CAR ROAD FIGHTER GAME DEVELOPMENT USING UNITY3D

Rohan Bhagat[1], Rohit Bhanore[2], Pooja Bangar[3], Prof. A.E. Patil[4]

[1 2 3]Student, Information Technology, Rajiv Gandhi Institute of Technology, Maharashtra, India
[4] Faculty, Information Technology, Rajiv Gandhi Institute of Technology, Maharashtra, India

ABSTRACT

Abstract— Game designing is a really interactive and creative part of modern IT culture. This has motivated various developers to create interesting games. Thus, we aim to develop an entertaining and fun game based on our childhood favour where we are focusing on developing a 3D racing car game, using a process based upon agile development; an evolutionary development method. Our game is a single player game which provides multiplayer functionality as well. The game will consist of various road maps with power boost challenges and speed timing to get user more addictive and entertain.

Since it is a racing genre game, the multiplayer gaming will focus on races with friends and let the users compete to be the best. Apart from this even time depended mode enables players to track and beat their own high scores. The platform of the game is developed using Unity 3D game engine which is multiplatform software building for games and main coding is in C#. Hence whole game development process will cover implementation of real-time graphics, physics engine, network support, as well as sound effects and background music. Thus we will try to create a game which is extremely entertaining with great graphics to make it visually appealing.

Keywords: Unity engine, 3D car racing, roadmaps, multiplayer, Entertaining game.

1. INTRODUCTION

Developing software applications is a time-consuming process, and with time-consuming processes come high costs. To address this issue, several software development methodologies, agile software development, have become widely used by software developers. One of the software development methodologies is the evolutionary software method, which allows the project to evolve through different stages of the project and this approach worked on our project where we choose to develop a 3D graphic computer game. Some requirements for the computer game development to 3D environment like 3D graphics, graphical effects, multi-platform. We decided to develop this game in Unity3D platform which gives better working these requirements.

The game is a single player type in racing genre. A racing game is one in which the player generally races other vehicles to secure first position. Unlike the racing genre, we feature only the main player car which has to beat the time set by AI. As a car is vehicle which needs petrol to continue its speed, in our game we will serve power boost on the ways to user alive on the game. The speed and power boost are two main objectives for user while playing with game. The multiplayer type game maximum six player can play same game with their respective environment while races other vehicles to secure first position. For multiplayer game users need to connect each other by LAN cable (Desktop version) and WLAN for android users.

To develop 3D game we have need develop 3D gaming objects, environment and UI, we should use Adobe tools. Adobe tools are also helpful for gaming effects for better graphics. As unity is multiplatform engine, we decided to develop game for desktop (windows platform) and android users. So our final product will be run on windows as well as android devices.

2. STEPS IN GAME DEVELOPMENT

2.1 Conceptualization and Initialization
The game development process includes conceptualization of the gaming idea. The game play, the age group, the ratings, game mechanics are all decided in this phase of game development. Various researches are also done to see that the idea is unique and also that it can be implemented.

2.2 Game Design Document
A game design document (GDD) is a highly descriptive Living design document of the design for a game. A GDD is created and edited by the development team and it is primarily used in the game industry to organize efforts within a development team.

2.3 Technical Requirements
It includes deciding on the platform based on the targeted audience. The tools and the materials required for the game design are also decided in this phase. For our project we would use windows for development of whole process with Unity tool to be installed. Other than Unity we would need software like Adobe tools for creating 3D objects. For android users we would use SDK tool and android device.

2.4 Modelling art and Level Designing
Based on the game concept, the theme and genre are finalized. In our case, the genre is Role playing. Design Game objects and strategically defining levels of game are main aspect by point of user testing mode. This phase should be define to user point of view which will comes more attentions.

2.5 3D model & Environment art
Game objects are an integral part of game for developing 3D model. In this phase, the different game objects are designed using the various software. Using adobe tools we should create 3D models of cars and maps.

2.6 Developing UI
The user interface, the menu are developed in this phase. The menu is define for users to get in to right direction in game flow. To define UI, we would use C# codes in unity platform. The Game menu will contains different items like play game, controls, sound and single-player or multi-player mode.

2.7 Adding Features & Mathematics
Different algorithm and mathematics are used for score calculation; these features and algorithm are added in this phase. The attraction of users are more when there are some features like points, records to get in play game. The physics involved in the game is also added. (Example: collision detection, path finding—A* algorithm etc).

2.8 Regression & Integration
After the pre requisites of game are developed. The game development process involves different level of phases while developing one game. This phase includes integrating all the phases to create a responsive and functional game.

2.9 Testing & Fixing Bugs:
The developed game is tested on android and windows platform to see if the game features are working well on the device. Also it is checked to see it supports the various devices and android versions Alpha-Beta testing is performed. If any problem is encountered, it is rectified.

3. SCREEN FLOW OF GAME DESIGN

The table 1 given,gives a comparison of the various technologies used in Health Monitoring Systems and highlights The car road fighter is the racing genre game of cars on road maps. The screen flow is the window for view of users. When user will play game then screen of device is important by user point of view. Thus flow of screen should be in static manner. In our game, first of all it show Menu for user interface when open the game. Menu will contains different options like New Game, sounds, controls, options Player name etc.

As shown in fig.1 , to play game we have to select mode, car and map. Figure show static plan of game how to be process to view by user at a time of playing game.
There are 6 different car model and 5 different maps in game design. User can choose any one car and map at a time and play the Game.

In single-player game, we feature only the main player car which has to beat the time set by AI. The game play on user device only and specific to user need only. Player can select car and map to play the game. So single player strategy is simple to play for user and can play by any type of player like first time playing the game.

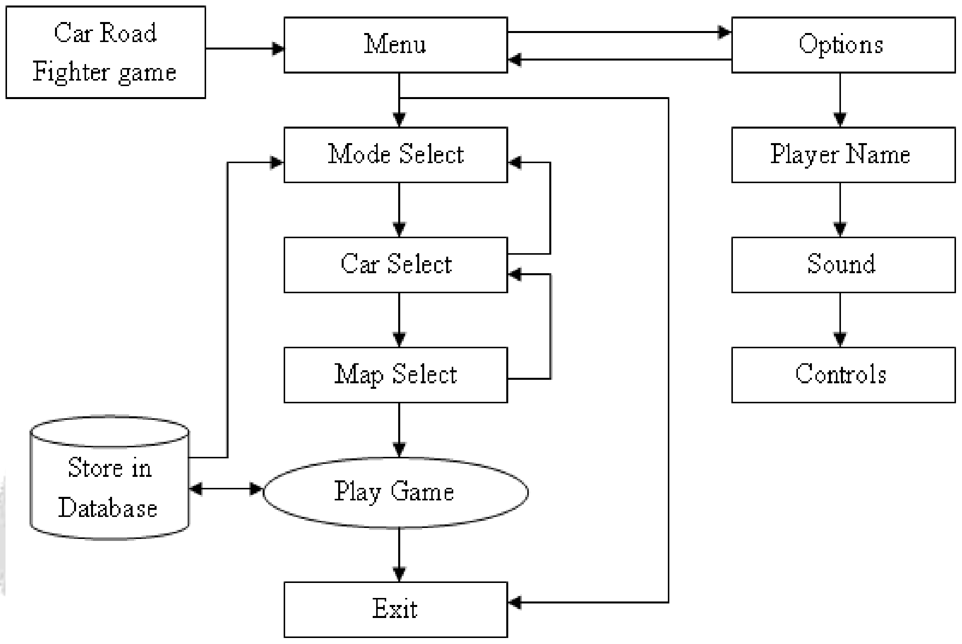

FIG.1: Screen flow of Game

Whereas, In Multi-player, six users can connect through network and play game between them to compute each other to secure first rank in the game. The game can consist of one host device to hosting the game and other will join the game on host server. The hosted device will select the map in this case. As selection of car are user specific for multiplayer mode.

4. GAMING MECHANICS

Every game that is played today is composed of some very common game mechanics: path finding, collision detection and input. These game mechanics have been around now for decades and have been improved on throughout the years. Here a few very common game mechanics should be use for thesis project.

4.1 Collision Detection

The simplest definition of collision detection in relation to games is to determine if two rectangles in the same 2D or 3D space are overlapping. The determining factors for what method of collision detection to use depending on the game design and precision of the collision data needed. In Unity there is a method already created to help any game enthusiast create a game involving collision detection. The method is called On Collision Enter (Collision). It is used in every game design.

4.2 Finite State Machine

A finite state machine at the simplest form is a model of how a system or a game will behave. Depending on the input from the player the state of the game can change. Each of the games described in the thesis project use a finite state machine to some extent.

4.3 Timer

The car racing game mainly work with timer to describe user ranking. Timer is main part of our methodology while working. In Unity a timer is constructed by using a local or global variable set to the desired time in seconds. Then just subtract Time.deltaTime from the variable which will decrease it by 1 second.

4.4 Path Finding

The general definition of path finding is plotting a path from a start point to an end point, done by a computer program or algorithm which is applied to a graph. In many cases the shortest path is the subject of interest to find. In the case of video games it is the same except it is done for a character or group of troops and it plots a path around obstacles on a map.

4.5 Dijkstra's Algorithm

It is most used in networking; routers use it to find the shortest path from a computer to a web address the web browser is searching for. It builds a list of hops it needs to take to get to the final address of the web address. This is done be-cause each hop to the next router is given a weight, and so it finds the route with the lowest weight cost and uses that for the route.

4.6 A* Search Algorithm

Using this method allows the algorithm to eliminate longer paths based off this approximation, in turn speeding up the resolution of the shortest path. Using this heuristic approach makes this algorithm faster than the Dijkstra's algorithm.

5. SOFTWARE REQUIREMENTS

5.1 Unity 3D

Unity 3D is a game engine used for developing games for multiplatform use. It is one of the best features of Unity to allow user to create a game able to run on multiple devices or systems. The coding is mostly in C# and the inbuilt assets are easier to access. Unity can handle and support several art assets and file formats from Maya, Blender, Adobe Photoshop and Illustrator. All these assets are handled by Unity's GUI (Graphical User Interface).

5.2 MonoDevelop

The Unity game engine works in conjunction with MonoDevelop for controlling the behaviour of objects. MonoDevelop is an open source Integrated Development Environment or IDE developed by Xamarin and the Mono community which is primarily used for development in the C# programming language. The C# scripts enable developers to control the logic and behaviours of objects within the Unity environment. In this way, the combined tools of Unity and MonoDevelop enable developers to focus on the development of the AI components of the game rather than on issues related to 3D graphics rendering and physics calculations.

5.3 Adobe Tools

This set of software is used for designing game objects, environment and UI.

6. IMPLEMENTATION

6.1 Player controlled car module

The player-controlled car module consists of four sub modules: (i) the Car Body, (ii) the WheelsHubs, (iii) Car Controller script , and (iv) Car User Control script. The body of the car is the 3D model that the player sees in the game environment; this sub module also contains the colliders, which enable the car to collide with objects in the game environment. The wheels of the car contain wheel colliders, which enable the wheels to make contact with the road and drive the car forward. The wheels sub module also contains a script which animates the rotation of the wheels as the car moves. The HUD component of the car is controlled by the CarController.cs script, which handles displaying the current speed, position, and lap of the car to the player.

6.2 Game controlled car module

The game-controlled car (or the AI-controlled car) consists of four submodules: (i) the body, (ii) the wheelsHub, (iii) the trigger area, and (iv) the AI control script. The body and wheels of the game-controlled car are identical to the body and wheel components of the player-controlled car. The trigger area component of the car is controlled by the CarSensor.cs script and detects when a wall enters into the trigger area. When this happens, the AI controller script is informed so that it would slightly adjust its control of the car. The AI controller script, called AICarControl.cs.

6.3 Environment module

The environment module consists of two sub modules: (i) the racetrack and (ii) the waypoints system. The racetrack is the road within the game environment on which the cars races, and the waypoints system represents the key positions around the racetrack. The latter enables the game-controlled cars to navigate around the racetrack.

6.4 Network lobby module

The LobbyPlayer is created from the LobbyPlayerPrefab when a player joins the lobby. One LobbyPlayer for each player created when client connects, or player is added. Add user scripts to this prefab to hold game-specific player data. This prefab must have a NetworkLobbyPlayer component.

6. RESULT

Racer was tested by over twenty users, who all reported that they thoroughly enjoyed the game. It was observed that there was approximately equal number of players who were able to win the race against the game-controlled cars. This suggests that not only did the game provide an entertaining gaming experience, it also provided a reasonably engaging and challenging game play.

In general, it can be concluded that the Unity platform supported efficient development of the race car game. The Unity platform supports implementing the race car's search for a path on the racetrack with its components of the waypoint system, the physics engine, and vector calculation functions, all of which are not available if the implementation was done using traditional AI search techniques. With these Unity components, the developer was able to implement the race car's search for a path on the track with less effort and more efficiently, and the developed race car can successfully mimic human driving behavior.

Future work would involve incorporating other path finding techniques into the existing game system so that it can be more exciting. The waypoint system can be used to support a different implementation of the search for the race car's path such that it is more dynamically determined based on its current position. This would make it more difficult for the human players to predict the behavior of the game-controlled cars. This unpredictable behavior would mean a more challenging gameplay for the user. Furthermore, while this game was designed for entertainment, the software can be extended to become a driver simulator for educational purposes. By modifying some of the parameter values of the car, such as the braking coefficients and torque values, and by modifying the game environment to resemble that of a municipal road system, Racer can be adapted to become a simulation software for learner-level drivers or for familiarizing drivers with the roadway layouts and traffic laws of foreign or unknown cities.

7. CONCLUSION

Development of the game system was made easier because of the implementation tools. The Unity game engine supports effective development of the game system of racer with its high-level abstraction programming tools and intuitive user interface. These features support developers in implementation of AI concepts so that they can focus on the game logic and ignore lower level development details such as graphics rendering and physics calculations. The combined tools of MonoDevelop and Unity support the implementation processes because MonoDevelop consists of auto correction features for many libraries and SDKs used in Unity. In the Unity platform enables the car racing system of the game to be more efficiently developed. If the Unity platform was not used, the racetrack would be mapped to a set of coordinates or nodes, which represent the 3D search space that covers the track. Then the path that the race car follows on the racetrack can be determined using either a blind or heuristic search algorithm, which identifies the nodes to be included in the path in the 3D space of the racetrack. The Unity engine has built-in high-level abstractions for trigger detection, which also reduced implementation efforts. Also Unity has drag-drop to keep simple to work.

Thus it will easier for us to develop car road fighter using unity which gives effective results as point of view by user. The game environment and effects would be entertaining and statically defined for player.

8. REFERENCES

[1] *Ryan Henson Creighton*. “Unity 3D Game Development by Example Beginner's Guide”, p.42-45, 2010.

[2] “Unity Game Development Essentials”, by *Will Goldstone*.

[3] *James Sugrue*. IOS 5 game development, August 2012.

[4] *Cheng Ming-zhi*. Unity game development technology, p.75-79, June 2012.

[5] *Xuan Yu-Song*. Unity3D game development, p.101-103, June 2012.

[6] *Shi Xiao-ming,Michelle Menard*. Unity game development practice, p.10-13, April 2012.

[7] *Zhao Ke-xin*. Several optimization Suggestions of using Unity to rapidly develop high quality games, p.5-8, 2011.

[8] *SU Zhi-tong,SHI Shao-kun, LI Jin-hong*. Computer Engineering and Design, v31, n 7, p.1631-1634, 2010.

[9] *Peng G,He Y, Sun Y, etal*. Three-Dimensional Game Modeling and Design Research Based on 3Dmax Software ET Intelligent Transport Systems, Advances in Computer Science, Environment, Ecoinformatics, and Education, 2011.

[10] *J. M. Field, M. F. Hazinski* and *M. R. Sayre*, “Part 1: executive summary: 2010 American Heart Association Guidelines for Cardiopulmonary Resuscitation and Emergency Cardiovascular Care”, Cir, (18 Suppl 3), vol. 122, **(**2010**)**, pp. 640–656.

[11] *S. M. Dorman*, Video and computer games: e_ect on children and implications for health education, Journal of School Health, vol. 67, no. 4, pp. 133138,1997.

[12] *M. Prensky*, Digital game-based learning, Computers in Entertainment, vol.1, no. 1, pp. 2124, 2003.

[13] *B. A. Foss* and *T. I. Eikaas*, Game play in engineering education concept and experimental results, International Journal of Engineering Education, vol. 22, no. 5, pp. 10431052, 2006.