**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**
**BELAGAVI**

***Mini Project Report on***

# "TOWN SIMULATION"

*Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work*

*Submitted By*

| | |
|---|---|
| **S.BHANU KIRAN** | USN: 1BY18CS128 |
| **SAI RITWIK REDDY.N** | USN: 1BY18CS095 |
| **RAVI KUMAR.M** | USN: 1BY18CS085 |

*Under the guidance of*

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

YELAHANKA, BENGALURU - 560064.

2020-21

**BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT**
**YELAHANKA, BENGALURU – 560064**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# CERTIFICATE

This is to certify that the Project work entitled **"TOWN SIMULATION"** is a bonafide work carried out by **S.BHANU KIRAN (1BY18CS128) and SAI RITWIK REDDY (1BY18CS095) and RAVI KUMAR (1BY18CS085)** in partial fulfillment for *Mini Project* during the year 2021. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*.It is also certified that all corrections/suggestions indicated for Internal Assessment have beenincorporated in this report.

**Signature of the Guide with date**
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

**Signature of HOD with date**
Dr.Bhuvaneshwari C M
Prof & Head
CSE,BMSIT&M

## INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

## INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

## DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

## DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

## PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

## ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. Bhuvaneshwari C M, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

S.BHANU KIRAN(1BY18CS128)

SAI RITWIK REDDY.N (1BY18CS095)

RAVI KUMAR.M(1BY18CS085)

# ABSTRACT

The main aim of the ***TOWN SIMULATION*** Computer Graphics Mini Project is to illustrate the concepts and usage of pre- built functions in OpenGL. Simulation of "**TOWN VIEW**" is being done using computer graphics. The development of the simulation has large scope to learn computer graphics and visualization from scratch. We will be using OpenGL utility toolkit to implement the algorithm, written in C language.

A Town is a human settlement. The main aim of the project is to design a town using computer graphics. The main idea being to attract the young tech-generation kids to understand the Simulation of town using computer graphics. Also an effective way of showing the view of town through the great visual experience. In this view we can move the vehicles using arrow keys across the frames. So, in this project we have simulated the town view using OpenGL functions. The user can see the view of the town and simulate the moving the vehicle using keyboard functionalities. The moving effect is conveyed properly using various concepts of OpenGL to traverse through the different frames.

We have tried to implement the project making it as user-friendly and error free as possible. We regret any errors that may have inadvertently crept in.

# TABLE OF CONTENTS

CHAPTER 1

# INTRODUCTION

## 1.1 COMPUTER GRAPHICS

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer hardware and software. The development of computer graphics, or simply referred to as CG, has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry. 2D computer graphics are digital images—mostly from two-dimensional models, such as 2D geometric models, text (vector array), and 2D data. 3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering images.

The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch- screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantagesof the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became animportant field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

## 1.2 OPEN GL

OpenGL is the most extensively documented 3D graphics API (Application Program Interface) to date. Information regarding OpenGL is all over the Web and in print. It is impossible to exhaustively list all sources of OpenGL information. OpenGL programs are typically written in C and C++. One can also program OpenGL from Delphi (a Pascal-like language), Basic, Fortran, Ada, and other languages. To compile and link OpenGL programs, one will need OpenGL header files. To run OpenGL programs one may need shared or dynamically loaded OpenGL libraries, or a vendor-specific OpenGL Installable Client Driver (ICD).

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).

## 1.3 GLUT

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which 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. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and cylinders. GLUT even has some limited support for creating pop-up menus. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross- platform) and to make learning OpenGL easier. All GLUT functions start with the glut prefix (for example, glutPostRedisplay marks the current window as needing to be redrawn).

**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 islargely 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 rectangularscreen 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 designs, 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 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.

## 2.2 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:

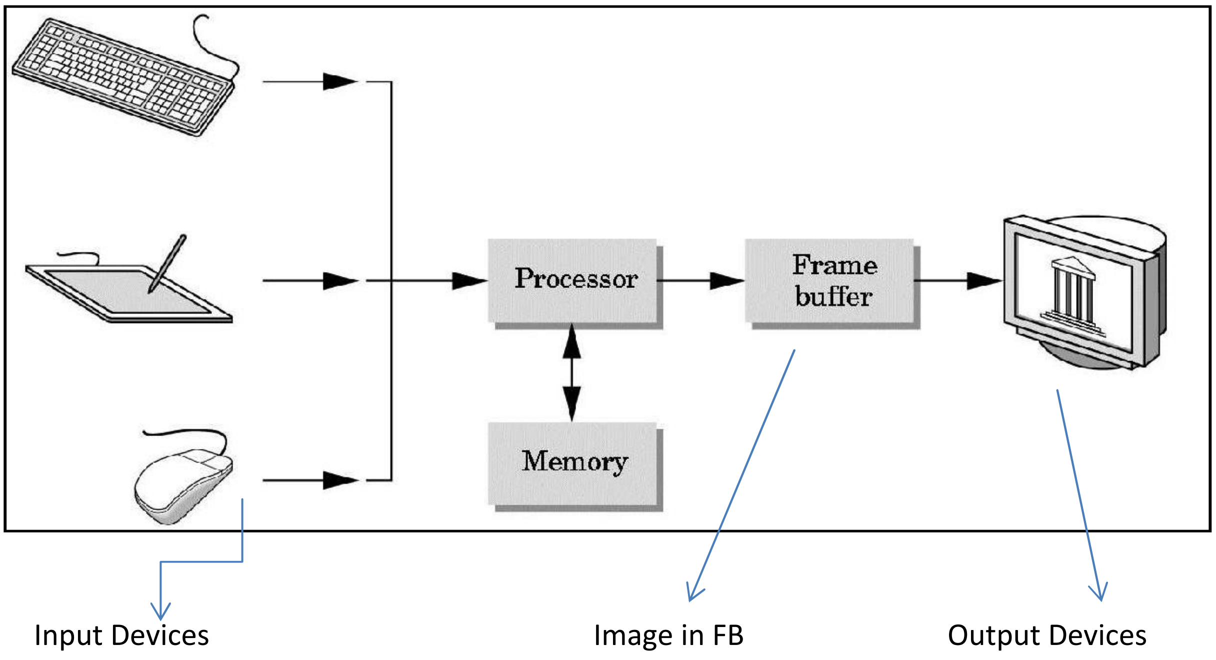


**FIGURE 2.2** BASIC GRAPHICS SYSTEM

CHAPTER 3

# PROJECT IN DETAIL

## 3.1 PROBLEM STATEMENT

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D, 3D and higher dimensional objects. Creating 3D objects, rotations and any other manipulations are laborious process with graphics implementation using text editor. OpenGL provides more features fordeveloping 3D objects with few lines by built in functions. The geometric objects are the building blocks of any individual .Thereby developing, manipulating, applying any transformation, rotation, scaling on them is the major task of any image development.

So in this project the main problem statement is to construct a town with help of computer graphics using Open GL.

## 3.2 BRIEF INTRODUCTION

In this Project, we are Simulating the view of town. The animation of the town view is developed using OpenGL Library in C.

A Town is a human settlement. The main aim of the project is to design a town using computer graphics. The main idea being to attract the young tech-generation kids to understand the Simulation of town using computer graphics. Also an effective way of showing the view of town through the great visual experience. In this view we can move the vehicles using arrow keys across the frames. So, in this project we have simulated the town view using OpenGL functions. The user can see the view of the town and simulate the moving the vehicle using keyboard functionalities. The moving effect is conveyed properly using various concepts of OpenGL to traverse through the different frames.

## 3.3 SCOPE AND MOTIVATION

Simulation is a way of creating a imaginary image with certain kind of animations in it. The main motivation of the project is to create a town simulation using computer graphics. The scope of the project is creating a town simulation using opengl graphic library.

**"The way you live your life here is just awesome."**

In this project we are trying to develop an easy, efficient and cost free method of simulation.

## 3.4 PROPOSED SYSTEM

We need something which is as interesting as a TV graphics and at the same timefilled with good stuffs.

So we are adding a good visual and making the story a video simulation which will create memories in the child's brain and he/she will remember it. Because visual is always more interesting and easier to remember.

With the use of different functionality like of making polygons and keyboard functions the whole simulation is plotted on the output screen. This project gives a user-friendly interface for the end user to interact with the frames. This project contains several different frames that are used in a particular sequence to help the user understand the simulation.

## 3.5 USER INTERFACE

The legend for user interaction is as follows:

| | | |
|---|---|---|
| ■ | **"Esc"** | To quit. |
| ■ | **"Arrow keys"** | To move the man along the road. |
| ■ | **"b"** | To stop the bus. |
| ■ | **"m"** | To move the man from the bus. |
| ■ | **"S"** | To move the bus. |

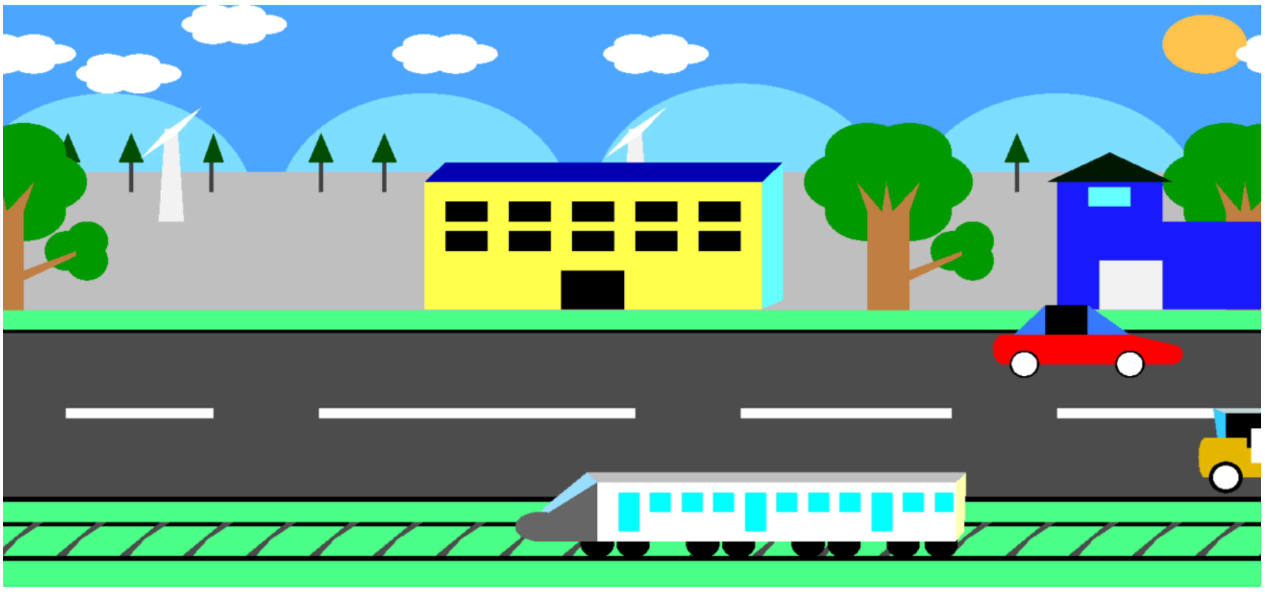


**FIGURE 3.5.1**: Town Simulation (A scenario)

CHAPTER 4

# REQUIREMENT SPECIFICATION

## 4.1 HARDWARE REQUIREMENTS

- Processor: INTEL / AMD
- Main memory: 2 GB RAM (Min.)
- Hard Disk: Built-in is sufficient
- Keyboard: QWERTY Keyboard
- Mouse: 2 or 3 Button mouse
- Monitor: 1024 x 768 display resolution

## 4.2 SOFTWARE REQUIREMENTS

- Programming language – C/C++ using OpenGL
- Operating system – Windows/Linux
- Compiler – C/C++ Compiler (GCC compier)
- IDE – Code blocks
- Functional Requirement – <GL/glut.h>

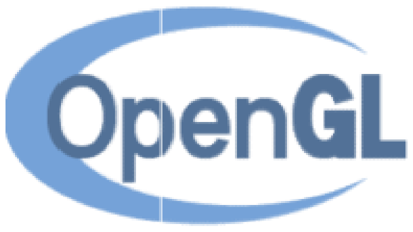


**FIGURE 4.2.1:** OpenGL API

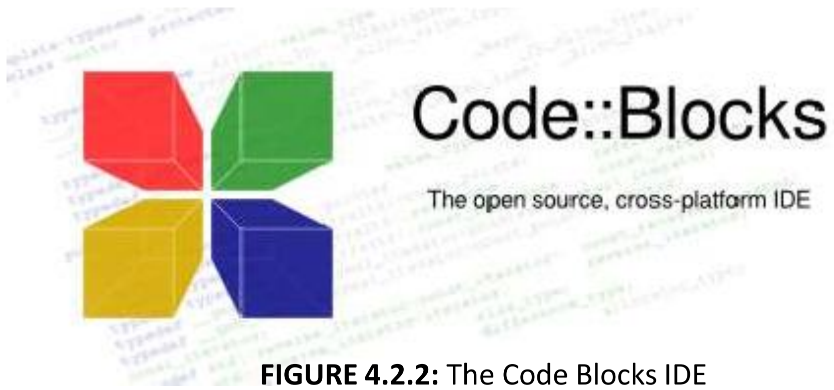


**FIGURE 4.2.2:** The Code Blocks IDE

# CHAPTER 5

# DESIGN AND IMPLEMENTATION

## 5.1 DESIGN

The “Town Simulation” is designed using some of the OpenGL inbuilt functions along with some user defined functions. So, this section comprises of the complete explanation of the design of the project using various opengl functions and user defined methods along with its explanation. This chapter is divided into two sections –

- **OPENGL Functions** – This section throws light on the various openGL functions used and its uses.
- **USER – DEFINED Functions** – This section demonstrates the various user defined functions and its purpose.

### 5.1.1 OPEN GL FUNCTIONS:

The different OpenGL functions that are used in the project is described below:

- **glClearColor(0.1,0.8,0.1,1.0)** :- glClearColor() specifies the red, green, blue, and alpha values used by glClear() to clear the color buffers. Values specifiedby glClearColor() are clamped to the range 0, 1
- **glMatrixMode(GL_MODELVIEW)** :- specify which matrix is the current matrix. Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE. The initial value is GL_MODELVIEW.
- **gluOrtho2D(0,1000,0,500)** :- define a 2D orthographic projection matrix. *left*, *right*- Specify the coordinates for the left and right vertical clipping planes.

*bottom*, *top* - Specify the coordinates for the bottom and top horizontal clipping planes.

- **glRasterPos2f(x, y)** :- The glRasterPos2 function uses the argument values for **x** and **y** while implicitly setting z and w to zero and one. The object coordinates presented by glRasterPos are treated just like those of a glVertex command. They are transformed by the current model view and projection matrices and passed to the clipping stage.
- **glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i])** :- glutBitmapCharacter renders a bitmap character using OpenGL. Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.
- **glutFullScreen()** :- glutFullScreen requests that the *current window* be made full screen. The exact semantics of what full screen means may vary by window system. The intent is to make the window as large as possible and disable any window decorations or borders added the window system.
- **glFlush()** :- Force execution of GL commands in finite time. 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.
- **glPointSize(4)** :- specify the diameter of rasterized points. glPointSize specifies the rasterized diameter of points. The value written to the shading language built-in variable gl_PointSize will be used.
- **glBegin(GL_POINTS)** :- delimit the vertices of a primitive or a group of like primitives. 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.
- **glVertex2i(278,273)** :- specify a vertex. 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.

- **glEnd()** :- delimit the vertices of a primitive or a group of like primitives. glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives.
- **glutSwapBuffers()** :- glutSwapBuffers swaps the buffers of the *current window* if double buffered. 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. An implicit glFlush is done by glutSwapBuffers before it returns. Ifthe *layer in use* is not double buffered, glutSwapBuffers has no effect.
- **glPushMatrix()** push and pop the current matrix stack. There is a stack of matrices for each of the matrix modes. The current matrix in any mode is the matrix on the top of the stack for that mode.glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, aftera glPushMatrix call, the matrix on top of the stack is identical to the one below it.
- **glPopMatrix()** :- push and pop the current matrix stack. glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.
- **glTranslate(a2,0,0)** :- multiply the current matrix by a translation matrix. glTranslate produces a translation by x y z . The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.
- **glutInit(&argc,argv)** :- glutInit is used to initialize the GLUT library. 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 withan error message to the user if GLUT cannot be properly initialized. glutInit also processes command line options, but the specific options parse are window system dependent.
- **glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)** :- sets the *initial display mode*. 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.

- **glutInitWindowSize(1000,500)** :- set the *initial window size* . Windows created by glutCreateWindow will be requested to be created with the current *initialwindow position* and *size*. The initial value of the *initial window size* GLUT state is 300 by 300. The *initial window size* components must be greater than zero.The intent of the *initial window position* and *size* values is to provide a suggestion to the window system for a window's initial size and position. The window system isnot obligated to use this information. A GLUT program should use the window's reshape callback to determine the true size of the window.
- **glutCreateWindow("HARE AND TORTOISE")** :- 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.
- **glutPositionWindow(50,50)** :- requests a change to the position of the *current window*. . For top-level windows, the x and y parameters are pixel offsets from the screen origin. For subwindows, the x and y parameters are pixel offsets from the window's parent window origin.
- **glutDisplayFunc(display1)** :- registers the callback function (or event handler) for handling window-paint event. The OpenGL graphic system calls back this handler when it receives a window re-paint request. In the example, we register the function display() as the handler.
- **glutKeyboardFunc(NormalKey)** :- 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 state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data.
- **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.

## 5.1.2 USER DEFINED FUNCTIONS:

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 "*Town Simulation*" are:

- **void display()** :- This function is used to display the town view. It calls other function like cloud(),make_cloud() and train(). It is also used to define and set the color. In the end there is a glflush() to display everything in the screen.
- **void sun()** :- This function is used to draw a filled color circle which stands for sun in the town view scene. The circle color is light yellow because this color denotes the sun in the early morning.
- **void cloud()** :- This function is used to draw a circular arc filled with white color which denotes the clouds in the town view.
- **void cloud_move_right()** :- This function is used to translate the clouds from left to right along the window.
- **void railline()** :- This function is used to draw a rail line along the road. This rail line is drawn as a rectangular box with black thick color borders.
- **void makerail()**:- It is the function used to design the train that moves on the rail line along the road.
- **void trainmove()** :- This function is used to move the train on the rail line. This function use gltranslatef() function to move the train that was deigned using makerail() function.
- **void windmill()**:- This function is used to design a windmill with a vertical post and a rotating fan. This windmill is placed at the near the hills that are visible from the road. we will be using a glrotatef() function to rotate the windmill. This gives an animation to the town view.

- **void house()** :- This function is used to build a house with the help 2d geometric shapes. This house is painted with rgb colors to look attractive.
- **void make_tree2()** :- This function is used to build the second type of tree with circular arcs in it.
- **void tree2()** :- This function is used to place the second type of the tree (that is built using make_tree2 function) on the canvas using glPopMatrix() and glPushMatrix().
- **void manmove()** :- This function is used to move the man from the bus to the house. This function uses keyboard functions to move the man. The man is translated using gltranslate() function along the road sides.
- **void privatecar()** :- This function is used to the car which will be moving on the screen along the road.
- **void spindisplay_left()** :- This function is a part of windmill() function. This function is used to control the spin of the fan that is used in the windmill. This function uses spin global variable for this purpose.
- **void init()** :- This function is basically used for initialization purpose. It initializes –

```
glClearColor(0.3,0.65,0.1,1.0)
glMatrixMode(GL_MODELVIEW)
glOrtho2D(-300,300,-300,300);
```

## 5.2 IMPLEMENTATION

This section speaks about the implementation of the project via the snapshots. It gives the detailed description of the way in which various scenes and frames are implemented.

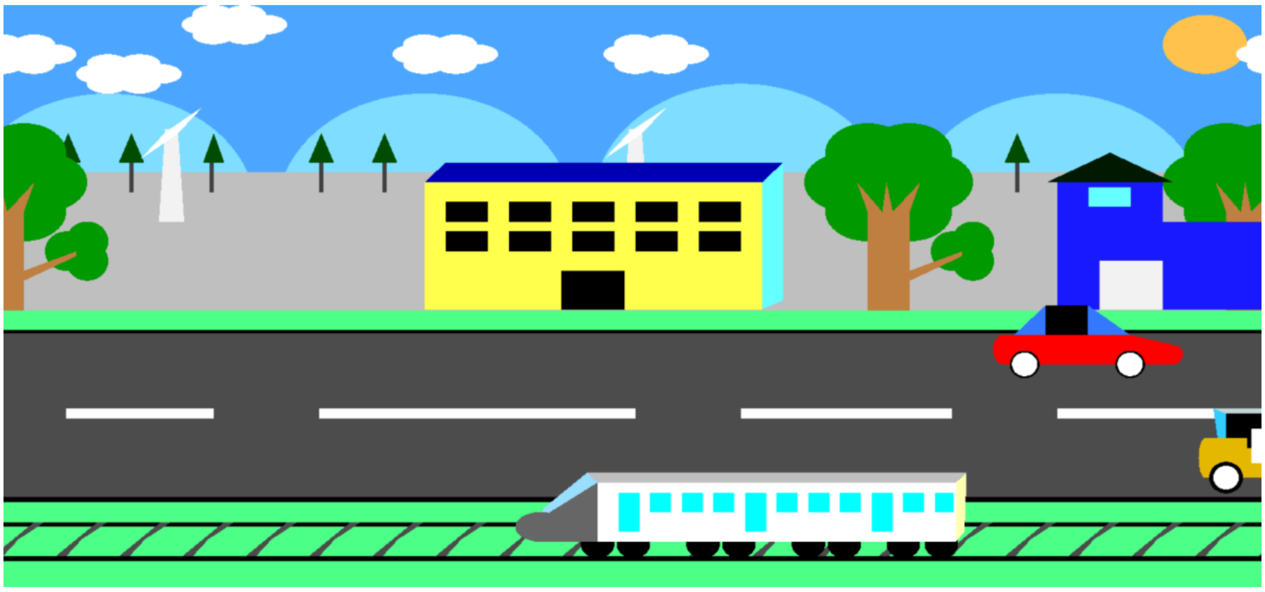


FIGURE 5.2.1

**First Screen** – This is the first scene when the project runs. It displays the town view where all the entities such as train, bus etc. are visible.

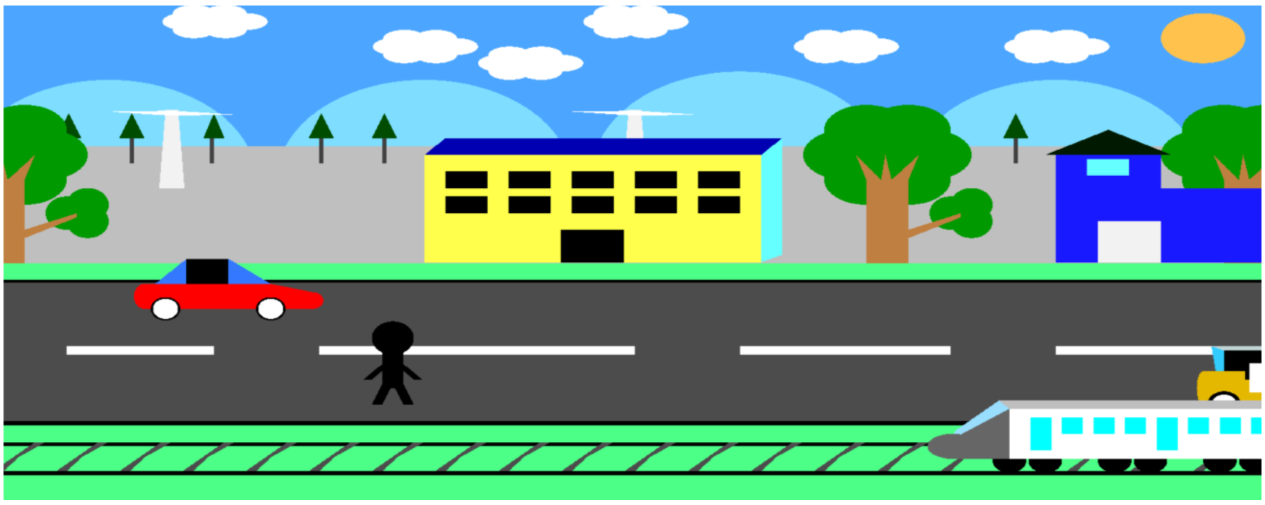


FIGURE 5.2.2

**Second Screen** – This is the second scene of the project. It displays man controlled using arrow keys.

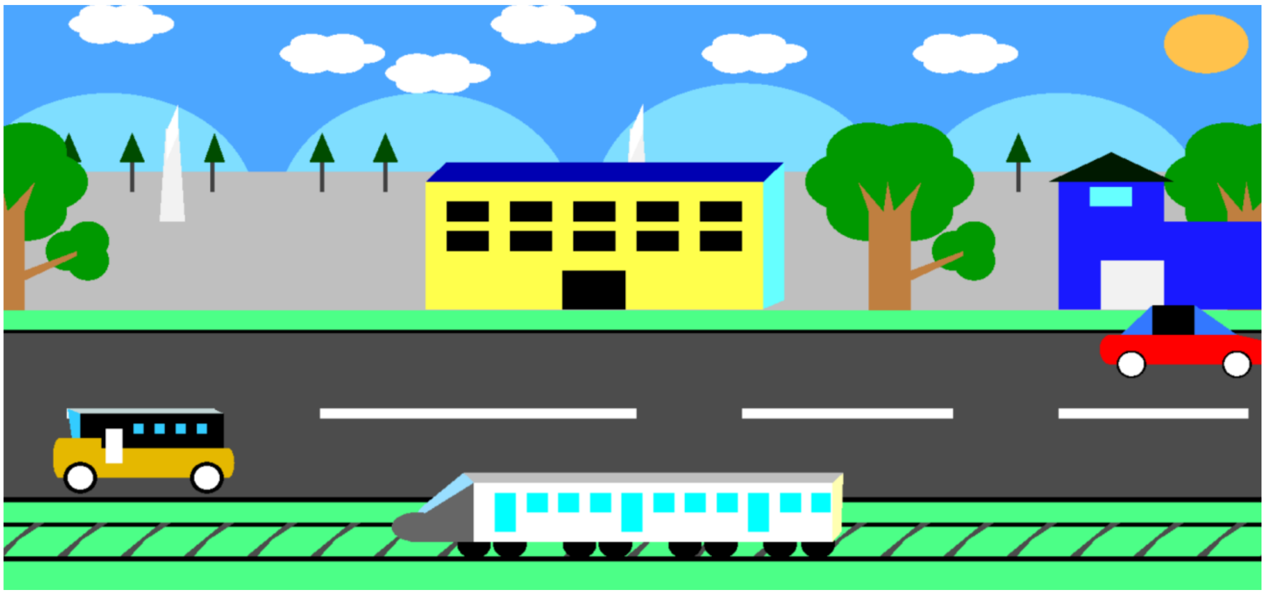


FIGURE 5.2.3

**Third Screen** – This is the third scene of the project. In this scene, if we press “s” on the key board then the bus starts moving.

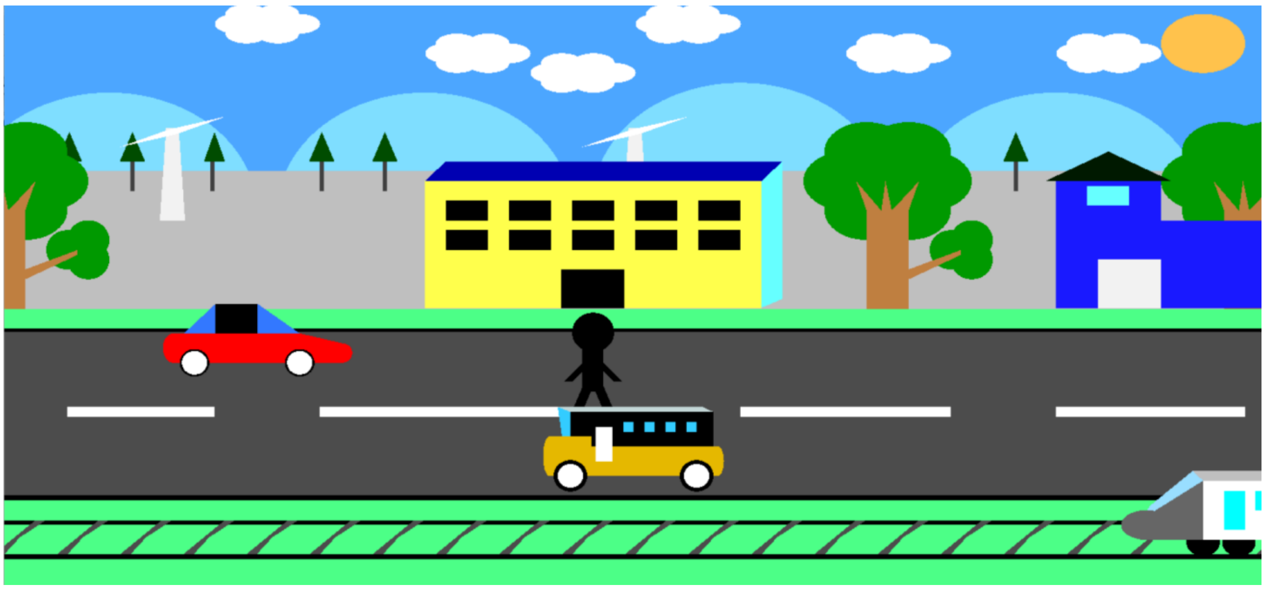
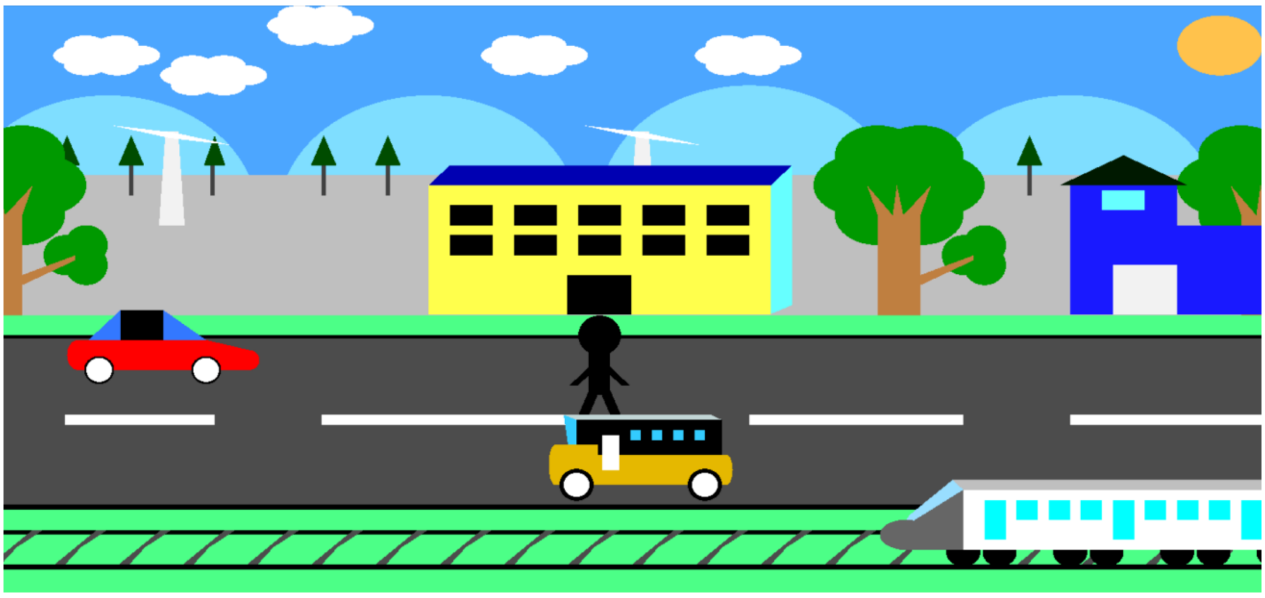


FIGURE 5.2.4

This is the **fourth scene** of the project. In this scene if we press “m” on the keyboard the a man gets out of the bus and he start moving to the house.

**FIGURE 5.2.5**

This is the **fifth scene** of the project. In this scene if we press “b” on the keyboard and it calls the idle function which stops the man who is moving from bus.



# CONCLUSION AND FUTURE ENHANCEMENT

## CONCLUSION

The very purpose of developing this project is to exploit the strength of OpenGL graphics capabilities for a interesting cause. The Town Simulation has been tested under Windows 10, and has been found to provide ease of use and manipulation to the user. The Town Simulation created for the Windows operating system can be used to draw lines, boxes, circles, ellipses, and polygons. It has a very simple and effective user interface.

We found designing and developing this Town as a very interesting and learning experience. It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. The graphics editor provides all and more than the features that have been detailed in the university syllabus.

## FUTURE ENHANCEMENTS

Some of the future enhancements would be:

- Support for advanced 3D representation of the entire scenario.
- Support for transparency of layers and originality that is, simulating the story in amore realistic way.
- Making the user interface of this project more user friendly which will certainlybe more effective.

# REFERENCES

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 4th Edition,Pearson Education, 2011.

[2] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL,5th edition. Pearson Education, 2008.

[3] Jackie.L.Neider,Mark Warhol,Tom.R.Davis,"OpenGL Red Book",Second RevisedEdition,2005.

[4] www.opengl.org