Visvesvaraya Technological University, Belgavi–590014
"Jnana Sangama",Belagavi-590014,Karnataka,India

MINI PROJECT REPORT
ON

"STEAM ENGINE"

As prescribed by VTU for fifth semester mini project for the award of

DBMS WITH MINI PROJECT LABORATORY
FOR
BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

For the Academic year
2020-2021
Submitted By:

ANEESH A A	4SH18CS011
MANOJ KUMAR Y S	4SH18CS041

Under the Guidance of

Mrs. NISHA COUTINHO
(Asst. professor, Department of CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SHREE DEVI INSTITUTE OF TECHNOLOGY

KENJAR, MANGALURU-574142

SHREE DEVI INSTITUTE OF TECHNOLOGY

(An Institution under VTU, Belagavi)

MANGALURU -574 142

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled "**STEAM ENGINE**" is a bonafide work carried out by **ANEESH A A, MANOJ KUMAR Y S** bearing USN's **4SH18CS011, 4SH18CS041** respectively in partial fulfillment for the award of degree of **Bachelor of Engineering in computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the degree of Bachelor of Engineering.

Signature of the Guide	Signature of HOD
Mrs. Nisha Coutinho	Prof.Anand S Uppar
Dept.of CSE	HOD.Dept.of CSE

EXTERNAL VIVA

Name of the Examiners	Signature with Date
1.	……..
2.	……...

SHREE DEVI INSTITUTE OF TECHNOLOGY

KENJAR, MANGALURU-574142

Department of Computer Science and Engineering

DECLARATION

We **ANEESH A A, MANOJ KUMAR Y S** bearing USN's 4SH18CS011, 4SH18CS041 respectively, students of fifth semester Bachelor of Engineering, Computer Science and Engineering, Shree Devi institute of Technology Mangaluru declare that mini project work entitled **"STEAM ENGINE"** has been duly executed by us under the guidance of Mrs .Nisha Coutinho , Asst. Professor, Department of Computer Science and Engineering, Shree Devi Institute Of Technology, Mangaluru and submitted for the requirements for 5th semester Data Base Application and Mini project of **Bachelor of Engineering** in **Computer Science and Engineering during the year 2020-2021.**

Date: ANEESH A A (4SH18CS011)

Place: Mangaluru MANOJ KUMAR Y S(4SH18CS041)

ABSTRACT

As the name specifies “STEAM ENGINE” is software developed for managing various activities in the hostel. For the past few years the number of educational institutions is increasing rapidly. Thereby the number of hostels also increasing for the accommodation of the students studying in this institutions. And hence there is a lot of strain on the person who are running the hostel and software’s are not usually used in context. This particular project deals with the problems on managing a hostel and avoid the problems which occur when carried.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

ACKNOWLEDGMENT

A successful project is a fruitful culmination of efforts of many people. Some directly involved and others who have quietly encouraged and extended their invaluable support throughout its progress.

We would also like to convey our heartfelt thanks to our **Management** for providing us with the good infrastructure laboratory facility, qualified and inspiring staff whose guidance was of great help in successful completion of this project.

We are extremely grateful and thankful to our beloved **Principal Dr. K. E. Prakash** for providing a congenial atmosphere and also the necessary facilities for achieving the cherished goal.

We are very thankful to our Director **Dr. K.E Prakash** for his unconditional support for the successful completion of our project.

We feel delighted to have this page to express my sincere thanks and deep appreciation to **Prof.Anand S.Uppar** ,**Head of the Department ,Computer Science and Engineering,** for his valuable guidance, keen insert and constant encouragement throughout the entire period of this project work.

We would like to thank my project guide **Mrs. Nisha Coutinho,** guide, Assistant Professor, **Computer Science & Engineering** for his valuable guidance and constant support throughout the project work.

We are thankful to all teaching and non-teaching staff for allowing us to successfully carryout the project work.

Finally, we also thank our family and friends who provided lot of support in this project work.

ANEESH A A

MANOJ KUMAR Y S

TABLE OF CONTENT

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

Introduction to OpenGL

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the OpenGL Utility Library (GLU). This 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 letters glu. To interface with the window system and to get input from external devices into our programs, we need at least one more system-specific library that provides the "glue" between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system. Fig 2.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

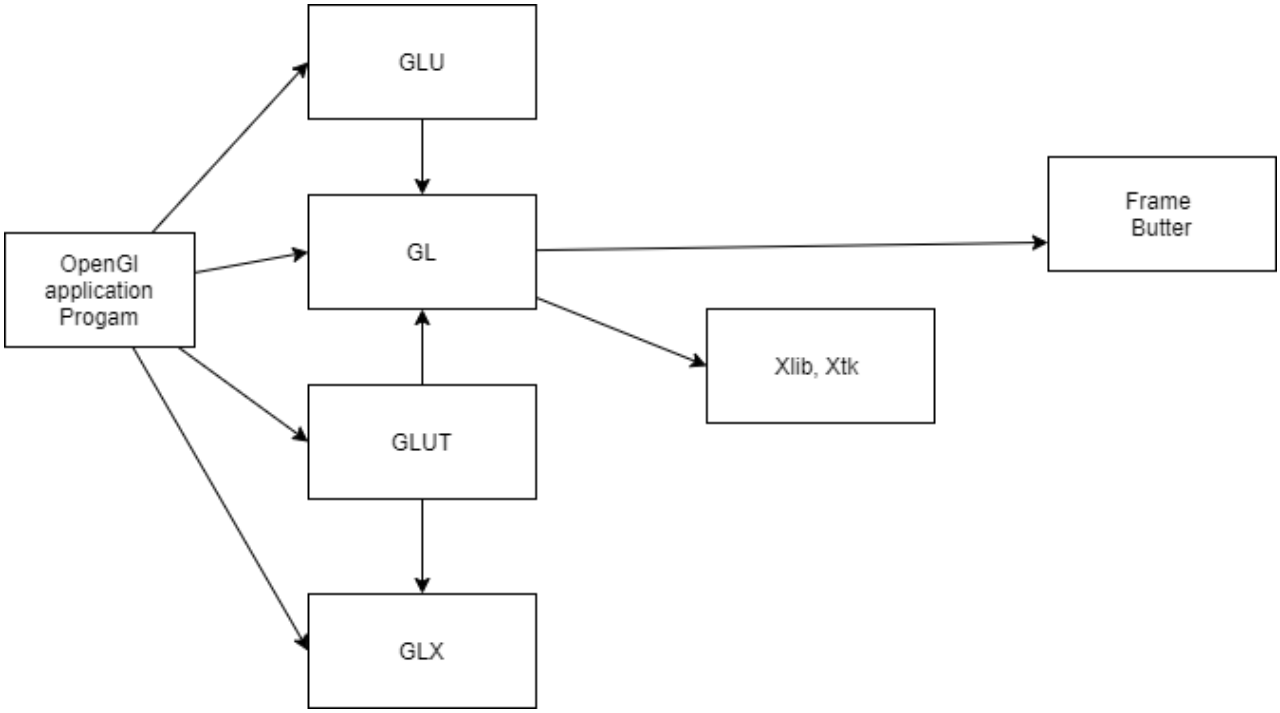


Fig 1.2 Library organization

1.1.1 OpenGL Common Syntax

OpenGL commands use the prefix **gl** and initial capital letter for each word making up command name. Similarly, OpenGL defined constants begin with GL_, use all capital letters and underscores to separate words(like GL_COLOR_BUFFER_BIT).

CHAPTER 2

LITRATURE SUREY

This project makes extensive use of translations, rotations and scaling for creating .

1. THE LINKS : http://www.cs.rutgers.edu/~decarlo/428/glman.html online man pages.

2. http://www.opengl.org online man pages.

3. http://nehe.gamedev.net OpenGL tutorials.

Provides the description of the following functions.

void glScalef(TYPE sx, TYPE sy, TYPE sz)

alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat.

Here in the above considered example we use scaling to minimize the length of the curve at each iteration. For this curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.

void glRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz)

alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz). TYPE here is GLfloat.

For a Koch curve we rotate by 60° about the z-axis.

void glTranslatef(TYPE x, TYPE y, TYPE z)

alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat. We need to translate to display the new position of the line from the old position and also to go out to the beginning of the next side while drawing.

void glLoadIdentity()

sets the current transformation matrix to an identity matrix.

void glPushMatrix()

pushes to the matrix stack corresponding to the current matrix mode.

void glPopMatrix()

pops from the matrix stack corresponding to the current matrix mode.

void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)

defines a two-dimensional viewing rectangle in the plane z=0.

void glutMouseFunc(myMouse)

refers to the mouse callback function. The function to callback is defined as

void myMouse(int button, int state, int x

{

if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)

if (glutGetModifiers() & GLUT_ACTIVE_SHIFT)

decrease a level of recursion

else increase a level of recursion

}

Here mouse interface is given to increase a level of recursion by clicking mouse button and also to decrease a level of recursion by doing the same holding the shift on the keyboard.

void glutKeyboardFunc(myKey)

refers to the keyboard callback function. The function to callback is defined as

void myKey(unsigned char key, int x, int y)

```
{
 if (c == 't')
                        exit
if (c == 's')
//STATEMENTS and repeat when finished
}
```

Here keyboard interface is given to quit, the user can quit by pressing ‘q’ and to see next example of the implementation, the user should press ‘n’.

void glutSwapBuffers()

swaps the front and back buffers.

User defined functions are used to color the curves in a standard cycle rainbow manner which becomes very easy for the user to identify the levels of recursion for the curves.

void glutInit(int *argc, charargv)**

I nitializes GLUT< the arguments from main are passed in and can by the application.

void glutCreateWindow(char *title)

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

void glutInitDisplaymode(unsigned int mode)

Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color modell(GLUT_RGB<GLUT_INDEX) and buffering (GLUT_SINGLE<GLUT_DOUBLE).

void glutIntitWindowSize(int width,in heights)

Specifies the initial height and width of the window in pixels

void dlutIntitWindowPosition(int x,int y)

Specifies the initial position of the top-left corner of the window in pixels.

void glViewport(int x,int y,GL sizei width, GL sizei height)

Specifies a width * height viewport in pixels whose lower-left corner is at (x,y) measured from he origin of the window.

void glutMainLoop()

Cause the program to enter an event-processing loop. It should be the statement in main.

void glutPostRedisplay()

Requests that the display callback be executed after the current callback returns.

CHAPTER3

REQUIREMENTS AND SPECIFICATIUON

Hardware Constraints

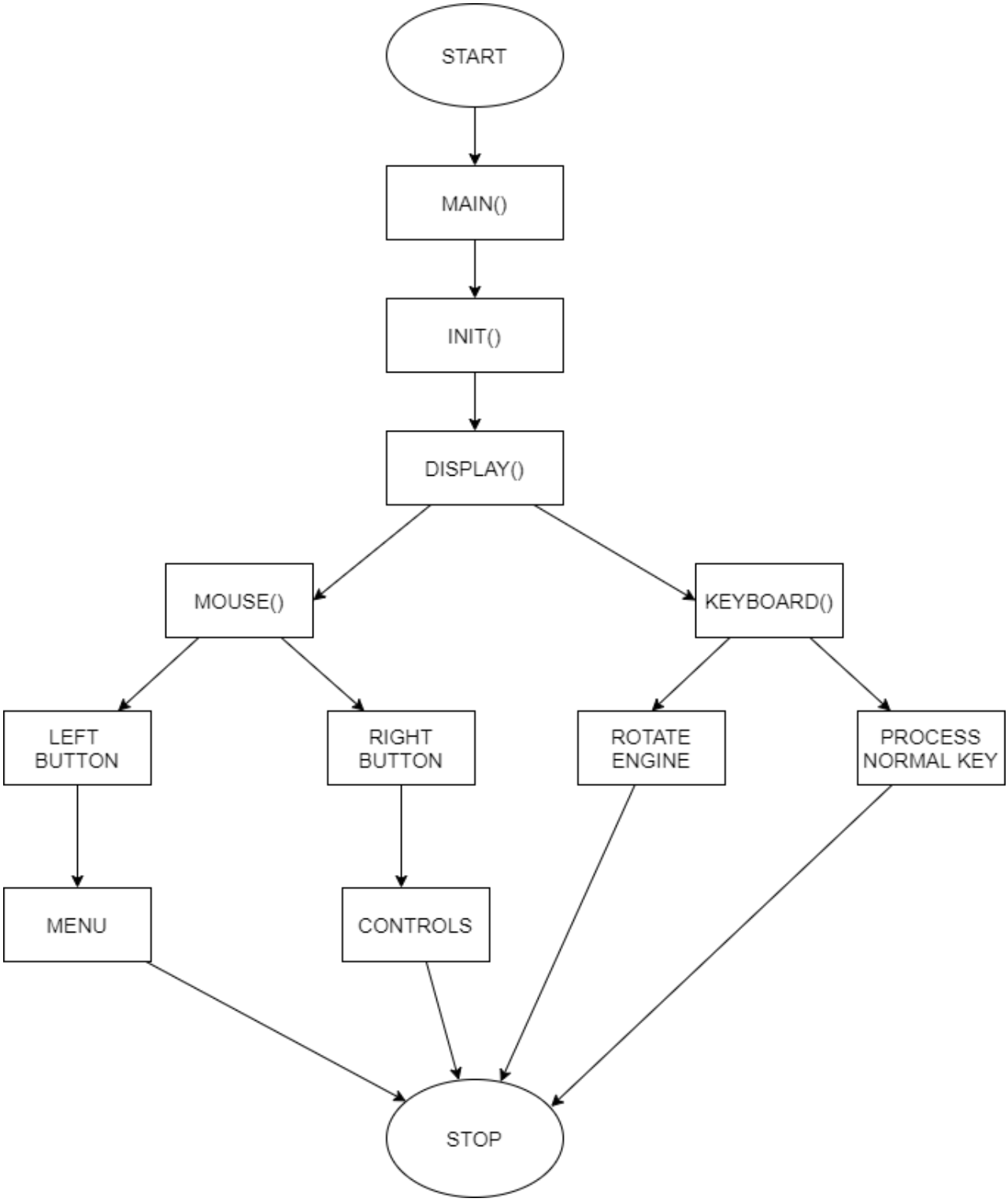
- Processor : Pentium PC
- RAM : 512MB
- Hard disk : 20GB(approx)
- Display : VGA color Monitor

Software Constraints

- Operating System : Windows 98SE/2000/XP/Vista/UBUNTU
- Language : OpenGL
- Compiler : Eclipse/Microsoft Visual Studio 2019

CHAPTER 4

SOFTWARE DESIGN


START
MAIN()
INIT()
DISPLAY()
MOUSE()
KEYBOARD()
LEFT BUTTON
RIGHT BUTTON
ROTATE ENGINE
PROCESS NORMAL KEY
MENU
CONTROLS
STOP

CHAPTER 5

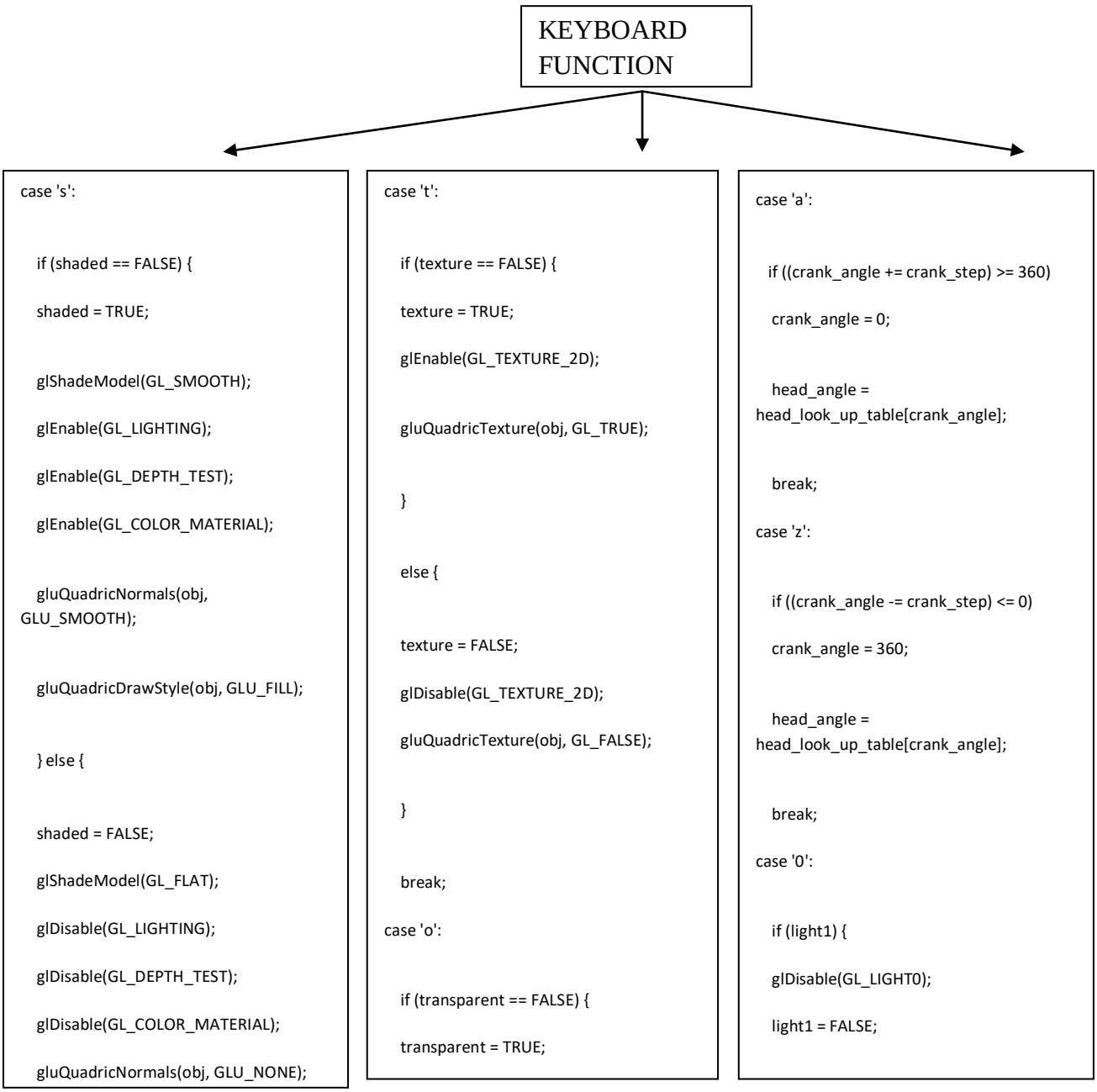
IMPLEMENTATION

5.1 KEYBOARD FUNCTION

5.2 DISPLAY FUNCTION

5.3 RESHAPE FUNCTION

5.1 KEYBOARD FUNCTION

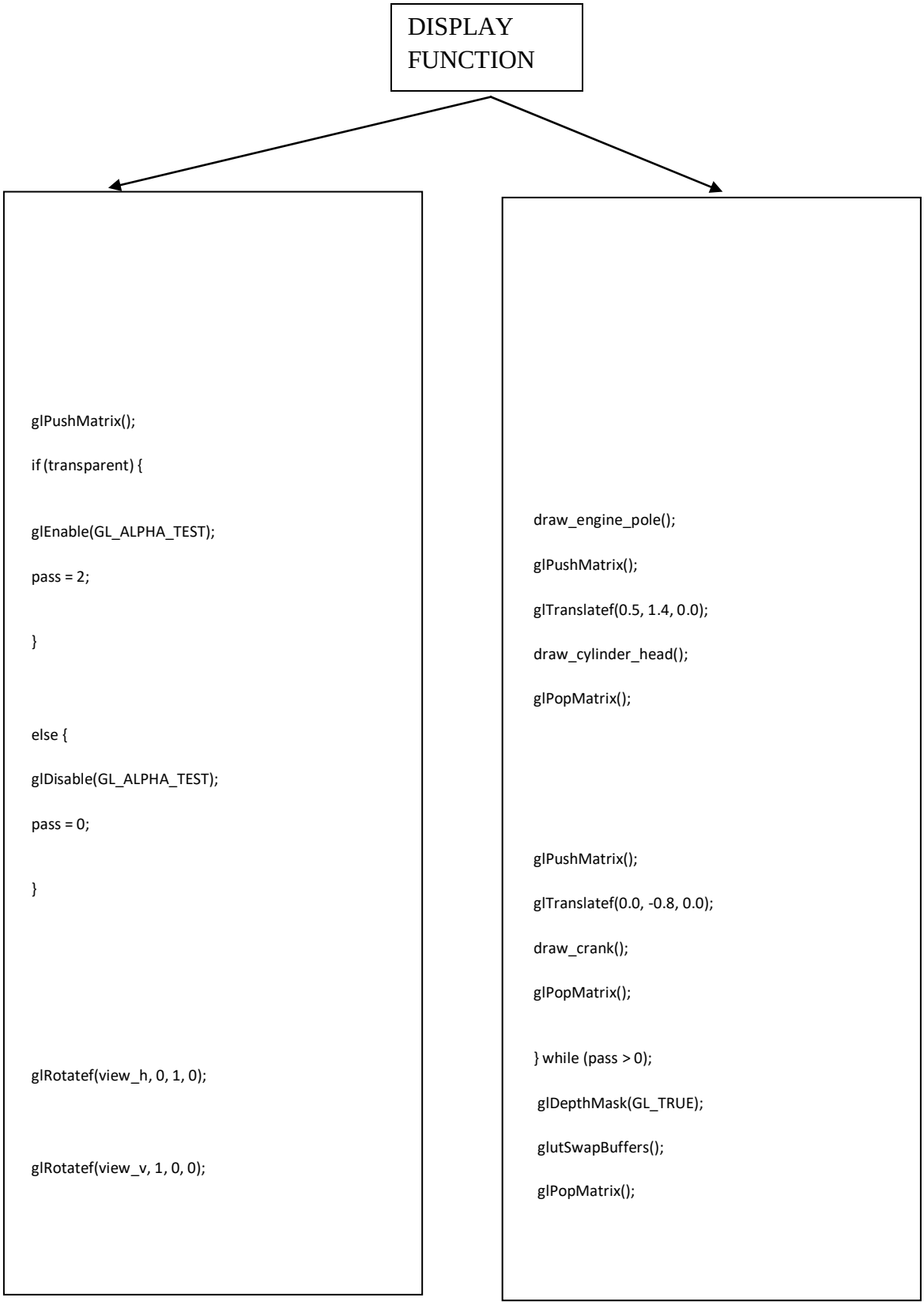

KEYBOARD FUNCTION
case 's':
if (shaded == FALSE) {
shaded = TRUE;
glShadeModel(GL_SMOOTH);
glEnable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST);
glEnable(GL_COLOR_MATERIAL);
gluQuadricNormals(obj, GLU_SMOOTH);
gluQuadricDrawStyle(obj, GLU_FILL);
} else {
shaded = FALSE;
glShadeModel(GL_FLAT);
glDisable(GL_LIGHTING);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

KEYBOARD FUNC (contd)

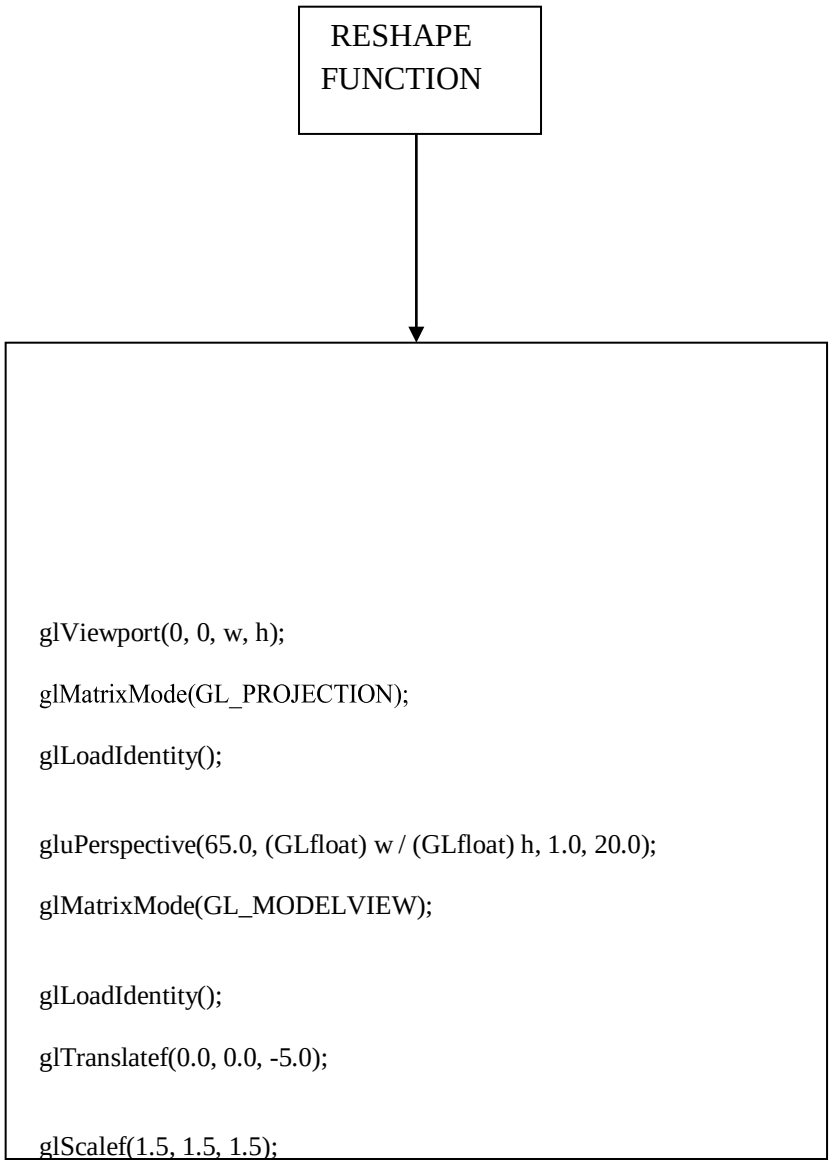
```
case '1':

   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':
```

```
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:
```

5.2 DISPLAY FUNCTION:



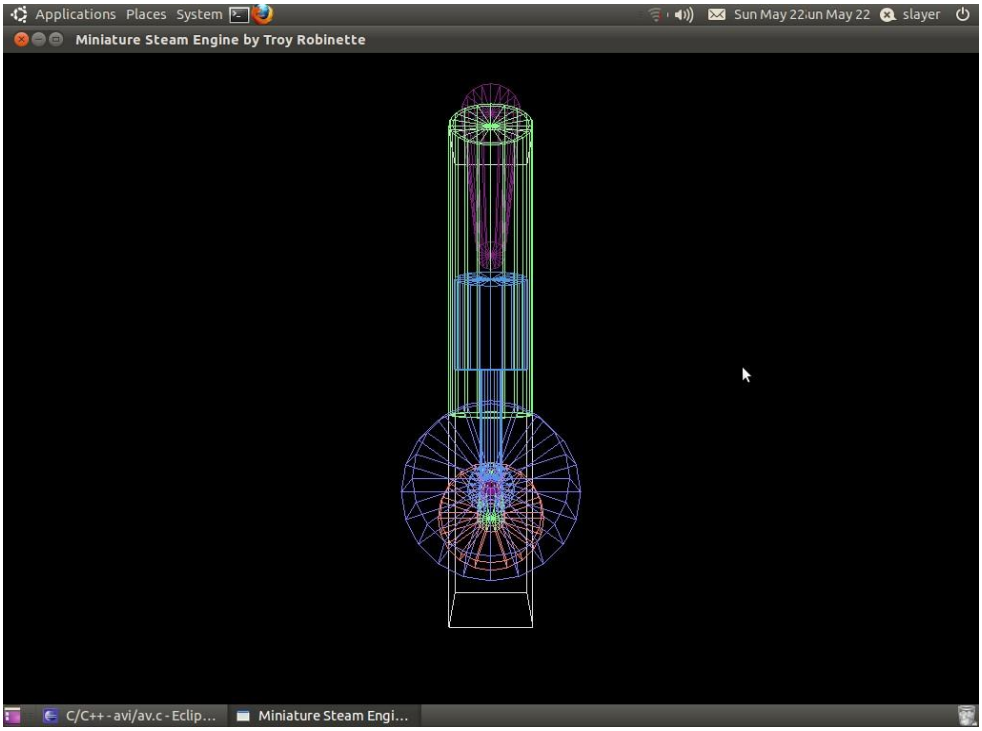
5.3 REPHASE FUNCTION



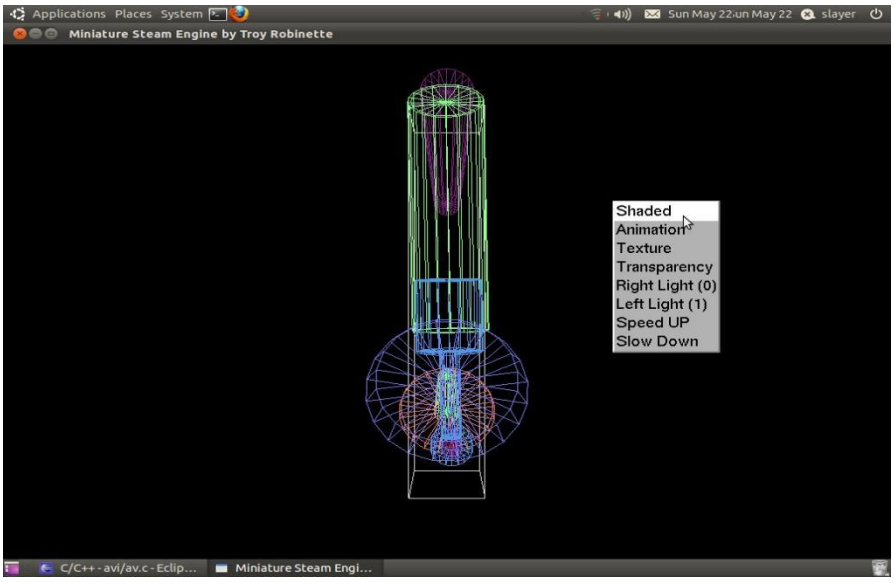
CHAPTER 6

SNAPSHOTS

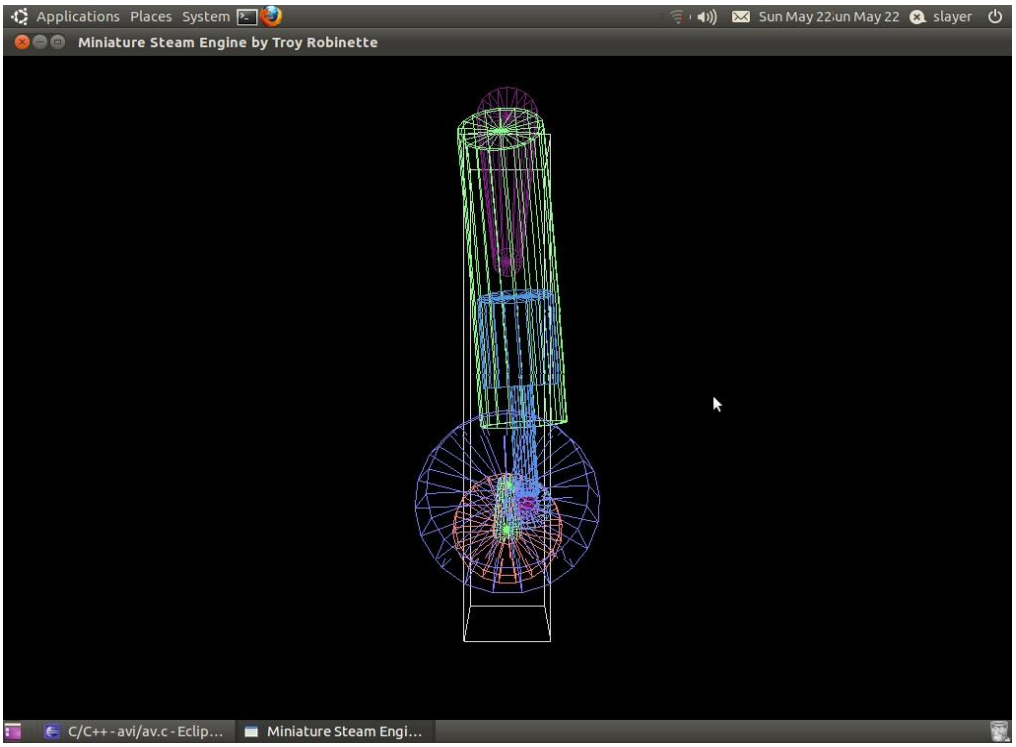
6.1 Initial View



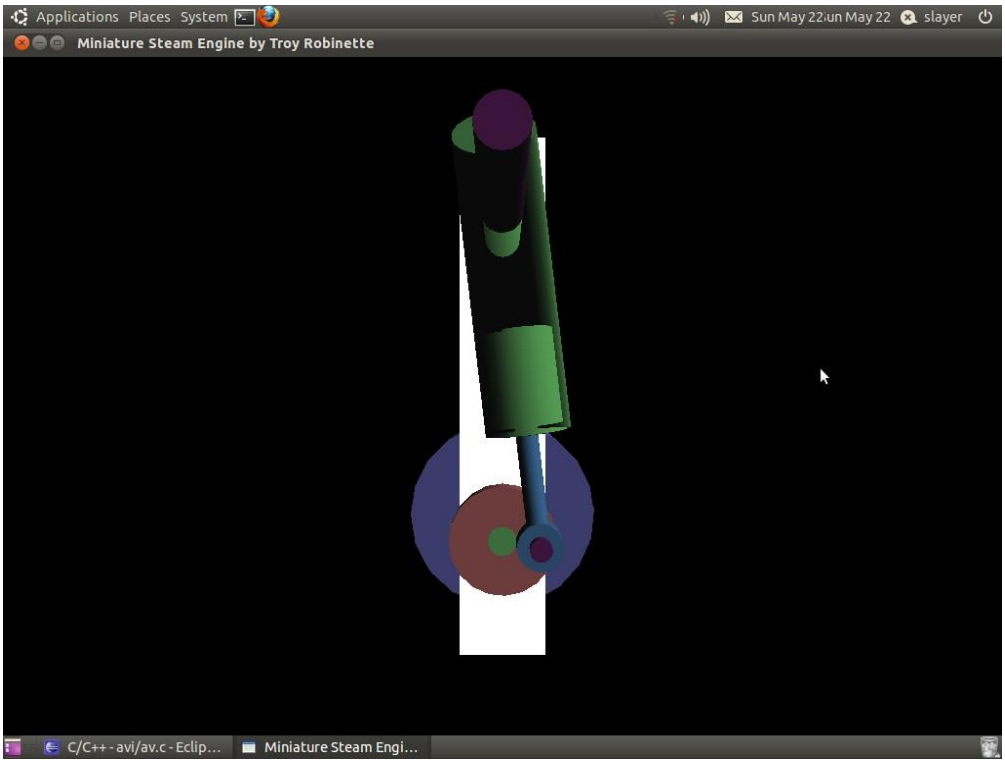
6.2 Menu Associated with Right mouse



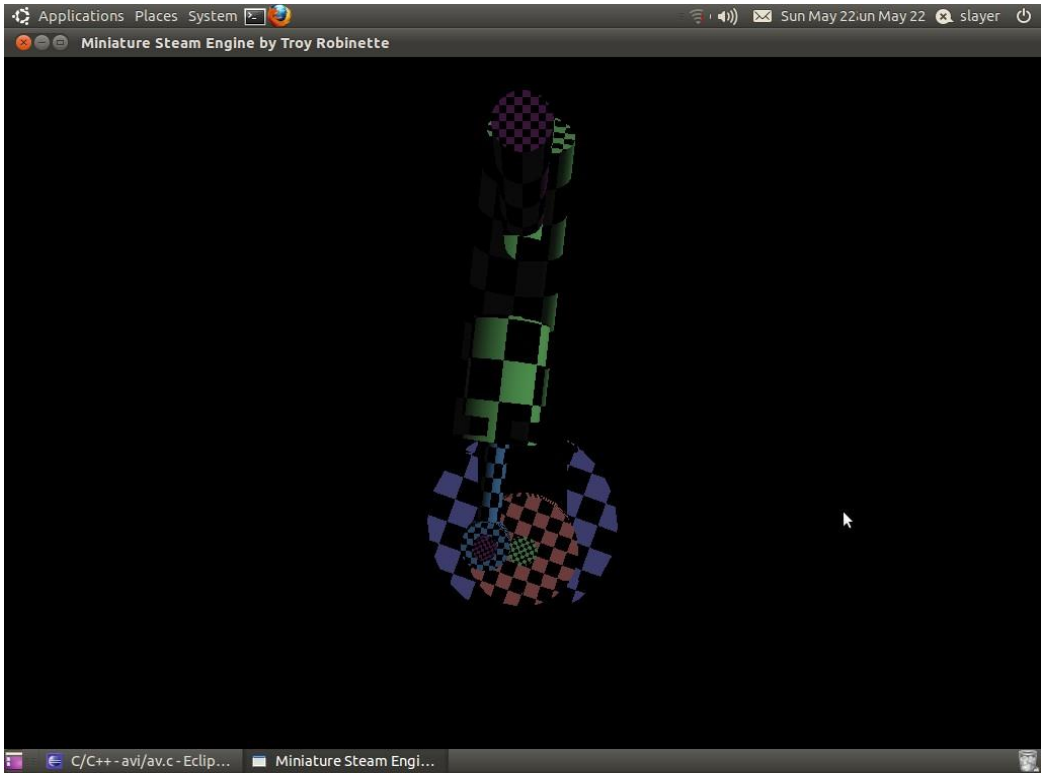
6.3 Rotating Steam engine



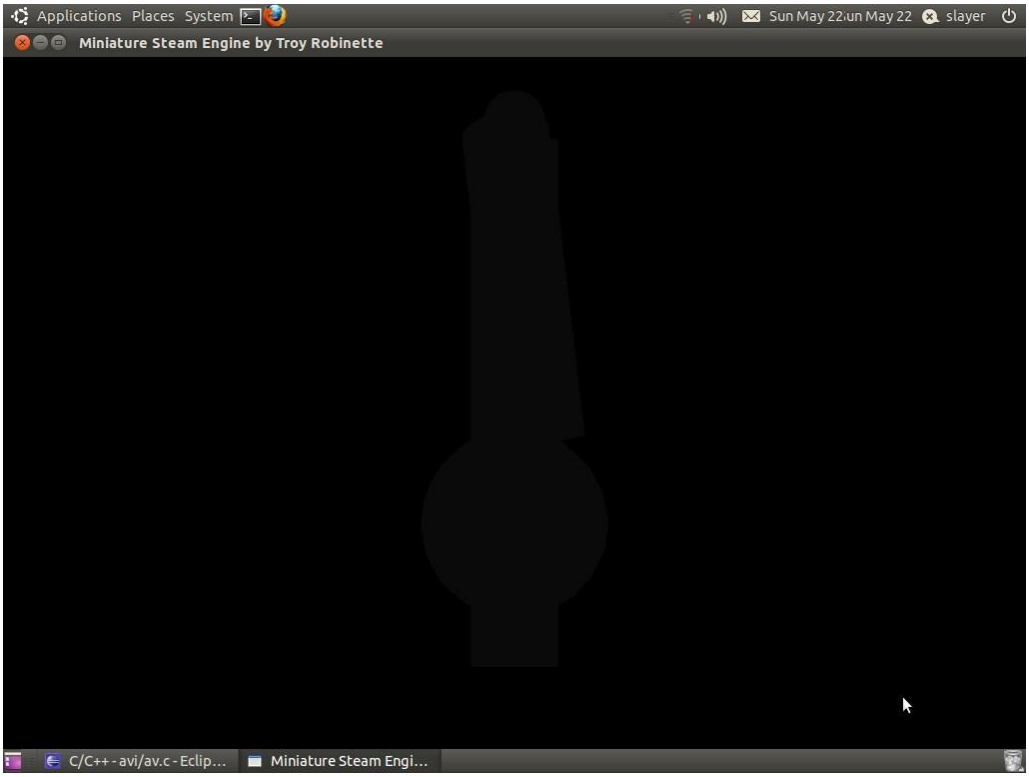
6.4 Shading



6.5 Texture



6.6 No Light (Transparency)



Chapter 7

CONCLUSION AND FUTURE SCOPE

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 by Engineers

APPENDIX

```
#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 cranckshaft.. */

#define MAGNITUDE  120

#define PHASE  270.112

#define FREQ_DIV  58

#define ARC_LENGTH  2.7

#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];
/*Identifiers for each Display list*/
GLint list_piston_shaded = 1;
GLint list_piston_texture = 2;
 GLint list_flywheel_shaded = 4;
GLint list_flywheel_texture = 8;

/*Variables used in the creation of glu objects */
GLUquadricObj *obj;
```

```
/*Draw a box by scaling a glut cube of size 1. Also checks the shaded correctly due to the shaded toggle to see which rendering stylke to use. NB Texture doesn't work correctly due to the cube being scaled. */

void

mBox(GLdouble x, GLdouble y, GLdouble z)

{

glPushMatrix();

glScalef(x, y, z);

if (shaded)

glutSolidCube (1);

glPopMatrix ();

}

/*Draws a cylinder using glu function, drawing flat disc's at each end, to give the appearance of it begin solid*/

void

myCylinder(GLUquadricObj * object,GLdouble outerradius, GLdouble innerRadius, GLdouble innerRadius, Global length)

{

glPushMatrix();

gluCylinder(object, outerRadius, outerRadius, length, 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();
}
/* Draw 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);
mmyCylinder(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();
}
```

```
/*Draw 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();
}
/*Draw the cylinder head at the appropriate angle, doing the necessary translation 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);
glTranslate(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 appropriate display list of a piston doing the necessary rotations before hand. */
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, 0.5, 1.0);
glTranslatef(90, 0.0, 1.0, 1.0);
myCylinder(obj, 0.3, 0.08, 0.12);
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_position_texture);
else
  glcallList(list_piston_shaded);
} else
draw_piston();
glPopMatrix();
}
/*Draws the complete crank. Piston also gets drawn through the crank bell function*/
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
   draw_flywheel();
 glPopMatrix();
glPopMatrix();
}
```

/* Main display routine. Clears the drawing buffer and if transparency is set, display the model twice, 1st time accepting those fragments with a 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 angle by 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_lock_up_table[crank_angle];
```

```
glutPostRedisplay();
}
/* Called wen a key is pressed. Checks if it reconises the key and if so acts on it, updating the screen. */
/* ARGSUED1*/
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;
  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_FALSE);
} else {
   texture = FALSE;
    glDisable(GL_TEXTURE_2D);
     gluQuadricTexture(obj, GL_FALSE);
}
   break;
case '0' :
  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;
}
gluPostDisplay();
}
```

```
/* 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_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. Translate the menu item identifier
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 = '0';

    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;
}
}
}
/* Make 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_table[i] =
MAGNITUDE *atan (
     (ARC_RADIUS * sin(PHASE – k/ FREQ_DIV )) /
     ((ARC_LENGTH – ARC_RADIUS * cos(PHASE – k / FREQ_DIV))));
}
}
/* Initializes texturing, lighting, display lists, and everything else associated 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);

glTextParameterf(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);

glT(exParameterf(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 0 enabled */

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);

glShadedModel(GL_SMOOTH);
```

```
/ * Initialies display lists */
glNewList(list_pistion_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_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_MODEVIEW);
  glLoadIdentity();
   gluTranslatef(0.0, 0.0, -5.0);
   glScalef(1.5, 1.5, 1.5);
}
/*Main program. An interactive model of a miniture steam engine.
   Sets system in Double Buffered mode and initializes all the call-back functions */
int
main(int argc, char **argv)
{
 puts(“Steam Engine\n”);
 puts(“Keyboard 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;
}
```

BIBLIOGRAPHY

1. Interactive Computer Graphics A Top-Down Approach with OpenGL -

Edward Angel, 5 th Edition, Addison-Wesley, 2008

2. The Official Guide to Learning OpenGL, by Jackie Neider, Tom Davis, Mason Woo (THE RED BOOK)

3. OpenGL Super Bible by Richard S. Wright, Jr. and Michael Sweet

4. http://www.opengl.org/ online man pages.

5. And lastly GOOGLE