Computer Graphics (UCS505)
Project
on
Cycle of Seasons

Submitted by:

Aakarsh Walia	(102153005)
Dewang Goyal	(102103552)
Anjali Singh	(102103779)

B.E. Third Year – COE 20

Submitted to:
Dr. Anupam Garg

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

JAN-JUN 2024

Table of Contents

Introduction to Project

Computer graphics is an exciting and rapidly evolving field that deals with the creation, manipulation, and display of visual content using computers. It involves the use of mathematical algorithms and computer programming to generate, animate, and render 2D and 3D images, videos, and animations. The application of computer graphics is vast and can be seen in various fields such as entertainment, education, engineering, medicine, and design. From creating visually stunning video games and movies to designing buildings and engineering products, computer graphics plays a vital role in enhancing the visual communication and problem-solving abilities of professionals across different domains. This project uses OpenGL which is a powerful graphics library that allows developers to create visually stunning 3D and 2D applications.

Our Project: Cycle of Seasons

A Dynamic 2D animation that shows the five seasons: Summer, Autumn, Winter, Spring and transitions between them like rainfall and snowfall etc.

Possible Applications of the Project:

- The season change cycle simulation can be an educational tool that can help individuals understand the different seasons and the changes that occur in the environment.
- The project can have various applications, such as in the field of gaming, where it can be used to create realistic and immersive game environments.
- It can also be used in the field of virtual reality, where users can experience the different seasons in a virtual world.
- Additionally, the season change cycle simulation can be utilized in the cartoon film industry for creating realistic outdoor scenes that showcase the changing of seasons.
- It can also be used by city planners and architects to simulate how the environment changes throughout the year and how different buildings and structures look during different seasons.

In this project, we have utilized OpenGL's advanced features to create a stunning visual representation of the season change cycle. From the textures and lighting to the movement of clouds, rain and snow, this project can offer a captivating experience that allows users to appreciate the beauty of nature and the changing of seasons.

Computer Graphics Concepts Used

1. **Translation:** Translation in computer graphics refers to the process of moving a 2D object in a specified direction by a certain distance. This transformation changes the position of the object without altering its shape, size or orientation. In 2D graphics, an object is represented by a set of vertices or points in a two-dimensional space. To translate the object, the coordinates of each point are modified by adding a constant value to its x and y coordinates.

   $$T(dx, dy, dz) = \begin{vmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

2. **Rotation:** Rotation in computer graphics refers to the process of rotating a 2D object around a fixed point. The object is transformed by an angle in a specified direction, either clockwise or counterclockwise, while maintaining its shape, size, and position. In 2D graphics, an object is represented by a set of vertices or points in a two-dimensional space. To rotate the object, each point is rotated by a specified angle around a fixed point. This is achieved by using a rotation matrix, which defines the transformation.

   $$R(\theta) = \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

3. **Scaling:** Scaling in computer graphics refers to the process of changing the size of a 2D object by multiplying its coordinates by a scaling factor. Scaling is a geometric transformation that changes the size of an object while preserving its shape and orientation. In 2D graphics, an object is represented by a set of vertices or points in a two-dimensional space. To scale the object, each point is multiplied by a scaling factor in the x and y directions. This is achieved by using a scaling matrix, which defines the transformation.

   $$S(sx, sy) = \begin{vmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

4. **Timer Functions:** Timer functions are an essential part of computer graphics, as they allow you to perform tasks at regular intervals or after a specific amount of time has elapsed. In computer graphics, timers are commonly used for animation, game development, and other tasks that require updates at regular intervals. Examples- Sleep(), time(), clock(), glutTimerFunc(), glutGet(GLUT_ELAPSED_TIME), etc.
5. **Trigonometric Circle Drawing:** Trigonometric circle drawing is a technique used in computer graphics to draw circles using trigonometric functions such as sine and cosine. The basic idea is to divide the circle into equal parts and then use trigonometric functions to compute the x and y coordinates of each point on the circle.
6. **Primitive types used:**
   - Points (**GL_POINTS**) - draws a single point at each vertex
   - Lines (**GL_LINES**) - draws a line between each pair of vertices
   - Triangles (**GL_TRIANGLES**) - draws a triangle for every set of three vertices
   - Triangle fan (**GL_TRIANGLE_FAN**) - draws a series of connected triangles where each additional vertex forms a new triangle with the first vertex and the previous vertex
   - Quads (**GL_QUADS**) - draws a quadrilateral for every set of four vertices
   - Polygon (**GL_POLYGON**) - draws a polygon with an arbitrary number of sides, defined by a set of vertices.
7. **Flags used:**
   - **GL_COLOR_BUFFER_BIT**: It is a flag used in OpenGL to clear the color buffer of the current rendering context. The color buffer is a section of memory used to store the colors of each pixel in the framebuffer, which is the image being rendered by OpenGL. When GL_COLOR_BUFFER_BIT is passed as an argument to the glClear function, the color buffer is cleared to the color specified by glClearColor, which is typically set at the beginning of the rendering loop. This function call is often used at the beginning of each frame to clear the previous frame's color buffer before rendering the new frame.
8. **GL Library Inbuilt Functions used:**
   - **glBegin:** Begins the definition of a new primitive.
   - **glEnd:** Ends the definition of a primitive.
   - **glVertex:** Defines a vertex for the current primitive.
   - **glColor:** Sets the current color used for subsequent drawing commands.

- **glMatrixMode:** Sets the current matrix mode, such as modelview or projection.
- **glLoadIdentity:** Replaces the current matrix with an identity matrix.
- **glLoadMatrix:** Replaces the current matrix with a specified matrix.
- **glPushMatrix:** Saves the current matrix on the matrix stack.
- **glPopMatrix:** Restores the current matrix from the matrix stack.
- **glViewport:** Sets the viewport, or the portion of the window where drawing occurs.
- **glClearColor:** Sets the color used to clear the color buffer.
- **glRotatef:** Applies a rotation to the current matrix.
- **glTranslatef:** Applies a translation to the current matrix.
- **glScalef:** Applies a scaling transformation to the current matrix.
- **glutMainLoop:** Enters the event processing loop for a GLUT window.
- **glutTimerFunc:** Registers a timer callback function to be called after a specified number of milliseconds.
- **glutInit:** Initializes the GLUT library.
- **glutCreateWindow:** Creates a new window with the specified title.
- **glutReshapeWindow:** Resizes a window to a specified width and height.
- **glutDisplayFunc:** Registers a display callback function to be called when the window needs to be redrawn.
- **glFlush:** Forces OpenGL to process any pending commands and draw the result to the screen.
- **glTranslated:** Translates the current matrix by a specified amount in each axis.
- **glLineWidth:** Sets the width of lines drawn using glBegin(GL_LINES) or glBegin(GL_LINE_STRIP).
- **glutPostRedisplay:** Marks the current window as needing to be redrawn.

9. **<u>Header Libraries Used:</u>**

- **windows.h:** Provides access to some low-level Windows operating system functions, such as memory management and thread creation.
- **GL/glut.h:** Provides access to the OpenGL Utility Toolkit (GLUT) library, which is used to create windows and handle user input events.
- **math.h:** Provides access to some common math functions, such as sine and cosine.

User Defined Functions

1. Update Functions:

rainUp() and cloudUp() functions are implemented using a timer function:

rainUp function updates the position of rain particles on the screen. It takes an integer value as an argument. It checks if the rain particles have reached the bottom of the screen and if so, it resets their position to the top. Then it updates the position of rain particles by subtracting rainSpeed from rainP. Finally, it posts a redisplay request to update the screen and sets the timer function to call rainUp again after a specified interval of 100 milliseconds.

cloudUp function updates the position of clouds on the screen. It takes an integer value as an argument. It checks if the clouds have reached the left edge of the screen and if so, it resets their position to the right edge. Then it updates the position of clouds by subtracting cloudS from cloudP. Finally, it posts a redisplay request to update the screen and sets the timer function to call cloudUp again after a specified interval of 100 milliseconds.

2. Circle Function:

It takes three float arguments: a, b, and r, which represent the x-coordinate, y-coordinate, and radius of the circle, respectively. The function uses specifies and uses the GL_TRIANGLE_FAN primitive type to make a circle. It then sets the center of the circle to (x,y) using glVertex2f command.

To draw the circle, the function calculates the vertices of the triangle fan by looping through a for loop. The loop generates ‘triangleAmount’ number of vertices for the circle, which is set to 100 in the code. It uses the cos and sin functions to calculate the x and y coordinates of each vertex based on the current angle i and the radius r. Finally, it specifies each vertex using glVertex2f command.

Similarly we made **HalfCircle Function.**

3. Windmill Function:

The function draws a windmill by making use of user-defined helper functions "quad" and "circle". The windmill is composed of a pillar and four blades. The pillar is drawn using the "quad" function, which is a helper function that draws a quadrilateral. The color of the pillar is set to a shade of brown using the "glColor3ub" function.

The four blades are drawn using the "quad" function as well. The color of the blades is set to a shade of yellow using the "glColor3ub" function. The blades are rotated around the center of the windmill using the "glRotatef" function. The rotation angle is determined by the variable "wSpeed", which is decreased by 0.15f after each frame to simulate the movement of the blades.

4. <u>House Function:</u>

The function draws a house and its surrounding environment using various helper functions.

First, two circles are drawn using the "circle" helper function to represent patches of grass near the house. The circles are drawn using different sizes and positions to create the appearance of depth.

Next, the house itself is drawn using a combination of the "glBegin" and "glEnd" functions. The house is a rectangle with a brownish-gray color, and it is created using the "GL_QUADS" option to draw a four-sided polygon. The house is scaled and translated to fit its desired position on the screen. A door is then drawn on the front of the house using a similar technique as the house itself. A triangular roof is drawn using the "GL_POLYGON" option.

5. <u>downFlower3 Function:</u>

This is a function for drawing three sets of flowers. Each set consists of four circles arranged around a central circle, with different colors for each set. The first set of flowers is drawn at the default position. Four circles are drawn in a cross shape around the center, with red petals and a yellow center. The second set of flowers is drawn by translating the coordinate system 0.42 units to the right. The same pattern is drawn as the first set. The third set of flowers is drawn by translating the coordinate system another 1.0 unit to the right, then 0.03 units down. The same pattern is drawn as the first two sets.

6. <u>Whiteflower Function:</u>

This is a code for drawing grass and a white flower. The grass is drawn using glColor3ub which sets the color to a dark green shade. Then, several triangles are drawn to form the blades of grass.

The white flower is drawn using glColor3ub which sets the color. The flower consists of several polygons that are translated and rotated to form the petals.

The glBegin() and glEnd() functions are used to define the vertices of each polygon, and the glTranslated() function is used to move the current coordinate system to a new location.

Finally, glLineWidth() function is used to set the width of the lines used to draw the grass, and line() function is used to draw the stems of the flower.

7. <u>Summer Function:</u>

It is responsible for rendering a scene of a summer landscape. It is implemented using the following functions:

- glClearColor(1.0f, 1.0f, 1.0f, 1.0f) sets the clear color of the screen to white
- glClear(GL_COLOR_BUFFER_BIT); clears the color buffer to the clear color set above
- sky(); renders the sky
- mountain(); renders the mountains
- glPushMatrix() and glPopMatrix() are used to save and restore the matrix state before and after translating the cloud
- cloudP is the position of the cloud on the x-axis, and glTranslatef(cloudP, 0.0f, 0.0f) translates the cloud to that position
- cloud(); renders the cloud
- glLoadIdentity(); resets the current matrix to the identity matrix
- circle() is used to draw circles for the sun and sun cloud
- ground(); renders the ground
- field(); renders the field
- roadGrass(); renders the grass on the side of the road
- roadFlower(); renders flowers on the side of the road
- river(); renders the river
- longGrass(); renders the long grass on the downside of the river
- windmill(); renders the windmill
- house(); renders the house

- treeLeaf(); renders the leaves of the tree
- tree(); renders the tree
- downGrass(); renders the grass on the downside of the house
- downFlower(); renders flowers on the downside of the house
- glutTimerFunc(5000, display_rainyDay, 0); sets a timer to switch the scene to a rainy day after 5000 milliseconds (5 seconds)

8. <u>rainyDay Function:</u>

The "rainDay" function is called to display the scene during a rainy day. It sets the clear color of the window to white and then draws the following objects, in order:

- Rain Cloud: A collection of circles that represents a cloud that is raining. The position of the cloud can be translated using the "RcloudP" variable.
- Ground: A rectangular shape that represents the ground.
- Field: A rectangular shape that represents a field.
- Grass Road: A rectangular shape that represents a road covered in grass.
- Flowers: A collection of small circles that represents flowers on the side of the road.
- River: A blue, wavy line that represents a river.
- Long Grass: A collection of long, thin triangles that represent long grass near the river.
- Windmill: A windmill that is scaled down and translated to the right of the scene.
- House: A rectangular shape that represents a house.
- Tree Leaf: A collection of circles that represents the leaves of a tree.
- Tree: A brown rectangle that represents the trunk of a tree.
- Down Grass: A collection of long, thin triangles that represent grass near the bottom of the scene.
- Down Flowers: A collection of small circles that represents flowers near the bottom of the scene.
- Rain: A collection of blue, diagonal lines that represent rain falling. The position of the rain can be translated using the "rainP" variable.

9. Autumn Function:

The autumn function sets the background color to white and clears the screen. It then generates the sky and mountain elements, followed by clouds that move horizontally based on the value of the variable "cloudP".

The ground and field are then generated, followed by a grass road, a river, and long grass. White flowers are then generated in various locations, followed by a windmill, more white flowers, a house, a tree with leaves, and more white flowers. Finally, down grass is generated, and a timer is set to call the winter function after 5 seconds.

10. Winter Function:

The sky is drawn using the "sky()" function which sets the color and draws a rectangle to fill the top portion of the screen. The mountain is drawn using the "mountain()" function which draws a triangular polygon with different shades of gray. The mountain snow is drawn using a series of polygons of different shapes and sizes with white color.

The ground is drawn using the "ground()" function which sets the color and draws a rectangle to fill the lower portion of the screen. The field is drawn using the "field()" function which sets the color and draws a rectangle with a green color. The road is drawn using the "roadGrass()" function which sets the color and draws a rectangle with a dark green color.

The river is drawn using a quadrilateral polygon with a light green color at the top and a light blue color at the bottom. It also adds four diagonal lines to give an illusion of water flow. The long grass is drawn using the "longGrass()" function which sets the color and draws a series of lines with varying thicknesses and lengths.

The windmill is drawn using the "windmill()" function which consists of several polygons and lines. It rotates based on the animation frame. The house is drawn using the "house()" function which sets the color and draws a series of polygons to create a house shape.

Finally, a tree is drawn using the "tree()" function which sets the color and draws a series of lines and polygons to create a tree shape.

11. <u>Spring Function</u>

The function begins by setting the clear color to white and clearing the color buffer. Then, it draws the sky using a light blue color, followed by the mountains using a darker shade of blue.

The ground is drawn using a green color, followed by a green field and a grassy road with flowers. The river is then drawn using a blue color.

After that, the function draws some long grass using a darker green color, and a windmill using various shades of brown. It then draws a house using a dark green color.

The trees are drawn using circles of different sizes and colors to create the effect of leaves. Finally, the function draws various types of flowers, including spring flowers, red flowers, and yellow flowers, as well as some down grass using a lighter green color.

12. <u>Main Function:</u>

Creates a window of size 800x700. The glutInit function initializes the GLUT library and sets up the program's command line arguments. The glutCreateWindow function creates a window with the given title. The glutReshapeWindow function sets the window's size.

The glutDisplayFunc function sets the display callback function to summer, which is responsible for drawing the initial summer scene.

There are several glutTimerFunc calls in the main function that set up timers to call various functions at specified intervals. These functions include RcloudUp, cloudUp, snowUp, rainUp.

Project Source Code

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
# define PI 3.14159265358979323846
/////////////////////////basic function
void circle(float a, float b, float r) {
    int i;
    int triangleAmount = 100;
    GLfloat twicePi = 2.0f * PI;
    GLfloat x = a; GLfloat y = b; GLfloat radius = r;
    glBegin(GL_TRIANGLE_FAN);
    glVertex2f(x, y);
    for (i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            x + (radius * cos(i * twicePi / triangleAmount)),
            y + (radius * sin(i * twicePi / triangleAmount))
        );
    }   glEnd();
}
void quad(float a, float b, float c, float d, float e, float f, float g, float h) {
    glBegin(GL_QUADS);
    glVertex2f(a, b);
    glVertex2f(c, d);
    glVertex2f(e, f);
    glVertex2f(g, h);
    glEnd();
}
void triangle(float a, float b, float c, float d, float e, float f) {
    glBegin(GL_TRIANGLES);
    glVertex2f(a, b);
    glVertex2f(c, d);
    glVertex2f(e, f);
    glEnd();
}
void line(float a, float b, float c, float d) {
    glBegin(GL_LINES);
    glVertex2f(a, b);
    glVertex2f(c, d);
    glEnd();
}
void HalfCircle(float x, float y, float r) {
    int i;
    int lineAmount = 100;
    GLfloat twicePi = 2.0f * PI;
    glBegin(GL_TRIANGLE_FAN);
    for (i = 50; i <= lineAmount; i++)
    {
        glVertex2f(x + (r * cos(i * twicePi / lineAmount)), y + (r * sin(i * twicePi
/ lineAmount)));
    }
    glEnd();
```

```
}
void rain()
{
    glBegin(GL_LINES);
    glVertex2f(-.85, 1.9);
    glVertex2f(-.8, 1.8);
    glVertex2f(-.55, 1.9);
    glVertex2f(-.5, 1.8);
    glVertex2f(-.25, 1.9);
    glVertex2f(-.2, 1.8);
    glVertex2f(.05, 1.9);
    glVertex2f(.1, 1.8);
    glVertex2f(.35, 1.9);
    glVertex2f(.4, 1.8);
    glVertex2f(.65, 1.9);
    glVertex2f(.7, 1.8);
    glVertex2f(.95, 1.9);
    glVertex2f(1.0, 1.8);
    glVertex2f(-1.0, 1.6);
    glVertex2f(-.95, 1.5);
    glVertex2f(-.7, 1.6);
    glVertex2f(-.65, 1.5);
    glVertex2f(-.4, 1.6);
    glVertex2f(-.35, 1.5);
    glVertex2f(-.1, 1.6);
    glVertex2f(-.05, 1.5);
    glVertex2f(.2, 1.6);
    glVertex2f(.25, 1.5);
    glVertex2f(.5, 1.6);
    glVertex2f(.55, 1.5);
    glVertex2f(.8, 1.6);
    glVertex2f(.85, 1.5);
    glVertex2f(-.85, 1.3);
    glVertex2f(-.8, 1.2);
    glVertex2f(-.55, 1.3);
    glVertex2f(-.5, 1.2);
    glVertex2f(-.25, 1.3);
    glVertex2f(-.2, 1.2);
    glVertex2f(.05, 1.3);
    glVertex2f(.1, 1.2);
    glVertex2f(.35, 1.3);
    glVertex2f(.4, 1.2);
    glVertex2f(.65, 1.3);
    glVertex2f(.7, 1.2);
    glVertex2f(.95, 1.3);
    glVertex2f(1.0, 1.2);
    glVertex2f(-1.0, 1.0);
    glVertex2f(-.95, .9);
    glVertex2f(-.7, 1.0);
    glVertex2f(-.65, 0.9);
    glVertex2f(-.4, 1.0);
    glVertex2f(-.35, 0.9);
    glVertex2f(-.1, 1.0);
    glVertex2f(-.05, 0.9);
    glVertex2f(.2, 1.0);
```

```
        glVertex2f(.25, .9);
        glVertex2f(.5, 1.0);
        glVertex2f(.55, 0.9);
        glVertex2f(.8, 1.0);
        glVertex2f(.85, 0.9);
        glVertex2f(-.85, .7);
        glVertex2f(-.8, .6);
        glVertex2f(-.55, .7);
        glVertex2f(-.5, 0.6);
        glVertex2f(-.25, .7);
        glVertex2f(-.2, 0.6);
        glVertex2f(.05, .7);
        glVertex2f(.1, .6);
        glVertex2f(.35, .7);
        glVertex2f(.4, .6);
        glVertex2f(.65, .7);
        glVertex2f(.7, .6);
        glVertex2f(.95, .7);
        glVertex2f(1.0, .6);
        glVertex2f(-1.0, .4);
        glVertex2f(-.95, .3);
        glVertex2f(-.7, .4);
        glVertex2f(-.65, 0.3);
        glVertex2f(-.4, .4);
        glVertex2f(-.35, 0.3);
        glVertex2f(-.1, .4);
        glVertex2f(-.05, 0.3);
        glVertex2f(.2, .4);
        glVertex2f(.25, .3);
        glVertex2f(.5, .4);
        glVertex2f(.55, 0.3);
        glVertex2f(.8, .40);
        glVertex2f(.85, 0.3);
        glVertex2f(-.85, .1);
        glVertex2f(-.8, 0.0);
        glVertex2f(-.55, .1);
        glVertex2f(-.5, 0.0);
        glVertex2f(-.25, .1);
        glVertex2f(-.2, 0.0);
        glVertex2f(.05, .1);
        glVertex2f(.1, .0);
        glVertex2f(.35, .1);
        glVertex2f(.4, .0);
        glVertex2f(.65, .1);
        glVertex2f(.7, .0);
        glVertex2f(.95, .1);
        glVertex2f(1.0, .0);
        glVertex2f(-1.0, -.2);
        glVertex2f(-.95, -.3);
        glVertex2f(-.7, -.2);
        glVertex2f(-.65, -0.3);
        glVertex2f(-.4, -.2);
        glVertex2f(-.35, -0.3);
        glVertex2f(-.1, -.2);
        glVertex2f(-.05, -0.3);
        glVertex2f(.2, -.2);
        glVertex2f(.25, -.3);
        glVertex2f(.5, -.2);
        glVertex2f(.55, -.3);
        glVertex2f(.8, -.2);
        glVertex2f(.85, -.3);
        glVertex2f(-.85, -.5);
        glVertex2f(-.8, -.6);
        glVertex2f(-.55, -.5);
        glVertex2f(-.5, -.6);
        glVertex2f(-.25, -.5);
        glVertex2f(-.2, -.6);
        glVertex2f(.05, -.5);
        glVertex2f(.1, -.6);
        glVertex2f(.35, -.5);
        glVertex2f(.4, -.6);
        glVertex2f(.65, -.5);
        glVertex2f(.7, -.6);
        glVertex2f(.95, -.5);
        glVertex2f(1.0, -.6);
        glVertex2f(-1.0, -.8);
        glVertex2f(-.95, -.9);
        glVertex2f(-.7, -.8);
        glVertex2f(-.65, -0.9);
        glVertex2f(-.4, -.8);
        glVertex2f(-.35, -0.9);
        glVertex2f(-.1, -.8);
        glVertex2f(-.05, -0.9);
        glVertex2f(.2, -.8);
        glVertex2f(.25, -.9);
        glVertex2f(.5, -.8);
        glVertex2f(.55, -.9);
        glVertex2f(.8, -.8);
        glVertex2f(.85, -.9);
        glVertex2f(-.85, -1.1);
        glVertex2f(-.8, -1.2);
        glVertex2f(-.55, -1.1);
        glVertex2f(-.5, -1.2);
        glVertex2f(-.25, -1.1);
        glVertex2f(-.2, -1.2);
        glVertex2f(.05, -1.1);
        glVertex2f(.1, -1.2);
        glVertex2f(.35, -1.1);
        glVertex2f(.4, -1.2);
        glVertex2f(.65, -1.1);
        glVertex2f(.7, -1.2);
        glVertex2f(.95, -1.1);
        glVertex2f(1.0, -1.2);
        glVertex2f(-1.0, -1.4);
        glVertex2f(-.95, -1.5);
        glVertex2f(-.7, -1.4);
        glVertex2f(-.65, -1.5);
        glVertex2f(-.4, -1.4);
        glVertex2f(-.35, -1.5);
        glVertex2f(-.1, -1.4);
```

```
    glVertex2f(-.05, -1.5);
    glVertex2f(.2, -1.4);
    glVertex2f(.25, -1.5);
    glVertex2f(.5, -1.4);
    glVertex2f(.55, -1.5);
    glVertex2f(.8, -1.4);
    glVertex2f(.85, -1.5);
    glEnd();
}
/////////////////////////////UPDATE FUNCTIONS
GLfloat rainP = 2.50f;
GLfloat rainSpeed = 0.1f;
void rainUp(int value)
{
    if (rainP < -1.0)
        rainP = .50f;
    rainP -= rainSpeed;
    glutPostRedisplay();
    glutTimerFunc(100, rainUp, 0);
}
GLfloat cloudP = 2.0f;
GLfloat cloudS = 0.02f;
void cloudUp(int value) {
    if (cloudP < -2.0)
        cloudP = 2.0f;
    cloudP -= cloudS;
    glutPostRedisplay();
    glutTimerFunc(100, cloudUp, 0);
}
GLfloat snowP = 0.0105f;
GLfloat snowS = 0.0105f;
void snowUp(int value) {
    if (snowP < -1.50)
        snowP = 1.0f;
      snowP -= snowS;
    glutPostRedisplay();
    glutTimerFunc(100, snowUp, 0);
}
GLfloat RcloudP = 3.5f;
GLfloat RcloudS = 0.02f;

void RcloudUp(int value) {
    if (RcloudP < -0.0)
        RcloudP  =  1.0f;
    RcloudP  -=  RcloudS;
    glutPostRedisplay();
    glutTimerFunc(100, RcloudUp, 0);
}
GLfloat wSpeed = 0.0f;
void windmill() {
    //pillar
    glColor3ub(230, 162, 80);
    quad(-.01, 0, -.03, -.35, .03, -.35, .01, 0);
    glPushMatrix();
    glRotatef(wSpeed, 0.0, 0.0, 0.1);
    //glColor3ub(247, 5, 5);
    glColor3ub(247, 203, 5);
    quad(0, 0, 0, .2, -.1, .2, 0, 0);
    quad(0, 0, 0, -.2, .1, -.2, 0, 0);
    quad(0, 0, .2, 0, .2, .1, 0, 0);
    quad(0, 0, -.2, 0, -.2, -.1, 0, 0);
    glPopMatrix();
    wSpeed -= 0.15f;
    //center
    glColor3ub(252, 252, 252);
    circle(0.0, 0.0, .015);
}


/////////////////////object function
void sky() {
    glBegin(GL_QUADS);
    glVertex2f(-1, 0);
    glVertex2f(1, 0);
    glVertex2f(1, 1);
    glVertex2f(-1, 1);
    glEnd();
}

void snowball() {
    circle(-1.5, .95, .01);
    circle(-.95, .75, .01);
    circle(-.65, 0.55, .01);
    circle(-.35, 0.46, .01);
    circle(-.05, 0.648, .01);
    circle(.25, .83, .01);
    circle(.55, 0.8, .01);
    circle(.75, 0.35, .01);
    circle(1., 0.26, .01);
    circle(-.45, 0.148, .01);
}
void mountain() {
    //right
    glColor3ub(61, 119, 135);
    triangle(.55, 0, 1.1, 0, .8, .22);
    glColor3ub(29, 78, 92);
    triangle(.55, 0, .65, 0, .8, .22);
    //main mountain
    glBegin(GL_QUADS);

    glColor3ub(75, 152, 173);
    glVertex2f(-.35, 0);
    glVertex2f(.75, 0);
    glVertex2f(.25, .45);
    glVertex2f(.1, .43);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3ub(255, 255, 255);
    glVertex2f(.25, .45);
    glVertex2f(.1, .43);
    glVertex2f(-.039, .3);
    glVertex2f(.08, .35);
    glVertex2f(.06, .25);
    //left up 2
    glVertex2f(.17, .355);
    glVertex2f(.25, .27);
    //left up 1
    glVertex2f(.25, .38);
    //rightt dopwn
    glVertex2f(.42, .3);
    glEnd();
    glBegin(GL_TRIANGLES);
    glColor3ub(61, 119, 135);
    //left 1
    glVertex2f(-1.1, 0);
    glVertex2f(-.75, 0);
    glVertex2f(-.9, .12);
    glColor3ub(29, 78, 92);
    glVertex2f(-1.1, 0);
    glVertex2f(-.98, 0);
    glVertex2f(-.9, .12);
    //left2
    glColor3ub(61, 119, 135);
    glVertex2f(-.8, 0);
    glVertex2f(-.3, 0);
    glVertex2f(-.55, .15);
    glColor3ub(29, 78, 92);
    glVertex2f(-.8, 0);
    glVertex2f(-.7, 0);
    glVertex2f(-.55, .15);
    //left big
    glColor3ub(61, 119, 135);
    glVertex2f(-.45, 0);
    glVertex2f(.1, 0);
    glVertex2f(-.2, .23);
    glColor3ub(29, 78, 92);
    glVertex2f(-.45, 0);
    glVertex2f(-.35, 0);
    glVertex2f(-.2, .23);
    ////mid
    glColor3ub(61, 119, 135);
    glVertex2f(-.1, 0);
    glVertex2f(.3, 0);
    glVertex2f(.1, .12);
    glColor3ub(29, 78, 92);
    glVertex2f(-.1, 0);
    glVertex2f(0, 0);
    glVertex2f(.1, .12);
    glEnd();
}
void house() {
    ///// houseGrass
      //left
    circle(0.52, -.33, .04);
    circle(0.49, -.34, .03);
```

```
        glTranslated(.4, -.005, 0);
        //right
        circle(0.52, -.31, .045);
        circle(0.55, -.325, .03);
        glLoadIdentity();
        //////////////////// house
        glScalef(1.2, 1.2, 0);
        glTranslated(.05, 0.0, 0);
        glBegin(GL_QUADS);
        glColor3ub(212, 205, 205);
        glVertex2f(.4, -.1);
        glVertex2f(.4, -.3);
        glVertex2f(.7, -.3);
        glVertex2f(.7, -.1);
        glEnd();
        //door
        glBegin(GL_QUADS);
        glColor3ub(242, 133, 31);
        glVertex2f(.52, -.2);
        glVertex2f(.52, -.3);
        glVertex2f(.58, -.3);
        glVertex2f(.58, -.2);
        glVertex2f(.43, -.21);
        glVertex2f(.43, -.26);
        glVertex2f(.49, -.26);
        glVertex2f(.49, -.21);
        glVertex2f(.61, -.21);
        glVertex2f(.61, -.26);
        glVertex2f(.67, -.26);
        glVertex2f(.67, -.21);
        glEnd();
        glBegin(GL_POLYGON);
        glColor3ub(245, 43, 32);
        glVertex2f(.39, -.05);
        glVertex2f(.37, -.18);
        glVertex2f(.73, -.18);
        glVertex2f(.71, -.05);
        glEnd();
        glBegin(GL_QUADS);
        glColor3ub(204, 51, 0);
        glVertex2f(.39, -.29);
        glVertex2f(.39, -.31);
        glVertex2f(.71, -.31);
        glVertex2f(.71, -.29);
        glEnd();
        glLoadIdentity();
    }
    void downFlower() {
        //red
        glColor3ub(235, 29, 2);
        circle(-.78, -.9f, .02);
        circle(-.72, -.9f, .02);
        circle(-.75, -.87f, .02);
        circle(-.75f, -.93, .02);
        //   center
        glColor3ub(255, 242, 10);
        circle(-.75f, -.9, .015);
        //yellow
        glScalef(0.9, .9, 0);
        glTranslated(0, 0, 0);
        glColor3ub(250, 197, 40);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center40
        glColor3ub(252, 23, 23);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.25, -.03, 0);
        glColor3ub(235, 29, 2);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center40
        glColor3ub(255, 242, 10);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.42, -.0, 0);
        glColor3ub(250, 197, 40);
        circle(-.78, -.9, .02); circle(-.72, -.9, .02);
        circle(-.75, -.87, .02); circle(-.75, -.93, .02);
        //   center
        glColor3ub(252, 23, 23);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.56, -.04, 0);
        glColor3ub(255, 186, 245);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center
        glColor3ub(245, 24, 65);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.7, -.01, 0);
        glColor3ub(235, 29, 2);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center
        glColor3ub(255, 242, 10);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.87, .07, 0);
        glColor3ub(255, 186, 245);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(245, 24, 65);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(.97, -.02, 0);
        glColor3ub(255, 186, 245);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(245, 24, 65);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.12, -.03, 0);
        glColor3ub(235, 29, 2);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(255, 242, 10);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.26, -.04, 0);
        glColor3ub(235, 29, 2);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center
        glColor3ub(255, 242, 10);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.5, .04, 0);
        glColor3ub(255, 186, 245);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(245, 24, 65);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.36, -.0, 0);
        glColor3ub(250, 197, 40);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        //   center
        glColor3ub(252, 23, 23);
        circle(-.75, -.9, .015); glLoadIdentity();
    }
    void downFlower3() {
        glColor3ub(235, 29, 2);
        circle(-.78, -.9f, .02);
        circle(-.72, -.9f, .02);
        circle(-.75, -.87f, .02);
        circle(-.75f, -.93, .02);
        //   center
        glColor3ub(255, 242, 10);
        circle(-.75f, -.9, .015);
        glTranslated(.42, -.0, 0);
        glColor3ub(250, 197, 40);
        circle(-.78, -.9, .02); circle(-.72, -.9, .02);
```

```
        circle(-.75, -.87, .02); circle(-.75, -.93, .02);
        //   center
        glColor3ub(252, 23, 23);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.0, -.03, 0);
        glColor3ub(235, 29, 2);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(255, 242, 10);
        circle(-.75, -.9, .015); glLoadIdentity();
        glTranslated(1.5, .04, 0);
        glColor3ub(255, 186, 245);
        circle(-.78, -.9, .02);
        circle(-.72, -.9, .02);
        circle(-.75, -.87, .02);
        circle(-.75, -.93, .02);
        glColor3ub(245, 24, 65);
        circle(-.75, -.9, .015); glLoadIdentity();
    }
    void tree() {
        glBegin(GL_POLYGON);
        glColor3ub(69, 30, 14);
        glVertex2f(-.69, -.2);
        glVertex2f(-.69, -.85);
        glVertex2f(-.57, -.85);
        glVertex2f(-.57, -.2);
        glEnd();
        //right down
        quad(-.6, -.22, -.4, -.29, -.39, -.28, -.55, -.2);
        //right mid
        quad(-.63, -.18, -.65, -.28, -.4, -.1, -.33, -.09);

        //right up
        quad(-.6, -.18, -.55, -.18, -.55, -.01, -.56, -.02);
        //left
        //left up
        quad(-.75, -.16, -.68, -.18, -.78, .02, -.8, .03);
        quad(-.63, -.18, -.62, -.28, -.9, -.14, -.89, -.12);
        glBegin(GL_QUADS);
        //root
        glVertex2f(-.83, -.82);
        glVertex2f(-.95, -.88);
        glVertex2f(-.81, -.85);
        glVertex2f(-.81, -.82);
        glVertex2f(-.81, -.82);
        glVertex2f(-.83, -.84);
        glVertex2f(-.63, -.89);
        glVertex2f(-.73, -.82);
        glEnd();
    }
    void downGrass() {
        glColor3ub(30, 74, 15);
        circle(-.9, -.9, .2);
        glColor3ub(30, 74, 15);
        circle(-.33, -.93, .115);
        glColor3ub(63, 163, 20);
        circle(-.7, -.9, .15);
        glColor3ub(63, 163, 20);
        circle(-.5, -.95, .1);
        glColor3ub(63, 163, 20);
        circle(-.18, -.95, .08);
        glColor3ub(63, 163, 20);
        circle(-.03, -.95, .11);


        glTranslated(.87, .0, 0);
        glColor3ub(49, 128, 15);
        circle(-.7, -.9, .15);
        glColor3ub(49, 128, 15);
        circle(-.5, -.95, .1);
        glLoadIdentity();
        glColor3ub(63, 163, 20);
        circle(.52, -.95, .09);
        glColor3ub(30, 74, 15);
        circle(0.9, -.9, .19);
        glColor3ub(63, 163, 20);
        circle(.7, -.93, .13);
    }
    void whiteFlower() {
        /////////grass
        glColor3ub(68, 173, 47);
        triangle(-.5, -1.0, -.45, -1.0, -.35, -.65);
        triangle(-.5, -1.0, -.45, -1.0, -.6, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.35, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.0, -.65);
        triangle(-.45, -1.0, -.4, -1.0, -.2, -.65);
        triangle(.15, -1.0, .2, -1.0, .35, -.65);
        triangle(.15, -1.0, .2, -1.0, .05, -.65);
        triangle(.45, -1.0, .5, -1.0, .35, -.65);
        triangle(.55, -1.0, .6, -1.0, .55, -.65);
        triangle(.6, -1.0, .65, -1.0, .8, -.65);
        triangle(.85, -1.0, .9, -1.0, 1.0, -.65);
        glLineWidth(2);
        glColor3ub(68, 173, 47);
        line(-.5, -1.0, -.43, -.65);
        line(-.3, -1.0, -.25, -.65);
        line(-.15, -1.0, -.1, -.65);
        line(.0, -1.0, .1, -.65);
        line(.25, -1.0, .2, -.6);
        line(.28, -1.0, .32, -.65);
        line(.35, -1.0, .4, -.65);
        line(.45, -1.0, .58, -.62);
        line(.53, -1.0, .5, -.65);
        line(.6, -1.0, .7, -.65);
        line(.75, -1.0, .85, -.65);
        //////////white flower
        glColor3ub(239, 245, 237);
        glTranslated(.032, .0, 0);
```

```
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.45, -.7);
glVertex2f(-.43, -.55);
glEnd();
glTranslated(.187, .02, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.45, -.7);
glVertex2f(-.43, -.55);
glEnd();
glTranslated(.15, .0, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.45, -.7);
glVertex2f(-.45, -.55);
glEnd();
glTranslated(.2, .0, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.486, -.73); glVertex2f(-.46, -.7);
glVertex2f(-.44, -.55);
glEnd();
glTranslated(.12, .04, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.46, -.7);
glVertex2f(-.52, -.55);
glEnd();
glTranslated(.11, .0, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.46, -.7);
glVertex2f(-.46, -.55);
glEnd();
glTranslated(.07, -.05, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.46, -.7);
glVertex2f(-.46, -.58);
glEnd();
glTranslated(.115, .02, 0);
glBegin(GL_POLYGON);
glVertex2f(-.5, -.7); glVertex2f(-.48, -.73); glVertex2f(-.46, -.7);
glVertex2f(-.5, -.58);
glEnd();
glTranslated(.058, .02, 0);
glBegin(GL_POLYGON);
glVertex2f(-.48, -.7); glVertex2f(-.48, -.73); glVertex2f(-.44, -.7);
glVertex2f(-.4, -.55);
glEnd();
glTranslated(.132, .02, 0);
glBegin(GL_POLYGON);
glVertex2f(-.48, -.7); glVertex2f(-.48, -.73); glVertex2f(-.44, -.7);
glVertex2f(-.4, -.57);
glEnd();
glTranslated(.148, .0, 0);
glBegin(GL_POLYGON);
glVertex2f(-.48, -.7); glVertex2f(-.48, -.73); glVertex2f(-.44, -.7);
glVertex2f(-.4, -.55);
glEnd();
```

```
        glLoadIdentity();
    }
    void river() {
        ///////////white   //Flower
        glScalef(0.25, .2, 0);
        glTranslated(2., .86, 0);
        whiteFlower();
        glLoadIdentity();
        glBegin(GL_QUADS);
        glColor3ub(150, 201, 38);
        glVertex2f(-1, -.00);
        glVertex2f(1, -.00);
        glVertex2f(1, -.01);
        glVertex2f(-1, -.01);
        glColor3ub(43, 155, 207);
        glVertex2f(-1, -.01);
        glVertex2f(1, -.01);
        glVertex2f(1, -.1);
        glVertex2f(-1, -.1);
        glEnd();
        glColor3ub(109, 170, 199);
        glLineWidth(.05);
        line(-0.85, -.06, -0.95, -.06);
        line(-0.5, -.035, -.56, -.035);
        line(0.0, -.06, -0.1, -.06);
        line(0.8, -.03, 0.9, -.03);
    }
    void longGrass() {
        /////////// long grass
        glScalef(0.24, .2, 0);
        glTranslated(0.8, .49, 0);
        glColor3ub(68, 173, 47);
        triangle(-.5, -1.0, -.45, -1.0, -.35, -.65);
        triangle(-.5, -1.0, -.45, -1.0, -.6, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.35, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.0, -.65);
        triangle(-.45, -1.0, -.4, -1.0, -.2, -.65);
        triangle(.15, -1.0, .2, -1.0, .35, -.65);
        triangle(.15, -1.0, .2, -1.0, .05, -.65);
        triangle(.45, -1.0, .5, -1.0, .35, -.65);
        triangle(.55, -1.0, .6, -1.0, .55, -.65);
        triangle(.6, -1.0, .65, -1.0, .8, -.65);
        triangle(.85, -1.0, .9, -1.0, 1.0, -.65);
        glLoadIdentity();
        glScalef(0.25, .2, 0);
        glTranslated(0.0, .49, 0);
        glColor3ub(68, 173, 47);
        triangle(-.5, -1.0, -.45, -1.0, -.35, -.65);
        triangle(-.5, -1.0, -.45, -1.0, -.6, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.35, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.0, -.65);
        triangle(-.45, -1.0, -.4, -1.0, -.2, -.65);
        triangle(.15, -1.0, .2, -1.0, .35, -.65);
        triangle(.15, -1.0, .2, -1.0, .05, -.65);
        triangle(.45, -1.0, .5, -1.0, .35, -.65);
        triangle(.55, -1.0, .6, -1.0, .55, -.65);
        triangle(.6, -1.0, .65, -1.0, .8, -.65);
        triangle(.85, -1.0, .9, -1.0, 1.0, -.65);
        glLoadIdentity();
        glScalef(0.25, .2, 0);
        glTranslated(-2.5, .49, 0);
        glColor3ub(68, 173, 47);
        triangle(-.5, -1.0, -.45, -1.0, -.35, -.65);
        triangle(-.5, -1.0, -.45, -1.0, -.6, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.35, -.65);
        triangle(-.2, -1.0, -.15, -1.0, -.0, -.65);
        triangle(-.45, -1.0, -.4, -1.0, -.2, -.65);
        triangle(.15, -1.0, .2, -1.0, .35, -.65);
        triangle(.15, -1.0, .2, -1.0, .05, -.65);
        triangle(.45, -1.0, .5, -1.0, .35, -.65);
        triangle(.55, -1.0, .6, -1.0, .55, -.65);
        triangle(.6, -1.0, .65, -1.0, .8, -.65);
        triangle(.85, -1.0, .9, -1.0, 1.0, -.65);
        glLoadIdentity();
    }
    void cloud()  {
        circle(.625, .58, .05);
        circle(.7, .6, .08);
        circle(.79, .582, .06);
        circle(.85, .58, .04);
        /////CLOUD LEFT
        circle(-.425, .585, .04);
        circle(-.5, .6, .07);
        circle(-.59, .592, .05);
        circle(-.64, .58, .03);
    }
    void ground() {
        glBegin(GL_QUADS);
        glVertex2f(-1, -1);
        glVertex2f(1, -1);
        glVertex2f(1, 0);
        glVertex2f(-1, 0);
        glEnd();
    }
    void field() {
        triangle(-.5, -.2, 1, -.4, 1, -.1);
        triangle(-1, -.4, -1, -.7, 0, -.5);
    }
    void treeLeaf() {
        //right big
        glColor3ub(41, 140, 42);
        circle(-0.36, -0.02, .22);
        //right small
        glColor3ub(95, 168, 47);
        circle(-0.4, -0.3f, .13);
        //left small
        glColor3ub(30, 74, 34);
        circle(-0.895, -0.13, .15);
        //left big
        glColor3ub(41, 140, 42);
        circle(-0.77, 0.07, .2);
        //mid
        glColor3ub(105, 184, 53);
        circle(-0.54, 0.1, .22);
    }
    void roadGrass() {
        //left
        circle(0.1, .005, .07);
        circle(0.2, .002, .04);
        //grass2 road right
        circle(0.8, .005, .035);
        circle(0.88, .005, .065);
    }
    void roadFlower() {
        glColor3ub(247, 183, 225);
        circle(0.025, .006, .013);
        circle(0.014, .025, .013);
        circle(0.0015, .005, .013);
        glColor3ub(255, 3, 3);
        circle(0.012, .012, .01);
        glTranslated(.12, .03, 0);
        glColor3ub(247, 183, 225);
        circle(0.025, .006, .013);
        circle(0.0015, .005, .013);
        glColor3ub(255, 3, 3);
        circle(0.012, .012, .01);
        glLoadIdentity();
        glTranslated(.87, .02, 0);
        glColor3ub(247, 183, 225);
        circle(0.025, .006, .013);
        circle(0.014, .025, .013);
        circle(0.0015, .005, .013);
        glColor3ub(255, 3, 3);
        circle(0.012, .012, .01);
        glLoadIdentity();
    }
    void rainCloud()
    {
        circle(-2.03, .95, .1);
        circle(-1.84, .93, .15);
        circle(-1.54, .93, .18);
        circle(-1.28, .93, .15);
        circle(-1.0, .93, .17);
        circle(-.93, .95, .1);
        circle(-.72, .93, .15);
        circle(-.5, .93, .18);
        circle(-.28, .93, .15);
        circle(0.0, .95, .17);
        circle(.2, .93, .15);
        circle(.43, .93, .2);
        circle(.65, .93, .13);
        circle(.8, .95, .15);
        circle(.95, .95, .08);
    }
```

```
void springFlower() {
    glColor3ub(247, 183, 225);
    circle(0.025, .006, .013);
    circle(0.014, .025, .013);
    circle(0.0015, .005, .013);
    glColor3ub(255, 3, 3);
    circle(0.012, .012, .01);
}
//////////////////////////////////DISPLY FUNCTION///////////////////
void summer();
void autumn();
void winter();
void spring();
void rainDay();
void Idle()
{
    glutPostRedisplay();
}
void display_autumn(int b)
{
    glutDisplayFunc(autumn);
}
void display_winter(int b)
{
    glutDisplayFunc(winter);
}
void display_spring(int b)
{
    glutDisplayFunc(spring);
}
void display_rainyDay(int b)
{
    glutDisplayFunc(rainDay);
}
void display_summer(int b)
{
    glutDisplayFunc(summer);
}
//////////////////////////////////////SUMMER////////////////////////////////////////
void summer() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    ///////sky
    glColor3ub(148, 208, 242);
    sky();
    /////mountain
    mountain();
    //////cloud
    glColor3ub(242, 245, 243);
    glPushMatrix();
    glTranslatef(cloudP, 0.0f, 0.0f);
    cloud();
    glPopMatrix();
    glLoadIdentity();
    ///////sun
    glColor3ub(247, 232, 99);
    circle(0.0, .75, .09);
    ////sun cloud
    glColor3ub(242, 245, 243);
    circle(-.08, .7, .04);
    circle(-.00, .71, .07);
    circle(0.09, .71, .05);
    /////////ground
    glColor3ub(150, 201, 38);
    ground();
    /////////field
    glColor3ub(178, 227, 79);
    field();
    ///////////grass road
    glColor3ub(47, 158, 68);
    roadGrass();
    roadFlower();
    /////////river
    river();
    /////////long Grass///// downside river
    longGrass();
    ////////windmill
    glScalef(0.7, 0.7, 0);
    glTranslated(0.9, 0.2, 0);
    windmill();
    glLoadIdentity();
    //////// house
    glColor3ub(41, 140, 42);
    house();
    //////////tree leaf
    treeLeaf();
    /////////tree
    tree();
    /////////down garss
    downGrass();
    ////////flower down
    downFlower();
    glutTimerFunc(20000, display_rainyDay, 0);
    Idle();
    glFlush();
}
//////////////////////////////////////RAINY_DAY//////////////////////////////////////
void rainDay()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    ////////sky
    glColor3ub(147, 185, 201);
    sky();
    ///////mountain
    mountain();
    ///////Rain cloud
    glColor3ub(175, 182, 186);
    glPushMatrix();
    glTranslatef(RcloudP, 0.0f, 0.0f);
    rainCloud();
    glPopMatrix();
    glLoadIdentity();
    /////////ground
    glColor3ub(130, 179, 82);
    ground();
    /////////filed
    glColor3ub(162, 201, 83);
    field();
    //////////grass road
    glColor3ub(47, 158, 68);
    roadGrass();
    roadFlower();
    //////////river
    river();
    /////////long Grass///// downside river
    longGrass();
    //////////////////////windmill
    glScalef(0.7, 0.7, 0);
    glTranslated(0.9, 0.2, 0);
    windmill();
    glLoadIdentity();
    //////// house
    glColor3ub(47, 158, 68);
    house();
    ///////////////////tree leaf
    treeLeaf();
    tree();
    /////////////down garss
    downGrass();
    downFlower();
    //////////////////rain();
    glPushMatrix();
    glTranslatef(0.0f, rainP, 0.0f);
    glColor3ub(210, 226, 247);
    glLineWidth(2);
    rain();
    glPopMatrix();
    glLoadIdentity();
    glutTimerFunc(20000, display_autumn, 0);
    Idle();
    glFlush();
}
//////////////////////////////////////AUTUMN////////////////////////////////////////
void autumn() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    ////////sky
    glColor3ub(136, 190, 247);
    sky();
    ////////mountain
    mountain();
    //////cloud
    glColor3ub(242, 245, 243);
    glPushMatrix();
```

```
glTranslatef(cloudP, 0.0f, 0.0f);
cloud();
glPopMatrix();
glLoadIdentity();
/////////ground
glColor3ub(150, 201, 38);
ground();
/////////filed
glColor3ub(178, 227, 79);
field();
//////////grass road
glColor3ub(47, 158, 68);
roadGrass();
////////////RIVER SIDE Flower
glScalef(0.25, .2, 0);
glTranslated(-.2, .85, 0);
whiteFlower();
glLoadIdentity();
glScalef(0.25, .2, 0);
glTranslated(.10, .85, 0);
whiteFlower();
glLoadIdentity();
glScalef(0.25, .2, 0);
glTranslated(.80, .85, 0);
whiteFlower();
glLoadIdentity();
glScalef(0.25, .2, 0);
glTranslated(2.5, .86, 0);
whiteFlower();
glLoadIdentity();
glScalef(0.25, .2, 0);
glTranslated(2.9, .86, 0);
whiteFlower();
glLoadIdentity();
//////////river
river();
//////////long grass
longGrass();
///////////   //Flower
glScalef(0.24, .2, 0);
glTranslated(0.8, .49, 0);
whiteFlower();
glLoadIdentity();
//////////////////////windmill
glScalef(0.7, 0.7, 0);
glTranslated(0.9, 0.2, 0);
windmill();
glLoadIdentity();
//////white flower
glScalef(0.35, .35, 0);
glTranslated(1.10, .00, 0);
whiteFlower();
glLoadIdentity();
//////// house
glColor3ub(47, 158, 68);
```

```
    house();
    ////////////////////tree leaf
    treeLeaf();
    tree();
    /////////////white flower
    glScalef(1.3, 1.0, 0);
    glTranslated(-.20, 0.1, 0);
    whiteFlower();
    glLoadIdentity();
    glScalef(1.20, 1., 0);
    glTranslated(.05, .08, 0);
    whiteFlower();
    glLoadIdentity();
    //////////////down garss
    downGrass();
    glutTimerFunc(20000, display_winter, 0);
    glFlush();
    Idle();
}
////////////////////////////////////////////WINTER////////////////////////////////
void winter() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    ////////sky
    glColor3ub(202, 237, 236);
    sky();
    ////////mountain
    mountain();
    ///////mountain snow
    ////main
    glBegin(GL_POLYGON);
    glColor3ub(255, 255, 255);
    glVertex2f(.25, .45);
    glVertex2f(.1, .43);
    glVertex2f(-.142, .2);
    glVertex2f(.04, .28);
    glVertex2f(.07, .18);
    glVertex2f(.17, .28);
    glVertex2f(.28, .15);
    glVertex2f(.35, .26);
    glVertex2f(.585, .15);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3ub(255, 255, 255);
    glVertex2f(-.9, .12);
    glVertex2f(-1.0, .06);
    glVertex2f(-.95, .05);
    glVertex2f(-.89, .075);
    glVertex2f(-.82, .06);
    glEnd();
    glBegin(GL_POLYGON);
    glColor3ub(255, 255, 255);
    glVertex2f(-.55, .15);
    glVertex2f(-.67, .08);
    glVertex2f(-.65, .05);
```

```
glVertex2f(-.55, .08);
glVertex2f(-.4, .06);
glEnd();
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(-.2, .23);
glVertex2f(-.356, .09);
glVertex2f(-.3, .07);
glVertex2f(-.2, .1);
glVertex2f(-.1, .07);
glVertex2f(-.03, .1);
glEnd();
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(.8, .22);
glVertex2f(.68, .12);
glVertex2f(.71, .085);
glVertex2f(.78, .11);
glVertex2f(.85, .08);
glVertex2f(.9, .11);
glVertex2f(.98, .09);
glEnd();
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(.1, .12);
glVertex2f(.0, .06);
glVertex2f(.05, .05);
glVertex2f(.22, .05);
glEnd();
/////////ground
glColor3ub(164, 199, 95);
ground();
/////////filed
glColor3ub(209, 232, 172);
field();
//////////grass road
glColor3ub(75, 166, 55);
roadGrass();
//////////flower
glTranslated(.12, .03, 0);
glColor3ub(247, 183, 225);
circle(0.025, .006, .013);
circle(0.0015, .005, .013);
glColor3ub(255, 3, 3);
circle(0.012, .012, .01);
glLoadIdentity();
//river
glBegin(GL_QUADS);
glColor3ub(150, 201, 38);
glVertex2f(-1, -.00);
glVertex2f(1, -.00);
glVertex2f(1, -.01);
glVertex2f(-1, -.01);
glColor3ub(177, 211, 227);
glVertex2f(-1, -.01);
```

```
glVertex2f(1, -.01);
glVertex2f(1, -.1);
glVertex2f(-1, -.1);
glEnd();
glColor3ub(211, 229, 237);
glLineWidth(.05);
line(-0.85, -.06, -0.95, -.06);
line(-0.5, -.035, -.56, -.035);
line(0.0, -.06, -0.1, -.06);
line(0.8, -.03, 0.9, -.03);
//////////long grass
longGrass();
//////////////////////windmill
glScalef(0.7, 0.7, 0);
glTranslated(0.9, 0.2, 0);
windmill();
glLoadIdentity();
//////// house
glColor3ub(75, 166, 55);
house();
///////////////house snow
glScalef(1.2, 1.2, 0);
glTranslated(.05, 0.0, 0);
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(.39, -.05);
glVertex2f(.375, -.13);
glVertex2f(.43, -.14);
glVertex2f(.46, -.17);
glVertex2f(.5, -.1);
glVertex2f(.53, -.13);
glVertex2f(.725, -.13);
glVertex2f(.71, -.05);
glEnd();
glLoadIdentity();
///////////////////tree
tree();
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(-.89, -.12);
glVertex2f(-.895, -.125);
glVertex2f(-.85, -.15);
glVertex2f(-.785, -.14);
glEnd();
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
glVertex2f(-.78, .02);
glVertex2f(-.8, .022);
glVertex2f(-.75, -.08);
glVertex2f(-.75, -.1);
glVertex2f(-.695, -.145);
glEnd();
glTranslated(-.65, -.5, 0);
glBegin(GL_POLYGON);
glColor3ub(255, 255, 255);
```

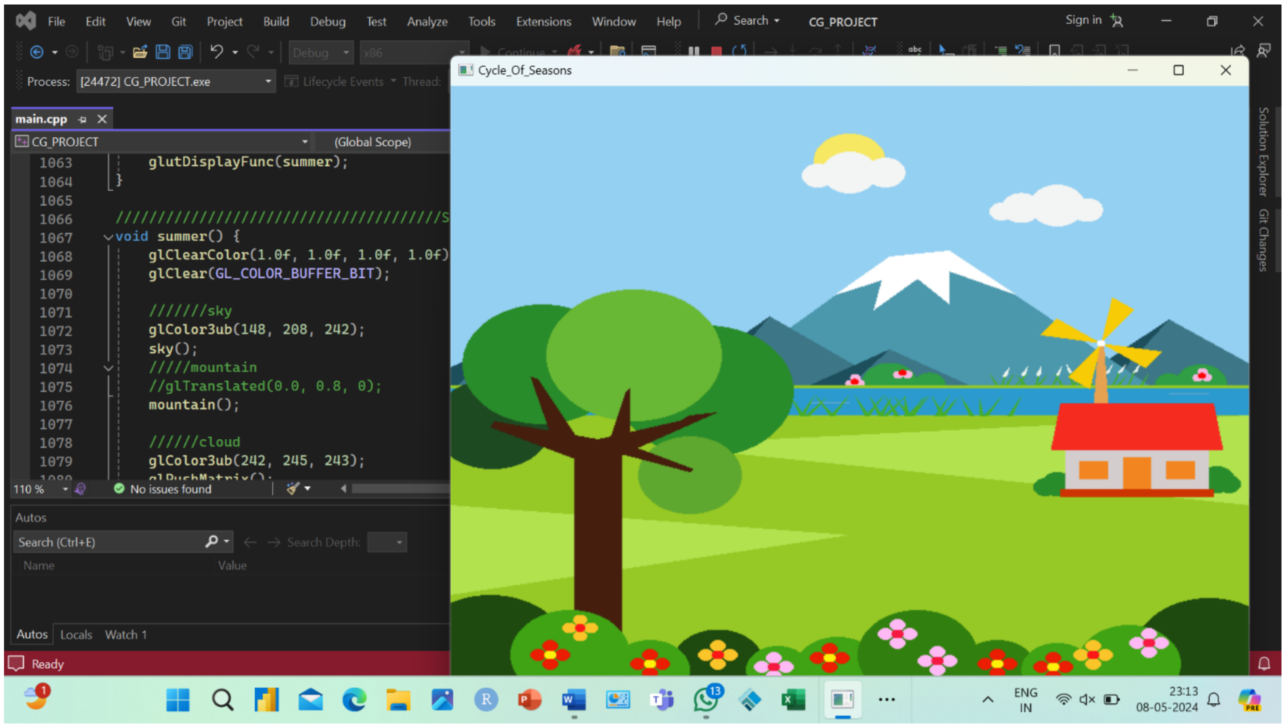
```
        glVertex2f(.1, .49);
        glVertex2f(.087, .475);
        glVertex2f(.055, .35);
        glVertex2f(.1, .43);
        glEnd();
        glLoadIdentity();
        // glTranslated(-.65,-.5,0);
        glBegin(GL_POLYGON);
        glColor3ub(255, 255, 255);
        glVertex2f(-.4, -.1);
        glVertex2f(-.34, -.09);
        glVertex2f(-.5, -.14);
        glVertex2f(-.45, -.15);
        glEnd();
        glBegin(GL_POLYGON);
        glColor3ub(255, 255, 255);
        glVertex2f(-.39, -.28);
        glVertex2f(-.48, -.235);
        glVertex2f(-.45, -.27);
        glEnd();
        glLoadIdentity();
        /////////////down garss
        downGrass();
        // snow
        glPushMatrix();
        glColor3ub(255, 255, 255);
        glTranslated(0.0f, snowP, 0.0f);
        glTranslated(0.2, -0.5, 0); snowball();
        glTranslated(-0.1, -0.45, 0); snowball();
        glTranslated(0.0, -0.5, 0); snowball();
        glTranslated(0.2, -0.5, 0); snowball();
        glTranslated(-0.1, -0.5, 0); snowball();
        //glLoadIdentity();
        glTranslated(0.2, 1.0, 0); snowball();
        glTranslated(0.2, .2, 0); snowball();
        glTranslated(-0.1, 0.3, 0); snowball();
        glTranslated(0.2, .4, 0); snowball();
        //glTranslated(-0.1,.1,0);snowball();
        glLoadIdentity();
        glPopMatrix();
        glLoadIdentity();
        glutTimerFunc(20000, display_spring, 0);
        glFlush();
        Idle();
    }
    //////////////////////////////////////////////////////SPRING/////////////////////
    void spring() {
        glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        ////////sky
        glColor3ub(132, 209, 250);
        sky();
        ////////mountain
        mountain();
        //////cloud
```

```
glColor3ub(242, 245, 243);
glPushMatrix();
glTranslatef(cloudP, 0.0f, 0.0f);
cloud();
glPopMatrix();
glLoadIdentity();
///////sun
glColor3ub(247, 232, 99);
circle(0.0, .75, .09);
////sun cloud
glColor3ub(242, 245, 243);
circle(-.08, .7, .04);
circle(-.00, .71, .07);
circle(0.09, .71, .05);
/////////ground
glColor3ub(120, 201, 38);
ground();
/////////filed
glColor3ub(161, 219, 70);
field();
//////////grass road
glColor3ub(47, 158, 68);
roadGrass();
roadFlower();
//////////river
river();
//////////long grass
longGrass();
//////////////////////windmill
glScalef(0.7, 0.7, 0);
glTranslated(0.9, 0.2, 0);
windmill();
glLoadIdentity();
//////// house
glColor3ub(75, 166, 55);
house();
glLoadIdentity();
///////////////////tree leaf
//right big
glColor3ub(41, 140, 42);
circle(-0.36, -0.05, .13);
//left big
circle(-0.77, 0.07, .15);
//right small
glColor3ub(78, 186, 28);
circle(-0.4, -0.3f, .08);
//mid
circle(-0.54, 0.1, .16);
//left small
glColor3ub(62, 186, 28);
circle(-0.895, -0.13, .08);

//////////////////tree
tree();
//////////////spring flower
```

```
    glTranslated(-0.3, -0.05, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.34, -0.03, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.6, 0.15, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.56, 0.19, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.54, 0.13, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.85, 0.11, 0.0);
    springFlower();
    glLoadIdentity();
    glTranslated(-0.87, 0.07, 0.0);
    glColor3ub(247, 183, 225);
    circle(0.025, .006, .013);
    circle(0.0015, .005, .013);
    glColor3ub(255, 3, 3);
    circle(0.012, .012, .01);
    glLoadIdentity();
    glTranslated(-0.37, -0.3, 0.0);
    glColor3ub(247, 183, 225);
    circle(0.025, .006, .013);
    circle(0.014, .025, .013);
    glColor3ub(255, 3, 3);
    circle(0.012, .012, .01);
    glLoadIdentity();
    ////////////////down garss
    downGrass();
    ///////////////flower down
    downFlower();
    glutTimerFunc(20000, display_summer, 0);
    Idle();
    glFlush();
}
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutCreateWindow("CycleOf_Seasons");
    glutReshapeWindow(800, 600);
    glutDisplayFunc(summer);
    glutTimerFunc(2050, RcloudUp, 0);
    glutTimerFunc(100, cloudUp, 0);
    glutTimerFunc(1400, snowUp, 0);
    glutTimerFunc(100, rainUp, 0);
    glutMainLoop();
    return 0;
}
```

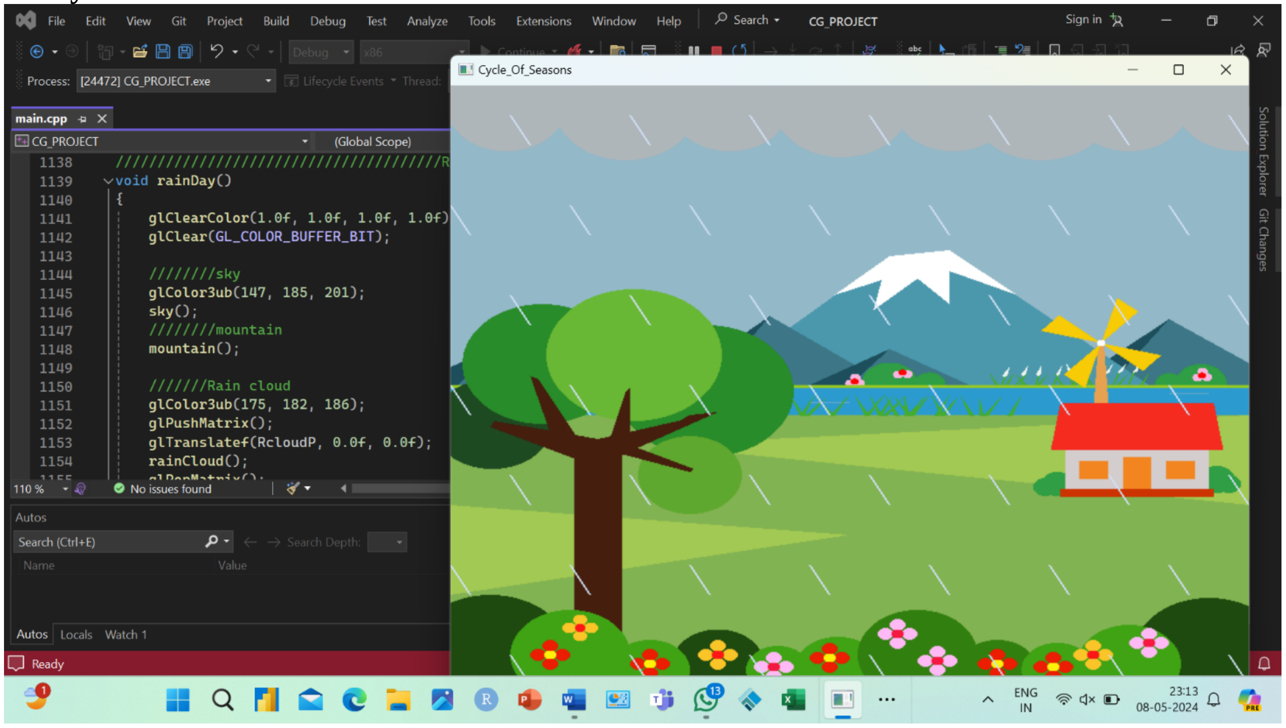
Output Screenshots

Summer:



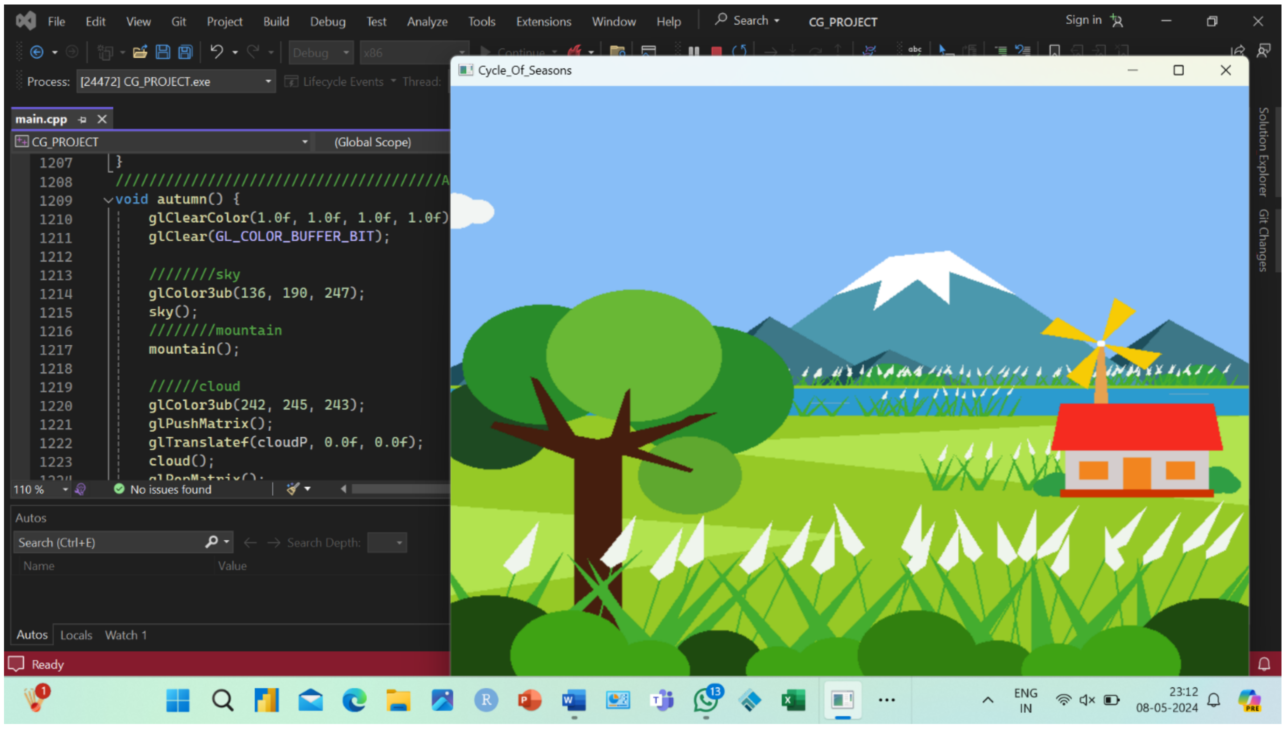
```
    glutDisplayFunc(summer);
}

////////////////////////////////////S
void summer() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);

    ///////sky
    glColor3ub(148, 208, 242);
    sky();
    /////mountain
    //glTranslated(0.0, 0.8, 0);
    mountain();

    //////cloud
    glColor3ub(242, 245, 243);
    glPushMatrix();
```

Rainy:



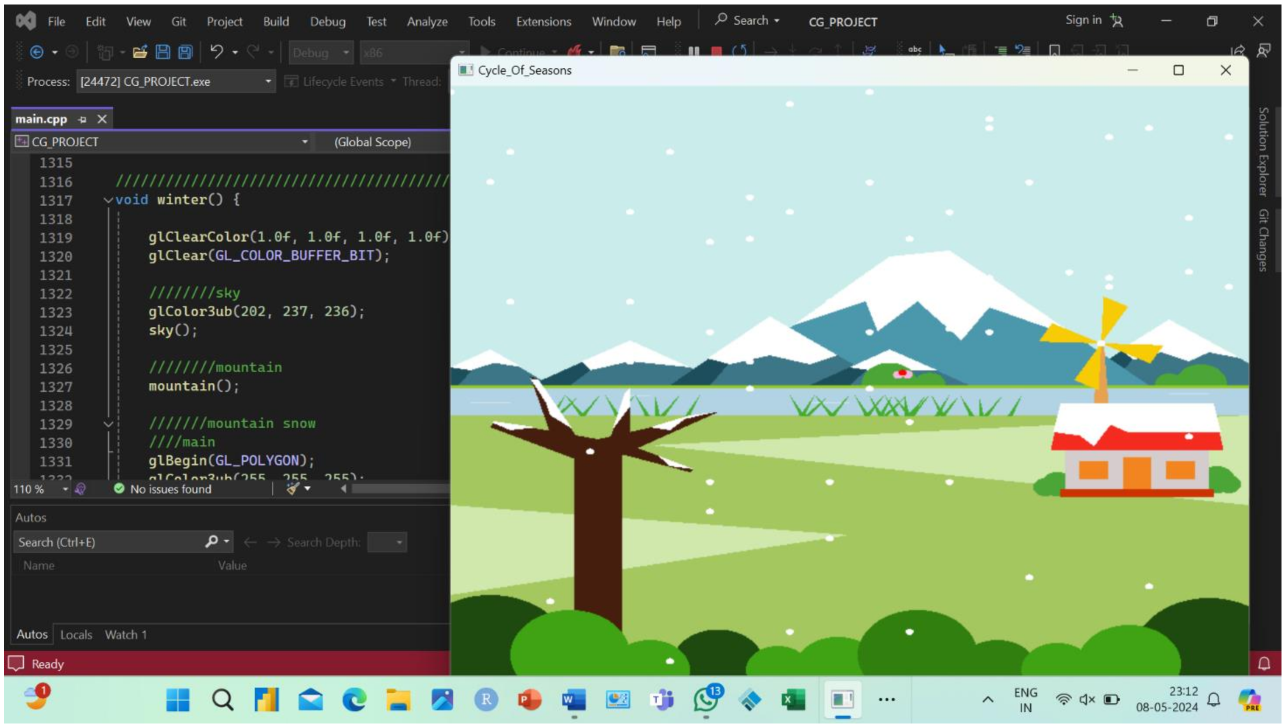
```
////////////////////////////////////R
void rainDay()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);

    ////////sky
    glColor3ub(147, 185, 201);
    sky();
    ////////mountain
    mountain();

    //////Rain cloud
    glColor3ub(175, 182, 186);
    glPushMatrix();
    glTranslatef(RcloudP, 0.0f, 0.0f);
    rainCloud();
    glPopMatrix();
```

Autumn:



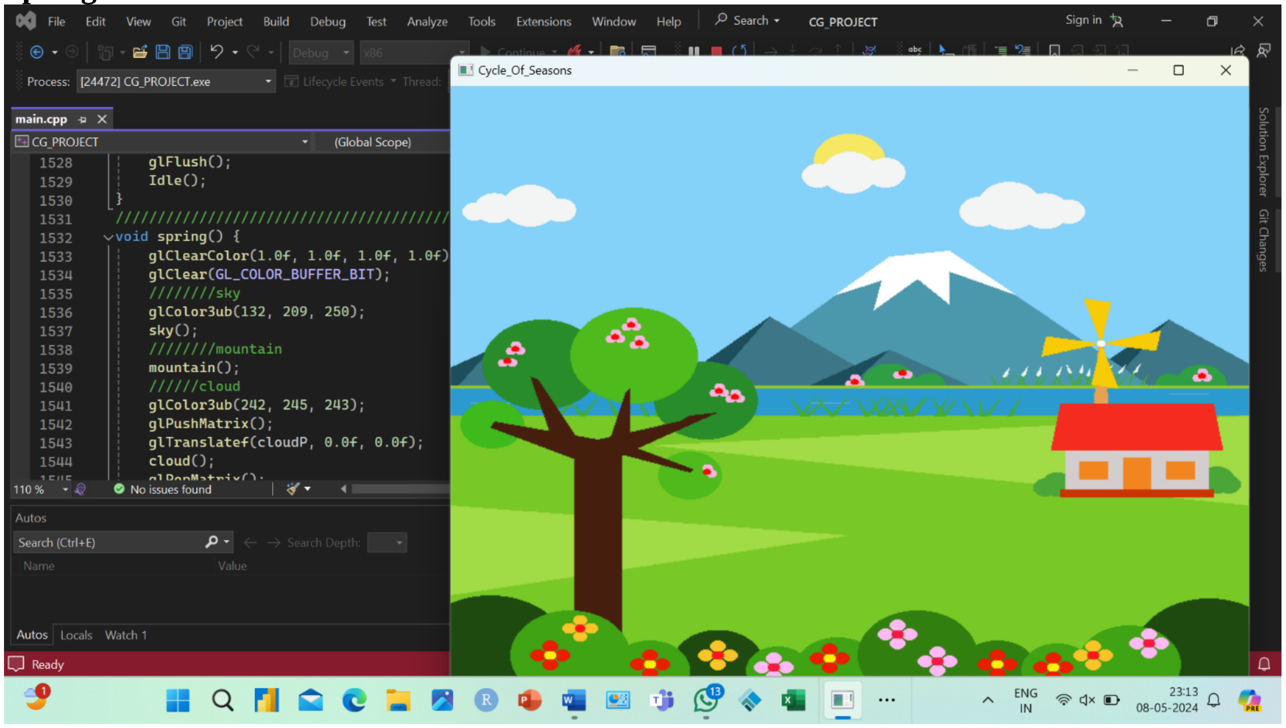
```
}
//////////////////////////////////////A
void autumn() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);

    ////////sky
    glColor3ub(136, 190, 247);
    sky();
    ////////mountain
    mountain();

    //////cloud
    glColor3ub(242, 245, 243);
    glPushMatrix();
    glTranslatef(cloudP, 0.0f, 0.0f);
    cloud();
```

Winter with Snowfall:



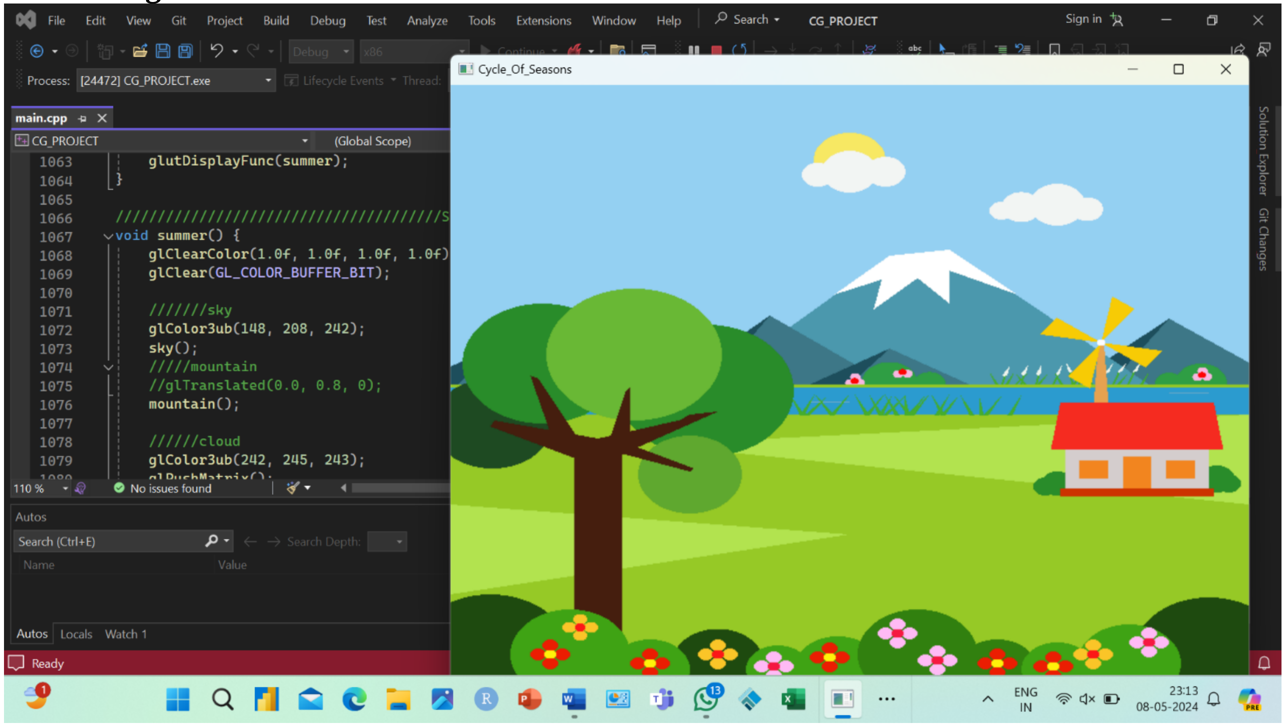
```

////////////////////////////////////////
void winter() {

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);

    ////////sky
    glColor3ub(202, 237, 236);
    sky();

    ////////mountain
    mountain();

    //////mountain snow
    ////main
    glBegin(GL_POLYGON);
```

Spring:



```
    glFlush();
    Idle();
}
///////////////////////////////////////
void spring() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);
    ////////sky
    glColor3ub(132, 209, 250);
    sky();
    ///////mountain
    mountain();
    /////cloud
    glColor3ub(242, 245, 243);
    glPushMatrix();
    glTranslatef(cloudP, 0.0f, 0.0f);
    cloud();
```

Summer again:



```
    glutDisplayFunc(summer);
}

////////////////////////////////////S
void summer() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f)
    glClear(GL_COLOR_BUFFER_BIT);

    //////sky
    glColor3ub(148, 208, 242);
    sky();
    /////mountain
    //glTranslated(0.0, 0.8, 0);
    mountain();

    /////cloud
    glColor3ub(242, 245, 243);
```

And the cycle goes on!

****