Computer Graphics LAB Project Report

ON

Village Scenario

SUBMITTED BY

Abdul Wahid
ID: **193-15-2992**

Ata-E-Elahi
ID: **193-15-2946**

MD RAKIBUL ISLAM
ID: **193-15-3015**

Samanta Sajjad Shraboni
ID: **193-15-3023**

AND
Abdulla Al Noman
ID: **193-15-2991**

Submitted To

Tanim Ahmed
Lecturer,
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

PREFACE

This report pertains to the scene titled " **Village scenario**” This is the final report of the Computer Graphics Lab's findings and recommendations (CSE-422). Computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer. It is also used for processing image data received from the physical world, such as photo and video content. Computer graphics development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, in general. Computer graphics also are essential to scientific visualization, a discipline that uses images and colors to model complex phenomena such as air currents and electric fields, and to computer-aided engineering and design, in which objects are drawn and analyzed in computer programs.

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty ALLAH for His divine blessing in making us possible to complete the project successfully. We really grateful and wish our best to Tanim Ahmed, Lecturer, Department of CSE, Daffodil International University, Dhaka.

INTRODUCTION

The main aim of this Mini Project is to illustrate the concepts of village scenario in OpenGL. In this project we have implemented a real-world scenario of a village so we have given the title as “A Village Scenario”. A village scenario is a project in computer graphics which is simple, amazing to look and useful. We have mainly created some artifacts in this mini project, like some houses, a sun, clouds, trees, and a road and a river and some boats on the river and also some hills. In the project some house’s is present in a small village and in the sky couple of clouds are there and there will be few trees beside the houses. There is also a big house beside a road and the road contains a flag of our country. There are some hills behind the houses and trees which are very overwhelming to watch. This scenario also contains a sun which is at the top. There is a river at the bottom side of the scenario which contains two moving boats. The big boat moves very fast and the small boat moves slower than the big one. This project not only concrete real-world objects but also build abstract, synthetic objects, such as mathematical surfaces of the objects. This is an overview of the project.

OBJECTIVES

The main objectives of this project are:

- To implement the features of graphics.
- To interface the applications of graphics to the real world.
- To give some benefits to the disability.
- To make the life easier.
- To become familiarization with Graphics and its logical coding.

FEATURES AND SCOPES

Computer Graphics has revolutionized almost every computer-based application in science and technology. Information technology is a trend today. As the volume of information increases, problem of storage arises. As time is money, in the 21st century people don’t have the time to read huge number of pages. In our project we are trying to represent a village picture with some animations. By using this project, we can a clear view a village look like without going into the village. Every part of the project has individual function name that can later easier to find. Though this project represents a 2D view of the village in future development we are going to implement it by 3D design.

METHODOLOGY

This project was done with the help of C++ Programming Language with the help of GLUT library function. Before going into a development process, we draw a simple picture with the help of pencil paper. After finalizing the drawing that paper pencil drawing import into adobe illustrator. In illustrator by using different shapes, we created an only line picture in illustrator. By completing the shape of the project later use color to give a more realistic look in the project. The illustrator image exported and placed in the GeoGebra calculator suite application. In GeoGebra we find the coordinates of the shapes that can later implemented in the code with the help of GULT library function. The codes are written in C++ language. The process of our work is given in Figure 1.

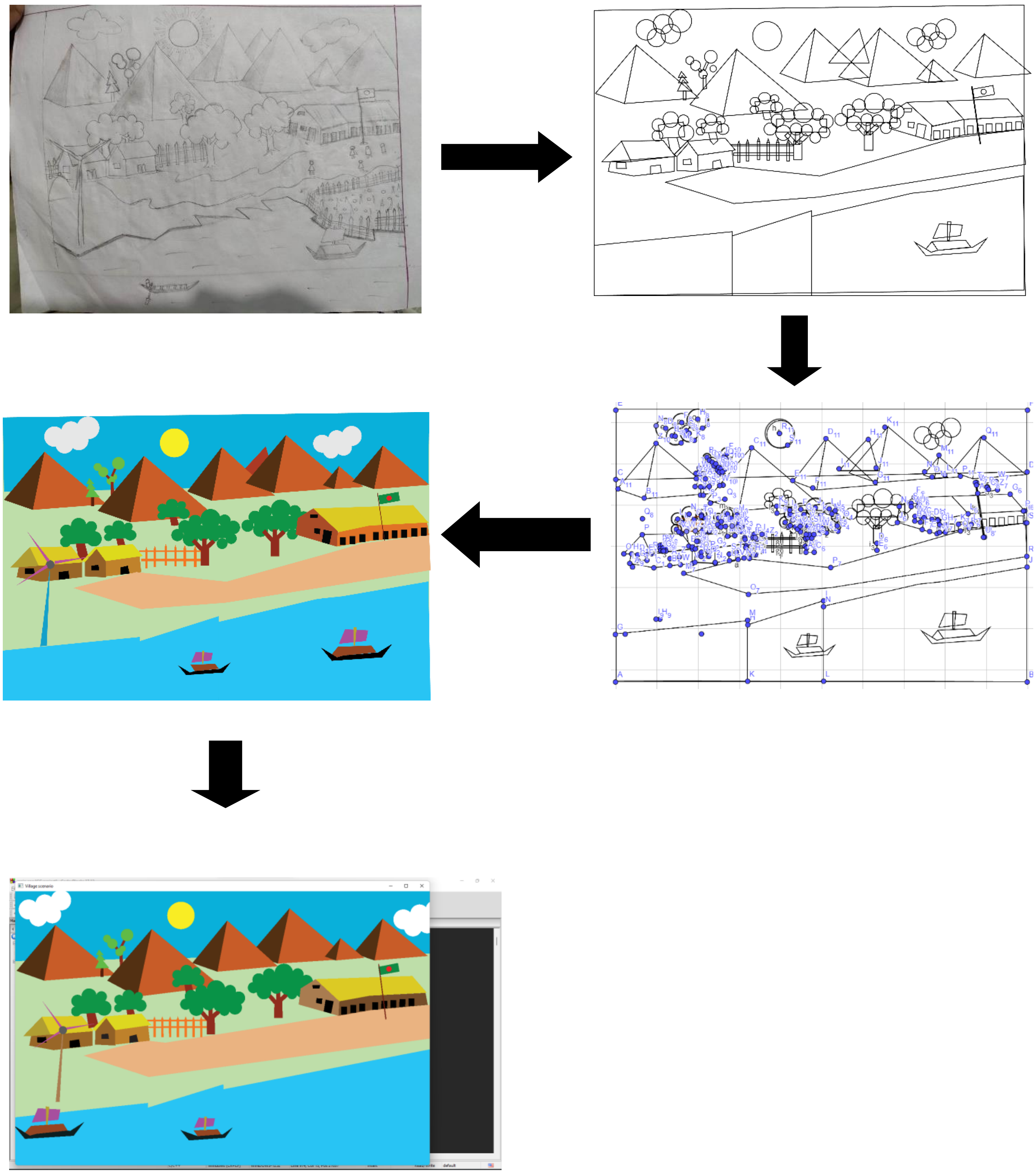


Figure 1. Process of development.

IMPLEMENTATION

1. Header Files

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. We use GLUT, GLU, GL, STDLIB, IOSTREAM.

<GL/glut.h>

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, that primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input.

<stdlib.h>

stdlib.h is the header of the general-purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions, and others. It is compatible with C++ and is known as cstdlib in C++. “<stdlib.h>” is a header file for Standard Library. “<stdlib.h>” contains header information for ‘Memory Allocation/Freeing’ functions.

<iostream>

Iostream provides us with various functions to handle the input and output stream in c++. This iostream header file contains various functions, including cin, cout, cin, and many more. With the help of this, we can read the input, print them, and also trace the error, if any.

2. OpenGL functions

OpenGL is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering. Some of the functions that are use in this project are given below:

glutInit: 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.

glutInitWindowPosition, glutInitWindowSize: Windows created by glutCreateWindow will be requested to be created with the current initial window position and size.

glutInitDisplayMode: The initial display mode is used when creating top-level windows, subwindows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay. GLUT_RGBA selects the RGBA color model, but it does not request any bits of alpha.

- GLUT_RGBA: Bit mask to select an RGBA mode window.
- GLUT_SINGLE: Bit mask to select a single buffered window. This is the default if neither GLUT_DOUBLE or GLUT_SINGLE is specified.
- GLUT_DOUBLE: Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.
- GLUT_DEPTH: Bit mask to select a window with a depth buffer.

glutCreateWindow: Creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

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.

glutDisplayFunc: Sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

glClearColor: glClearColor specifies the red, green, blue, and alpha values used by glClear to clear the color buffers. Values specified by glClearColor are clamped to the range 0 1 .

glutMainLoop: 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 return. It will call as necessary any callbacks that have been registered.

glBegin: glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted. It has multiple parameters GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, and GL_POLYGON.

glVertex: glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called.

glTranslate: glTranslate produces a translation by x y z . The current matrix is multiplied by this translation matrix, with the product replacing the current matrix, as if glMultMatrix were called with the following matrix for its argument.

glutSolidSphere: Renders a sphere centered at the modeling coordinates origin of the specified radius. The sphere is subdivided around the Z axis into slices and along the Z axis into stacks. Parameters are radius, slice, stacks.

Main function

```
int main(int argc, char *argv[])

{

    glutInit(&argc, argv);

    glutInitWindowSize(1212.08,800);

    glutInitWindowPosition(10,10);

    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);



    glutCreateWindow("Village scenario");

    //glutReshapeFunc(resize);

    gluOrtho2D(-250,250,-165,165);

    glClearColor(1,1,1,0);

    glutDisplayFunc(display);

    glutTimerFunc(1000,timer,0);



    glutMainLoop();



    return EXIT_SUCCESS;

}
```

3. User define functions

In this project there are multiple shapes like circle, triangles, polygon. By using these shapes, we are draw different objects in our projects.

To identify these objects and maintain them easily we build custom functions which are house, tree, sky, fence, cloud, sun, boat, flag, school, mountain. For creating circle easily a circle functions have been declared which takes parameters as radius, center point x, center point y.

Circle function

```
void circle(float radius,float x=0,float y=0)

{

    glPushMatrix();

    glTranslated(x,y,0);

    glScaled(1,1,0);

    glutSolidSphere(radius,100,100);

    glPopMatrix();

}
```

Animation function

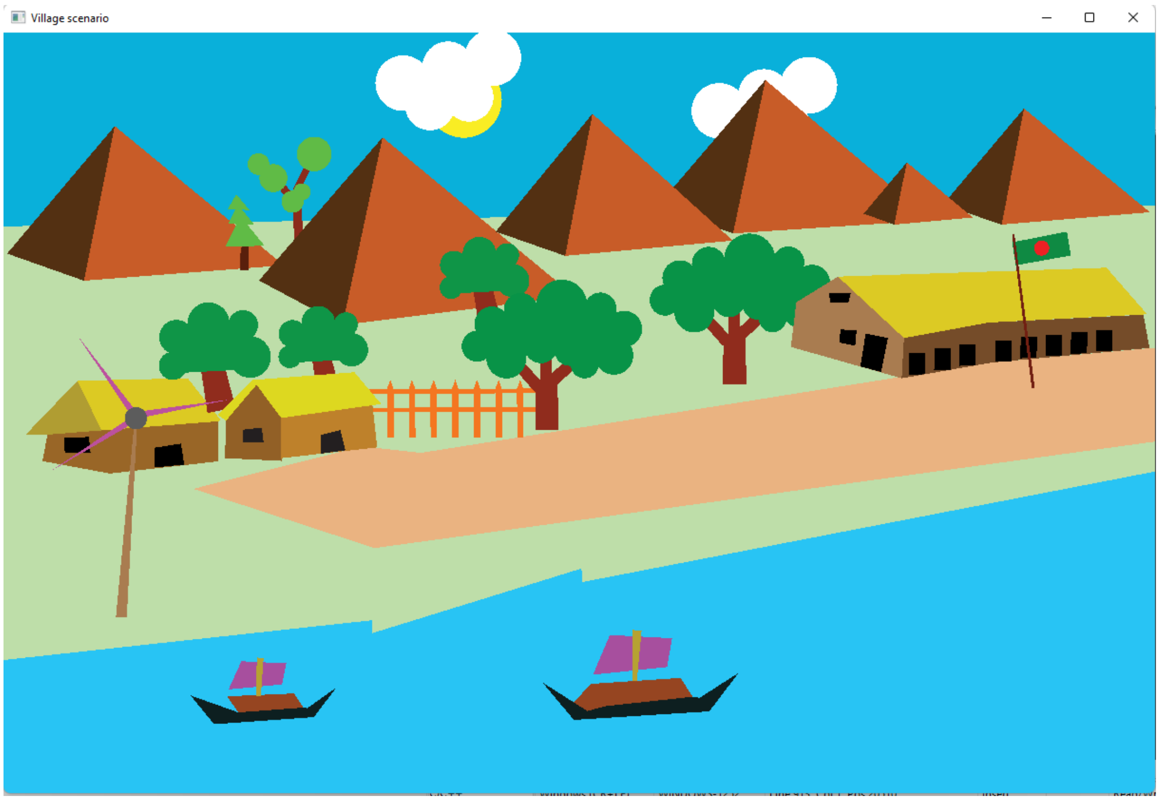
```
void timer(int)

{



    glutPostRedisplay();



    if(boat1x>-300)

        boat1x-=0.15;

    else

    {

        boat1x=300;

    }



    if(boat2x<150)

        boat2x+=1;
    else

    {

        boat2x=-400;

    }



    if(cloud1x<500)

        cloud1x+=0.2;

    else

    {

        cloud1x=-50;

    }

    if(cloud2x>-450)

        cloud2x-=0.2;

    else

    {

        cloud2x=-50;

    }

    glutTimerFunc(1000/60,timer,0);

}

void tree6()

{

    glPushMatrix();

    glColor3ub(96,187,70);

    glBegin(GL_POLYGON);

    glVertex2f(-148.9325, 94.89213);

    glVertex2f(-152.68311, 88.35881);

    glVertex2f(-143.24609, 87.99585);

    glEnd();
    glBegin(GL_POLYGON);

    glVertex2f(-148.9325, 94.89213);

    glVertex2f(-152.68311, 88.35881);

    glVertex2f(-143.24609, 87.99585);

    glEnd();


    glBegin(GL_POLYGON);

    glVertex2f(-148.20657, 90.4156);

    glVertex2f(-152.44114, 81.94648);

    glVertex2f(-141.1893, 81.7045);

    glEnd();


    glBegin(GL_POLYGON);

    glVertex2f(-147.0219, 85.16512);

    glVertex2f(-153.71821, 72.2508);

    glVertex2f(-136.59479, 72.44213);

    glEnd();


    glColor3ub(89,31,12);

    glBegin(GL_POLYGON);

    glVertex2f(-147.0219, 71.96382);

    glVertex2f(-143.48242, 72.05948);

    glVertex2f(-143.67375, 61.72803);

    glVertex2f(-147.21322, 61.91935);

    glEnd();


    glPopMatrix();


}
```

SOURCE CODE

GitHub repository: https://github.com/abdul-wahid789/Village-scenario

GitHub Repository QR code.

PROJECT OUTPUT



CONCLUSION

We have put in place a “A Village Scenario” which is a colorful and simple mini project. And this project includes a lot of options in it. We have implemented the idea of adding simple objects as we have used in this project and if we add some more objects like people and other stuff then it will look even better. For the future implementations we will try to create a school and some vehicles moving in the road and some animals.

CONTRIBUTIONS

1. Samanta Sajjad Shraboni (193-15-3023):
   Sketch the initial scenario of the project by using paper pencil. Create the tree5, tree6, boat1 and animate boat1.

2. Abdul Wahid (193-15-2992):
   Paper pencil drawing illustrate in Adobe illustrator for better representation of the project. After that This illustration placed in GeoGebra calculator suite for coordinate of the objects. Create house2, schoolhouse, fence, river, boat2 and animate boat2.

3. Ata-E-Elahi (193-15-2946):
   Finding the coordinate of tree1, tree2, tree3, tree4 and implement them in code. Also finding the coordinate and create road and flag

4. MD RAKIBUL ISLAM (193-15-3015):
   Finding the coordinate and create the clouds, mountains and animate the clouds.

5. Abdulla Al Noman (193-15-2991):
   Finding the coordinate and create the house1, sun, windmill.