1. Introduction:

1.1Introduction

Nature, in the broadest sense, is the natural, physical, or material world or universe. "Nature" can refer to the phenomena of the physical world, and also to life in general. The study of nature is a large, if not the only, part of science. Although humans are part of nature, human activity is often understood as a separate category from other natural phenomena.

Within the various uses of the word today, "nature" often refers to geology and wildlife. Nature can refer to the general realm of living plants and animals, and in some cases to the processes associated with inanimate objects–the way that particular types of things exist and change of their own accord, such as the weather and geology of the Earth. It is often taken to mean the "natural environment" or wilderness–wild animals, rocks, forest, and in general those things that have not been substantially altered by human intervention, or which persist despite human intervention. For example, manufactured objects and human interaction generally are not considered part of nature, unless qualified as, for example, "human nature" or "the whole of nature". This more traditional concept of natural things which can still be found today implies a distinction between the natural and the artificial, with the artificial being understood as that which has been brought into being by a human consciousness or a human mind. Depending on the particular context, the term "natural" might also be distinguished from the unnatural or the supernatural.

In this project we try to illustrate using open GL to simulate the sunrise as part of the nature and showing the effect using open GL and its feature's.

1.2 Motivation

A few years ago, a nature-conservationist friend of mine was birdwatching on the shores of a lake in East of INDIA, when he saw an otter. He had seen them many times before, but the experience always thrilled him. In his excitement he flagged down a passing car and asked its occupants if they had ever seen an otter, and would they like to? Alas, they did not share his enthusiasm; they had indeed seen otters before and thought them nothing to write home about. They drove on, leaving my friend to return, deflated, to his abandoned telescope, from whose field of view the otter had departed.

1.3 Outline of the Project Report

The outcome of the project is that after it has been developed it humans will realize the wonderful beauty of nature and they will think before taking some action related to destroy nature which are drastically increase now a days. As a result of that earth will be safe ahead many years from global warming.

2. Problem Definition and Requirements

2.1 Problem Definition

Nature rejuvenation using OpenGL functions.

2.2 Requirement Specification

- This software requires RAM that cannot be less than 520 MB.
- The processor cannot be less than 2 GHz speed.
- Hard disk of 20GB or more disk space.
- Operating System can be Linux , Windows XP or later
- Language tool that is to be used is OpenGL.
- Compiler that can be used is GNU/C++ compiler.

2.3 Functional Requirements

- Display the 2-D simulated model of nature.
- Displayed nature should have scenery and sun being the main should move.
- It should show the effect of shading technique used for visual interaction for e.g. it should differentiate between day and night.
- It must provide facilities to change the viewer positions appropriately based on the need of the user.

2.4 System Requirements

- Any GPU that is compatible with OpenGL 3.2. (Integrated graphic cards Intel HD 4000 or above).

3. Design and Implementation

3.1User Interface Design

Keyboard events:

- Press left and right navigation key to movement if sun.

3.2 Various features

- Easy to Understand.
- No complicated mouse or Keyboard events are used.

4. SOURCE CODE :

```
#include<iostream>
#include<stdlib.h>
#ifdef __APPLE__
#include<openGL/openGL.h>
#include<GLUT/glut.h>
#else
#include<GL/glut.h>
#endif

using namespace std;

float ballX = -1.0f;
float ballY = 0.2f;
float ballZ = -1.2f;
float colR=3.0;
float colG=1.5;
float colB=1.0;
float bgColR=0.0;
float bgColG=0.0;
float bgColB=0.0;

static int flag=1;
float x=0.0,y=0.0,x1=5.0;

void update()
{
```

```
x+=0.01;
x1-=0.02;
if(x>6)
{
x=-6;
x1=4;
}
}

void drawBall(void)
{
glColor3f(colR,colG,colB); //set ball colour
glTranslatef(ballX,ballY,ballZ); //moving it toward the screen a bit on creation
glutSolidSphere (0.05, 30, 30); //create ball.


}
void drawAv(void)
 {
glBegin(GL_POLYGON);                 // polygon for the hill
glColor3f(0.0,1.0,0.0);
glVertex3f(-0.9,0.0,-1.0);
glVertex3f(-0.4,0.3,-1.0);
glVertex3f(-0.1,0.0,-1.0);
glEnd();

glBegin(GL_POLYGON);                 // polygon for the hill
glColor3f(0.0,1.0,0.0);
```

```
glVertex3f(-0.2,0.0,-1.0);
glVertex3f(0.1,0.3,-1.0);
glVertex3f(0.3,0.0,-1.0);
glEnd();
glBegin(GL_POLYGON);            // polygon for the hill
glColor3f(0.0,1.0,0.0);
glVertex3f(0.2,0.0,-1.0);
glVertex3f(0.5,0.4,-1.0);
glVertex3f(0.9,0.0,-1.0);
glEnd();

glBegin(GL_POLYGON);            //polygon for the house roof
glColor3f(1.0,1.0,0.0);
glVertex3f(0.4,-0.2,-1.0);
glVertex3f(0.5,-0.1,-1.0);
glVertex3f(0.6,-0.2,-1.0);
glEnd();

glBegin(GL_POLYGON);            // polygon for the house window wall red
glColor3f(1.0,0.5,1.0);
glVertex3f(0.6,-0.2,-1.0);
glVertex3f(0.8,-0.2,-1.0);
glVertex3f(0.8,-0.4,-1.0);
glVertex3f(0.6,-0.4,-1.0);
glEnd();

glBegin(GL_POLYGON);            //polygon for the house roof
```

```
glColor3f(.0,1.0,0.0);
glVertex3f(0.5,-0.1,-1.0);
glVertex3f(0.7,-0.1,-1.0);
glVertex3f(0.8,-0.2,-1.0);
glVertex3f(0.6,-0.2,-1.0);
glEnd();
glBegin(GL_POLYGON);            // polygon for the house door
glColor3f(1.0,1.0,0.0);
glVertex3f(0.47,-0.4,-1.0);
glVertex3f(0.47,-0.3,-1.0);
glVertex3f(0.52,-0.3,-1.0);
glVertex3f(0.52,-0.4,-1.0);
glEnd();

glBegin(GL_POLYGON);            // polygon for the house
glColor3f(1.0,0.0,0.0);
glVertex3f(0.4,-0.4,-1.0);
glVertex3f(0.4,-0.2,-1.0);
glVertex3f(0.6,-0.2,-1.0);
glVertex3f(0.6,-0.4,-1.0);
glEnd();

glBegin(GL_POLYGON);            // polygon for road
glColor3f(0.5,0.35,0.05);
glVertex3f(0.9,0.0,-1.0);
glVertex3f(0.0,-0.6,-1.0);
glVertex3f(0.9,-0.9,-1.0);
```

```
glEnd();
}
void drawClouds(void){
glTranslatef(x1,0.8,-5.0);
glBegin(GL_POLYGON);
glColor3f(1.0,1.0,1.0);
glVertex3f(1.0,0.7,-5.0);
glVertex3f(1.5,1.0,-5.0);
glVertex3f(0.7,1.5,-5.0);
glVertex3f(0.0,2.0,-5.0);
glVertex3f(-0.7,1.5,-5.0);
glVertex3f(-1.4,1.6,-5.0);
glVertex3f(-1.7,1.0,-5.0);
glVertex3f(-1.5,0.7,-5.0);
glVertex3f(-1.0,0.5,-5.0);
glEnd();

glTranslatef(x,0.8,-1.2);
glBegin(GL_POLYGON);
glColor3f(1.0,0.5,0.0);
glVertex3f(1.0,0.7,-5.0);
glVertex3f(1.5,1.0,-5.0);
glVertex3f(0.7,1.5,-5.0);
glVertex3f(0.0,2.0,-5.0);
glVertex3f(-0.7,1.5,-5.0);
glVertex3f(-1.4,1.6,-5.0);
glVertex3f(-1.7,1.0,-5.0);
```

```
glVertex3f(-1.5,0.7,-5.0);
glVertex3f(-1.0,0.5,-5.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(1.0,1.7,1.5);
glVertex3f(-2.0,-5.0,-5.0);
glVertex3f(-1.0,-5.5,-5.0);
glVertex3f(2.0,-5.0,-5.0);
glVertex3f(1.0,-5.5,-5.0);
glVertex3f(0.0,-5.5,-5.0);
glVertex3f(-0.5,-5.5,-5.0);
glVertex3f(-0.9,-5.5,-5.0);
glColor3f(1.0,0.0,0.0);
glVertex3f(-0.4,-5.0,-5.0);
glVertex3f(0.0,-3.0,-5.0);
glVertex3f(0.4,-5.0,-5.0);
glEnd();
update();
}
void keyPress(int key, int x, int y)
{
if(key==GLUT_KEY_RIGHT)
ballX -= 0.05f;
if(key==GLUT_KEY_LEFT)
ballX  += 0.05f;
glutPostRedisplay();
```

```
}
void initRendering()
{
glEnable(GL_DEPTH_TEST);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING); //Enable lighting
glEnable(GL_LIGHT0); //Enable light #0
glEnable(GL_LIGHT1); //Enable light #1
glEnable(GL_NORMALIZE); //Automatically normalize normals
//glShadeModel(GL_SMOOTH); //Enable smooth shading
}
//Called when the window is resized
void handleResize(int w, int h) {
//Tell OpenGL how to convert from coordinates to pixel values
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION); //Switch to setting the camera perspective
//Set the camera perspective
glLoadIdentity(); //Reset the camera
gluPerspective(45.0,                //The camera angle
(double)w / (double)h, //The width-to-height ratio
1.0,                   //The near z clipping coordinate
200.0);                //The far z clipping coordinate
}
void drawScene()
{
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glClearColor(bgColR,bgColG,bgColB,0.0);
```

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

//Add ambient light
GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //Color (0.2, 0.2, 0.2)
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

//Add positioned light
GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Color (0.5, 0.5, 0.5)
GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f}; //Positioned at (4, 0, 8)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

//Add directed light
GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f}; //Color (0.5, 0.2, 0.2)
//Coming from the direction (-1, 0.5, 0.5)
GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

//drawing the SUN
glPushMatrix();
drawBall();
glPopMatrix();
//drawing the Mount Avarest
glPushMatrix();
drawAv();
```

```
glPopMatrix();

//drawing the Clouds
glPushMatrix();
drawClouds();
glPopMatrix();

glutSwapBuffers();
}
//float _angle = 30.0f;
void update(int value) {

if(ballX>0.9f)
{
ballX = -0.8f;
ballY = 0.2f;
flag=1;
colR=2.0;
colG=1.50;
colB=1.0;
bgColB=0.0;
}
if(flag)
{
ballX += 0.001f;
ballY +=0.0003f;
colR-=0.001;
//colG+=0.002;
colB+=0.005;


bgColB+=0.001;
if(ballX>0.01)
{
flag=0;


}
}
if (!flag)
{
ballX += 0.001f;
ballY -=0.0003f;
colR+=0.001;
colB-=0.01;
bgColB-=0.001;
if(ballX<-0.3)
{
flag=1;


}
}
glutPostRedisplay(); //Tell GLUT that the display has changed
//Tell GLUT to call update again in 25 milliseconds
glutTimerFunc(25, update, 0);
}
```

```
int main(int argc,char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(400,400);
glutCreateWindow("Sun");
initRendering();
glutDisplayFunc(drawScene);
glutFullScreen();
glutSpecialFunc(keyPress);
glutReshapeFunc(handleResize);
glutTimerFunc(25, update, 0);
glutMainLoop();
}
```

Output Snapshot :

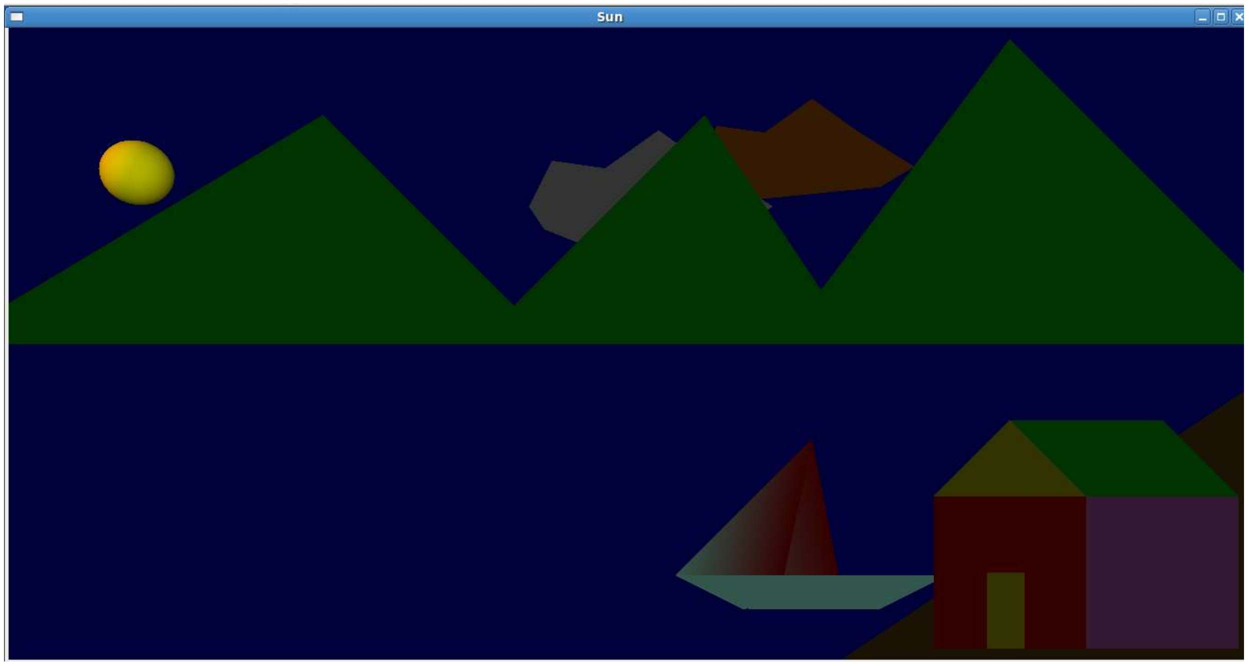

Figure : 1. Sunrise

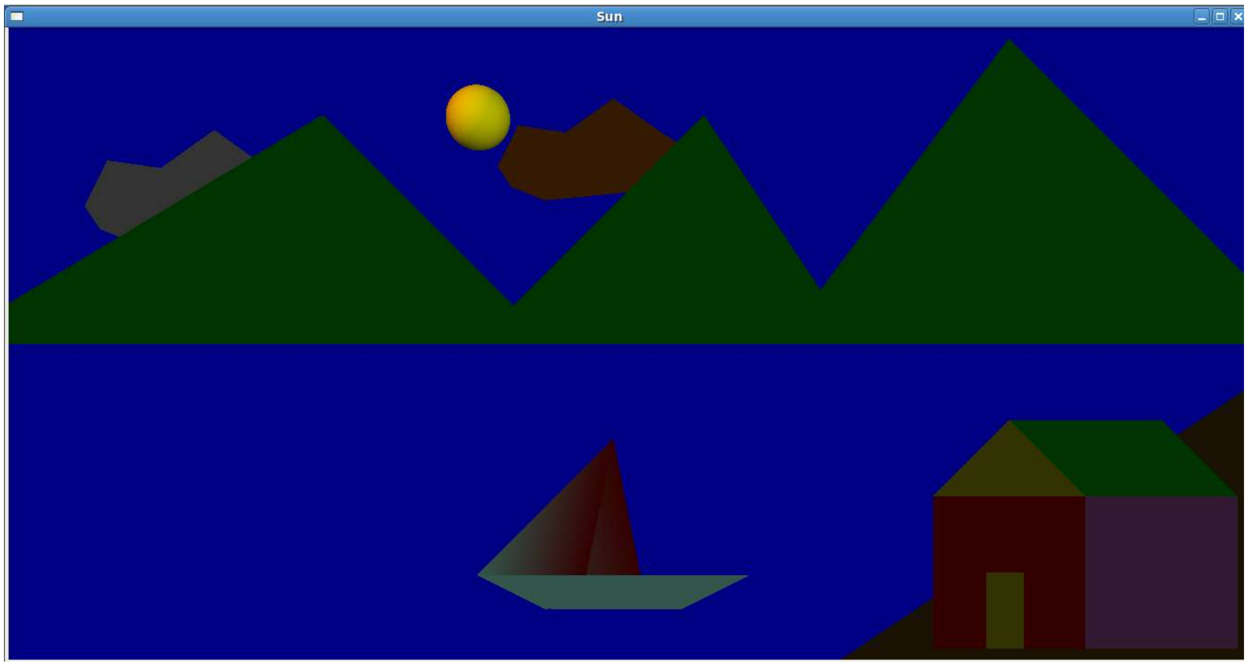

Figure : 2. Mid day

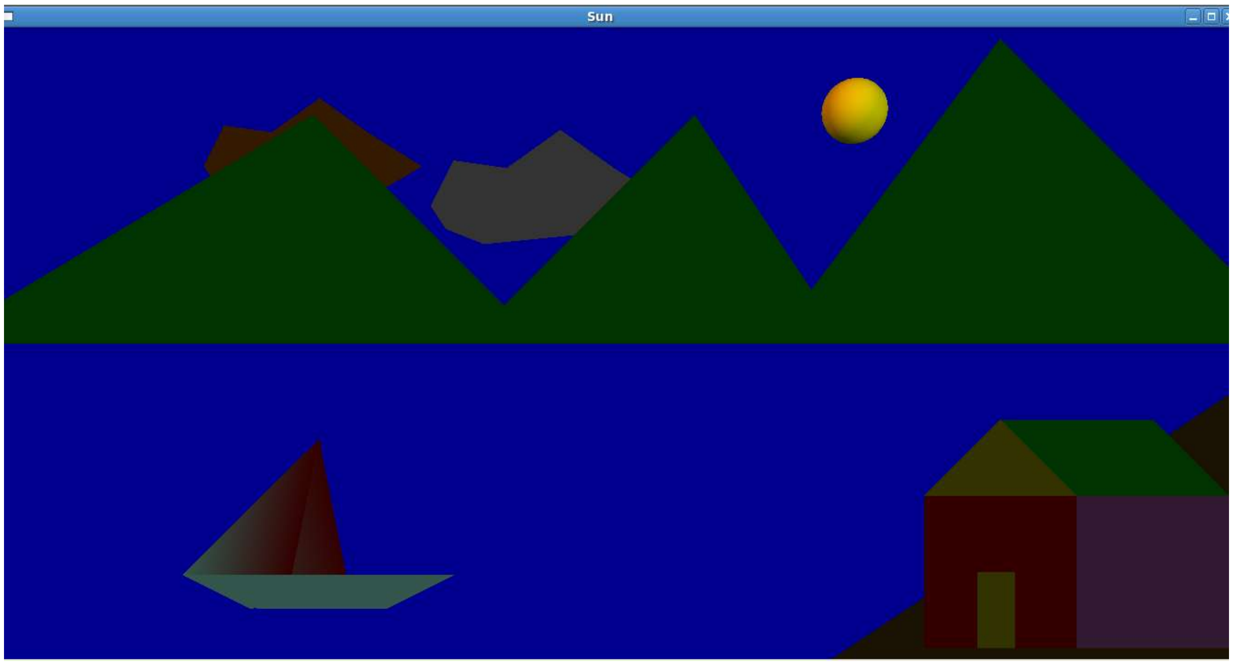

Figure :3. Before Sunset

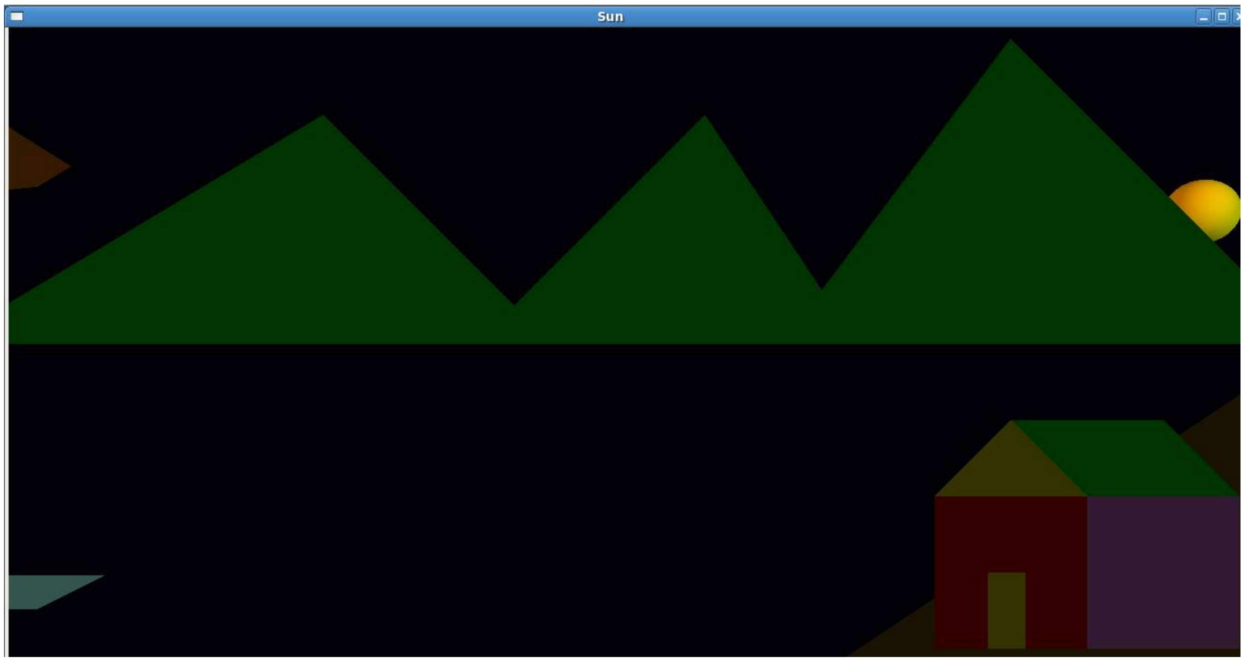

Figure : 4. Sunset

6. CONCLUSION AND FUTURE SCOPE

CONCLUSION

The main intension of the project is to illustrate the use of openGL functions and demonstrate its effect on computer graphics. By using this project many programmer in computer graphic field can easily understand the use of different functions. This project can work in windows/Linux platform with few variances.

FUTURE SCOPE

The developed system can be enhanced further by adding more openGL functions to show verity of effects to the user interface. Code can be optimized for simple user understanding. We can also use events like mouse events for the future use. Finally, this project can be incorporated for more user understanding by showing additional affect on the nature in day to day life.

REFERENCES

1. Edward Angel: Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008.

2. F.S. Hill Jr.: Computer Graphics Using OpenGL, 3rd Edition, PHI, 2009.

3. Donald Hearn and Pauline Baker: Computer Graphics-OpenGL Version, 3rd Edition, Pearson Education, 2004.

4. http://www.tutorialspoint.com/computer_graphics/

5. https://www.courses.psu.edu/art/art201_jxm22/tutorials.html

6. http://lodev.org/cgtutor/

7. http://www.graphics.cornell.edu/online/tutorial/

8. http://www.opengl-tutorial.org/beginners-tutorials/

9. https://opengl/