VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnanasangama, Belagavi, Karnataka

VI Semester
GRAPHICS PACKAGE

SIMULATION OF DIFFERENT TYPES OF PENDULUM

Submitted By

DIVYENDU DUTTA 1BI13CS050

for the academic year 2015-2016

BANGALORE INSTITUTE OF TECHNOLOGY
Department Computer Science & Engineering
K.R. Road, V.V.Puram, Bengaluru-560 004

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnanasangama, Belagavi, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V.Puram, Bengaluru-560 004

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the implementation of **GRAPHICS PACKAGE** entitled "**SIMULATION OF DIFFERENT TYPES OF PENDULUM**" has been successfully completed by **DIVYENDU DUTTA (1BI13CS050)** of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in Computer Science & Engineering of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY during the academic year 2015-2016.

Maya B S	**N Thanuja**	**Dr. S. NANDAGOPALAN**
Assistant Professor	Assistant Professor	Professor and Head
Dept. of CS&E	Dept. of CS&E	Dept. of CS&E
BIT, Bengaluru	BIT,Bengaluru	BIT,Bengaluru

External Examiner: **1.** **2.**

ACKNOWLEDGEMENT

While presenting this Graphics project on “Simulation of different types of pendulum”, I feel that it is my duty to acknowledge the help rendered to me by various people.

Firstly, I am grateful to my institution, “Bangalore Institute of Technology” for providing me with a congenial atmosphere to carry out the project successfully.

I would like to express my heartfelt gratitude to our **Head of Department Dr.Nandagopalan** whose guidance and support was truly invaluable.I would also like to express my gratitude to my teacher **Girija.J**, Assistant Professor for her able guidance and valuable advice at every stage of my project which helped in the successful completion of my project.

DIVYENDU DUTTA

USN : 1BI13CS050

CONTENTS

INTRODUCTION

1.1 Introduction To Computer Graphics

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures.
- There is virtually no area in which graphical displays cannot be used to some advantage.
- Graphics provide a so natural means of communicating with the computer that they have become widespread.
- Interactive graphics is the most important means of producing pictures since the invention of photography and television.
- We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.

1.2 How to Operate

The user can right click on output screen and see the option related to project to work. It also can be brought to one point by right clicking.

The main aim of the project is to see pendulums moving and make them move or pause in motion. And to simulate seconds pendulum. The keybords keys ‘a’ , ‘d’ move camera from left to right whereas keys ‘w’, ‘s’ move camera from top to down.

1.3 Introduction to Open GL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming

interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPEN Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, FORTRAN, Python, Perl and Java and offers complete independence from network protocols and topologies.

1.4 The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: **GL, GLU, and glut**

OpenGL User Interface Library (GLUI) is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT) which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications. It is window- and operating system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management

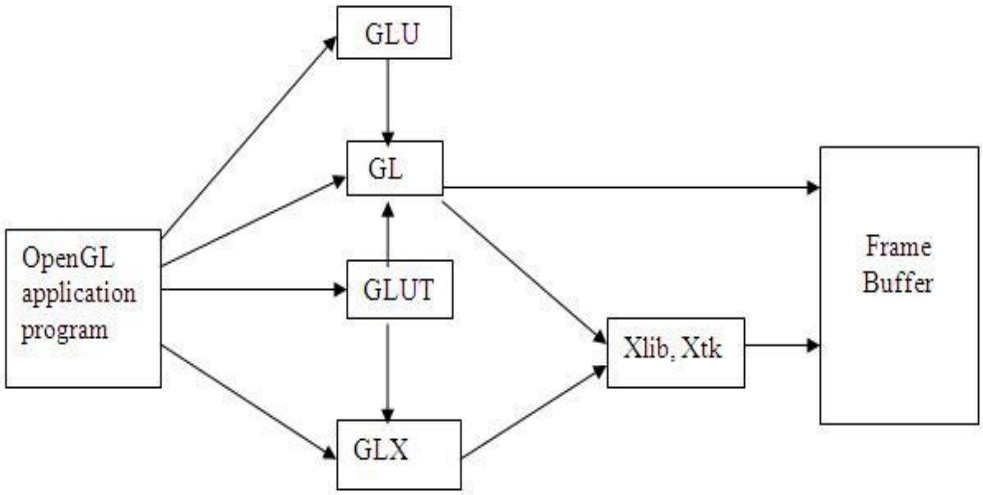

Fig 1.1 OpenGL Interface

The requirements can be broken down into 2 major categories namely hardware and software requirements. The formal specification the minimal hardware facilities expected in a system in which the project has to be run. The latter specifies the essential software needed to build and run the project.

The standard output device is 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 hardware requirements are minimal and software can run wit minimal requirements.

The basic requirements are as enlisted below:

1. Processor: Intel 486/ Pentium processor or a processor with higher specification.
2. Processor speed: 500MHz or above

3. RAM:64MB or above
4. Storage space:2MB or above
5. Monitor resolution: A color monitor with minimum resolution of 640*480.

The editor has been implemented on the OpenGL platform and mainly requires an appropriate version of eclipse compiler to be installed and functional in ubuntu. Though it is implemented in OpenGL, it is very much performed and independent with the restriction, that there is support for the execution of C and C++ files. Text Modes is recommended.

1. An MS-DOS based operating system like window 98, window 2000 or window XP is the platform required to develop the 3D simulation.
2. A C/C++ (integrated with OpenGL) compiler like eclipse is required for compiling the sourse code to make the executable file which can then be directly executed.
3. A built in graphics library; glut.h is required for drawing the layout of the game.

   Glut32.dl for running the application.

Design

2.1 Simple Pendulum

A pendulum is a weight suspended from a pivot so that it can swing freely. When a pendulum is displaced sideways from its resting, equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. When released, the restoring force combined with the pendulum's mass causes it to oscillate about the equilibrium position, swinging back and forth. The time for one complete cycle, a left swing and a right swing, is called the period. The period depends on the length of the pendulum and also to a slight degree on the amplitude, the width of the pendulum's swing.

2.2 Damping Pendulum

Damping pendulum is similar to simple pendulum but, its amplitude keeps reducing every second as per damping coefficient.

A problem that is difficult to solve analytically (but quite easy on the computer) is what happens when a damping term is added to the pendulum equations of motion.
Equation of damped pendulum is shown below:

$$\frac{d^2\theta}{dt^2} + \frac{\alpha d\theta}{mdt} + \frac{g \sin\theta}{l} = 0$$

Table2.1 Various parameters used

Parameter	Symbol	Value
Max. amplitude	θ_{max}	**45°**
		30°
Damping coeff.	**α**	**12**
Mass of bob	**m**	**50kg**
Acc. due to gravity	**g**	**9.81m/s^2**
Length of string	**l**	**2m,3m**

2.3 Runge Kutta 4th order

Runge–Kutta family is generally referred to as "**RK4**", "**classical Runge–Kutta method**" or simply as "***the* Runge–Kutta method**".

Let an initial value problem be specified as follows.

$$\dot{y} = f(t, y), \quad y(t_0) = y_0.$$

Here, *y* is an unknown function (scalar or vector) of time *t* which we would like to approximate; The function *f* and the data t_0, y_0 are given.

Now pick a step-size *h*>0 and define

$$y_{n+1} = y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

for *n* = 0, 1, 2, 3, . . . , using

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1),$$
$$k_3 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_2),$$
$$k_4 = f(t_n + h, y_n + hk_3).$$

Here y_{n+1} is the RK4 approximation of $y(t_{n+1})$, and the next value (y_{n+1}) is determined by the present value (y_n).

2.4 Circular Pendulum

A **circular pendulum** is a weight (or bob) fixed on the end of a string (or rod) suspended from a pivot. Its construction is similar to an ordinary pendulum; however, instead of rocking back and forth, the bob of a conical pendulum moves at a constant speed in a circle with the string (or rod) tracing out a cone.

2.5 Seconds Pendulum

A seconds pendulum is a pendulum whose period is precisely two seconds;
one second for a swing in one direction and one second for the return swing, a frequency of 1/2 Hz. A pendulum is a weight suspended from a pivot so that it can swing freely.

2.6 FUNCTIONS USED

Void glColor3f (float red, float green, float blue);

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. ‘f ’ gives the type that is float. The arguments are in the order RGB (Red, Green, and Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

Void glClearColor (ant red, int green, int blue, int alpha);

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

Void glFlush ();

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

Void glutKeyboardFunc (void (*func)(unsigned char key, int x, int y);

glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed.

Void glutInit (int *argc, char **argv);

GlutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options.glutInit also processes command line options, but the specific options parse are window system dependent.

Void glutReshapeFunc (void (*func) (int width, int height));

GlutReshapeFunc sets the reshape callback for the *current window*. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the *current window* is set to the window that has been reshaped.

If a reshape callback is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered callback), the default reshape callback is used. This default callback will simply

Void glutMainLoop (void);

GlutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never

GlutPostRedisplay ()

GlutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

Complete Source Code of the package:

```
#include<iostream>
#include<string.h>
#include<time.h>
#include<cmath>
#include<GL/glut.h>
using namespace std;
#define pi 3.1415
GLfloat theta=0.0;
GLfloat thetamax_seconds=20*(pi/180),l_seconds=10.0;
GLfloat g=98;
GLint t=0;
GLint val1=0;
static int w;
static int damp_pend1;
static int circ_pend1;
static int mainmenu;
GLint counter1=0;
GLint counter11=0;
GLint count1=0;
GLdouble theta1[3000];
GLdouble theta11[3000];
GLdouble theta2[3000];
GLfloat total_theta[3000];
GLint flag=0;
GLint pflag=0;
GLfloat x2,y2;
GLint hr=0;
GLint min1=0;
GLint sec=0;   // for seconds pendulum time display
GLdouble viewer[]={0.0,-5.0,30.0};
GLdouble viewer1[]={0.0,10.0,5.0};
GLfloat viewer_angle=90*(pi/180);
GLfloat viewer_angle1=0*(pi/180);
GLfloat theta_circular2=0;
GLfloat theta_circular22=0;
double ti=0,tf=20;
double theta01=45*(pi/180);
double theta011=30*(pi/180);
```

```
double v0=2.5; // initial omega(w) value
int N=2500;
double alpha=12,m=50;
int val01;
int val011;

void rk1(GLfloat l){
   double h = (tf - ti)/N;
   double t = ti;
   GLdouble theta, v, k11, k12, k21, k22, k31, k32, k41, k42;

   theta = theta01;
   v = v0;
   int i;
   theta1[0] = theta;
   //vs[0] = v;

   for(i = 1; i<N; i++){
      k11 = h*v;
      k12 = -h*(alpha/m)*v -h*(g/l)*sin(theta);

      k21 = h*(v + k11/2);
      k22 = -h*(alpha/m)*(v + k12/2) - h*(g/l)*sin(theta + k12/2);

      k31 = h*(v + k21/2);
      k32 = -h*(alpha/m)*(v + k22/2)-h*(g/l)*sin(theta + k22/2);

      k41 = h*(v + k31);
      k42 = -h*(alpha/m)*(v + k32) - h*(g/l)*sin(theta + k32);

      theta = theta + (k11 + 2*k21 + 2*k31 + k41)/6;
      v = v + (k12 + 2*k22 + 2*k32 + k42)/6;
      t = t + h;

      theta1[i] = theta;

      //vs[i] = v;
  }
   i--;
      while(theta1[i]<=0.0)
```

```
  {   theta1[i+1]=theta1[i]+0.002;  i++; }
       val01=i;
      // cout<<"val "<<val01;
}


  void timer1(int id)
  {
    if(pflag==0)
    {
       counter1++;
       if(counter1>=val01)
       {
       theta1[counter1]=0;
       }

    }
    glutPostRedisplay();
}
void timer2(int id)
  {
    if(pflag==0)
    {
       theta_circular2++;
       if(theta_circular2>=360*(pi/180))
        theta_circular2=0;
    }
    glutPostRedisplay();
}

  void timer3(int id)
  {
     // cout<<"count 1 in timer3 is "<<count1<<"\n";
    if(pflag==0)
    {
       count1++;
       if(count1>=4*t)
        count1=0;
    }
    glutPostRedisplay();
```

```
}

void displaypen(GLfloat theta,GLfloat l)
{
   // glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

   GLfloat x1=0.0,y1=0.0;
   if(0<=theta<pi/2)
   {
    x2=l*sin(theta);
    y2=-l*cos(theta);

   }
   else if(pi/2<=theta<pi)
   {
      x2=-l*cos(pi-theta);
      y2=l*sin(pi-theta);

    }
    else if(-pi/2<=theta<0)
    {
        x2=l*sin(theta);
        y2=-l*cos(theta);
     }
     else if(-pi<=theta<-pi/2)
     {
        x2=l*cos(pi-abs(theta));
        y2=-l*sin(pi-abs(theta));
      }

    glColor3f(0.0,1.0,0.0);
    glLineWidth(4.0);
    glBegin(GL_LINES);
    glVertex3f(0,0,0.0);
    glVertex3f(x2,y2,0.0);
    glEnd();

   // drawpixel(x2,y2);

    glPushMatrix();
```

```
   GLfloat mat_ambient[]={1.0,1.0,1.0,1.0};
    GLfloat mat_diffuse[]={1.0,1.0,1.0,1.0};
    GLfloat mat_shininess[]={1.0,1.0,1.0,1.0};

    GLfloat lightintensity[]={1.0,1.0,1.0,1.0};
    GLfloat lightposition[]={3.0,0.0,3.0,0.0};

    glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
    glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
    glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

    glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
    glLightfv(GL_LIGHT0,GL_SHININESS,lightintensity);
    glColor3f(1.0,0.0,0.0);
    glTranslatef(x2,y2,0);
    glutSolidSphere(3,50,50);
    glPopMatrix();
}

void display_circular_pen(GLfloat theta,GLfloat l_circular,GLfloat theta_max_circular)
{
     GLfloat x1=0.0,y1=0.0;
     GLfloat h=l_circular*cos(theta_max_circular);
     GLfloat r_circular=l_circular*sin(theta_max_circular);
     GLfloat x3,z3;

        x3=r_circular*cos(theta);
        z3=r_circular*sin(theta);
    glColor3f(0.0,1.0,0.0);
    glLineWidth(4.0);
    glBegin(GL_LINES);
    glVertex3f(0,0,-5.0);
    glVertex3f(x3,-h,z3);
    glEnd();

   // drawpixel(x2,y2);

    glPushMatrix();
    GLfloat mat_ambient[]={1.0,0.0,0.0,1.0};
```

```
    GLfloat mat_diffuse[]={1.0,0.0,0.0,1.0};
    GLfloat mat_shininess[]={1.0,0.0,0.0,1.0};

    GLfloat lightintensity[]={0.7,0.7,0.7,1.0};
    GLfloat lightposition[]={3.0,0.0,10.0,0.0};

    glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
    glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
    glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

    glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,lightintensity);
    glColor3f(1.0,0.0,0.0);
    glTranslatef(x3,-h,z3);
   // glScalef(1.3,1.3,1.3);
    glutSolidSphere(3,50,50);
    glPopMatrix();

}

void display_seconds_pen(GLfloat theta)
{
    // glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

   GLfloat x1=0.0,y1=0.0;
   if(0<=theta<pi/2)
   {
    x2=l_seconds*sin(theta);
    y2=-l_seconds*cos(theta);

   }
   else if(pi/2<=theta<pi)
   {
      x2=-l_seconds*cos(pi-theta);
      y2=l_seconds*sin(pi-theta);

    }
    else if(-pi/2<=theta<0)
    {
```

```
        x2=l_seconds*sin(theta);
         y2=-l_seconds*cos(theta);
      }
      else if(-pi<=theta<-pi/2)
      {
         x2=l_seconds*cos(pi-abs(theta));
         y2=-l_seconds*sin(pi-abs(theta));
       }

    // glPushMatrix();
     //glScalef(2,2,2);
     glColor3f(1.0,1.0,0.0);
     glLineWidth(4.0);
     glBegin(GL_LINES);
     glVertex3f(0,0,-5.0);
     glVertex3f(x2,y2,-5.0);
     glEnd();

    // drawpixel(x2,y2);

     glPushMatrix();
     GLfloat mat_ambient[]={0.0,1.0,0.0,1.0};
     GLfloat mat_diffuse[]={0.0,1.0,0.0,1.0};
     GLfloat mat_shininess[]={0.0,1.0,0.0,1.0};

     GLfloat lightintensity[]={0.7,0.7,0.7,1.0};
     GLfloat lightposition[]={3.0,0.0,8.0,0.0};

     glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
     glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
     glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

     glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
     glLightfv(GL_LIGHT0,GL_DIFFUSE,lightintensity);
     glColor3f(1.0,0.0,0.0);
     glTranslatef(x2,y2,-5);
   //  glScalef(2,2,2);
     glutSolidSphere(2,50,50);
     glPopMatrix();
```

```
 }
 void display_time(char *str,GLint x,GLint y,GLint z)
{
    GLint i;
    GLint len=strlen(str);
    glColor3f(1.0,1.0,1.0);
    glRasterPos3i(x,y,z);
    for(i=0;i<len;i++)
     glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,str[i]);
 }

 void display_text(char *str,GLint x,GLint y,GLint z)
{
    GLint i;
    GLint len=strlen(str);

    glColor3f(0.0,0.0,0.0);
    glRasterPos3i(x,y,z);
    for(i=0;i<len;i++)
     glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
 }

void display1(GLfloat l,GLfloat thetamax)
{
   char str[200];
    rk1(l);
   glutTimerFunc(10,timer1,1);
   glClearColor(0.0,0.0,0.0,0.0);
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(viewer[0],viewer[1],viewer[2],0.0,0.0,0.0,0.0,1.0,0.0);

   sprintf(str,"Length of pendulum is %.1f m, amplitude is %.1f
degrees\n",l/10,thetamax*(180/pi));
   display_time(str,-30,20,0);
   sprintf(str,"Damping Pendulum\n");
   display_time(str,-30,15,0);
```

```
  displaypen(theta1[counter1],l);
    glutSwapBuffers();
 }
 void display2(GLfloat l_circular,GLfloat theta_max_circular)
{
      glutTimerFunc(90,timer2,1);

      glClearColor(0.0,1.0,1.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    char str[100];
     cout<<"angles are "<<viewer1[0]<<" "<<viewer1[1]<<" "<<viewer1[1]<<"\n";
    sprintf(str,"Length of pendulum is %.1f m, amplitude is %.1f
degrees\n",l_circular/10,theta_max_circular*(180/pi));
    display_time(str,-30,20,0);
    sprintf(str,"Circular Pendulum\n");
    display_time(str,-30,15,0);

    display_circular_pen(theta_circular2,l_circular,theta_max_circular);
    glutSwapBuffers();
 }

void display3(int count1)
{

    char str_time[50];
    glutTimerFunc(1,timer3,1); //3.99
     glClearColor(1.0,0.0,1.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    if((count1==2*t) || (count1==4*t-1))
    {
       sec++;
       if(sec>=60)
       {
          min1++;
          sec=0;
            if(min1>=60)
            {
              hr++;
              min1=0; sec=0;
```

```
                if(hr>=24)
                {
                   hr=0; min1=0; sec=0;
                }
            }
        }
    }

    sprintf(str_time,"Time %d : %d : %d\n",hr,min1,sec);
    glColor3f(1,1,1);
    display_time(str_time,-3,5,0);
    sprintf(str_time,"Seconds Pendulum..L=1m\n");
    display_time(str_time,-3,10,0);
    display_seconds_pen(total_theta[count1]);
    glutSwapBuffers();
}

void disp()
{
     char str[50];
     glClearColor(0.0,0.0,1.0,0.0);
     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
     glClearColor(1.0,1.0,0.0,0.0);
    sprintf(str,"COMPUTER GRAPHICS AND VISUALIZATION PROJECT\n");
     display_text(str,-43,30,0);
     sprintf(str,"SIMULATION OF DIFFERENT TYPES OF PENDULUM\n");
     display_text(str,-40,20,0);
     sprintf(str,"A3 BATCH\n");
     display_text(str,20,-30,0);
     sprintf(str,"DIVYENDU DUTTA\n");
     display_text(str,20,-35,0);
     sprintf(str,"1BI13CS050\n");
     display_text(str,20,-40,0);
     if(val1==1) display1(30.0,45*(pi/180));
     else if(val1==2)  display11(20.0,30*(pi/180));
     else if(val1==3) display2(20.0,30*(pi/180));
     else if(val1==4) display22(30.0,60*(pi/180));
     else if(val1==5) display3();
    glutSwapBuffers();
```

```
  }
  void keybd(unsigned char key, int x, int y){
    if(key==113) //q to quit
      exit(0);
    else if(key==32)   // space to pause/resume
    {     if(pflag==0)
             pflag=1;
          else
          pflag=0;
    }
    if(key=='a') {
      viewer_angle=viewer_angle*(180/pi);
      viewer_angle+=90.0;
      viewer_angle=viewer_angle*(pi/180);
      viewer[0]=30.0*cos(viewer_angle);
      viewer[2]=30.0*sin(viewer_angle);
      }
    if(key=='d') {
       viewer_angle=viewer_angle*(180/pi);
       viewer_angle-=90.0;
       viewer_angle=viewer_angle*(pi/180);
       viewer[0]=30.0*cos(viewer_angle);
       viewer[2]=30.0*sin(viewer_angle);
       }

      if((key=='a')||(key=='d'))
       glutPostRedisplay();

 }
void calc_theta(GLfloat l)
{
    int thetamax_second=thetamax_seconds*100000;
    float thetamax_second1=(float)thetamax_second/100000;
    while(theta<=thetamax_second1)
    {
        theta=thetamax_seconds*sin(sqrt(g/l)*t);
        if(theta<0.0)
        theta=-theta;
        theta2[t]=theta;
```

```
t++;
    }

    GLint i=0,j;
    for(j=t-1;j>=0;j--)
   { total_theta[i]=theta2[j]; i++; }
    for(j=0;j<t;j++)
    { total_theta[i]=-theta2[j]; i++; }
    for(j=t-1;j>=0;j--)
    { total_theta[i]=-theta2[j];  i++; }
    for(j=0;j<t;j++)
   {  total_theta[i]=theta2[j]; i++; }

 }
 void reshape(int w, int h)
{
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
    if(w<=h)
   glOrtho(-50.0,50.0,-50.0*((GLfloat)h/(GLfloat)w),50.0*((GLfloat)h/(GLfloat)w),-
100.0,100.0);
     else
     glOrtho(-50.0*((GLfloat)w/(GLfloat)h),50.0*((GLfloat)w/(GLfloat)h),-50.0,50.0,-
100.0,100.0);

   glMatrixMode(GL_MODELVIEW);
   glutPostRedisplay();
}

void menu(int);
void createMenu()
{
    damp_pend1=glutCreateMenu(menu);
    glutAddMenuEntry("Q=45 & L=3m",1);
    glutAddMenuEntry("Q=30 & L=2m",2);

    circ_pend1=glutCreateMenu(menu);
    glutAddMenuEntry("Q=30 & L=2m",3);
    glutAddMenuEntry("Q=60 & L=3m",4);
```

```
mainmenu=glutCreateMenu(menu);
    glutAddSubMenu("Damped pendulum",damp_pend1);
    glutAddSubMenu("Circular pendulum",circ_pend1);
    glutAddMenuEntry("Seconds pendulum",5);

    glutAddMenuEntry("EXIT",0);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
 }
 void menu(int value)
{
    if(value==0)
    {
      glutDestroyWindow(w);
      exit(0);
    }
    val1=value;
    glutPostRedisplay();
 }

 int main(int argc,char *argv[])
 {
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE|GLUT_DEPTH);

    glutInitWindowPosition(50,50);
    glutInitWindowSize(800,800);
    w=glutCreateWindow("pendulum");
    calc_theta(10.0);
    glutReshapeFunc(reshape);
    createMenu();
    glutDisplayFunc(disp);
    glutKeyboardFunc(keybd);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glShadeModel(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glutMainLoop();
    return 0;
 }
```

SNAPSHOTS OF RESULTS

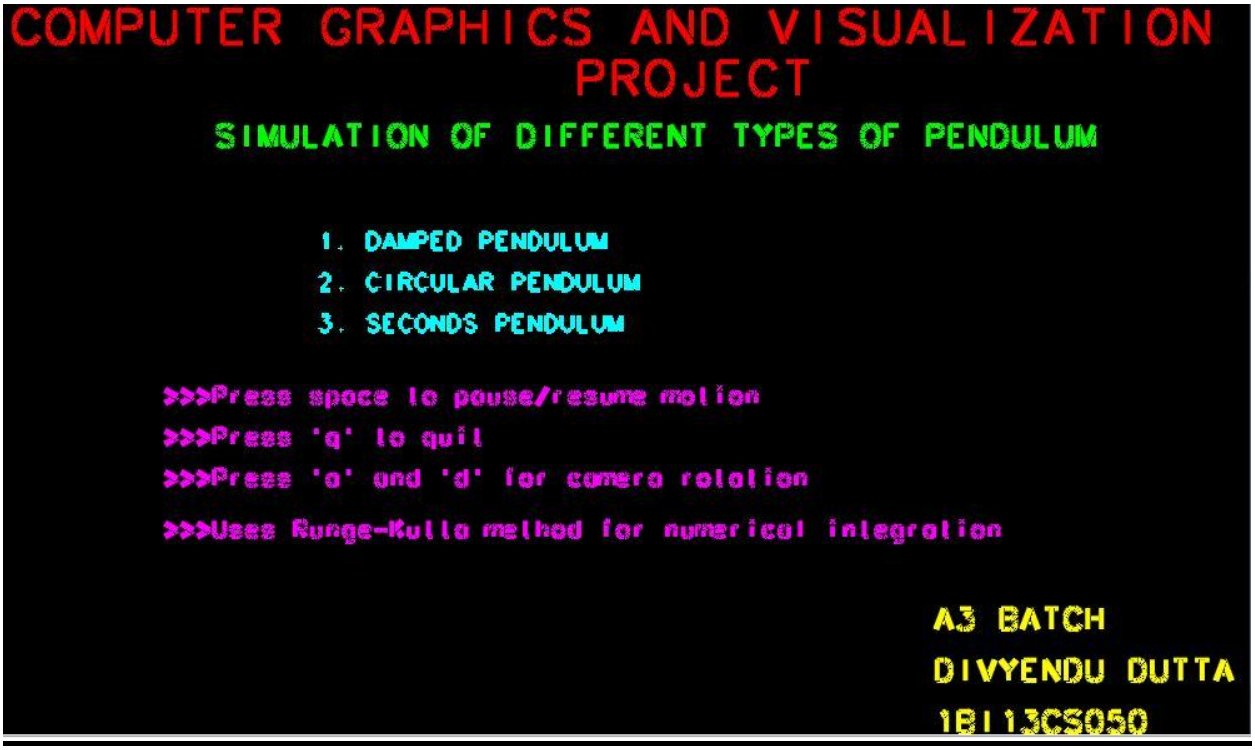

Fig 4.1 START SCREEN

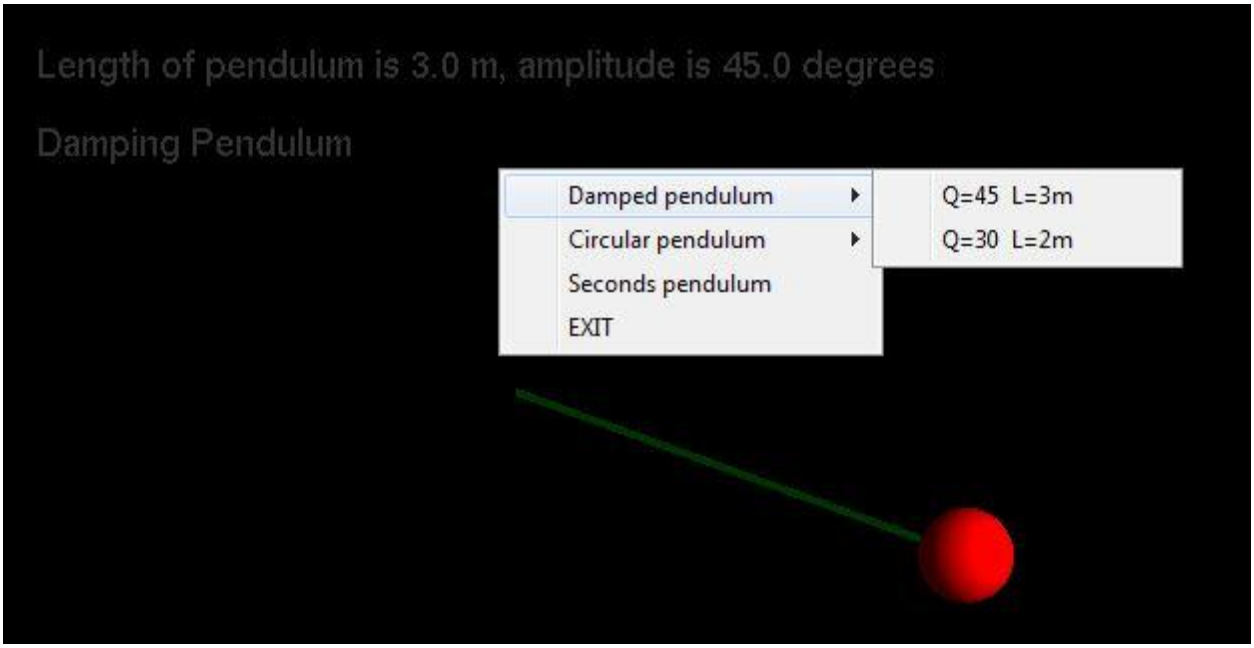

Fig 4.2 DAMPING PENDULUM

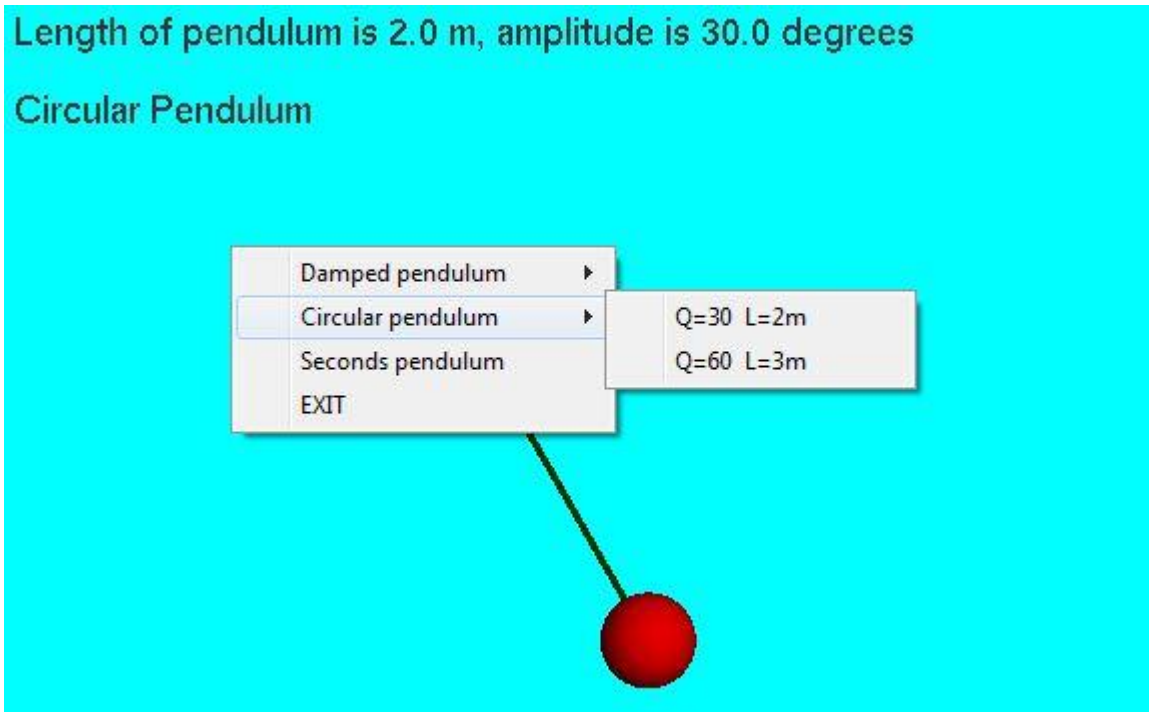

Fig 4.3 CIRCULAR PENDULUM

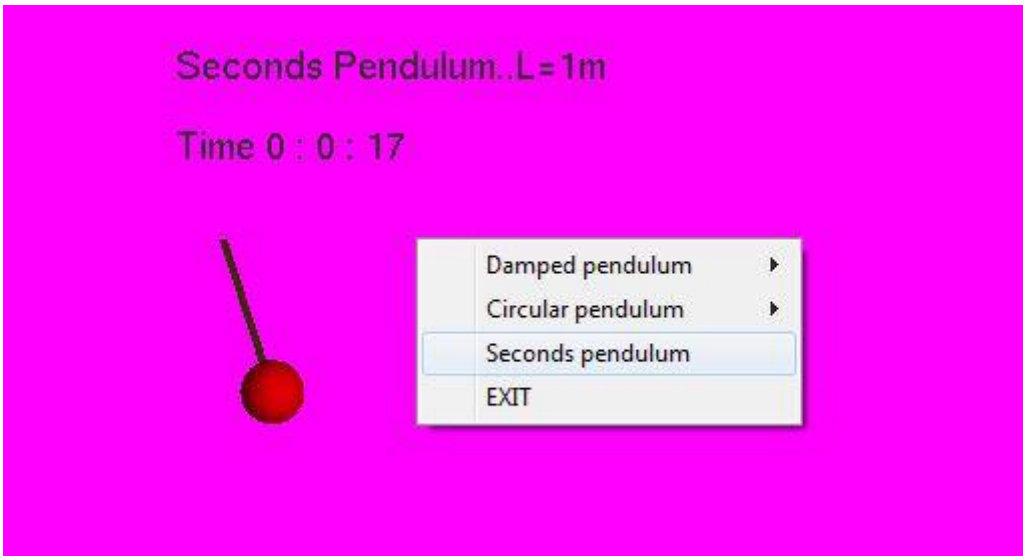

Fig 4.4 SECONDS PENDULUM

Conclusion

The development of computer graphics has made computer to interact with and better for understanding and interpreting many types of data. Developments computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry.

I started with modest aim with no prior experience in any programming project as this, but ended up in learning many things, fine tuning the programming skill and getting into the real world of software development with an exposure to corporate environment. During the development of any software of significant utility, I faced the trade-off between speed of execution and amount of memory consumed. This is simple interactive application. It has open source code and no security features has been included. The user is free to alter the code for feature enhancement. Checking and verification of all possible types of function are taken care. Care was taken to avoid dugs. Bugs may be reported to creator as the need may be. So, I conclude on note that we are looking forward to develop more such project with an appetite to learn more in computer graphics.

BIBLIOGRAPHY

- Interactive computer Graphics: A Top-Down Approach with OpenGL-Edward Angel, 5th edition, Addison-Wesllesley,2008

- OpenGL Redbook and Bluebook for reference.

- www.opengl.org for OpenGL tutorials.

- www.stackoverflow.com