CSP PROJECT 2023

ZORO

MAY 25

Submitted to : Dr Shiraz khurana
Submitted by : Manas Thakur
Risabh Balodhi
Taniya Choudhary

INTRODUCTION

The purpose of this project is to develop a mobile and windows gaming application called “Zoro” that offers users a complete new experience about car racing game .

Preference For App

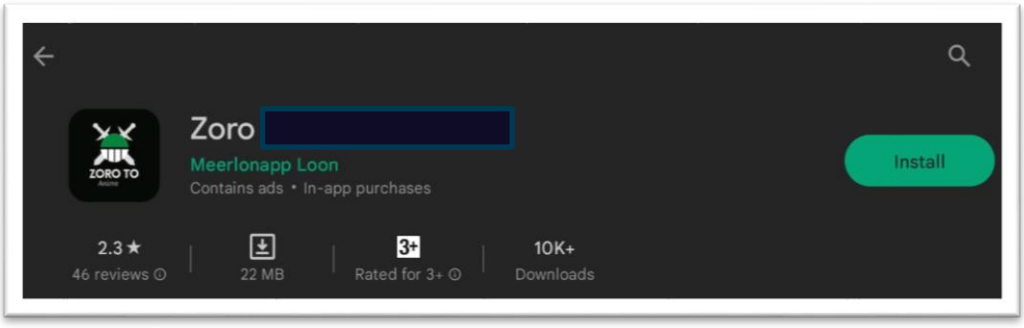
This application will be helpful for amateurs , which are trying to experience racing games on another level and want to develop passion and love for racing games. All age groups can play this game to escape from reality and stress for some time. They can play Zoro with their friends too.

Features Of Zoro

• **Simple to play**. The user interface and the rules of playing Zoro must be intuitive. A player should learn how to play while playing the game.
• **Multiplayer**. Zoro game enables multiplayer system . You can play 1 vs 1 with your friends.
• **Short response time.** The startup time of the game must be low. The response time to a mouse click must allow real time playing.

• **Interactive.** Zoro should support smooth car movements.
• **User interface**. To support the player as much as possible during the Zoro game, all relevant information of the driven car must be visible to the player. This includes the status if a car is crunched or not and all properties needed for driving the car. This includes at least the speed and the position of the car.
• **Different car types**. The Game must support two different kinds of controlled cars. One kind must be a fast car that does not change its direction until it reaches the boundary of the game board. The other is a slow car, which may change its direction any time. Both kinds of cars should be able to change their speed. They have a different minimum and maximum speed.
• **Customizability**. Zoro must be adaptable for the differences of two markets in North America and Europe, that is, it must support the metric and imperial measurement systems.
• **Multiple platforms**. The game must run on different platforms.

How to download



- Zoro can be easily downloaded from Google play store or from Tap Tap.
- Apk is also provided on official website.

PROBLEM DEFINATION

1. Problem

The computer market for car games is moving towards complex, photo-realistic 3D car simulation games. The problems are long startup times and high learning curves for the player. Furthermore these games require a large amount of CPU time, which forces the players to upgrade their computer system to the hardware needs for these games.

Solitaire, Minesweeper or Moorhuhn are examples for the success of simple and easy to use games. They have a high fun factor, no learning curve and short startup times. As the duration of a game is short, they can be played in nearly every situation.

The goal of Zoro is to provide a simple, fast and cheap car game, which has a high fun factor by crashing cars together. It should not need special hardware requirements.

2. Objectives

The objectives of the Zoro game are to:

- provide a simple, fast and interactive car game with a high fun factor.
- to be platform independent.
- require no extra administrational knowledge. The player should be able to do all administrative work (e.g. installing).

3. Example scenarios

In this section, we present typical scenarios of system usage for illustration.

3.1. Starting Zoro

The Zoro user interface displays a game board, which consists of an arbitrary number of cars, a tool bar with game control buttons and an instrument panel. One car, which is placed in the upper left corner of the board, has a different chassis than all other cars. The same chassis is visible in the instrument panel with the label “your car”. This is Risabh’s playing car. All other cars are distributed over the board. Two types of cars with different chassis are visible. These are slow and fast computer-controlled cars. The game is ready to play.

3.2. Playing a Zoro game

Risabh presses the “start” button on the tool bar. All cars start to drive in different directions with different speeds. The car, driven by Risabh, is located at the upper left corner and waits for instructions from Risabh. An instrument panel displays the current speed and position of the driven car. Risabh can control the car speed, by clicking on the speed control button. He can change the driving direction by clicking with the mouse at a specific position on the board. His driven car changes its direction to the clicked position. A car changes its direction also, if it reaches any boundary of the game board. The angle of incidence is the angle of reflection.

3.3. Winning a Zoro game

If Risabh drives his car into a user or computer-controlled car from the higher game board position, the user or computer-controlled car gets crunched. It does not drive for 3 secs. If He can repeat this with all of this user or computer-controlled cars, He wins and a dialog window with the text “Congratulation, you win!” appears. The window can be closed by pressing the “OK” button.

3.4. Losing a Zoro game

If any computer-controlled or user controlled car drives into Risabh’s car from a higher game board position or Risabh cars hit 10 cones, Risabh loses and a “You lost!” dialog window appears. The window can be closed by pressing the “OK” button.

3.5. Configuring Zoro

Risabh presses the menu item “Preferences” from the Zoro menu. A window with the Zoro settings appears. It contains a number of fast and slow cars, a number 'frame per second' and three images with the labels “slow car”, “fast car” and “driven car”. A “select” button is located next to each image. Risabh can change the number of slow and fast cars and the number of the frames per second. By pressing a “select” buttons, Risabh can select images from a library that are used as car chassis. After changing the settings, He presses the “OK” button and the preferences window disappears. The changes are reflected immediately on the game board.

4. Requirements

The Zoro player should be able to:

- download and install the Zoro game
- start the Zoro application
- stop the Zoro application
- start the game

- stop the game
- change the direction of the driven car
- change the speed of the driven car
- collide the user driven car with a computer-controlled car
- configure the cars.

5. Target environment

- The Zoro application must be created and designed in Unity Hub.
- The Zoro application must be built In C or Java or Python languages.
- The game must run at least on Windows and Android.

6. Deliverables

- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Software Project Management Plan (SPMP)
- Object Design Document (ODD)
- Software Project Management Plan (SPMP)
- User Manual
- Zoro executables: application class files
- Zoro HTML download page

7. Acceptance Criteria

The delivered system is accepted if:

- The Zoro executable runs on Windows and Mac OS X.
- The startup time of Zoro is less than 10 seconds.
- The car movement is done in more than 5 frames per second.
- The user-driven car changes its direction 0.5 seconds after the user clicked the mouse.

PROJECT-DESIGN DOCUMENT

Project Concept

1 Player Control	You control a	*Car*	in this	*Top Down* game
	where	*Arrow keys*	makes the player	*Moves vertically and Horizontally*
2 Basic Gameplay	During the game,	*Cones and boosters* appear	On	*Tracks*
	and the goal of the game is to	*Collect boosters , avoid cones and finish the race.*		

3

Sound & Effects

There will be sound effects

whenever player moves ,collects booster and hits cones or Is destroyed.

and particle effects

whenever car drifts and hit by another car.

[*optional*] There will also be

description of any other expected special effects or animation in the project.

4

Gameplay Mechanics

As the game progresses,

New landscapes and different weather conditions will occur

making it

Difficult to control the car

There will also be

Day and Night mode

5

User Interface

The

score

will

Decrease

whenever

player hits the cones with car.

At the start of the game, the title

Cars

will appear

and the game will end when

If player car hits 10 cones or lost to a different car.

6

Other Features

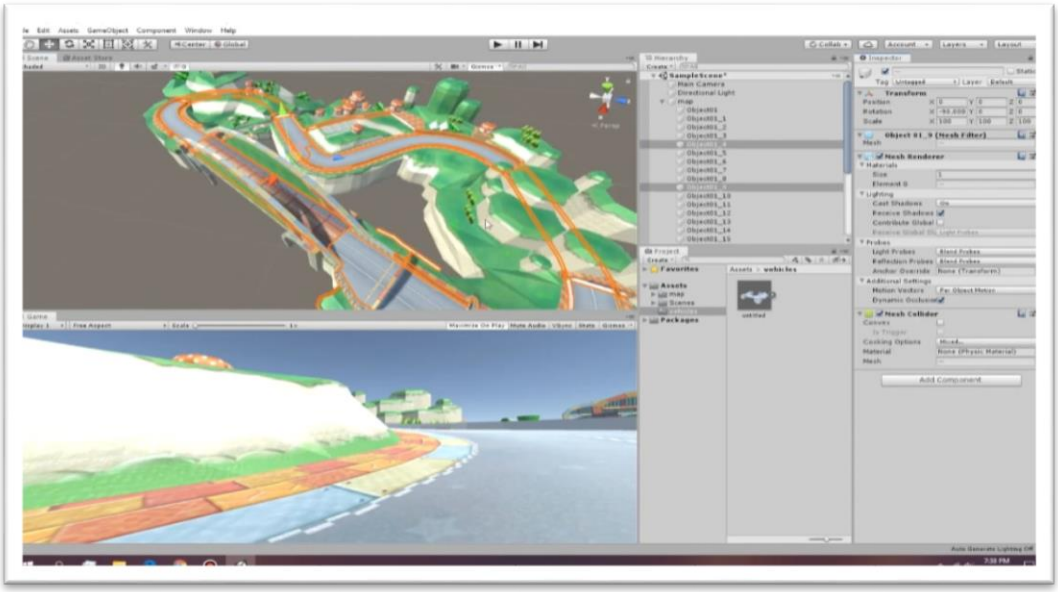
The game will be an offline multiplayer game , two players can play against each other . one player can play with a,w,s,d keys and other with arrow keys. If one player collides Its car with other car , the car will be destroyed and will resets In 3 secs.

BACKGROUND STUDY

In a making of a game everyone should research everything about the type of game they are making and collect all type of information about it and a mentor to guide them.

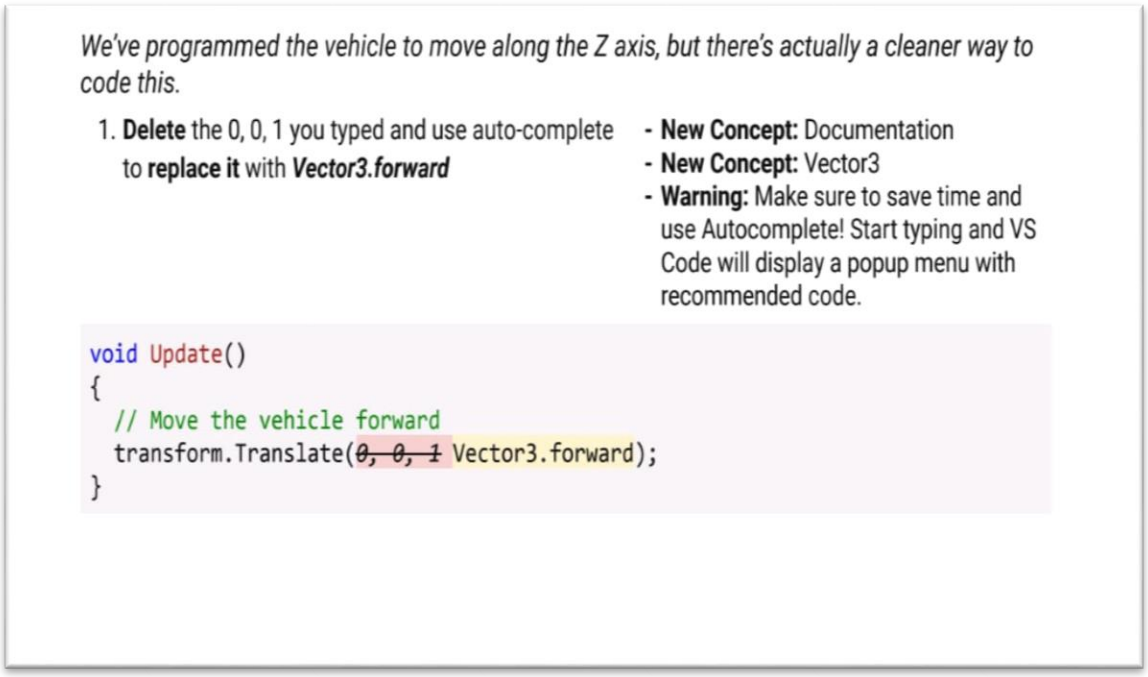
1. MENTOR

- Our mentor (Dr Shiraz khurana) was the first one to show us the way to develop a game . He helped us to increase our creativity to solve problems and gave us inspiration to design a solution to every problem we see. He helped us not only in development of the racing



game but in real life too.

- Our mentor guided us throughout the game development by providing us free applications like Unity hub to create our game of choice and let us show our creativity.
- He thoroughly guided us how to use Unity hub and different functions of applications as well .
- He also provided us notes , articles and pdf about game development which included everything like assets or prototype which includes built in materials such as landscapes , obstacles , vehicle skins etc.



We've programmed the vehicle to move along the Z axis, but there's actually a cleaner way to code this.

1. **Delete** the 0, 0, 1 you typed and use auto-complete to **replace it** with ***Vector3.forward***

- **New Concept:** Documentation
- **New Concept:** Vector3
- **Warning:** Make sure to save time and use Autocomplete! Start typing and VS Code will display a popup menu with recommended code.

```
void Update()
{
  // Move the vehicle forward
  transform.Translate(0, 0, 1 Vector3.forward);
}
```

- He also provided us different codes for the functioning and the movement of the game.

  For an example : above code is used for moving vehicle forward.

2. APPLICATION USED (UNITY HUB)

- Unity hub is the application we used to develop our racing game
- Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at apple

Worldwide Developers Conference as a Mac OS X game engine. The engine has since been gradually extended to support a variety of desktop, mobile, console and virtual reality platforms. It is particularly popular for iOS and Android mobile game development, is considered easy to use for beginner developers, and is popular for indie game development.

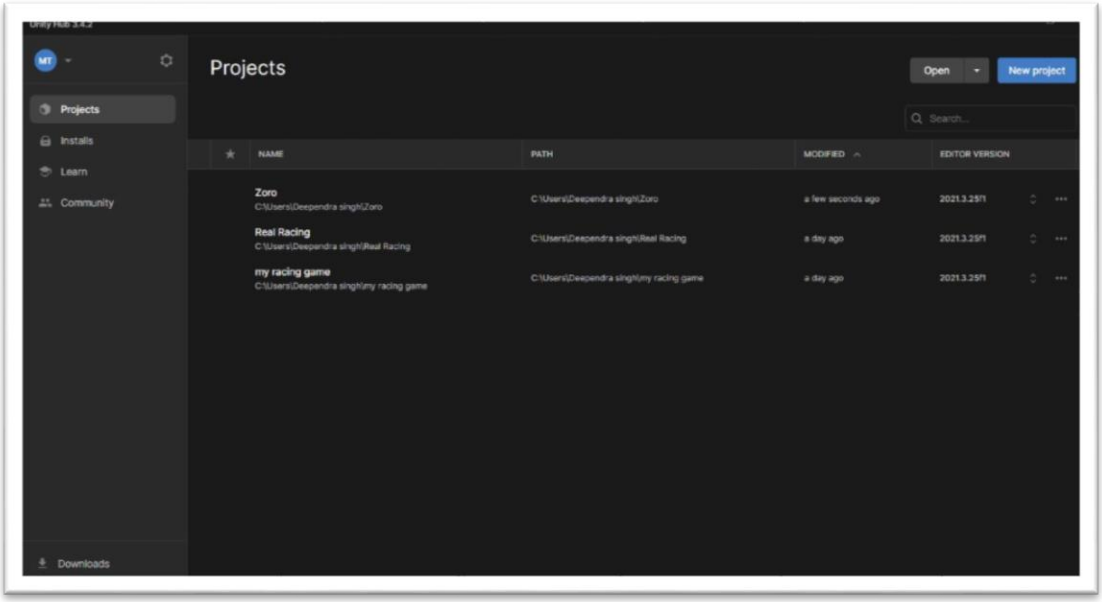
- The engine can be used to create three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations and other

experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering, constructions.

2.1 SUPPORTED PLATFORMS

- Unity is a cross-platform engine. The Unity editor is supported on Windows, macOS, and the Linux platform, while the engine itself currently supports building games for more than 19 different platforms, including mobile, desktop, consoles, and virtual reality. Unity 2020 LTS officially supports the following platforms

- Mobile platforms iOS, Android (Android TV), tvOS;
- Desktop platforms Windows(Universal Windows Platform), Mac, Linux



- Web platform WebGL;
- Console platforms PlayStation (PS4,PS5), Xbox (Xbox One, Xbox Series X/S, Nintendo Switch, Stadia;

- Virtual/Extended reality platforms Oculus, PlayStation VR, Google's ARCore, Apple's ARKit, Windows Mixed Reality(HoloLens), Magic Leap, and via Unity XR SDK Steam VR, Google Cardboard. and the United States Armed Forces.

3. LANGUAGE USED AND APPLICATION USED FOR PROGRAMS

3.1 LANGUAGE

- Language we used for writing codes and programs for game development is C.
- C is an imperative procedural language, supporting structured programming, lexical variable scope and recursion, with a static type system.

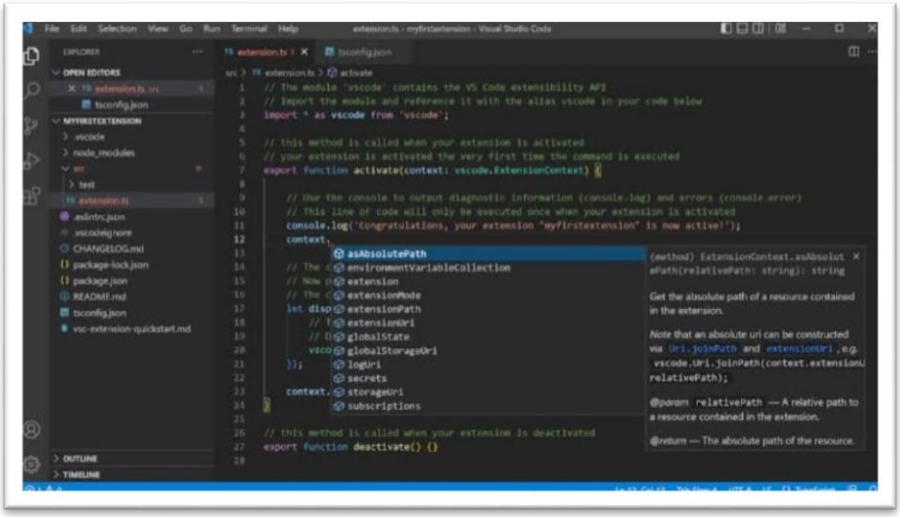
- It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions,

all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming.

- A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

3.2 APPLICATION

- Application we used for game development is Visual Studio Code.
- A Visual Studio Code is a source-code editor that can be used with a



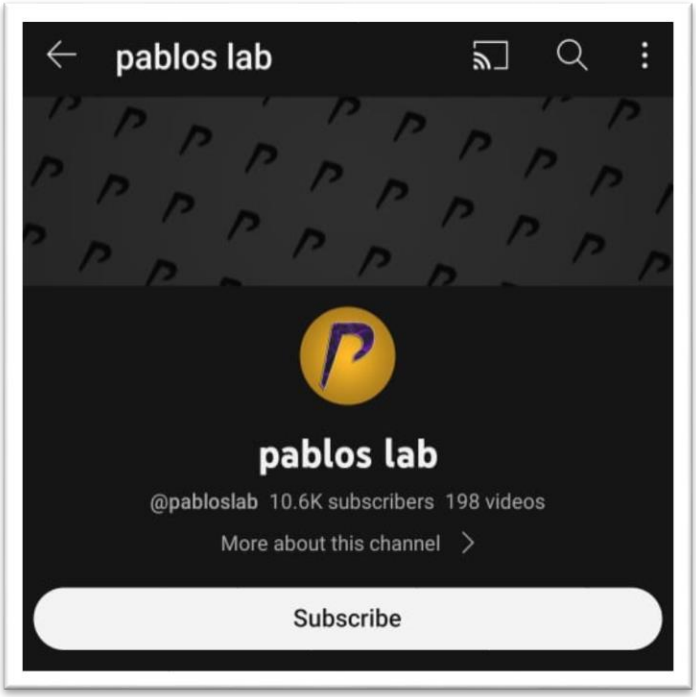
variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

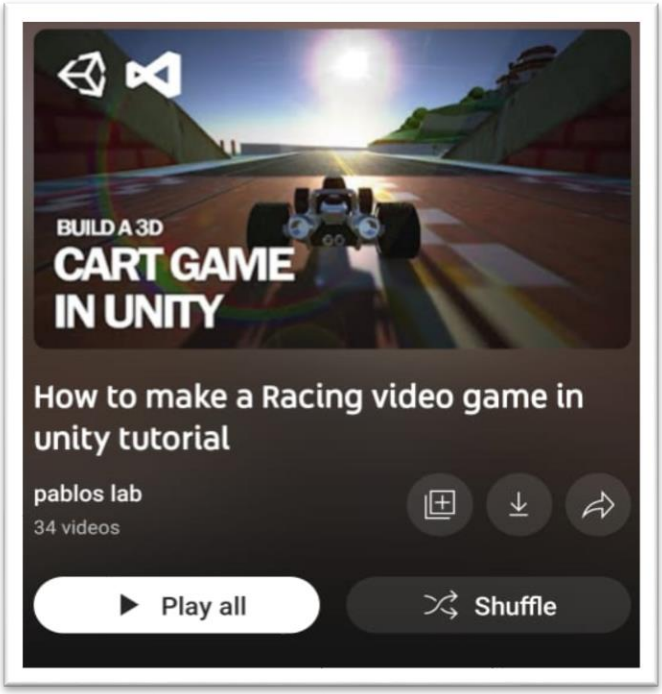
- Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax

highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Market place.

4. YOUTUBE

- For more specific and deep way of development of the game , we have taken help from YouTube.
- Pablo lab is a youtuber who helps with making and creating different type of games on Unity app.
- Details of videos were very good , everything was clear and understandable.





- About 70 percent of work of our game development is done because of Pablo labs .
- Link to Pablo labs YouTube channel : https://youtu.be/vO_18bhd2nA