**GAME DESIGN USING UNITY 3D (KARESU)**

**Mini Project Report**

Submitted on partial fulfillment for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**In**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

By

S. SAITEJAASH (17L31AO4H2)

ABISHEK ACHARYA (17L31A04C2)

CH. AKHIL (17L31A04D9)

N. JAYA SUNDER REDDY(17L31A04E0)

N. SAMPATH KUMAR(17L31A04D5)

**Under the Guidance of**

**Mrs. R. AMANI**

**(**Asst. professor, ECE Department)



**DEPARTMENT OF ELECTRONICS AND COMMUICATION**

**VIGNAN’S INSTITUTE OF INFORMATION TECHNOLOGY**

(AUTONOMOUS) Affiliated to JNTU Kakinada & Approved by AICTE, New Delhi

Re-Accredited by NBA & NAAC (CGPA of 3.41/4.00)

ISO 9001:2008, ISO 14001:2004, OHSAS 18001:2007 Certified Institution

VISAKHAPATNAM-530 039

**FEBRUARY 2020**

**VIGNAN’S INSTITUTE OF INFORMATION TECHNOLOGY**

**Department of Electronics and communication Engineering**

****

**CERTIFICATE**

This is to certify that the mini project report entitled “**Gesture Controlled Robot”** is the bonafide record of mini project work carried out under my supervision by **S.Sai Tejaash (17L31AO4H2),AbhishekAcharya(17L31A04C2),N.JayaSunder Reddy(17L31A04E0), CH.Akhil(17L31A04D9**),**N.Sampath Kumar(17l31A04D5)** during the academic year 2019-2020, impartial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering of Jawaharlal Nehru Technological University, Kakinada. The results embodied in this project report have not been submitted to any other University or Institute for the award for the award of any Degree or Diploma.

**Head of the Department Project Guide**

**Dr. P. SATYANARAYANA MURTY Mrs. R. AMANI**

**(prof.) (Asst. prof)**

**ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to acknowledgement the assistance and cooperation we have receives from several from several presonsa while under taking this B. Tech, Third year mini Project. We owe special debt of gratitude to **Mrs. R. AMANI (Asst. prof)** Department of Electronics and Communication Engineering, For her constant support and guidance throughout the course of our work. Her sincerity ,thoroughness and preserveance have been a constant source of inspiration to us.

We also take the opportunity to acknowledge the contribution of **Prof. P. Satyanarayana Murty,** Head ,Deapatment of Electronics and communication Engineering ,for his full support and assisitance during the development of the mini project.

We want to thank **Dr. B. Arundhati,** Principal of VIIT and the management for providing all the necessary facilities.We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development for our mini project. Last but not the Least, we acknowledge our friends for their contribution in the completion of the mini project.

1. S. Sai Tejaash,17L31A04H2
2. Abhishek Acharya,17L31A04C2
3. N. Jaya Sunder Reddy ,17L31A04E0
4. CH. AKHIL,17L31A04D9

5. N. Sampath Kumar,17L31A0D5

**ABSTRACT**

**Game Designing**  is the art of applying [design](https://en.wikipedia.org/wiki/Design) and aesthetics to create a [game](https://en.wikipedia.org/wiki/Game) for entertainment or for educational, exercise, or experimental purposes. Increasingly, elements and principles of game design are also applied to other interactions, in the form of [gamification](https://en.wikipedia.org/wiki/Gamification).

The project report focuses on a new portable video game system that will read and play Macromedia Flash games. The main purpose of this particular project is to develop a low-cost game system that will allow individuals to watch and/or interact with all types of Flash formatted files. In order to meet this goal, the group will assemble a console comparable to the Nintendo Game Boy system with the features of a Pocket PC.

**Chapter 1**

**INTRODUCTION**

1. **INTRODUCTION**
   1. **Introduction**

Video game development is the process of creating a video game. Development is undertaken by a game developer, which may range from one person to a large business. Traditional commercial PC and console games is normally funded by a publisher and take several years to develop. 3-dimensional UNITY games can take less time and can be produced cheaply by individuals and small developers. The 3-dimensional UNITY game industry has seen a rise in recent years with the growth of new online distribution systems and the mobile game market. The first video games were developed in the 1960s, but required mainframe computers and were not available to the general public. Commercial game development began in the 1970s with the advent of first generation video game consoles and home computers. Due to low costs and low capabilities of computers, a lone programmer could develop a full game. However, approaching the 21st century, ever-increasing computer processing power and heightened consumer expectations made it difficult for a single developer to produce a mainstream console or PC game. The average price of producing a video game slowly rose from US$1–4 million in 2000 to over $5 million in 2006, then to over $20 million by 2010. Mainstream PC and console games are generally developed in phases. First, in pre-production, pitches, prototypes, and game design documents are written. If the idea is approved and the developer receives funding, a full-scale development begins. This usually involves a 20–100 person team of various responsibilities, such as designers, artists, programmers, testers, etc.

**1.2 BACKGROUND**

Despite the economic instability and crisis deeply affecting the world, the analysts published that the game industry has grown at a rate of 57% surprisingly. Even as I type these words, millions of people are playing games in front of their computers. The reason of this growth can be stated that the game industry can appeal any user with different tastes. Sun Microsystems realized the opportunities in this field and developed a ASSEMBLY CSHARP library known as UNITY 3D a 3-dimensional game library used for game development. The reasons for creating this game with ASSEMBLY CSHARP are many, it eliminates the need for expensive hardware, it does not require a team to develop a three dimensional ASSEMBLY CSHARP game one person can do it, and it saves cost in developing a game. The reason for developing a 3-dimensional ASSEMBLY CSHARP game is that it is a currently growing and vibrant market with more people playing computer games and traditional games on console meaning a greater market share potential and in other to understand how to develop 3-dimensional games more you must have a background knowledge of 2-dimensional game development. The project proposal for this game contains a plan for the development of a 3-dimensional Arcade game.

**1.3** **PROJECT OBJECTIVES**

The purpose of the project is to design and implement a 3-dimensional game written in ASSEMBLY CSHARP using the JAVA gaming library known as Unity 3D. The project includes a complete level of game with documentation. The level will include everything that should be available in an arcade game like the popular Nintendo classic Super Mario game. The game will be a single-player/multi-player arcade game. The goals of this project is to create an easy to use, pick up and play game that could be played by all ages as long as they have a desktop computer or a laptop pc. The reason was as stated above that they are more gamers playing video games every day meaning a larger potential market.

**1.4 SCOPE OF PROJECT**

The scope of the project is to develop a 3-demensional ASSEMBLY CSHARP game. The system shall use the Key Event library under the Unity 3D library to detect when the keys are been pressed on the keyboard to control the animated character in the game (Sprites). The project will be based on creating an arcade game with the goal in mind of being fun. Listed below are the scopes that I will be covering in the development of this game:

1. Single Player/Multi Player
2. PC based
3. 3-dimensional platform
4. Three level
5. Arcade-based
6. 3D platform for GUI and menu systems
7. Testing for 0 bugs in game
8. Written in Assembly Csharp

**1.5 PROJECT JUSTIFICATION**

* Project title: Design and implementation of a 3-dimensional unity game
* Platform: Windows 8 operating system

**“**

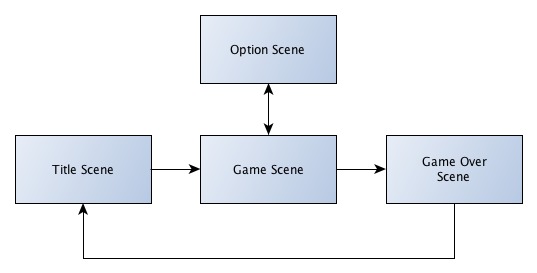
**I wouldn’t recommend [designing computer games] for someone with a weak heart (or) large appetite …**

**--JON FREEMAN , 1984**

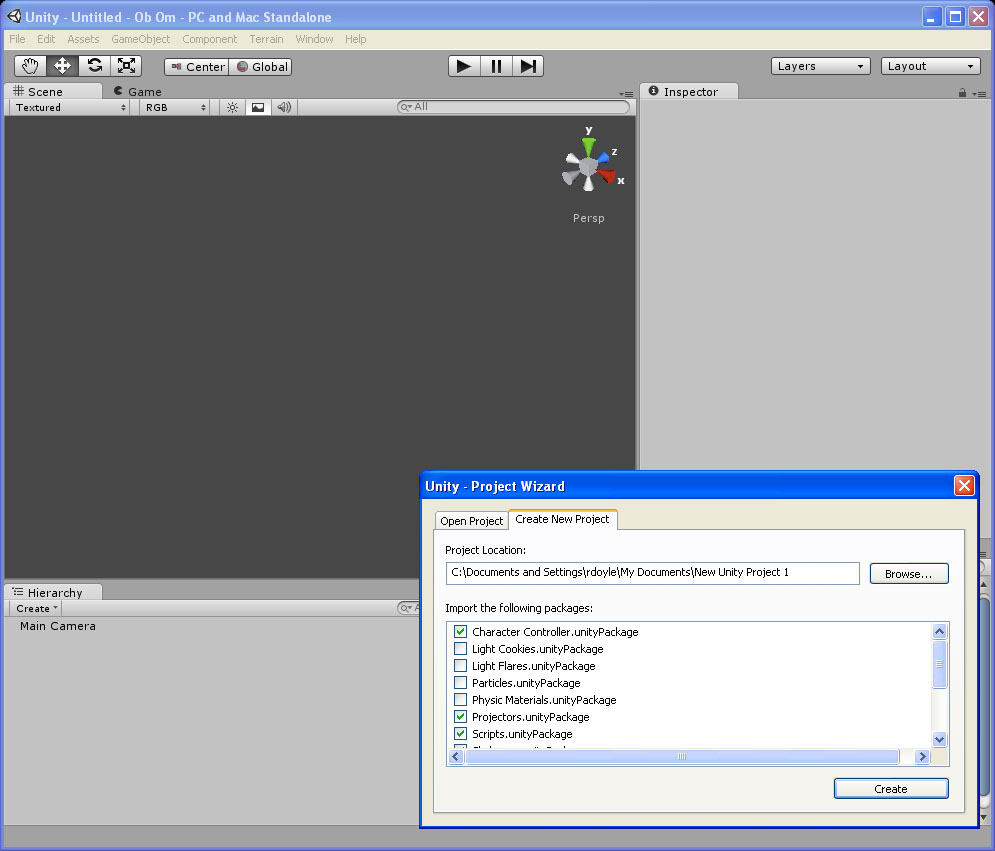
**“**

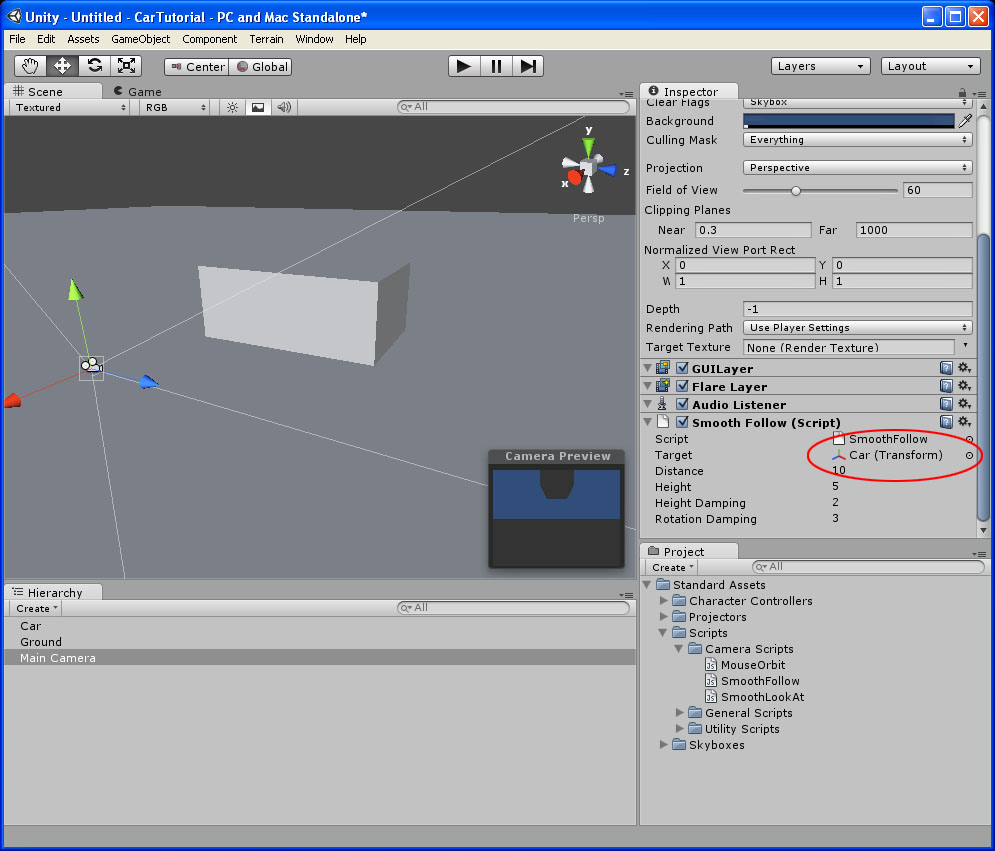
**Chapter 2**

**HARDWARE DESCRIPTION**

****

**Fig1: Block diagram of unity 3D**

* 1. **The Basics:**  
       
     First things first, we are not going to use anything that Unity doesn't have to start with. We will get to making a car and importing it and doing whatever later.  
     We start with a nice NEW scene. Into this scene we are going to add the following packages:  
     Character Controller.unityPackage  
     Projectors.unityPackage  
     Scripts.unityPackage  
     Skyboxes.unityPackage  
       
     NOTE: Not a single one of these it totally nessecary, they will be used later on in the more advanced areas. The only thing I usually import is the Scripts, because it contains the SmoothFollow script.  
       
     

OK with a new scene, lets create a box, name it Ground. set its scale to 500,1,500 and it's position to 0,0,0. Being the ground, this is of course what we are going to be driving on, so were making it nice and big.  
Next, we are going to create a box, and name it Car. Set its scale to 3,1,5 and its position to 0,5,0. This is the building block of the car.  
OK, we have 3 pieces in our scene, Car, Ground and Main Camera.  
From the Standard Assets folder, we are going to the Scripts folder and the Camera Scripts folder in that. We find a SmoothFollow script in there. Drag that onto the Main Camera in the Hierarchy view.  
In the Hierarchy view, click on the Main Camera, and in the Inspector, scroll down to the Smooth Follow (Script). The Target is None (Transform). We need to drag the Car from the Hierarchy view onto the None (Transform) label there.  
  
  
Now, we are going to add a Rigidbody to the mix. So with the Car selected, goto Component/Physics/Rigidbody.  
Scroll down to the Rigidbody in the inspector and change the Mass to 1000;  
The next thing we are going to add, is a Directional Light. (GameObject/CreateOther/Directional Light) Rotate it some in the screen so that not everything looks so bland.  
Now we can play it, and you will see the camera moving some, and a block, it falls, hits the ground, and it's still pretty bland.

OK, lets create some wheels. (GameObject/CreateOther/Sphere)  
Create one, and we want the Scale to be 1,0.2,1. We will need to rotate it, and move it to a location where a wheel should be. The rotation should be 0,0,90 and the position should be about 1.5, 4.5, 1.8. This is all in how you look at it though.  
OK, next, we are going to set this wheel up. First, we need to parent the wheel to the car, and label it... In my case I am going to label the wheel FrontRight. Next, delete the Sphere Collider from the wheel. (Click on the Gear beside the Sphere Collider in the Inpsector and click Remove Component.) Now Add a WheelCollider to the Wheel. (Component/Physics/Wheel Collider)  
OK, now we have a basic wheel collider attached to one wheel. NOTICE how the radius of the wheel collider is 0.5. This is because the Scale of the original scale of the wheel is 1. This will be very important in later tutorials.  
OK, now if we run it, it is going to drop, hit the ground, but will hobble on the one wheel we set up.  
If you are along with me, then we can go to the next step.  
Click on the wheel we just set up, duplicate it (Ctrl-D) and move it to the back. (the measurements have changed, now we are in the local measurements, so it is 0.5, -0.5, -0.366) Rename this one, RearRight  
Select Both wheels, and duplicate them. Then move them over to the other side. (move them over, then change thier X values to -0.5) Rename them repsectively FrontLeft and RearLeft.

OK, now with everything in place, we run it, it falls to the ground, but has no springs, so it sits there.  
So lets build up some suspension on it. This is what is going to make it react like a car.  
For each wheel, we need to set 6 values. The Spring Distance (All wheels should be 0.25), the Suspension Spring.Spring (1500 for the front wheels, 1000 for the rear), the Suspension Spring.Damper (All wheels are 2), the Suspension Spring.Target Position (All wheels should be 0.25), the Forward Friction.Stiffness Factor (All wheels should be 0.02) and the Sideways Friction.Stiffness Factor(All wheels should be 0.02)  
This means, that a 1000 KG car takes 5000 (or five times the mass) units of force to keep it springy

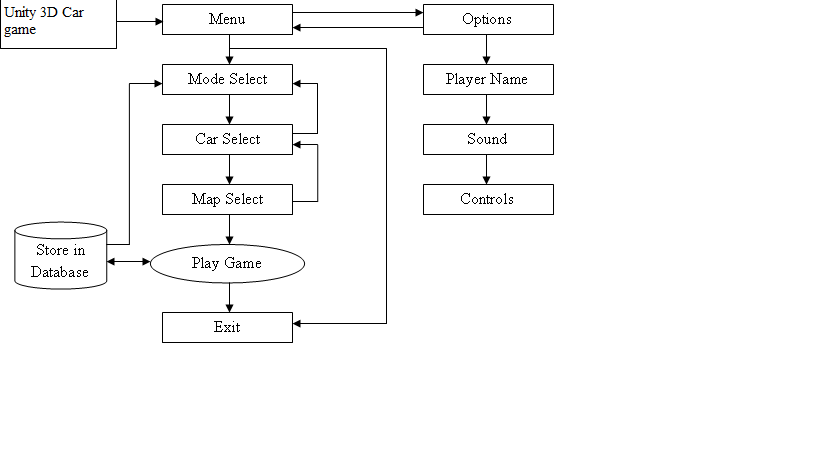
**Chapter 3**

**SOFTWARE DESCRIPTION**

**3.SOFTWARE DESCRIPTION**

**3.1 FLOW DIAGRAM**

:



Now for a simple piece of code to make it all work:

**CSharp:**

*//CarController1.js*

**var** wheels : Transform[];

1. **var enginePower=150.0;**
3. **var** power=0.0;
4. **var** brake=0.0;
5. **var** steer=0.0;
7. **var** maxSteer=25.0;
9. function Start(){
10. rigidbody.centerOfMass=Vector3(0,-0.5,0.3);

**}**

2. function Update () {
3. power=Input.GetAxis("Vertical") \* enginePower \* Time.deltaTime \* 250.0;
4. steer=Input.GetAxis("Horizontal") \* maxSteer;
5. **brake=Input.GetKey("space") ? rigidbody.mass \* 0.1: 0.0;**
7. GetCollider(0).steerAngle=steer;
8. GetCollider(1).steerAngle=steer;
10. **if(brake > 0.0){**
11. GetCollider(0).brakeTorque=brake;
12. GetCollider(1).brakeTorque=brake;
13. GetCollider(2).brakeTorque=brake;
14. GetCollider(3).brakeTorque=brake;
15. **GetCollider(2).motorTorque=0.0;**
16. GetCollider(3).motorTorque=0.0;
17. } **else** {
18. GetCollider(0).brakeTorque=0;
19. GetCollider(1).brakeTorque=0;
20. **GetCollider(2).brakeTorque=0;**
21. GetCollider(3).brakeTorque=0;
22. GetCollider(2).motorTorque=power;
23. GetCollider(3).motorTorque=power;
24. }
25. **}**
27. function GetCollider(n : **int**) : WheelCollider{
28. **return** wheels[n].gameObject.GetComponent(WheelCollider);
29. }

