“ROAD HORIZON” - A COMPUTER GRAPHICS PROJECT

A PROJECT REPORT

SUBMITTED BY :-

ADARSH THAKUR (194CA002)

PARAS NARAYAN GAUTAM (194CA030)

TO:-

MS. DEEPTHI

In partial fulfillment for the award of the degree

MASTER OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL

APRIL 2021

Mathematical and Computational Sciences

National Institute Of Technology Karnataka, Surathkal

DECLARATION

We hereby to declare that the **'Computer Graphics'** project **Report** entitled "**ROAD HORIZON**" which is being submitted to the National Institute of Technology Karnataka, Surathkal, is record of an original work done by us under the guidance of **Ms. DEEPTHI** in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications in the department of Mathematical and Computational Sciences, is a bonafide report of the work prepared by us. This material is collected from various sources with utmost care and is based on facts and truth.

ADARSH THAKUR (194CA002)

PARAS NARAYAN GAUTAM (194CA030)

MCA IV SEM

NITK, SURATHKAL

CERTIFICATE

Certified that this project report **"ROAD HORIZON" - A COMPUTER GRAPHICS PROJECT"** is the bonafide work of **Adarsh Thakur (194CA002)** and **Paras Narayan Gautam (194CA030)** who carried out the project under my supervision. This is to further certify to the best of my knowledge, that this project has not been carried out earlier in this institute and the university.

Signature

MS. DEEPTHI

Certified that the above mentioned project has been duly carried out as per the norms of the college and statutes of the university.

Signature

(Dr. Shyam S. Kamath)

HEAD OF THE DEPARTMENT

ACKNOWLEDGEMENT

I wish to express my profound and sincere gratitude to MS. DEEPTHI Department of Mathematical and Computational Science NITK, Surathkal, who guided me into the intricacies of this project.

I am highly grateful to our team who evinced keen interest and invaluable support in the progress and successful completion of my project work.

I am indebted to our team for their constant encouragement, co-operation and help. Words of gratitude are not enough to describe the accommodation and fortitude which they have shown throughout my endeavor.

MS. DEEPTHI

CONTENT

INTRODUCTION

Basically, here in this project we design the **2D** car racing game by using opengl library as a major.

The objective of this game is to survive as long as possible and get to the High scores while avoiding the obstacles on the tracks.

Opengl is actually stands for :-

OpenGL, short for "Open Graphics Library," is an application programming interface (API) designed for rendering 2D and 3D graphics. It provides a common set of commands that can be used to manage graphics in different applications and on multiple platforms.

DESCRIPTION

The game is created in openGL and Code Blocks.

▹ The User will play the game to achieve highest score.

▹ The user can select the playing mode according to its choice whether he/she wants to play in medium mode or in hard mode.

▹ Although the game automatically increases speed of obstacles(car), every time when the score is greater than 50(i.e. Level up).

▹ On collision with any of the obstacles the scores will stop.

▹ On collision, the user will have two choices.

▹ According to his choice, either the user can restart the game or user can end it.

METHODOLOGY

Major Opengl Standard functions used in this Project :-

So here we use some Opengl functions to functionalise the concept to get practical output.

1. **glortho(left, bottom, near, right, top, near) :-**

   The glOrtho function describes a perspective matrix that produces a parallel projection. The (*left*, *bottom*, *near*) and (*right*, *top*, *near*) parameters specify the points on the near clipping plane that are mapped to the lower-left and upper-right corners of the window, respectively, assuming that the eye is located at (0, 0, 0). The *far* parameter specifies the location of the far clipping plane. Both *zNear* and *zFar* can be either positive or negative.

The corresponding matrix is shown in the following image.

$$\begin{pmatrix} \frac{2}{right-left} & 0 & 0 & t_x \\ 0 & \frac{2}{top-bottom} & 0 & t_y \\ 0 & 0 & \frac{-2}{far-near} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

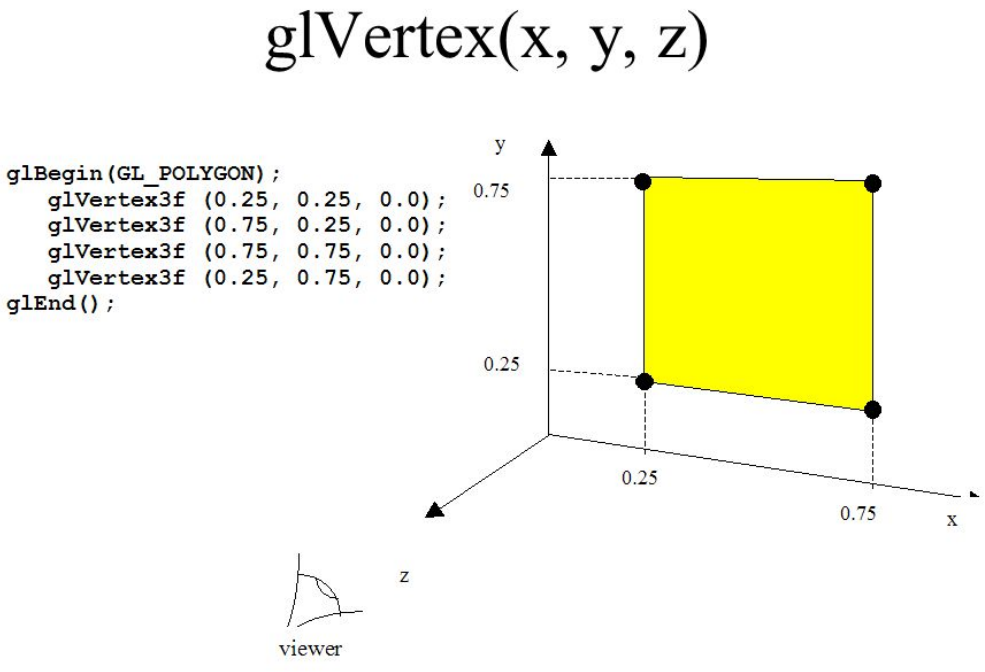
$$t_x = -\frac{right+left}{right-left}$$

$$t_y = -\frac{top+bottom}{top-bottom}$$

$$t_z = -\frac{far+near}{far-near}$$

2. glvertex() :-

glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called.

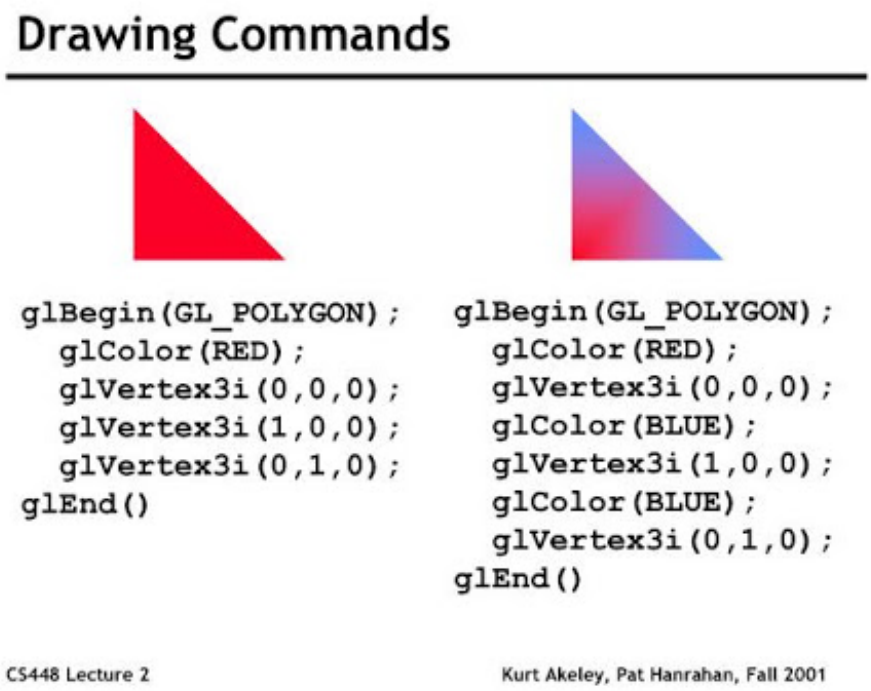


3. glbegin() :-

glBegin delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies which of ten ways the vertices are interpreted. Taking *n* as an integer count starting at one, and *N* as the total number of vertices specified.

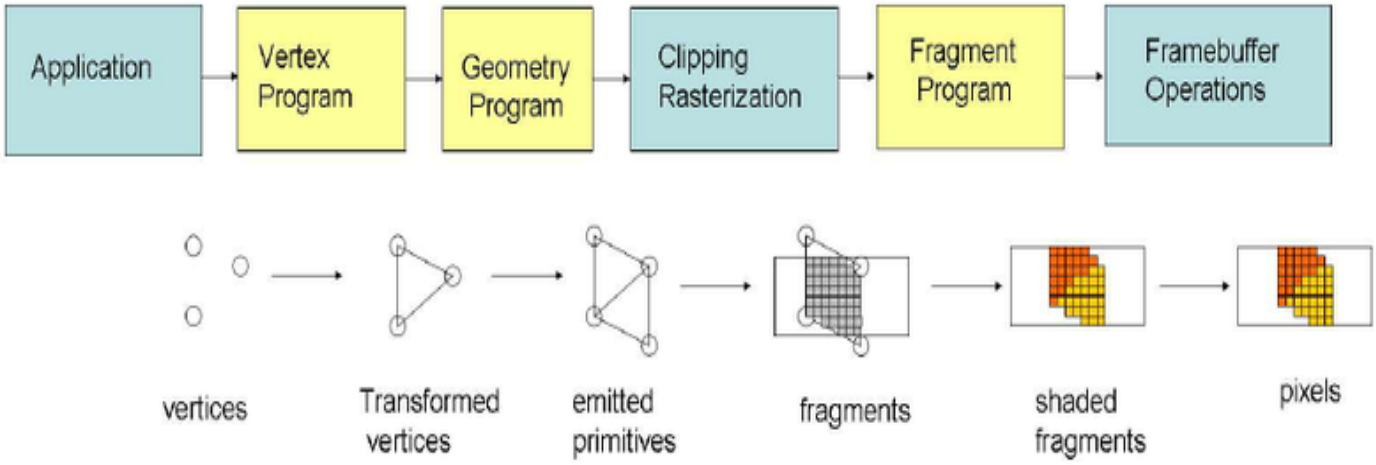
4. glcolor() :-

glColor() is used for to sets a new four-valued RGBA color.



The OpenGL Visualization Programming Pipeline :-

Rendering Pipeline is the sequence of steps that OpenGL takes when rendering objects. Vertex attribute and other data go through a sequence of steps to generate the final image on the screen.



Reasons to choose the Opengl :-

- **Industry standard**

  An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.

- **Stable**

  OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers toadopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

- **Reliable and portable**

  All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

- **Evolving**

  Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

- **Scalable**

  OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

- **Easy to use**

  OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features.

- **Well-documented**

  Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

The major functions used to curve this project

Details of three important functions:-

1. **glutDisplayFunc(display)**:- In this function our first screen and start game window are call accordingly when user wants to play the game.

   glutDisplayFunc sets the display callback for the *current window*. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the *current window* is set to the window needing to be redisplayed.

2. **glutSpecialFunc(spe_key)**:- In this function we have set Top, Bottom, Up,Down keys tasks.

   glutSpecialFunc sets the special keyboard callback for the *current window*. The special keyboard callback is triggered when keyboard function or directional keys are pressed. The key callback parameter is a GLUT_KEY_* constant for the special key pressed. The x and y callback parameters indicate the mouse in window relative coordinates when the key was pressed.

3. **glutKeyboardFunc(processKeys)**:- In process key we are again set all the variable values, so the game could run as the number of times user wants.

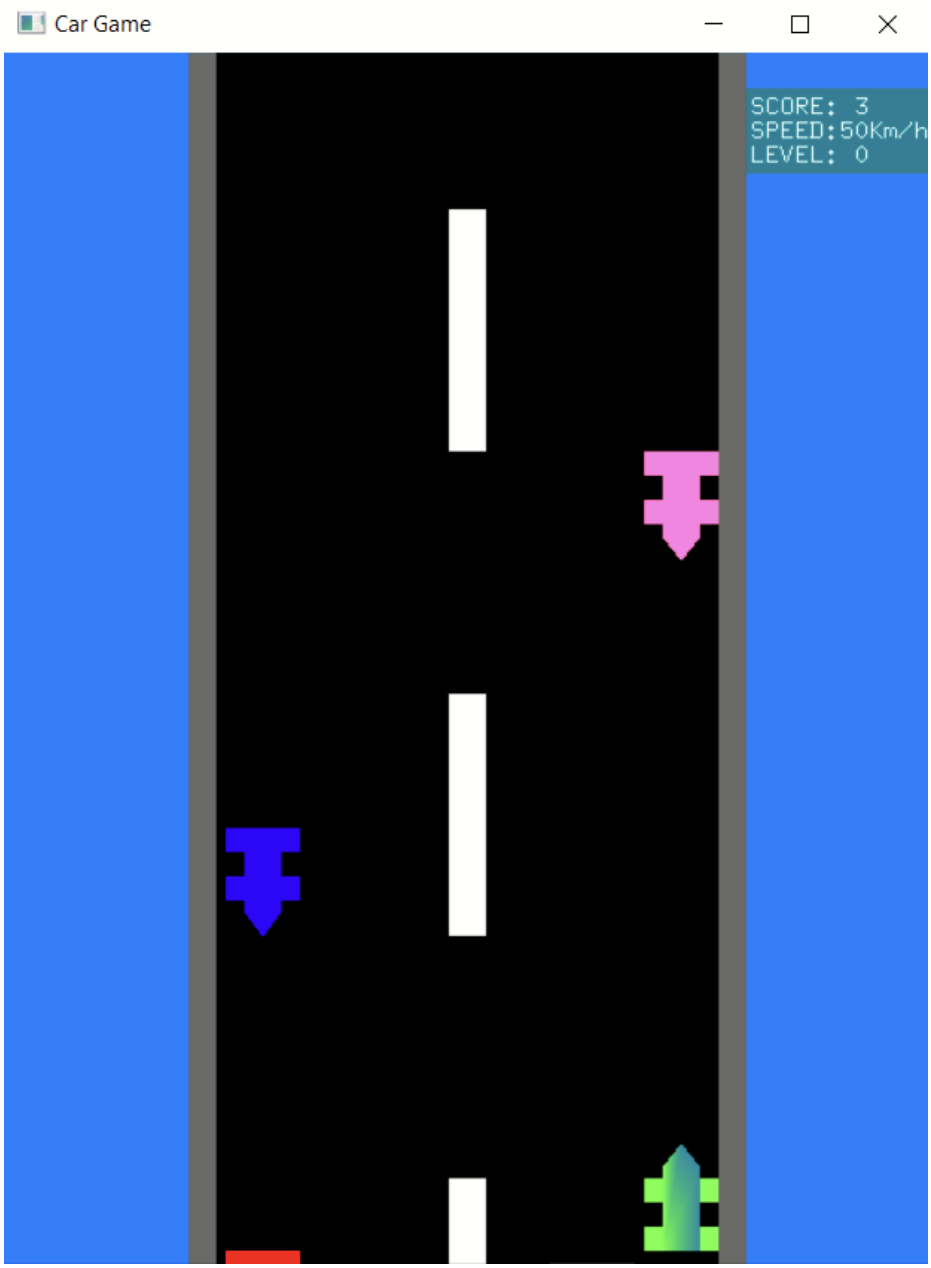
   glutKeyboardFunc sets the keyboard callback for the *current window*. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored.

WORKING :-

STEP 1: Run the game then the user will redirect to front page of the game in which user can read the instruction to play.

To start the game user have to press the spacebar and to Exit the game user have to press ESC button.

STEP 2 :- After pressing the Spacebar user will redirect to the Start game window, here user can shift its Green car position with the left , right arrow buttons and increase or decrease the speed of obstacles with up and down arrow buttons of the keyboard.



STEP 3 :- On collision with any opposite coming car the game will over and user will redirect to the front page screen of the game with displaying your previous game score with giving the option Press Space to Start like a Play Again option.

FEATURES

1. **SPEED CONTROL :-** User can select the speed of obstacles, whether user wants to play the game at high speed or low speed.
2. **OPPOSITE CARS :-** The obstacles are not coming randomly, they are coming with respect to the position of user car this make our game much harder.
3. **COLLISION :-** To make a racing game playable, collisions are required. Collisions govern the way a player interacts with the surrounding environment. It serves as the basis for resting motion, it allows a player to drive very smartly to avoid the collision from those cars who are coming from the front. and it enables the player indulge in the game continuously.

CONCLUSION

We gained a substantial amount of understanding of how the pipeline in OpenGL works and of the various data structures that can be used in OpenGL. We acquired a much better understanding of how compilers work and learned how to use cmake to make compilation much quicker and more portable. We do certain practiced on opengl standard functions, and how it actually works and gained the experience in setting up a flow of data that will be more lasting.

REFERENCES

1. https://en.wikipedia.org/wiki/OpenGL
2. https://www.opengl.org/about/
3. https://www.opengl.org/resources/libraries/glut/spec3/node49.html
4. https://www.opengl.org/resources/libraries/glut/spec3/node54.html
5. https://www.opengl.org/resources/libraries/glut/spec3/node46.html