VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014

A Computer Graphics and Visualization

Mini-Project Report

On

"STEAM ENGINE"

A Mini-project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum.

Submitted by:

JEEVAN RAJU(1DT19CS061)
AND
MADHU SUDHAN S (1DT19CS074)

Under the Guidance of:

Prof. K. Deepa Shree

Asst. Prof. Dept of CSE

Prof. A. Shalini

Asst. Prof. Dept of CSE

Department of Computer Science and Engineering

(Accredited by NBA, NAAC A+, New Delhi)

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Kanakapura Road, Udayapura, Bangalore-560 082

2021-2022

DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT,

Kanakapura Road, Udayapura, Bangalore -560 082

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Accredited by NBA, NAAC A+, New Delhi)

CERTIFICATE

This is to certify that the Mini-Project on Computer Graphics and Visualization work entitled "Steam Engine" has been successfully carried out by Jeevan Raju (1DT19CS061) and Madhu Sudhan (1DT19CS074) a bonafide students of Dayananda Sagar Academy of Technology and Management in partial fulfilment of the requirements for the award of degree in Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Prof. K. Deepa Shree Asst. Prof. Dept of CSE	Prof. A. Shalini Asst. Prof. Dept of CSE

Dr. C. NANDINI

Vice Principal & HOD, Dept. of CSE

Examiners: **Signature with Date**

1:

2:

ABSTRACT

Computer graphics is the process of making the design, 2D, 3D and animation of an object. Computer graphics can do many things, including modeling, simulation and visualization of an object or a problem.

Modeling is a representation of how people describe or explain an object, system, or a concept, whichis usually manifested by simplification or idealization. This can be represented by physical models (mockups, prototypes), the model image (design drawings, computer images), or mathematical formulas.

OpenGL support this modeling capability as OpenGL has additional features to better produce something more realistic. OpenGL allows us to create a graph that can be run on any operating system only minor adjustment.

The 3-D graphics package designed here provides an interface for the users for handling the display and manipulation of Towers of Hanoi. The Keyboard and Mouse is used as input device.

The Engine is made up of a Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and a Crank.
The viewer is allowed to rotate the Engine's Crank either in clock wise or in anti-clock wise direction.
The viewer can also slow up or slow down the Crank speed.

Transparency is set, displays the model twice, first time accepting those fragments with a ALPHA value of 1 only, then with DEPTH_BUFFER writing disabled for those with other values. Initially when the animation is not called ,the crank angle will not change and the window is idle. When called increments the crank angle by ANGLE_STEP updates the head angle and notifies the system that the screen needs to be updated. When a menu option has been selected, it translates the menu item identifier into a keystroke, then calls the keyboard function. A menu will be associated with the mouse too.
The viewer can also see the shaded and textured steam engine

ACKNOWLEDGEMENT

While presenting this Graphics Project on **“STEAM ENGINE”**, I feel that it is our duty to acknowledge the help rendered to us by various persons.

Firstly, I thank God for showering his blessings on me. I am grateful to my institution Dayananda Sagar Academy of Technology for providing me a congenial atmosphere to carry out the project successfully.

I would like to express my heartfelt gratitude to **Dr. M. RaviShankar** Principal, DSATM, Bangalore, for extending his support.

I would also like to express my heartfelt gratitude to **Dr. C Nandini,** Prof. Vice Principal& HOD, Computer Science and Engineering Department, whose guidance and support was truly invaluable.

I am very grateful to my guide, **Prof. K Deepa Shree, Asst. Prof. Dept. of CSE and Prof. A. Shalini, Asst. Prof. Dept of CSE,** Department of Computer Science, for their valuable guidance and advice at every stageof my project which helped me in the successful completion of my project.

I would also indebted to my Parents and Friends for their continued moral and material support throughout the course of project and helping me in finalize the presentation.

My heartful thanks to all those have contributed bits, bytes and words to accomplish this Project.

Thanking you all,

Jeevan Raju (1DT19CS061)

Madhu Sudhan(1DT19CS074)

TABLE OF CONTENTS

1. INTRODUCTION **6**

1.1 Introduction to Steam Engine 6

1.2 Computer Graphics 7

1.3 A Brief Hostory of OpenGL 8

1.3.1 What is OpenGL? 8

1.3.2 Programming Using OpenGL 9

1.4 OpenGL Related Libraries 10

2. REQUIREMENTS SPECIFICATIONS **11**

2.1 Hardware Requirements 11

2.2 Software Requirements 11

2.3 Miscellaneous Requirements 11

2.4 Language Used 11

3. SYSTEM DESIGN **12**

3.1 System Architecture 12

4. IMPLEMENTATION **13**

4.1 Keyboard Function 13

4.2 Display Function 15

4.3 Reshape Function 16

5. SNAPSHOTS **17**

5.1 Initial View 17

5.2 Menu associated with Right Mouse Button 18

5.3 Rotating Steam Engine 19

5.4 Shading 19

5.5 Texture 20

5.6 No Light(Tranparency) 20

6. CONCLUSION AND FUTURE ENHANCEMENT **21**

REFERENCES 22

APPPENDIX 23

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION TO STEAM ENGINE

The aim of this project is to create a STEAM ENGINE. The Engine is made up of a Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and a Crank. The viewer is allowed to rotate the Engine's Crank either in clock wise or in anti-clock wise direction. The viewer can also slow up or slow down the Crank speed.

First a Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and a Crank is created using myCylinder( ) function. The main primitives used inside the myCylinder( ) function to create a Cylinder is gluCylinder( ) and gluDisk( ) . So every time, myCylinder( ) function is called inside the functions used to create Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and Crank. The parts mentioned above are combined to form a Steam Engine image. We can make Steam Engine transparent and display. In display function, at first it clears the drawing buffer and if transparency is set, displays the model twice, first time accepting those fragments with a ALPHA value of 1 only, then with DEPTH_BUFFER writing disabled for those with other values. Initially when the animation is not called, the crank angle will not change and the window is idle. When called increments the crank angle by ANGLE_STEP, updates the head angle and notifies the system that the screen needs to be updated. When a menu option has been selected, it translates the menu item identifier into a keystroke, then calls the keyboard function. A menu will be associated with the mouse too. The viewer can also see the shaded and textured steam engine

The controls are:-

1.	'a'	- > To rotate crank anti-clock wise.
2.	'z'	- > To rotate crank clock wise.
3.	'+' and '-'	- > To speed up and speed down
4.	'o'	- > Transparency.
5.	'0' and 1'	- > Right light and Left light respectively
6.	's' and 't'	- > Shading and Texture respectively

c

1.2 Computer Graphics

The totality of computer graphics software encompasses the concepts from data structures, from data base design and management, from the psychology, ergonometric of the man-machine interface, from programming languages and operating system.

Numerous computer graphics standards can be grouped following categories.

- First is the graphics application interface, where ideas are translated into a form that is understandable by a computer system. Current representative standards are the GKS, GKS-3D, and the Programmer's Hierarchical Interactive Graphics Standards (PHIGS).
- The Second is concerned with the storage and transmission of data between graphics manufacturing systems. The current standard in this area is the Initial Graphics Exchange Specification (IGES).

A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

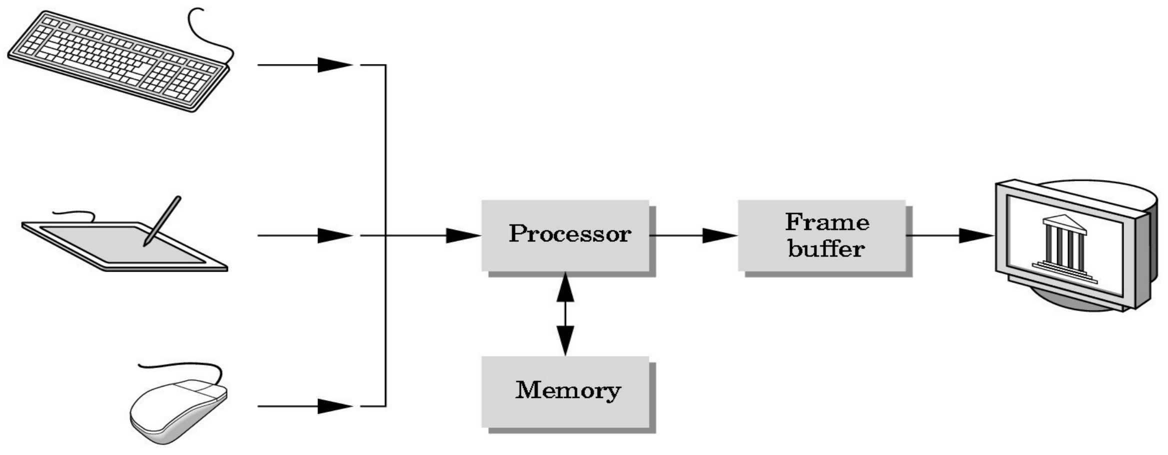


Fig. 1.1: A graphics system

1.3 A Brief History Of OpenGl

OpenGL was developed by Silicon Graphics and is popular in the video games industry where it competes with Direct3D on Microsoft Windows IrisGL, a propietary graphics API, is the precursor of OpenGL. It is developed by Silicon Graphics Inc. (SGI). IrisGL was used as the starting point of an open standard for computer graphics that would save time porting applications by avoiding direct hardware access. After SGI cleaned up IrisGL and opened up the standard to other companies, OpenGL was born.

In 1992 the OpenGL Architectural Review Board (OpenGL ARB) was established. The OpenGL ARB is a group of companies that maintain and update the OpenGL standard.

In 2003 the first OpenGL (Exchange Specification) ES specification was released. OpenGL ES is a subset of OpenGL designed for mobile phones, embedded devices and video game systems.

In 2004 the OpenGL 2.0 specification was released, including the GLSL (OpenGL Shading

1.3.1 What is Open-GL?

- OpenGL is a Software Interface to Graphics Hardware
- OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms
- It Consists of about 250 Distinct Commands.
- It is Hardware-independent Interface
    - No command for windows or user input handling
    - Does not include low-level I/O management
- It is developed primarily by SGI
- It consists of 2D/3D graphics, lower-level primitives (polygons)
- It is Basis for higher-level libraries/toolkits

What does it do?

- Main purpose is to render two and three dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels.

1.3.2 Programming using OpenGL: A first Introduction

OpenGL is an API …

• Application programmers' interface: link between

- low-level: graphics hardware
- high-level: application program you write

OpenGL is a…

Library for 2D and 3D graphics programming

- 200+ functions for building application programs
- Portable to many platforms (Win, Mac, Unix, Linux)
- Callable from many programming languages (C, Java,Perl, Python

Operatios

- Specify geometric primitives (lines, pixels, polygons ...)
- Apply geometric transformations
- Specify camera, light, color, texture information, etc.

• No windowing or (platform-specific) input/interaction

- functions— these are the jobs of GLUT

1.4 OpenGL-Related Libraries

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to allow you to simplify your programming tasks, including the following:

- The OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. GLU routines use the prefix glu.

- The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window system APIs. GLUT is the subject of the next section, and it's described in more detail in Mark Kilgard's book OpenGL Programming for the X Window System (ISBN 0-201-48359-9). GLUT routines use the prefix glut. "How to Obtain the Sample Code" in the Preface describes how to Obtain the source code for GLUT, using ftp.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 Hardware Requirements:

- Intel® Pentium 4 CPU or higher versions
- 128 MB or more RAM.
- A standard keyboard, and Microsoft compatible mouse
- VGA monitor.

2.2 Software requirements:

- The graphics package has been designed for OpenGL; hence the machine must Have code blocks
- Software installed preferably 6.0 or later versions with mouse driver installed.
- GLUT libraries, Glut utility toolkit must be available.
- Operating System: Windows
- Version of Operating System: Windows XP, Windows NT and Higher
- Language: C
- Code::Blocks: cross-platform Integrated Development Environment (IDE)

2.3 MISCELLANEOUS REQUIREMENTS:

All the required library and header files should be available in the include directories. The files associated with this editor should be placed in either the same folder or in a specified folder.

2.4 LANGUAGE USED IN CODING:

C/C++ and OpenGL as an API.

CHAPTER 3

SYSTEM DESIGN

Design of any software depends on the architecture of the machine on which that software runs, for which the designer needs to know the system architecture. Design process involves design of suitable algorithms, modules, subsystems, interfaces etc.

3.1 System Architecture

CONTROL FLOW DIAGRAM

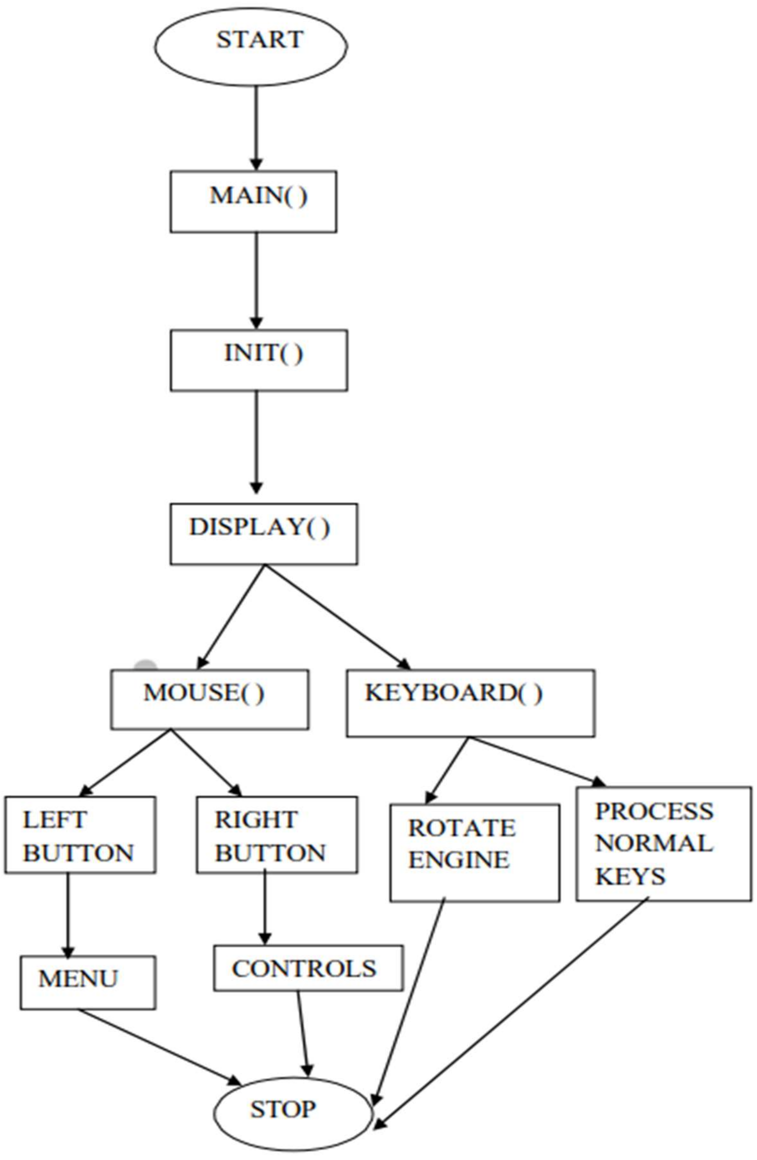


Fig. 3.1: System Control flow diagram

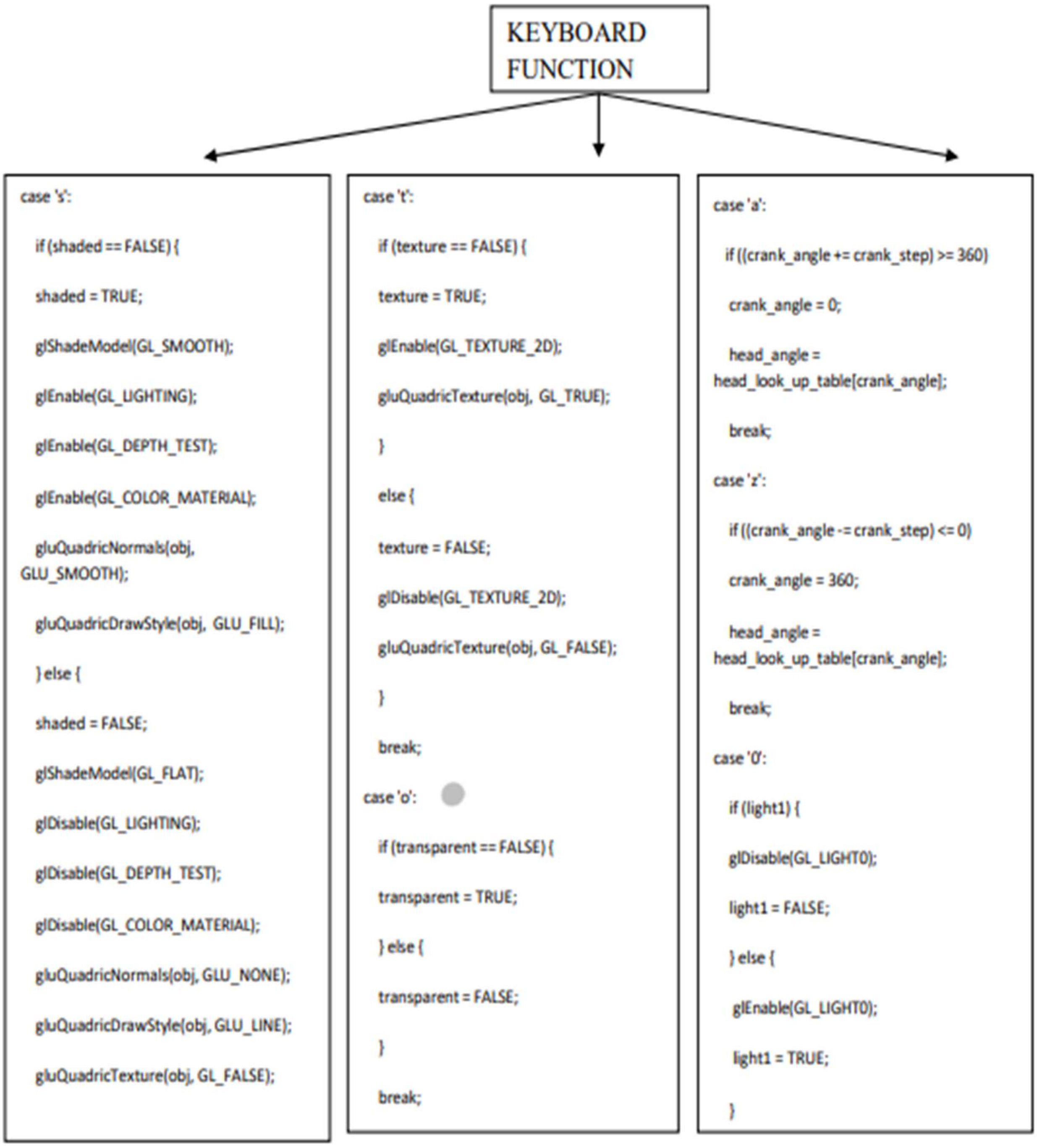
CHAPTER 4

IMPLEMENTATION

The implementation stage of this model involves the following phases.

- Implementation of OpenGL built in functions.
- User defined function Implementation.

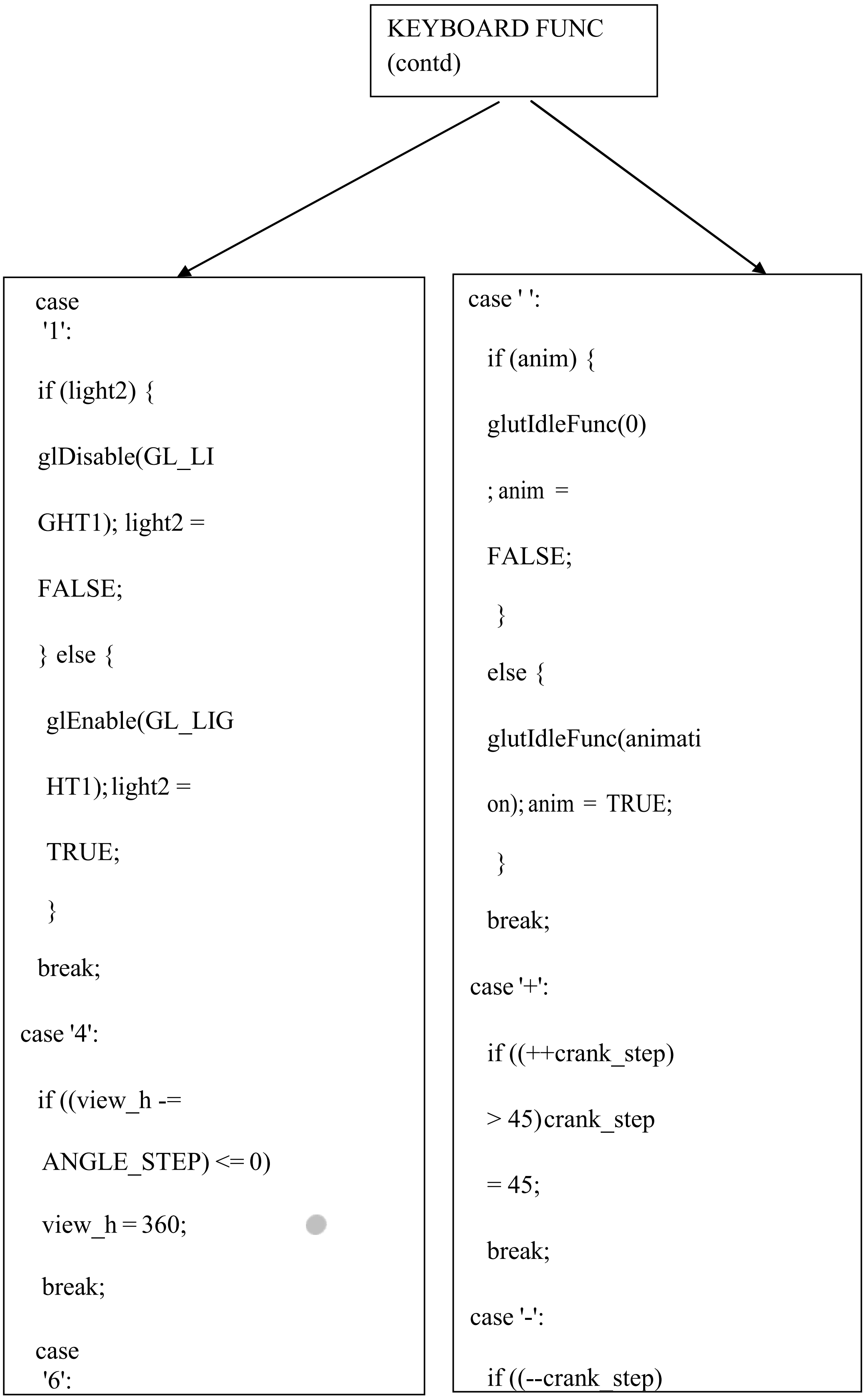
4.1 : KEYBOARD FUNCTION



```
case 's':
  if (shaded == FALSE) {
  shaded = TRUE;
  glShadeModel(GL_SMOOTH);
  glEnable(GL_LIGHTING);
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_COLOR_MATERIAL);
  gluQuadricNormals(obj,
GLU_SMOOTH);
  gluQuadricDrawStyle(obj, GLU_FILL);
  } else {
  shaded = FALSE;
  glShadeModel(GL_FLAT);
  glDisable(GL_LIGHTING);
  glDisable(GL_DEPTH_TEST);
  glDisable(GL_COLOR_MATERIAL);
  gluQuadricNormals(obj, GLU_NONE);
  gluQuadricDrawStyle(obj, GLU_LINE);
  gluQuadricTexture(obj, GL_FALSE);
```

```
case 't':
  if (texture == FALSE) {
  texture = TRUE;
  glEnable(GL_TEXTURE_2D);
  gluQuadricTexture(obj, GL_TRUE);
  }
  else {
  texture = FALSE;
  glDisable(GL_TEXTURE_2D);
  gluQuadricTexture(obj, GL_FALSE);
  }
  break;
case 'o':
  if (transparent == FALSE) {
  transparent = TRUE;
  } else {
  transparent = FALSE;
  }
  break;
```

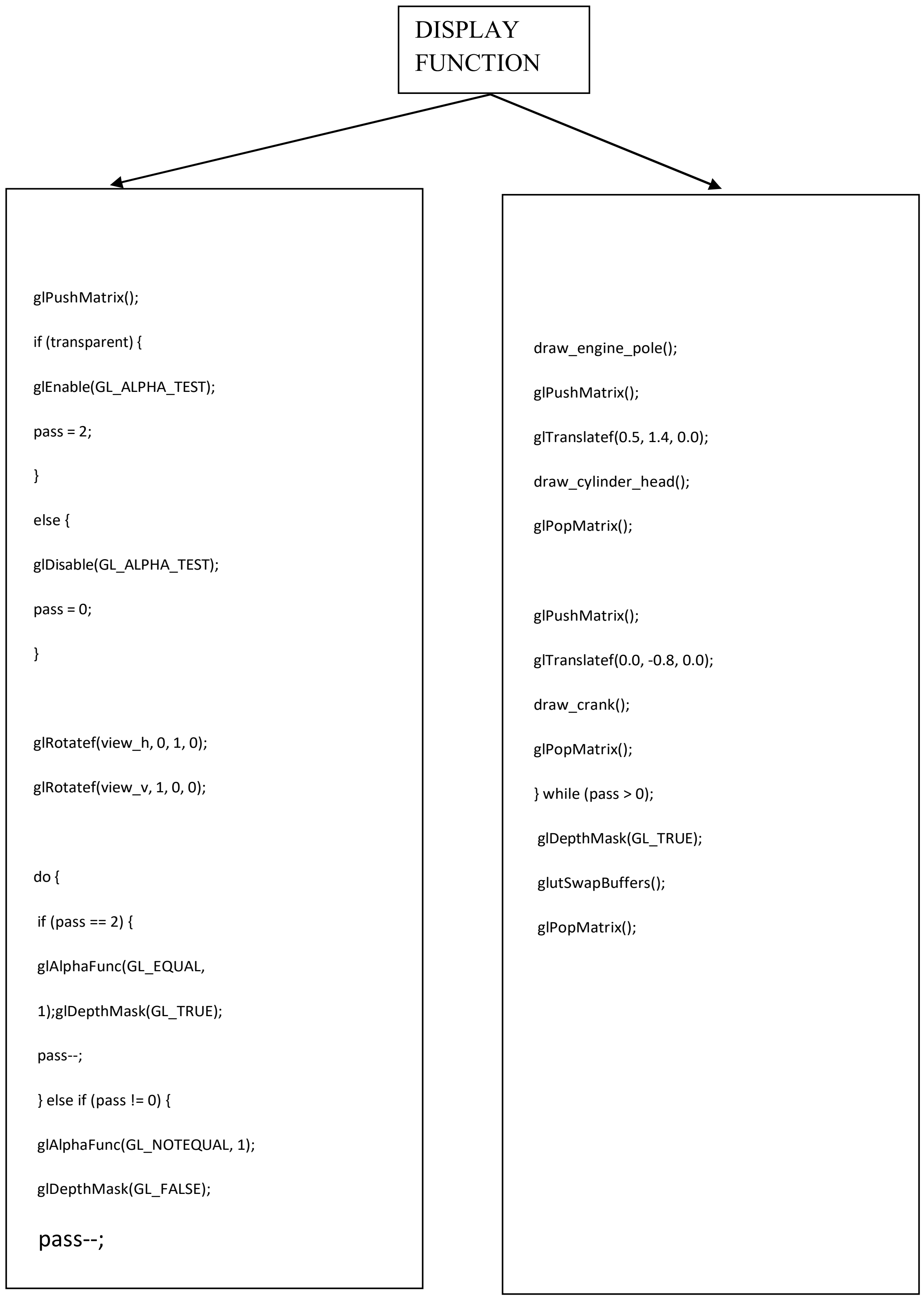
```
case 'a':
  if ((crank_angle += crank_step) >= 360)
  crank_angle = 0;
  head_angle =
head_look_up_table[crank_angle];
  break;
case 'z':
  if ((crank_angle -= crank_step) <= 0)
  crank_angle = 360;
  head_angle =
head_look_up_table[crank_angle];
  break;
case '0':
  if (light1) {
  glDisable(GL_LIGHT0);
  light1 = FALSE;
  } else {
   glEnable(GL_LIGHT0);
   light1 = TRUE;
  }
```



```
case
 '1':

 if (light2) {

 glDisable(GL_LI

 GHT1); light2 =

 FALSE;

 } else {

  glEnable(GL_LIG

  HT1); light2 =

  TRUE;

  }

 break;

case '4':

 if ((view_h -=

  ANGLE_STEP) <= 0)

  view_h = 360;

  break;

case
 '6':
```

```
case ' ':

  if (anim) {

  glutIdleFunc(0)

  ; anim =

  FALSE;

   }

  else {

  glutIdleFunc(animati

  on); anim = TRUE;

   }

  break;

case '+':

  if ((++crank_step)

  > 45) crank_step

  = 45;

  break;

case '-':

  if ((--crank_step)
```

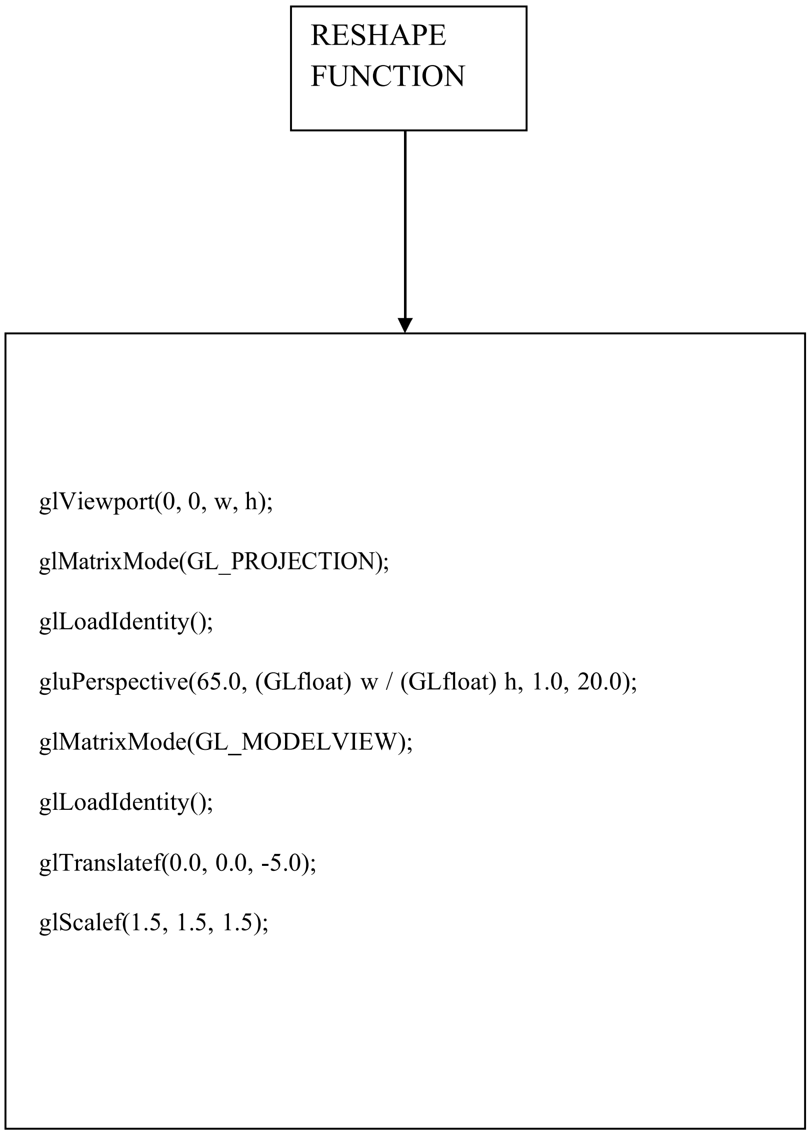
4.2 : DISPLAY FUNCTION



```
glPushMatrix();
if (transparent) {
glEnable(GL_ALPHA_TEST);
pass = 2;
}
else {
glDisable(GL_ALPHA_TEST);
pass = 0;
}

glRotatef(view_h, 0, 1, 0);
glRotatef(view_v, 1, 0, 0);

do {
 if (pass == 2) {
 glAlphaFunc(GL_EQUAL,
 1);glDepthMask(GL_TRUE);
 pass--;
 } else if (pass != 0) {
 glAlphaFunc(GL_NOTEQUAL, 1);
 glDepthMask(GL_FALSE);
 pass--;
```

```
draw_engine_pole();
glPushMatrix();
glTranslatef(0.5, 1.4, 0.0);
draw_cylinder_head();
glPopMatrix();

glPushMatrix();
glTranslatef(0.0, -0.8, 0.0);
draw_crank();
glPopMatrix();
} while (pass > 0);
 glDepthMask(GL_TRUE);
 glutSwapBuffers();
 glPopMatrix();
```

4.3: RESHAPE FUNCTION



```
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(65.0, (GLfloat) w / (GLfloat) h, 1.0, 20.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -5.0);
glScalef(1.5, 1.5, 1.5);
```

CHAPTER 5

SNAPSHOTS

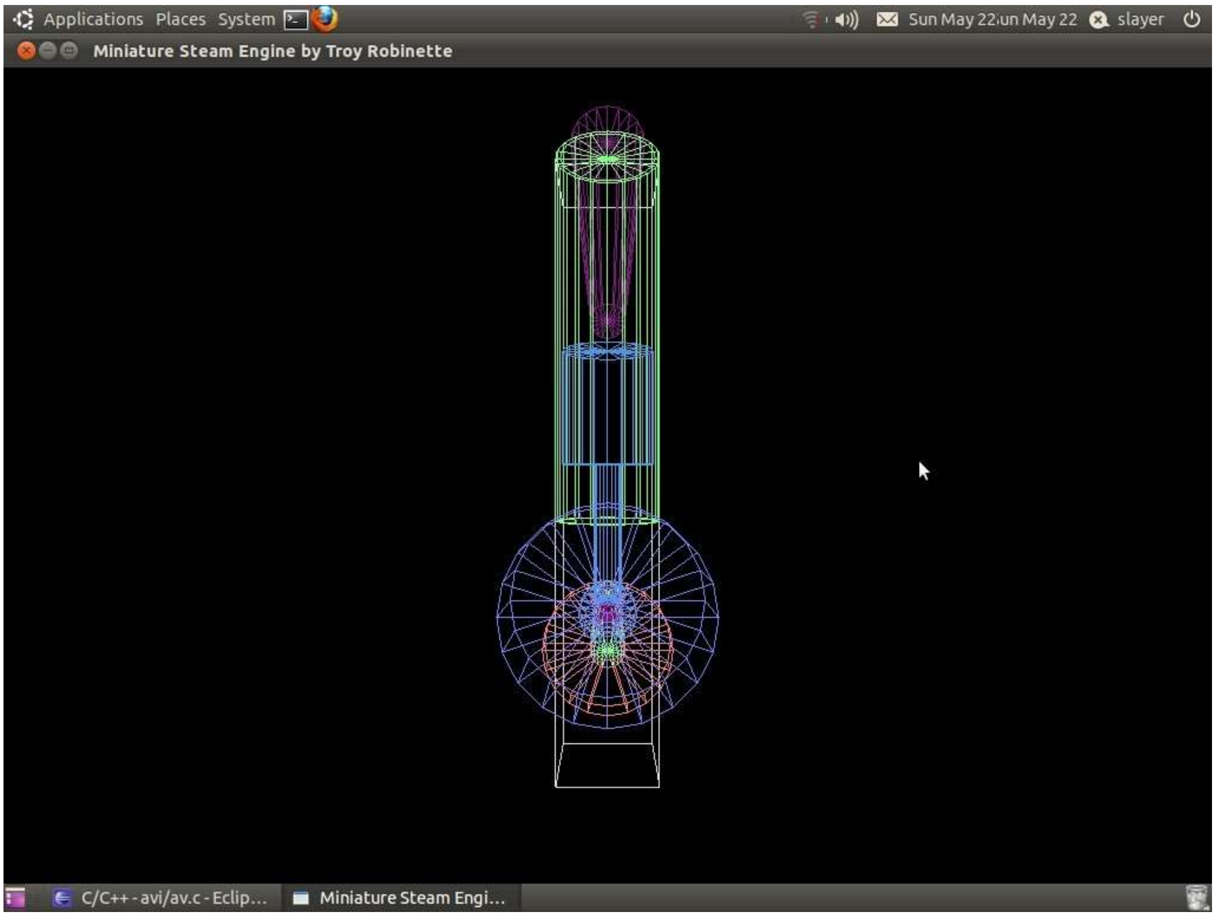

Applications Places System
Sun May 22 Sun May 22 slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.1: Initial View

Explanation of the figure

The Initial View Represents the Engine is made up of a Piston, Engine Pole, Cylinder Head, Flywheel, Crank Bell and a Crank

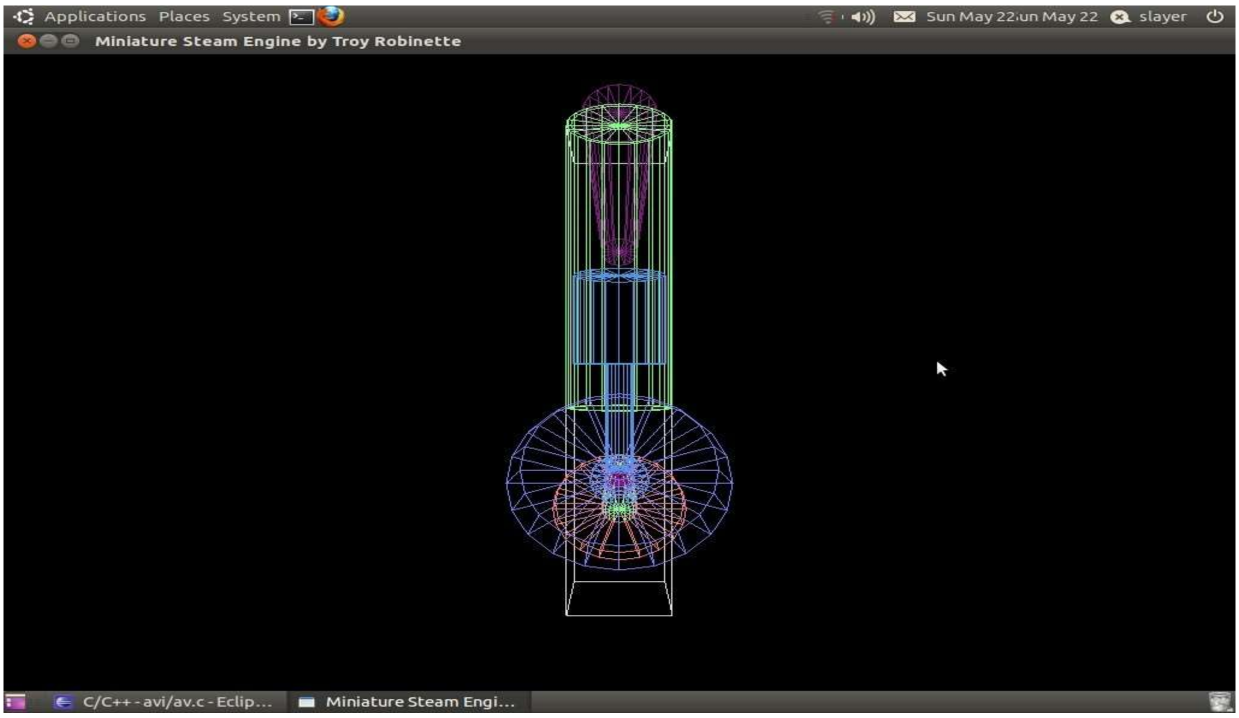

Applications Places System
Sun May 22 un May 22
slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.2: Menu associated with Right Mouse Button

This figure shows the menu of the project in which,

- ‘a’ - > To rotate crank anti-clock wise.
- ‘z’ - > To rotate crank clock wise.
- ‘+’ and ‘-‘ - > To speed up and speed down
- ‘o’ - > Transparency.
- ‘0’ and 1’ - > Right light and Left light respectively
- ‘s’ and ‘t’ - > Shading and Texture respectively

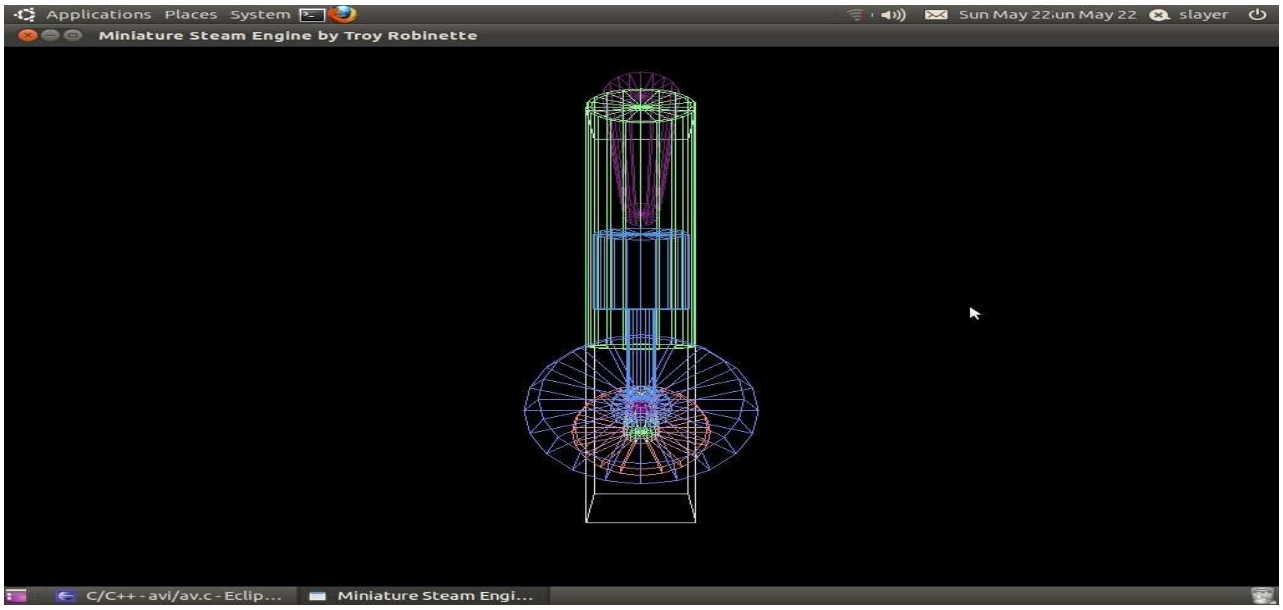

Applications Places System
Sun May 22 un May 22
slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.3: Rotating Steam Engine

Explanation of the figure

This figure shows how the Engine is rotated with respect to axis.

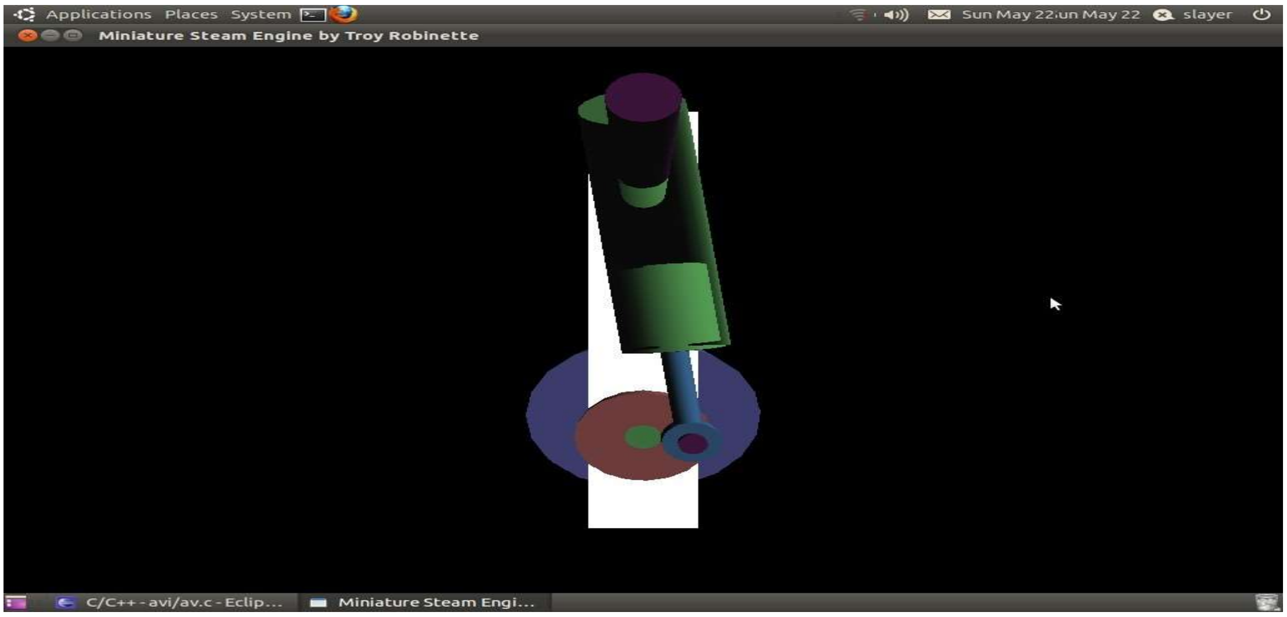

Applications Places System
Sun May 22 un May 22
slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.4:Shading

Explanation of the figure:

Here the shade of the object is changed from Transparency to solid

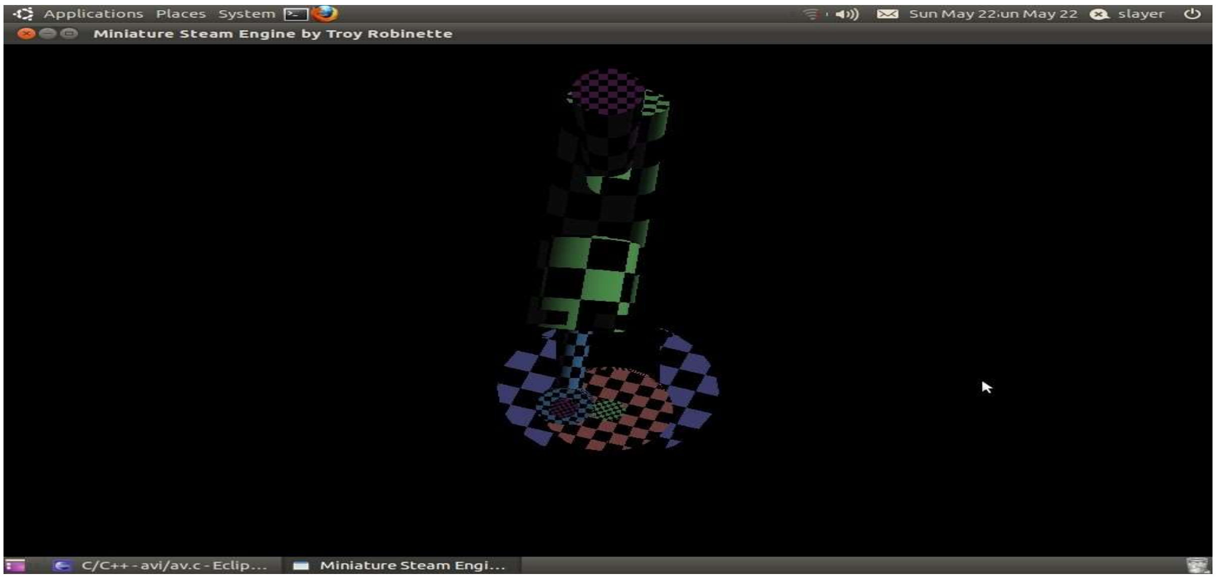

Applications Places System
Sun May 22 slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.5: Texture

Explanation of the figure:

Here the shade of the object is changed from Solid to Texture

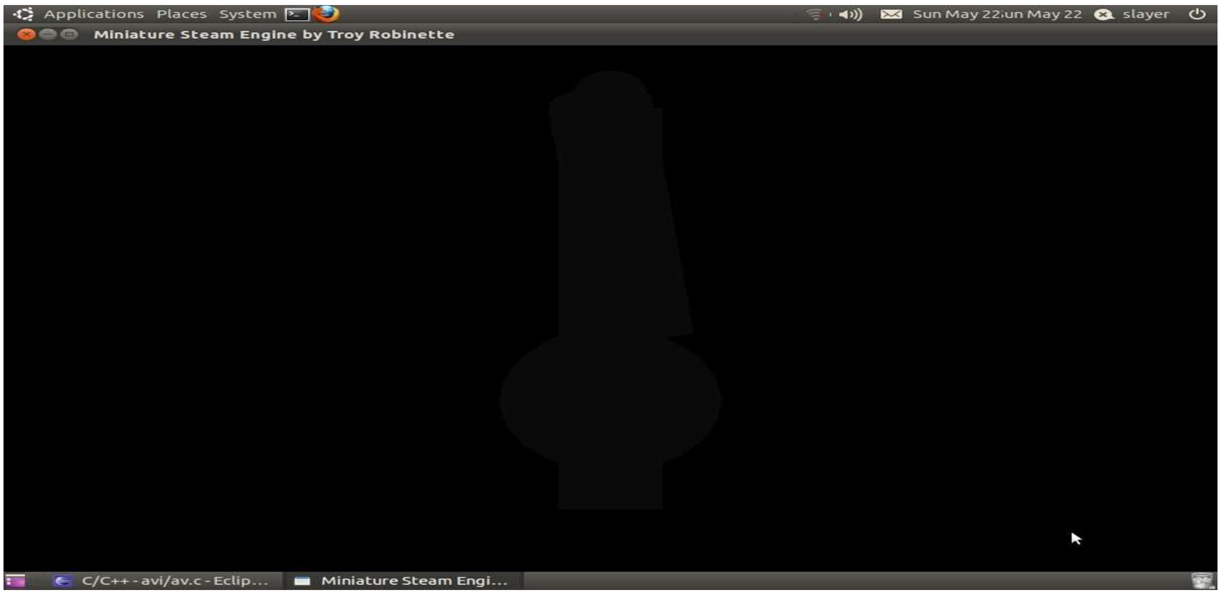

Applications Places System
Sun May 22 slayer
Miniature Steam Engine by Troy Robinette
C/C++ - avi/av.c - Eclip...
Miniature Steam Engi...

Fig. 5.6: No Light (Tranparency)

Explanation of the figure:

Here there is no light present on the screen the object get disappeared

CHAPTER 6

FUTURE ENHANCEMENT

This project allows the user to rotate the piston in a Steam Engine. Its like a miniature Steam Engine Simulation.

Future scope:

1. SIMULATOR

Not only the movement of piston, we can make the whole parts in the steam engine working so that it will be a simulator of steam engine. By modifying this project we can construct a fully fledged simulator. Students who are studying about the steam engine can work through this and it will be very helpful for them. Almost a complete picturization of a steam engine can be done through this.

2.DESIGN OF STEAM ENGINES

Engineers who build Steam Engines can design their model by looking this project. They get a good picturization by seeing this and it will be helpful for them in building steam engines. So this project will be benefited to Engineers

REFERENCES

Books:

1. Computer Graphics with OpenGL Version,3rd Edition, Donald Hearn & Pauline Baker 2.Interactive Computer Graphics-A Top Down approach with OpenGL,5rd Edition, Edward Angel

Websites:

1. http://www.amazon.in/Computer-Graphics-Multimedia-Udit-Agarwal/dp/935014316X?tag=googinhydr18418-21
2. http://en.wikipedia.org/wiki/Computer_graphics
3. http://stackoverflow.com/questionsquickly
4. http://www.opengl-tutorial.org/intermediate-tutorials/
5. http://www.opengl-tutorial.org/
6. https://open.gl/
7. http://www.cs.uccs.edu/~ssemwal/indexGLTutorial.html
8. http://www.videotutorialsrock.com/
9. http://ogldev.atspace.co.uk/
10. https://www.opengl.org/sdk/docs/tutorials/
11. http://learnopengl.com/
12. http://lazyfoo.net/tutorials/OpenGL/
13. http://en.wikibooks.org/wiki/OpenGL_Programming

APPENDIX

A.1 Source code

```
#include <stdio.h>
#include <GL/glut.h>
#include <math.h>

#define TRUE   1
#define FALSE 0

/* Dimensions of texture image. */#define
IMAGE_WIDTH  64
#define IMAGE_HEIGHT 64

/* Step to be taken for each rotation. */#define
ANGLE_STEP 10

/* Magic numbers for relationship b/w cylinder head and crankshaft. */
  #define   MAGNITUDE  120
  #define   PHASE          270.112
  #define   FREQ_DIV      58
  #define   ARC_LENGHT2.7
  #define   ARC_RADIUS 0.15

/* Rotation angles */
GLdouble view_h = 270, view_v = 0, head_angle = 0;GLint
crank_angle = 0;

/* Crank rotation step. */
GLdouble crank_step = 5;

/* Toggles */
GLshort shaded = TRUE, anim = FALSE;
GLshort texture = FALSE, transparent = FALSE;
GLshort light1 = TRUE, light2 = FALSE;

/* Storage for the angle look up table and the texture map */GLdouble
head_look_up_table[361];
GLubyte image[IMAGE_WIDTH][IMAGE_HEIGHT][3];

/* Indentifiers for each Display list */GLint
list_piston_shaded = 1;
GLint list_piston_texture = 2; GLint
```

```
list_flywheel_shaded = 4; GLint
list_flywheel_texture = 8;

/* Variable used in the creaton of glu objects */
GLUquadricObj *obj;


/* Draws a box by scaling a glut cube of size 1. Also checks the shadedtoggle to see
    which rendering style to use. NB Texture doesn't work correctly due to the cube
    being scaled. */
void
myBox(GLdouble x, GLdouble y, GLdouble z)
{
   glPushMatrix();
   glScalef(x, y, z);if
   (shaded)
   glutSolidCube(1); else
   glutWireCube(1);
   glPopMatrix();
}

/* Draws a cylinder using glu function, drawing flat disc's at each end,to give the
    appearence of it being solid. */
void
myCylinder(GLUquadricObj * object, GLdouble outerRadius,
   GLdouble innerRadius, GLdouble lenght)
{
   glPushMatrix();
   gluCylinder(object, outerRadius, outerRadius, lenght, 20, 1);
   glPushMatrix();
   glRotatef(180, 0.0, 1.0, 0.0);
   gluDisk(object, innerRadius,    outerRadius, 20, 1);
   glPopMatrix();
   glTranslatef(0.0, 0.0, lenght);
   gluDisk(object, innerRadius, outerRadius, 20, 1);
   glPopMatrix();
}

/* Draws a piston. */void
draw_piston(void)
{
   glPushMatrix();
   glColor4f(0.3, 0.6, 0.9, 1.0);
   glPushMatrix();
   glRotatef(90,0.0,1.0, 0.0);
   glTranslatef(0.0, 0.0, -0.07);
   myCylinder(obj, 0.125, 0.06, 0.12);
   glPopMatrix();

   glRotatef(-90, 1.0, 0.0, 0.0);
   glTranslatef(0.0, 0.0, 0.05);
   myCylinder(obj, 0.06, 0.0, 0.6);
   glTranslatef(0.0, 0.0, 0.6);
   myCylinder(obj, 0.2, 0.0, 0.5);
   glPopMatrix();
}

/* Draws the engine pole and the pivot pole for the cylinder head. */void
draw_engine_pole(void)
{
   glPushMatrix(); glColor4f(0.9, 0.9,
   0.9, 1.0);
   myBox(0.5, 3.0, 0.5);
   glColor3f(0.5, 0.1, 0.5);
   glRotatef(90, 0.0, 1.0, 0.0);
   glTranslatef(0.0, 0.9, -0.4);
   myCylinder(obj, 0.1, 0.0, 2);
   glPopMatrix();
}

/* Draws the cylinder head at the appropreate angle, doing the necesarytranslations for
    the rotation. */
void
draw_cylinder_head(void)
{
   glPushMatrix(); glColor4f(0.5, 1.0,
   0.5, 0.1);
   glRotatef(90, 1.0, 0.0, 0.0);
   glTranslatef(0, 0.0, 0.4);
   glRotatef(head_angle, 1, 0, 0);
   glTranslatef(0, 0.0, -0.4);
   myCylinder(obj, 0.23, 0.21, 1.6);
   glRotatef(180, 1.0, 0.0, 0.0);
   gluDisk(obj, 0, 0.23, 20, 1);
   glPopMatrix();
}

/* Draws the flywheel. */void
draw_flywheel(void)
{
   glPushMatrix(); glColor4f(0.5, 0.5,
   1.0, 1.0);
   glRotatef(90, 0.0, 1.0, 0.0);

   myCylinder(obj, 0.625, 0.08, 0.5);
   glPopMatrix();
}

/* Draws the crank bell, and the pivot pin for the piston. Also calls the appropreate display
    list of a piston doing the nesacary rotations beforehand.  */
void
draw_crankbell(void)
{
   glPushMatrix(); glColor4f(1.0, 0.5,
   0.5, 1.0);
   glRotatef(90, 0.0, 1.0, 0.0);
   myCylinder(obj, 0.3, 0.08, 0.12);

   glColor4f(0.5, 0.1, 0.5, 1.0);
   glTranslatef(0.0, 0.2, 0.0);
   myCylinder(obj, 0.06, 0.0, 0.34);

   glTranslatef(0.0, 0.0, 0.22);
   glRotatef(90, 0.0, 1.0, 0.0);
   glRotatef(crank_angle - head_angle, 1.0, 0.0, 0.0);if (shaded) {
           if (texture) glCallList(list_piston_texture);
           else
               glCallList(list_piston_shaded);
       } else
           draw_piston();
   glPopMatrix();
}
```

```
/* Draws the complete crank. Piston also gets drawn through the crank bellfunction. */
void
draw_crank(void)
{
   glPushMatrix(); glRotatef(crank_angle, 1.0,
   0.0, 0.0);glPushMatrix();
   glRotatef(90, 0.0, 1.0, 0.0);
   glTranslatef(0.0, 0.0, -1.0);
   myCylinder(obj, 0.08, 0.0, 1.4);
   glPopMatrix();
   glPushMatrix();
    glTranslatef(0.28, 0.0, 0.0);

   draw_crankbell();
   glPopMatrix();
   glPushMatrix();
   glTranslatef(-0.77, 0.0, 0.0);if
   (shaded) {
            if (texture) glCallList(list_flywheel_texture);
            else
               glCallList(list_flywheel_shaded);
         } else
            draw_flywheel();
   glPopMatrix();
   glPopMatrix();
}

/* Main display routine. Clears the drawing buffer and if transparency isset, displays the
    model twice, 1st time accepting those fragments witha ALPHA value of 1 only, then
    with DEPTH_BUFFER writing disabled for those with other values. */
void
display(void)
{
   int pass;

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glPushMatrix();
      if (transparent) {
```

```
                glEnable(GL_ALPHA_TEST);  pass
                = 2;
            } else {
                glDisable(GL_ALPHA_TEST);
                pass = 0;
            }

            /* Rotate the whole model */
            glRotatef(view_h, 0, 1, 0);
            glRotatef(view_v, 1, 0, 0);

            do {
                if (pass == 2) {
                    glAlphaFunc(GL_EQUAL, 1);
                    glDepthMask(GL_TRUE);
                    pass--;
                } else if (pass != 0) {
                    glAlphaFunc(GL_NOTEQUAL,1)


                    glDepthMask(GL_FALSE)
                    ;pass--;
                }
                draw_engine_pole();

                glPushMatrix(); glTranslatef(0.5, 1.4,
                    0.0);draw_cylinder_head();
                glPopMatrix();

                glPushMatrix(); glTranslatef(0.0, -0.8,
                    0.0);draw_crank();
                glPopMatrix();
            } while (pass > 0);
            glDepthMask(GL_TRUE)
            ;glutSwapBuffers();
        glPopMatrix();
    }

    /* Called when the window is idle. When called increments the crank angleby
        ANGLE_STEP, updates the head angle and notifies the system that the screen needs
        to be updated. */
```

```
void
animation(void)
{
  if ((crank_angle += crank_step) >= 360)
    crank_angle = 0;
  head_angle = head_look_up_table[crank_angle];
  glutPostRedisplay();
}

/* Called when a key is pressed. Checks if it reconises the key and if soacts on it,
   updateing the screen. */
/* ARGSUSED1
*/void
keyboard(unsigned char key, int x, int y)
{
  switch (key) {
  case 's':
    if (shaded == FALSE) { shaded
      = TRUE;
      glShadeModel(GL_SMOOTH);
      glEnable(GL_LIGHTING);
      glEnable(GL_DEPTH_TEST);
      glEnable(GL_COLOR_MATERIAL);

      gluQuadricNormals(obj, GLU_SMOOTH);
      gluQuadricDrawStyle(obj, GLU_FILL);
    } else {
      shaded = FALSE;
      glShadeModel(GL_FLAT);
      glDisable(GL_LIGHTING);
      glDisable(GL_DEPTH_TEST);
      glDisable(GL_COLOR_MATERIAL);
      gluQuadricNormals(obj, GLU_NONE);
      gluQuadricDrawStyle(obj, GLU_LINE);
      gluQuadricTexture(obj, GL_FALSE);
    }
    if (texture && !shaded);else
      break;
  case 't':
    if (texture == FALSE) { texture =
        TRUE;
        glEnable(GL_TEXTURE_2D);
        gluQuadricTexture(obj, GL_TRUE);
    } else {
        texture = FALSE;
        glDisable(GL_TEXTURE_2D);
        gluQuadricTexture(obj, GL_FALSE);
    }
    break; case
'o':
    if (transparent == FALSE) {
        transparent = TRUE;
    } else {
        transparent = FALSE;
    }
    break;

case 'a':
    if ((crank_angle += crank_step) >= 360)
        crank_angle = 0;
    head_angle = head_look_up_table[crank_angle];break;
case 'z':
    if ((crank_angle -= crank_step) <= 0)
        crank_angle = 360;
    head_angle = head_look_up_table[crank_angle];break;
case '0':
    if (light1) {


        glDisable(GL_LIGHT0);
        light1 = FALSE;
    } else {
        glEnable(GL_LIGHT0);
        light1 = TRUE;
    }
    break; case
'1':
    if (light2) {
        glDisable(GL_LIGHT1);
        light2 = FALSE;
    } else {
        glEnable(GL_LIGHT1);
        light2 = TRUE;
    }
    break; case
'4':
    if ((view_h -= ANGLE_STEP) <= 0)
        view_h = 360;
    break; case
'6':
    if ((view_h += ANGLE_STEP) >= 360)
        view_h = 0;
    break; case
'8':
    if ((view_v += ANGLE_STEP) >= 360)
        view_v = 0;
    break;
case '2':
  if ((view_v -= ANGLE_STEP) <= 0)
        view_v = 360;
    break;
case ' ':
    if (anim) {
        glutIdleFunc(0);
        anim = FALSE;
    } else { glutIdleFunc(animation);
        anim = TRUE;
    }
    break; case
'+':
    if ((++crank_step) > 45)
        crank_step = 45;
    break; case
'-':


    if ((--crank_step) <= 0)
        crank_step = 0;
    break;
default:
    return;
    }
    glutPostRedisplay();
}

/* ARGSUSED1
*/void
special(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_LEFT:
        if ((view_h -= ANGLE_STEP) <= 0)
            view_h = 360;
        break;
    case GLUT_KEY_RIGHT:
        if ((view_h += ANGLE_STEP) >= 360)
            view_h = 0;
        break;
    case GLUT_KEY_UP:
        if ((view_v += ANGLE_STEP) >= 360)
            view_v = 0;
        break;
    case GLUT_KEY_DOWN:
        if ((view_v -= ANGLE_STEP) <= 0)
            view_v = 360;
        break;
    default:
        return;
    }
    glutPostRedisplay();
}
/* Called when a menu option has been selected. Translates the menu itemidentifier into a
    keystroke, then call's the keyboard function. */
void
menu(int val)
{
    unsigned char key;

    switch (val) {
    case 1:
        key = 's';
        break;
    case 2:
        key = ' ';
        break;
    case 3:
        key = 't';
        break;
    case 4:
        key = 'o';
        break;
    case 5:
        key = '0';
        break;
    case 6:
        key = '1';
        break;
    case 7:
        key = '+';
        break;
    case 8:
        key = '-';
        break;
    default:
        return;
    }
    keyboard(key, 0, 0);
}
void
create_menu(void)
{
    glutCreateMenu(menu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutAddMenuEntry("Shaded", 1);
    glutAddMenuEntry("Animation", 2);
    glutAddMenuEntry("Texture", 3);
    glutAddMenuEntry("Transparency", 4);
    glutAddMenuEntry("Right Light (0)", 5);
    glutAddMenuEntry("Left Light (1)", 6);
    glutAddMenuEntry("Speed UP", 7);
    glutAddMenuEntry("Slow Down", 8);
```

```
}

/* Makes a simple check pattern image. (Copied from the redbook example"checker.c".)
    */
void
make_image(void)
{

  int i, j, c;

  for (i = 0; i < IMAGE_WIDTH; i++) { for (j =
    0; j < IMAGE_HEIGHT; j++) {
      c = (((i & 0x8) == 0) ^ ((j & 0x8) == 0)) * 255;
      image[i][j][0] = (GLubyte) c;
      image[i][j][1] = (GLubyte) c;
      image[i][j][2] = (GLubyte) c;
    }
  }
}

/* Makes the head look up table for all possible crank angles. */void
make_table(void)
{
  GLint i;
  GLdouble k;

  for (i = 0, k = 0.0; i < 360; i++, k++) {
    head_look_up_table[i] =
      MAGNITUDE * atan(
      (ARC_RADIUS * sin(PHASE - k / FREQ_DIV)) /
      ((ARC_LENGHT - ARC_RADIUS * cos(PHASE - k / FREQ_DIV))));
  }
}

/* Initialises texturing, lighting, display lists, and everything elseassociated with the
    model. */
void
myinit(void)
{
  GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};GLfloat
  mat_shininess[] = {50.0};
```

```
GLfloat light_position1[] = {1.0, 1.0, 1.0, 0.0};
GLfloat light_position2[] = {-1.0, 1.0, 1.0, 0.0};

glClearColor(0.0, 0.0, 0.0, 0.0);

obj = gluNewQuadric();
make_table();
make_image();

/* Set up Texturing */
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexImage2D(GL_TEXTURE_2D, 0, 3, IMAGE_WIDTH,

   IMAGE_HEIGHT, 0, GL_RGB, GL_UNSIGNED_BYTE,
   image);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);glTexParameterf(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);

/* Set up Lighting */
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
glLightfv(GL_LIGHT0, GL_POSITION, light_position1);
glLightfv(GL_LIGHT1, GL_POSITION, light_position2);

/* Initial render mode is with full shading and LIGHT 0enabled. */
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glDepthFunc(GL_LEQUAL);
glEnable(GL_DEPTH_TEST);
glDisable(GL_ALPHA_TEST);

glColorMaterial(GL_FRONT_AND_BACK,
GL_DIFFUSE);glEnable(GL_COLOR_MATERIAL);
glShadeModel(GL_SMOOTH);
```

```
    /* Initialise display lists */ glNewList(list_piston_shaded,
    GL_COMPILE);
       draw_piston();
    glEndList();
    glNewList(list_flywheel_shaded, GL_COMPILE);
       draw_flywheel();
    glEndList();

    gluQuadricTexture(obj, GL_TRUE);
    glNewList(list_piston_texture, GL_COMPILE);
       draw_piston();
    glEndList();
    glNewList(list_flywheel_texture, GL_COMPILE);
       draw_flywheel();
    glEndList(); gluQuadricTexture(obj,
    GL_FALSE);
}

/* Called when the model's window has been reshaped. */void


myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(65.0, (GLfloat) w / (GLfloat) h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -5.0); /* viewing transform */glScalef(1.5, 1.5, 1.5);
}
/* Main program. An interactive model of a miniture steam engine.
     Sets system in Double Buffered mode and initialises all the call-backfunctions. */
int
main(int argc, char **argv)
{
    puts("Steam Engine\n");

    puts("Keypad Arrow keys (with NUM_LOCK on) rotates object."); puts("Rotate
    crank: 'a' = anti-clock wise 'z' = clock wise"); puts("Crank Speed : '+' = Speed up
    by 1                    '-' = Slow Down by 1");puts("Toggle  : 's' = Shading  't' =
    Texture");
    puts("                    : ' ' = Animation             'o' = Transparency"); puts("
                              : '0' = Right Light           '1' = Left Light"); puts("
    Alternatively a pop up menu with all toggles is attached");puts("to the left
    mouse button.\n");

    glutInitWindowSize(400, 400);
    glutInit(&argc, argv);

    /* Transperancy won't work properly without GLUT_ALPHA */
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH |
GLUT_MULTISAMPLE);
    glutCreateWindow("Steam Engine");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(special); create_menu();
    myinit();
    glutReshapeFunc(myReshape);
    glutMainLoop();
    return 0;                   /* ANSI C requires main to return int. */
}
```

A.2 Personal Details

NAME: JEEVAN RAJU

USN: 1DT19CS061

SEMESTER AND SECTION: 6TH SEM, B SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID:

NAME: MADHU SUDHAN S

USN: 1DT19CS074

SEMESTER AND SECTION: 6TH SEM, B SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID: m3s6364@gmail.com