Bangladesh University of Business and Technology (BUBT)


# Report on Graphical Village (#ville)

CSE 342- Computer Graphics

## Supervised By

**Jubayer Al Mahmud**

Assistant Professor

Dept. of CSE, BUBT

## Submitted by

**Ahmed Mahir Shoaib**
18192103235
Intake: 41
Section: 6
Dept. of CSE, BUBT

**Emran Hossain Khan**
18192103231
Intake: 41
Section: 6
Dept. of CSE, BUBT

**Khandoker Maliha Fairuz**
17193103002
Intake: 39
Section: 1
Dept. of CSE, BUBT

# ACKNOWLEDGEMENTS

We take this occasion to thank God, almighty for blessing us with His grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed project adviser Jubayer Al Mahmud, Assistant Professor, Department of CSE, BUBT for his invaluable guidance during the course of this project work. We extend my sincere thanks to him for his continuously helped throughout the project and without his guidance, this project would have been an uphill task.

Last but not the least, we would like to thank friends for the support and encouragement they have given us during the course of our work.

Ahmed Mahir Shoaib

Emran Hossain Khan

Khandoker Maliha Fairuz

# Abstract

Computer graphics is an effective medium for presentation and design. In early days of its usage, it has been used specially for presentation. Then it turned into started to use computer graphics in design development stage. Even more, in recent times you could get a concept from it. With tracing the change, we can see a centripetal movement of usage from the perimeter to the core of the design field. This paper will describe how this change occurred with what kind of effort.

# DEDICATION

We would like to dedicate this software to our parents, teachers and freedom fighters.

# Index

# Chapter 1.1

## Introduction:

OpenGL (Open Graphics Library) is a software interface to graphics hardware. The interface consists of over 250 different function calls which can be used to draw complex two and three-dimensional scenes from simple geometric primitives such as points, lines, and polygons. There are also routines for rendering the scenes with control over lighting, object surface properties, transparency, anti-aliasing and texture mapping. OpenGL was developed by Silicon Graphics Inc. in 1992 to be a more ‘open’ (portable) version of their early SGI GL Graphics Library. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different graphics hardware platforms.

Routines in the OpenGL graphics library can be called from most major languages. Window management is not supported in the basic version of OpenGL but there are additional libraries built on top of OpenGL which you can use for many of the things which OpenGL itself does not support. For using OpenGL, we can use many languages. Like: C, C++, Objective-C, Python etc.

With OpenGL, we can do graphics work. In this case, we have chosen a village simulation type project. This project gives us a nice and clear idea about computer graphics.

# Chapter 2.1

## Requirement Analysis:

### Software Requirements

**Operating system:** Windows 7 or up, Ubuntu 16.04, MacOS Mojave or up.

**Editor:** Visual Studio, Codeblocks etc.

**Language:** C, C++, Python, Objective – C etc.

### Hardware requirements

**Processor:** AMD, Intel. Above x86.

**Processor speed:** 500 MHz and above

**RAM:** 64 MB or above storage space 4GB and more

**Monitor Resolution:** A color monitor with a minimum resolution of 640*480

## Platform

The package is implemented using Microsoft visual C++ under the windows programming environment. OpenGL and associated toolkits are used for the package development.

# Chapter 3.1

## Design:

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

•GL/glut.h: This header files provides an interface with application programs that are installed on the system.

•stdio.h: Input & Output operations can also be performed in C using C standard input, output library.

•math.h: contains constant definitions and external subordinate declarations for the math subroutine library.

•Windows.h: handles the output window creation.

Description of OpenGL functions Glut library functions used are:

main(): Execution of any program always starts with main function irrespective of where it is written in a program.

glutSwapBuffers(void): Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

GLUT_RGB specifies the structure of each pixel. GLUT_DEPTH is a buffer to hold the depth of each pixel.

glutCreateWindow(char *title): The 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.

glutReshapeFunc():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.

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.

glutBitmapCharacter(font,c): Without using any display lists, glutBitmapCharacter renders the character ‘c’ in the named bitmap font.

glPointSize(size): glPointSize specifies the rasterized diameter of points.

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.

glRasterPos3f(x , y, z):This function is used to set the cursor at the x, y, z location. glBegin(enum) and glEnd(): 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 often ways the vertices are interpreted.

glColor3f(R,G,B): This function is used to pick a color of specified R,G,B value.

glClear(GLbitfield): glClear sets the bit plane area of the window to values previously selected by glClearColor, glClearIndex, glClearDepth, glClearStencil, and glClearAccum. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time using glDrawBuffer.

glVertex2f(x,y): This function is used to draw a vertex at location x,y.

glEnable(GLenum): This function is used to enable the various functionalities like enabling the light, enabling the Z buffer.

glDisable(GLenum): This function is used to disable the various functionalities like enabling the light, enabling the Z buffer.

glRectf(x0,y0,x1,y1): This function is used to draw rectangle using the two diagonal points.

glClearColor(R,G,B,A):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 to1 .

glMaterialfv(GLenum,GLenum,GLfloat *): glMaterial assigns values to material parameters. There are two matched sets of material parameters. One, the front-facing set, is used to shade points, lines, bitmaps, and all polygons (when two-sided lighting is disabled), or just front-facing polygons (when two-sided lighting is enabled).

glutSolidTeapot(size): glutSolidTeapot and glutWireTeapot render a solid or wireframe teapot respectively. Both surface normals and texture coordinates for the teapot are generated. The teapot is generated with OpenGL evaluators

glutSolidCube(size): glutSolidCube and glutWireCube render a solid or wireframe cube respectively. The cube is centered at the modeling coordinate's origin with sides of length size.

glutDisplayFunc(void (*func)()): 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 (if no overlay display callback is registered) the

layer in use is set to the normal plane. The display callback is called with no parameters.

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

glutMouseFunc(void (*func)(int button, int state, int x, int y)): glutMouseFunc sets the mouse callback for the current window. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The

button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON.

glutMotionFunc(void (*func)(int x, int y)): glutMotionFunc set the motion callback for the current window. The motion callback for a window is called when the mouse moves within the window while one or more mouse buttons are pressed.

glutPassiveMotionFunc(void(*func)(int x, int y)): glutPassiveMotionFunc set the passive motion callback for the current window. The passive motion callback for a window is called when the mouse moves within the window while no mouse buttons are pressed.

glMatrixMode(GLenum) mode: Specifies which matrix stack is the target for subsequent matrix operations. Values accepted are: GL_MODELVIEW, GL_PROJECTION.The initial value is GL_MODELVIEW. Additionally, if the ARB tension extension is supported, GL_COLOR is also accepted.

gluPerspective(fovy, aspect, near, far): gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect = 2.0means the viewer's angle of view is twice as wide in x as it is in y.

glOrtho(left, right, top, bottom, Znear, Zfar): glOrtho describes a transformation that produces a parallel projection. The current matrix is multiplied by this matrix and the result replaces the current matrix.

Description of User-defined functions

Various user-defined functions used are:

void drawstring(float x, float y, float z, char *string): Used to display text.

void drawpoint(int x, int y): Used to draw point.

void paint(int x, int y): Used for PAINT BRUSH function.

void eraser(int x, int y): Used for ERASER function.

void reset(): It resets the variables in which vertices are stored, after a polygon is drawn.

void draw_pixel(GLfloat x, GLfloat y): It's used to draw the points of a circle.

void draw_circle(GLfloat p, GLfloat q, GLfloat r): It's used to draw a CIRCLE using MIDPOINT CIRCLE DRAWING algorithm

void draw_circle1(GLfloat p, GLfloat q, GLfloat r): It's used to draw the circle option on the tool bar.

void edgedetect(float x1, float y1, float x2, float y2, int *le, int *re): It's used to detect edges of the polygon to be filled.

void scanfill(GLint num1, GLint num2): It's used to FILL a Polygon using SCAN LINE ALGORITHM.

void detect_point(GLint num1, GLint num2, int x, int y): It's used to detect which POLYGON TO BE FILLED.

void display(void): It's user-defined DISPLAY function.

void keys(unsigned char key, int x, int y): It's used to get FILENAME from the KEYBOARD.

void divide_trinagle(GLfloat *a, GLfloat *b, GLfloat *c, int k): It's used for drawing the 2D Sierpinski gasket.

void tetra(GLfloat *a, GLfloat * b, GLfloat * c, GLfloat * d): It's used in drawing 3D Sierpinsk i gasket.

void divide_tetra(GLfloat * a, GLfloat * b, GLfloat * c, GLfloat * d, int m): It's used in drawing 3D Sierpinski gasket.

void myMouse(int btn, int state, int x, int y): It's used to handle all the mouse events.
void myReshape(GLsizei w, GLsizei h): It's user defined RESHAPE FUNCTION.
void point_size(int id): Used to change the pixel size.

# Chapter 4.1

## Screen Shots:

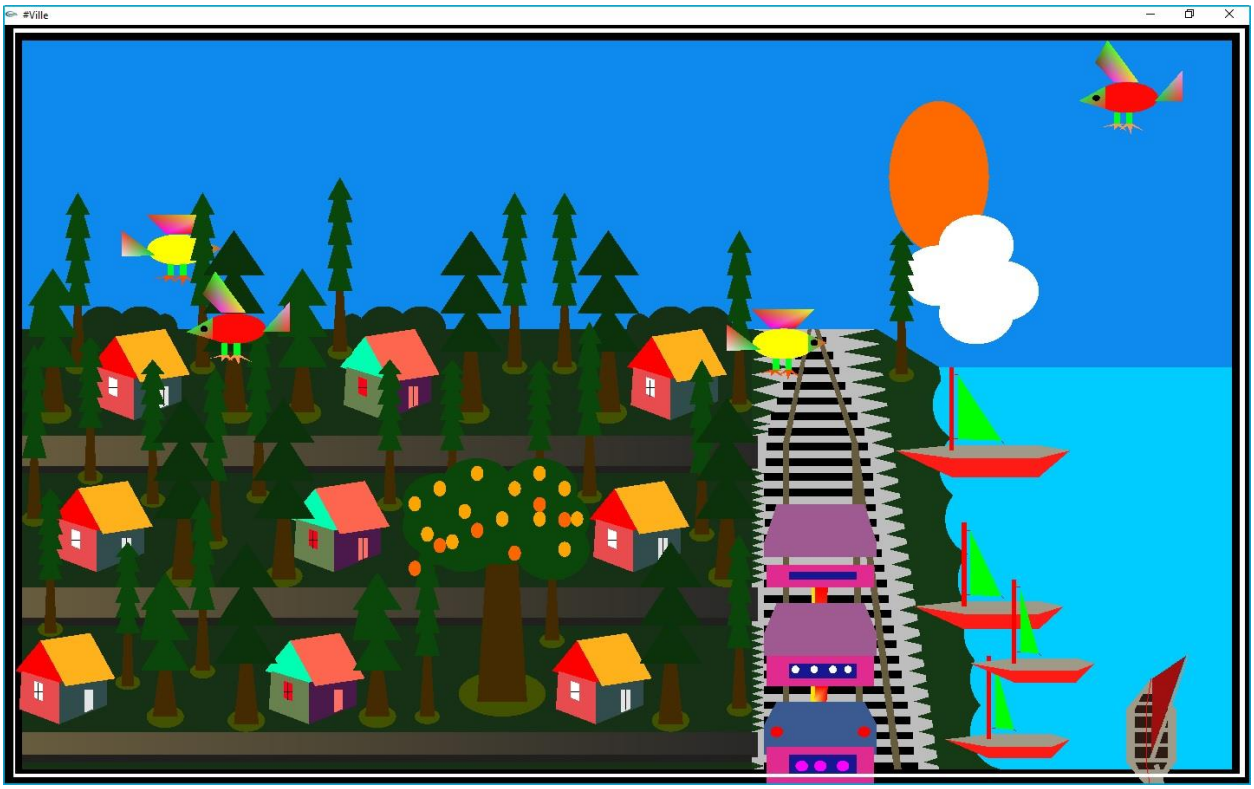


**Fig: Day Mode**

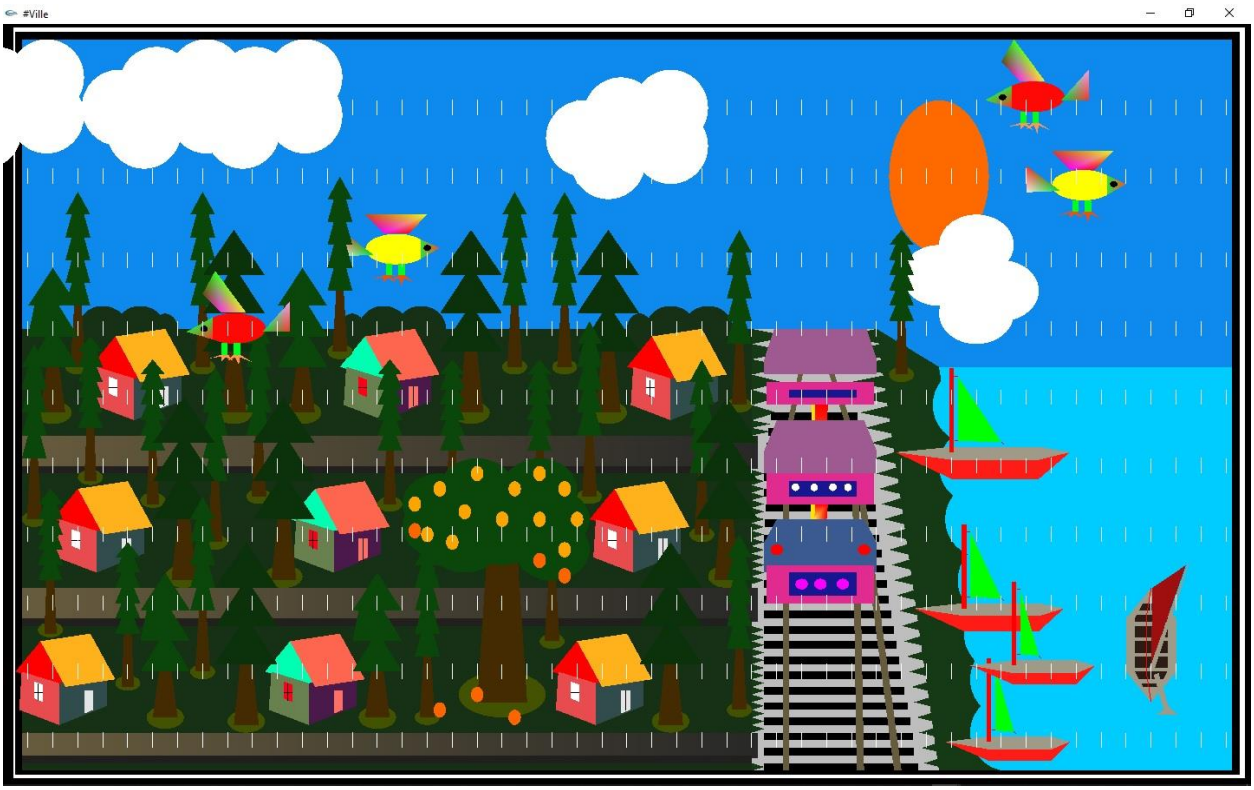


**Fig: Day Rain Mode**

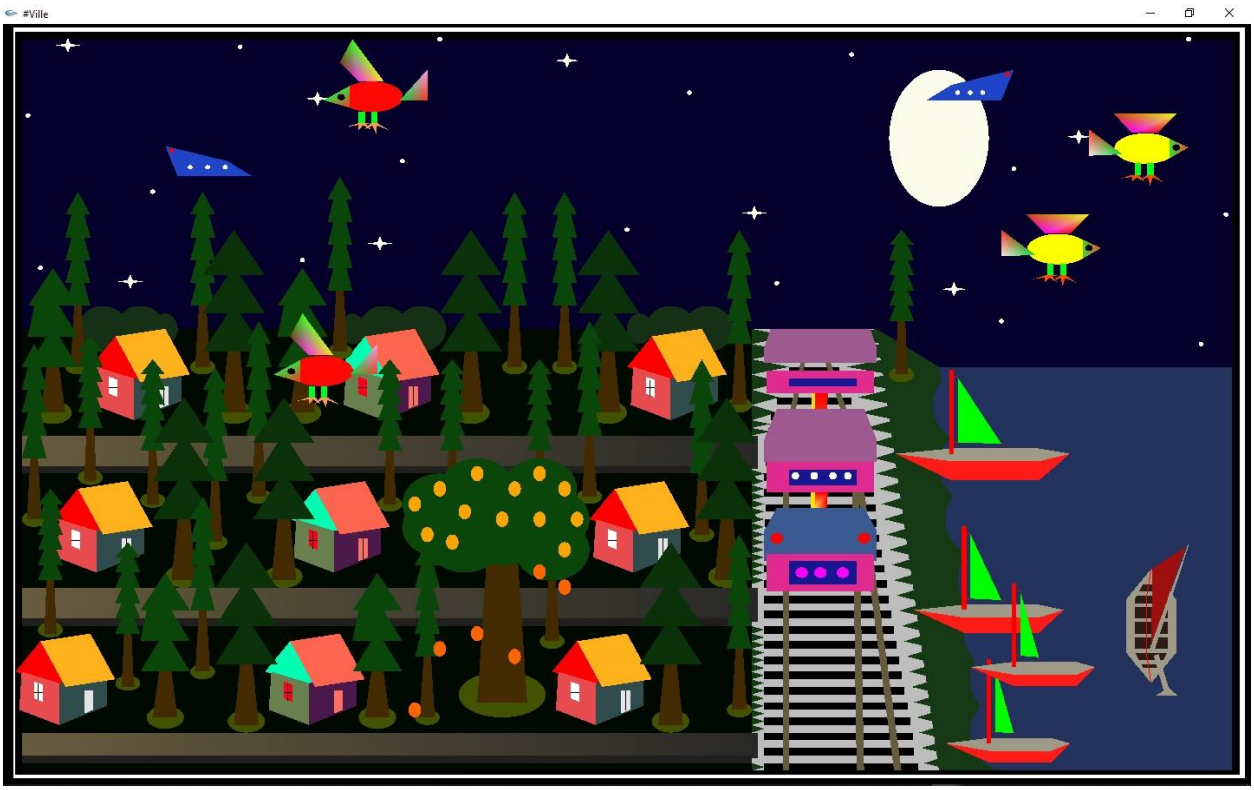


**Fig: Night Mode**

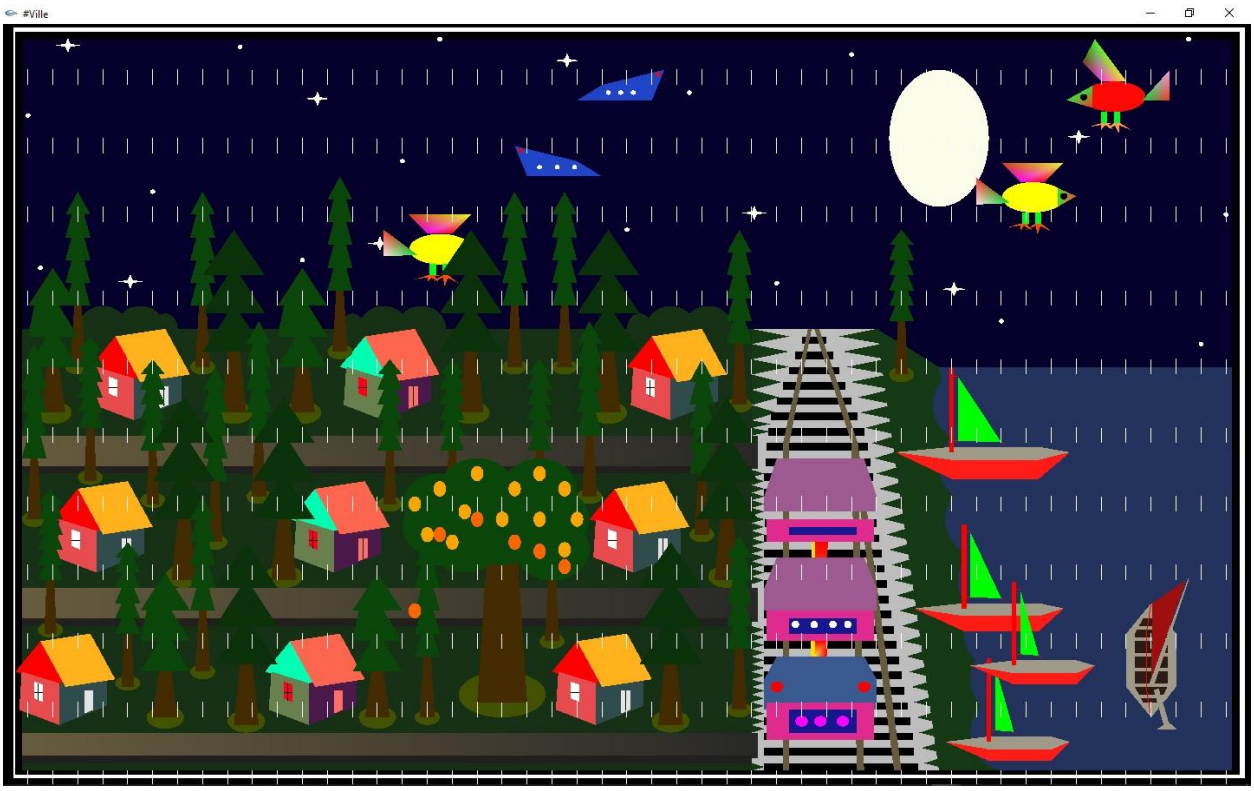


**Fig: Night Rain Mode**

# Chapter 6

## Conclusion:

The task is nicely suited for designing 2d and 3D items, in addition to carrying out basic graphics functionalities like drawing an easy line, creating a cube, circle, square, erasing and filling them. However, if carried out on a huge scale with enough assets, it has the potential to come to be a general stand-alone GUI based utility for the Windows operating system.

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. These graphic functions will also work in tandem with various rules involved in the game. Since this project works on dynamic values, it can be used for real-time computation.

The project enabled me to work with the mid-level. OpenGL complexity and the project demonstrates the scope of the OpenGL platform as a premier game developing launch pad. Hence, it has indeed been useful in developing many games. OpenGL in its own right is good for low cost and simple game development. It serves as an important stepping stone for venturing into other fields of Computer Graphics design and applications.