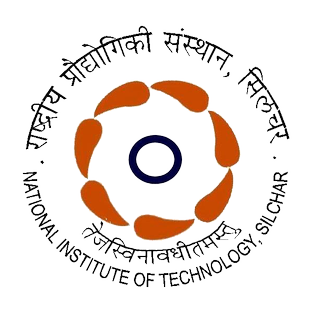
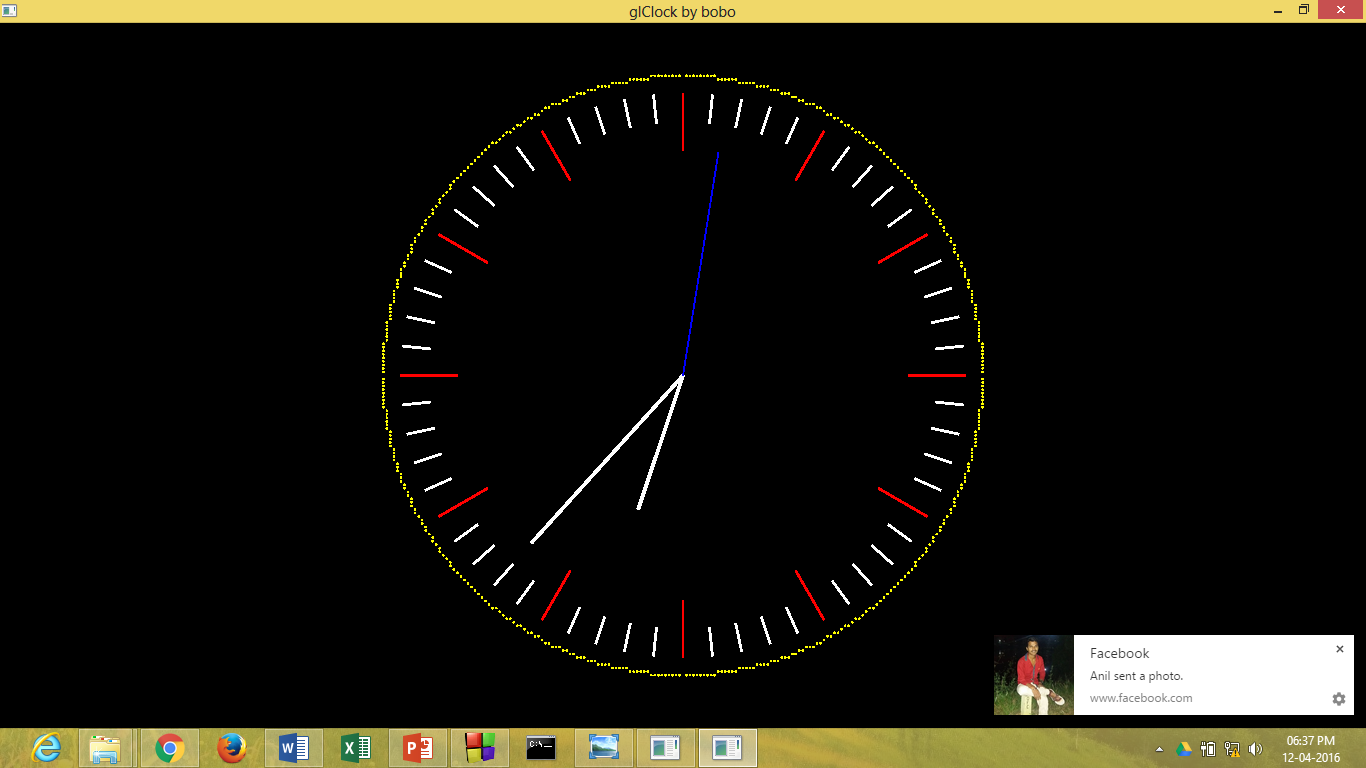
**::Computer GraphicsProject(CS\_1206)::**



|  |  |
| --- | --- |
| **Students Names::** | **Registration Number::** |
| *Srikar Krishna* | *14-1-5-091* |
| *Rathlavath Seetharam* | *14-1-5-097* |
| *Shashank Mallick* | *14-1-5-093* |

**:Project Name: Analog Clock Using Open GL:**



**CONTENTS::**

* **INTRODUCTION**
* **REQUIREMENT ANALYSIS**
* **DESIGN**
* **IMPLEMENTATION**
* **TESTING**
* **USER MANUAL**

**::INTRODUCTION::**

The aim of this project is to implement an application package of Computer graphics using OpenGL. Here we represent the concepts displaying Analog clock in OpenGL.

**::ANALOG CLOCK::**

A clock or watch is called "analog" when it has moving hands and hours marked from 1 to 12 to show you the time

Analog clocks usually indicate time using angles. The most common [clock face](http://en.wikipedia.org/wiki/Clock_face)uses a fixed numbered dial or dials and moving hand or hands. It usually has a circular scale of 12[hours,](http://en.wikipedia.org/wiki/Hour) which can also serve as a scale of 60[minutes,](http://en.wikipedia.org/wiki/Minute) and 60[seconds](http://en.wikipedia.org/wiki/Seconds)if the clock has a second hand. Many other styles and designs have been used throughout the years, including dials divided into 6, 8, 10, and 24 hours.

* The background color of the clock is black.
* Each solid cube is representing seconds.
* It contains 3 needles one for hour, one for minute, and one for seconds.
* Large needle represents minute, small needle represent hour and thin needle represent seconds.
* There is a proper delay between needle moments and timings.
* Needle moves from 1 to 12 in clockwise direction.

The package requires simple in-built functions found in <GL/glut.h>libraryThis header file, in addition to the usual header files is needed for the working of the project. For running the program, any basic PC running compatible version of Microsoft Visual Studio is sufficient.

This program should use memory as less as possible, dynamic memory allocation is preferable to accomplish this task. There is no error message produced in this program.

**:: SOFTWARE REQUIREMENT::**

The project requires access to the OpenGL graphics library functions. Some of these library functions are contained in unique to C++ library header files such as

1.GL/glut.h

2.math.h

3.windows.h

4.time.h

To write, compile and link the program, a suitable C integrated development environment is required Code Blocks may be suitable in this regard.

Software requirement are:

1. Window Operating System & Code Blocks

2. Linux

**::DESIGN::**

 Name of the function: print\_screen

Input parameters: void print\_screen( )

Function description: This function displays digital clock

Name of the function: TimerFunction

Input parameter: void glutTimerFunc (33, TimerFunction, 1)

Function description : It register the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc.

Name the function: ChangeSize

Input parameter : Void ChangeSize(GLsizei w, GLsizei h)

Function Description: It sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. The width and height parameters of the callback specify new size in pixels.

**::Function Used::**

|  |  |
| --- | --- |
| Function Name | Function Description |
| glutPostRedisplay | Marks the current window as needing to be redisplayed. |
| glutReshapeFunc | Sets the reshape call back for the Current window |
| glutDisplayfunc | Sets the display call back for the current window |
| glutTimerfunc | Registers a timer call back to be triggered in specified numbers of milliseconds. |
| glLoadIdentity | Replaces the current matrix with the identity matrix. |

**::IMPLEMENTATION::**

#include <gl/glut.h>

#include <gl/gl.h>

#include <math.h>

#include <time.h>

#include <sys/timeb.h>

const float clockR = 80.0f,

clockVol = 100.0f,

angle1min = M\_PI / 30.0f,

minStart = 0.9f,

minEnd = 1.0f,

stepStart = 0.8f,

stepEnd = 1.0f;

float angleHour = 0,

angleMin = 0,

angleSec = 0;

int xc = 0, yc = 0.5;

float r=85;

void newLine(float rStart, float rEnd, float angle){

float c = cos(angle), s = sin(angle);

glVertex2f( clockR\*rStart\*c, clockR\*rStart\*s);

glVertex2f( clockR\*rEnd\*c, clockR\*rEnd\*s);

}

void plot\_point(void)

{

int x=0,y=r;

float pk=(5.0/4.0)-r;

int k;

while(x < y)

{

x = x + 1;

if(pk < 0)

pk = pk + 2\*x+1;

else

{

y = y - 1;

pk = pk + 2\*(x - y) + 1;

}

glBegin(GL\_POINTS);

glColor3f(1.0f, 1.0f, 0.0f);

glVertex2i(xc+x, yc+y);

glVertex2i(xc+x, yc-y);

glVertex2i(xc+y, yc+x);

glVertex2i(xc+y, yc-x);

glVertex2i(xc-x, yc-y);

glVertex2i(xc-y, yc-x);

glVertex2i(xc-x, yc+y);

glVertex2i(xc-y, yc+x);

glEnd();

}

glFlush();

}

void print\_screen(void){

int i;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0f, 0.0f, 0.0f);

glLineWidth(2.0f);

glEnable(GL\_LINE\_SMOOTH);

glEnable(GL\_POINT\_SMOOTH);

glEnable(GL\_POLYGON\_SMOOTH);

plot\_point();

glBegin(GL\_LINES);

for(i=0; i<60; i++){

if(i%5){

if(i%5 == 1)

glColor3f(1.0f, 1.0f, 1.0f);

newLine(minStart, minEnd, i\*angle1min);

}else{

glColor3f(1.0f, 0.0f, 0.0f);

newLine(stepStart, stepEnd, i\*angle1min);

}

}

glEnd();

glLineWidth(3.0f);

glBegin(GL\_LINES);

newLine(0.0f, 0.5f, -angleHour+M\_PI/2);

newLine(0.0f, 0.8f, -angleMin+M\_PI/2);

glEnd();

glLineWidth(1.0f);

glColor3f(0.0f, 0.0f, 1.0f);

glBegin(GL\_LINES);

newLine(0.0f, 0.8f, -angleSec+M\_PI/2);

glEnd();

glutSwapBuffers();

}

void SetupRC(void){

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

}

void ChangeSize(GLsizei w, GLsizei h){

GLfloat aspectRatio;

if(h == 0)

h = 1;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

aspectRatio = (GLfloat)w / (GLfloat)h;

if (w <= h)

glOrtho (-clockVol, clockVol, -clockVol / aspectRatio, clockVol / aspectRatio, 1.0, -1.0);

else

glOrtho (-clockVol \* aspectRatio, clockVol \* aspectRatio, -clockVol, clockVol, 1.0, -1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void TimerFunction(int value){

struct timeb tb;

time\_t tim=time(0);

struct tm\* t;

t=localtime(&tim);

ftime(&tb);

angleSec = (float)(t->tm\_sec+ (float)tb.millitm/1000.0f)/30.0f \* M\_PI;

angleMin = (float)(t->tm\_min)/30.0f \* M\_PI + angleSec/60.0f;

angleHour = (float)(t->tm\_hour > 12 ? t->tm\_hour-12 : t->tm\_hour)/6.0f \* M\_PI+

angleMin/12.0f;

glutPostRedisplay();

glutTimerFunc(33,TimerFunction, 1);

}

int main(int argc, char\* argv[]){

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

glutCreateWindow("glClock by bobo");

glutDisplayFunc(print\_screen);

glutReshapeFunc(ChangeSize);

glutTimerFunc(33, TimerFunction, 1);

SetupRC();

glutMainLoop();

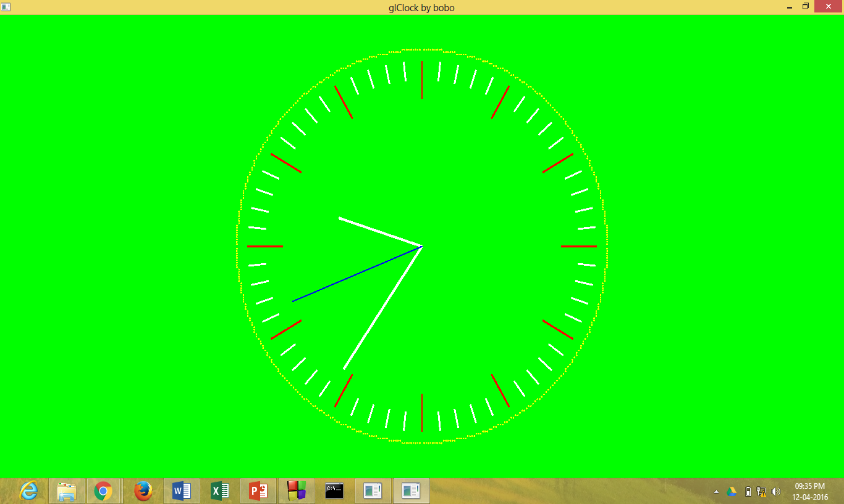
return 0;

}

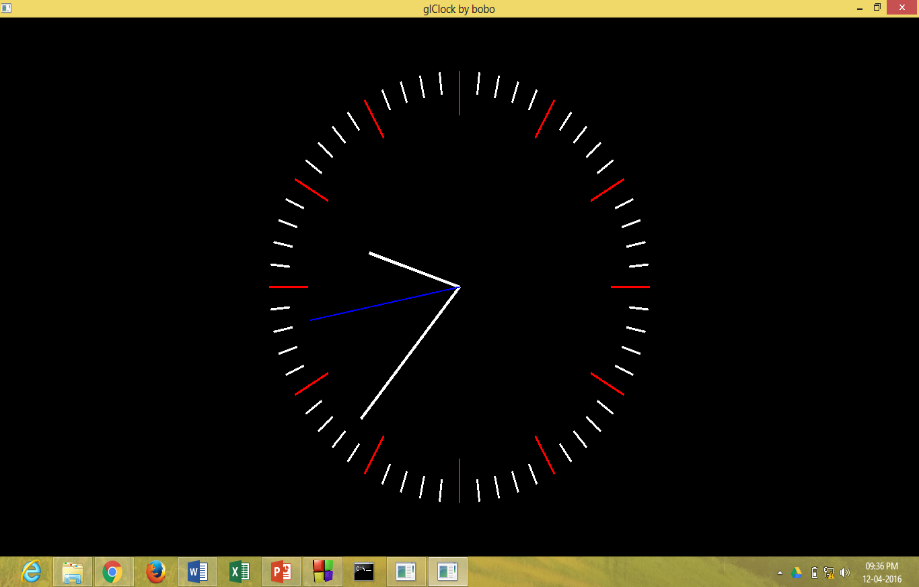
**::TESTING::**

Testing process of this code is very simple. When we execute the program it displays the current system time.

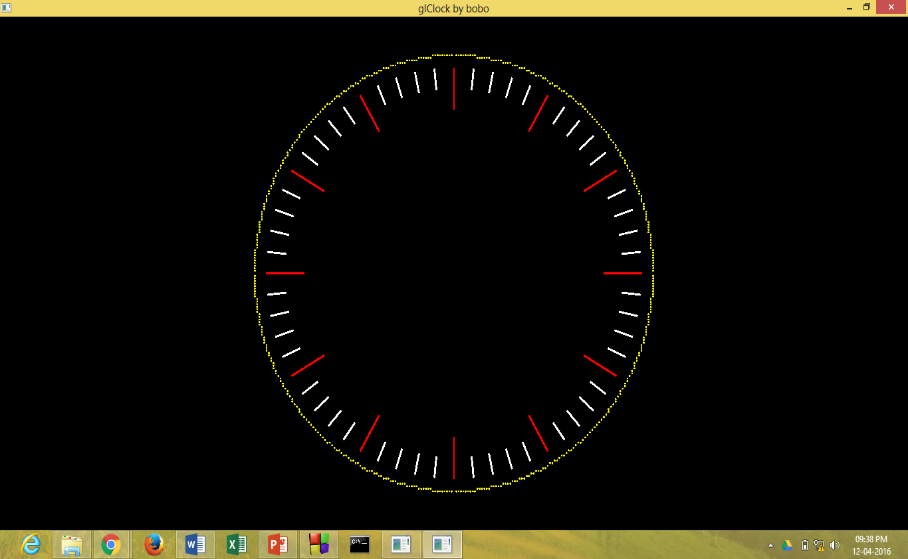
1.Changing the background color to green.



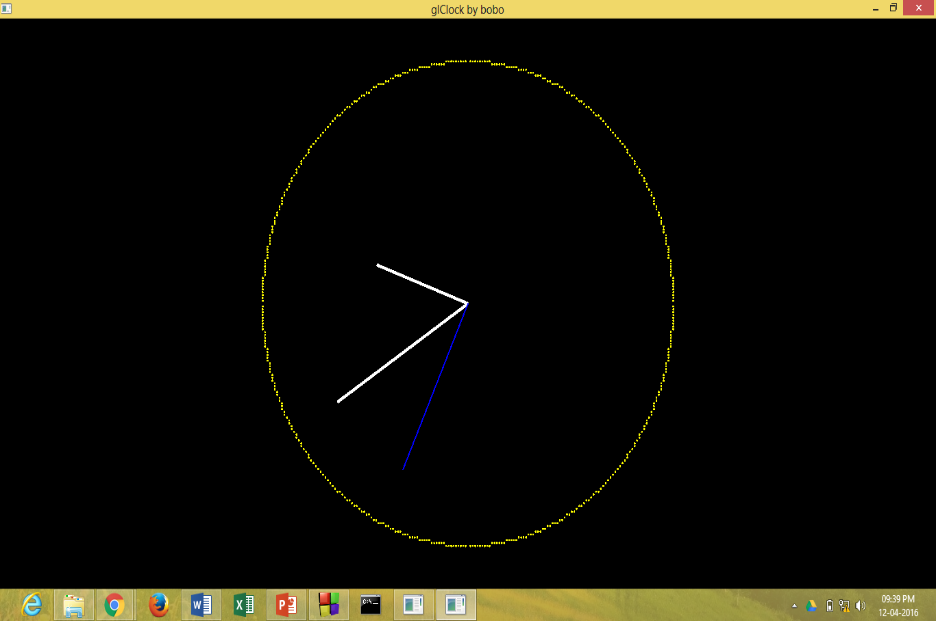
2.Removing the circle around the clock .



3. Removing hour, minutes & seconds hand .



4.Removing the indicator marks.



**::CONCLUSION::**

This project shows the graphical representation of a working analog clock. The objective of this program is to implement simple and basic functions of OpenGL.

Computer graphics plays a major role in today’s world where visualization takes the upper hand as compared to textual interaction. This is largely true as we can see user interface becoming more and more attractive all thanks to major leaps in the fields of computer graphics. The project is implemented using graphics OpenGL package provided by C.