CHAPTER - 1

INTRODUCTION

1.1 Aim of the Project

The aim of this project is to demonstrate the change in color of the leaves in different seasons and also to develop a 2D Display which supports operations such as Movement, Color change, and also transformation operations like translation, rotation, scaling etc. on objects.

1.2 Overview of the Project

Nature truly displays its color palette throughout the 4 seasons (spring, summer, winter and fall). For years, scientists have worked to understand the changes that happen to trees and shrubs in the. Although we don't know all the details, we do know enough to explain the basics and help enjoy more fully Nature's multicolored display. Three factors influence leaf color-leaf pigments, length of night, and weather, but not quite in the way we think. The timing of color change and leaf fall are primarily regulated by the calendar, that is, the increasing length of night. None of the other environmental influences-temperature, rainfall, food supply, and so on-are as unvarying as the steadily increasing length of night during autumn. As days grow shorter, and nights grow longer and cooler, biochemical processes in the leaf begin to paint the landscape with Nature's palette.

1.3 Outcome of the project

Computer graphics are graphics created using computers and, more generally, there presentation and manipulation of image data by a computer.

CHAPTER – 2

DESIGN AND IMPLEMENTATION

2.1 Algorithm

Step1: Initialize the graphics window and its size using GLUT functions.

Step 2: Register the menu and display call backs in main function.

Step3: When right key is pressed hierarchical menu appears on window.

Step 4: If Spring is selected then leaves change remain green as default color is green.

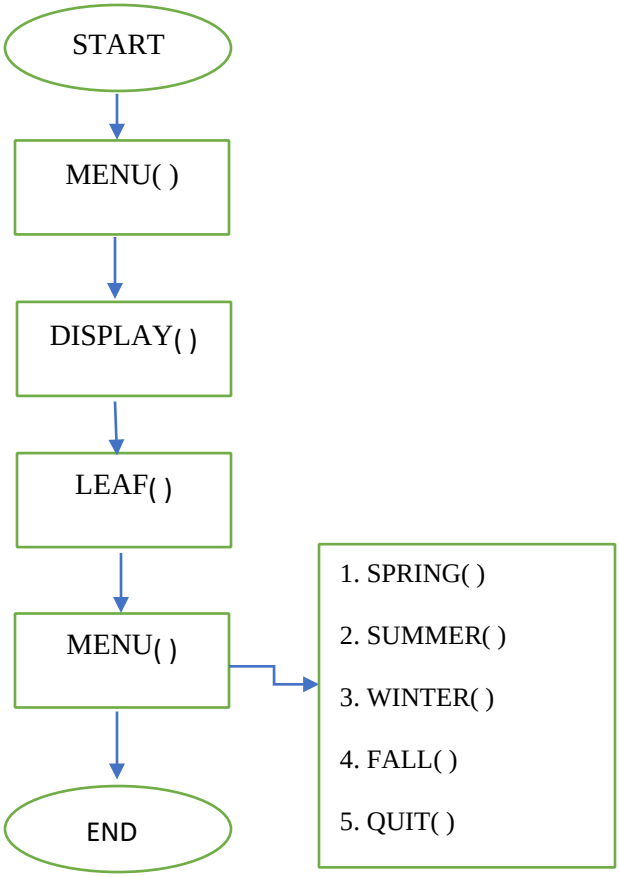
Step 5: If Summer is selected then leaves change to yellow.

Step 6: If Winter is selected then leaves change to brown.

Step 7: If Fall is selected then leaves change to red.

Step 8: If Quit option is selected program is terminated.

2.2 Flow Chart



2.2.1 Data Flow Diagram

2.3 Open API's Used with Description

2.3.1 MODULE DESCRIPTION

The program is divided into four classes:

- **MODULE1: DISPLAY**

This function deals with the displaying of the coordinate axes and number of mentioned leaves. It also deals with the transformation of the current color to the required weather color until the specified weather color. This class also controls the speed at which the leaf falls which depends on its size.

- **MODULE2: LEAF**

This function uses the translate, rotate, and scale function to display the leaf at the mentioned position. It uses various vertex functions to display the stem , leaf and veins of the leaf.

- **MODULE3: MENU**

This class describes the menu to be displayed which provides options to choose or select a particular season. It also provides option to quit or terminate the program.

- **MODULE4:MAIN**

The main function uses all these functions and through the help of menu and mouse interactions lets the users change the color of the leaf. The display stops when the user selects the quit option.

2.3.2 USER DEFINED FUNCTIONS:

- **FUNCTION FOR DISPLAYING THE LEAF:**

display ( )

{

Both the x and y coordinate axes are drawn. The transformation of leaves' color is controlled using a for loop. Speed at which leaf falls is monitored using the the y coordinate value and the size of the leaf

}

- **FUNCTION FOR DRAWING THE LEAF:**

Leaf(int x, int y, float size, float angle)

{

Here the translate, rotate and scaling functions are used to draw the leaf at the mentioned position, also the stem, leaf and the veins are drawn using vertex functions. Parameters passed are the values of x and y coordinates and the size and angle of rotation of the leaf.

}

- **FUNCTION FOR DISPLAYING THE MENU :**

menu(int op)

{

Here, the code for displaying the menu is described. Menu to select different seasons is described also code for quit option is written all using a switch statement and providing different cases for each season and quit option. Parameter passed is the option chosen.

}

Spring( )

{

Specifies weather color values for green color.

}

Summer( )
{

Specifies weather color values for yellow color.

}

Winter( )
{

Specifies weather color values for brown color.

}

Fall( )
{

Specifies weather color values for red color.

}

2.3.3 IN BUILT FUNCTIONS USED:

- **RAND**

Syntax:

rand( );

Description:

Generates a random value, in this case for the x coordinate values.

- **SLEEP**

Syntax:

Sleep( );

Parameters:

Milliseconds for which the program must sleep.

Description:

Specifies time for which program must sleep.

- **TRANSLATE FUNCTION**

Syntax:
glTranslatef(x,y,z); **Parameters:**

x, y and z values

Description:

Accepts axis along which it must translate the object.

- **ROTATE FUNCTION**

Syntax:

glRotatef(angle,x,y,z);

Parameters:

Angle of rotation, x, y and z values.

Description:

Accepts the angle of rotation and x, y and z values.

- **SCALE FUNCTION**

Syntax:

glScalef(size,size,size); **Parameters:**

Sizes of scaling in all three axes.

Description:

Accepts scaling factor for all three axes.

- **POST REDISPLAY**

Syntax:

Void glutPostRedisplay( );

Description:

glutPostRedisplay marks the normal plane of current window as needing to be redisplayed. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.

- **COLOR FUNCTION**

Syntax :

glColor3f(red, green , blue);

Parameters :

red *:* The new red value for the current color.
green*:*The new green value for the current color.
blue*:* The new blue value for the current color.

Description:

This function accepts different colors.

2.4 Source Code

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

using namespace std;

float curColor[3] = { 1, 1, 0 };

float weatherColor[3] = { 0, 1, 0 };
float perc = 1;

int leafCount = 7;
float leafy[100] = { 0, -100, 100, 200, 124, -212, -300 };
float leafx[100] = { 0, 20, 154, -200, 358, -52, 350 };
float leafsize[100] = { 1, 0.50, 0.55, 0.6, 0.7, 0.75, 0.8};
float leafangle[100] = { 0, 25, 50, 90, 251, 158, 0 };
float leafrotatespeed[100] = { 0.15, -0.2, 0.15, 0.1, -0.05, -0.07, 0.15 };

void spring() {
perc = 0;
weatherColor[0] = 0.1;
weatherColor[1] = 0.8;
weatherColor[2] = 0;
}

void summer() {
perc = 0;
weatherColor[0] = 1.0;
weatherColor[1] = 0.8;
weatherColor[2] = 0;
}
void winter(){
perc = 0;
weatherColor[0] = 0.6;
weatherColor[1] = 0.4;
weatherColor[2] = 0.12;
}
void fall() {
perc = 0;
weatherColor[0] = 0.9;
weatherColor[1] = 0.2;
weatherColor[2] = 0;
}

void leaf(int x, int y, float size, float angle) {

glLoadIdentity();
glTranslatef(x, y, 0);
```

```
glRotatef(angle, 0, 0, 1);
glScalef(size, size, size);
   //leaf1-------------------------------------------------------
// Stem
glColor3f(curColor[0]*0.6, curColor[1]*0.6, curColor[2]*0.6);
glBegin(GL_POLYGON);
glVertex2f(- 4, - 0);
glVertex2f(- 4, -50);
glVertex2f(+ 4, -50);
glVertex2f(+ 4, - 0);
glEnd();


// Leaf
glColor3f(curColor[0]*(100-(rand()%20))/100.0,
                    curColor[1]*(100-(rand()%20))/100.0,
                    curColor[2]*(100-(rand()%20))/100.0);
glBegin(GL_POLYGON);
glVertex2f(-  0, +  0);
glVertex2f(- 65, - 10);
glVertex2f(- 45, + 15);
glVertex2f(-140, + 48);
glVertex2f(-120, + 60);
glVertex2f(-155, + 95);
glVertex2f(-107, + 88);
glVertex2f(-115, +120);
glVertex2f(- 30, + 77);
glVertex2f(- 45, +167);
glVertex2f(- 20, +150);
glVertex2f(+  0, +203);
glVertex2f(+ 20, +150);
glVertex2f(+ 45, +167);
glVertex2f(+ 30, + 77);
glVertex2f(+115, +120);
glVertex2f(+107, + 88);
glVertex2f(+155, + 95);
glVertex2f(+120, + 60);
glVertex2f(+140, + 48);
glVertex2f(+ 45, + 15);
glVertex2f(+ 65, - 10);
glVertex2f(+  0, +  0);
glEnd();

//Veins
glColor3f(curColor[0]*0.2, curColor[1]*0.2, curColor[2]*0.2);
glBegin(GL_LINE_STRIP);
glVertex2f(+0, +  0);
glVertex2f(+3, + 50);
glVertex2f(-2, +100);
glVertex2f(+0, +200);
glEnd();
```

```
glBegin(GL_LINE_STRIP);
glVertex2f(+  3, + 50);
glVertex2f(+ 20, + 30);
glVertex2f(+120, + 75);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(+2.5, +30);
glVertex2f(-20, +20);
glVertex2f(-120, +75);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(+20, +120);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(-20, +115);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-1.5, + 80);
glVertex2f(-15, +90);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f( 0, + 65);
glVertex2f(+22,+ 75);
glEnd();

//LEAF2------------------------------------------------------------
//stem
glTranslatef(x+100.0,y+100.0,0.0);
glScalef(1,1,0);
glColor3f(curColor[0]*0.6, curColor[1]*0.6, curColor[2]*0.6);
glBegin(GL_POLYGON);
glVertex2f(- 4, - 0);
glVertex2f(- 4, -50);
glVertex2f(+ 4, -50);
glVertex2f(+ 4, - 0);
glEnd();

   //leaf
glColor3f(curColor[0]*(100-(rand()%20))/100.0,
                  curColor[1]*(100-(rand()%20))/100.0,
                  curColor[2]*(100-(rand()%20))/100.0);
glBegin(GL_POLYGON);
glVertex2f(-  0, +  0);
```

```
glVertex2f(- 25, + 70);
glVertex2f(- 20, +150);
glVertex2f(+  0, +203);
glVertex2f(+ 20, +150);
glVertex2f(+ 30, + 77);
glVertex2f(+  0, +  0);
glEnd();

//Veins
glColor3f(curColor[0]*0.2, curColor[1]*0.2, curColor[2]*0.2);
glBegin(GL_LINE_STRIP);
glVertex2f(+0, +  0);
glVertex2f(+3, + 50);
glVertex2f(-2, +100);
glVertex2f(+0, +200);
glEnd();
glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(+20, +120);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(-20, +115);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-1.5, + 80);
glVertex2f(-20, +90);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f( 0, + 65);
glVertex2f(+22,+ 75);
glEnd();

//leaf3----------------------------------------------------------
    glTranslatef(x+200,y+200,0);
    glScalef(1,1,0);
// Stem
glColor3f(curColor[0]*0.6, curColor[1]*0.6, curColor[2]*0.6);
glBegin(GL_POLYGON);
glVertex2f(- 4, - 0);
glVertex2f(- 4, -50);
glVertex2f(+ 4, -50);
glVertex2f(+ 4, - 0);
glEnd();


// Leaf
glColor3f(curColor[0]*(100-(rand()%20))/100.0,
```

```
                curColor[1]*(100-(rand()%20))/100.0,
                curColor[2]*(100-(rand()%20))/100.0);
glBegin(GL_POLYGON);
glVertex2f(-  0, +  0);
glVertex2f(- 45, + 15);
glVertex2f(- 30, + 77);
glVertex2f(+  0, +203);
glVertex2f(+ 30, + 77);
glVertex2f(+ 45, + 15);
glVertex2f(+  0, +  0);
glEnd();

//Veins
glColor3f(curColor[0]*0.2, curColor[1]*0.2, curColor[2]*0.2);
glBegin(GL_LINE_STRIP);
glVertex2f(+0, +  0);
glVertex2f(+3, + 50);
glVertex2f(-2, +100);
glVertex2f(+0, +200);
glEnd();

   glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(+20, +120);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-2, +100);
glVertex2f(-20, +115);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-1.5, + 80);
glVertex2f(-15, +90);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f( 0, + 65);
glVertex2f(+22,+ 75);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(+4, +45);
glVertex2f(+20, +50);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(-0.01, +45);
glVertex2f(-30, +60);
glEnd();
```

```
//leaf 4----------------------------------------------------------

   glTranslatef(x+400,y+400,0);
   glScalef(2.0,2.0,0);
// Stem
glColor3f(curColor[0]*0.6, curColor[1]*0.6, curColor[2]*0.6);
glBegin(GL_POLYGON);
glVertex2f(- 4, - 0);
glVertex2f(- 4, -50);
glVertex2f(+ 4, -50);
glVertex2f(+ 4, - 0);
glEnd();

   //leaf
glColor3f(curColor[0]*(100-(rand()%20))/100.0,
                  curColor[1]*(100-(rand()%20))/100.0,
                  curColor[2]*(100-(rand()%20))/100.0);
glBegin(GL_POLYGON);
glVertex2f(0,0);
glVertex2f(-6,-10);
glVertex2f(-10,-10);
glVertex2f(-15,-16);
glVertex2f(-20,-19);
glVertex2f(-24,-17);
glVertex2f(-30,-13);
glVertex2f(-34,-10);
glVertex2f(-38,-5);
glVertex2f(-40,0);
glVertex2f(-40,5);
glVertex2f(-39,10);
glVertex2f(-38,15);
glVertex2f(-35,20);
glVertex2f(-30,28);
glVertex2f(-20,37);
glVertex2f(-18,40);
glVertex2f(-10,48);
glVertex2f(-3,58);
glVertex2f(-2,61);
glVertex2f(-1,64);
glVertex2f(0,65);
glVertex2f(0,0);
glVertex2f(6,-10);
glVertex2f(10,-10);
glVertex2f(15,-16);
glVertex2f(20,-19);
glVertex2f(24,-17);
glVertex2f(30,-13);
glVertex2f(34,-10);
glVertex2f(38,-5);
glVertex2f(40,0);
glVertex2f(40,5);
```

```
glVertex2f(39,10);
glVertex2f(38,15);
glVertex2f(35,20);
glVertex2f(30,28);
glVertex2f(20,37);
glVertex2f(18,40);
glVertex2f(10,48);
glVertex2f(3,58);
glVertex2f(2,61);
glVertex2f(1,64);
glVertex2f(0,65);
glVertex2f(0,0);
   glEnd();

//Veins
glColor3f(curColor[0]*0.2, curColor[1]*0.2, curColor[2]*0.2);
   glBegin(GL_LINE_STRIP);
glVertex2f(+0, +  0);
glVertex2f(0,65);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0,0);
glVertex2f(-25,10);
glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0,0);
glVertex2f(25,10);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +10);
glVertex2f(-20,20);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +10);
glVertex2f(20,20);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +20);
glVertex2f(-15,30);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +20);
glVertex2f(15,30);
   glEnd();
```

```
glBegin(GL_LINE_STRIP);
glVertex2f(0, +30);
glVertex2f(-12,40);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +30);
glVertex2f(12,40);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +40);
glVertex2f(-5,46);
   glEnd();

glBegin(GL_LINE_STRIP);
glVertex2f(0, +40);
glVertex2f(5,46);
   glEnd();

}
void display(){
glClear(GL_COLOR_BUFFER_BIT);
glClearColor(0, 0, 0, 0);
   glLoadIdentity();
   glColor3f(0, 1, 0);
 for (int i = 0; i<leafCount; i++)
       {
           leaf(leafx[i], leafy[i], leafsize[i], leafangle[i]);
           leafy[i] -= 1*leafsize[i];
           leafangle[i] += leafrotatespeed[i];

         if (leafy[i] < -300-200*leafsize[i])
          {
                    leafy[i] = 400;
                    leafx[i] = rand()%701 - 350;
             }
       }

 for (int i = 0; i<3; i++)
          curColor[i] = perc*weatherColor[i] + (1-perc)*curColor[i];
   if(perc < 1)
          perc += 0.00001;
 glFlush();
Sleep(1);
glutPostRedisplay();
}

void menu(int op)
{
switch (op)
```

```
  {
 case 1:
          spring();
          glutPostRedisplay();
          break;
 case 2:
          summer();
          glutPostRedisplay();
          break;
 case 3:
          winter();
          glutPostRedisplay();
          break;
 case 4:
          fall();
          glutPostRedisplay();
          break;
 case 5:
          exit(0);
}

}

int main(int argc, char *argv[])
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(800, 600);
glutCreateWindow("SEASONS");
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-400, 400, -300, 300);
glMatrixMode(GL_MODELVIEW);

glutCreateMenu(menu);
glutAddMenuEntry("spring", 1);
glutAddMenuEntry("summer", 2);
glutAddMenuEntry("winter", 3);
glutAddMenuEntry("fall", 4);
glutAddMenuEntry("Quit", 5);
glutAttachMenu(GLUT_RIGHT_BUTTON);

glutDisplayFunc(display);
glutMainLoop();
}
```

CHAPTER - 3

RESULT ANALYSIS

3.1 Snap Shots

FIG 3.1.1 MENU DISPLAY

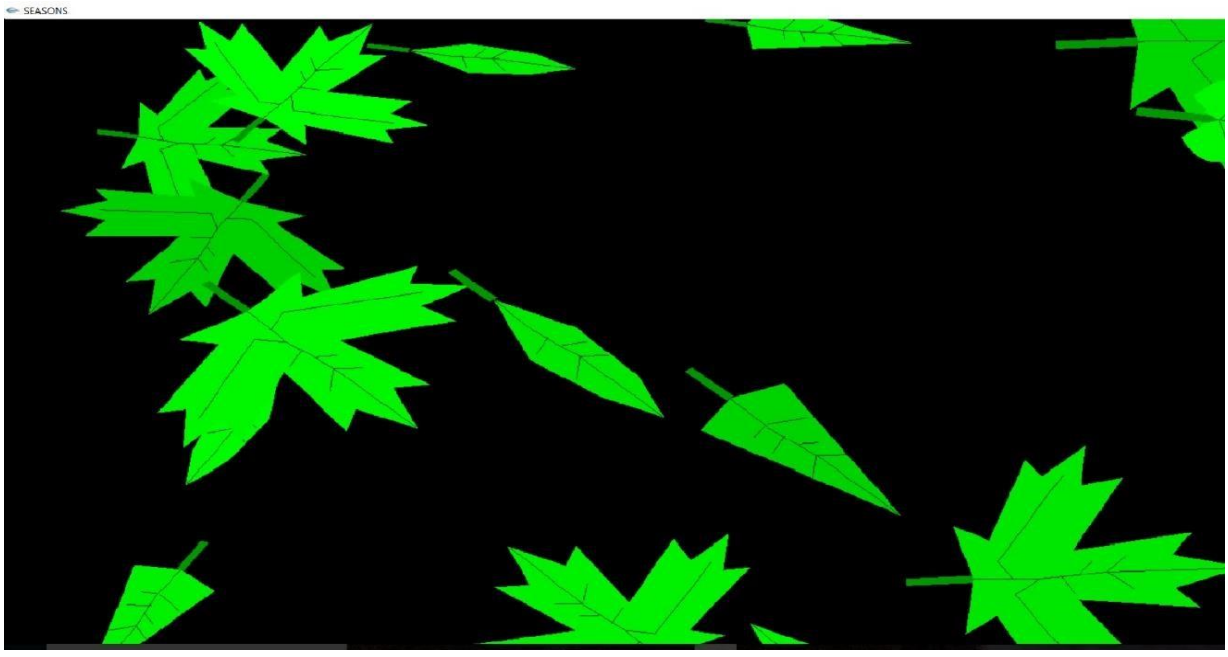


FIG.3.1.2 SPRING

FIG 3.1.3 SUMMER

FIG 3.1.4 WINTER

FIG 3.1.5 FALL

3.2 Disscussion

The user controls the color of the leaves by selecting a season from the options in the displayed menu. According to the season the falling leaves change into the assigned color.

The colors assigned for the seasons are:

- Spring: Green
- Summer: Yellow
- Winter: Brown
- Fall: Red

If the package is being used from the designing end, then the user may also control the number of the leaves, the rotation speed, the speed at which the leaves are falling and other such functions.

CHAPTER-4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

The project efficiently and correctly been built and the previously existing errors have been rectified and corrected. The required functions stated earlier have been implemented. After all testing process, the program is now ready to be used. We can use the menus efficiently to change the season with respect to our needs and adjust it. Further edits can also be done in case we want some changes.

4.2 Future Work

In future the following enhancements could be done:

- Providing Background colors.
- Providing High Quality Graphics.

CHAPTER-5

REFERENCE

[1] The Red Book-OpenGL programming Guide,6th edition

[2] Edward Angel Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th edition, Addison and Wesley

[3] Donald Hearn and Pauline Baker: Computer Graphics- OpenGl version, 3rdedition, Pearson Education

[4] F.S. Hill Jr.: Computer Graphics Using OpenGL,3rdedition, PHI.

Websites:

- http://www.opengl.org/registry/
- www.na.fs.fed.us
- www.openglforum.com
- http://www.esf.edu
- www.vtucs.com