VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA” ,BELAGAVI-590018

A Mini Project Report On

“3D Car Animation”

Submitted in the partial fulfillment of the requirement for the award of degree of

VI Semester degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

For the Academic
Year 2021-22

Submitted by

Mr. Bavage Anket Ashokrao	**1AR19CS008**
Mr. T Jaya Krishna	**1AR19CS058**

Under the Guidance of

Dr. Rajshekhar Patil
Professor,
Dept. of CSE

AIEMS

BENGALURU

B.V.V. Sangha’s

AMRUTA INSTITUTE OF ENGINEERING & MANAGEMENT SCIENCES

BIDAI INDUSTRIAL AREA BENGALURU – 562109

BVV Sangha, Bagalkot

AMRUTA INSTITUTE OF ENGINEERING & MANAGEMENT SCIENCES

Approved by AICTE, New Delhi

Recognized by Government of Karnataka & Affiliated to VTU, Belagavi

AIEMS BENGALURU

CERTIFICATE

This is to certify that the mini project report entitled **"3D Car Animation"** is a bonafide work carried out by **Mr. Bavage Anket Ashokrao [1AR19CS008]** & **Mr. T Jaya Krishna [1AR19CS058]** in partial fulfilment of award of **Bachelor of Engineering in Computer Science & Engineering** of the Visvesvaraya Technological University, Belagavi, during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The sixth semester mini project has been approved as it satisfies the academic requirements in respect to Computer Graphics Lab (18CSL67) associated with the degree mentioned.

....	....
Signature of Guide	Signature of HOD
Dr. Rajshekhar Patil	**Dr. M S Patel**
Prof., Dept. of CSE	Head, Dept. of CSE,
AIeMS , Bangalore.	AIeMS , Bangalore.

External

Examiner Name	**Signature**
1.____________________	____________________
2.____________________	____________________

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my effort with success.

We are grateful to our institution, **B.V.V Sangha's Amruta Institute of Engineering & Management Sciences (AIEMS),** with its ideals and inspirations for having provided us with the facilities, which has made this, project a success.

We earnestly thank **Dr. Santosh M Muranal, Principal, AIEMS,** for facilitating academic excellence in the college and providing us with the congenial environment to work in, that helped us in completing this project.

We wish to extend our profound thanks to **Dr. M S Patel, Professor & Head, Department of Computer Science & Engineering, AIEMS,** for giving us the consent to carry out this project.

We would like to express our sincere thanks to our guide **Dr. Rajshekhar Patil, Professor, Department of Computer Science & Engineering, AIEMS,** for his immense help during the project and also for his valuable suggestions on the project report preparations, which helped us in the successful completion of the project.

We would like to thank all the faculties of **Computer Science & Engineering Department**, for their valuable advice and support.

We would like to express our sincere thanks to our parents and friends for their support.

Mr. Bavage Anket Ashokrao **[1AR19CS008]**

Mr. T Jaya Krishna **[1AR19CS058]**

DECLARATION

We, **Mr.Bavage Anket Ashokrao [1AR19CS008]** and **Mr.T Jaya Krishna [1AR19CS058]** students of VI semester B.E, Department of Computer Science & Engineering, AMRUTA INSTITUTE OF ENGINEERING AND MANAGEMENT SCIENCES, Bengaluru, declare
that mini project work entitled **"3D Car Animation"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2021-2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Place:

Date:

Mr. Bavage Anket Ashokrao	**Mr. T Jaya Krishna**
[1AR19CS008]	**[1AR19CS058]**

ABSTRACT

3D computer graphics or three-dimensional computer graphics (in contrast with 2D computer graphics) are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. Such images may be stored for viewing later or displayed in real time. 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire frame model and 2D computer raster graphics in the final rendered display. In computer graphics software, 2D applications may use 3D techniques to achieve effects such as lighting, and 3D may use 2D rendering techniques.

3D computer graphics are often referred to as 3D models. Apart from the rendered graphic, the model is contained within the graphical data file. However, there are differences: a 3D model is mathematical representation of any three-dimensional object. A model is not technically a graphic until it is displayed. A model can be displayed visually as a two-dimensional image through a process called 3D rendering or used in non-graphical computer simulations and calculations. With 3D printing, 3D models are similarly rendered into a 3D physical representation of the model, with limitations to how accurate the rendering can match the virtual model

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SNAPSHOTS

CHAPTER 1
INTRODUCTION

1.1 Computer Graphics

In this era of computing, computer graphics has become one of the most powerful and interesting fact of computing. It all started with display of data on hardcopy and CRT screen. Now computer graphics is about creation, retrieval, manipulation of models and images.

Graphics today is used in many different areas. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

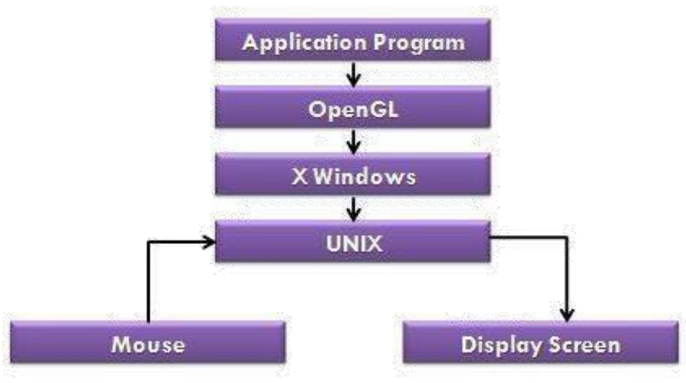


Fig:1.1 Overview Of Computer Graphics

1.2 OpenGL Concept

1.2.1 Interface

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Functions in the main GL library have names that begins with *gl* and are stored in a library usually referred to as GL. The second is the **OpenGL Utility Library(GLU)**. The library uses only GL functions but contains code for creating common objects and simplifying viewing.

All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with the letter *glu*. Rather than using a different library for each system we use a readily available library called OpenGL Utility Toolkit(GLUT),which provides the minimum functionality that should be expected in any modern windowing system.

1.2.2 Overview

- OpenGL(Open Graphics Library) is the interface between a graphics program and graphics hardware. *It is streamlined*. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.

- It is a state machine. At any moment during the execution of a program there is a current model transformation.
- It is a rendering pipeline. The rendering pipeline consists of the following steps:

i. Defines objects mathematically.

ii. Arranges objects in space relative to a viewpoint.

iii. Calculates the color of the objects.

1.2.3 Opengl Architecture:

1. Pipeline Architectures:

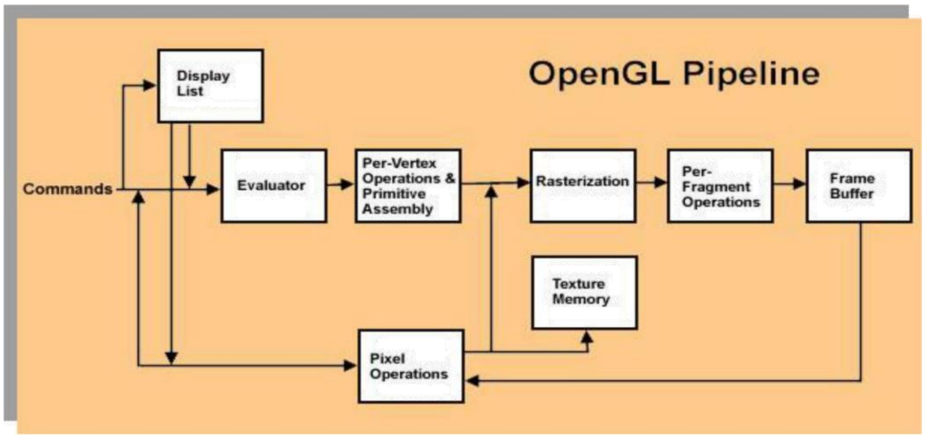


Fig:1.2.3.1 Opengl Pipeline Architecture

2. OpenGL Engine and Drivers:

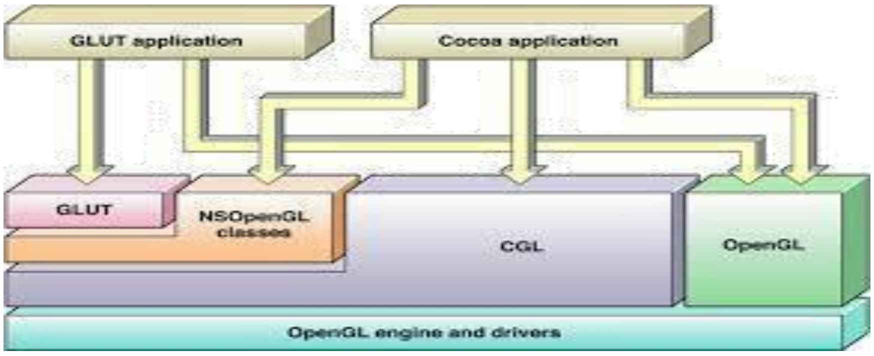


Fig:1.2.3.2 Opengl Engine And Drivers

3. Application development-API's

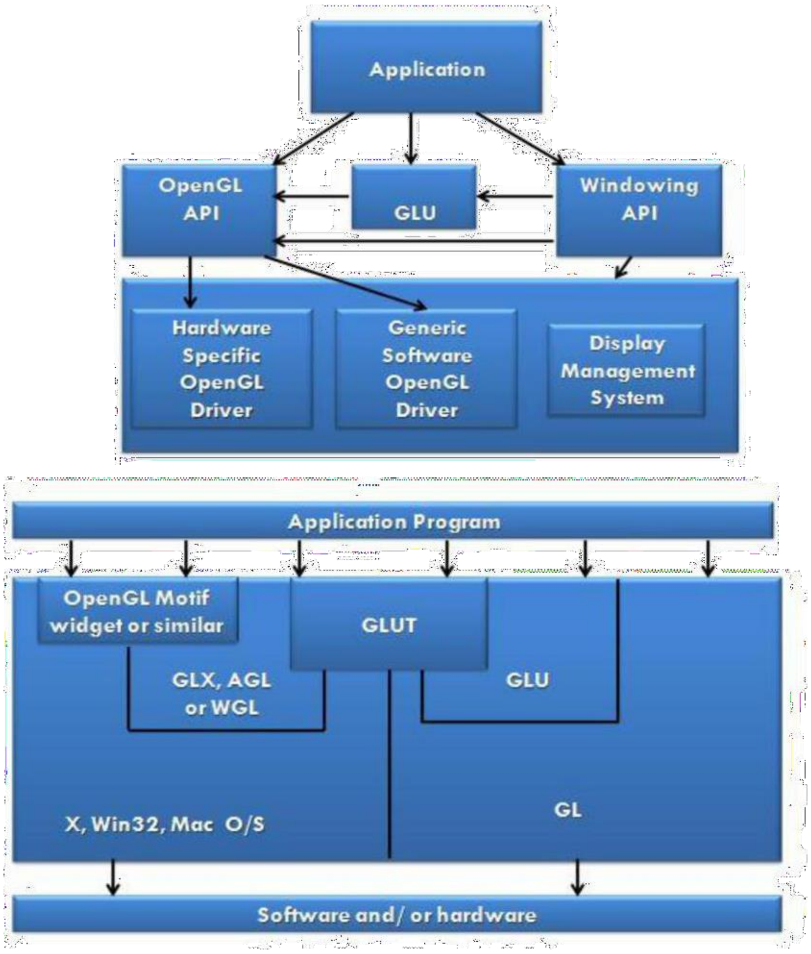

Application
OpenGL API
GLU
Windowing API
Hardware Specific OpenGL Driver
Generic Software OpenGL Driver
Display Management System
Application Program
OpenGL Motif widget or similar
GLUT
GLX, AGL or WGL
GLU
X, Win32, Mac O/S
GL
Software and/ or hardware

Fig:1.2.3.3 Applications Development(API's)

The above diagram illustrates the relationship of the various libraries and window system Components.

Generally, applications which require more user interface support will use a library designed to support those types of features (i.e. Buttons, menu and scroll bars, etc.) Such as motif or the win32 API.

Prototype applications, or ones which do not require all the bells and whistles of a full gui, may choose to use glut instead because of its simplified programming model and window system independence.

Display lists:

All data, whether it describes geometry or pixels, can be saved in A *display list* for current or later use. (1 alternative to retaining data in A displaylist is processing the data immediately - also known as *immediate mode*.) When A display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

Evaluators:

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions. Evaluators provide A method to derive the vertices used to represent the surface from the control points. The method is A polynomial mapping, which can produce surface normal, texture coordinates, colors, and spatial coordinate values from the control points.

Per-vertex operations

For vertex data, next is the "per-vertex operations" stage, which converts. The vertices into primitives. Some vertex data (for example, spatial coordinates) are transformed by 4 X 4 floating-point matrices. Spatial coordinates are projected from A position in the 3D world to A position on your screen.

If advanced features are enabled, this stage is even busier. If texturing is used, texture coordinates may be generated and transformed here. If lighting is enabled, the lighting calculations are performed using the transformed vertex, surface normal, light source position, material properties, and other lighting information to produce A color value.

Primitive assembly

Clipping, A major part of primitive assembly, is the elimination of portions of geometry which fall outside A half-space, defined by A plane. Point clipping simply passes or rejects vertices; line or polygon clipping can add additional vertices depending upon how the line or polygon is clipped.

In some cases, this is followed by perspective division, which makes distant geometric objects appear smaller than closer objects. Then view port and depth (Z coordinate) operations are applied. If culling is enabled and the primitive is A polygon, it then may be rejected by A culling test. Depending upon the polygon mode, A polygon may be drawn as points or lines. The results or this stage are complete geometric primitives, which are the transformed

Hardware interface:

The standard output device, as mentioned earlier has been assumed to be a color monitor. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The mouse, the main input device, has to be functional. A keyboard is also required. .

Apart from these hardware requirements, there should be sufficient hard disk space and primary memory available for proper working of the package.

Software interface:

The editor has been implemented on the dos platform and mainly requires an appropriate version of the compiler to be installed and functional. Though it has been implemented on dos, it is pretty much platform independent

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 Hardware requirements:

- CPU: Intel/AMD CPU
- RAM (Main memory): 512 MB
- Hard disk: 10MB of free space
- Hard disk speed (in RPM): 5400 RPM
- Hard disk speed (in RPM): 5400 RPM
- Keyboard: Standard keyboard with arrow keys
- Keyboard: Standard keyboard with arrow keys

2.2 Software requirements:

- Operating System : Windows 10/11
- Compiler, ide : Ubuntu
- Mouse driver
- Graphic driver

CHAPTER 3

SYSTEM DESIGN

In this project we are presenting 3d car animation, which is one the best opengl car program. This 3D opengl program have so many advance features in it. We can see below the all features that included in this 3D animated program.

Features of the 3d car animation opengl program:

- Start screen: as you execute the program, first thing come is the start screen where you will see the details of the project. The start screen has name of college and the name of projects. It also has the user interaction instructions about the use of mouse and keyboard functionality. There is option of going to main screen, by pressing 'space' key.
- Main screen: after you get in to main screen from the start screen you will see the 3D car drawn in blue color. If you press right mouse button you will get the menu options to select for particular functionality to execute. Details is mentioned in below user interaction section.
- Comments: most of the codes in this 3D opengl program, is commented hence easy to understand the whole program.

❖ Have user interaction both using keyboard as well as mouse. Below is the description of the uses and functionality of keys and menus in this 3D opengl program.

1. Mouse interaction: press right mouse button to get the mouse. Following is the menus-

- Car model mode - this is default mode of car display which will display the only car.
- Car driving mode - this will display the driving mode, which includes the long road and green filed.
- Wheel effect - this is one of the finest effects, it will animate the car while it moved as it Is moving in the car.
- Toggle light - this will apply the light effect on/off when selected.
- Car colors - this menu has submenu which allow to select the color of car. The submenu has the following options - blue, red, green, black, yellow and grey.
- Day mode - by default we have day mode on, in this project so while in night mode you can select this to toggle to daylight mode.
- Night mode - this menu will let you switching to the night mode, by showing darkness around.

2. Keyboard interaction: below is complete description against each key and what they do when pressed.

- X- rotate the car in 'x' direction
- Y- rotate the car in 'y' direction
- Z- rotate the car in 'z' direction
- A- increase the size of car in 'x' direction
- S- increase the size of car in 'y' direction
- Q- increase the size of car in 'z' direction
- U- camera top view
- F- camera side view
- Left arrow key - move car in forward direction
- Right arrow key- move car in backward direction
- Spacebar - enter the main screen from start screen / esc - exit from the program.

CHAPTER 4
IMPLEMENTATION SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <string.h>
#define ESCAPE 27
GLint  window;
GLint window2;
GLint Xsize=1000;
GLint Ysize=800;
float i,theta;
GLint nml=0,day=1;
char name3[]="PROJECT:  3D CAR  ANIMATION";

GLfloat xt=0.0,yt=0.0,zt=0.0,xw=0.0;
GLfloat xs=1.0,ys=1.0,zs=1.0;
GLfloat xangle=0.0,yangle=0.0,zangle=0.0,angle=0.0;

GLfloat r=0,g=0,b=1;
GLint light=1;
int count=1,flg=1;
int view=0;
int flag1=0,aflag=1;            //to switch car driving mode
int flag2=0,wheelflag=0; //to switch fog effect
GLUquadricObj *t;
```

```
static void SpecialKeyFunc( int Key, int x, int y );


GLvoid Transform(GLfloat Width, GLfloat Height)
{
  glViewport(0, 0, Width, Height);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(45.0,Width/Height,0.1,100.0);
 glMatrixMode(GL_MODELVIEW);
}
GLvoid InitGL(GLfloat Width, GLfloat Height)
{


  glClearColor(1.0, 1.0, 1.0, 1.0);
  glLineWidth(2.0);              /* Add line width, ditto */
 Transform( Width, Height ); /* Perform the transformation */
 t=gluNewQuadric();
        gluQuadricDrawStyle(t, GLU_FILL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);


GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
GLfloat diffuseLight[] = { 0.8f, 0.8f, 0.8, 1.0f };
GLfloat specularLight[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat position[] = { 1.5f, 1.0f, 4.0f, 1.0f };


glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
glLightfv(GL_LIGHT0, GL_POSITION, position);
}
void init()
{
   glClearColor(0,0,0,0);
         glPointSize(5.0);
         glMatrixMode(GL_PROJECTION);
         glLoadIdentity();
         glOrtho(0.0,900.0,0.0,600.0,50.0,-50.0);
         glutPostRedisplay();
}
void display_string(int x, int y, char *string, int font)
{
   int len,i;
         glColor3f(0.8,0.52,1.0);
         glRasterPos2f(x, y);
   len = (int) strlen(string);
   for (i = 0; i < len; i++) {
if(font==1)
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
         if(font==2)
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,string[i]);
         if(font==3)
               glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,string[i]);
if(font==4)
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10,string[i]);
         } }
```

```
void display1(void)
{
  glClearColor(1.0 ,1.0 ,1.0,1.0);
  glClear(GL_COLOR_BUFFER_BIT);
        display_string(190,540,"Amruta institute of engineering & management sciences",1);
        display_string(225,500,name3,1);
        display_string(390+10,470,"HELP",2);
        display_string(10,450,"MOUSE",2);
        display_string(10,410,"PRESS RIGHT BUTTON FOR MENU",3);
        display_string(10,370,"KEYBOARD",2);
        display_string(10,340,"X-Y-Z KEYS FOR CORRESPONDING ROTATION",3);
        display_string(10,280+30,"U-F FOR CAMERA VIEW SETTINGS",3);
        display_string(10,250+30,"USE LEFT ARROW(<-) AND RIGHT ARROW(->) TO
MOVE CAR",3);
        display_string(10,220+30,"ESCAPE TO EXIT",3);
        display_string(250,150+30,"PRESS SPACE BAR TO ENTER",2);
        glutPostRedisplay();
        glutSwapBuffers();
 }

GLvoid DrawGLScene()
{
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
if(view==0)
{
init();
display1();
```

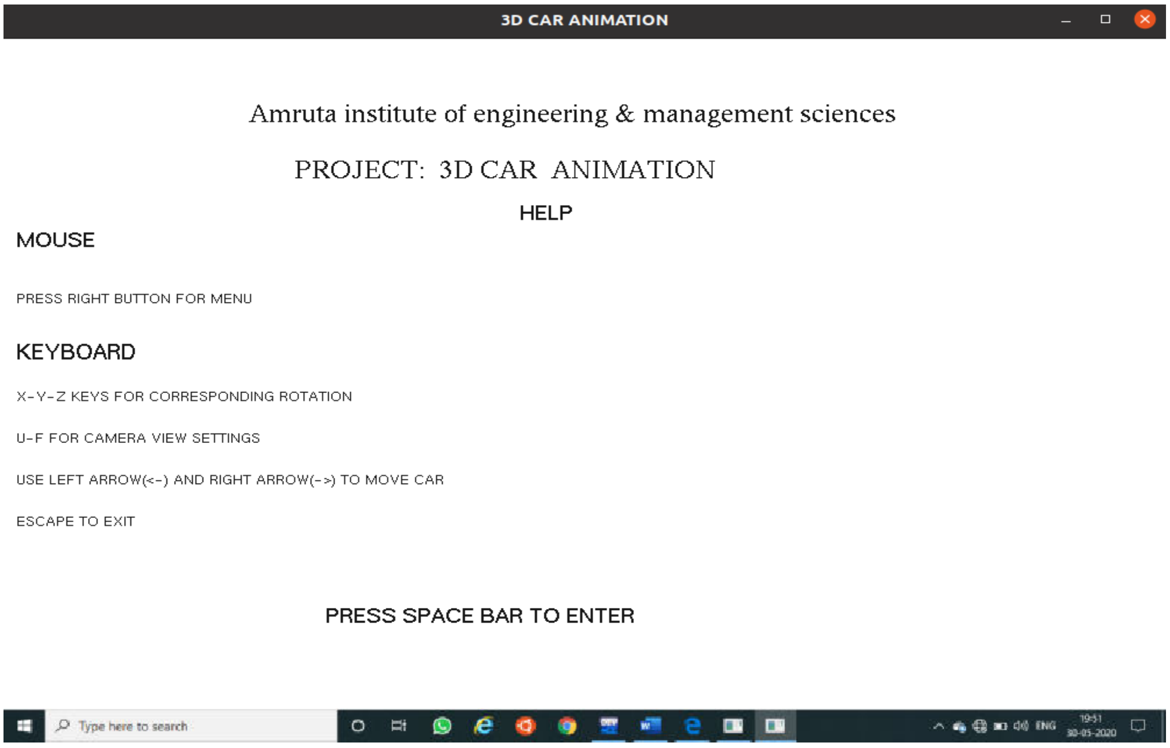
```
}
else
{
  if(count==1)
InitGL(Xsize,Ysize);
  if(aflag==1)/* Initialize our window. */
  glClearColor(1,1,1,1);
  else
              glClearColor(0.1,0.1,0.1,0);
  glPushMatrix();
  glLoadIdentity();
  glTranslatef(-1.0,0.0,-3.5);
  glRotatef(xangle,1.0,0.0,0.0);
  glRotatef(yangle,0.0,1.0,0.0);
  glRotatef(zangle,0.0,0.0,1.0);
  glTranslatef(xt,yt,zt);
  glScalef(xs,ys,zs);
  glEnable(GL_COLOR_MATERIAL);
 if(flag2==1)
  {
  GLfloat fogcolour[4]={1.0,1.0,1.0,1.0};
glFogfv(GL_FOG_COLOR,fogcolour);
glFogf(GL_FOG_DENSITY,0.1);
glFogi(GL_FOG_MODE,GL_EXP);
    glFogf(GL_FOG_START,3.0);
  glFogf(GL_FOG_END,100.0);
  glHint(GL_FOG_HINT, GL_FASTEST);
  glEnable(GL_FOG);
  }
 if(flag2==0)
  {
             glDisable(GL_FOG);
  }
 if(!aflag){
  glBegin(GL_POINTS);
  glColor3f(1,1,1);
  glPointSize(200.0);
  int ccount=0;
  float x=10,y=10;
  while(ccount<20)
  {    glVertex2f(x,y);
             x+=10;
             y+=10;

 if(y>Ysize) y-=10;
             if(x>Xsize) x-=10;
             ccount++;
  }
  glEnd();}
 glColor3f(1.0,.75,0.0);
 glPointSize(30.0);
 glBegin(GL_POINTS);
 glVertex3f(0.2,0.3,0.3);
  glVertex3f(0.2,0.3,0.5);
  glEnd();
  glPointSize(200.0);
glBegin(GL_QUADS);                /* OBJECT MODULE*/
```

```
/* top of cube*/

 //************************FRONT BODY****************************************

 glColor3f(r,g,b);

 glVertex3f( 0.2, 0.4,0.6);

 glVertex3f(0.6, 0.5,0.6);

 glVertex3f(0.6, 0.5,0.2);

 glVertex3f( 0.2,0.4,0.2);

 /* bottom of cube*/

 glVertex3f( 0.2,0.4,0.6);

 glVertex3f(0.6,0.2,0.6);

 glVertex3f(0.6,0.2,0.2);

 glVertex3f( 0.2,0.2,0.2);

 /* front of cube*/

 glVertex3f( 0.2,0.2,0.6);

 glVertex3f(0.2, 0.4,0.6);

 glVertex3f(0.2,0.4,0.2);

 glVertex3f( 0.2,0.2,0.2);

   /* back of cube.*/

 glVertex3f(0.6,0.2,0.6);

 glVertex3f(0.6,0.5,0.6);
 glVertex3f(0.6,0.5,0.2);

  glVertex3f( 0.6,0.2,0.2);

/* left of cube*/

  glVertex3f(0.2,0.2,0.6);

  glVertex3f(0.6,0.2,0.6);

  glVertex3f(0.6,0.5,0.6);

  glVertex3f(0.2,0.4,0.6);

  /* Right of cube */

  glVertex3f(0.2,0.2,0.2);

  glVertex3f( 0.6,0.2,0.2);

  glVertex3f( 0.6,0.5,0.2);

  glVertex3f( 0.2,0.4,0.2);

//****************************************************************************

  glVertex3f(0.7,0.65,0.6);

  glVertex3f(0.7,0.65,0.2);

  glVertex3f(1.7,0.65,0.2);        //top cover

  glVertex3f(1.7,0.65,0.6);

//***************************back guard******************************

  glColor3f(r,g,b);            /* Set The Color To Blue*/

  glVertex3f( 1.8, 0.5,0.6);

  glVertex3f(1.8, 0.5,0.2);
  glVertex3f(2.1, 0.4, 0.2);

  glVertex3f(2.1,0.4,0.6);

 /* bottom of cube*/

  glVertex3f( 2.1,0.2,0.6);

  glVertex3f(2.1,0.2,0.2);

  glVertex3f(1.8,0.2,0.6);

  glVertex3f( 1.8,0.2,0.6);

  /* back of cube.*/

  glVertex3f(2.1,0.4,0.6);

  glVertex3f(2.1,0.4,0.2);

  glVertex3f(2.1,0.2,0.2);

  glVertex3f(2.1,0.2,0.6);

   /* left of cube*/

  glVertex3f(1.8,0.2,0.2);

  glVertex3f(1.8,0.5,0.2);

  glVertex3f(2.1,0.4,0.2);

  glVertex3f(2.1,0.2,0.2);

  /* Right of cube */

  glVertex3f(1.8,0.2,0.6);

  glVertex3f(1.8,0.5,0.6);

  glVertex3f(2.1,0.4,0.6);
```

```
  glVertex3f(2.1,0.2,0.6);

//*****************MIDDLE BODY************************************

  glVertex3f( 0.6, 0.5,0.6);

  glVertex3f(0.6, 0.2,0.6);

  glVertex3f(1.8, 0.2, 0.6);

  glVertex3f(1.8,0.5,0.6);

  /* bottom of cube*/

  glVertex3f( 0.6,0.2,0.6);

  glVertex3f(0.6,0.2,0.2);

  glVertex3f(1.8,0.2,0.2);

  glVertex3f( 1.8,0.2,0.6);

  /* back of cube.*/

  glVertex3f(0.6,0.5,0.2);

  glVertex3f(0.6,0.2,0.2);

  glVertex3f(1.8,0.2,0.2);

  glVertex3f(1.8,0.5,0.2);

  //********************ENTER WINDOW**********************************

  glColor3f(0.3,0.3,0.3);

  glVertex3f( 0.77, 0.63,0.2);

  glVertex3f(0.75, 0.5,0.2);        //quad front window

  glVertex3f(1.2, 0.5, 0.2);
```

```
glVertex3f( 1.22,0.63,0.2);

glVertex3f(1.27,0.63,.2);

glVertex3f(1.25,0.5,0.2);       //quad back window

glVertex3f(1.65,0.5,0.2);

glVertex3f(1.67,0.63,0.2);

glColor3f(r,g,b);

glVertex3f(0.7,0.65,0.2);

glVertex3f(0.7,0.5,.2);      //first separation

glVertex3f(0.75,0.5,0.2);

glVertex3f(0.77,0.65,0.2);



glVertex3f(1.2,0.65,0.2);

glVertex3f(1.2,0.5,.2);      //second separation

glVertex3f(1.25,0.5,0.2);

glVertex3f(1.27,0.65,0.2);



glVertex3f(1.65,0.65,0.2);

glVertex3f(1.65,0.5,.2);     //3d separation

glVertex3f(1.7,0.5,0.2);

glVertex3f(1.7,0.65,0.2);
```

CHAPTER 5
RESULTS


3D CAR ANIMATION

Amruta institute of engineering & management sciences

PROJECT: 3D CAR ANIMATION

HELP

MOUSE

PRESS RIGHT BUTTON FOR MENU

KEYBOARD

X-Y-Z KEYS FOR CORRESPONDING ROTATION

U-F FOR CAMERA VIEW SETTINGS

USE LEFT ARROW(<-) AND RIGHT ARROW(->) TO MOVE CAR

ESCAPE TO EXIT

PRESS SPACE BAR TO ENTER

Fig 5.1 Starting menu of the project

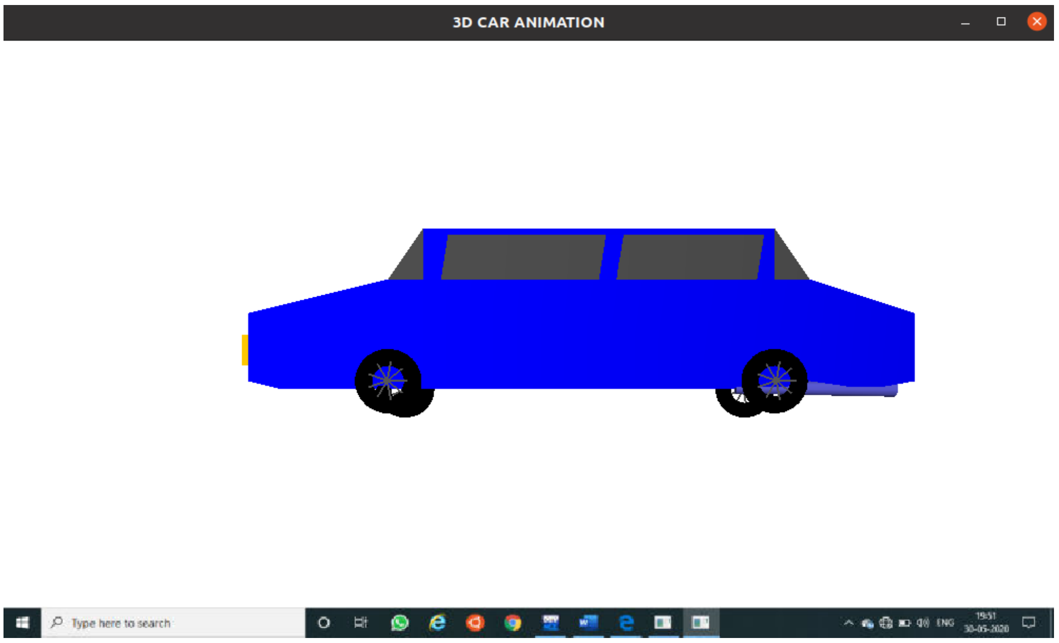


Fig 5.2 Inside the Project

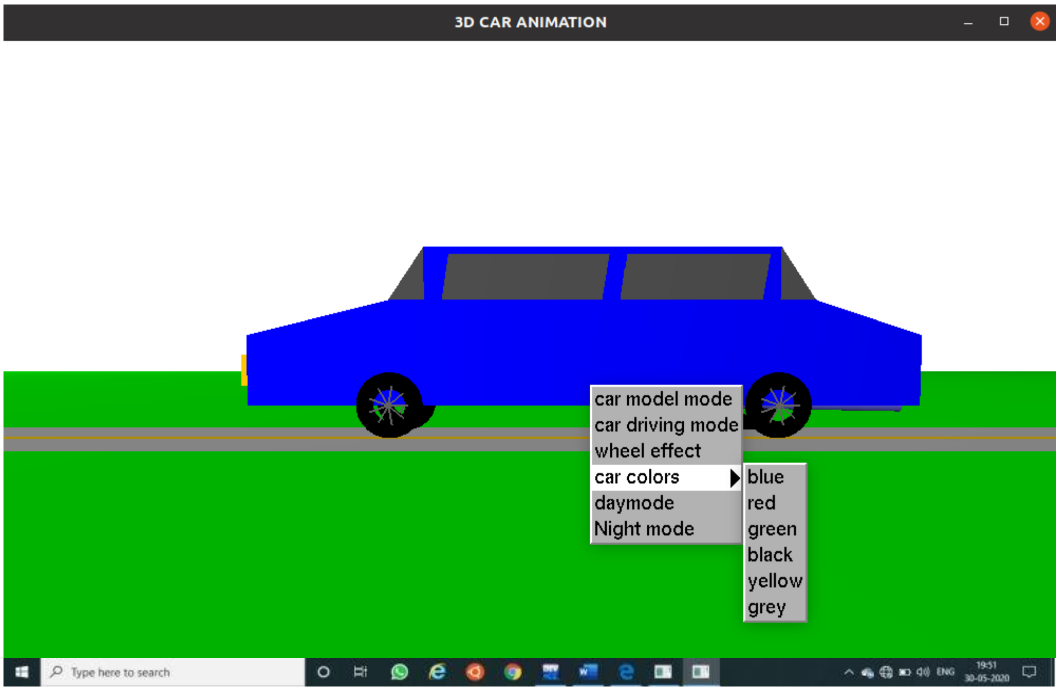


Fig 5.3 Different Car Options

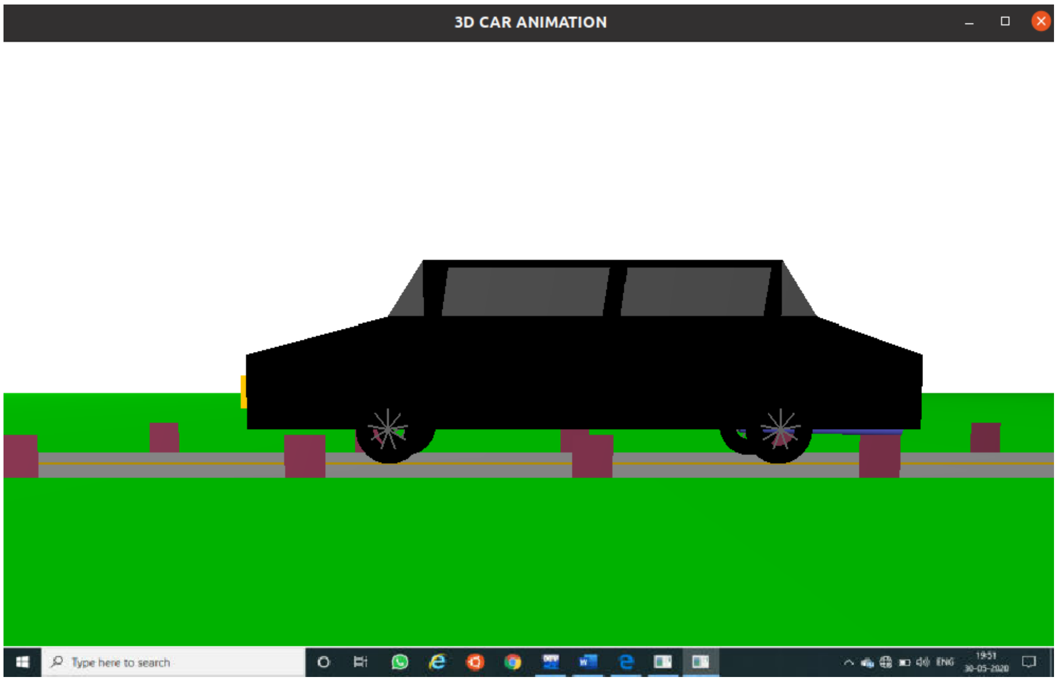


Fig 5.4 Wheel Effect

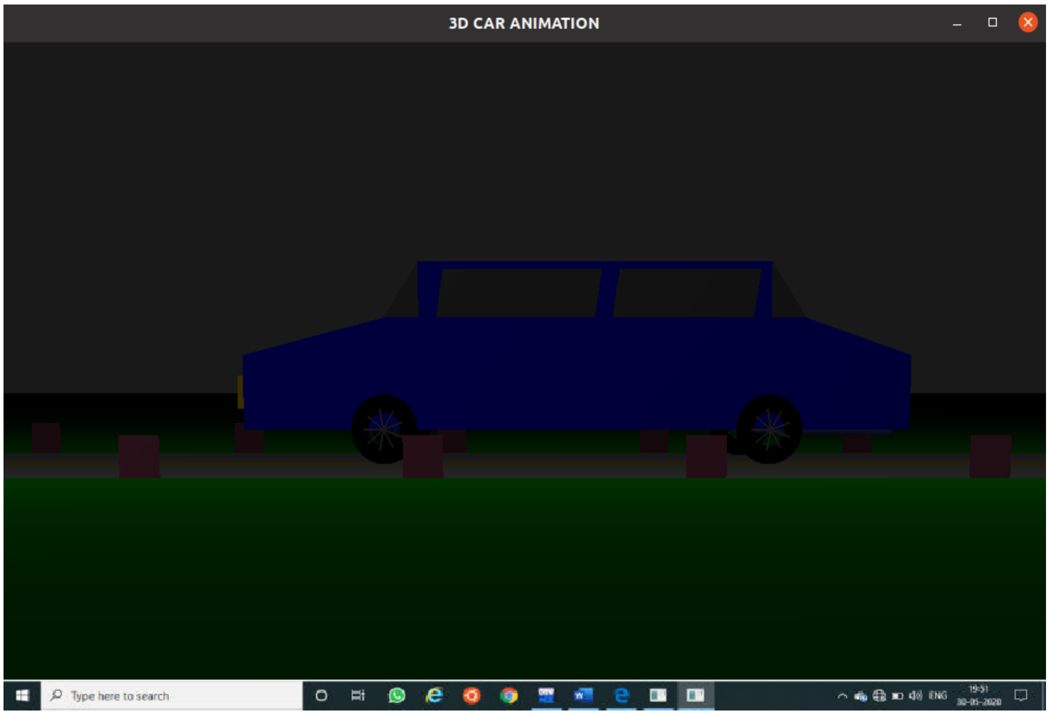


Fig 5.5 Night Mode

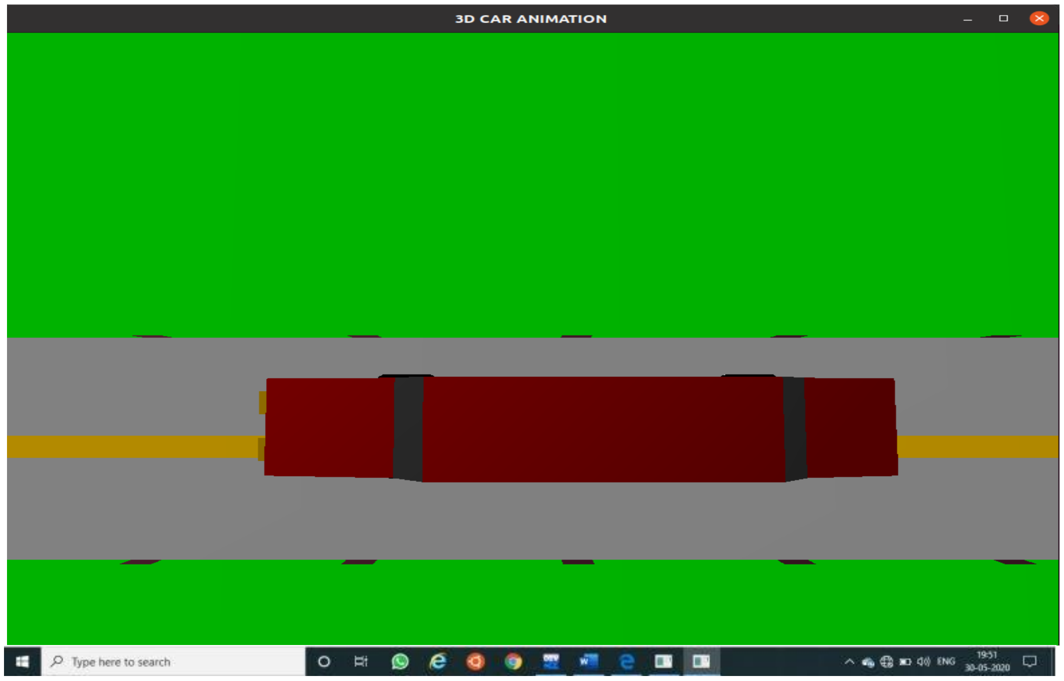


Fig 5.6 Car Top view

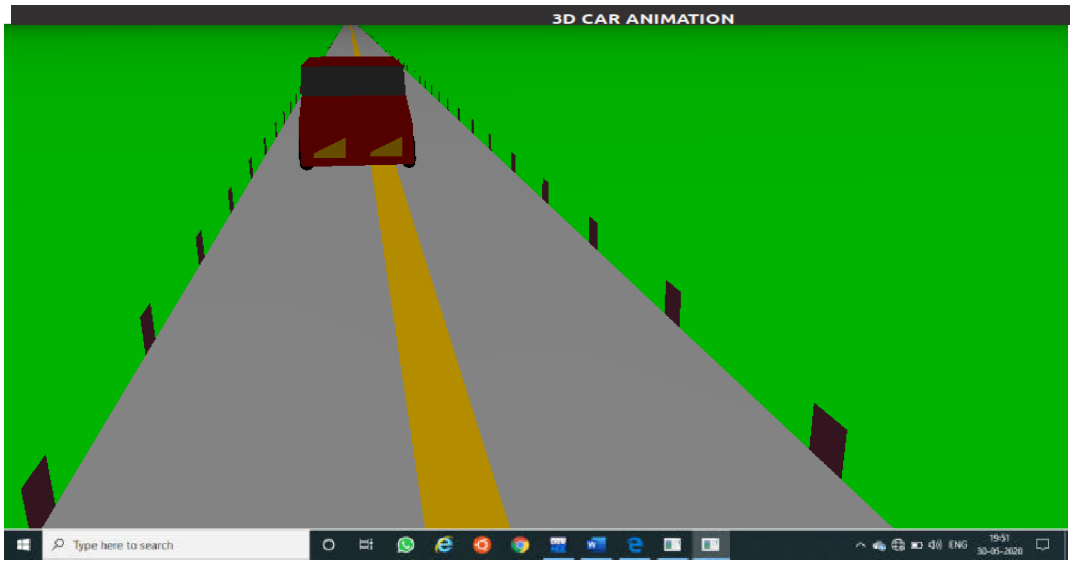


Fig 5.7 Car Back view

CHAPTER 6

CONCLUSION

It has been an interesting journey through the development of this project. At the beginning we used our limited knowledge to implement only the basic features. However, through the months of development, new issues and bugs led to new ideas which led to newer methods of implementation which, in turn, led to us learning even more features of the opengl and apply more creative and efficient ways to perform the older functions. New methods helped us in adding flexibility to the various parts of the program, making the further addition of newer features easier and less time consuming which, again, led to the possibility of adding even more features. This sequential chain reaction of progress and ideas has enabled to learn so much through the months of working on this project and we have done our best to add as many features as we could and provide a user interface that is easy and intuitive to use.

Before concluding, it is worth mentioning that this project would never have been possible without the tremendous amount of encouragement by the staff and guides of our department.

We are content with the outcome of this project and are hopeful that it meets the requirements expected and we wish that it may inspire others to be creative and critical in the field of computer graphics and have a newfound appreciation for the open graphics library.

Although it isn't noticeable on a relatively newer and powerful processor, the method of applying the transforms pixel by pixel is a very taxing process, especially for larger images of 1080p or higher. A possible feature to implement could be to perform all the transforms using multithreading to split the workload onto multiple threads, reducing the minimum computational time required between time steps.

- Since contrast modifies the rgb values instead of hsl, the contrast should be the last thing that should be changed after changing any brightness or saturation.
- Addition of complete pixel blur in both directions, and several other transforms is also notable.

- Allowing the user to select his own color model from RGB, HSL, HSV etc. To allow more fine tuning of the image.
- More efficient memory management techniques to prevent heavy usage of memory

CHAPTER 7

BIBILIOGRAPHY

Reference Books:

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd/4th Edition, Pearson Education,2011

[2] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008

Web Sites:

http://msdn.microsoft.com

http://codeproject.com

http://stackoverflow.com