TRIBHUWAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

OOP Final Project Report

SpaceRider

Submitted By:

SWASTAB ACHARYA (076BCT092)

SUBAS ADHIKARI (076BCT083)

SAILESH SHIWAKOTI (076BCT063)

Submitted To:

Department of Computer Engineering

Submission Date: 8/24/2021

Acknowledgment

This final project on C++ is possible mainly because of the task we were assign to by the Department of Electronics and Computer Engineering. We are extremely thankful to for giving us the platform to design something which is related to computer programming, team work, and utilization of one's knowledge developed so far.

We are thankful for our respected sir, Er Daya Sagar Baral for his lectures, labs and his book which helped us to understand the concept of C++ and also helped to solve the problems faced during the process of designing and implementing.

We are thankful for our friends, family and to name a few everyone for assisting their supports, giving their invaluable time and helping us in many ways which can't be simply expressed by writing.

We are also thankful to all the creators of necessary contents on the internet which almost mostly was necessary for main task we built.

Finally, we are very excited to share our happiness and experience we have been through so far to everyone, and we want to say this has been possible mainly because of them. That's all.

Abstract

This project work was given to us as a minor project for our academic session B.E. (Computer) Second Year First Part as prescribed in the syllabus designed by IOE, TU. We developed a 3D game, named SpaceRider, from scratch using C++ and DirectX APIs where the player has to direct a spaceship to dodge the asteroids, planets and all other objects of space and to safely arrived to Earth from any point like Mars or Jupiter. The scoring is based on player's speed and time it takes to reach the earth.

Table of Contents

I. Objectives

The objectives of our project are as follows:

- To create a project on Object Oriented Programming (OOP).
- To explore the features of C++ language.
- To use DirectX APIs to create a graphics engine from a scratch and make any game of our choice.
- To experience the amount of help, supports and enjoyment it has while working in a team, with friends.

II. Introduction

This game, SpaceRider is simply a use of matrices, vectors and other mathematical concepts to render an OBJ file created using applications like Blender by using the graphics pipeline created by DirectX and manipulating such objects like spaceship, asteroids, planets and controlling the camera and spaceship moments in an interactive way.

III. Application

By giving a little time and effort we can uplift this game by integrating audio engine and physics engine which is more like a commercial game and implementation to this level is easier as coding is done from scratch and adding other features can be as well done from scratch rather than using other APIs as DirectX is fully capable of handling any multimedia and C++ on the other hand is a powerful programming language.

IV. Literature survey

The book named "The secrets of object-oriented programming" was used as reference, YouTube tutorials, DirectX11 documentation were used for designing the engine and game. For text-rendering we used DirectX Tool Kit's helper class "SpriteBatch.h".

V. Existing System

This game features are not totally new than other games related to space shooter games and asteroids collisions but the APIs we used and framework we have created is totally new able to program any game.

VI. Methodology

This project is based on C++ programming language utilizing DirectX API and "Object Oriented Programming" concept. Different classes were created with required number of private and public member data and member functions for smooth running of the program preserving the concept of data hiding. The concept of code reusability, data abstraction were implemented in the project.

We used VisualStudio2019 community as C++ IDE , Windows 10 as target platform and DirectX 11 SDK(already installed) for 3D game engine.

In the program, these classes are used, serving their own purposes:

- **SwapChain**

This class handles swapchain which is collection of frame buffers that is used to show the render frame on the screen and also used for frame stabilization uses double buffering technique.

- **DeviceContext**

This class handles device context which is an extension of Directx device that allows us to generate rendering commands to send to the video driver for execution. The driver then redirects the command to the GPU/CPU for final elaboration or final rendering on the screen and also allows us set pipeline states

- **VertexShader**

This handles the program that executes on the GPU that allows us to transform only the vertices.

- **PixelShader**

This handles the pixel rendering functions on the screen.

- **ConstantBuffer**

This is like a vertex buffer but any data can be sent to either pixel shader, or vertex shader or hull shader.

- **Math**

This class handled transformation matrices viz. translation matrix, rotation matrix, and scale matrix which manipulates the coordinates of coordinate systems we used namely model coordinate space which is the system in which all the original position of 3D model are placed, world coordinate space which is the system in which we place all our 3D objects, view coordinate space which is system as seen by the camera and then finally screen or projection space in which coordinates as seen by the camera are projected either orthographically or by perspective method.

- **IndexBuffer**

Like a vertex buffer, it is a memory buffer where a series of indices are inserted and can be used to send to device context for drawing a quadrilateral or any polygon.

- **InputSystem**

This class is, like a graphics engine, a sub system of a game engine and it handles input devices like keyboard and mouse using OS APIs having functions like `::GetCursorPos()` and `::GetKeyboardState()` etc and input listener are derived to override pure virtual functions.

- **FirstPersonCamera**

Given the rotation and translation of a matrices of camera, the position of camera in world space is obtained by multiplying it with world space and the position of an object from a world matrix to camera matrix is obtained by multiplying it with inverse of camera position which can be projected into far-plane of frustum, also by getting Z-direction of it we can build third-person camera.

- **Lightening**

For this we used Phong reflection model which describes the way a surface reflects light as a combination of the ambient reflection, diffuse reflection of rough surfaces, and specular reflection of shiny surfaces, mainly coded in "pixelshader.hlsl" files to obtain directional light and point light. The reflections are obtained by dot product between light direction and camera direction with respect to point in space obtained by normalize function and reflected vector can be obtained by intrinsic function, `reflect()` of DirectX.

- **Material**

This handles the material which is, in computer graphics, a substance or mixture of substances that constitute an object mainly defined by vertex and pixel shader.

The programming methods we've used can as summarized below:

- Analyzing the concept that can be used to develop proper program.
- Discussion on the topic that might be faced onwards.
- Making the project schedule.
- Initial coding for creating logic.
- Coding the program.
- Execution and testing the program.
- Debugging.
- Program Documentation

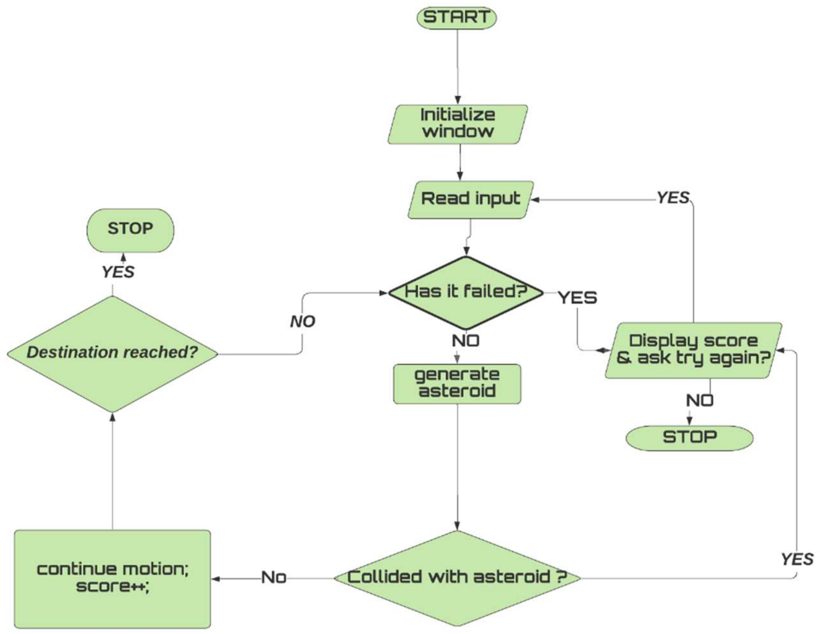
VII. Implementation

The game starts with the spaceship in an unknown position in space and player has to pilot until it arrives destination i.e., Earth.

In overall, the following methods were implemented for the completion of the project, which is regarded as the basic steps for developing the software and listed as follows:

- **Study, research and analysis:** Gathering information about the topic of the project.
- **Proposal submission:** Documenting the information about the project was carried out before starting the coding.
- **Designing of the layout of the project:** Proper positioning of the components required for the project was designed properly.
- **Coding:** Coding of the project was done as mentioned in the proposal.
- **Testing, modification and Documentation:** Project is tested and modified continuously as per requirement to make it more attractive and error free.

Block diagram



VIII. Results

Finally, we completed the project within the deadline and we successfully completed it by lot of works collaborating with each other. Although we could not make the game having all the features a game usually have, but we are satisfied to our work as it was best we could do in given time and also, we learnt and experienced about basics of building a project.

IX. Problems Faced and Solutions

We, during the development of this project, faced a lot of errors and face solved them by discussing with team mates and doing bit more research each time we encountered a problem. The main problem we faced was, we were unable to link the libraries required for the project like "d3dcompiler.lib" and other errors related to API were solved by updating the codes of tutorial having functions of deprecated version with the latest version of them.

X. Limitations and Future Enhancements

We can add the audio and physics engine to our game to make it more realistic. Moreover, we can add other features like spaceships racing and shooting asteroids as well as other spaceship for more entertainment.

XI. Conclusion and Recommendations

Through the journey of this project, it has taught that development of a project by first making better framework helps to better organize code, implementing the design by having the requirements fulfilled for achieving any feature needed.

It is recommended to involve in this kind of project which is helpful for tasks related to computer graphics and 3D designs.

XII. References

- Daya Sagar Baral and Diwakar Baral, "The Secrets of Object-oriented Programming", Bhundipuran Prakasan
- https://docs.microsoft.com/en-us/windows/win32/direct3d11/dx-graphics-overviews
- https://github.com/microsoft/DirectXTK/wiki
- Frank D. Luna "Introduction to 3D Game Programming with DirectX 11"