VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum- 590014

Computer Graphics Project Report

On

"SIMULATION OF WEATHER CONDITIONS"

Submitted in partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering
in
Computer Science And Engineering
Of
Visvesvaraya Technological University,Belgaum

Submitted By

NAME :J SRI SAI SANGEETHA
USN:1CD16CS050

NAME : HARPREETH KAUR J
USN : 1CD16CS047

Under the Guidance of

Mrs. Jayanthi M.G
Assistant Professor,
CSE Department,CITech

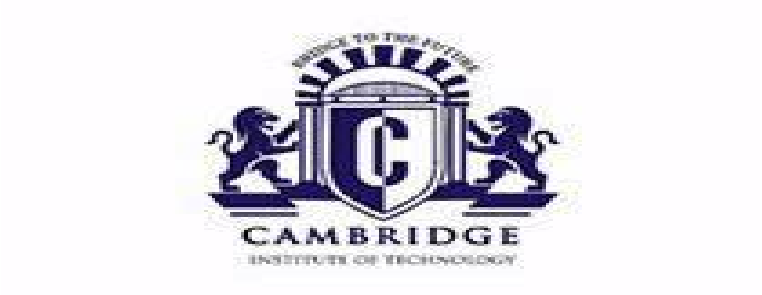
Mr. Ganesh D.R Associate Professor,
CSE Department,CITech

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K R Puram,Bengaluru-560036

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Computer Graphics mini project work entitled "**Simulation Of Weather Conditions"** has been carried out by **Ms. J Sri Sai Sangeetha(1CD16CS050)** and **Ms. Harpreeth Kaur J(1CD16CS047)** are bonafide students of Cambridge Institute of Technology in partial fulfillment for **Sixth Semester** of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University ,Belgaum during the year **2019-2020**. It is certified that all corrections/suggestions that Are indicated for the internal Assesment have been Incorporated in the report deposited In the departmental library. The Graphics Project Report has been approved as it likely Satisfies the academic requirements in Respect of project work prescribed for the said semester.

Internal Project Guides:

1) Mrs. Jayanthi M.G
Dept.of CSE,CITech

2) Mr. Ganesh D.R
Dept. of CSE, CItech

HOD, Dept of CSE

Dr. ShashiKumar D.R
Dept. Of CSE, CITech

Examiners:

1) **2)**

ABSTRACT

‘A picture is worth a thousand words’ goes the ancient Chinese proverb. This has become a cliché In our society after the advent of inexpensive and simple techniques for producing pictures.Computers have become a powerful medium for the rapid and economical production of pictures.There is virtually no area in which graphical displays cannot be used to some advantage. GraphicsProvide a natural means of communicating with computer that they have become so widespread.The fields in which Computer Graphics find their uses are many. Some of them being UserInterfaces, Computer Aided Design, Office automation, Desktop Publishing, Plotting of Mathematical, scientific or industrial data, Simulation, Arts, Presentations, Cartography to name a Few.

Here we have tried to incorporate and Present various simulations of Weather Conditions in 2D,Using OPENGL libraries in Dev-c++ software.

The basic Weather Simulator works with a key-logger that enables the user to provide their input accordingly. Mouse-logger is also used to control the movements of objects in the scene. Here,We created a scene consisting of several objects in a village, on which different weather changes.Are shown based on the season selected through the input. This simulator enables the users to differenciate and analyze different seasonal changes accordingly.

ACKNOWLEDGMENT

We would like to place our record a deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible.

I am extremely thankful to **Dr. Suresh L,** Principal, CITech, Bangalore, for providing me the academic ambience and everlasting motivation to carry out this work and shaping our careers.

I express my sincere gratitude to **Dr. Shashikumar D.R.,** HOD, Dept. of Computer Science and Engineering, CITech, Bangalore, for his stimulating guidance, continuous encouragement and motivation throughout the course of present work.

I also wish to extend my thanks to **Mrs. Jayanthi M.G,** Associate Professor, Project Coordinator, Dept. of CSE, CITech, Bangalore, for there critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

I also wish to extend my thanks to **Mr. Ganesh D.R,** Assistant Professor Dept. of CSE ,CITech for his/her guidance and impressive technical suggestions to complete my seminar work.

Finally to all my friends, classmates who always stood by me in difficult situations also helped me in some technical aspects and last but not the least I wish to express deepest sense of gratitude to my parents who were a constant source of encouragement and stood by me as pillar of strength for completing this work successfully.

J SRI SAI SANGEETHA (1CD16CS050)

HARPREETH KAUR J (1CD16CS047)

CONTENTS

Chapter 1

INTRODUCTION

Although computer graphics is a vast field that encompasses almost any graphical aspect, we are mainly interested in the generation of images of 2-dimensional scenes. Computer imagery has applications for film special effects, simulation and training, games, medical imagery, flying logos, etc. Computer graphics relies on an internal model of the scene, that is, a mathematical representation suitable for graphical computations. One such real life scene we found interesting is the working of the Simulation of Weather Conditions and we have tried to incorporate this in this project.

Weather Simulator is used as a analyzer to differentiate various weather conditions based on the the input given from key-logger.

It works using the principle of seasonal changes and reflection of those changes are viewed on an object present in the scene, creating a realistic weather simulation in a virtual environment. Here we simulate different types of weather using traditional rendering techniques on well defined objects.

This would lead to interesting animations such as impact of rain and snow on various objects and interactions between them in the environment.

The main advantages of Weather Simulator are as follows:

- It helps the user to differentiate between various seasonal changes.
- Regions can be evacuated if hurricanes or floods are expected
- Farmers can know when to plant or harvest their crops
- People can choose where and when to take their holidays by looking through good weather conditions

The Weather Simulator Performs Following Functions:

The simulator collects as much of data as possible about the current state of atmosphere particularly like temperature, humidity and wind and renders the simulation of changes onto the objects present on the screen to determine how atmosphere evolves in the future.

Numerical weather prediction models are computer simulations of the atmosphere.

The chaotic nature of the atmosphere and incomplete understanding of the process means that interactions between the objects have become less accurate.

The ongoing climate changes make it increasingly difficult to predict certain aspects of weather according to a new study from Stockholm university. Hence this simulator makes reliable weather forecasts tremendously important across the globe.

Hence the output from the simulator provides the basis of the weather prediction where he input is provided by the user through the key-board logger.

Chapter 2

LITERATURE SURVEY

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptual or abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive- the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction. A Bitmap is an ones and zeros representation of points (pixels, short for 'picture elements') on the screen. Bitmap graphics provide easy-to-use and inexpensive graphics based applications.

The concept of 'desktop' is a popular metaphor for organizing screen space. By means of a window manager, the user can create, position, and resize rectangular screen areas, called windows, that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern – recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many design, implementation, and construction processes, the information pictures can give is virtually indispensable.

Computer graphics is the creation and manipulation of pictures with the aid of computers. It is divided into two broad classes:

- Non-Interactive Graphics.
- Interactive Graphics.

2.1.1 Non-Interactive graphics –

This is a type of graphics where observer has no control over the pictures produced on the screen. It is also called as Passive graphics.

2.1.2 Interactive Graphics-

This is the type of computer graphics in which the user can control the pictures produced. It involves two-way communication between user and computer. The computer upon receiving signal from the input device can modify the displayed picture appropriately. To the user it appears that the picture changes instantaneously in response to his commands. The following fig. shows the basic graphics system:

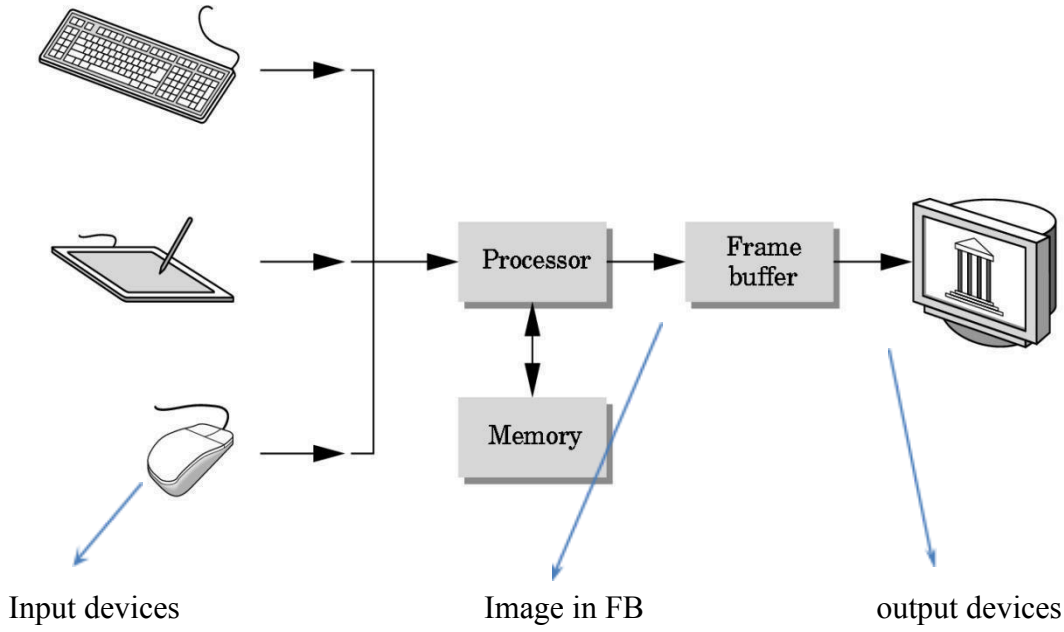


Fig 2.1: Basic Graphics System.

2.2 About OpenGL -

OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

In Computer graphics, a 3-dimensional primitive can be anything from a single point to an ‘n’ sided polygon. From the software standpoint, primitives utilize the basic 3-dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

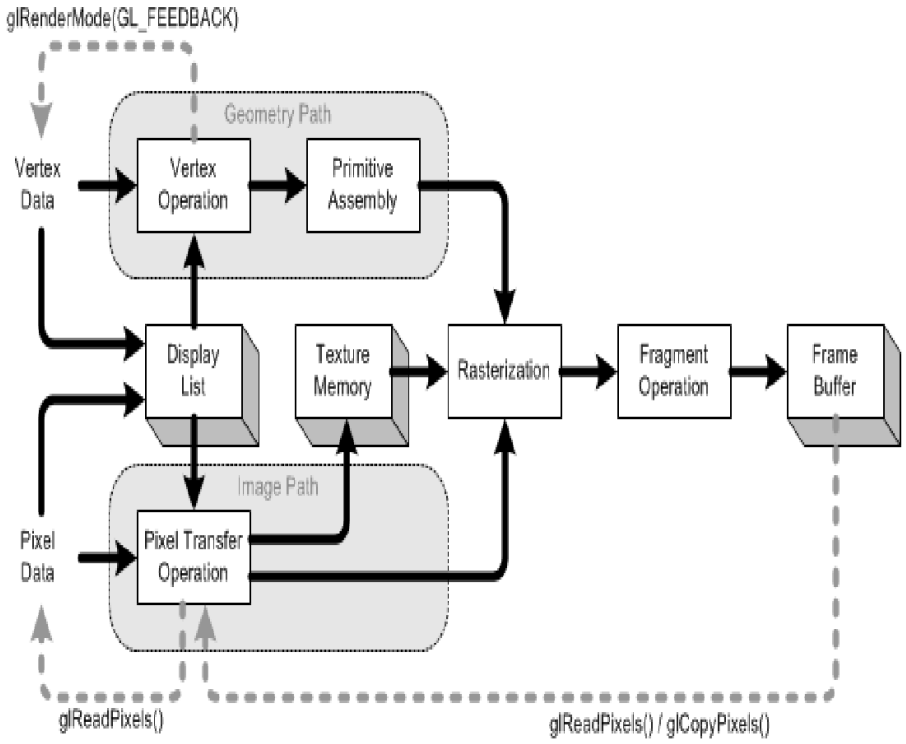


Fig 2.2: The OpenGL rendering pipeline.

2.3 Advantages of OpenGL

With different 3D accelerators, by presenting the programmer to hide the complexities of interfacing with a single, uniform API.

To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Hardware requirements- Pentium 3 or higher processor, 128 MB or more RAM, a standard keyboard. If the user wants to save the created files a secondary storage medium can be used.

3.2 Software requirements- The graphics package has been designed for Dev C++ Application, hence the machine must have that installed preferably 5.0 or later versions with the glut library functions installed. Other graphics libraries are used.

3.3 Development platform- Dev C++

3.4 Language used in coding- C++ language.

3.5 Functional Requirements-

The whole project is divided into many small parts known as functions and these functions would take care of implementing particular parts of this project which makes the programming easy and effective. Each function takes the responsibility of designing the assigned parts hence we combined all the functions at a particular stage to get the final required design. The functions that are used to implement the Weather Simulator are:

- **void car () -** This function depicts the bus by drawing some horizontal, vertical lines and polygons on the window and translating them in the direction of that of the road. This function makes use of the OpenGL functions to define the window size, length of the horizontal lines, to provide the color for the car and make them to translate in required direction.
- **void cloudB () -** This function depicts a bigger cloud in the scene. This function designs the cloud by merging several circles. It defines the structure of the cloud.
- **void a()-** This function is used to draw a mini cloud in the scene.
- **void c()-** This function is used to draw a single cloud in the scene.

- **void cloud()-** This function is used to merge three clouds at a time.
- **void day () -** This function depicts the scene. Polygons are created by plotting the points at the proper distances to resemble the background of the scene and then these points are joined with lines to

make the background. This function use the OpenGL inbuilt functions especially for plotting and joining the points.

- **void ground () -** This function is used to draw the road in the scene. It is created by plotting the points at the proper distances and joining the lines using OpenGL functions.
- **void divider () -** This function is used to draw the dividers on the road.
- **void tree () -** This function is used to draw tree objects in the scene using several clouds and line segments.
- **void house()-** This function is used to draw a house using polygons and lines.
- **void flower()-** This function displays the flowers on the tree in fall season.
- **void grass()-** This function draws the grass on the grass and surroundings of the house.
- **void fence()-** This function provides the fencing on both the ends of the road.
- **void bird()-** This function is used to draw the bird object in the sky.
- **int main (int argc, char ** argv) -** This function is used to initialize the display function using glutDisplay() openGL method.
- **void display()-** In this function first we should print the instructions that we would be displayed on the pop-up window.
- **Void handleKeypress (unsigned char key, int x, int y) -** This function used to provide the keyboard interface to the user. Using this function we can change the seasons we want to choose. For example we can choose ‘s’ for summer season and ‘f’ for spring season etc…

Chapter 4

ALGORITHM DESIGN AND ANALYSIS

4.1 Pseudo Code:

The main algorithm for the Lift-Over Bridge can be written as below:

- Initially define the void day() method and keep running it throughout the course of the program
- Initialize the void car(), void cloudB(),void cloud(),void ground(),void tree(),void divider(),void house(),void fence() and void bird () functions.
- Initialize the keyboard and mouse interface functions.
- Keep the above scenery in place on the window and wait for the user to press the mouse button
- As soon as the above step is performed, the following actions should be performed simultaneously:
    - All the objects should be initialized and start moving.
    - The corresponding window associated with each of the season choosen through the keyboard should be displayed.
    - All the properties of the objects should be displayed according to the season.
    - Appropriate color of the sky changes when the season chosen is changed.
- The objects should move continuously.
- The user can press ‘m’ to understand the different keys used to operate on different seasons
- The user can control the speed of movement of objects through mouse clicks, left click of mouse decreases the speed of the moving object and right click increases the speed of the moving object
- During any time of execution of the program, if the user presses the exit button the window should exit immediately.

4.2 Analysis:

The functions and their algorithms are as follows:

- **void day ():**

Since we need the blue sky, we've use the combination of green (0.5) and blue (1.0).The POLYGON function serves the purpose of covering the entire screen with blue color starting from the vertices (0, 0) to (1350, 0) and then from (1350, 690) to (690, 0).Then the black lines (1, 1, and 1) that represent the waves continuously translate from 0-2000 at a distance of 100 units from each other.

- **void ground():**

This function represents the road structure in a combination of black and grey colors. They have been drawn using the GL_POLYGON function with edges in combination.

- **void car():**

We have translated a bus over the bridge. Constructing the bus was again a challenge but was easier to implement once we had got the boat done. It was implemented by drawing a set of regular polygons and then merging them in parts to look like a bus. We used 3 sets each consisting of 4 points each to get the layout followed by a set of 16 points for the carrier. Then came the set of headlights with 2 points each followed by a set of 2 points for the horn grill and finally another couple of points for the side windows.

- **void house():**

This function represents a house structure in a combination of brown, blue and golden colors. They have been drawn using the GL_POLYGON function with edges in combination .

- **Void flower():**

This function draws flowers in combination of various colors during the fall season.

- **Void display():**

This function basically displays all the above described functions on the screen as we flush the output onto the screen form the frame buffer.

- **Void keyboard():**

This function basically changes the color of the boat as we have implemented the suitable color codes for their respective colors. The colors that the boat can have are red, green, blue, yellow, cyan and magenta.

The main concept behind the usage of the above 6 colors are that they are the additive and the subtractive colors:

❖ **Additive color**:

- Form a color by adding amounts of three primaries
- CRTs, projection systems, positive film
- Primaries are Red (R), Green (G), Blue (B)

❖ **Subtractive color:**

- Form a color by filtering white light with cyan (C), Magenta (M), and Yellow (Y) filters
- Light-material interactions
- Printing
- Negative film

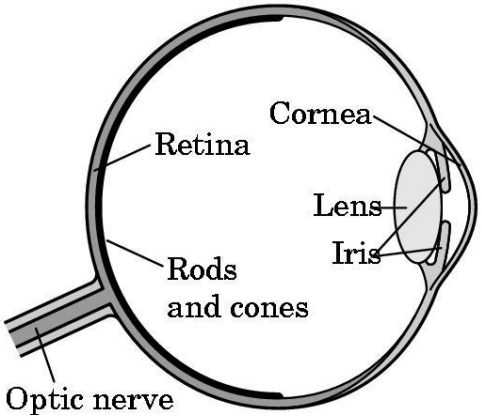


Fig 4.1

The above figure of the eye shows the Rhodes and cones. Rhodes is responsible to detect the additive colors and the cones are in charge of detecting the subtractive colors.

- **Void main ():**

This function puts the whole program together. It says which function to execute first and which one at the end. However, here we have used int main () since eclipse expects the main to have a return value.

Chapter 5

IMPLEMENTATION

The Weather Simulator can be implemented using some of the OpenGL inbuilt functions along with some user defined functions. The inbuilt OpenGL functions that are used mentioned under the FUNCTIONS USED category. The user defined functions are mentioned under USER DEFINED FUNCTIONS category.

4.1 Functions Used –

- **Void glColor3f (float red, float green, float blue):**

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

- **Void glClearColor(int red, int green, int blue, int alpha):**

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

- **Void glutKeyboardFunc():**

Where func () is the new keyboard callback function. glutKeyboardFunc sets the keyboard callback for the *current window*. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window for relative coordinates when the key gets pressed. Prototype is as given below:

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

When a new window is created, no keyboard callback is initially registered, and the ASCII keystrokes that are within the output window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

- **Void GLflush():**

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

- **Void glMatrixMode(GLenum mode):**

Where "mode" specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted are:

GL_MODELVIEW, *GL_PROJECTION and GL_TEXTURE*

The initial value is *GL_MODELVIEW*.

The function *GlMatrixMode* sets the current matrix mode. *Mode* can assume one of these values:

GL_MODELVIEW : Applies matrix operations to the model view matrix stack.

GL_PROJECTION: Applies matrix operations to the projection matrix stack.

- **void viewport(GLint x, GLint y, GLsizei width, GLsizei height):**

Here, (x, y) specifies the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0).

Width, height: Specifies the width and height of the viewport. When a GL context is first attached to a surface (e.g. window), width and height are set to the dimensions of that surface.

Viewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (x_{nd}, y_{nd}) be normalized device coordinates. Then the window coordinates (x_w, y_w) are computed as follows:

$$x_w = (x_{nd} + 1)\ {}^{width}/_{2} + x$$

$$y_w = (y_{nd} + 1)\ {}^{height}/_{2} + y$$

Viewport width and height are silently clamped to a range that depends on the implementation. To query this range, we call the Glgetinteger with argument *GL_MAX_VIEWPORT_DIMS.*

- **void glutInit (int *argc, char **argv):**

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

- **void glutReshapeFunc (void (*func) (int width, int height)):**

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

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

- **glOrtho ( ):**

Syntax: *void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);*

The function defines an orthographic viewing volume with all parameters measured from the center of the projection plane.

- **void glutMainLoop(void);**

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

- **glutPostRedisplay():**

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

4.2USER DEFINED FUNCTIONS:

- **void car () -** This function depicts the bus by drawing some horizontal, vertical lines and polygons on the window and translating them in the direction of that of the road. This function makes use of the OpenGL functions to define the window size, length of the horizontal lines, to provide the color for the car and make them to translate in required direction.
- **void cloudB () -** This function depicts a bigger cloud in the scene. This function designs the cloud by merging several circles. It defines the structure of the cloud.
- **void a()-** This function is used to draw a mini cloud in the scene.
- **void c()-** This function is used to draw a single cloud in the scene.
- **void cloud()-** This function is used to merge three clouds at a time.
- **void day () -** This function depicts the scene. Polygons are created by plotting the points at the proper distances to resemble the background of the scene and then these points are joined with lines to

make the background. This function use the OpenGL inbuilt functions especially for plotting and joining the points.

- **void ground () -** This function is used to draw the road in the scene. It is created by plotting the points at the proper distances and joining the lines using OpenGL functions.
- **void divider () -** This function is used to draw the dividers on the road.

- **void tree () -** This function is used to draw tree objects in the scene using several clouds and line segments.
-

- **void house()-** This function is used to draw a house using polygons and lines.
- **void flower()-** This function displays the flowers on the tree in fall season.
- **void grass()-** This function draws the grass on the grass and surroundings of the house.
- **void fence()-** This function provides the fencing on both the ends of the road.
- **void bird()-** This function is used to draw the bird object in the sky.
- **int main (int argc, char ** argv) -** This function is used to initialize the display function using glutDisplay() openGL method.

- **void display()-** In this function first we should print the instructions that we would be displayed on the pop-up window.

- **Void handleKeypress (unsigned char key, int x, int y) -** This function used to provide the keyboard interface to the user. Using this function we can change the seasons we want to choose. For example we can choose ‘s’ for summer season and ‘f’ for spring season etc…

Chapter 6

DISCUSSIONS AND SNAPSHOTS

6.1 DISCUSSIONS:

Before we take up this topic, there we are many other practical topics that we were thinking to design using the OpenGL functions. For example DOS Paint shop, Text editors which we would use in our day to day activities. Unfortunately we couldn't take up those topics since we were not able to meet the requirement at this particular stage. This made us to search for a simple and practical topic to design.

In beginning we tried to analyze topics that can be designed using simple OpenGL functions. In this way successfully we found many topics, the Weather Simulator was one of that.

One of the most famous examples of the bascule type is the Time Series Weather Radar Simulator, which is capable of generating simulated raw time series data for a weather. A field of thousands of scatters is populated within the field of view of the virtual radar. Thus, this new radar simulator allows for the test and analysis of advanced topics, such as phased array antennas, clutter mitigation schemes, waveforms design studies and spectral based models. After selecting this project, the very first step that we did was to note down the possible functions that we would require implementing this scenario.

The very first object that we had to design was a polygon which was the base for all the remaining objects. The polygon was constructed as mentioned below.

- Using the predefined method glBegin(GL_POLYGONS) we have constructed a polygon by providing four points.
- four polygons were constructed in this way for sky, houses, road and trees.

Our next goal was to design the objects and place it on the polygons created. We had no idea about the way in which the objects were to be created and placed . So we went through with many examples of designing polygons, circles and various shapes and we applied the same logic. Slowly we tried to design various objects using the above logic.

First we decided to build a 2D house in OpenGL using this logic and we followed the following sequence:

- First we made different shapes, which are used later for making a whole house starting from a point to joining it with rest of the points.
- We created a variable y that holds a value and then added it to each of the vertex say glVertex2i(0,0+y);
- By adding y we just moved the point vertically upward in direction of Y-axis.
- The same way we added x variable to each vertex to move house horizontally.

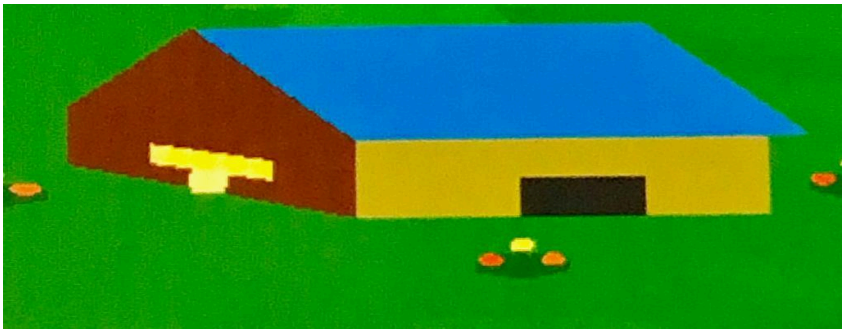


Fig 6.1.1

- Our next part was to develop a moving car. Here we used a simple polygon –triangle, quadrilateral and circle.
- Firstly we created a function drawBall() to design the wheel of the car, later we used init() functions, window resize function, simple keyboard and main functions.

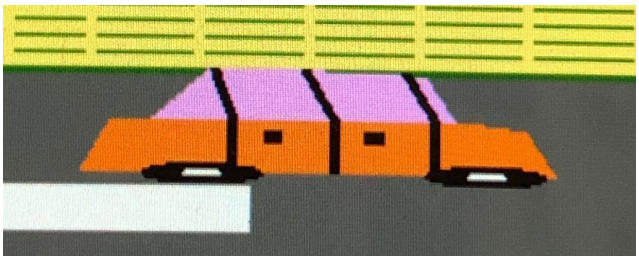


Fig 6.1.2

Rest of the things i.e. using the mouse and keyboard interface for the user was provided by writing

separate functions and by calling the user defined functions in proper manner.

The whole project gave some remarkable experience for us. From the beginning itself we had lots of confusions in our mind, whether the project would work as per our requirement or not. When we made an attempt to run the project as normal we encountered with many syntax errors, parameter mismatch and so on. Debugging those errors was not a difficult task for us because of the beautiful platform provided by Dev-c++. Once we made the corrections we execute the project at the first time the output was not exactly what we wanted so again made some changes in user defined functions to get the desired action.

6.2 SNAPSHOTS:

- **Snap Shot of initial stage of Weather Simulator:**

Fig 6.2.1

- **Snap Shot representing summer season:**

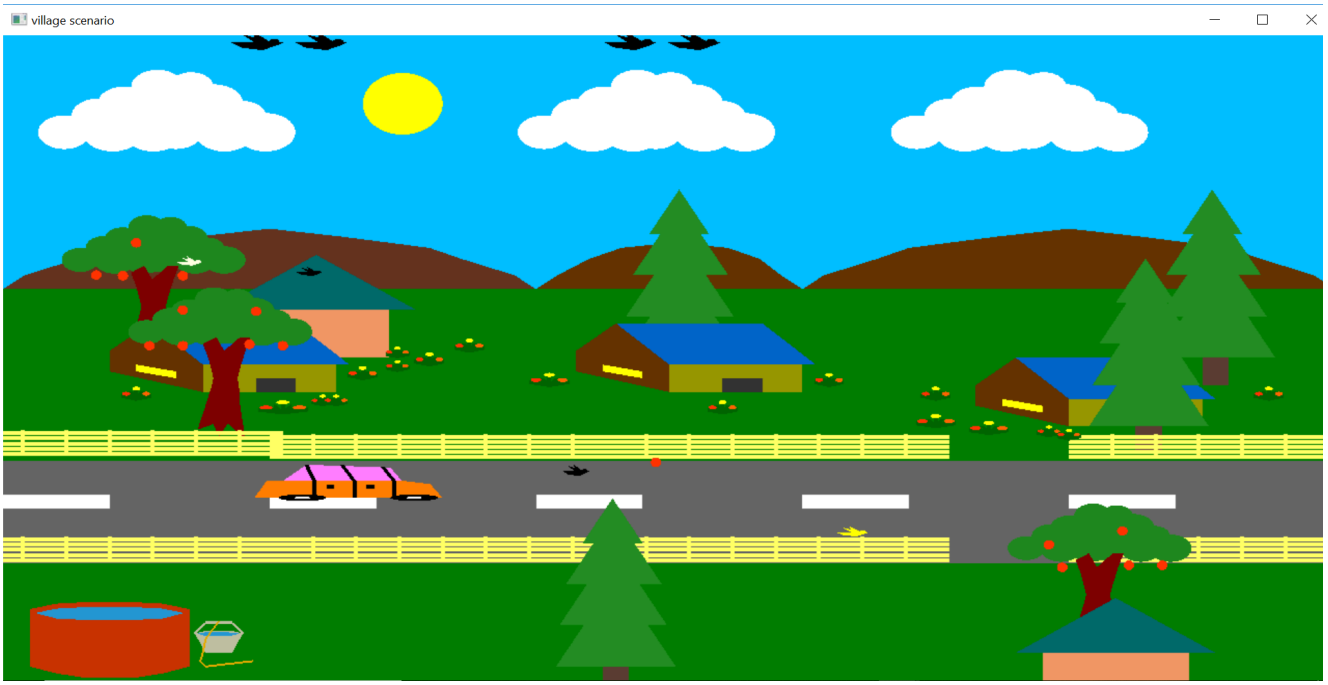


Fig 6.2.2

- **Snap Shot representing winter season:**

Fig 6.2.3

- **Snap Shot representing spring season:**

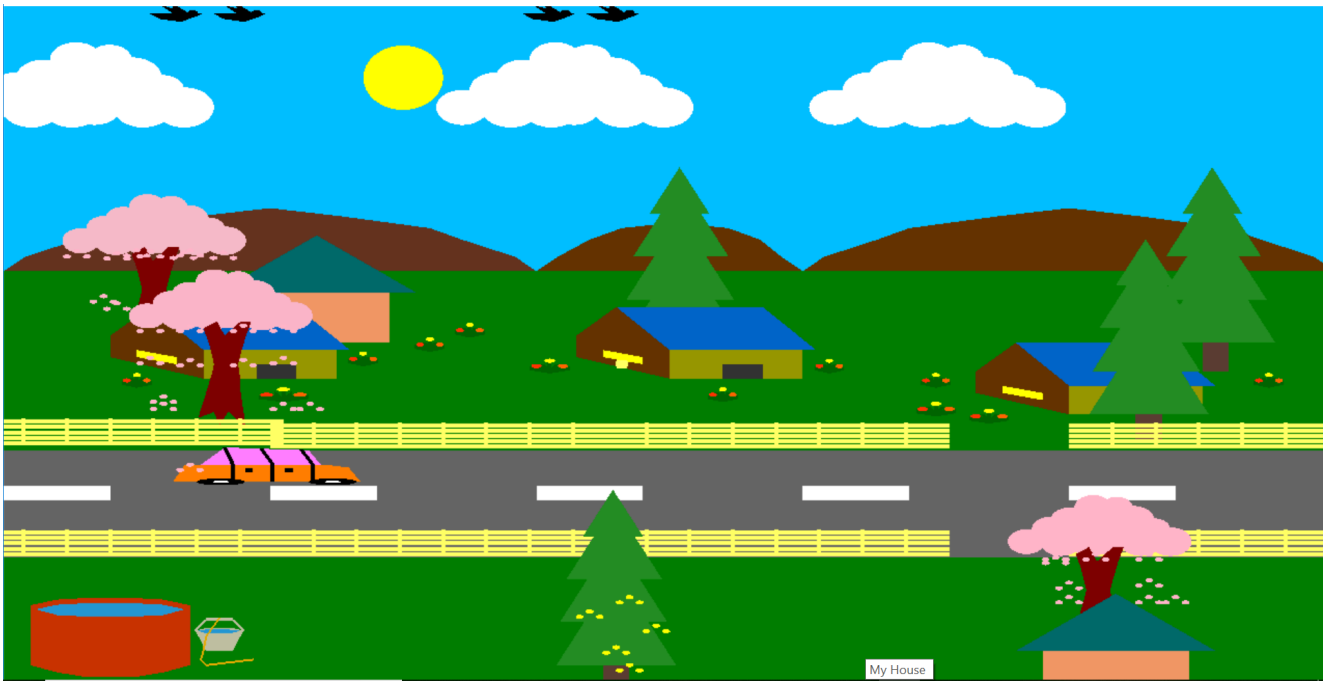


Fig 6.2.4

- **Snap Shot representing rainy season:**

Fig 6.2.5

Chapter 7:

CONCLUSIONS AND FUTURE SCOPE

7.1 General Constraints:

- As the software is being built to run on Dev-C++ Software, which gives access to limited conventional memory, the efficient use of the memory is very important.
- As the program needs to be run even on low-end machines the code should be efficient and optimal with the minimal redundancies.
- Needless to say, the computation of algorithms should also be robust and fast.
- It is built assuming that the standard output device (monitor) supports colors.

7.2Assumptions and Dependencies:

- One of the assumptions made in the program is that the required libraries like GL, GLU and glut have been included.
- The user's system is required to have the C compiler of the appropriate version.
- The system is also expected to have a keyboard and mouse connected since we provide the inputs via these devices.

7.3 Further Enhancements:

The following are some of the features that can be included in the revised versions of this code are:

- Sounds of rain, car, snow can be incorporated.
- Support for different types of vehicles all moving simultaneously on the road.
- Support for advanced 3D representation of the entire scenario.
- Support for transparency of layers and originality.

Chapter 8:

BIBLIOGRAPHY

8.1 Book References:

- Ariponnammal, S. and Natarajan, S. (1994) 'Transport Phenomena of Sm Sel-X Asx', Pramana – Journal of Physics Vol.42, No.1, pp.421-425.
- Barnard, R.W. and Kellogg, C. (1980) 'Applications of Convolution Operators to Problems in Univalent Function Theory', Michigan Mach, J., Vol.27, pp.81–94.
- Shin, K.G. and McKay, N.D. (1984) 'Open Loop Minimum Time Control of Mechanical Manipulations and its Applications', Proc.Amer.Contr.Conf., San Diego, CA, pp. 1231-1236.

8.2 Web References:

- www.opengl.org
- www.google.com
- www.sourcecode.com
- www.pearsoned.co.in
- www.wikipedia.org

Chapter 9

APPENDICES

9.1 Source Code:

```
#include <iostream>
#include<stdio.h>
 #include<stdlib.h>
 #include<math.h>
#include<GL/glut.h>
#include<windows.h>
#define PI 3.1416
   GLint i;
 GLfloat cx=0,str=500.0,mn=500.0;
 GLfloat sr=0.0,sg=0.749,sb=1.0;

void circle(GLdouble rad) {
 GLint points = 50;
 GLdouble delTheta = (2.0 * PI) / (GLdouble)points;
  GLdouble theta = 0.0;
    glBegin(GL_POLYGON);
 for( i = 0; i <=50; i++, theta += delTheta )
  {
 glVertex2f(rad * cos(theta),rad * sin(theta));
  }
 glEnd();
  }
void car()
{
glColor3f(1.0, 0.5, 0.0);
glBegin(GL_POLYGON);
glVertex3f(-0.52, 0.2, 0.0);
glVertex3f(-0.9, 0.2, 0.0);
glVertex3f(-0.87, 0.5, 0.0);
glVertex3f(-0.52, 0.5, 0.0);
glEnd();

glColor3f(1.0, 0.5, 0.0);
glBegin(GL_POLYGON);
   glVertex3f(-0.43, 0.2, 0.0);
glVertex3f(-0.52, 0.2, 0.0);
glVertex3f(-0.52, 0.5, 0.0);
glVertex3f(-0.46, 0.45,0.0);
glEnd();

glColor3f(1.0, 0.5, 1.0);
glBegin(GL_POLYGON);
glVertex3f(-0.53, 0.5, 0.0);
   glVertex3f(-0.83, 0.5, 0.0);
glVertex3f(-0.77, 0.8, 0.0);
   glVertex3f(-0.56, 0.75, 0.0);
glEnd();

   glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.65, 0.2, 0.0);
glVertex3f(-0.65, 0.5, 0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.65, 0.5, 0.0);
glVertex3f(-0.68, 0.79, 0.0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.75, 0.2, 0.0);
glVertex3f(-0.75, 0.5, 0.0);
glEnd();
```

```
glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.75, 0.5, 0.0);
glVertex3f(-0.77, 0.8, 0.0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.55, 0.2, 0.0);
glVertex3f(-0.55, 0.5, 0.0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
   glVertex3f(-0.55, 0.5, 0.0);
glVertex3f(-0.58, 0.77, 0.0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.6, 0.4, 0.0);
glVertex3f(-0.62, 0.4, 0.0);
glEnd();

glColor3f(0.0, 0.0, 0.0);
   glBegin(GL_LINES);
glVertex3f(-0.7, 0.4, 0.0);
glVertex3f(-0.72, 0.4, 0.0);
glEnd();
 glColor3f(0.0, 0.0, 0.0);
glPushMatrix();
 glTranslatef(-0.78, 0.2, 0.0);
 circle(0.06);
glPopMatrix();
glColor3f(0.0, 0.0, 0.0);
glPushMatrix();
glTranslatef(-0.5, 0.2, 0.0);
circle(0.06);
glPopMatrix();
 glColor3f(1.0, 1.0, 1.0);
 glPushMatrix();
 glTranslatef(-0.5, 0.2, 0.0);
 circle(0.02);
 glPopMatrix();
 glColor3f(1.0, 1.0, 1.0);
 glPushMatrix();
  glTranslatef(-0.78, 0.2, 0.0);
  circle(0.02);
  glPopMatrix();
  }
  void cloudB()  {
 glPushMatrix();
 glTranslatef(4.0,5.5,0.0);
  circle(4.0);
   glPopMatrix();
     //right
 glPushMatrix();
 glTranslatef(-8.0,5.5,0.0);
 circle(3.5);
  glPopMatrix();
   //top left
     glPushMatrix();
                glTranslatef(-3.5,9.0,0.0);
                 circle(3.5);
                  glPopMatrix();
                    //top right
glPushMatrix();
              glTranslatef(1.0,9.0,0.0);
                           circle(3.0);
                            glPopMatrix();
                                      //middle
```

```
 glPushMatrix();
 glColor3f(1.0, 1.0 ,1.0);
  glTranslatef(-1.5,5.5,0.0);
  circle(4);
   glPopMatrix();
     }
  void a() //mini Cloud
   { //left
    glPushMatrix();
    glTranslatef(4.0,5.5,0.0);
    circle(4);
     glPopMatrix();
                 //right
                  glPushMatrix();
                   glTranslatef(-8.0,5.5,0.0);
                   circle(3.5);
                    glPopMatrix();
                      //top left
                  glPushMatrix();
                  glTranslatef(-3.5,9.0,0.0);
                  circle(3.5);
                   glPopMatrix();
                     //top right
                     glPushMatrix();
                      glTranslatef(1.0,9.0,0.0);
                                circle(3.0);
                                 glPopMatrix();//middle
                                   glPushMatrix(); glTranslatef(-1.5,5.5,0.0);
                                    circle(4); glPopMatrix();

                                          }

 void c() //One Single Cloud
 {
  glPushMatrix();
   glColor3f(1.0,1.0,1.0);
 glTranslatef(35.0,10.0,0.0);
 a();
  glPopMatrix();
 glPushMatrix();
  glColor3f(1.0,1.0,1.0);
   glTranslatef(28.0,16.0,0.0);
   a();
    glPopMatrix();
      glPushMatrix();
  glColor3f (1.0,1.0,1.0);
   glTranslatef(20.0,10.0,0.0);
    a();
     glPopMatrix();
  }

   void cloud() // Three Cloud
    {
  glPushMatrix();
   glTranslatef(-15.0,25.0,0.0);
   glScalef(0.7,0.7,0.0);
    c();
    glPopMatrix();
  glPushMatrix();
  glTranslatef(20.0,25.0,0.0);
   glScalef(0.7,0.7,0.0);
   c();
    glPopMatrix();
  glPushMatrix();
  glTranslatef(-60.0,25.0,0.0);
   glScalef(0.7,0.7,0.0);
    c();
    glPopMatrix();
     }

 void rect()
 {
  glRectf(-16.0, -16.0, 16.0, 16.0);
```

```
}
 GLfloat ss=0.0;
 void day()
 {
 glBegin(GL_POLYGON);
 // blue sky
 glColor3f(sr,sg,sb);
 glVertex3f(-50,-3,0.0);
 glVertex3f(-50,50,0.0);
  glVertex3f(80.0,50.0,0.0);
  glVertex3f(80.0,-3.0,0.0);
   glEnd();
 glPushMatrix();
 //sun
 glTranslatef(ss,0.0,0.0);
 glTranslatef(-20.0,40.0,0.0);
  glScalef(1.0,1.5,0.0);
 glColor3f(1.0,1.0,0.0);
 circle(3);
  glPopMatrix();
   }

 void ground()
 {
 glColor3f(0.4,0.4,0.4);
  glPushMatrix();
    glTranslatef(-70.0,-42.0,0.0);
    glBegin(GL_POLYGON);
    glVertex3f(-10.0, 0.0,0.0);
     glVertex3f(-10.0,10.0,0.0);
glVertex3f (600.0,10.0,0.0);
 glVertex3f (600.0,0.0,0.0);
    glEnd();
     }
void divider()
{
glColor3f(1.0,1.0,1.0);
glTranslatef(-30.0,-43.0,0.0);
glBegin(GL_POLYGON);
glVertex3f(-10.0,4.0,0.0);
glVertex3f(-10.0,6.0,0.0);
glVertex3f(-2.0,6.0,0.0);
glVertex3f(-2.0,4.0,0.0);
glEnd();
}
void divider1()
{
glColor3f(1.0,1.0,1.0);
glTranslatef(-10.0,-43.0,0.0);
glBegin(GL_POLYGON);
glVertex3f(-10.0,4.0,0.0);
glVertex3f(-10.0,6.0,0.0);
glVertex3f(-2.0,6.0,0.0);
glVertex3f(-2.0,4.0,0.0);
glEnd();
}
void divider2()
{
glColor3f(1.0,1.0,1.0);
glTranslatef(10.0,-43.0,0.0);
glBegin(GL_POLYGON);
glVertex3f(-10.0,4.0,0.0);
glVertex3f(-10.0,6.0,0.0);
glVertex3f(-2.0,6.0,0.0);
glVertex3f(-2.0,4.0,0.0);
glEnd();
}
void divider3()
{
glColor3f(1.0,1.0,1.0);
glTranslatef(30.0,-43.0,0.0);
glBegin(GL_POLYGON);
glVertex3f(-10.0,4.0,0.0);
```

```
glVertex3f(-10.0,6.0,0.0);
glVertex3f(-2.0,6.0,0.0);
glVertex3f(-2.0,4.0,0.0);
glEnd();
}
void divider4()
{
glColor3f(1.0,1.0,1.0);
glTranslatef(50.0,-43.0,0.0);
glBegin(GL_POLYGON);
glVertex3f(-10.0,4.0,0.0);
glVertex3f(-10.0,6.0,0.0);
glVertex3f(-2.0,6.0,0.0);
glVertex3f(-2.0,4.0,0.0);
glEnd();
}

 void moon() //moon
 {
 glPushMatrix();
  glTranslatef(mn,0.0,0.0);
   glTranslatef(20.0,35.0,0.0);
   glScalef(1.0,1.5,0.0);
 glColor3f(1.0,1.0,1.0);
  circle(3.5);
  glPopMatrix();
  glutPostRedisplay();
  }
  void triangle(void)
  {
 glColor3f(0.137255,0.556863,0.137255);
  glBegin(GL_POLYGON);
  glVertex3f(0.0, 0.0, 0.0);
   glVertex3f(9.0, 13.0, 0.0);

glVertex3f(18.0, 0.0, 0.0);
  glEnd();
  }
   void grass()
    {
 glPushMatrix();
    glColor3f(0.8,0.196078,0.6);
  glTranslatef(38.0,16.0,0.0);
   glScalef(0.1,0.1,0.0);
   cloud();
    glPopMatrix();
  }
 void tree2() {
 glPushMatrix();
 glTranslatef(3.0,8.0,0.0);
  triangle();
   glPopMatrix();
    glPushMatrix();
 glTranslatef(3.5,14.0,0.0);
  glScalef(0.9,0.9,0.0);
   triangle();
    glPopMatrix();
                glPushMatrix();
                 glTranslatef(4.5,20.0,0.0);
                  glScalef(0.8,0.8,0.0);
                  triangle();
                  glPopMatrix();
                   glPushMatrix();
                            glTranslatef(7.0,26.0,0.0);
 glScalef(0.5,0.5,0.0);
  triangle();
  glPopMatrix();
   //gora
    glPushMatrix();
 glBegin(GL_POLYGON);
   glColor3f(0.36,0.25,0.20);
    glVertex3f(10.0, 4.0, 0.0);
glVertex3f(10.0, 8.0, 0.0);
```

```
 glVertex3f(14.0, 8.0, 0.0);
 glVertex3f(14.0, 4.0, 0.0);
   glEnd();
    glPopMatrix();
}
 void tree() //green leaves
 {
 glPushMatrix();
  glTranslatef(35.0,10.0,0.0);
  a();
  glPopMatrix();
 glPushMatrix();
  glTranslatef(28.0,16.0,0.0);
   a();
    glPopMatrix();
 glPushMatrix();
 glTranslatef(20.0,10.0,0.0);
  a();
  glPopMatrix();
  }
 void treebody() //tree body
 {
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
 glVertex3f(0.0,0.0,0.0);
  glVertex3f(2.5,2.0,0.0);
   glVertex3f(4.0,-2.0,0.0);
   glVertex3f(1.0,-4.0,0.0);
    glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
 glVertex3f(7.0,2.0,0.0);
  glVertex3f(9.0,2.0,0.0);
   glVertex3f(8.0,-2.0,0.0);
    glVertex3f(4.0,-2.0,0.0);
 glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
  glVertex3f(1.0,-4.0,0.0);
   glVertex3f(4.0,-2.0,0.0);
   glVertex3f(8.0,-2.0,0.0);
    glVertex3f(7.0,-7.0,0.0);
 glVertex3f(1.5,-7.0,0.0);
  glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
  glVertex3f(1.5,-7.0,0.0);
   glVertex3f(7.0,-7.0,0.0);
    glVertex3f(6.5,-10.0,0.0);
 glVertex3f(2.0,-10.0,0.0);
  glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
  glVertex3f(2.0,-10.0,0.0);
   glVertex3f(6.5,-10.0,0.0);
   glVertex3f(6.8,-13.0,0.0);
    glVertex3f(1.5,-13.0,0.0);
 glEnd();
 glBegin(GL_POLYGON);
glColor3f(0.502, 0.000, 0.000);
glVertex3f(1.5,-13.0,0.0);
glVertex3f(6.8,-13.0,0.0);
glVertex3f(7.0,-18.0,0.0);
glVertex3f(0.5,-18.0,0.0);
 glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
glVertex3f(0.5,-18.0,0.0);
 glVertex3f(7.0,-18.0,0.0);
glVertex3f(3.0,-27.0,0.0);
 glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.502, 0.000, 0.000);
```

```
 glVertex3f(0.5,-18.0,0.0);
 glVertex3f(2.5,-23.0,0.0);
  glVertex3f(-1.0,-25.0,0.0);
   glEnd();
glBegin(GL_POLYGON);
glColor3f(0.502, 0.000, 0.000);
 glVertex3f(7.0,-18.0,0.0);
 glVertex3f(8.0,-28.0,0.0);
 glVertex3f(4.0,-22.0,0.0);
 glEnd();
 }
 ////////////////////// / //***HOME Start***//// / ////////////////////
  void home1()
   {
   //1st Home / /1
glColor3ub(102,51,0);
glBegin(GL_POLYGON);
 glVertex2d(3.0,14.0);
  glVertex2d(3.0,11.0);
   glVertex2d(10.0,8.0);
glVertex2d(10.0,12.0);
 glVertex2d(6.0,18.0);
glEnd();
glColor3ub(153,153,0);
glBegin(GL_POLYGON);
 glVertex2d(10.0,8.0);
  glVertex2d(10.0,12.0);
   glVertex2d(20.0,12.0);
glVertex2d(20.0,8.0);
glEnd();
glColor3ub(0,100,200);
 glBegin(GL_POLYGON);
  glVertex2d(10.0,12.0);
  glVertex2d(6.0,18.0);
   glVertex2d(17.0,18.0);
glVertex2d(21.0,12.0);
glEnd();
glColor3ub(255,255,0);
glBegin(GL_POLYGON);
 glVertex2d(5.0,11.0);
  glVertex2d(5.0,12.0);
   glVertex2d(8.0,11.0);
glVertex2d(8.0,10.0);
glEnd();
glColor3ub(50,50,50);
glBegin(GL_POLYGON);
 glVertex2d(14.0,8.0);
 glVertex2d(14.0,10.0);
 glVertex2d(17.0,10.0);
 glVertex2d(17.0,8.0);
 glEnd();

}
 void house() {
glColor3ub(240,150,100);
 glBegin(GL_POLYGON);
  glVertex2d(33.0,23.0);
   glVertex2d(44.0,23.0);
glVertex2d(44.0,30.0);
 glVertex2d(33.0,30.0);
  glEnd();
    glColor3ub(0,105,105);
              glBegin(GL_POLYGON);
              glVertex2d(31.0,30.0);
               glVertex2d(46.0,30.0);
               glVertex2d(38.5,38.0);
                glEnd();
                 }
                        //HOME END///
                        void well()
{
   glBegin(GL_POLYGON);
   glColor3ub(204, 51, 0);
```

```
glVertex2f(-0.9f,-0.35f);
glVertex2f(-0.9f,-0.55f);
glVertex2f(-0.85f,-0.575f);
glVertex2f(-0.8f,-0.59f);
glVertex2f(-0.7f,-0.59f);
glVertex2f(-0.65f,-0.575f);
glVertex2f(-0.6f,-0.55f);
glVertex2f(-0.6f,-0.35f);
glEnd();

glBegin(GL_POLYGON);
// glColor3ub(255, 102, 51);
glColor3ub(38, 154, 214);
glVertex2f(-0.9f,-0.35f);
glVertex2f(-0.85f,-0.375f);
glVertex2f(-0.8f,-0.38f);
glVertex2f(-0.7f,-0.38f);
glVertex2f(-0.65f,-0.375f);
glVertex2f(-0.6f,-0.35f);
glVertex2f(-0.65f,-0.33f);
glVertex2f(-0.7f,-0.325f);
 glVertex2f(-0.8f,-0.325f);
 glVertex2f(-0.85f,-0.33f);
glEnd();

glLineWidth(5);
 glBegin(GL_LINES);
glColor3ub(204, 51, 0);
 glVertex2f(-0.9f,-0.35f);
glVertex2f(-0.85f,-0.33f);//
glVertex2f(-0.85f,-0.33f);
glVertex2f(-0.8f,-0.325f);//
glVertex2f(-0.8f,-0.325f);
glVertex2f(-0.7f,-0.325f);//
glVertex2f(-0.7f,-0.325f);
glVertex2f(-0.65f,-0.33f);//
 glVertex2f(-0.65f,-0.33f);
 glVertex2f(-0.6f,-0.35f);
glEnd();
glBegin(GL_POLYGON);
glColor3ub(194, 194, 163);
glVertex2f(-0.59f,-0.43f);
glVertex2f(-0.57f,-0.5f);
glVertex2f(-0.52f,-0.5f);
glVertex2f(-0.5f,-0.43f);
glVertex2f(-0.52f,-0.42f);
glVertex2f(-0.57f,-0.42f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(38, 154, 214);
glVertex2f(-0.585f,-0.43f);
glVertex2f(-0.568f,-0.44f);
glVertex2f(-0.528f,-0.44f);
glVertex2f(-0.505f,-0.43f);
glVertex2f(-0.528f,-0.425f);
glVertex2f(-0.57f,-0.425f);
glEnd();

glLineWidth(3);
 glBegin(GL_LINES);
glColor3ub(194, 194, 163);
glVertex2f(-0.59f,-0.43f);
glVertex2f(-0.57f,-0.39f);//
glVertex2f(-0.57f,-0.39f);
glVertex2f(-0.55f,-0.39f);//
glVertex2f(-0.55f,-0.39f);
glVertex2f(-0.52f,-0.39f);//
glVertex2f(-0.52f,-0.39f);
glVertex2f(-0.5f,-0.43f);//
glEnd();

glLineWidth(2);
```

```
    glBegin(GL_LINES);
   glColor3ub(230, 172, 0);
   glVertex2f(-0.545f,-0.385f);
   glVertex2f(-0.57f,-0.45f);//
   glVertex2f(-0.57f,-0.45f);
   glVertex2f(-0.575f,-0.5f);//
   glVertex2f(-0.575f,-0.5f);
   glVertex2f(-0.58f,-0.53f);//
   glVertex2f(-0.58f,-0.53f);
   glVertex2f(-0.57f,-0.55f);//
   glVertex2f(-0.57f,-0.55f);
   glVertex2f(-0.48f,-0.53f);//
   glEnd();
}
void bird()
{
//bird1//
      int i;
GLfloat
a=0.175f,b=0.8f,c=0.15f,d=0.8f,e=0.14f,f1=0.84f,f2=0.1f,f3=0.11f,f4=0.79f,f5=0.12f,f6=0.78f,f7=0.16f,f8=0.77f,f9=0.19f,f10=0.201f
,f11=0.83f,f12=0.144f;
GLfloat mm=0.182f; GLfloat nn=.801f; GLfloat radiusmm =.01f;
int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;

glBegin(GL_TRIANGLE_FAN);
   glColor3f(0.0,0.0,0.0);
//              glColor3ub(225, 225, 208);
              glVertex2f(mm, nn); // center of circle
              for(i = 0; i <= triangleAmount;i++) {
                        glVertex2f(
                    mm + (radiusmm * cos(i *  twicePi / triangleAmount)),
                          nn + (radiusmm * sin(i * twicePi / triangleAmount))
                        );
              }
glEnd();
   glBegin(GL_POLYGON);
   glColor3f(0.0,0.0,0.0);
   //glColor3ub(225, 225, 208 );
   glVertex2f(f2,b);
   glVertex2f(f3,f4);
   glVertex2f(f5,f6);
   glVertex2f(f7,f8);
   glVertex2f(f9,f4);
   glVertex2f(f10,d);
   glEnd();

   glBegin(GL_TRIANGLES);
glColor3f(0.0,0.0,0.0);
   //glColor3ub(217, 217, 217);
   glVertex2f(a,b);
   glVertex2f(c,d);
   glVertex2f(e,f1);
 /*  for(i=0;i<=30;i++)
   {
     glVertex2f(a+=0.175f,b);
     glVertex2f(c+=0.15f,d);
     glVertex2f(e+=0.14f,f1);
}*/
   glEnd();
   glBegin(GL_TRIANGLES);
   glColor3f(0.0,0.0,0.0);
 //  glColor3ub(242, 242, 242 );
   glVertex2f(a,b);
   glVertex2f(f12,d);
   glVertex2f(f5,f11);
   /*for(i=0;i<=0;i++)
   {
     glVertex2f(a+=0.175f,b);
     glVertex2f(f12+=0.144f,d);
     glVertex2f(f5+=0.12f,f11);
}*/
```

```
    glEnd();

/////2nd bird////
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,0.0);
   //glColor3ub(225, 225, 208 );
   glVertex2f(-0.02f,0.8f);
   glVertex2f(-0.01f,0.79f);
   glVertex2f(0.0f,0.78f);
   glVertex2f(0.04f,0.77f);
   glVertex2f(0.07f,0.79f);
   glVertex2f(0.081f,0.8f);
   glEnd();

   glBegin(GL_TRIANGLES);
  glColor3f(0.0,0.0,0.0);
  // glColor3ub(217, 217, 217);
   glVertex2f(0.055f,0.8f);
   glVertex2f(0.03f,0.8f);
   glVertex2f(0.02f,0.84f);
   glEnd();


   glBegin(GL_TRIANGLES);
   glColor3f(0.0,0.0,0.0);
   //glColor3ub(242, 242, 242 );
   glVertex2f(0.055f,0.8f);
   glVertex2f(0.024f,0.8f);
   glVertex2f(0.0f,0.83f);
   glEnd();

GLfloat mmm=0.062f; GLfloat nnn=.801f; GLfloat radiusmmm =.01f;

glBegin(GL_TRIANGLE_FAN);
glColor3f(0.0,0.0,0.0);
   //glColor3ub(225, 225, 208);
            glVertex2f(mmm, nnn); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                mmm + (radiusmmm * cos(i *  twicePi / triangleAmount)),
                    nnn + (radiusmmm * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();
/////3rd bird/////
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,0.0);
  // glColor3ub(225, 225, 208 );
   glVertex2f(-0.72f,0.8f);
   glVertex2f(-0.71f,0.79f);
   glVertex2f(-0.7f,0.78f);
   glVertex2f(-0.66f,0.77f);
   glVertex2f(-0.63f,0.79f);
   glVertex2f(-0.619f,0.8f);
   glEnd();

   glBegin(GL_TRIANGLES);
            glColor3f(0.0,0.0,0.0);
  // glColor3ub(217, 217, 217);
   glVertex2f(-0.645f,0.8f);
   glVertex2f(-0.67f,0.8f);
   glVertex2f(-0.68f,0.84f);
   glEnd();

   glBegin(GL_TRIANGLES);
            glColor3f(0.0,0.0,0.0);
 //  glColor3ub(242, 242, 242 );
   glVertex2f(-0.645f,0.8f);
   glVertex2f(-0.676f,0.8f);
   glVertex2f(-0.7f,0.83f);
   glEnd();

GLfloat mmmm=-0.638f; GLfloat nnnn=.801f;
```

```
glBegin(GL_TRIANGLE_FAN);
            glColor3f(0.0,0.0,0.0);
  //  glColor3ub(225, 225, 208);
            glVertex2f(mmmm,nnnn); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                  glVertex2f(
                mmmm + (radiusmmm * cos(i *  twicePi / triangleAmount)),
                    nnnn + (radiusmmm * sin(i * twicePi / triangleAmount))
                  );
            }
glEnd();
////4th bird////
GLfloat mmmmm=-0.518f; GLfloat nnnnn=.801f;

glBegin(GL_TRIANGLE_FAN);
  glColor3f(0.0,0.0,0.0);
    //glColor3ub(225, 225, 208);
            glVertex2f(mmmmm, nnnnn); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                  glVertex2f(
                mmmmm + (radiusmm * cos(i *  twicePi / triangleAmount)),
                    nnnnn + (radiusmm * sin(i * twicePi / triangleAmount))
                  );
            }
glEnd();
   glBegin(GL_POLYGON);
   glColor3f(0.0,0.0,0.0);
//glColor3ub(225, 225, 208 );
   glVertex2f(-0.6f,0.8f);
   glVertex2f(-0.59f,0.79f);
   glVertex2f(-0.58f,0.78f);
   glVertex2f(-0.54f,0.77f);
   glVertex2f(-0.51f,0.79f);
   glVertex2f(-0.499f,0.8f);
   glEnd();

   glBegin(GL_TRIANGLES);
   glColor3f(0.0,0.0,0.0);
  // glColor3ub(217, 217, 217);
   glVertex2f(-0.525f,0.8f);
   glVertex2f(-0.55f,0.8f);
   glVertex2f(-0.56f,0.84f);
   glEnd();


   glBegin(GL_TRIANGLES);
   glColor3f(0.0,0.0,0.0);
  // glColor3ub(242, 242, 242 );
   glVertex2f(-0.525f,0.8f);
   glVertex2f(-0.556f,0.8f);
   glVertex2f(-0.58f,0.83f);
   glEnd();
// glutPostRedisplay();

}
void backgroundtree()
{
   //pamtrees
            glBegin(GL_POLYGON);
glColor3ub(102,51,30);
glVertex2f(-1.0f,0.45f);
glVertex2f(-0.98f, 0.5f);
   glVertex2f(-0.93f, 0.56);
   glVertex2f(-0.9f,0.6f);
   glVertex2f(-0.82f, 0.64);
   glVertex2f(-0.75f, 0.67);
   glVertex2f(-0.68f, 0.64);
    glVertex2f(-0.6f,0.6f);
    glVertex2f(-0.57f, 0.56);
    glVertex2f(-0.52f, 0.5f);
    glVertex2f(-0.5f,0.45f);
    glVertex2f(-1.0f,0.45f);
```

```
glEnd();

glBegin(GL_POLYGON);
glColor3ub(102,51,0);
glVertex2f(1.0f,0.45f);
glVertex2f(0.98f, 0.5f);
   glVertex2f(0.93f, 0.56);
   glVertex2f(0.9f,0.6f);
   glVertex2f(0.82f, 0.64);
   glVertex2f(0.75f, 0.67);
   glVertex2f(0.68f, 0.64);
    glVertex2f(0.6f,0.6f);
    glVertex2f(0.57f, 0.56);
    glVertex2f(0.52f, 0.5f);
    glVertex2f(0.5f,0.45f);
    glVertex2f(1.0f,0.45f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(102,51,0);
glVertex2f(-0.5f,0.45f);
glVertex2f(-0.48f, 0.5f);
   glVertex2f(-0.45f, 0.56);
   glVertex2f(-0.42f,0.6f);
   glVertex2f(-0.37f, 0.62);
   glVertex2f(-0.32f, 0.6);
   glVertex2f(-0.29f, 0.56f);
    glVertex2f(-0.27f, 0.5f);
   glVertex2f(-0.25f,0.45f);
   glEnd();

   glBegin(GL_POLYGON);
glColor3ub(102,51,0);
glVertex2f(0.5f,0.45f);
glVertex2f(0.48f, 0.5f);
   glVertex2f(0.45f, 0.56);
   glVertex2f(0.42f,0.6f);
   glVertex2f(0.37f, 0.62);
   glVertex2f(0.32f, 0.6);
   glVertex2f(0.29f, 0.56f);
    glVertex2f(0.27f, 0.5f);
   glVertex2f(0.25f,0.45f);
   glEnd();

   glBegin(GL_POLYGON);
glColor3ub(102,51,0);
glVertex2f(-.25f,0.45f);
glVertex2f(-0.23f, 0.5f);
   glVertex2f(-0.18f, 0.56);
   glVertex2f(-0.15f,0.6f);
   glVertex2f(-0.07f, 0.64);
   glVertex2f(-0.00f, 0.67);
   glVertex2f(0.07f, 0.64);
   glVertex2f(0.15f,0.6f);
   glVertex2f(0.18f, 0.56);
   glVertex2f(0.23f, 0.5f);
   glVertex2f(.25f,0.45f);
glEnd();

}
 void triangle1(void)
  {
 glColor3f(1.0,1.0,1.0);
  glBegin(GL_POLYGON);
  glVertex3f(0.0, 0.0, 0.0);
   glVertex3f(9.0, 13.0, 0.0);

glVertex3f(18.0, 0.0, 0.0);
  glEnd();
  }

void grass1()
{
```

```
    glLineWidth(4);
   glBegin(GL_LINES);
   glColor3ub(0, 102, 0);
glVertex2f(-.05f, -0.35f);
glVertex2f(-0.0f, -0.4f);//
glVertex2f(0.05f, -0.35f);
glVertex2f(0.0f, -0.4f);//
glVertex2f(0.027f, -0.33f);
glVertex2f(0.0f, -0.4f);//
glVertex2f(-0.027f, -0.33f);
glVertex2f(0.0f, -0.4f);//
glVertex2f(0.0f, -0.3f);
glVertex2f(0.0f, -0.4f);//
glVertex2f(-0.075f, -0.37f);
glVertex2f(-0.0f, -0.4f);//
glVertex2f(0.0745f, -0.37f);
glVertex2f(-0.0f, -0.4f);//
   glEnd();
   int i;
   int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;


GLfloat e=-.05f; GLfloat f=-.35f; GLfloat radius11 =.02f;

glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 51, 0);
            glVertex2f(e, f); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                e + (radius11 * cos(i *  twicePi / triangleAmount)),
                    f+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

GLfloat g=0.05f; GLfloat h=-0.35f;
glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 102, 0);
            glVertex2f(g, h); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                g + (radius11 * cos(i *  twicePi / triangleAmount)),
                    h+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();
GLfloat a1=0.0f; GLfloat b1=-0.3f;

glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 255, 0);
            glVertex2f(a1, b1); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                a1 + (radius11 * cos(i *  twicePi / triangleAmount)),
                    b1 + (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();
}

void flower()
{

 int i;
   int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;


GLfloat e=-.05f; GLfloat f=-.35f; GLfloat radius11 =.02f;

glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 182, 203);
```

```
            glVertex2f(e, f); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                e + (radius11 * cos(i *  twicePi / triangleAmount)),
                    f+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

GLfloat g=0.05f; GLfloat h=-0.35f;
glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 182, 203);
            glVertex2f(g, h); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                g + (radius11 * cos(i *  twicePi / triangleAmount)),
                    h+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();
GLfloat a1=0.0f; GLfloat b1=-0.3f;

glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 182, 203);
            glVertex2f(a1, b1); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                a1 + (radius11 * cos(i *  twicePi / triangleAmount)),
                    b1 + (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

}
void flower1()
{

 int i;
   int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;


GLfloat e=-.05f; GLfloat f=-.35f; GLfloat radius11 =.02f;

glBegin(GL_TRIANGLE_FAN);
   glColor3f(1.0,1.0,0.0);
            glVertex2f(e, f); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                e + (radius11 * cos(i *  twicePi / triangleAmount)),
                    f+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

GLfloat g=0.05f; GLfloat h=-0.35f;
glBegin(GL_TRIANGLE_FAN);
        glColor3f(1.0,1.0,0.0);
            glVertex2f(g, h); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                g + (radius11 * cos(i *  twicePi / triangleAmount)),
                    h+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();
GLfloat a1=0.0f; GLfloat b1=-0.3f;

glBegin(GL_TRIANGLE_FAN);
glColor3f(1.0,1.0,0.0);
            glVertex2f(a1, b1); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
```

```
                    a1 + (radius11 * cos(i *  twicePi / triangleAmount)),
                        b1 + (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

}

void apple()
{

int i;
   int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;


GLfloat e=-.05f; GLfloat f=-.35f; GLfloat radius11 =.02f;

glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 51, 0);
            glVertex2f(e, f); // center of circle
            for(i = 0; i <= triangleAmount;i++) {
                    glVertex2f(
                e + (radius11 * cos(i *  twicePi / triangleAmount)),
                    f+ (radius11 * sin(i * twicePi / triangleAmount))
                    );
            }
glEnd();

}
void StartingText()
{
   char ch, text[120];
   sprintf(text, "BEAUTY OF NATURE  BEAUTIFUL VILLAGE SCENARIO",5.00,8.00);
   glColor3f(0, 0, 0);
   glRasterPos2f( -20 , 12 );
   for(int i = 0; text[i] != '\0'; i++)
   {
      if(text[i]==' '  && text[i+1]==' ')
      {
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, text[i]);
         glRasterPos2f( -32 , 02 );
      }
      else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, text[i]);
   }
     sprintf(text,"   PRESS S FOR SUMMER VIEW,PRESS R FOR RAIN VIEW,PRESS W FOR WINTER VIEW,PRESS F FOR
SPRING VIEW",5.00,8.00);
   glColor3f(0, 0, 0);
   glRasterPos2f( -100 , 0 );
   for(int i = 0; text[i] != '\0'; i++)
   {
       if(text[i]==' '  && text[i+1]==' ')
      {
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
         glRasterPos2f( -30.5 , -10 );
      }
      else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
   }
   sprintf(text,"  CREATED BY-",5.00,8.00);
   glColor3f(0, 0, 0);
   glRasterPos2f( -100 , 0 );
   for(int i = 0; text[i] != '\0'; i++)
   {
       if(text[i]==' '  && text[i+1]==' ')
      {
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
         glRasterPos2f( 18 , -25 );
      }
      else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
   }
      sprintf(text,"  J SRI SAI SANGEETHA, HARPREETH KAUR J",5.00,8.00);
   glColor3f(0, 0, 0);
   glRasterPos2f( -100 , 0 );
```

```
    for(int i = 0; text[i] != '\0'; i++)
    {
        if(text[i]==' '  && text[i+1]==' ')
       {
          glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
          glRasterPos2f( 18 , -30 );
       }
       else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, text[i]);
    }


}
void fence()
{
     glLineWidth(4);
     glBegin(GL_LINES);
    glColor3ub(255, 255, 102);
    glVertex2f(-2.0f,-0.1f);
    glVertex2f(-1.6f,-0.1f);

    glColor3ub(255, 255, 102);
    glVertex2f(-2.0f,-0.05f);
    glVertex2f(-1.6f,-0.05f);

    glColor3ub(255, 255, 102);
    glVertex2f(-2.0f,0.0f);
    glVertex2f(-1.6f,0.0f);

     glColor3ub(255, 255, 102);
    glVertex2f(-2.0f,0.05f);
    glVertex2f(-1.6f,0.05f);

     glColor3ub(255, 255, 102);
    glVertex2f(-2.0f,0.1f);
    glVertex2f(-1.6f,0.1f);

     glColor3ub(255, 255, 102);
    glVertex2f(-1.95f,0.13f);
    glVertex2f(-1.95f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.9f,0.13f);
    glVertex2f(-1.9f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.85f,0.13f);
    glVertex2f(-1.85f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.8f,0.13f);
    glVertex2f(-1.8f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.75f,0.13f);
    glVertex2f(-1.75f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.7f,0.13f);
    glVertex2f(-1.7f,-0.12f);

    glColor3ub(255, 255, 102);
    glVertex2f(-1.65f,0.13f);
    glVertex2f(-1.65f,-0.12f);

    glEnd();
}

void Display(void)
{
    //std::cout<<"Display 1 called"<<std::endl;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glPushMatrix();
```

```
    glTranslatef(0,0,-20);
   // glScalef(30.0,30.0,0.0);
    StartingText();
     glutPostRedisplay();
   //glutSwapBuffers();
    glPopMatrix();
    glFlush();
    glutSwapBuffers();
  //  glutPostRedisplay();
}


GLfloat xw=-40.0;
GLfloat yw,f1,f2,f3,f4,f5,f6;
void disp()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
   day();
    glPopMatrix();
///ground///
 glPushMatrix();
 glTranslatef(0.0,-35.0,0.0);
  glScalef(3.5,3.0,0.0);
 glColor3f(0.0, 0.5, 0.0);
  rect();
   glPopMatrix();

    glPushMatrix();
 glTranslatef(30.0,-5.0,0.0);
  glScalef(80.0,40.0,0.0);
// glColor3f(0.0, 0.5, 0.0);
  backgroundtree();
   glPopMatrix();

glPushMatrix();
 glColor3f (1.0,1.0,1.0);
                glScalef(40.0,40.0,0.0);
                glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();

   glPushMatrix();
  glTranslatef(cx,17.0,0.0);
                glScalef(40.0,40.0,0.0);
 //  glTranslatef(cx,17.0,0.0);
   bird();
   glPopMatrix();

    glPushMatrix();
 glColor3f (1.0,1.0,1.0);
 // glTranslatef(cx,20.0,0.0);
  // cloudB();
    glPopMatrix();

//tree1
 glPushMatrix();
  glColor3ub(248, 188, 203);
 //glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-49.0,9.5,0.0);
   glScalef(0.4,0.5,0.0);
   tree();
    glPopMatrix();
 glPushMatrix();
 glTranslatef(-40.4,15.4,0.0);
 glScalef(0.4,0.5,0.0);
  treebody();
   glPopMatrix();  //  ///

 glPushMatrix();
 moon();
  glPopMatrix();  ///
 glPushMatrix();
```

```
  glTranslatef(cx,0.0,0.0);
   glScalef(0.8,1.0,0.0);
    cloud();
 glPopMatrix();  //***/// / //home 2///
 glPushMatrix();
 glTranslatef(-65.0,-20.0,0.0);
 house();
 glPopMatrix();  //***/// / //home 1///
 glPushMatrix();
  glTranslatef(-45.0,-10.0,0.0);
   home1();
    glPopMatrix();
///home 2///
 glPushMatrix();
  glTranslatef(20.0,-15.0,0.0);
   home1();
    glPopMatrix();
  //***/// / //tree typeA 1///
   glPushMatrix();
   glTranslatef(-5.0,-5.0,0.0);
   glScalef(0.5,1.0,0.0);
   tree2();
   glPopMatrix();
    //***/// / //tree typeB 1///
                glPushMatrix();
                glTranslatef(35.0,-5.0,0.0);
                glScalef(0.5,1.0,0.0);
                tree2();
                 glPopMatrix();
                  //***/// / //tree typeC 1///
                  glPushMatrix();
                   glTranslatef(30.0,-15.0,0.0);
                    glScalef(0.5,1.0,0.0);
                              tree2();
                               glPopMatrix();
                                //***///


///tree2///
 glPushMatrix();
  glColor3ub(250, 182, 203);
// glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-44.0,-1.0,0.0);
  glScalef(0.4,0.5,0.0); tree();
   glPopMatrix();
 glPushMatrix();
  glTranslatef(-35.0,5.0,0.0);
   glScalef(0.4,0.5,0.0);
   treebody();
   glPopMatrix();
    //***/// / //********home 4********///
 glPushMatrix();
  glTranslatef(-10.0,-10.0,0.0);
  home1();
  glPopMatrix();
  glPopMatrix();
    //***/// / //********road********///
      glPushMatrix();
                glTranslatef(-10.0,15.0,0.0);
                 ground();
                  glPopMatrix();
glPopMatrix();
  glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  ground();
  glPopMatrix();
  glPopMatrix();
  //divider ///
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider();
  glPopMatrix();
    glPushMatrix();
```

```
 glTranslatef(-10.0,20.0,0.0);
 divider1();
 glPopMatrix();
   glPushMatrix();
 glTranslatef(-10.0,20.0,0.0);
 divider2();
 glPopMatrix();
  glPushMatrix();
 glTranslatef(-10.0,20.0,0.0);
 divider3();
 glPopMatrix();
  glPushMatrix();
 glTranslatef(-10.0,20.0,0.0);
 divider4();
 glPopMatrix();
   //************/// / //BUS///
glPushMatrix();
 glTranslatef(cx, -18.0, 0.0);
  glScalef(30.0, 8.0, 0.0);
   car();
glPopMatrix();
 //***///  ///BUS2///
glPushMatrix();
glTranslatef(cx, -8.0, -15.0);
 glScalef(30.0, 8.0, 0.0);
 car();
  glPopMatrix();
   glPushMatrix();
glTranslatef(160.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();

   //***///  ///tree3///
    glPopMatrix();
  glPushMatrix();
glTranslatef(100.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

   //***///  ///tree3///
glPushMatrix();
 glColor3ub(255, 182, 203);
//glColor3f(0.133, 0.545, 0.133);
 glTranslatef(22.0,-32.5,0.0);
  glScalef(0.4,0.5,0.0);
   tree();
glPopMatrix();
glPopMatrix();
glPushMatrix();
glTranslatef(30.5,-26.5,0.0);
 glScalef(0.4,0.5,0.0);
 treebody();
  glPopMatrix();  //***///  ///home 3///
glPushMatrix();
 glTranslatef(-5.0,-70.0,0.0);
 house();
 glPopMatrix(); glPopMatrix();
  //***///  ///tree typeD 1///
   glPushMatrix();
glTranslatef(-10.0,-50.0,0.0);
glScalef(.5,1,0);
 tree2();
 glPopMatrix(); ///***///
 //well
 glPushMatrix();
glTranslatef(-12.0,-20.0,0.0);
```

```
glScalef(40.0,40.0,0.0);
 well();
 glPopMatrix(); ///***///
            glPushMatrix();
//          glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
// bird();
 glPopMatrix(); ///***///
glPushMatrix();
glTranslatef(-15.0,10.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix(); ///***///
            glPushMatrix();
glTranslatef(-18.0,8.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
            glPushMatrix();
glTranslatef(-23.0,6.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
glPushMatrix();
glTranslatef(-29.0,1.0,0.0);
glScalef(25.0,15.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-40.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(20.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(24.0,-2.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(45.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(20.0,-1.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(-9.0,5.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(4.0,1.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
             glPushMatrix();
glTranslatef(12.0,5.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix(); ///flower///
             glPushMatrix();
glTranslatef(-33.5,20.1,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
```

```
 glPushMatrix();
glTranslatef(-36.5,20.1,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();


  glPushMatrix();
glTranslatef(-39.0,20.3,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-44.5,20.3,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();

 glPushMatrix();
glTranslatef(-41.5,13.0,0.0); //down
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-41.5,20.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-42.5,14.0,0.0);//down
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();

 //2ndtree
glPushMatrix();
glTranslatef(-39.0,9.9,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
                                        glPushMatrix();
glTranslatef(-36.0,9.9,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
glPushMatrix();
glTranslatef(-32.0,9.9,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
                                        glPushMatrix();
glTranslatef(-29.0,9.9,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
  //3rd tree
                                        glPushMatrix();
glTranslatef(29.0,-22.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
glPushMatrix();
glTranslatef(32.0,-22.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();
glPushMatrix();
glTranslatef(36.0,-22.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
  glPopMatrix();

  glPushMatrix();
```

```
glTranslatef(xw,yw,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 //fallflowertree1
   glPushMatrix();
glTranslatef(-36.0,yw,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-39.0,f1,0.0); //2tree
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 glPushMatrix();
                glTranslatef(-32.0,f2,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 glPushMatrix();
                glTranslatef(-29.0,f1,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
    glPushMatrix();
                glTranslatef(-36.0,f3,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();


glPushMatrix();
glTranslatef(-29.0,-1.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower 2tree
 glPopMatrix();
                glPushMatrix();
glTranslatef(-27.0,-1.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower 2tree
 glPopMatrix();

glPushMatrix();
glTranslatef(-38.0,-1.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
glPushMatrix();
glTranslatef(-38.0,0.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();

 glPushMatrix();
glTranslatef(-36.0,-9.5,0.0);
glScalef(15.0,15.0,0.0);
flower();
 glPopMatrix();
glPushMatrix();
glTranslatef(38.0,-28.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower 3tree
 glPopMatrix();
                glPushMatrix();
glTranslatef(36.0,-28.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower 3tree
 glPopMatrix();
glPushMatrix();
glTranslatef(30.0,-28.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower tree
```

```
 glPopMatrix();
glPushMatrix();
glTranslatef(30.0,-26.0,0.0);
glScalef(15.0,15.0,0.0);
 flower();//downflower 2tree
 glPopMatrix();

 glPushMatrix();
                glTranslatef(29.0,f4,0.0);
glScalef(15.0,15.0,0.0);
 flower();//falling3
 glPopMatrix();
   glPushMatrix();
                glTranslatef(36.0,f5,0.0);
glScalef(15.0,15.0,0.0);
 flower();
 glPopMatrix();
 glPushMatrix();
                glTranslatef(-3.0,f6,0.0);
glScalef(15.0,15.0,0.0);
 flower1();
 glPopMatrix();

  glPushMatrix();
glTranslatef(1.0,-40.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflowery 3tree
 glPopMatrix();
                glPushMatrix();
glTranslatef(-10.0,-42.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower 3tree
 glPopMatrix();
glPushMatrix();
glTranslatef(1.0,-42.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower tree
 glPopMatrix();
glPushMatrix();
glTranslatef(-10.0,-40.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower 2tree
 glPopMatrix();

 glPushMatrix();
glTranslatef(-1.0,-32.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower 3tree
 glPopMatrix();
                glPushMatrix();
glTranslatef(-4.0,-35.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower 3tree
 glPopMatrix();
glPushMatrix();
glTranslatef(-3.0,-28.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower tree
 glPopMatrix();
glPushMatrix();
glTranslatef(-6.0,-30.0,0.0);
glScalef(15.0,15.0,0.0);
 flower1();//downflower 2tree
 glPopMatrix();



glPushMatrix();
glTranslatef(75.0,-9.5,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix(); glPushMatrix();
glTranslatef(75.0,-9.5,0.0);
```

```
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(100.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
  glPushMatrix();
glTranslatef(160.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
                glPushMatrix();
glTranslatef(75.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

   glPushMatrix();
glTranslatef(100.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
  glPushMatrix();
glTranslatef(160.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
                glPushMatrix();
glTranslatef(75.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

   glPushMatrix();
glTranslatef(0.0,-0.0,0.0);
glScalef(2.0,3.0,0.0);
 fence();
  glPopMatrix();



 glFlush();
}
void bird1(){
//bird1//
      int i;

GLfloat mm=0.182f; GLfloat nn=.801f; GLfloat radiusmm =.01f;
int triangleAmount = 20;
GLfloat twicePi = 2.0f * PI;

glBegin(GL_TRIANGLE_FAN);
//              glColor3ub(225, 225, 208);
                glVertex2f(mm, nn); // center of circle
                for(i = 0; i <= triangleAmount;i++) {
                        glVertex2f(
                    mm + (radiusmm * cos(i *  twicePi / triangleAmount)),
                        nn + (radiusmm * sin(i * twicePi / triangleAmount))
                        );
                }
glEnd();
```

```
    glBegin(GL_POLYGON);
//    glColor3f(0.0,0.0,0.0);
    //glColor3ub(225, 225, 208 );
    glVertex2f(0.1f,0.8f);
    glVertex2f(0.11f,0.79f);
    glVertex2f(0.12f,0.78f);
    glVertex2f(0.16f,0.77f);
    glVertex2f(0.19f,0.79f);
    glVertex2f(0.201f,0.8f);
    glEnd();

    glBegin(GL_TRIANGLES);
//              glColor3f(0.0,0.0,0.0);

    //glColor3ub(217, 217, 217);
    glVertex2f(0.175f,0.8f);
    glVertex2f(0.15f,0.8f);
    glVertex2f(0.14f,0.84f);
    glEnd();


    glBegin(GL_TRIANGLES);
  //  glColor3f(0.0,0.0,0.0);
  //  glColor3ub(242, 242, 242 );
    glVertex2f(0.175f,0.8f);
    glVertex2f(0.144f,0.8f);
    glVertex2f(0.12f,0.83f);
    glEnd();
}

void display1()
{
 glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
   day();
    glPopMatrix();
///ground///
 glPushMatrix();
 glTranslatef(0.0,-35.0,0.0);
  glScalef(3.5,3.0,0.0);
 glColor3f(0.0, 0.5, 0.0);
  rect();
   glPopMatrix();

    glPushMatrix();
 glTranslatef(30.0,-5.0,0.0);
  glScalef(80.0,40.0,0.0);
// glColor3f(0.0, 0.5, 0.0);
  backgroundtree();
   glPopMatrix();

glPushMatrix();
 glColor3f (1.0,1.0,1.0);
              glScalef(40.0,40.0,0.0);
              glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();

   glPushMatrix();
   glTranslatef(cx,17.0,0.0);
              glScalef(40.0,40.0,0.0);
     bird();
   glPopMatrix();


    glPushMatrix();
 glColor3f (1.0,1.0,1.0);
 // glTranslatef(cx,20.0,0.0);
  // cloudB();
    glPopMatrix();

//tree1
 glPushMatrix();
```

```
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-49.0,9.5,0.0);
   glScalef(0.4,0.5,0.0);
   tree();
    glPopMatrix();
  glPushMatrix();
  glTranslatef(-40.4,15.4,0.0);
  glScalef(0.4,0.5,0.0);
   treebody();
    glPopMatrix();  //  ///

  glPushMatrix();
  moon();
   glPopMatrix();  ///
  glPushMatrix();
   glTranslatef(cx,0.0,0.0);
    glScalef(0.8,1.0,0.0);
     cloud();
  glPopMatrix();  //***/// / //home 2///
  glPushMatrix();
  glTranslatef(-65.0,-20.0,0.0);
  house();
  glPopMatrix();  //***/// / //home 1///
  glPushMatrix();
   glTranslatef(-45.0,-10.0,0.0);
    home1();
     glPopMatrix();
 ///home 2///
  glPushMatrix();
   glTranslatef(20.0,-15.0,0.0);
    home1();
     glPopMatrix();
   //***/// / //tree typeA 1///
     glPushMatrix();
     glTranslatef(-5.0,-5.0,0.0);
     glScalef(0.5,1.0,0.0);
     tree2();
     glPopMatrix();
      //***/// / //tree typeB 1///
                 glPushMatrix();
                 glTranslatef(35.0,-5.0,0.0);
                 glScalef(0.5,1.0,0.0);
                 tree2();
                  glPopMatrix();
                   //***/// / //tree typeC 1///
                   glPushMatrix();
                    glTranslatef(30.0,-15.0,0.0);
                     glScalef(0.5,1.0,0.0);
                               tree2();
                                glPopMatrix();
                                 //***///


 ///tree2///
  glPushMatrix();
  glColor3f(0.133, 0.545, 0.133);
   glTranslatef(-44.0,-1.0,0.0);
   glScalef(0.4,0.5,0.0); tree();
    glPopMatrix();
  glPushMatrix();
   glTranslatef(-35.0,5.0,0.0);
    glScalef(0.4,0.5,0.0);
    treebody();
    glPopMatrix();
     //***/// / //********home 4********///
 glPushMatrix();
  glTranslatef(-10.0,-10.0,0.0);
  home1();
  glPopMatrix();
  glPopMatrix();
    //***/// / //********road********///
      glPushMatrix();
                 glTranslatef(-10.0,15.0,0.0);
```

```
                ground();
                 glPopMatrix();
glPopMatrix();
  glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  ground();
  glPopMatrix();
  glPopMatrix();
  //divider ///
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider1();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider2();
  glPopMatrix();
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider3();
  glPopMatrix();
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider4();
  glPopMatrix();
    //************/// / //BUS///
 glPushMatrix();
  glTranslatef(cx, -19.0, 0.0);
   glScalef(30.0, 8.0, 0.0);
    car();
 glPopMatrix();
  //***///  ///BUS2///
 glPushMatrix();
 glTranslatef(cx, -8.0, -15.0);
  glScalef(30.0, 8.0, 0.0);
  car();
   glPopMatrix();
   glPushMatrix();
glTranslatef(160.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();

    //***///  ///tree3///
     glPopMatrix();
    glPushMatrix();
glTranslatef(100.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
    glPushMatrix();
glTranslatef(125.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(22.0,-32.5,0.0);
   glScalef(0.4,0.5,0.0);
    tree();
 glPopMatrix();
 glPopMatrix();
glPushMatrix();
glTranslatef(30.5,-26.5,0.0);
 glScalef(0.4,0.5,0.0);
 treebody();
  glPopMatrix();  //***///  ///home 3///
 glPushMatrix();
```

```
  glTranslatef(-5.0,-70.0,0.0);
  house();
  glPopMatrix(); glPopMatrix();
   //***///  ///tree typeD 1///
    glPushMatrix();
glTranslatef(-10.0,-50.0,0.0);
glScalef(.5,1,0);
 tree2();
 glPopMatrix(); ///***///
 //well
 glPushMatrix();
glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
 well();
 glPopMatrix(); ///***///
                glPushMatrix();
//              glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
// bird();
 glPopMatrix(); ///***///
glPushMatrix();
glTranslatef(-15.0,10.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
glTranslatef(-18.0,8.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-23.0,6.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
glPushMatrix();
glTranslatef(-29.0,1.0,0.0);
glScalef(25.0,15.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-40.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(20.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(24.0,-2.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(45.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(20.0,-1.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-9.0,5.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(4.0,1.0,0.0);
glScalef(15.0,15.0,0.0);
```

```
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(12.0,5.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
  glPushMatrix();
glTranslatef(75.0,-9.5,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(100.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
  glPushMatrix();
glTranslatef(160.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
                 glPushMatrix();
glTranslatef(75.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

glPushMatrix();
                glTranslatef(-20.4,6.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-20.4,8.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-26.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-25.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///



 //right grass
   glPushMatrix();
                glTranslatef(28.5,-3.5,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
   glPushMatrix();
                glTranslatef(30.0,-4.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix();
glPushMatrix();
                glTranslatef(10.0,-40.0,0.0);
glScalef(25.0,22.0,0.0);
 bird1(); //wood
 glPopMatrix();
   glPushMatrix();
```

```
                glTranslatef(-10.0,-33.5,0.0);
                glColor3f(0.0,0.0,0.0);
glScalef(20.0,25.0,0.0);
 bird1(); //btree
 glPopMatrix();

     glPushMatrix();
                glTranslatef(-30.0,-0.5,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
           glColor3f(1.0,1.0,0.9);
               glTranslatef(-39.0,1.0,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
   glTranslatef(-30.0,5.0,0.0);
glScalef(20.0,10.0,0.0);
 fence();
 glPopMatrix();

                glPushMatrix();
glTranslatef(-40.0,27.2,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

                 glPushMatrix();
glTranslatef(-35.5,27.2,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

                 glPushMatrix();
glTranslatef(-39.0,32.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

                 glPushMatrix();
glTranslatef(-42.0,27.3,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

 glPushMatrix();
glTranslatef(0.0,0.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-35.5,22.2,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

                 glPushMatrix();
glTranslatef(-30.0,22.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

                 glPushMatrix();
glTranslatef(-28.0,17.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

 glPushMatrix();
glTranslatef(-30.5,17.2,0.0);
glScalef(20.0,35.0,0.0);
```

```
 apple();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-38.0,17.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-35.5,17.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

 glPushMatrix();
glTranslatef(30.5,-15.3,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();
 glPushMatrix();
glTranslatef(38.0,-15.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();
                glPushMatrix();
glTranslatef(35.5,-15.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();

 glPushMatrix();
glTranslatef(35.0,-10.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();
                glPushMatrix();
glTranslatef(29.5,-12.0,0.0);
glScalef(20.0,35.0,0.0);
 apple();
 glPopMatrix();


 glFlush();
}

 void day2()
 {
                glBegin(GL_POLYGON);
 // blue sky
// glColor3f(0.0,0.0,0.9);
// glColor3ub(211,211,217);
 //glColor3ub(177,156,217);
glColor3ub(177,211,217); //snowfall
 glVertex3f(-50,-3,0.0);
 glVertex3f(-50,50,0.0);
  glVertex3f(80.0,50.0,0.0);
  glVertex3f(80.0,-3.0,0.0);
   glEnd();

 }
 void bubbles()
 {
                glColor3f(1.0,1.0,1.0);
                glutSolidSphere(0.5,8,10);
 }
 GLfloat q1,q2,q3,q4,q5;


void display2()
{
glClear(GL_COLOR_BUFFER_BIT);

    glPushMatrix();
   day2();
```

```
    glPopMatrix();
  ///ground///
  glPushMatrix();
  glTranslatef(0.0,-35.0,0.0);
   glScalef(3.5,3.0,0.0);
  glColor3f(1.0, 1.0, 1.0);
   rect();
    glPopMatrix();

     glPushMatrix();
  glTranslatef(30.0,-5.0,0.0);
   glScalef(80.0,40.0,0.0);
 // glColor3f(0.0, 0.5, 0.0);
   backgroundtree();
    glPopMatrix();

     glPushMatrix();
     glColor3f (1.0,1.0,1.0);
     glTranslatef(-20.0,44.0,0.0);
                glScalef(0.9,0.8,0.0);
    //          glTranslatef(cx,17.0,0.0);
       cloudB();
    glPopMatrix();

    glPushMatrix();
     glColor3f (1.0,1.0,1.0);
     glTranslatef(10.5,44.0,0.0);
                glScalef(0.9,0.8,0.0);
    //          glTranslatef(cx,17.0,0.0);
       cloudB();
    glPopMatrix();

  glPushMatrix();
     glColor3f (1.0,1.0,1.0);
     glTranslatef(45.0,44.0,0.0);
                glScalef(0.9,0.8,0.0);
    //          glTranslatef(cx,17.0,0.0);
       cloudB();
    glPopMatrix();


 glPushMatrix();
  glColor3f (1.0,1.0,1.0);
   glTranslatef(cx,17.0,0.0);
                glScalef(40.0,40.0,0.0);
    //          glTranslatef(cx,17.0,0.0);
      cloudB();
    glPopMatrix();
     glPushMatrix();
  glColor3f (1.0,1.0,1.0);
   glTranslatef(cx,20.0,0.0);
   // cloudB();
     glPopMatrix();
 //tree1
  glPushMatrix();
  glColor3f(0.133, 0.545, 0.133);
   glTranslatef(-49.0,9.5,0.0);
    glScalef(0.4,0.5,0.0);
    tree();
     glPopMatrix();
  glPushMatrix();
  glTranslatef(-40.4,15.4,0.0);
  glScalef(0.4,0.5,0.0);
   treebody();
    glPopMatrix();  //  ///

  glPushMatrix();
  // glTranslatef(cx,0.0,0.0);
    glScalef(0.8,1.0,0.0);
     cloud();
  glPopMatrix();  //***/// / //home 2///
  glPushMatrix();
  glTranslatef(-65.0,-20.0,0.0);
```

```
 house();
 glPopMatrix();  //***/// / //home 1///
 glPushMatrix();
  glTranslatef(-45.0,-10.0,0.0);
   home1();
    glPopMatrix();
///home 2///
 glPushMatrix();
  glTranslatef(20.0,-15.0,0.0);
   home1();
    glPopMatrix();
  //***/// / //tree typeA 1///
   glPushMatrix();
   glTranslatef(-5.0,-5.0,0.0);
   glScalef(0.5,1.0,0.0);
   tree2();
   glPopMatrix();
    //***/// / //tree typeB 1///
                glPushMatrix();
                glTranslatef(35.0,-5.0,0.0);
                glScalef(0.5,1.0,0.0);
                tree2();
                 glPopMatrix();
                  //***/// / //tree typeC 1///
                  glPushMatrix();
                   glTranslatef(30.0,-15.0,0.0);
                    glScalef(0.5,1.0,0.0);
                              tree2();
                               glPopMatrix();
                                //***///


///tree2///
 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-44.0,-1.0,0.0);
  glScalef(0.4,0.5,0.0); tree();
   glPopMatrix();
 glPushMatrix();
  glTranslatef(-35.0,5.0,0.0);
   glScalef(0.4,0.5,0.0);
   treebody();
   glPopMatrix();
    //***/// / //********home 4********///
glPushMatrix();
 glTranslatef(-10.0,-10.0,0.0);
 home1();
 glPopMatrix();
 glPopMatrix();
   //***/// / //********road********///
     glPushMatrix();
                glTranslatef(-10.0,15.0,0.0);
                 ground();
                  glPopMatrix();
glPopMatrix();
  glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  ground();
  glPopMatrix();
  glPopMatrix();
  //divider ///
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider1();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider2();
  glPopMatrix();
```

```
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider3();
  glPopMatrix();
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider4();
  glPopMatrix();
    //************/// / //BUS///
 glPushMatrix();
  glTranslatef(cx, -19.0, 0.0);
   glScalef(30.0, 8.0, 0.0);
    car();
 glPopMatrix();
  //***///  ///BUS2///
 glPushMatrix();
 glTranslatef(cx, -8.0, -15.0);
  glScalef(30.0, 8.0, 0.0);
  car();
   glPopMatrix();
   glPushMatrix();
glTranslatef(160.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();

    //***///  ///tree3///
     glPopMatrix();
   glPushMatrix();
glTranslatef(100.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(22.0,-32.5,0.0);
   glScalef(0.4,0.5,0.0);
    tree();
 glPopMatrix();
 glPopMatrix();
 glPushMatrix();
 glTranslatef(30.5,-26.5,0.0);
  glScalef(0.4,0.5,0.0);
  treebody();
   glPopMatrix();  //***///  ///home 3///
 glPushMatrix();
  glTranslatef(-5.0,-70.0,0.0);
  house();
  glPopMatrix(); glPopMatrix();
   //***///  ///tree typeD 1///
    glPushMatrix();
glTranslatef(-10.0,-50.0,0.0);
glScalef(.5,1,0);
 tree2();
 glPopMatrix(); ///***///
 //well
 glPushMatrix();
glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
 well();
 glPopMatrix(); ///***///
                glPushMatrix();
//              glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
// bird();
 glPopMatrix(); ///***///
glPushMatrix();
```

```
glTranslatef(-15.0,10.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
glTranslatef(-18.0,8.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-23.0,6.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
glPushMatrix();
glTranslatef(-29.0,1.0,0.0);
glScalef(25.0,15.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-40.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(20.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(24.0,-2.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(45.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(20.0,-1.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(-9.0,5.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(4.0,1.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                 glPushMatrix();
glTranslatef(12.0,5.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
  glPushMatrix();
glTranslatef(75.0,-9.5,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(100.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
```

```
 glPopMatrix();
 glPushMatrix();
glTranslatef(160.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();
                glPushMatrix();
glTranslatef(75.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();

glPushMatrix();
              glTranslatef(-20.4,6.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
              glPushMatrix();
              glTranslatef(-20.4,8.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
              glPushMatrix();
              glTranslatef(-26.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
              glPushMatrix();
              glTranslatef(-25.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///



 //right grass
   glPushMatrix();
              glTranslatef(28.5,-3.5,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
   glPushMatrix();
              glTranslatef(30.0,-4.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
              glTranslatef(10.0,-40.0,0.0);
glScalef(25.0,22.0,0.0);
 bird1(); //wood
 glPopMatrix();
   glPushMatrix();
              glTranslatef(-10.0,-33.5,0.0);
              glColor3f(0.0,0.0,0.0);
glScalef(20.0,25.0,0.0);
 bird1(); //btree
 glPopMatrix();

     glPushMatrix();
              glTranslatef(-30.0,-0.5,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
          glColor3f(1.0,1.0,0.9);
              glTranslatef(-39.0,1.0,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
  glTranslatef(-30.0,5.0,0.0);
glScalef(20.0,10.0,0.0);
 fence();
```

```
 glPopMatrix();
  glPushMatrix();
   glTranslatef(0.0,0.0,0.0);
glScalef(20.0,20.0,0.0);
 apple();
 glPopMatrix();
 // 1 cloud
                glPushMatrix();
glTranslatef(-25.0,q1,0.0);
 bubbles();
  glPopMatrix();
    glPushMatrix();
glTranslatef(-23.0,q5,0.0);
 bubbles();
  glPopMatrix();

   glPushMatrix();
glTranslatef(-28.0,q2,0.0);
 bubbles();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-31.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-33.0,q4,0.0);
 bubbles();
 glPopMatrix();

                glPushMatrix();
glTranslatef(-36.0,q1,0.0);
 bubbles();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-39.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(-41.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-15.0,q5,0.0);
 bubbles();
 glPopMatrix();
 glPushMatrix();
            glTranslatef(-20.0,q4,0.0);
 bubbles();
 glPopMatrix();

   glPushMatrix();
glTranslatef(-39.0,21.0,0.0);
 glScalef(1.0,1.0,0.0);
 cloud();
  glPopMatrix();


              //2 cloud
                 glPushMatrix();
glTranslatef(-5.0,q1,0.0);
 bubbles();
  glPopMatrix();
   glPushMatrix();
                glPushMatrix();
glTranslatef(-0.5,q4,0.0);
 bubbles();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-10.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(1.0,q3,0.0);
```

```
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(3.0,q1,0.0);
 bubbles();
 glPopMatrix();

                glPushMatrix();
glTranslatef(6.0,q1,0.0);
 bubbles();
  glPopMatrix();
   glPushMatrix();
glTranslatef(9.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(12.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(14.0,q5,0.0);
 bubbles();
 glPopMatrix();
 //3 cloud
  glPushMatrix();
glTranslatef(20.0,q4,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(22.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(25.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(28.0,q1,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(30.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(33.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(36.0,q1,0.0);
 bubbles();
 glPopMatrix();
              glPopMatrix();
  glPushMatrix();
glTranslatef(38.0,q2,0.0);
 bubbles();
  glPopMatrix();
  glPushMatrix();
glTranslatef(40.0,q3,0.0);
 bubbles();
 glPopMatrix();
  glPushMatrix();
glTranslatef(45.0,q5,0.0);
 bubbles();
 glPopMatrix();

                        glPopMatrix();
  glPushMatrix();
glTranslatef(35.0,-26.5,0.0);
glScalef(0.5,0.3,0.0);
 cloudB();
 glPopMatrix();
  glPushMatrix();
```

```
 glTranslatef(-0.5,5.5,0.0);
 glScalef(0.5,0.3,0.0);
  cloudB();
  glPopMatrix();
glPushMatrix();
 glTranslatef(-37.0,19.5,0.0);
 glScalef(0.5,0.3,0.0);
  cloudB();
  glPopMatrix();
glPushMatrix();
 glTranslatef(-26.0,2.5,0.0);
 glScalef(0.3,0.2,0.0);
  cloudB();
  glPopMatrix();
 glPushMatrix();
 glTranslatef(-1.5,21.3,0.0);
 glScalef(0.25,0.55,0.0);
   triangle1();
    glPopMatrix();
 glPushMatrix();
 glTranslatef(-8.0,-30.0,0.0);
 glScalef(0.1,0.1,0.0);
  triangle1();
    glPopMatrix();
 glPushMatrix();
 glTranslatef(10.0,22.0,0.0);
 glScalef(4.0,4.7,0.0);
  bubbles();
    glPopMatrix();
     glPushMatrix();
 glTranslatef(10.0,16.0,0.0);
 glScalef(5.0,10.0,0.0);
 bubbles();
    glPopMatrix();
 glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 glTranslatef(10.0,22.0,0.0);
 //glScalef(0.8,0.8,0.0);
 glutSolidSphere(0.5,8,11);
    glPopMatrix();
 glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 glTranslatef(8.3,22.0,0.0);
 //glScalef(1.0,1.0,0.0);
  glutSolidSphere(0.5,8,11);
    glPopMatrix();
glPushMatrix();
 glTranslatef(15.0,30.0,0.0);
 glScalef(100.0,30.0,0.0);
  apple();
    glPopMatrix();
glPushMatrix();
 glTranslatef(8.3,21.0,0.0);
 glScalef(0.09,0.09,0.0);
    triangle();
    glPopMatrix();




 glFlush();
}
void rain(){
 // Draw particles
      glColor3f(0.5, 0.5, 1.0);
      glBegin(GL_LINES);
        glVertex2f(10.0,10.8);
        glVertex2f(10.0,12.8);
      glEnd();
}
void display3()
{
```

```
 glClear(GL_COLOR_BUFFER_BIT);

    glPushMatrix();
   day2();
    glPopMatrix();
///ground///
 glPushMatrix();
 glTranslatef(0.0,-35.0,0.0);
  glScalef(3.5,3.0,0.0);
  rect();
   glPopMatrix();

    glPushMatrix();
 glTranslatef(30.0,-5.0,0.0);
  glScalef(80.0,40.0,0.0);
// glColor3f(0.0, 0.5, 0.0);
  backgroundtree();
   glPopMatrix();

    glPushMatrix();
    glColor3f (1.0,1.0,1.0);
    glTranslatef(-20.0,44.0,0.0);
                glScalef(0.9,0.8,0.0);
   //           glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();

   glPushMatrix();
    glColor3f (1.0,1.0,1.0);
    glTranslatef(10.5,44.0,0.0);
                glScalef(0.9,0.8,0.0);
   //           glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();

 glPushMatrix();
    glColor3f (1.0,1.0,1.0);
    glTranslatef(45.0,44.0,0.0);
                glScalef(0.9,0.8,0.0);
   //           glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();


glPushMatrix();
 glColor3f (1.0,1.0,1.0);
  glTranslatef(cx,17.0,0.0);
                glScalef(40.0,40.0,0.0);
   //           glTranslatef(cx,17.0,0.0);
     cloudB();
   glPopMatrix();
    glPushMatrix();
 glColor3f (1.0,1.0,1.0);
  glTranslatef(cx,20.0,0.0);
  // cloudB();
    glPopMatrix();
//tree1
 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-49.0,9.5,0.0);
   glScalef(0.4,0.5,0.0);
   tree();
    glPopMatrix();
 glPushMatrix();
 glTranslatef(-40.4,15.4,0.0);
 glScalef(0.4,0.5,0.0);
  treebody();
   glPopMatrix();  //  ///

 glPushMatrix();
 // glTranslatef(cx,0.0,0.0);
   glScalef(0.8,1.0,0.0);
    cloud();
```

```
 glPopMatrix();  //***/// / //home 2///
 glPushMatrix();
 glTranslatef(-65.0,-20.0,0.0);
 house();
 glPopMatrix();  //***/// / //home 1///
 glPushMatrix();
  glTranslatef(-45.0,-10.0,0.0);
   home1();
    glPopMatrix();
///home 2///
 glPushMatrix();
  glTranslatef(20.0,-15.0,0.0);
   home1();
    glPopMatrix();
  //***/// / //tree typeA 1///
   glPushMatrix();
   glTranslatef(-5.0,-5.0,0.0);
   glScalef(0.5,1.0,0.0);
   tree2();
   glPopMatrix();
///tree2///
 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(-44.0,-1.0,0.0);
  glScalef(0.4,0.5,0.0); tree();
   glPopMatrix();
 glPushMatrix();
  glTranslatef(-35.0,5.0,0.0);
   glScalef(0.4,0.5,0.0);
   treebody();
   glPopMatrix();
    //***/// / //********home 4********///
glPushMatrix();
 glTranslatef(-10.0,-10.0,0.0);
 home1();
 glPopMatrix();
 glPopMatrix();
   //***/// / //********road********///
     glPushMatrix();
               glTranslatef(-10.0,15.0,0.0);
                ground();
                 glPopMatrix();
glPopMatrix();
  glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  ground();
  glPopMatrix();
  glPopMatrix();
  //divider ///
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider1();
  glPopMatrix();
    glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider2();
  glPopMatrix();
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider3();
  glPopMatrix();
   glPushMatrix();
  glTranslatef(-10.0,20.0,0.0);
  divider4();
  glPopMatrix();
    //***********/// / //BUS///
 glPushMatrix();
  glTranslatef(cx, -19.0, 0.0);
   glScalef(30.0, 8.0, 0.0);
```

```
    car();
 glPopMatrix();
  //***///  ///BUS2///
 glPushMatrix();
 glTranslatef(cx, -8.0, -15.0);
  glScalef(30.0, 8.0, 0.0);
  car();
   glPopMatrix();
   glPushMatrix();
glTranslatef(160.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
 glPopMatrix();

    //***///  ///tree3///
     glPopMatrix();
    glPushMatrix();
glTranslatef(100.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();

 glPushMatrix();
 glColor3f(0.133, 0.545, 0.133);
  glTranslatef(22.0,-32.5,0.0);
   glScalef(0.4,0.5,0.0);
    tree();
 glPopMatrix();
 glPopMatrix();
 glPushMatrix();
 glTranslatef(30.5,-26.5,0.0);
  glScalef(0.4,0.5,0.0);
  treebody();
   glPopMatrix();  //***///  ///home 3///
 glPushMatrix();
  glTranslatef(-5.0,-70.0,0.0);
  house();
  glPopMatrix(); glPopMatrix();
   //***///  ///tree typeD 1///
    glPushMatrix();
glTranslatef(-10.0,-50.0,0.0);
glScalef(.5,1,0);
 tree2();
 glPopMatrix(); ///***///
 //well
 glPushMatrix();
glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
 well();
 glPopMatrix(); ///***///
                glPushMatrix();
//              glTranslatef(-12.0,-20.0,0.0);
glScalef(40.0,40.0,0.0);
// bird();
 glPopMatrix(); ///***///
glPushMatrix();
glTranslatef(-15.0,10.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
glTranslatef(-18.0,8.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-23.0,6.0,0.0);
glScalef(15.0,15.0,0.0);
```

```
 grass1();
 glPopMatrix();
glPushMatrix();
glTranslatef(-29.0,1.0,0.0);
glScalef(25.0,15.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
glTranslatef(-40.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(20.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(24.0,-2.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(45.0,3.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(20.0,-1.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(-9.0,5.0,0.0);
glScalef(20.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(4.0,1.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
                glPushMatrix();
glTranslatef(12.0,5.0,0.0);
glScalef(15.0,15.0,0.0);
 grass1();
 glPopMatrix();
  glPushMatrix();
glTranslatef(75.0,-9.5,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(100.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
   glPushMatrix();
glTranslatef(125.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
  glPushMatrix();
glTranslatef(160.0,-10.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
                 glPushMatrix();
glTranslatef(75.0,-25.0,0.0);
glScalef(65.0,15.0,0.0);
 fence();
  glPopMatrix();
```

```
glPushMatrix();
                glTranslatef(-20.4,6.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-20.4,8.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-26.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
                glPushMatrix();
                glTranslatef(-25.0,1.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///



 //right grass
   glPushMatrix();
                glTranslatef(28.5,-3.5,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix(); ///***///
   glPushMatrix();
                glTranslatef(30.0,-4.0,0.0);
glScalef(12.0,12.0,0.0);
 grass1();
 glPopMatrix();
 glPushMatrix();
                glTranslatef(10.0,-40.0,0.0);
glScalef(25.0,22.0,0.0);
 bird1(); //wood
 glPopMatrix();
   glPushMatrix();
                glTranslatef(-10.0,-33.5,0.0);
                glColor3f(0.0,0.0,0.0);
glScalef(20.0,25.0,0.0);
 bird1(); //btree
 glPopMatrix();

     glPushMatrix();
                glTranslatef(-30.0,-0.5,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
           glColor3f(1.0,1.0,0.9);
                glTranslatef(-39.0,1.0,0.0);
glScalef(20.0,20.0,0.0);
 bird1();
 glPopMatrix();
 glPushMatrix();
  glTranslatef(-30.0,5.0,0.0);
glScalef(20.0,10.0,0.0);
 fence();
 glPopMatrix();
  glPushMatrix();
  glTranslatef(0.0,0.0,0.0);
glScalef(20.0,20.0,0.0);
 apple();
 glPopMatrix();
 // 1 cloud
                  glPushMatrix();
glTranslatef(-25.0,q1,0.0);
glScalef(-0.01,1.9,0.0);
      rain();
  glPopMatrix();
```

```
    glPushMatrix();
glTranslatef(-23.0,q5,0.0);
 rain();
  glPopMatrix();

   glPushMatrix();
glTranslatef(-28.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
rain();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-31.0,q3,0.0);
rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-33.0,q4,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
 glPopMatrix();

                glPushMatrix();
glTranslatef(-36.0,q1,0.0);
 rain();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-39.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(-41.0,q3,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(-15.0,q5,0.0);
glScalef(-0.01,1.9,0.0);
rain();
 glPopMatrix();
 glPushMatrix();
            glTranslatef(-20.0,q4,0.0);
 rain();
 glPopMatrix();

   glPushMatrix();
glTranslatef(-39.0,21.0,0.0);
 glScalef(1.0,1.0,0.0);
 cloud();
  glPopMatrix();


            //2 cloud
              glPushMatrix();
glTranslatef(-5.0,q1,0.0);
glScalef(-0.01,1.9,0.0);
rain();
  glPopMatrix();
   glPushMatrix();
              glPushMatrix();
glTranslatef(-0.5,q4,0.0);
 rain();
  glPopMatrix();
   glPushMatrix();
glTranslatef(-10.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(1.0,q3,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(3.0,q1,0.0);
glScalef(-0.01,1.9,0.0);
```

```
 rain();
 glPopMatrix();

                glPushMatrix();
glTranslatef(6.0,q1,0.0);
 rain();
  glPopMatrix();
   glPushMatrix();
glTranslatef(9.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(12.0,q3,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(14.0,q5,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
 glPopMatrix();
 //3 cloud
  glPushMatrix();
glTranslatef(20.0,q4,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(22.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(25.0,q3,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(28.0,q1,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(30.0,q2,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(33.0,q3,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(36.0,q1,0.0);
 rain();
 glPopMatrix();
            glPopMatrix();
  glPushMatrix();
glTranslatef(38.0,q2,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
  glPopMatrix();
  glPushMatrix();
glTranslatef(40.0,q3,0.0);
 rain();
 glPopMatrix();
  glPushMatrix();
glTranslatef(45.0,q5,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
 glPopMatrix();

 glPushMatrix();
glTranslatef(-26.0,2.5,0.0);
glScalef(-0.01,1.9,0.0);
 rain();
```

```
 glPopMatrix();


 glFlush();

}



 void init(void)
 {
 glClearColor (0.0, 0.749, 1.0, 0.0);
 glOrtho(-50.0,50.0, -50.0, 50.0, -1.0, 1.0);
}

 void spinDisplay() {
 cx = cx +0.07;
 if(cx>70) {
 cx = -70;
}
                yw=yw-0.03;
                if(yw<13.0){
                yw=22.0;
  }
  f1-=0.05;
  if(f1<-1.0)
   {
      f1=9.9;
   }
    f2-=0.08;
if(f2<-1.0){
      f2=9.9;
}
   f3-=0.08;
if(f3<-1.0)
   {
      f3=9.9;
}
    f4-=0.05;
if(f4<-28.0)
   {
      f4=-22.2;
}
 f5-=0.08;
if(f5<-28.0)
   {
      f5=-22.2;
}
  f6-=0.08;
if(f6<-40.0)
   {
      f6=-35.0;
}


   q1-=0.1;
   if(q1<0.9)
   {
      q1=32.4;//y
   }

   q2-=0.09;
   if(q2<-25.0)
   {
     // p2=0.0;
      q2=32.4;
   }
   q3-=0.25;
   if(q3<-28.0)
   {
     // p3=0.0;
      q3=32.4;
   }
```

```
   q4-=0.3;
if(q4<-30.0)
   {
     // p3=0.0;
      q4=32.4;
   }
     q5-=0.09;
if(q5<13.0)
   {
     // p3=0.0;
      q5=40.5;

   }

 glutPostRedisplay();
  }
 void mouse(int key, int state, int x, int y) {
 switch(key) {
 case GLUT_LEFT_BUTTON: if (state == GLUT_DOWN) {
 glutIdleFunc(spinDisplay);
 }
 break;
 case GLUT_MIDDLE_BUTTON: case GLUT_RIGHT_BUTTON: if (state == GLUT_DOWN) {
 glutIdleFunc(NULL); }
 break;
 default: break; } }
 void keyDisplayAnimationDay() {
 str=500.0;
  sr=0.0;
 sg=0.749;
  sb=1.0;
 ss=0.0;
  mn=500.0;
//  glPushMatrix();
//disp();
//glPopMatrix();
   glutPostRedisplay();
   }
  void handleKeypress(unsigned char key, int x, int y) {
 switch (key) {
                case 'm':
                glutDestroyWindow(1);
          glutInitWindowSize (1350, 690);
           glutInitWindowPosition (0, 0);
       glutCreateWindow("village scenario");
          init();
          glutKeyboardFunc(handleKeypress);
        glutMouseFunc(mouse);
          glutDisplayFunc(StartingText);
          glutPostRedisplay();
          break;
    case  's':
      // glutDestroyWindow(1);
        glutInitWindowSize (1350, 690);
           glutInitWindowPosition (0, 0);
       glutCreateWindow("village scenario");
       init();
       glutKeyboardFunc(handleKeypress);
        glutMouseFunc(mouse);
          glutDisplayFunc(display1);
           glutPostRedisplay();
          break;

       case 'w':
          glutDestroyWindow(1);
          glutInitWindowSize (1350, 690);
           glutInitWindowPosition (0, 0);
       glutCreateWindow("village scenario");
          init();
          glutKeyboardFunc(handleKeypress);
        glutMouseFunc(mouse);
          glutDisplayFunc(display2);
          glutPostRedisplay();
```

```
            break;
        case 'f':
                glutDestroyWindow(1);
            glutInitWindowSize (1350, 690);
             glutInitWindowPosition (0, 0);
         glutCreateWindow("village scenario");
            init();
            glutKeyboardFunc(handleKeypress);
          glutMouseFunc(mouse);
            glutDisplayFunc(disp);
            glutPostRedisplay();
            break;

        case 'r':
                glutDestroyWindow(1);
            glutInitWindowSize (1350, 690);
             glutInitWindowPosition (0, 0);
         glutCreateWindow("village scenario");
            init();
            glutKeyboardFunc(handleKeypress);
          glutMouseFunc(mouse);
            glutDisplayFunc(display3);
            glutPostRedisplay();
            break;

      glutPostRedisplay();
}
}
 int main( int argc,char ** argv)
 {
 printf(">><< Press N for nightmood>><< Press D for day mood >><<\n\n");
  printf("Click Mouse Left/Right Butt on for cloud movement\n\n");
  glutInit(&argc,argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (1350, 690);
    glutInitWindowPosition (0, 0);
glutCreateWindow("My House ");
init();
 glutDisplayFunc(StartingText);
 glutDisplayFunc(disp);
  glutMouseFunc(mouse);
   glutKeyboardFunc(handleKeypress);
    glutMainLoop();
 return 0;
}
```

9.2 Appendix (Figures):

SL NO	FIGURE NO	PAGE NO	FIG NAME
1	2.1	4	Basic Graphics System
2	2.2	5	Opengl rendering pipeline
3	4.1	10	Fig of eye(Rhodesand cones)
4	6.1.1	17	Image of house object
5	6.1.2	17	Image of car object
6	6.2.1	19	Snapshot of initial position of weather simulator
7	6.2.2	19	Snapshot representing summer season
8	6.2.3	20	Snapshot representing winter season
9	6.2.4	20	Snapshot representing spring season
10	6.2.5	21	Snapshot representing rainy season