**A project report on**

**IMPLEMENTATION OF PATH FINDING ARTIFICIAL INTELLIGENCE USING  
A\* ALGORITHM FOR COMPUTER GAMES**

**Submitted in partial fulfilment of the requirements for the**

**degree of**

**BACHELOR OF COMPUTER APPLICATION WORK CARRIED OUT AT**

**BSOFT Technologies**

**FOR THE ACADEMIC YEAR 2016-2017**



Submitted by

RAMJITH C S (WM14BCAR22)

**Internal GuideExternal Guide**

**Mr.Sibin Sebastianmr.Rejoy VM**

**(Asst.professor BCA dept.) (Software Developer ,Bsoft**

**WMO IMAM GAZZALI ARTS AND SCIENCE COLLEGE**

**KOOLIVAYAL**

**Department of computer science**

**kannur university**

**WMO IMAM GAZZALI ARTS AND SCIENCE COLLEGE**

**KOOLIVAYAL**



**certificate**

Certified that this report titled “**IMPLEMENTATION OF PATH FINDING ARTIFICIALINTELLIGENCEUSINGA\* ALGORITHM FOR COMPUTER GAMES**”is a bonafide record of the project work done by Sri/ Kum **Ramjith C S**under our supervision and guidance, towards partial fulfillment of the requirement for award of the Degree of Bachelor of Computer Application (BCA) of the Kannur University

**Faculty in-charge Head of the Department.**

**Examiners**

1.

2.

**Name of Candidate :RAMJITH C S**

**Registration No:WM14BCAR22**

**Examination Center: W.M.O Imam Gazzali Arts and Science College**

**Date of Exam**:

**DECLARATION**

We hereby declare that this project work entitled “**IMPLEMENATATION OF PATH FINDING ARTIFICIAL INTELLIGENCE USING A\*STAR ALGORITHM FOR COMPUTER GAMES”**has been completed and implemented in partial fulfilment of the requirement for the award of Bachelor of Computer Application WMO IGASC Koolivayal. This information submitted herein true and original to the best of our knowledge.

Place :sulthanbathery Yours Truly,

Date:28/04/2017 Bsoft Technologies

**ACKNOWLEDGEMENT**

We are indebted to our beloved principal **Mrs.Bollamma** and management of WMO IGASC Koolivayal for their valuable support and motivation. We take this opportunity to express our sincere gratitude to **Mr.Jafer p** HOD Dept. Of Computer Application and our internal guide, for his valuable guidance and support .We are also thankful to lecturers of Computer Science for their valuable guidance and support.

We thank all teaching and non – teaching staffs of our college to their support and help.

Last but not least We thank our parents and friends who helped in completion of our project.

**ORGANIZATIONAL OVERVIEW**

Geometry Graphics Factory is a farm working on Computer Graphics and Visualization under BSoft Technologies. Firm deals with applications of computer graphics in domains such as Game Industry, Web Applications, Parallel Programming. Mobile applications etc.

**1 Technology**

Geometry Graphics Factory works on programming languages such as C sharp, C++ for gaming purpose. Firm also deals with Game engines like unity 3D three.js etc. In web domain it works

Withphp,JavaScript etc. Firm also has experience in writing programs in Game Engines and also writing codes for making Game Engines. We also work in android Technologies

**2 Software Projects**

Firm usually works on graphics software projects for different platforms using game engines. It also deals with applications of graphics visualization in Web domain. We work on Android platform

especially on back end services

**3 Training**

Geometry Graphics Factory offers training in project oriented works in Game, Mobile and Web domains. In either case Geometry Graphics Factory takes the projects which are suitable for application of computer graphics

**INDEX**

|  |  |  |
| --- | --- | --- |
| S/N | CONTENTS | PAGE NO |
| 1 | Introduction   * 1. Abstract   2. Hardware configuration   3. Software configuration |  |
| 2 | System analysis  2.1 Description of present the system  2.2 Limitations of present system  2.3 The proposed system -its advantages and Features  2.4 Requirement specification |  |
| 3 | Design  3.1input design  3.2output design  3.3overall description  3.4 Description of database  3.5 list of fields or attributes  3.6 Context diagram of the proposed system with addition level of expansion  3.7THREE-TIER ARCHITECTURE  3.8 menu tree |  |
| 4 | 4.1Introduction  4.2Types of Testing  4.2.1.Black Box Testing  4.2.2.White Box Testing  4.3levels of testing  4.3.1.Unit Testing  4.3.2. Integration Testing  4.3.2. Integration Testing  4.3.4.Validation Testing  4.4validation, test cases and test requirements |  |
| 5 | Implementation and maintenance  5.1 Implementation procedure |  |
| 6 | Conclusion  6.1 scope for feature enhancement |  |
| 7 | Reference |  |
| 8 | Appendix  8.1.program – file table that shows the files/tables used by each program  8.2 screen layouts for each data entry screen |  |

**1.INTRODUCTON**

It is a racing game. We have to implement Wheel colliders for forward movement and shock absorber action .Player will move one car. The enemy car will be moved by AI sensor algorithms and the path is decided by A\* algorithms before the race. Scene will be generated randomly, Just look at the tiles in A\* and we make tiles on road. We divide the tiles on road as one with object and other with no object. We use the A\* to figure out collection of tiles to reach path. We make waypoint from the point in the order of their path. Then we race the path using AI.

The game is a contemporary video game, and its objective is to offer the player a challenging and enjoyable experience in a car race against a game-controlled car. Although the player’s goal within the game is to win the race against the game-controlled car, the AI techniques adopted in the game are primarily designed to give the player an enjoyable time racing his or her car. In other words, the objective of winning by either side is not given the highest priority.

The project is more of a geometrical problem with physics than about game playing mode as in commercial Game. We concentrate on programming part of it than artistic designing part of the Game, and player controls the movement of the vehicle In this project, we have to first code the movement and AI for the player vehicle. We also need to make the interactive responsive section such that vehicle move as per the input given by the player. The player has to avoid obstacles as per user input while others cars in the scene which runs on AI alone, has to do it alone. Once Player finished the finishing point, score will be shown in descending order with rank of other players

**1.1 ABSTRACT**

This project presents a car racing simulator game, in which the human player races a car against one game-controlled car in a three-dimensional environment. The objective of the game is not to defeat the human player, but to provide the player with a challenging and enjoyable experience. To ensure that this objective can be accomplished, the game incorporates artificial intelligence (AI) techniques, which enable the carto be controlled in a manner that mimics natural driving. The project presents the use of AI techniques in contemporary video games, and discusses the AI techniques that were implemented in the development of this game. A comparison of the AI techniques implemented in the Unity platform with traditional AI search techniques is also included. The enemy car will be moved by AI sensor algorithms and the path is decide by A\* algorithm before the race.

**1.2 Hardware configuration**

Processor: We use Intel i7,i5 processor. We can use other processors too but it may slow down the graphics

GPU: Graphics card with DX9 (shader model 2.0) capabilities. Anything made since 2004 should work.

Mouse: 3 button

Keyboard: General keyboard

**1.3Software Requirements**

Operating System:  Windows 7 SP1+, 8, 10; Mac OS X 10.8+.

Front End: Unity GUI

Back End : Unity

**1.3.1Application Software**

**1.3.1.1Unity**

Unity Technologies is revolutionizing the game industry with Unity, the breakthrough development platform for creating games and interactive 3D and 2D experiences like training simulations and medical and architectural visualizations, across mobile, desktop, web, console and other platforms.

Unity was created with the vision to democratize game development and level the playing field for developers across the globe. Through industry-disrupting pricing and business plans, incredible ease of porting to multiple platforms, a deep yet approachable project editor and our innovative game promotion, user retention and analytics services Unity makes making successful games far easier than ever before.

We make codes, testing environments…etc in Unity 3D

**1.3.1.2Front End Design**

Front end in Gaming software is quite well defined. We simulate visuals and it is our project. So we have scene objects and corresponding 2D UIs in canvas as front end

**1.3.1.3Design Software**

We used Unity Game Engine’s 3D UI canvas for visually pleasing UI buttons and score messages. We used textures from open sources to make the scene nice. We have imported designed FBX models from open sources

**1.3.1.4Operational Feasibility**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development

**2.SYSTEM ANALYSIS**

System analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. It is a general term that refers an orderly, structured process for identifying and solving problems. System analysis is a process life cycle methodology since it relates to four significant phases in the life cycle of all information system’s study, design, development and implementation. The definition of system analysis includes not only the process of putting parts together to form a new whole.

A System Analyst is a person who investigates and assimilates information about the way a system is to be developed. The system analyst has to test the correct working of the implemented system using the data. If a project doesn’t work satisfactorily, the system analyst has to recognize the entire work. Hence he acts as a link between the computer and the user. The main aim of analyzing a system is to produce a new system, which will operate effectively, efficiently and economically.

There’re four stages in the development of a new system :-

1. Study phase
2. Design phase
3. Development phase
4. Operational phase

The study phase includes understanding all about how the existing system works and what all is required from a new system.

The design phase is concerned with the designing of the new system. These include designing the interface, tables etc.

The development phase is where the actual programs for the system are developed from the designs in the design phase.

The last stage is the operational phase where the system created in the design phase implemented and its working is observed.

**2.1 DESCRIPTION OF THE PRESENT SYSTEM**

Artificial intelligence (AI) is the technique used in computer games to simulate an independent behavior which can be characters or vehicles etc. There are different ways we can implement artificial intelligence in computer games. It can vary from simple AI to complex AI. If it is simple AI it may not look perfect, if it is complex AI it may consume time, so we need to find out a balance between these two. There are so many AI techniques available nowadays in computer games, so people go for separate libraries or some people even improve the existing algorithm. Generally speaking people go for libraries like “Rain” which is the name of a library that can be used for computer games. There are so many algorithms to figure out a path in computer games like sensor algorithms, A star algorithms etc. Some of these may consume a lot of time while others may not use as much. These are the current techniques and issues for path finding

**2.2. LIMITATIONS OF PRESENT SYSTEM**

Most of the system limits the playing space to have a simple AI program which is fine from the player’s perspective. But in order to have proper AI machine you need to use a proper algorithm and the issue is that we are given a wide variety of algorithms. Even if u use A star algorithm it takes a lot of time. Another issue is that it runs on the same thread as the game which means that the game display and the A star algorithm runs on the same thread , and this implementation can cause a lot of delays like if the algorithm takes more time so does the game. Many games do not support random scene generation system which means the player is able to predict the scenes the game loses its purpose. Another issue arises when dealing with games involving vehicles. When dealing with the physics of the vehicle u must have a proper wheel mechanism, it should really reflect how a car goes through a plane and in many cases you don’t find it. How the car reacts when passing over an obstacle is not reflected. When coming to the issues in point of view most games does not support multiple views.

**2.3 THE PROPOSED SYSTEM-ITS ADVANTAGES AND FEATURES**

We are making a game in two processors i.e. a threaded system which means that there will be no lag during gameplay and that the game will be more efficient compared to the other system . Another positive side is the random scene generation which means each time the game is loaded a different scene is available to the player. We have also included sensor algorithm to let the car find its way around the obstacle. The included A star Algorithm computes the path faster with the help of threads. The game is made in such a way that the algorithmic flow can be seen in the game if required. The database used in the game stores the game scores even after exiting the game. This project also includes wheel colliders and physics, following cameras, sound effects dependent on speedometers, randomized paths and scenes which give a more realistic touch to the vehicles.

**2.3.1Functionalities**

Car Movement

Enemy movement

.

Database update

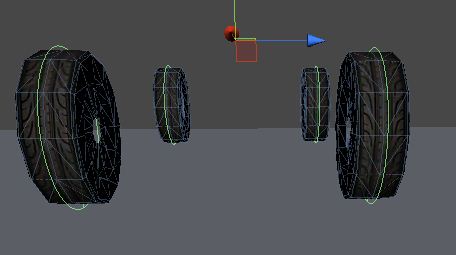
Menu manger

Physics and collision management

**2.3.1.1Wheel colliders**

One of the major functionality in unity is colliders .in our racing game we use wheel colliders for tire rotation .Example for wheel collider shown below.

*Figure2.3.1.1*



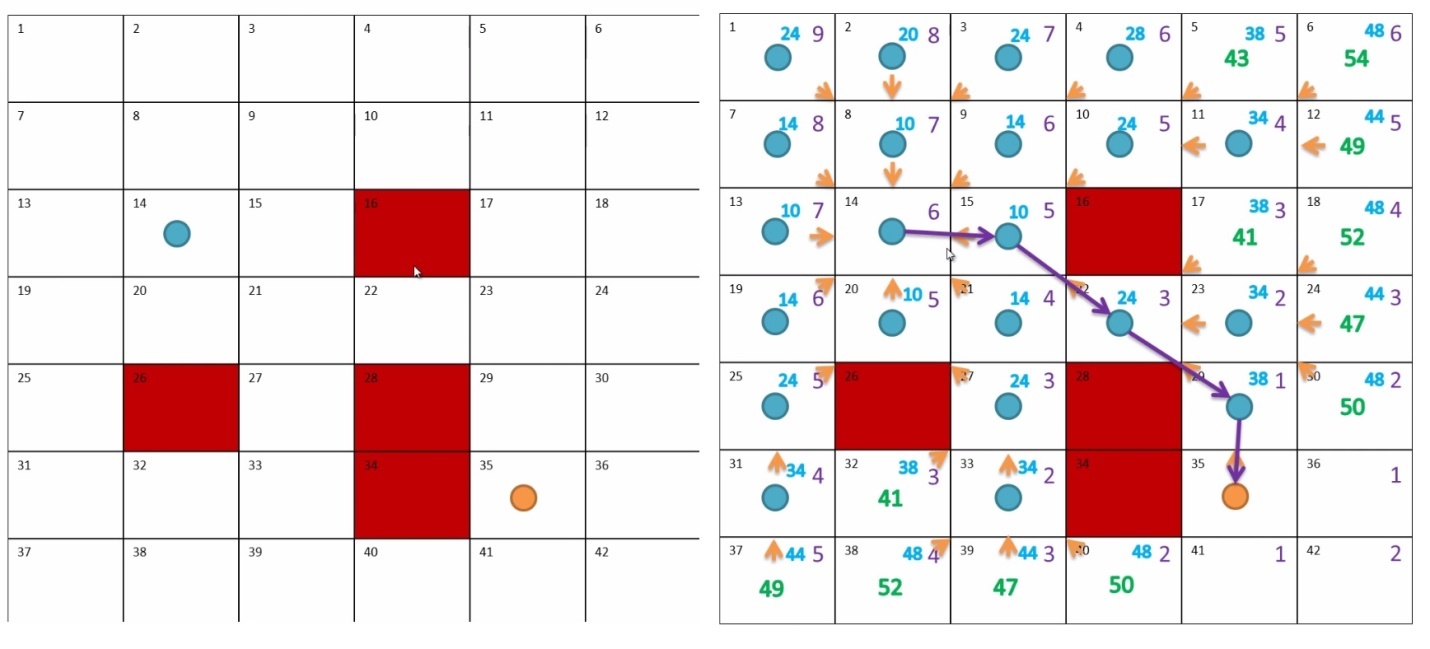
* We use wheel collider in this project for vehicle movement
* The Wheel Collider is a special collider for grounded vehicles. It has built-in collision detection, wheel physics, and a slip-based tire friction model. It can be used for objects other than wheels, but it is specifically designed for vehicles with wheels.

**2.3.1.2A star algorithm**

* The competing vehicle will figure out the path from starting point to finishing point by running A\* path finding artificial Intelligence algorithms
* AI will chase the path computed by A\* algorithm
* We use grid structure for A star path finding

Below given figure shows the function behind the A star path finding

*Figure2.3.1.2*



We referred Unity AI Game Programming,second edition book for learn coding in A star algorithm for the artificial intelligence .

**2.4. REQUIREMENT SPECIFICATION**

Also known as Requirements Engineering is the process of determining the user expectations for a new or improved product. It is the first step in project life cycle which includes meeting with the game interested users and understanding their requirements. This is the most significant phase as any misinterpretation in this stage may cause validation issues later. Our aim is to study graphics implementation in AI domain. We require meshes to represent our models. As the Ai takes much computation time we must avoid meshes that use up a lot of time to compute. This project also requires an external database to store the game score. We must also ensure to make this in an acceptable frame rate.

**3.DESIGN**

**3.1INPUT DESIGN**

We have three input sections

1. Geometric Mesh, Texture and Shader input
2. Input from Keyboard for controlling car movement
3. Input to score scorecard regarding player name

**3.1.1MESH**

* A mesh consists of triangles arranged in 3D space to create the impression of a solid object. A triangle is defined by its three corner points or vertices. In the Meshclass, the vertices are all stored in a single array and each triangle is specified using three integers that correspond to indices of the vertex array.

**3.1.2TEXTURE**

* Texture is a method that texture plane or game scene .we have to use texture for rendering game objects.

**3.1.3 SHADER**

**Shaders** are small scripts that contain the mathematical calculations and algorithms for calculating the colour of each pixel rendered, based on the lighting input and the Material configuration.In the field of computer graphics, a **shader** is a computer program that is used to do shading: the production of appropriate levels of light, darkness, and color within an image, or, in the modern era, also to produce special effects or do video post-processing.

**3.2OUTPUT DESIGN**

Our output is the pixels rendered on computer screen. We group output in to scenes

1. Welcome Scene
2. Main Menu Scene
3. Help Menu Scene
4. Player Scene
5. Database Scene

In output design each pages or design are relatively connected to one scene. Output design need provide proper interface to the user .scene screen shots are given in chapter 8.

**3.3 OVERALL DESCRPTION**

**High level architecture**

****

**3.4 DESCRIPTION DATABASE**

SQLite is an in-process library that implements a [self-contained](https://www.sqlite.org/selfcontained.html), [serverless](https://www.sqlite.org/serverless.html), [zero-configuration](https://www.sqlite.org/zeroconf.html), [transactional](https://www.sqlite.org/transactional.html) SQL database engine. The code for SQLite is in the [public domain](https://www.sqlite.org/copyright.html) and is thus free for use for any purpose, commercial or private. SQLite is the [most widely deployed](https://www.sqlite.org/mostdeployed.html) database in the world with more applications than we can count, including several [high-profile projects.](https://www.sqlite.org/famous.html)

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database [file format](https://www.sqlite.org/fileformat2.html) is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between [big-endian](http://en.wikipedia.org/wiki/Endianness) and [little-endian](http://en.wikipedia.org/wiki/Endianness) architectures. These features make SQLite a popular choice as an [Application File Format](https://www.sqlite.org/appfileformat.html). Think of SQLite not as a replacement forv[Oracle](http://www.oracle.com/database/index.html) but as a replacement for [fopen()](http://man.he.net/man3/fopen)

SQLite is a compact library. With all features enabled, the [library size](https://www.sqlite.org/footprint.html) can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function in lining and loop unrolling can cause the object code to be much larger.) If optional features are omitted, the size of the SQLite library can be reduced below 300KiB. SQLite can also be made to run in minimal stack space (4KiB) and very little heap (100KiB), making SQLite a popular database engine choice on memory constrained gadgets such as cell phones, PDAs, and MP3 players. There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments.

SQLite is [very carefully tested](https://www.sqlite.org/testing.html) prior to every release and has a reputation for being very reliable. Most of the SQLite source code is devoted purely to testing and verification. An automated test suite runs millions and millions of test cases involving hundreds of millions of individual SQL statements and achieves [100% branch test coverage](https://www.sqlite.org/testing.html#coverage). SQLite responds gracefully to memory allocation failures and disk I/O errors. Transactions are [ACID](http://en.wikipedia.org/wiki/ACID) even if interrupted by system crashes or power failures. All of this is verified by the automated tests using special test harnesses which simulate system failures. Of course, even with all this testing, there are still bugs. But unlike some similar projects (especially commercial competitors) SQLite is open and honest about all bugs and provides [bugs lists](http://www.sqlite.org/src/rptview?rn=2) and minute-by-minute [chronologies](http://www.sqlite.org/src/timeline) of bug reports and code changes.

The SQLite code base is supported by an [international team](https://www.sqlite.org/crew.html) of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the [published interface spec](https://www.sqlite.org/c3ref/intro.html), [SQL syntax](https://www.sqlite.org/lang.html), and database [file format](https://www.sqlite.org/fileformat2.html). The source code is absolutely free to anybody who wants it, but [professional support](https://www.sqlite.org/support.html#prosupport) is also available.

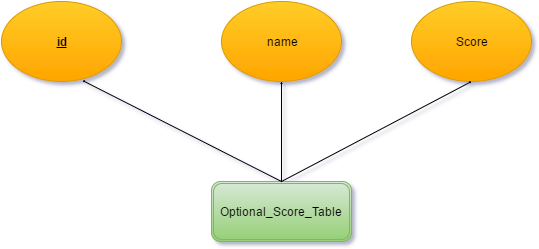
*DATABASE SCORE STORING TABLE*



**3.4.1.DATA BASE FOR PROPOSED SYSTEM**

**3.4.1.1ER DIAGRAM DATABASE**

***Figure3.4.1***



**3.5. LIST OF FIELDS OR ATTRIBUTES**

Here we need four fields that are given below

**3.5.1ATTRIBUTES**

* **Player ID**
* Player ID is used for uniquely identify the player
* **Player name**
* Player name is used for storing score player. If two more have same name then players are identified by player ID
* **Score**

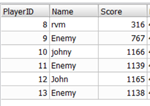
Score is calculated using below given section of code

scoref=(1.0f/Game\_Controller.Instance.Timer\_Player)\*100000.0f;

int score = (int)scoref;

time is used divide one to obtain in such way the with least time will have highest score. Storing score we have only 3 attributes given above .we need only limited space of database for store score

*figure.3.5.1*



**3.6. CONTEXT DIAGRAM OF THE PROPOSED SYSTEM WITH ADDITIONAL LEVEL OF EXPANSION**

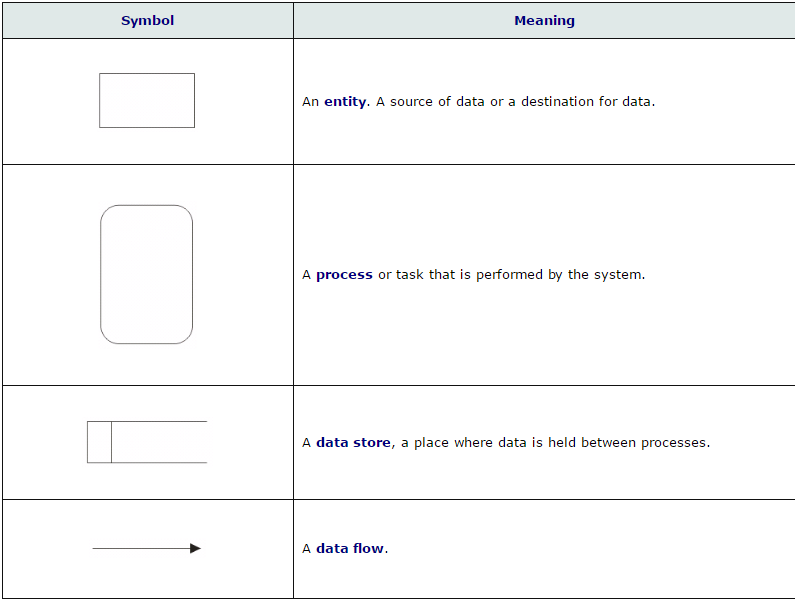
**3.6.1 DEFINITION OF DATA FLOW DIAGRAM**

A data flow diagram (DFD is a graphical representation of the "flow" of data through an [information system](https://en.wikipedia.org/wiki/Information_system), modelling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the [visualization](https://en.wikipedia.org/wiki/Data_visualization) of [data processing](https://en.wikipedia.org/wiki/Data_processing) (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel unlike a [flowchart](https://en.wikipedia.org/wiki/Flowchart) which also shows this information.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

**3.6.2 SYSMBOLS OF DFD**

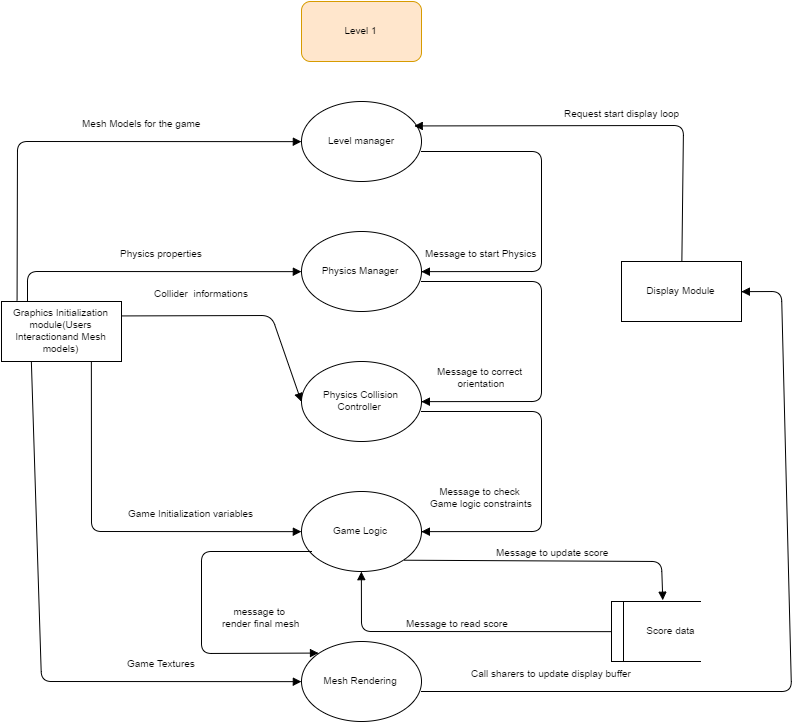


**3.6.3Context diagram**

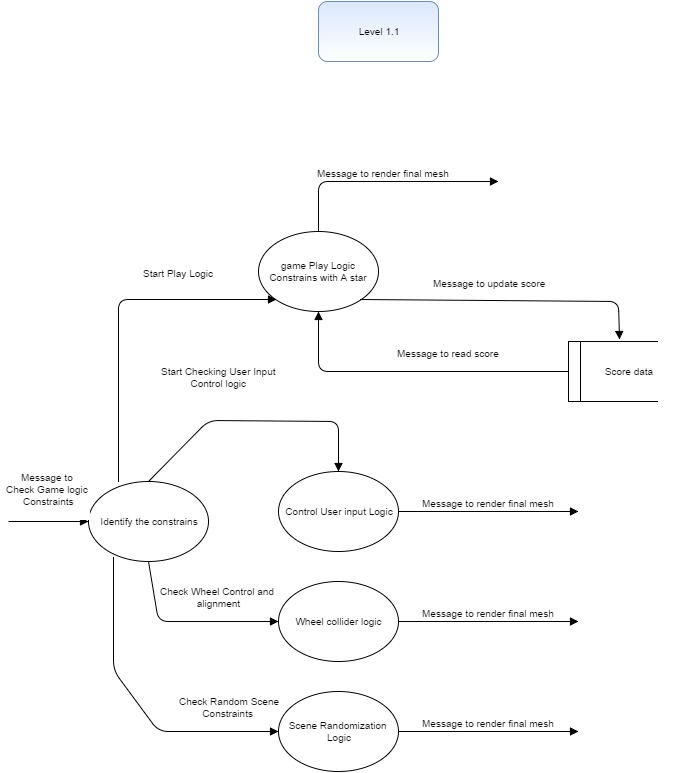
**Level -0**

C:\Users\Ramjith C S\Downloads\Game Context.png

**Level-1**



**Level 1.1**



**3.7.THREE-TIER ARCHITECTURE**

Three-tier architecture is a client–server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.

***Figure3.6.1***

Game

GUI

Unity

AI,Colliders,Physics,menumanagement.etc

**3.7.1.Unity 3D**

Unity3D is a powerful cross-platform 3D engine and a user friendly development environment. Easy enough for the beginner and powerful enough for the expert; Unity should interest anybody who wants to easily create 3D games and applications for mobile, desktop, the web, and consoles.

The Unity application is a complete 3D environment, suitable for laying out levels, creating menus, doing animation, writing scripts, and organizing projects. The user interface is well organized and the panels can be fully customized by dragging and dropping.

The Project panel is where all the assets within a project are stored. When assets are imported, they will first appear here. The hierarchy panel is where assets are organized in a scene. Assets from the Project panel can be dragged into the Hierarchy panel to add them to the current scene. The Inspector panel lets you inspect and adjust all the attributes of a selected asset, everything from its position and rotation, to whether it’s affected by gravity or able to cast a shadow. The Scene panel is a 3D viewport where you can physically arrange assets by moving them around in 3D space. You can navigate the viewport by panning, rotating, and zoomingthe view

**3.7.2GUI(Graphical Interface Unit)**

The GUI, is a type of [user interface](https://en.wikipedia.org/wiki/User_interface) that allows [users](https://en.wikipedia.org/wiki/User_(computing)) to [interact with electronic devices](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) through graphical [icons](https://en.wikipedia.org/wiki/Computer_icon) and visual indicators such as [secondary notation](https://en.wikipedia.org/wiki/Secondary_notation), instead of [text-based user interfaces](https://en.wikipedia.org/wiki/Text-based_user_interface), typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep [learning curve](https://en.wikipedia.org/wiki/Learning_curve) of [command-line interfaces](https://en.wikipedia.org/wiki/Command-line_interface) (CLIs), which require commands to be typed on a [computer keyboard](https://en.wikipedia.org/wiki/Computer_keyboard).

The actions in a GUI are usually performed through [direct manipulation](https://en.wikipedia.org/wiki/Direct_manipulation) of the graphical elements. Beyond computers, GUIs are used in many handheld [mobile devices](https://en.wikipedia.org/wiki/Mobile_device) such as [MP3](https://en.wikipedia.org/wiki/MP3) players, portable media players, gaming devices, [smartphones](https://en.wikipedia.org/wiki/Smartphone) and smaller household, office and [industrial controls](https://en.wikipedia.org/wiki/Distributed_control_system). The term *GUI* tends not to be applied to other lower-[display resolution](https://en.wikipedia.org/wiki/Display_resolution) [types of interfaces](https://en.wikipedia.org/wiki/User_interface#Types), such as [video games](https://en.wikipedia.org/wiki/Video_game) or not including flat screens, like [volumetric displays](https://en.wikipedia.org/wiki/Volumetric_display) because the term is restricted to the scope of two-dimensional display screens able to describe generic information. The best gui provide a good response from users and easy interact with game.

**3.7.3 FUNCTIONALITIES**

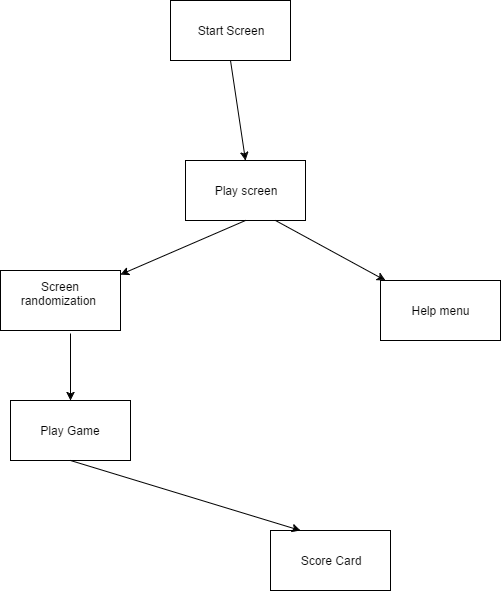
**3.7.3.1Colliders**.

 Collider components define the shape of an object for the purposes of physical collisions. A collider, which is invisible, need not be the exact same shape as the object's mesh and in fact, a rough approximation is often more efficient and indistinguishable in gameplay.

**3.7.3.2PHYSICS**

To have convincing physical behaviour, an object in a game must accelerate correctly and be affected by collisions, gravity and other forces. Unity’s built-in physics engines provide components that handle the physical simulation for you. With just a few parameter settings, you can create objects that behave passively in a realistic way (ie, they will be moved by collisions and falls but will not start moving by themselves). By controlling the physics from scripts, you can give an object the dynamics of a vehicle, a machine, or even a piece of fabric

**3.8. MENU TREE**



**Figure 16.1**

**Contents of menu tree**

* Start scene
* Paly scene
* Screen randomization
* Help menu
* Play game
* Score card

**4.1.TESTING**

**13.1.Introduction**

Testing is the process of executing a program with the intent of finding errors. Testing is a destructive process. The money invested in the development of tests pays off if enough errors are found which would have been more costly if they would have been detected much later. Tests allow proofing the presence of faults but not the correctness of a program.

Testing is the major quality measure employed during software development. After the coding phase, computer programs are available that can be executed for testing purposes. Testing not only has to uncover errors introduced during coding, but also locates errors committed during the previous phases. Thus the aim of testing is to uncover requirements, design or coding errors in the program. A necessary part of a test case is a definition of the expected output or result. A programmer should avoid attempting to test his or own program. A programming organization should not test its own programs. Thoroughly inspect the result of each test. Test cases must be written for invalid and unexpected, as well as valid and expected, input conditions. Examining a program to see if it does not do what it is supposed to do is only half of the battle. The other half is seeing whether the program does not do what it is not supposed to do. Avoid throw –away test cases unless the program is truly a throw-away program. Do not plan a testing effort under the tacit assumption that no errors will be found. The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated and perform allocated functions.

During testing we tried to make sure that the product does exactly what is supposed to do. Testing is the final verification and validation activity within the organization itself. In the testing stage we try to achieve the following goals: to a firm the quality of the product, to find and eliminate a residual error from previous stages, to validate the software as a solution to the original problem, to demonstrate the presence of all specified functionality in the product, to eliminate the operational reliability of the system. During testing the major activities are concentrated on the examination and modification of the source code.

**13.2.Types of Testing**

The test case design methods applied are

* Black box testing
* White box testing

**14.2.1.Black Box Testing**

It is a testing without knowledge of the internal working of the item being tested. For example, when black box testing is applied to software engineering, the tester would only know the “legal” inputs and what the expected output should be, but not how the program actually arrives at those outputs. It is because of this that black box testing can be considered testing with respect to the specifications, no other knowledge of the program is necessary. For this reason, the tester and the programmer can be independent of one another, avoiding programmer bias toward his own work. For this testing test groups are often used, “test groups are sometimes called professional idiots…

People who are good at designing incorrect data.” Also, due to the nature of black box testing: the test plan can begin as soon as the specifications are written. The opposite of this would be glass box testing, where test data derived from direct examination of the code to be tested. For glass box testing, the test cases cannot be determined until the code has actually be written. Both of these testing techniques have advantages and disadvantages, but when combined, they help to ensure thorough testing of the product. In this project I test black box testing. Here I check whether I got particular output from the particular input.

**14.2.2.White Box Testing**

It is performed based on the knowledge of how the system is implemented. White box testing includes analyzing data flow, control flow, information flow, coding practices and exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

**14.3 LEVELS OF TESTING**

The following are the levels of testing:

* Unit testing
* Integration testing
* Output testing
* Validation testing

**14.3.1.Unit Testing**

Here they test module individually and integrate the overall system. Unit testing focuses verification efforts even in the smallest unit of software design in each module. This is also known as “Module Testing”.

The modules of the system are tested separately. There are some validation checks for the fields. Unit testing focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. Unit testing can be performed from the bottom up or top down. For each module in the bottom up testing, a short program execute the modules and provide the needed data, so that the module is asked to perform the way it will when embedded within in the larger system. When bottom level modules are tested, attention turns to those on the next level that use the lower level ones. Top down testing begins with the upper level modules, since the detailed activities usually performed in lower level routines are not provided, stubs are written. A stub is a module shell that can be called by upper level module and that when reached properly, will return a message to the calling module, indicating that the proper interaction occurred. No attempt is made to verify the connection of the lower level modules.

**14.3.2. Integration Testing**

Data can be lost across an interface: One module can have an adverse effect on others; sub-functions when combined may not produce the desired major functions. Integration testing is a symmetric testing for constructing the program structure. While at the same time conducting to uncover errors associated within the interface? The objective is to take unit tested modules and to combine them and test it as a whole. This testing is done with simple data and the developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance. At the culmination of the black box testing, software is completely assembled as a package.

**14.3.3. Output Testing**

After performing the validation, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the output generated or displayed by the system under consideration. Hence the output format is considered in two ways: One is on screen and another in printed format.

**14.3.4.Validation Testing**

Verification testing runs the system in a simulated environment using simulated data. This simulated test is sometimes called alpha testing. This simulated test is primarily looking for errors and monitions regarding end user and decisions design specifications that where specified in the earlier phases but not fulfilled during construction.

Validation refers to the process of using software in a live environment in order to find errors. The feedback from the validation phase generally produces changes in the software to deal with errors and failures that are uncovered. Than a set of users sites is selected that puts the system into use on a live basis. They are called beta tests.The beta test suits use the system in day-to-day activities. They process live transactions and produce normal system output. The system is live in every sense of the word; except that the users are aware they are using a system that can fail. But the transactions that are entered and persons using the system are real. Validation may continue for several months. During the course of validating the system, failure may occur and the software will be changed. Continued use may produce.

**14.4VALIDATION, TEST CASES AND TEST REQUIREMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Test case title** | **Description** | **Expected outcome** | **Result** |
| 1 | Car Player movement testing | Put the Car on bridge. Observe the orientation in Scene window. | It moves to the desired location as per scene target | Passed |
| 2 | Test Car Player Turn | Put the Car player on bridge. Observe the orientation after rotation in scene window | It moves to the desired location as per scene target with proper turn | Passed |
| 3 | Test Enemy AI | Checking path tracing | Enemy traces path | Passed |
| 4 | Test Random Scene | About to run random scene agorithm | Random scene generated | Passed |
| 5 | Check Thread | Check the parallel execution | Display never get blocked | Pass |
| 8 | Check scene navigation | Pass through all menus | Not stucked anywhere | passed |
| 9 | Check collision | Check how the wheel collider aligns with bridge | It simulates wheel movement | passed |
| 10 | Visualize debug draws | Write code for scene not for program | Path will be displayed on scene | passed |

**5.IMPLEMENTATION AND MAINTAINCE**

A crucial phase in the system life cycle is the successful implementation of the new system design. Implementation simply means converting a new system design to operation. This involves creating computer-compactable files, training operating staffs, and installing hardware, terminals and communication network before functioning of the organization.

In system implementation, user is crucial for minimizing resistance to change and giving the new system to a chance to prove it worth. Training aids, such as user friendly manuals, a data dictionary, job performance aids that communicate information about the new system, and ‘about as’ screens, provide the user with good start of the new system.

The important aspect of system analyst is to make sure that the new design is implemented to establish standards. The term implementation has different meaning, ranging from the conversation of a basic application to a complete replacement of a computer system. The procedure, however, is virtually same. Implementation is used here to mean the process of converting a new or revised system design into an operational one. Conversion is one aspect of implementation.

**5.1 IMPLEMENTATION PROCEDURE**

Step1: Study the requirements properly and make clarification about the details of project with guide.

Step2: After clarification we defined the hardware and software requirements for the system and all software’s are installed.

Step3: Database design and GUI designing had been started after defining the technical requirements.

Step4: The GUI whatever we developed will be connected with the database.

Step5: After connecting we had done the coding for all the controls in the GUI.

Step6: For all modules unit testing and integration testing had been performed. All modules in the system are made secure by applying sessions.

**5.2MAINTAINCE**

**Software maintenance** in **software** engineering is the modification of a **software** product after delivery to correct faults, to improve performance or other attributes. A common perception of **maintenance** is that it merely involves fixing defects. We have used proper classes such that it modification can be done properly and it is done with minimum softwares. To detect errors we have simulated AI progress by another program that shows colors for open list,closed list and selected path. We also keep the code in version control system called git. We get proper version with corresponding update details

**6. CONCLUSION**

We have coded a 3D car racing game with random scene generation . We have used A star algorithm for finding shortest path ………We have made this project in vector space with proper geometry manipulation. We have successfully made infinite scenes in limited memory limit. We have score board and database system to store score board. We have two threaded system for fastest processing ,one thread is for random scene generation and another one is for game play

**6.1.SCOPE FOR FEATURES AND FEATURE ENHANCEMENT**

Game is a product of collective work of large number of people from programmers to artists . Here we studied one of the applications of AI in Game domain. We have used S star algorithm. We can look for more efficient and fast algorithm than A star. There is score for using advanced A Star Algorithms too. Since unity is provided with GPU coding facilities we can think about adding parallelism to path finding. We can also think about mobile version of it . There is scope for artistic work in this project

**7.REFERENCE**

* Unity AI Game Programming, second edition,Ray Barrera, Aung SithuKyaw, Clifford Peters,Leverage the power of Unity 5 to create fun and unbelievable AI entities in your games!
* Learning C# by Developing Games with Unity 3D Beginner's Guide: Learn the Fundamentals of C# to Create Scripts for Your GameObjects
* <https://unity3d.com/>
* <https://unity3d.com/get-unity/download>

**8.APPENDIX**

**8.1CODING**

These are the sections of code

**8.1.1Game Controller**

* + All objects are stored her. This code is not destroyed and is active from the beginning till the end.

**8.1.2 Menu and scene Manager**

* + The purpose of this section is to manage menu items and navigate scenes. The code of this part is responsible for the above mentioned functions.

**8.1.3 Code to manipulate car physics**

* + This code will control the player car movements. It will control the turning, forward movement, braking and the wheel colliders.

**8.1.4 Enemy car code section**

* + This sections consist of the AI systems and also tracing the path found by the A star algorithm.

**8.1.5 A star algorithm section**

* + This section is responsible for performing the A star computation for calculating the path which must be followed by the enemy car which is performed on a separate thread.

**8.1.6 The Database code section**

* + This section handles the database functionalities such as updating the score. It creates a score based on the time taken by the player to complete the course.

**8.1.7The help and main menu section**

* + This section creates the help menu and helps the player to navigate through the game to start the gameplay.

**8.1.8Scene generation code section**

* + This section is responsible for the creation of random scenes in the game and also the random placing of the player car within the game.

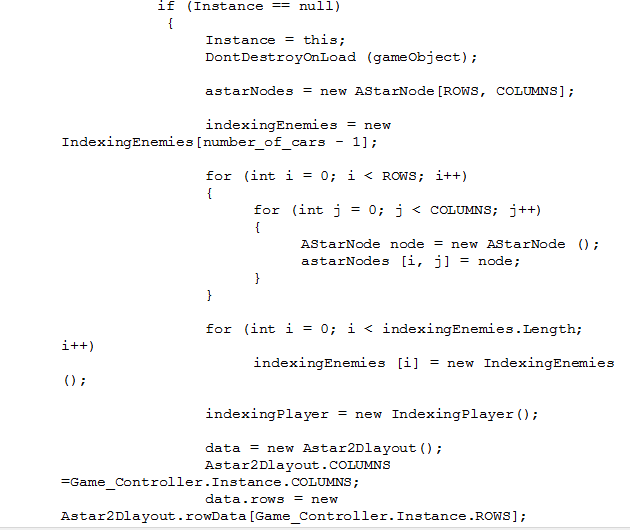
**8.1.9Multimedia code section**

* + This section is responsible for the effects of the game such as the sound effects and the background music.

**8.2.PROGRAM LIST**

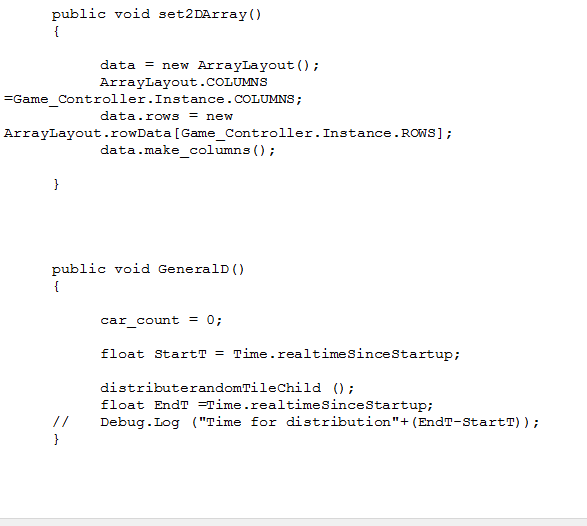
[**Game\_Controller.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.7&disp=safe&realattid=f_j1uqg51t6&zw&saddbat=ANGjdJ-_loUVOy1_zhZF-9LgzP9qWgKPt5Fcp4uffs3JO2BZIom_IXnH8dgpw9KtSdEkFMicWvGmrRoeHmv5ClutfrJ7LV9xROTXh0X82h-wXAHRorN1UQ6V46WABHZx5s-LinwqKjbGqOpb37JJNSQyfIfP0hF0o3GJvIMB2YeU_C6RHhTPyvkzUTvO-LmhQigN2qBgogxg0AyI-WnWDW-6VTPULgGfYsrrOar77QOjPYKCOz-gZQ3-8LPC8yU9ukGGJLX8H4ykjsJ8Ka_ixITGMS5nwC8pRME2n4aju5yosfY9YaiE2UWDYgTAIPI5HY7qOZ_qIr376EI7kHA1rLOBUOHv5FLqiDcK9GG5i-wfH1V9QsyhUa6Gt1xeCjF6wWVdR5NaHiRxrWrwirvEhr9J7r1GN14M4oT6LTWVcTf2XnJQ2X2Kv4WgPojJjF5nmcrHoRshzvL4nYbz5hRn7FdW4Mtubb8vi9rfR3E_IsQGiCsSt2b-MLP5tiCow9_y5tJwyrnNzeGBd3eh8G4O-qknAhanuxnaa6sBNAhLYuOM5I_LsQdAJZHgoAopKTXiQcFQd3WbgAwoM98GQUOm1Kt4Lgasw8EvWcta6s2v3Y3mGD52tqiKrirlfc9mxRlVs4nycmE7qJV9BonDg4O4wE2CvWXD7Ex_sald84LiPQ)

***Figure 8.2.1***



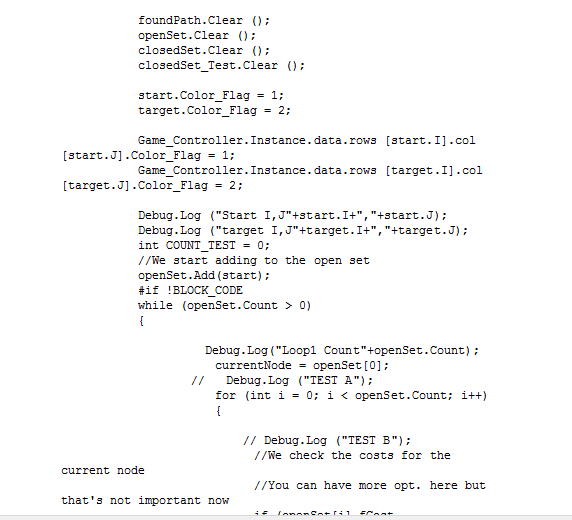
[**rootScene.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.8&disp=safe&realattid=f_j1uqg5267&zw&saddbat=ANGjdJ8OQH0DJEF2ekw_duxP5B2irA7uZB0WsWIEeGYDsB-1N3GW1wejrxFwuKxW1PBFfocGN7zmkWI0SnDXBtRxgutbbE7tHSBVU_KBqqAcg0qEumndwT-LCUH9UgCQf9KhRW41pR_VlUT8zszpXeXss5V-M5AlcI741K0P0CWZsW8H20_8kmcaQOdX2LPSe8iMzZx92W4AifQadZ638pD7WY50mQ7yfqrBFy9x6UNwWrUAXHfCFnkPfserJoa7r8i5qOEjw-cx_2Scvgugi_nFlgk24oowgrs0xwk5F8bp_n8pmsYuM84OXurrk0MblKUexYYI8Sz-mMCo-A35mcmEHOJKuOL3FFCSQyl0zwd5XnROvpujyLFh5nw3kBUbT-73US32bCuavcznWdtxnkM26BvzQ5D3KF3ofH21BiShMDiz3QrmIXhl6qyuyJe2AUkW9b1PtSJWB1VISQhdTMFyMUsJMm34MgvFYoExODFcndk8NVPM1lfljmIh0CwqTHCrv9zwZDh--cKOVGLdf3K_8LZZjirxR6K2NFWMMnrl5RUNwUJKBs0Xzw3Zrnq-fVjToyYMJYeJsGjm-TaXR3mOTSpYZ_gL8xCvl9FCilkcBvz293T8ZoHwx_O3MSqu90PUg6rd6eTM2k8ERB1DpV7hH7QUelAp6HKigQ3T2A)

***Figure 8.2.2***



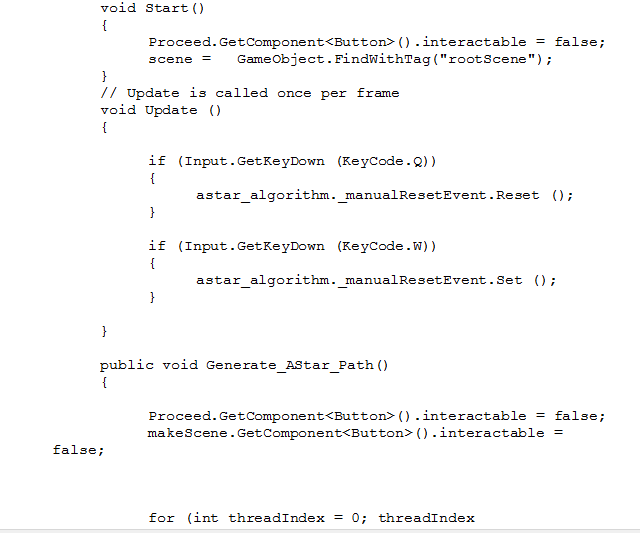
[**Astar.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.10&disp=safe&realattid=f_j1uqi9xc9&zw&saddbat=ANGjdJ_9YnOci5msxI4Ux6sIJQr4Q_fLVH8_MRmTvTA5NettwXFT6kfjLQZGuDi_PZgVMx7srnyl66IpO1Gzg5lwW_ANc3PyTpE_SpbEXonZ2RjAUPutDmQ5nj1fJAmtd7gLn9lxS4rZebYTxiM4CLLbBrWFliJ9S5TWZy70T3D3RIyz4KONK_sDdPgrWit-TWAHQNxlQHb74LcEypfUe6VMtaEVcyJhkReddMZDi8YVvPw_2aDAmQ7NGR54ca4npkR2PYa5YiOnQCHds_L16-lREGkonUWJS5R8_uAZeUVbnt3mRTGEma1hPF6zoDCCVvaGfpNaVoyf_E7kmbrm4gL4zWdCEiXKAhYlYt7Hs5rlEJioxl4wqd1WoALWaQjjyqOOt-iSKiwQBcnLARlnEpuzTP6EkFElglqMDahMFYnzLU4S2wlzpcP6UTH4EUVqNiPNKJtLYjacROHSTnE5ZU5APM3tGRciGWyl55bN2oqZcGUmt7scHGblSIbLCgw4AmlIYBY7nbuo0JeMB_A9VTUy_SnOrNIQAfL7dw8vSN4aOZHWtzIqoYep5whB5IbhupFbhhF3sQ2xtyhsptAfxcjyb0kDrB6z2k1M0pfqdAPSvwjxcH0FjReHZryNwKmWvrstF1ffioQ5lqYWHinKBsjMBPA6W2dF086O6aaf_w)

***Figure 8.2.3***



[**MenuManager\_Selector.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.11&disp=safe&realattid=f_j1uqi9y512&zw&saddbat=ANGjdJ9E625oTDVEX7nC_F-gmT1XjQav_mqKwEBR9ZF1qgn3LC4xDaP66NuDCPAT5E1wzmsvcp0l6Vc7YM7Ka4jkq0VetmnooUUx_ebaNji9HuGGEiUXvBPu71OJm5gIuSgnBu3ML4IwXPUPdrHLidI0SViq80mUZy7x7kwR0TtZMuxZgjqD0ID2UIDvG_CHtyfF8KBYu0GhJBFixmsJQkuii0pvcivXSZGR2pNAYIRevgTRIciV7_er8JU-l6XSCj3W-UV3Lib7jxmWlBeyJWxBPZj7SodAasRL1Gwqb4fgPMe4gJZAoQ37hSd671wW2F6_rrZv_ptUM_hB7QMGsJtvs2s60727nbZJ5ycW_m7GgeeKWXUcxPVxESOk60C_1LbRq7UI_k2Kf1GZD2omq8CMvph_5tQarA0by7I0ewvqhIfNUfn0GlDONK116c9F3IOmUNEnvD5N456EIXXPvgjLdEVuI98t8cBZG2-A_4S_l5T9N9Ue5TOxYChfGXipwjzuCwR-N_7SEV3s18PDj2P_G2T-1kwXWbKn_A0vV5V6t_AxMnSdE9gPyZK9u68EmG-SIyNPTEwpP4-sM1EB18GLi_Jzc4Z3-v8ra_b99H2rCqyPWeUTA7Y9X3RPz4h8JfZ2oagBY0Fem-WziXdX_r1zrkl4-wzU3Moj-csdcA)

***Figure 8.2.4***



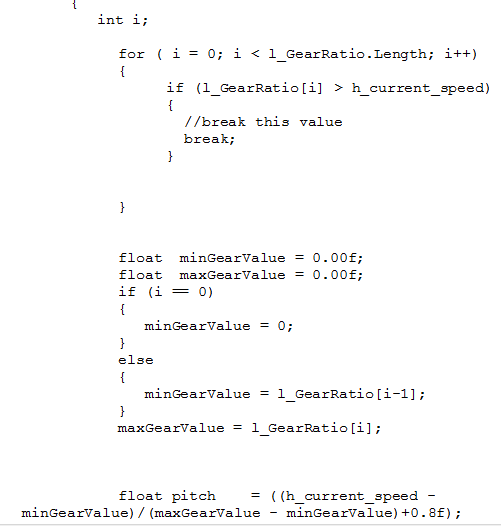
[**AI.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.13&disp=safe&realattid=f_j1uqk1yf12&zw&saddbat=ANGjdJ-6nzuTQrpf_ud_O1-YLQDn4wsyS11dUd_geRufky8pgZnXOs0GKzTZUXr5ywk8bETyKDEJVEBgL0wEcpmJnjeNiCmup8KtduKLH2IBtIjOTpXvXfPoWfchhxrsgMIObenuJOC01YXbM8goqHdLe0CMupc_L7SgjOJBNXJOR7Es6LfY1DfJm1TYue4IfLVQrWZnG52SsLT8zTccuIO6PWSBxyNUE7mdRldUqBhK05hANE53qla5hf-bB4zCR5VsI4Riv6-UPlaYFkAU7Q9qLAbHi5o-LGiU49Elh5WXHsMcU1aOLnA46Z4VZL_cRPOFY3Y0SAzQJeV4LKMds6A5jmR1DvVnXcH8pmxHD2qDWH6N9SZWXkpBUPoW4VaODW3IXn9jRo1CHQQvQU1eOr54gIrQM8cyQpXtrqno8C7r8rN8SdSqOL2S0JEfvMFmgNJsrXtZoDypmXapYQr0K3dwt25gHO8GJWvCP2RWAc-8XlBdc1nP5--6SS4OxMewgIzuQUMOYvz53VA70ehlqqaoyc0eVqOgOrVZc04CN2KIXDQBpmPD3yzpcZHpkLloDhKWjSsavzWlifQzaKrqKAARpLTsdT435SAfLjHUj2tnqemVOjUykPMIZ_35xu5WF9R4fqEpTI3whfvjpk1irp7e7CgpkKRouoddJZwCmQ)

***Figure 8.2.5***



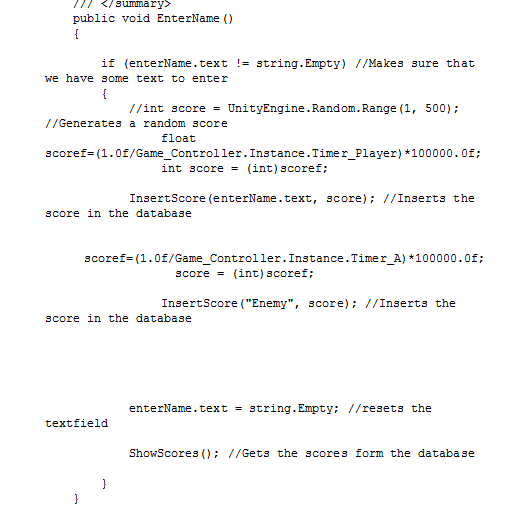
[**controllerCar.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.15&disp=safe&realattid=f_j1uqlg6214&zw&saddbat=ANGjdJ-lQ8fqwmquz1SSSD6mSCyLTgjxpWCpqGbUxmGTuozirzfkXT9W7koJVsXYQ1Opn-vSPFLsSLmH8HQg7K_V8MIFQFV-H2n1DT7y4fvibPCrMLHPDBRU7uPtjEwesTLbll8Q6ZOzRn33YxCijyKLCja9qzJWbXytJmd-8alpihQUbzuNhiY_bGUwED8tUsj8pYGbxm0007k4MuYzuCyMkAOx4f9Ohcri2gSrwDqvJWussavjQx0Ov5PU_iyG6wMV9Ssvl_coTx_lvgdPPEsMkD6xPp9_kMxZIbEQdR0Nt4htpzfBDyVsXDsvXvSP2Oy1grq5_JDl18JafFsRtLmGqWuhskO_zfRLF9bM2rNSYYqxcLhfOQUZp63IpqZ06LCu4yyuszTT3FUa2oAx5UJpV5A0IoHSJkE0jV_It2CcjEglmakT1sJ1NQw1XUAuS0J9ndjhZpYTLU2pW9YFnyWXisikNb0jlOWetcFHYpxDvwcJvZluscdgM4SWcOS5gMvkhMmvVtFcR3JlPRempvoYzSji-k-5timJM9mXxgaz6YO2AGj3PYkEAR3GcbrCiBQBivBuc-InHb6_Q2lrLJWLeiOQu8VGb4VWgjAvqa6AI_d2ZCCLQhc6i7GgPT2hgds4jvf-jzbdojnJTPDvpZy_kvl-y2K0H61Obo1Hvw)

***Figure 8.2.6***



[**HighScoreManager.cs**](https://mail-attachment.googleusercontent.com/attachment/u/1/?ui=2&ik=69859c1fd7&view=att&th=15b9b0d3d619295c&attid=0.16&disp=safe&realattid=f_j1uqma1815&zw&saddbat=ANGjdJ-UZF4gNJKPKFKq4QHT0C3T5D3lG_d1HIZGAdIiH1c4SGycqgqLGFhRCkP4h8ewVuPaVc20AK-g5HjpkVzi3ialXK3FFmgsrZFWiZnmTV6GEw6U4pDJYuBatQu1NJiPWNyrhr5GQiQHim2dJ_udRMoM3j8gKivkzC85fAjUSt2_Yi2MjUmBfyz4yQ6DSttPYJQ19hv4KHodqDRkKtJ_bRL9sALHcuL-bNP49JuITOe9IfdBcypgFcWk3Hqr-FGr-SlC8iSWZcwyKUDPsb1DtlXw_Mjn0Ic9xFobhEaHS04j1hV7H8rW7UTOeYHcYOVJh8HReOYyznCEBj4HYUhvsuYLV6Kvos_Qah1U64yGMkBpxAUU0BXqcsO_yC0CGyska-6b5axWOwFb_yotJ0d62FgJwpM96989CR0eAYvn3AL_P2Zm58C4qRIb9Jvh3P27cIC0naF-ZHBforimGndSNBty_sTEiiQ8tTeMetWOG4ois8gaGGOCbNJn62C64ITb7f56KZsNpZ31olziwYNe_5zp6kZt1FeSLka2oP6Bclw6ICJI6mbS-zKfxLe68vYMZ1HZ318WpUekR1Oho7z5sdpQM2imQ_-qo48vIAXpxclAZTwgOe3DoOkxLPbrG6LUP67b38NdcX_nHtcbfnUoAGnUehBb2b_rjRVqlQ)

***Figure 8.2.7***



**8.1.PROGRAM – FILE TABLE THAT SHOWS THE FILES/TABLES USED BY EACH PROGRAM**

***Table 8.1***

|  |  |
| --- | --- |
| Program | Files |
| HighScoreManger.cs | HighScore.sql |
| Gamemanager.cs | FBX meshes,shaders |
| Menumanager.cs A | FBX meshes,shaders |
| Game­\_Controller.cs | FBX meshes,shaders |
| rootscene.cs | FBX meshes,shaders |
| AI.cs | FBX meshes,shaders |
| controllerCar.cs | FBX meshes,shaders |
| Astar.cs | FBX meshes,shaders |
| Timer.cs | FBX meshes,shaders |

The above shows program names and corresponding file table.each details has been specified in coding section.

**8.1.1TABLE NAMES**

1. Only one table named high score. Just write about how we make time to point by taking inverse. And also write how to use sqlite manager.
2. We use sqlite for store score and player details, we have highscoremanger code for calculating scores of players and store properly to the database

***Figure8.1.1***



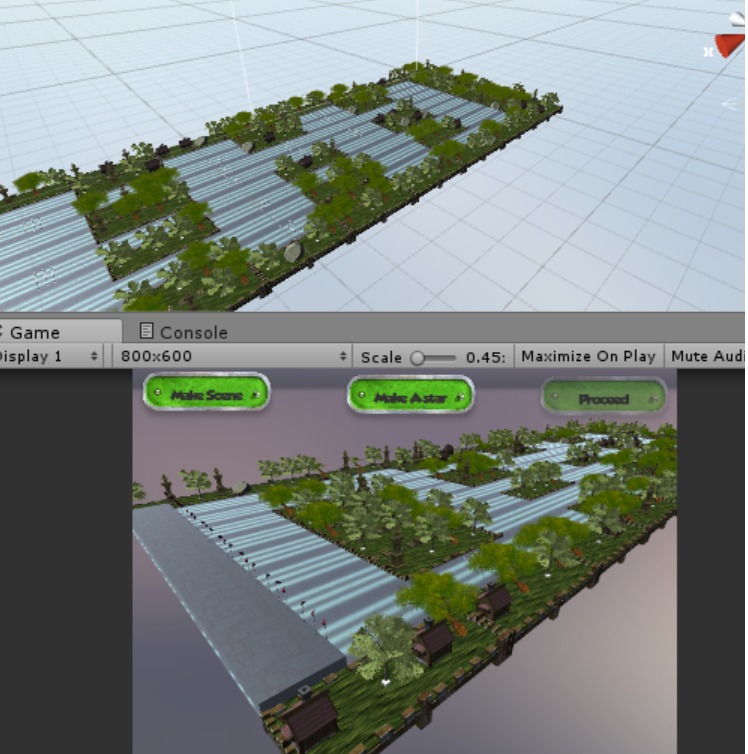
**8.2.SCREEN LAYOUTS FOR EACH DATA ENTRY SCREEN**

***Figure 8.2.1***



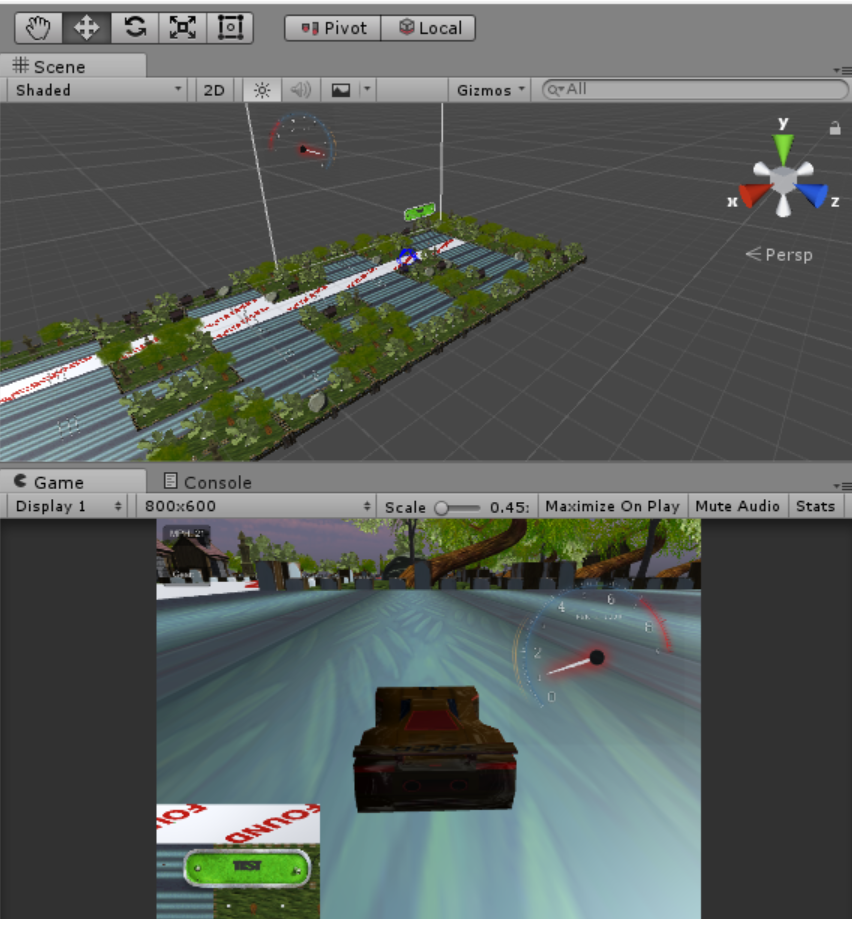
***Figure 8.2.2***





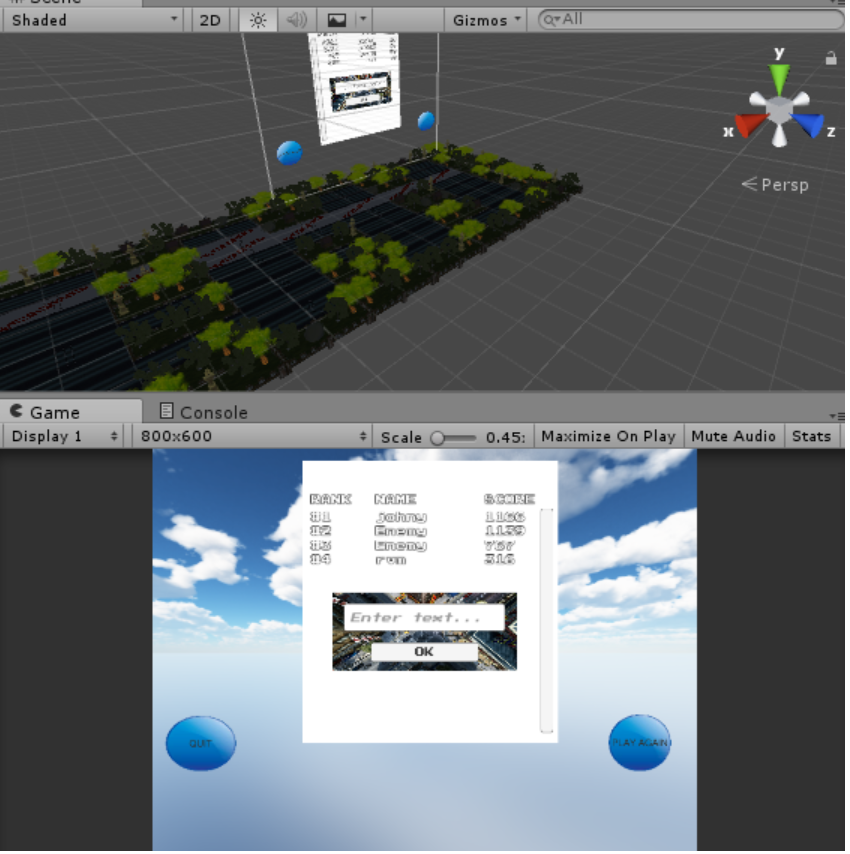
***Figure 8.2.3***

***Figure 8.2.4***



***Figure8.2.5***

This figure score card .in this scene we have text box for entry the player name



***Figure 8.2.6***

This figure shows a database table with player score and name with player id

