

**American International University-Bangladesh (AIUB) Department of Computer Science**

(Fall 2020-21)

**Project: River-Side Scenario**

**Computer Graphics**

Section: ​**E**

Prepared & Submitted by:

Group: ​**-**

|  |  |  |
| --- | --- | --- |
| **SL.** | **Name** | **ID** |
| **01** | **ESHEAK AHMED** | **18-38088-2** |
| **02** | **MD RIFAT UZ ZAMAN** | **18-38024-2** |
| **03** | **NUSART JAHAN** | **18-38099-2** |
| **04** | **RUBAIA AKTER TONNI** | **18-38024-2** |

**Submitted​** ​**to:**

**ANEEM AL AHSAN RUPAI**

**Date of Submission:**

00-00-2020

## **Table of contents:**

In Our Project, we have used lots of things which we did not use before like OpenGL. We used some algorithms, color models, different kinds of shapes, additional libraries, plotted the value in the graph, and implemented it graphically. There is some information that how we get the knowledge of these staff is given below.

OpenGL (Open Graphics Library) is the computer industry's standard application program interface (API) for defining 2-D and 3-D graphic images. Prior to OpenGL, any company developing a graphical application typically had to rewrite the graphics part of it for each operating system platform and had to be cognizant of the graphics hardware as well. With OpenGL, an application can create the same effects in any operating system using any OpenGL-adhering graphics adapter. OpenGL specifies a set of "commands" or immediately executed functions. Each command directs a drawing action or causes special effects.

A list of these commands can be created for repetitive effects. OpenGL is independent of the windowing characteristics of each operating system. But it provides special "glue" routines for each operating system that enable OpenGL to work in that system's windowing environment. OpenGL comes with many built-in capabilities requestable through the API. These include hidden surface removal, alpha blending (transparency), antialiasing, texture mapping, pixel operations, viewing and modeling transformations, and atmospheric effects.

On the most fundamental level, OpenGL is a software interface that allows a programmer to communicate with graphics hardware.

We used some functions and loops over there for using C++ language C++ is a cross-platform language that can be used to create high-performance applications. It was developed by Bjarne Stroustrup, as an extension to the C language. C++ gives programmers a high level of control over system resources and memory.

We studied two types of color models which are RGB and CMYK.

* **RGB Color Model**: ​ The RGB color model is composed of the primary colors Red, Green, and Blue. This system defines the color model that is used in most color CRT monitors and color raster graphics.

* **CMYK Color Model**: ​ The CMYK color model stands for Cyan, Magenta, and Yellow which are the complements of red, green and blue respectively. This system is used primarily for printing.

We used some shapes like rectangle, polygon, circle defining via vertex value for every pixel we used different values for the position. We use some additional libraries like math.h and pi to implement the code easily. These are built-in libraries we just called once in code for understanding the compiler that we use some built-in function over there.

# **Introduction:**

As the goal of the project was to develop a river-side scene. In this project, we are trying to show a simple riverside scenario here by using OpenGL. The project is implemented by using the C++ programming language in Code Blocks (13.12) IDE. A visual C++ compiler is required to show this scenario. We know that Computer graphics is the backbone of any design and any visual representation. In this project, we design some houses, trees, clouds, roads, sun, boats, cars, a plane, windmills, sky, river, etc. We want to show the usual picture of a riverside small town on a sunny day. This project is well suited for designing 2D objects, as well as for carrying out basic graphic functionalities like drawing simple Quads, Polygons, Triangles. The project is enabled to work with mid-level OpenGL complexity. In this project, cars and boats are moving. We use the necessary color to visualize this scenario properly. And we use necessary library functions, user-defined functions to implement this scenario properly.

**Project Proposal:**

Project Name: Rainy Season Scenario

Features Description

Feature List

1. Simple and complex shapes

2. Movement

3.Rotation

4. Rain Mode

5. Day Mode

We have demonstrated a Computer Graphics (CG) project developed using Opengl (C++). where a scene represents day view of moving clouds, cars, boats, and planes also raining view are implemented in OpenGL. In this scenario, in the day view boat will be moving in the river which can be controlled by the user or operator. Also, here clouds, planes, and cars will be moving. In this scenario, the sun is shining brightly. These are the main part of the riverside scenario project. This scenario also has many more objects or features like houses, hilly areas, rivers, trees, bench, traffic signals, roads, grass areas, sky, windmills, sun, clouds. In this Rainy Season scenario project, the main objective thing is an animation of day view which makes this computer graphics simple project very beautiful, colorful, and effective. Out of the many features available, the project demonstrates some popular and commonly used features of OpenGL, such as Rendering. Transformation, Rotation. Lighting. Scaling etc. We use necessary library functions, user-defined functions, and loop which is implemented by using the C++ language. We want to slow the usual picture of a riverside small town on a sunny day. This project is well suited for designing 2D objects, as well as for carrying out basic graphic functionalities like drawing simple Quads, Polygons, Circles, Triangles. The project is enabled to work with mid-level OpenGL complexity. OpenGL is good for low cost and simple scenario development. It serves as an important steppingstone for venturing into other fields of Computer Graphics design and applications.

Top of Form

Bottom of Form

Aa

**Schematic Diagram:**

**List of Objects:**

In this project, we want to show a clear riverside scenario. We make this view by using OpenGL. OpenGL is a software interface that allows a programmer to communicate with graphics hardware. Out of the many features available, the project demonstrates some popular and commonly used features of OpenGL, such as Rendering, Transformation, Rotation, Lighting, Scaling, etc. We use necessary library functions, user-defined functions, and loop here which is implemented by using the C++ language. We want to show the usual picture of a riverside small town on a sunny day. In this project, we design some houses, trees, clouds, roads, sun, boats, cars, planes, windmills, sky, river, etc. This project is well suited for designing 2D objects, as well as for carrying out basic graphic functionalities like drawing simple Quads, Polygons, Triangles. The project enabled me to work with mid-level OpenGL complexity.

In this project, cars and boats are moving. We use the necessary color to visualize this scenario properly. We use two types of color models. We use RGB and CMYK models to show the proper color.

Code:

|  |
| --- |
| #include <iostream>  #include <windows.h>  #include <GL/glut.h>  #include <math.h>    #define PI 3.14    using​ ​namespace​ ​std​;    void​ ​drawSky​(​void​)  {  glBegin(GL\_QUADS);  glColor3ub(​224​, ​224, ​ 224​ )​ ; //(166, 189, 219); // Maximum Green​ glVertex2i(​0​, ​800​); glVertex2i(​1050​, ​800​); glVertex2i(​1050​, ​1000​); glVertex2i(​0​, ​1000​); glEnd();  }  void​ ​drawSkyBottom​(​void​)  {  glBegin(GL\_QUADS);  glColor3ub(​30​, ​86​, ​49​); ​// Maximum Green glVertex2i(​0​, ​800​); glVertex2i(​1050​, ​750​); glVertex2i(​1050​, ​900​); glVertex2i(​900​, ​860​); glVertex2i(860​ , ​ 820​ )​ ; glVertex2i(​800​, ​870​); glVertex2i(​700​, ​870​); glVertex2i(​600​, ​820​); glVertex2i(​0​, ​840​); glEnd();  }  void​ ​drawRiver​(​void​)  { glBegin(GL\_QUADS); |

glColor3ub(​102​, ​178, ​ 255​ )​ ; glVertex2i(​0​, ​600​); glVertex2i(​1050​, ​600​); glVertex2i(​1050​, ​800​); glVertex2i(​0​, ​800​); glEnd();

}

void​ ​drawBoatOne​(​void​)

{ ​// Part one glBegin(GL\_POLYGON); glColor3ub(​0, ​ ​0​, ​0​); glVertex2i(​50​, ​720​); glVertex2i(​120​, ​720​); glVertex2i(​130​, ​740​); glVertex2i(​40, ​ 740​ ​); glEnd(); ​// Part two glBegin(GL\_POLYGON); glColor3ub(​255​, ​255​, ​0​); glVertex2i(​60​, ​740​); glVertex2i(​110​, ​740​); glVertex2i(​90, ​ 750​ )​ ; glVertex2i(​80​, ​750​); glEnd(); ​// Part three glBegin(GL\_QUADS); glColor3ub(​213​, ​96​, ​5​); glVertex2i(​50​, ​740​); glVertex2i(​55​, ​740​); glVertex2i(​55​, ​780​); glVertex2i(​50​, ​780​); glEnd();

}

void​ ​drawBoatTwo​(​void​)

{ ​// Part one glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​0​); glVertex2i(​850​, ​770​); glVertex2i(​920, ​ ​770​); glVertex2i(​930​, ​790​); glVertex2i(​840​, ​790​); glEnd();

​// Part two

glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​255​); glVertex2i(​860​, ​790​); glVertex2i(​910​, ​790​); glVertex2i(​890​, ​800​); glVertex2i(​880​, ​800​); glEnd();

​// Part three glBegin(GL\_QUADS); glColor3ub(​213​, ​96​, ​5​); glVertex2i(​850​, ​790​); glVertex2i(​855​, ​790​); glVertex2i(​855​, ​830​); glVertex2i(​850​, ​830​); glEnd();

}

void​ ​drawGrass​(​void​)

{

glBegin(GL\_QUADS);

glColor3ub(​76​, ​154​, ​42​); ​// Maximum Green glVertex2i(​0​, ​220​); glVertex2i(​1050​, ​220​); glVertex2i(1050​ , ​ 600​ )​ ; glVertex2i(​0​, ​600​); glEnd();

}

void​ ​drawLandShades​(​void​)

{

​//land shade 1 glBegin(GL\_POLYGON);

glColor3ub(​46​, ​139​, ​87​); ​// Maximum Green glVertex2i(​1050​, ​220​); glVertex2i(​1050​, ​320​); glVertex2i(​690​, ​220​); glEnd();

​// land shade2 glBegin(GL\_POLYGON);

glColor3ub(​46​, ​224​, ​87​); ​// Maximum Green glVertex2i(​1050​, ​330)​ ; glVertex2i(​0​, ​300​); glVertex2i(​0​, ​650​); glVertex2i(​1050​, ​380​); glEnd();

​//land shade 3 glBegin(GL\_POLYGON);

glColor3ub(​0​, ​204​, ​102​); ​// Maximum Green glVertex2i(​1050​, ​620​); glVertex2i(​1050​, ​500​); glVertex2i(​500​, ​490​); glVertex2i(​500​, ​400​); glEnd();

}

void ​ drawRoads​ (​ void​ )​

{ ​//Road 1 glBegin(GL\_QUADS); glColor3ub(​96​, ​106​, ​116​); ​// Red glVertex2i(​0​, ​10​); glVertex2i(​1050​, ​10​); glVertex2i(​1050​, ​200​); glVertex2i(​0​, ​200​); glEnd(); ​//Road 2 glBegin(GL\_POLYGON); glColor3ub(​96​, ​106​, ​116​); glVertex2i(​520​, ​200​); glVertex2i(​690​, ​200​); glVertex2i(​480​, ​600​); glVertex2i(​370​, ​600​); glEnd(); ​//Road 3 glBegin(GL\_POLYGON); glColor3ub(​96​, ​106​, ​116​); glVertex2i(​0​, ​600​); glVertex2i(​500​, ​530)​ ; glVertex2i(​480​, ​605​); glVertex2i(​0​, ​630​); glEnd();

}

void​ ​drawRoadSigns​(​void​)

{

​//sign 1

glBegin(GL\_QUADS); ​// Each set of 4 vertices form a quad

glColor3ub(​255​, ​255​, ​255​); ​// Red glVertex2i(​50​, ​100​); glVertex2i(​200​, ​100​); glVertex2i(​200​, ​110​); glVertex2i(​50​, ​110​); glEnd();

​//sign 2

glBegin(GL\_QUADS); ​// Each set of 4 vertices form a quad glColor3ub(​255​, ​255​, ​255​); ​// white glVertex2i(​250​, ​100​); glVertex2i(​400​, ​100​); glVertex2i(​400​, ​110​); glVertex2i(​250​, ​110​);

glEnd();

​//sign 3

glBegin(GL\_QUADS); ​// Each set of 4 vertices form a quad

glColor3ub(​255​, ​255​, ​255​); ​// white glVertex2i(​450​, ​100​); glVertex2i(​600​, ​100​); glVertex2i(600​ , ​ 110​ )​ ; glVertex2i(​450​, ​110​); glEnd();

​//sign 4

glBegin(GL\_QUADS); ​// Each set of 4 vertices form a quad glColor3ub(​255​, ​255​, ​255​); ​// white glVertex2i(​650​, ​100​); glVertex2i(​800​, ​100​); glVertex2i(​800​, ​110​); glVertex2i(​650​, ​110​);

glEnd();

​//sign 5 glBegin(GL\_QUADS); ​// Each set of 4 vertices form a quad

glColor3ub(​255​, ​255​, ​255​); ​// white

glVertex2i(​850​, ​100​); glVertex2i(​990​, ​100​); glVertex2i(​990​, ​110​); glVertex2i(​850​, ​110​);

glEnd();

}

void​ ​drawCarOne​(​void​)

{

glBegin(GL\_QUADS); glColor3ub(​255​, ​255​, ​204​); glVertex2i(​795​, ​50​); glVertex2i(​885​, 50​ )​ ; glVertex2i(​880​, ​80​); glVertex2i(​800​, ​80​); glEnd();

glBegin(GL\_POLYGON); glColor3ub(​240​, ​128​, ​128​); glVertex2i(​800​, ​80​); glVertex2i(​870​, 80​ )​ ; glVertex2i(​850​, ​110​); glVertex2i(​810​, ​110​); ​//glVertex2i(720,100); glEnd();

​//wheel 1 glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{ glColor3ub(​1.0​, ​0​, ​1.0​); ​float​ pi = 3.1416​ ;​

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​815​; ​float​ y = r \* ​sin​(A) + ​50​; glVertex2f(x, y);

}

glEnd();

​//wheel 2 glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​1.0​, ​0​, ​1.0​); ​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​865​; ​float​ y = r \* ​sin​(A) + ​50​; glVertex2f(x, y);

}

glEnd();

}

void​ ​drawCarTwo​(​void​)

{ glBegin(GL\_QUADS); glColor3ub(​255​, ​255, ​ 204​ )​ ; glVertex2i(​95​, ​50​); glVertex2i(​185​, ​50​); glVertex2i(​180​, ​80​); glVertex2i(​100​, ​80​); glEnd();

glBegin(GL\_POLYGON); glColor3ub(​53​, ​153​, ​255​); glVertex2i(​100​, ​80​); glVertex2i(​170​, ​80​); glVertex2i(​150​, ​110​); glVertex2i(​110​, ​110​); ​//glVertex2i(720,100); glEnd(); // wheel 1 glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{ glColor3ub(​1.0​, ​0​, ​1.0​); ​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float x = r \* ​ cos​ (​ A) + 115​ ;​ ​float​ y = r \* ​sin​(A) + ​50​; glVertex2f(x, y);

} glEnd(); ​//wheel 2

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​1.0​, ​0​, ​1.0​); ​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​165​; ​float​ y = r \* ​sin​(A) + ​50​; glVertex2f(x, y);

}

glEnd();

}

void​ ​drawCarThree​(​void​)

{ glBegin(GL\_QUADS); glColor3ub(​255​, ​255​, ​204​); glVertex2i(​475​, ​150​); glVertex2i(​595​, ​150​); glVertex2i(​590​, ​180​); glVertex2i(​480​, ​180​); glEnd();

glBegin(GL\_POLYGON); glColor3ub(​153​, ​153​, ​255​); glVertex2i(​500​, ​180​); glVertex2i(​590​, ​180​); glVertex2i(​570​, ​210​); glVertex2i(​510​, ​210​); ​//glVertex2i(720,100); glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​1.0​, ​0​, ​1.0​); ​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​505​;

​float​ y = r \* ​sin​(A) + ​150​;

glVertex2f(x, y);

} glEnd(); glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{ glColor3ub(1.0​ , ​ 0​ , ​ 1.0​ )​ ; ​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​570​; ​float​ y = r \* ​sin​(A) + ​150​; glVertex2f(x, y);

} glEnd();

}

void​ ​drawHomes​(​void​)

{

​//1st home part1 glBegin(GL\_POLYGON); glColor3ub(​225​, ​204​, 229​ )​ ; // Maximum Green​ glVertex2i(​600​, ​550​); glVertex2i(​660​, ​530​); glVertex2i(​660​, ​590​); glVertex2i(​630​, 640​ )​ ; glVertex2i(​600​, ​590​); glEnd();

​// part 2 glBegin(GL\_POLYGON);

glColor3ub(​102​, ​0​, ​102​); ​// Maximum Green glVertex2i(​630​, ​640​); glVertex2i(​640​, ​630​); glVertex2i(​600​, ​585​); glVertex2i(​590​, ​585​); glEnd();

​// part 3 glBegin(GL\_POLYGON); glColor3ub(​102​, ​0​, ​102​); ​// Maximum Green glVertex2i(660​ , ​ 580​ )​ ; glVertex2i(​720​, ​590​);

glVertex2i(​740​, ​640​); glVertex2i(​630​, ​640​); glEnd();

​// part 4 glBegin(GL\_POLYGON); glColor3ub(​255​, ​204​, ​255​); ​// Maximum Green glVertex2i(​660​, ​530​); glVertex2i(​720​, ​545)​ ; glVertex2i(​720​, ​590​); glVertex2i(​660​, ​580​); glEnd();

​//first home door 1` glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​0​); ​// Maximum Green glVertex2i(​610​, ​545​); glVertex2i(​640​, ​535​); glVertex2i(​640​, ​580​); glVertex2i(​610​, ​585​); glEnd();

​// 2nd home part 1 glBegin(GL\_POLYGON); glColor3ub(​255​, ​255​, ​255​); ​// Maximum Green glVertex2i(​720​, ​530​); glVertex2i(​800​, ​520​); glVertex2i(​800​, ​600​); glVertex2i(​760​, ​670​); glVertex2i(​720​, ​595​); glEnd();

​// part 2 glBegin(GL\_POLYGON); glColor3ub(​255​, ​51​, ​51​); ​// Maximum Green glVertex2i(​770​, ​660​); glVertex2i(​760​, ​670)​ ; glVertex2i(​710​, ​595​); glVertex2i(​725​, ​605​); glEnd();

​// part 3 glBegin(GL\_POLYGON);

glColor3ub(​255​, ​51​, ​51​); ​// Maximum Green glVertex2i(800​ , ​ 595​ )​ ; glVertex2i(​900​, ​600​);

glVertex2i(​860​, ​670​); glVertex2i(​760​, ​670​); glEnd();

​// part 4 glBegin(GL\_POLYGON); glColor3ub(​153​, ​204​, ​255​); ​// Maximum Green glVertex2i(​800​, ​520​); glVertex2i(​890​, ​540​); glVertex2i(​890​, ​600​); glVertex2i(​800​, ​595​); glEnd();

​// 2nd home door 2 glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​0​); ​// Maximum Green glVertex2i(​745​, ​528​); glVertex2i(​775​, ​525​); glVertex2i(​775​, ​580​); glVertex2i(​745​, ​576​); glEnd();

​// 3rd home(part 1) glBegin(GL\_POLYGON); glColor3ub(​255​, ​204​, ​204​); glVertex2i(​0​, ​480​); glVertex2i(​100​, ​450​); glVertex2i(​100​, ​530​); glVertex2i(​0​, ​540​); glEnd();

​// 3rd home (part 2) glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​102​); glVertex2i(​0​, ​540​); glVertex2i(​100​, ​530​); glVertex2i(​150​, ​600​); glVertex2i(​0​, ​590​); glVertex2i(​150​, ​600​); glVertex2i(​200​, ​530​); glVertex2i(​200​, ​530​); glVertex2i(​190​, ​530​); glVertex2i(​190​, ​530​); glVertex2i(​130​, ​600​); glEnd();

​// 3rd home (part 3) glBegin(GL\_POLYGON); glColor3ub(​192​, ​192​, ​192​); glVertex2i(​100​, ​450​); glVertex2i(​100​, 530​ )​ ; glVertex2i(​140​, ​590​); glVertex2i(​190​, ​530​); glVertex2i(​200​, ​480​); glEnd();

​// 4th home (part 1) glBegin(GL\_POLYGON); glColor3ub(​204​, ​0​, ​0)​ ; glVertex2i(​150​, 500​ ​); glVertex2i(​250​, ​490​); glVertex2i(​300​, ​580​); glVertex2i(200​ ​, ​575​); glEnd();

​// 4th home (part 2) glBegin(GL\_POLYGON); glColor3ub(​255​, ​229​, ​204​); glVertex2i(​170​, ​500​); glVertex2i(​250​, ​490​); glVertex2i(​250, ​ 400​ )​ ; glVertex2i(​170​, ​440​); glEnd();

​// 4th home (part 3) glBegin(GL\_POLYGON); glColor3ub(​204​, ​209​, ​255​); glVertex2i(​250​, ​400​); glVertex2i(​350​, ​420​); glVertex2i(​350​, ​490​); glVertex2i(​300​, ​580​); glVertex2i(​250​, ​490​); glEnd();

​// 4th home (part 4) glBegin(GL\_POLYGON); glColor3ub(204​ , ​ 0​ , ​ 0​ )​ ; glVertex2i(​360​, ​480​); glVertex2i(​300​, ​580​); glVertex2i(​290​, ​570​); glVertex2i(​340​, 480​ ​); glEnd();

​//4th home (door) glBegin(GL\_POLYGON); glColor3ub(​0​, ​0​, ​0​); glVertex2i(​280​, ​405​); glVertex2i(​325​, ​410​); glVertex2i(​320​, ​490​); glVertex2i(​280​, ​485​);

glEnd();

}

void​ ​drawTrees​(​void​)

{

​//tree 1(part 1) glBegin(GL\_POLYGON);

glColor3ub(​153​, ​76​, ​0​); ​// Maximum Green glVertex2i(​540​, ​500​); glVertex2i(​560​, ​500​); glVertex2i(​560, ​ 580​ )​ ; glVertex2i(​540​, ​580​); glEnd();

​//tree1 (leaf 1) glBegin(GL\_POLYGON); glColor3ub(​0​, ​51​, ​25​); ​// Maximum Green glVertex2i(​510​, ​580​); glVertex2i(​590​, ​580)​ ; glVertex2i(​550​, ​630​); glEnd();

​//tree1 (leaf 2) glBegin(GL\_POLYGON); glColor3ub(​0​, ​102​, ​0​); ​// Maximum Green glVertex2i(​510​, ​610​); glVertex2i(​590​, ​610​); glVertex2i(​550​, ​660​); glEnd();

​//tree1 (leaf 3) glBegin(GL\_POLYGON); glColor3ub(​76​, ​153​, ​0​); ​// Maximum Green glVertex2i(​510​, ​640​); glVertex2i(​590​, ​640​); glVertex2i(​550​, ​740​);

glEnd();

​//tree 2( PART 1) glBegin(GL\_POLYGON); glColor3ub(​102​, ​0​, ​0​); ​// Maximum Green glVertex2i(​920​, ​570​); glVertex2i(​930​, ​570​); glVertex2i(​930​, ​620​); glVertex2i(​920​, ​620​); glEnd();

​// tree 2 (leaf 1) glBegin(GL\_TRIANGLES); glColor3ub(​76​, ​153​, ​0​); ​// Maximum Green glVertex2i(​900​, ​620​); glVertex2i(​950​, ​620​); glVertex2i(​920​, ​700)​ ; glEnd(); ​// tree 2(leaf 2) glBegin(GL\_TRIANGLES); glColor3ub(​0​, ​102, ​ 0​ )​ ; // Maximum green​ glVertex2i(​900​, ​660​); glVertex2i(​950​, ​660​); glVertex2i(​925​, ​720​); glEnd();

​//corner tree (part 1) glBegin(GL\_POLYGON); glColor3ub(​315​, ​0​, ​0​); glVertex2i(​1020​, ​480​); glVertex2i(​1030​, ​400​); glVertex2i(​990​, ​400​); glVertex2i(​1000​, ​480​); glVertex2i(​970​, ​560​); glVertex2i(​975​, ​565​); glVertex2i(​985​, ​525​); glVertex2i(​995, ​ 525​ )​ ; glVertex2i(​1000​, ​675​); glVertex2i(​1005​, ​535​); glVertex2i(​1050​, ​580​); glVertex2i(​1000​, ​460​); glVertex2i(​980​, ​400​); glEnd();

​//corner tree (bottom part) glBegin(GL\_POLYGON); glColor3ub(​40​, ​0​, 0​ )​ ; // Maximum Green​ glVertex2i(​1030​, ​400​); glVertex2i(​1050​, ​420​); glVertex2i(​1050​, ​380​); glVertex2i(​1030​, ​380​); glVertex2i(​1010​, ​370​); glVertex2i(​900​, ​380​); glVertex2i(​990​, ​400​); glVertex2i(​1000​, 430​ ​); glEnd();

​//corner tree (leaf 1) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​51​, ​95​, ​0​);

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​56.85​;

​float​ x = r \* ​cos​(A) + 1010​ ;​ ​float​ y = r \* ​sin​(A) + ​655​; glVertex2f(x, y);

} glEnd();

​//corner tree (leaf 2) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​51​, ​90​, ​0​);

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​42.85​;

​float​ x = r \* ​cos​(A) + ​1050​; ​float​ y = r \* ​sin​(A) + ​727​; glVertex2f(x, y);

} glEnd();

​//corner tree (leaf 2) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​51​, ​95​, 0​ )​ ;

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​44.85​;

​float​ x = r \* ​cos(​ A) + 1040​ ;​ ​float​ y = r \* ​sin​(A) + ​680​; glVertex2f(x, y);

} glEnd();

​//corner tree (leaf 3)

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​51​, ​95​, ​0​);

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​31.85​;

​float​ x = r \* ​cos​(A) + ​970​; ​float​ y = r \* ​sin​(A) + ​580​; glVertex2f(x, y);

} glEnd();

​//corner tree (leaf 4) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for (​ int​ i = ​ 0​ ; i < ​ 200​ ; i++)​

{

glColor3ub(​51​, ​97​, ​0​);

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2 \* pi) / ​ 200​ ;​

​float​ r = ​48.85​;

​float​ x = r \* ​cos​(A) + ​1050​; ​float​ y = r \* ​sin​(A) + ​590​; glVertex2f(x, y);

} glEnd();

​// tree 3(part 1) glBegin(GL\_POLYGON); glColor3ub(​102​, ​0​, ​0​); glVertex2i(​400​, ​490​); glVertex2i(​420​, ​490​); glVertex2i(​420​, ​380​); glVertex2i(​400​, ​380​); glEnd();

​// tree 3 (leaf 1)

glBegin(GL\_POLYGON); glColor3ub(​0​, ​51​, ​25​); glVertex2i(​340​, ​490​); glVertex2i(​480​, ​490​); glVertex2i(​410​, ​600​); glEnd();

​// tree 3 (leaf 2)

glBegin(GL\_POLYGON); glColor3ub(​0​, ​51​, ​25​); glVertex2i(​340​, ​530​); glVertex2i(​480​, ​530​); glVertex2i(​410​, ​650​); glEnd();

​// tree 3 (leaf 3)

glBegin(GL\_POLYGON); glColor3ub(​0​, ​51​, ​25​); glVertex2i(​340​, ​570​); glVertex2i(​480​, ​570​); glVertex2i(​410​, ​700​); glEnd();

}

void​ ​drawCloudOne​(​void​)

{

​//cloud 1

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​241​, ​241​, ​241​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416;​

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​30.85​;

​float​ x = r \* ​cos​(A) + ​350​; ​float​ y = r \* ​sin​(A) + ​920​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​240​, ​240​, ​240​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​30.85​;

​float​ x = r \* ​cos​(A) + ​320​; ​float​ y = r \* ​sin​(A) + ​900​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​240​, ​240​, ​240​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​30.85​;

​float​ x = r \* ​cos​(A) + ​380;​ ​float​ y = r \* ​sin​(A) + ​880​; glVertex2f(x, y);

} glEnd();

}

void​ ​drawCloudTwo​(​void​)

{ //cloud 2

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​240​, ​240, ​ 240​ )​ ; //249,215,28 //255,140,0​

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​38.85​;

x = r \* cos​ (​ A) + 70​ ;​ y = r \* ​sin​(A) + ​950​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​240​, ​240​, ​240​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float r = ​ 38.85​ ;​

​float​ x = r \* ​cos​(A) + ​30​; ​float​ y = r \* ​sin​(A) + ​925​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); // Draw a Red 1x1 Square centered at​ origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(240​ , ​ 240​ , ​ 240​ )​ ; ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​38.85​;

​float​ x = r \* cos​ (​ A) + 100​ ;​ ​float​ y = r \* ​sin​(A) + ​910​; glVertex2f(x, y);

} glEnd();

}

​void​ ​drawCloudThree​(​void​)

{

​//cloud 3

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​249​, ​249​, ​249​); ​//249,215,28 //255,140,0

pi = ​3.1416​;

A = (i \* ​2​ \* pi) / ​200​;

​float​ r = 35.85​ ​;

​float​ x = r \* ​cos​(A) + ​790​; ​float​ y = r \* ​sin​(A) + ​950​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(249​ , ​ 249​ , ​ 249​ )​ ; //249,215,28 //255,140,0​

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​28.85​;

​float x = r \* ​ cos​ (​ A) + 760​ ;​ ​float​ y = r \* ​sin​(A) + ​930​; glVertex2f(x, y);

} glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{ glColor3ub(​249​, ​249​, ​249​); ​//249,215,28 //255,140,0

​float pi = ​ 3.1416​ ;​

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​27.85​;

​float​ x = r \* ​cos​(A) + ​835​; ​float​ y = r \* ​sin​(A) + ​970​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​249​, ​249​, ​249​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

A = (i \* ​2​ \* pi) / 200​ ​; r = ​27.85​;

​float​ x = r \* ​cos​(A) + 835​ ;​ ​float​ y = r \* ​sin​(A) + ​940​; glVertex2f(x, y);

} glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​249​, ​249​, ​249​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​25.85​;

​float​ x = r \* ​cos​(A) + ​865​; ​float​ y = r \* ​sin​(A) + ​945​; glVertex2f(x, y);

}

glEnd();

}

void​ ​drawSun​(​void​)

{

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​255​, ​205​, ​40​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​30.85​;

​float​ x = r \* ​cos​(A) + ​350​; ​float​ y = r \* ​sin​(A) + ​860​; glVertex2f(x, y);

}

glEnd();

}

void​ ​drawSignals​(​void​)

{

​//Signal (part 1)

glBegin(GL\_QUADS);

glColor3ub(​0​, ​0​, ​0​);

glVertex2i(​98​, ​420​); glVertex2i(​87​, ​420​); glVertex2i(​87​, ​200​); glVertex2i(​98​, ​200​);

glEnd();

​//Signal (part 2) glBegin(GL\_QUADS); glColor3ub(​172​, ​172​, 172​ )​ ; glVertex2i(​105​, ​420​); glVertex2i(​80​, ​420​); glVertex2i(​80​, ​340​); glVertex2i(​105​, ​340​); glEnd();

​//SIGNAL ( RED light) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​255, ​ 0​ , ​ 0​ )​ ; //249,215,28 //255,140,0​

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​92​; ​float​ y = r \* ​sin​(A) + ​405​; glVertex2f(x, y);

} glEnd();

​//SIGNAL (yellow light) glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{ glColor3ub(​255​, ​255​, ​0​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​92​; ​float​ y = r \* ​sin​(A) + ​380​; glVertex2f(x, y);

} glEnd();

​//SIGNAL (Green light)

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​0​, ​153​, ​0​); //249,215,28 //255,140,0​

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​10.85​;

​float​ x = r \* ​cos​(A) + ​92​; ​float​ y = r \* ​sin​(A) + ​355​; glVertex2f(x, y);

} glEnd();

}

void​ ​drawBench​(​void​)

{

​// bench (part 1) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​780​, ​425​); glVertex2i(​775​, ​425​); glVertex2i(​775​, ​388​); glVertex2i(​780​, ​388​);

glEnd();

​// bench (part 2) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​700​, ​425​); glVertex2i(​705​, ​425​); glVertex2i(​705​, ​388​); glVertex2i(​700​, ​388​); glEnd();

​// bench (part 3)

glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​800​, ​420​); glVertex2i(​680​, ​420​); glVertex2i(​680, ​ 413​ )​ ; glVertex2i(​800​, ​413​); glEnd();

​// bench (part 4) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, 0​ )​ ; glVertex2i(​800​, ​410​); glVertex2i(​680​, ​410​); glVertex2i(​680​, ​403​); glVertex2i(​800​, ​403​); glEnd(); ​// bench (part 5)

glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​800​, ​400​); glVertex2i(​680​, ​400​); glVertex2i(​680, ​ 393​ )​ ; glVertex2i(​800​, ​393​); glEnd(); ​// bench (part 6) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​800​, ​388​); glVertex2i(​680​, ​388​); glVertex2i(​670​, ​368​); glVertex2i(​810​, ​368​); glVertex2i(​670​, ​368​); glVertex2i(​810​, ​368​); glVertex2i(​810​, ​360​); glVertex2i(​670​, ​360​); glEnd(); ​// bench (part 7) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​680​, ​360​); glVertex2i(​685​, ​360​); glVertex2i(​687​, ​330​); glVertex2i(​680​, ​330​); glEnd(); ​// bench (part 8) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​690​, ​360​); glVertex2i(​695​, ​360​); glVertex2i(​697​, ​340​); glVertex2i(​690​, ​340)​ ; glEnd();

​// bench (part 9) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​800​, ​360​); glVertex2i(​795​, ​360​); glVertex2i(​795​, ​330​); glVertex2i(​802​, ​330​);

glEnd();

​// bench (part 10) glBegin(GL\_QUADS); glColor3ub(​153​, ​76​, ​0​); glVertex2i(​790​, ​360​); glVertex2i(​785​, ​360​); glVertex2i(​785​, ​340​); glVertex2i(​792​, ​340​); glEnd();

}

void​ ​drawWindmillOne​(​void​)

{

glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​); glVertex2i(​490​, ​910​); glVertex2i(​485​, ​910​); glVertex2i(​485​, ​830​); glVertex2i(​490​, ​830​);

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

for​ ​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​0​, ​153​, ​0​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* 2​ ​ \* pi) / ​200​;

​float r = ​ ​5.0​;

​float​ x = r \* ​cos​(A) + ​488​; ​float​ y = r \* ​sin​(A) + ​910​; glVertex2f(x, y);

} glEnd();

glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​);

glVertex2i(​488​, ​915​); glVertex2i(​483​, ​915​); glVertex2i(​450​, ​945​); glVertex2i(​450​, ​945​);

glEnd();

glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(​488​, ​915​); glVertex2i(​493​, ​915​); glVertex2i(​505​, ​950​); glVertex2i(​505​, ​950​);

glEnd();

glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(493​ , ​ 906​ )​ ; glVertex2i(​489​, ​906​); glVertex2i(​500​, ​866​); glVertex2i(​500​, ​866​);

glEnd();

} void​ ​drawWindmillTwo​(​void​)

{

glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​); glVertex2i(​575​, ​910​); glVertex2i(​580​, ​910​); glVertex2i(​580​, ​835​); glVertex2i(​575​, ​835​);

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​0​, ​153​, ​0​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

float​ A = (i \* ​ 2​ \* pi) / ​ 200​ ;​ ​float​ r = ​4.80​;

​float​ x = r \* ​cos​(A) + ​577​; ​float​ y = r \* ​sin​(A) + ​910​; glVertex2f(x, y);

}

glEnd();

glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​); glVertex2i(​579​, ​915​); glVertex2i(​583​, ​913​); glVertex2i(​597​, ​945​); glVertex2i(​597​, ​945​);

glEnd();

glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(​575​, ​914​); glVertex2i(​573​, ​908​); glVertex2i(​553​, ​950​); glVertex2i(​553​, ​950​);

glEnd();

glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(​576​, ​907​); glVertex2i(​572​, ​907​); glVertex2i(​565​, ​866​); glVertex2i(​565​, ​866​);

glEnd();

} void​ ​drawWindmillThree​(​void​)

{

glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​); glVertex2i(​650​, 910​ )​ ; glVertex2i(​655​, ​910​); glVertex2i(​655​, ​845​); glVertex2i(​650​, ​841​);

glEnd();

glBegin(GL\_POLYGON); ​// Draw a Red 1x1 Square centered at origin

​for​ (​int​ i = ​0​; i < ​200​; i++)

{

glColor3ub(​0​, ​153​, ​0​); ​//249,215,28 //255,140,0

​float​ pi = ​3.1416​;

​float​ A = (i \* ​2​ \* pi) / ​200​;

​float​ r = ​5.0​;

​float​ x = r \* ​cos(​ A) + 653​ ;​ ​float​ y = r \* ​sin​(A) + ​910​; glVertex2f(x, y);

} glEnd(); glBegin(GL\_QUADS);

glColor3ub(​0​,​0​, ​0​); glVertex2i(650​ , ​ 913​ )​ ; glVertex2i(​647​, ​908​); glVertex2i(​620​, ​940​); glVertex2i(​620​, ​940​); glEnd(); glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(​655​, ​914​); glVertex2i(​659​, ​911​); glVertex2i(​680​, ​950​); glVertex2i(​680​, ​950​); glEnd(); glBegin(GL\_QUADS); glColor3ub(​0​,​0​, ​0​); glVertex2i(​658, ​ 906​ )​ ; glVertex2i(​654​, ​906​); glVertex2i(​665​, ​866​); glVertex2i(​665​, ​866​); glEnd();

}

void​ ​drawPlane​(​void​)

{ glBegin(GL\_QUADS); glColor3ub(​150​,​150​,​150​); glVertex2i(​40​, ​965​); glVertex2i(​90​, ​970​); glVertex2i(​90​, ​955​); glVertex2i(​20​, ​955​); glEnd();

glBegin(GL\_TRIANGLES); glColor3ub(​150​,​150​,​150​); glVertex2i(​90​, ​970​); glVertex2i(​90​, ​955​); glVertex2i(​100​, ​960​); glEnd();

glBegin(GL\_TRIANGLES); glColor3ub(​0​,​0​,​200​); glVertex2i(​45​, ​959​); glVertex2i(​60​, ​959​); glVertex2i(​15​, ​935​);

glEnd();

glBegin(GL\_TRIANGLES); glColor3ub(​0​,​0​,​200​); glVertex2i(​70​, ​963​); glVertex2i(​55​, ​963​); glVertex2i(​45​, ​990​); glEnd();

glBegin(GL\_QUADS); ​// Draw a Red 1x1 Square centered at origin glColor3ub(​0​,​0​,​200​); glVertex2i(​20​, ​957​); glVertex2i(​30​, ​959​); glVertex2i(​25​, ​980​); glVertex2i(​20​, ​980​); glEnd();

}

// Variables int​ \_carOne = ​0​; int​ \_carTwo = ​0​; int​ \_carThree = ​0​; float​ \_boatOne = ​0.0​; float​ \_boatTwo = ​0.0​; float​ \_cloudTwo = ​0.0​; float \_cloudThree = ​ 0.0​ ;​ float​ \_plane = ​0.0​;

// Glut display function void​ ​myDisplay​(​void​)

{

glClear(GL\_COLOR\_BUFFER\_BIT); ​// clear the screen

drawSky(); drawSkyBottom(); drawRiver();

​// Boat 1 with animation glPushMatrix(); glTranslated(\_boatOne, ​0​, ​0​); drawBoatOne(); glPopMatrix();

​// Boat 2 with animation glPushMatrix(); glTranslated(\_boatTwo, ​0​, ​0​); drawBoatTwo(); glPopMatrix(); drawGrass(); drawLandShades(); drawRoads(); drawRoadSigns();

​//car 1 with animation glPushMatrix(); glTranslated(\_carOne, ​0​, ​0​); drawCarOne(); ​// Car One glPopMatrix(); ​//car 2 glPushMatrix(); glTranslated(\_carTwo, ​0​, ​0​); drawCarTwo(); glPopMatrix();

​// Car 3 with animation glPushMatrix(); glTranslated(\_carThree, ​0​, ​0​); drawCarThree(); ​// Car three glPopMatrix(); drawHomes(); drawTrees();

drawCloudOne();

​// cloud 2 with animation glPushMatrix(); glTranslated(\_cloudTwo, ​0, ​ ​0​); drawCloudTwo();

glPopMatrix();

​// cloud 3 with animation glPushMatrix(); glTranslated(\_cloudThree, 0​ ​, ​0​); drawCloudThree(); glPopMatrix();

​// plane with animation glPushMatrix(); glTranslated(\_plane, ​0​, ​0​); drawPlane(); glPopMatrix();

drawSun(); drawSignals(); drawBench(); drawWindmillOne(); drawWindmillTwo(); drawWindmillThree();

glutSwapBuffers();

}

void​ ​update​(​int​ value)

{

​// Car one: moving right

\_carOne += ​3.0​;

​if​ (\_carOne > ​250​)

{

\_carOne = ​-900​;

}

​// Car Two: moving right

\_carTwo += ​5.0​;

​if​ (\_carTwo > ​950​)

{

\_carTwo -= ​2000​;

}

​// Car one: moving left

\_carThree -= ​2.0​;

​if​ (\_carThree < ​-640​)

{

\_carThree = ​640​;

}

​// Boat one : Right

\_boatOne += ​0.80​;

​if​ (\_boatOne > ​1200​)

{

\_boatOne -= ​1300​;

}

​// Boat two : Left

\_boatTwo -= 0.50​ ;​

​if​ (\_boatTwo < ​-900​)

{

\_boatTwo = ​400​;

}

​// cloud 2 move : Right

\_cloudTwo += ​0.20​;

​if​ (\_cloudTwo > ​1200​)

{

\_cloudTwo -= ​1300​;

}

​// cloud 3 move : Right

\_cloudThree += ​0.20​;

​if​ (\_cloudThree > ​1200​)

{

\_cloudThree -= ​1300​;

}

​// plane move : Right

\_plane += ​0.70​;

​if​ (\_plane > ​1200​)

{

\_plane -= ​1300​;

}

glutPostRedisplay(); ​// Update the GLUT glutTimerFunc(​25​, update, ​0​); ​// Re-call it

} void​ ​initRendering​(​void​)

{

glClearColor(​0.0​, ​0.0​, ​0.0​, ​0.0​); ​// set the bg color to a

|  |
| --- |
| bright black  glMatrixMode(GL\_PROJECTION); ​// set up appropriate matrices-  glLoadIdentity(); ​// to be explained  gluOrtho2D(​0.0​, ​1050.0​, ​0.0​, ​1000.0​); glMatrixMode(GL\_MODELVIEW);  } int​ ​main​(​int​ iArgc, ​char​ \*\*cppArgv)  { glutInit(&iArgc, cppArgv); ​// initialize the toolkit  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH); ​// set the display mode  glutInitWindowSize(​640​, ​480​); ​// set window size  glutInitWindowPosition(​150​, ​150​); ​// set window upper left corner position on screen  glutCreateWindow(​"River Side Scenario"​); ​// open the screen window glutTimerFunc(​25​, update, 0​ ​); glutDisplayFunc(myDisplay);  initRendering(); ​// additional initializations as necessary glutMainLoop(); ​// go into a endless loop    ​return​ ​0​;  } |

# Conclusion

In this project, we want to show a clear riverside scenario. We make this view by using OpenGL. OpenGL is a software interface that allows a programmer to communicate with graphics hardware. Out of the many features available, the project demonstrates some popular and commonly used features of OpenGL, such as Rendering, Transformation, Rotation, Lighting, Scaling, etc. We use necessary library functions, user-defined functions, and loop here which is implemented by using the C++ language. We want to show the usual picture of a riverside small town on a sunny day. In this project, we design some houses, trees, clouds, roads, sun, boats, cars, planes, wind-mills, sky, river, etc. This project is well suited for designing 2D objects, as well as for carrying out basic graphic functionalities like drawing simple Quads, Polygons, Triangles. The project enabled me to work with mid-level OpenGL complexity.

In this project, cars and boats are moving. We use the necessary color to visualize this scenario properly. We use two types of color models. We use RGB and CMYK models to show the proper color.

## 1. Used header files

* **windows.h**:​ It handles the output window creation.
* **GL/glut.h**:​ This header file provides an interface with an application program that is installed on the system.
* **Math.h**:​ This contains constant definitions and external subordinate declarations for the math subroutine library.

## 2. Used functions (Built-in)

* **glClearColor( )**:​ Set the background color.
* **glMatrixMode(GL\_PROJECTION)**:​ Set up appropriate matrices. It specifies which matrix stack is the target for subsequent matrix operations.
* **glLoadIdentity( )**:​ It replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.
* **gluOrtho2D(left, right, top, bottom)**:​ It sets screen coordinates to be used in this system. It sets the upper and lower value of x-axis and y-axis.
* **glMatrixMode(GL\_MODELVIEW)**:​ Set up appropriate matrices.
* **glClear(GL\_COLOR\_BUFFER\_BIT)**:​ It clears the window screen.
* **glBegin(GL\_QUADS)**:​ It sets four vertices to create an independent quadrilateral.
* **glColor3ub( )**:​ It sets the color of this polygon body.
* **glVertex2i( )**:​ It sets the integer vertex value.
* **glEnd( )**:​ When we need to close a function part, then we will use this function.
* **glBegin(GL\_POLYGON)**:​ This function is used to draw a single, convex polygon. Vertices 1 through N define this polygon.
* **glBegin(GL\_TRIANGLES)**:​ It treats each triplet of vertices as an independent triangle. Normally, it sets three points to draw a triangle shape.
* **glFlush( ):** It guarantees that all OpenGL commands made up to that point will complete executions in a finite amount of time after calling glFlush().
* **main( ):** Execution of any program always starts with the main function irrespective of where it is written in a program.
* **glutInit(&iArgc, cppArgv):** It initializes the toolkit. It will initialize the glut library and negotiate a session with the window system.
* **glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB)**:​ The initial display mode is used when creating top level windows, sub windows to determine the OpenGL display mode. GLUT\_RGB specifies the structure of each pixel. It sets the display mode.
* **glutInitWindowSize( ):** It sets the window size.​
* **glutInitWindowPosition( ):** It sets the window upper left corner position on screen.
* **glutCreateWindow(“River-Side Scenario”):** It creates the screen window. When we run our code, then a screen window will open. This screen window’s name is River-Side Scenario.
* **glutDisplayFunc(myDisplay):** It sets the display callback for the current window.
* **glPushMatrix():** The glPushMatrix() saves the current screen to stack.
* **glPopMatrix():** The loads the data from the stack.​

## 3. Used functions (User-defined)

* void **myDisplay**​ (​ void): It is a user-defined function used for initializing display functions.
* void **update**​ (​ int value): It is a recursive function used for rendering animations like moving, scaling, rotating etc.
* void **initRendering**​ (​ void): It is a user-defined function used for clearing the bg color, setting ortho etc.
* Void **draw\_** ​ (​ void): This is a format of our functions which are used to draw different shapes of our project. Like: drawSun(), drawRiver() etc.