of music sound cue objects, have “Looping” nodes which restart sound after ending. Idle and chase sound cues of enemy, have Looping and Doppler nodes. Other enemy sound cues do not have Doppler nodes. Sound Attenuation object which has properties about 3D sound simulating, was added into these sound cues. With Sound Attenuation and Doppler nodes, sounds were simulated as 3D.

6.21. Sprite Sheets and Textures

Sprite sheet is a page which consists of pixel art animation frames. In Unreal Engine editor,

sprite sheets are texture objects and they are extracted as sprites to create flipbooks. Sprite is basically a pixel art image and flipbook is 2D animation in Unreal Engine.
Textures were drawn by using Aseprite program. After importing them into Unreal Engine editor, “Apply Paper2D Texture Settings” option was selected for setting default sprite properties.

6.22. Editor Folders

There are 2 main folders in Unreal Engine editor. Their names are “Content” and “C++ Classes”.
In the game project, “Content” folder has 9 subfolders and their names are “enemies”, “fonts”, “levels”, “materials”, “meshes”, “projectiles”, “sounds”, “textures” and “weapons”. They have objects which are relevant to folder names.
In the game project, “C++ Classes” folder does not have any subfolders. In this folder, created C++ classes are stored.

6.23. Play Mode

Play Mode is a playing mode on Unreal Engine editor. It is useful for testing. In user interface of editor, there is button whose name is “Play” and it actives Play Mode.

6.24. Launch Mode

Launch Mode is a playing mode on Unreal Engine. Its playing method is launching exported game. It is useful for testing. In user interface of editor, there is button whose name is “Launch” and it actives Launch Mode.
Launch Mode is not equal to Play Mode. A game project works perfectly in Play Mode but in Launch Mode, some bugs may occur.

7. GAMEPLAY

Game project is a simple shooter game. After launching game, player clicks start button in main menu to open arena level. In arena level, player starts at a location. Spawned enemies chase and attack to player. Player tries to survive by attacking and running away from enemies. Every kill increases player score. After death of player, high score and current score are displayed.

8. CONCLUSION

Although Unreal Engine is a popular game engine, official C++ documentation and unofficial tutorials about Unreal Engine, are not enough. Unexpected bugs occurs so many times, while developing. Investigations of solutions and drawing artworks, take too much time. However, developing video games with Unreal Engine, improves coding skills. At the end of this project, the developer gained the ability to solve problems with his own solutions instead of researching them on internet.

REFERENCES

[1] bugra-sarikaya (June 10, 2022), game_project: https://github.com/bugra-sarikaya/game_project
[2] Steam (June 10, 2022) Viscerafest: https://store.steampowered.com/app/1406780/Viscerafest/
[3] Unreal Engine (June 10, 2022): https://www.unrealengine.com/
[4] Visual Studio (June 10, 2022): https://visualstudio.microsoft.com/
[5] Steam (June 10, 2022) Aseprite: https://store.steampowered.com/app/431730/Aseprite
[6] Audacity (June 10, 2022): https://www.audacityteam.org/
[7] GitHub (June 10, 2022): https://github.com/
[8] GitHub Desktop (June 10, 2022): https://desktop.github.com/
[9] Unreal Engine (June 10, 2022), First Person Shooter Tutorial: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/CPPTutorials/FirstPersonShooter/
[10] YouTube (June 10, 2022), reubs: https://www.youtube.com/c/ReubenWardTutorials
[11] YouTube (June 10, 2022), Hitbox: https://www.youtube.com/channel/UCh6TE-rm-8wJlwhInNzc39A
[12] YouTube (June 10, 2022), Tolga Uzun: https://www.youtube.com/c/BlenderE%C4%9Fitim
[13] YouTube (June 10, 2022), Unreal Kafası: https://www.youtube.com/channel/UC9vq1VYvvwRrrJgFi4VdqZg
[14] YouTube (June 10, 2022), freeCodeCamp.org: https://www.youtube.com/c/Freecodecamp
[15] Unreal Engine (June 10, 2022), Unreal Engine 4 Documentation: https://docs.unrealengine.com/4.27/en-US/
[16] Stack Overflow (June 10, 2022): https://stackoverflow.com/
[17] Reddit (June 10, 2022): https://www.reddit.com/
[18] DeviantArt (June 10, 2022): https://www.deviantart.com/
[19] ArtStation (June 10, 2022): https://www.artstation.com/

[20] Funnyjunk (June 10, 2022): https://funnyjunk.com/

[21] GameBanana (June 10, 2022): https://gamebanana.com/

[22] SoundCloud (June 10, 2022): https://soundcloud.com/

[23] Bandcamp (June 10, 2022): https://bandcamp.com/

[24] Steam (June 10, 2022), Sang-Froid - Tales of Werewolves: https://store.steampowered.com/app/227220/SangFroid__Tales_of_Werewolves/

[25] Unreal Engine (June 10, 2022), Frequently Asked Questions (FAQs): https://www.unrealengine.com/en-US/faq

[26] Epic Game (June 10, 2022): https://www.epicgames.com/

[27] Adobe (June 10, 2022) ,Compare Plans: https://www.adobe.com/products/photoshop/compare-plans.html

[28] Nitrogen (June 10, 2022), "Blueprints vs C++ - Which One Should You Learn in 2021?": https://youtu.be/ueOqcwlFfMc

[29] Unreal Engine (June 10, 2022), Unreal® Engine End User License Agreement: https://www.unrealengine.com/en-US/eula/unreal

[30] Karl Casey (June 10, 2022), Broken: https://karlcasey.bandcamp.com/track/broken

[31] Steam, (June 10, 2022), Half-Life 2: https://store.steampowered.com/app/220/HalfLife_2/

[32] Reddit (June 10, 2022), "Need all Half-Life 2 sounds!": https://www.reddit.com/r/HalfLife/comments/e15qj4/comment/f8mse54/?utm_source=share&utm_medium=web2x&context=3

[33] Steam (June 10, 2022), DOOM Eternal: https://store.steampowered.com/app/782330/DOOM_Eternal/

[34] Doom Slayer (June 10, 2022), End of Level: https://soundcloud.com/doom-slayer-868995838/end-of-level?in=doom-slayer-1/sets/doom-eternal-ost-2020

[35] Doom Slayer (June 10, 2022), Usuper Gore: https://soundcloud.com/doom-slayer-1/usuper-gore?in=doom-slayer-1/sets/doom-eternal-ost-2020

[36] Doom Slayer (June 10, 2022), The Super Gore Nest: https://soundcloud.com/doom-slayer-1/the-super-gore-nest?in=doom-slayer-1/sets/doom-eternal-ost-2020

[37] GameBanana (June 10, 2022), "Doom(2016) low on health Killsound": https://gamebanana.com/sounds/36226

[38] GameDev.tv (June 10, 2022), "Nested Array crashing":

https://community.gamedev.tv/t/nested-array-crashing/142553
[39] Unreal Engine Forums (June 10, 2022), “problem with a struct”: https://forums.unrealengine.com/t/problem-with-a-struct/157923
[40] Bryan Flynn (June 10, 2022), Doom Mark V Pistol: https://www.artstation.com/artwork/zo50Z
[41] Steam (June 10, 2022), DOOM: https://store.steampowered.com/app/379720/DOOM/
[42] Reddit (June 10, 2022), “Here it is! The fully upgraded Bioshock Pistol from the first game!!”: https://www.reddit.com/r/Bioshock/comments/m8kqsz/here_it_is_the_fully_upgraded_bioshock_pistol/
[43] Fandom (June 10, 2022), Plasma Defender (GRA): https://fallout.fandom.com/wiki/Plasma_Defender_(GRA)
[44] Steam (June 10, 2022), Fallout: New Vegas: https://store.steampowered.com/app/22380/Fallout_New_Vegas/
[45] Steam (June 10, 2022), Project Warlock: https://store.steampowered.com/app/893680/Project_Warlock/
[46] Funnyjunk (June 10, 2022): https://loginportal.funnyjunk.com/large/pictures/92/1f/921ff4_7543729.jpg
[47] Funnyjunk (June 10, 2022): https://loginportal.funnyjunk.com/large/pictures/74/e9/74e95d_7543729.jpg
[48] Funnyjunk (June 10, 2022), “DOOM ETERNAL Art book leaked”: https://funnyjunk.com/Doom+eternal+art+book+leaked/cMphMpZ/

RESUME

Buğra Sarıkaya was born on November 16, 1995, in Mersin, Turkey. He is currently studying in Çukurova University Computer Engineering department in fourth grade.

ACKNOWLEDGEMENTS

I am much obliged to developers who shares their knowledge on internet. Special thanks to who wrote the documentation of Unreal Engine. Their one sentence explanations, were very helpful.