COMPUTER GRAPHICS (UCS505)

PROJECT NAME
BUS SIMULATION

3C33

B.E. 3RD YEAR – COE/CSE

SUBMITTED BY –

DHRUVI SINGH (102203536)

SMRITI SINGH (102203604)

TANYA (102203548)

SUBMITTED TO –
DR. KUNTAL CHOWDHURY

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

PATIALA – 147001

TABLE OF CONTENT

INTRODUCTION

1. Project Overview:

- The Computer Graphics Bus Simulation project is a dynamic, interactive display system designed to enhance the commuter experience at bus stops using computer graphics.
- It aims to simulate a real-time 3D environment where passengers can view live bus schedules, arrival and departure times, and track bus positions through animated visuals.
- Built using OpenGL and GLUT, the system features visually rich elements like a 3D-modeled bus stop, animated buses, environmental objects (like lamp posts, trees), and human figures for realism.
- A graphical user interface (GUI) with basic interactive elements (like clickable regions or buttons) allows users to search routes or receive instructions.
- The project combines visual appeal (through lighting, textures, and motion) with functional feedback, enhancing usability and realism.
- Ultimately, it is a prototype for smart public transport systems that deliver accurate, timely, and accessible transit information.

2. Scope of the Project:

- **3D Modeling & Scene Rendering:** Includes rendering of a realistic bus stop environment with elements such as shelters, benches, lamp posts, trees, roads, and characters using OpenGL primitives.
- **Real-Time Bus Simulation:** Implements animation of buses arriving, halting, and departing using translation functions and timer callbacks for realistic motion.
- **User Interaction (UI):** Adds a basic interactive interface for keyboard/mouse input to control or view specific bus routes or simulate user choices.
- **Text Rendering:** Displays real-time instructions, route names, and feedback prompts using GLUT bitmap fonts.
- **Lighting and Visual Effects**: Uses OpenGL lighting and shading to create depth, realism, and ambient effects in the 3D space.
- **Animation and Motion Graphics**: Utilizes animated movements (e.g., buses moving along roads) and updates based on user inputs or timers.
- **Modular Design:** Divides the environment into reusable components (e.g., buses, people, text, lamp posts), supporting extensibility for future upgrades.
- **Scalability for Real-Time Integration (optional future scope):** Architecture allows for future integration with sensors or live APIs to fetch actual bus data.
- **Educational Utility**: Demonstrates core computer graphics concepts such as transformations, lighting, polygon modeling, and animation.

USER-DEFINED FUNCTIONS

S No.	Function Name	Function Description
1	wheel1	Draws front wheels of the first bus.
2	wheel2	Draws rear wheels of the first bus.
3	polygon	Renders a colored polygon (face of the bus).
4	colorcube	Assembles and renders the full 3D model of the first bus.
5	bus_stop	Draws the 3D structure of the bus stop including shelter and chairs.
6	road2	Renders road layout with lane markings.
7	text	Displays "BUS STOP" label on the screen.
8	text1	Displays the name of the bus.
9	text2	Shows user instructions like "Pick up the woman...".
10	text3	Displays the next instruction or message after action.
11	text4d	Instructional message for drop-off location.
12	text5d	Displays success message "Mission Accomplished!".
13	line	Draws decorative or detailed lines on bus.
14	tree12	Renders a 3D tree near the stop.
15	tree1	Draws another 3D tree near the road.
16	woman	Renders a 3D woman character with face and full body.
17	man	Renders a 3D man character.
18	lamppost	Draws a decorative lamppost with bulbs and lanterns.
19	lamppost1	Renders a second lamppost at a different location.
20	lamppost2	Draws a third lamppost variation.
21	lamppost4	Another variation of a lamppost, possibly at a new scene.
22	wheel1d	Draws front wheels of the second bus.
23	wheel2d	Draws rear wheels of the second bus.
24	polygond	Polygon renderer for second bus (similar to polygon).
25	colorcubed	Assembles and renders second 3D bus model.
26	womand	Draws a second woman figure (e.g., dropped version).
27	road2d	Draws the extended road layout for second scene.
28	text1_bus1	Displays the label “TIET BUS(CSE)” for the first bus using bitmap fonts.
29	text1_bus2	Displays the label “TIET BUS(ENC)” for the second bus.
30	textd	Placeholder function that can display text at a specific position (currently blank).
31	text1d	Displays “TIET BUS(CSE)” on the destination scene after drop-off.
32	text2d	Displays the text “TIET” on the second scene.
33	text3d	Displays the text “STOP” in the destination scene.
34	buildingd	Draws a building with a door and multiple windows for the second scene.
35	lined	Draws decorative black lines on the front face of the second bus.
36	walld	Renders left, right, and middle brick-patterned walls around the building.
37	gated	Draws a decorative gate for the destination area using black polygons.
38	treed	Renders a large tree with a trunk and three conical layers of leaves.
39	tree1d	Renders another tree variant for the second scene.
40	tree2d	Draws an additional tree on the destination layout.
41	shrubd	Renders a bush with green cubes, spheres, and red flowers.
42	shrub1d	Variant of the shrub, positioned differently with yellow flowers.

43	shrub2d	Another shrub instance, with red flowers and different position.
44	shrub3d	Additional shrub near the right side, with red detailing.
45	shrub4d	Final shrub variant with red and pink flower highlights.
46	stopd	Draws a stop sign with a grey bar, torus, and sphere — used as a visual bus stop indicator.
47	intro	Renders the intro background with multiple-colored layered polygons.
48	texti	Displays the title screen text, including department, project name, team members, and guide.
49	mouse	Mouse event handler — activates the scene when left-clicked.
50	bus_move	Animates the first bus and associated woman figure as it approaches and leaves the stop.
51	bus_move2	Animates the second bus movement on a separate road and controls its transition.
52	bus_moved	Animates the bus at the destination scene with a drop-off sequence.
53	SpecialKeyFunc	Handles keyboard inputs like UP and RIGHT arrows to trigger display updates.
54	display	Main render function that handles scene switching, drawing, and animation sequencing.
55	myreshape	Handles reshaping the viewport and maintaining the aspect ratio.
56	main	Initializes the GLUT environment, sets display parameters, and starts the main loop.

CODE SNIPPETS

```
#include<openGL/gl.h>
#include<GLUT/glut.h>
#include <stdio.h>
#include <string.h>
#include<math.h>
#include <stdlib.h>



int x = -150, o = 0, xd = -150;
int x1 = 0, z = 0;
float a = 0, a1 = 0, moving, angle = 0;
float z5 = 0, u = 0, flag12 = 0, begin;
float k = 0, y2 = 0;
int flag = 0, flag55 = 0, var = 0, flag1 = 0, flag551 = 0, vari = 0, vard = 0, varid = 0, then = 0;
float p = 0.0, q = 0.0;
#define maxx 10
#define maxy 12
#define dx 27.7
#define dy 12
GLfloat xangle = 0.0, yangle = 0.0, zangle = 0.0;   /* axis angles */
// Add these near other bus position variables
int x2 = -150;  // Position for second bus
int flag2 = 0, flag552 = 0, var2 = 0;  // Control flags for second bus

/*--------------------------------------------------------------------*/
/*************************DECLARATIONS*************************/
/*--------------------------------------------------------------------*/

GLfloat vertices[][3] = { {160,390,-70},{425,390,-70},
                          {425,510,-70}, {160,520,-70},

                          {135,370,70}, {400,370,70},
                          {400,490,70}, {135,500,70},

                          {135,447,70},{400,447,70},
                          {425,467,-70},{410,520,-70},

                          {385,500,70}, {160,467,-70},
```

```
                          {320,467,-70},{320,520,-70},

{380,520,-70},{380,390,-70},{320,390,-70} };

GLfloat colors[][3] = { {1.0,1.0,0.0},{0.0,0.6,0.7},{.3,.4,.5} };

GLfloat verticesd[][3] = { {160,390 - 175,-70},{425,390 - 175,-70},
                           {425,510 - 175,-70}, {160,520 - 175,-70},

                           {135,370 - 175,70}, {400,370 - 175,70},
                           {400,490 - 175,70}, {135,500 - 175,70},

                           {135,447 - 175,70},{400,447 - 175,70},
                           {425,467 - 175,-70},{410,520 - 175,-70},

                           {385,500 - 175,70}, {160,467 - 175,-70},
                           {320,467 - 175,-70},{320,520 - 175,-70},

{380,520 - 175,-70},{380,390 - 175,-70},{320,390 - 175,-70} };

GLfloat colorsd[][3] = { {1.0,1.0,0.0},{0.0,0.6,0.7},{.3,.4,.5} };
Tabnine | Edit | Test | Explain | Document
void text1_bus1();
Tabnine | Edit | Test | Explain | Document
void text1_bus2();
//    FUNCTION wheel
/*--------------------------------------------------------------------*/
Tabnine | Edit | Test | Explain | Document
void wheel1()
{
    glColor3f(0, 0, 0);

    glPushMatrix();
    glTranslatef(345, 377, -70);
    glutSolidTorus(5, 15, 100, 90);
    glPopMatrix();

    glPushMatrix();
```

```
    glPushMatrix();
    glTranslatef(190, 377, -70);
    glutSolidTorus(5, 15, 100, 90);//front two wheels tyre
    glPopMatrix();

    glColor3ub(100, 100, 100);

    glPushMatrix();
    glTranslatef(345, 377, -70);
    glutSolidTorus(5, 5, 10, 69);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(190, 377, -70);
    glutSolidTorus(5, 5, 10, 69);
    glPopMatrix();// front two wheels
}

Tabnine | Edit | Test | Explain | Document
void wheel2()
{
    glColor3f(0, 0, 0);

    glPushMatrix();
    glTranslatef(180, 370, 70);
    glutSolidTorus(5, 15, 100, 90);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(335, 370, 70);
    glutSolidTorus(5, 15, 100, 90);
    glPopMatrix();            //back two wheels tyre

    glColor3ub(100, 100, 100);

    glPushMatrix();
    glTranslatef(335, 370, 70);
    glutSolidTorus(5, 5, 10, 69);
```

```
    glPushMatrix();
    glTranslatef(180, 370, 70);
    glutSolidTorus(5, 5, 10, 69);
    glPopMatrix();      //back two wheels
}
/*-------------------------------------------------------------------------*/
|   |   |   |   //    FUNCTION cube
/*-------------------------------------------------------------------------*/
Tabnine | Edit | Test | Explain | Document
void polygon(int a, int b, int c, int d, int E, int f)
{
    glBegin(GL_POLYGON);
    glColor3fv(colors[E]);
    glVertex3fv(vertices[a]);
    glVertex3fv(vertices[b]);
    glVertex3fv(vertices[c]);
    glVertex3fv(vertices[d]);
    if (f != 0)
    |   glVertex3fv(vertices[f]);
    glEnd();
}

Tabnine | Edit | Test | Explain | Document
void colorcube()
{
    int i;
    wheel1();
    polygon(0, 1, 5, 4, 0, 0);


    polygon(13, 14, 18, 0, 0, 0);
    polygon(15, 16, 17, 18, 2, 0);
    polygon(16, 11, 2, 1, 0, 17);


    polygon(0, 4, 8, 13, 0, 0);
    polygon(1, 10, 9, 5, 0, 0);
    polygon(9, 10, 2, 6, 1, 0);
    polygon(4, 5, 9, 8, 0, 0);
```

```
Tabnine | Edit | Test | Explain | Document
void bus_stop()
{

    /**********  ground   **********/

    glColor3ub(100, 100, 100);
    glBegin(GL_POLYGON);
    glVertex3i(340 - 200, 470, -110);
    glVertex3i(680 - 200, 470, -110);
    glVertex3i(710 - 200, 500, -240);
    glVertex3i(370 - 200, 500, -240);
    glEnd();
    glColor3ub(100, 100, 100);
    glBegin(GL_POLYGON);
    glVertex3i(340 - 200, 470, -110);
    glVertex3i(680 - 200, 470, -110);
    glVertex3i(680 - 200, 450, -110);
    glVertex3i(340 - 200, 450, -110);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3i(680 - 200, 470, -110);
    glVertex3i(710 - 200, 500, -240);
    glVertex3i(710 - 200, 480, -240);
    glVertex3i(680 - 200, 450, -110);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3i(710 - 200, 500, -240);
    glVertex3i(710 - 200, 480, -240);
    glVertex3i(370 - 200, 480, -240);
    glVertex3i(370 - 200, 500, -240);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3i(370 - 200, 480, -240);
    glVertex3i(370 - 200, 500, -240);
    glVertex3i(340 - 200, 470, -110);
    glVertex3i(340 - 200, 450, -110);
    glEnd();
```

```
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(700 - 200, 600, -120);
    glVertex3i(350 - 200, 600, -120);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3i(350 - 200, 620, -120);
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(720 - 200, 640, -240);
    glVertex3i(380 - 200, 640, -240);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(720 - 200, 640, -240);
    glVertex3i(720 - 200, 620, -240);
    glVertex3i(700 - 200, 600, -120);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3i(350 - 200, 600, -120);
    glVertex3i(350 - 200, 620, -120);
    glVertex3i(380 - 200, 640, -240);
    glVertex3i(380 - 200, 620, -240);
    glEnd();


    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINES);
    glVertex3i(350 - 200, 620, -120);
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(720 - 200, 640, -240);
    glVertex3i(700 - 200, 620, -120);
    glVertex3i(700 - 200, 600, -120);
    glEnd();
```

```
/************* left  ************/
glColor3ub(10, 50, 80);
glBegin(GL_POLYGON);
glVertex3i(370 - 200, 610, -140);
glVertex3i(400 - 200, 625, -200);
glVertex3i(400 - 200, 490, -200);
glVertex3i(370 - 200, 480, -140);
glEnd();
/************* mid  ************/

glColor3ub(10, 170, 80);
glBegin(GL_POLYGON);
glVertex3i(395 - 200, 580, -200);
glVertex3i(690 - 200, 580, -200);
glVertex3i(690 - 200, 520, -200);
glVertex3i(395 - 200, 520, -200);
glEnd();

glColor3f(0, 0, 0);
glBegin(GL_LINES);
glVertex3i(395 - 200, 580, -200);
glVertex3i(690 - 200, 580, -200);
glVertex3i(690 - 200, 520, -200);
glVertex3i(395 - 200, 520, -200);
glEnd();
/************* right  ************/

glColor3ub(10, 50, 80);
glBegin(GL_POLYGON);
glVertex3i(690 - 200, 625, -200);
glVertex3i(670 - 200, 610, -140);
glVertex3i(670 - 200, 475, -140);
glVertex3i(690 - 200, 490, -200);
glEnd();


/***************** chair *****************/
```

```
Tabnine | Edit | Test | Explain | Document
357  void road2()
358  {
359      /************* left part of road  *********/
360      int x, y;
361      glColor3ub(7, 255, 13);
362      glBegin(GL_POLYGON);
363      glVertex2i(0, 650);
364      glVertex2i(1000, 650);
365      glVertex2i(1000, 0);
366      glVertex2i(0, 0);
367      glEnd();
368
369      glColor3ub(30, 40, 50);
370      glBegin(GL_POLYGON);
371      glVertex2i(0, 420);
372      glVertex2i(1000, 420);
373      glVertex2i(1000, 300);
374      glVertex2i(0, 300);
375      glEnd();
376
377      glBegin(GL_POLYGON);
378      glVertex2i(750, 650);
379      glVertex2i(900, 650);
380      glVertex2i(1000, 0);
381      glVertex2i(650, 0);
382      glEnd();
383
384
385      /************  STRIPES  ************/
386
387      glColor3f(1.0, 0.9, 0.0);
388      for (x = 0;x < 1000;x = x + 60)
389      {
390          glBegin(GL_POLYGON);
391          glVertex2f(x, 352.5 + 19);
392          glVertex2f(x, 357.5 + 19);
393          glVertex2f(x + 30, 357.5 + 19);
394          glVertex2f(x + 30, 352.5 + 19);
395          glEnd();
```

```
void text1()
{
    char string2[] = "TIET BUS(CSE) ";
    void* font2 = GLUT_BITMAP_TIMES_ROMAN_24;
    int k;
    glColor3f(0.0, 0.0, 0.0);
    glRasterPos3f(230 + p, 400, 70);
    for (k = 0;k < strlen(string2);k++)
        glutBitmapCharacter(font2, string2[k]);
}
Tabnine | Edit | Test | Explain | Document
void text2()
{
    glBegin(GL_POLYGON);
    glColor3f(1.0, 1.0, 1.0);
    glVertex2i(830 - 500, 120 + 150);
    glVertex2i(1000 - 500 + 40, 120 + 150);
    glVertex2i(1000 - 500 + 40, 35 + 150);
    glVertex2i(830 - 500, 35 + 150);
    glEnd();


    char string2[] = "Pick up the woman";
    void* font2 = GLUT_BITMAP_TIMES_ROMAN_24;
    int k;
    glColor3f(0.0, 0.0, 0.0);
    glRasterPos3f(830 - 500 + 20, 100 + 150, 70);
    for (k = 0;k < strlen(string2);k++)
        glutBitmapCharacter(font2, string2[k]);

    char string3[] = " at the bus stop";
    void* font3 = GLUT_BITMAP_TIMES_ROMAN_24;
    glColor3f(0.0, 0.0, 0.0);
    glRasterPos3f(830 - 500 + 20, 80 + 150, 70);
    for (k = 0;k < strlen(string3);k++)
        glutBitmapCharacter(font3, string3[k]);

    char string4[] = " using the arrow ";
    void* font4 = GLUT_BITMAP_TIMES_ROMAN_24;
```

```
    void* font5 = GLUT_BITMAP_TIMES_ROMAN_24;
    glColor3f(0.0, 0.0, 0.0);
    glRasterPos3f(830 - 500 + 20, 40 + 150, 70);
    for (k = 0;k < strlen(string5);k++)
        glutBitmapCharacter(font5, string5[k]);

}
Tabnine | Edit | Test | Explain | Document
void text3()
{
    glBegin(GL_POLYGON);
    glColor3ub(0, 0, 0);
    glVertex2i(830 - 500, 120 + 150);
    glVertex2i(1020 - 500 + 40, 120 + 150);
    glVertex2i(1020 - 500 + 40, 35 + 150);
    glVertex2i(830 - 500, 35 + 150);
    glEnd();

    char string2[] = "YAY!Now get her ";
    void* font2 = GLUT_BITMAP_TIMES_ROMAN_24;
    int k;
    glColor3ub(240, 0, 0);
    glRasterPos3f(832 - 500 + 20, 100 + 150, 70);
    for (k = 0;k < strlen(string2);k++)
        glutBitmapCharacter(font2, string2[k]);

    char string3[] = "to her college";
    void* font3 = GLUT_BITMAP_TIMES_ROMAN_24;
    glColor3ub(240, 0, 0);
    glRasterPos3f(832 - 500 + 20, 100 + 130, 70);
    for (k = 0;k < strlen(string3);k++)
        glutBitmapCharacter(font3, string3[k]);


    char string4[] = "Just straight ahead.";
    void* font4 = GLUT_BITMAP_TIMES_ROMAN_24;
    glColor3ub(240, 0, 0);
    glRasterPos3f(834 - 500 + 20, 100 + 110, 70);
```

```
void text5d()
{
    glBegin(GL_POLYGON);
    glColor3ub(20, 3, 5);
    glVertex2i(830 - 500, 120 - 50);
    glVertex2i(1060 - 500, 120 - 50);
    glVertex2i(1060 - 500, 35 - 50);
    glVertex2i(830 - 500, 35 - 50);
    glEnd();

    char string2[] = "Mission Accomplished! ";
    void* font2 = GLUT_BITMAP_TIMES_ROMAN_24;
    int k;
    glColor3ub(255, 255, 255);
    glRasterPos3f(832 - 500, 100 - 50, 70);
    for (k = 0;k < strlen(string2);k++)
        glutBitmapCharacter(font2, string2[k]);

    char string3[] = "Parking is right ahead";
    void* font3 = GLUT_BITMAP_TIMES_ROMAN_24;
    glColor3ub(255, 255, 255);
    glRasterPos3f(832 - 500, 100 - 70, 70);
    for (k = 0;k < strlen(string3);k++)
        glutBitmapCharacter(font3, string3[k]);




}
/*-----------------------------------------------------------------
    //    FUNCTION line
/*-----------------------------------------------------------------
Tabnine | Edit | Test | Explain | Document
void line()
{
    // lines on d front face
    glBegin(GL_POLYGON);
    glColor3ub(0, 0, 0);
```

```
void tree12()
{
    //trunk1
    glColor3ub(95, 6, 5);
    double len = 100;
    double thick = 20;
    glPushMatrix();
    glTranslated(100 + 450, 150 + 330, 0.0);
    glScaled(thick, len, thick);
    glutSolidCube(1.0);
    glPopMatrix();

    //leaves1


    glColor3f(0.0, 0.2, 0.0);
    glPushMatrix();
    glLoadIdentity();

    glTranslated(100 + 450, 230 + 310, 0.0);
    glutSolidCone(70, 140, 3, 2);
    glPopMatrix();

    //leaves2

    glColor3f(0.0, 0.2, 0.0);
    glPushMatrix();
    glLoadIdentity();
    glTranslated(100 + 450, 260 + 310, 0.0);
    glutSolidCone(60, 120, 3, 2);
    glPopMatrix();

    // leaves3

    glColor3f(0.0, 0.2, 0.0);
    glPushMatrix();
    glLoadIdentity();
    glTranslated(100 + 450, 290 + 310, 0);
    glutSolidCone(50, 100, 3, 2);
```

```
690  void woman()
691  {
692      //face
693      glColor3ub(0, 0, 0);
694      glPushMatrix();
695      glTranslatef(540, 495, 0);
696      glutSolidTorus(1, 10, 100, 90);
697      glPopMatrix();
698      glColor3ub(255, 191, 128);
699      glPushMatrix();
700      glTranslatef(540, 494, 0);
701      glutSolidTorus(7, 7, 100, 90);
702      glPopMatrix();
703      glColor3ub(0, 0, 0);
704      glBegin(GL_LINES);
705      glVertex2i(540, 494);
706      glVertex2i(540, 490); //nose
707      glVertex2i(531, 498);
708      glVertex2i(532, 499);
709      glVertex2i(532, 499);
710      glVertex2i(537, 498);//eyebrow
711      glVertex2i(549, 498);
712      glVertex2i(548, 499);
713      glVertex2i(548, 499);
714      glVertex2i(543, 498);//eyebrow
715      glEnd();
716      //ear right
717      glBegin(GL_POLYGON);
718      glColor3ub(255, 191, 128);
719      glVertex2i(540 - 14, 494 + 1);
720      glVertex2i(540 - 14, 490 + 1);
721      glVertex2i(538 - 14, 489 + 1);
722      glVertex2i(538 - 14, 495 + 1);
723      glEnd();
724      //ear left
725      glBegin(GL_POLYGON);
726      glColor3ub(255, 191, 128);
727      glVertex2i(554, 495);
728      glVertex2i(556, 496);
```

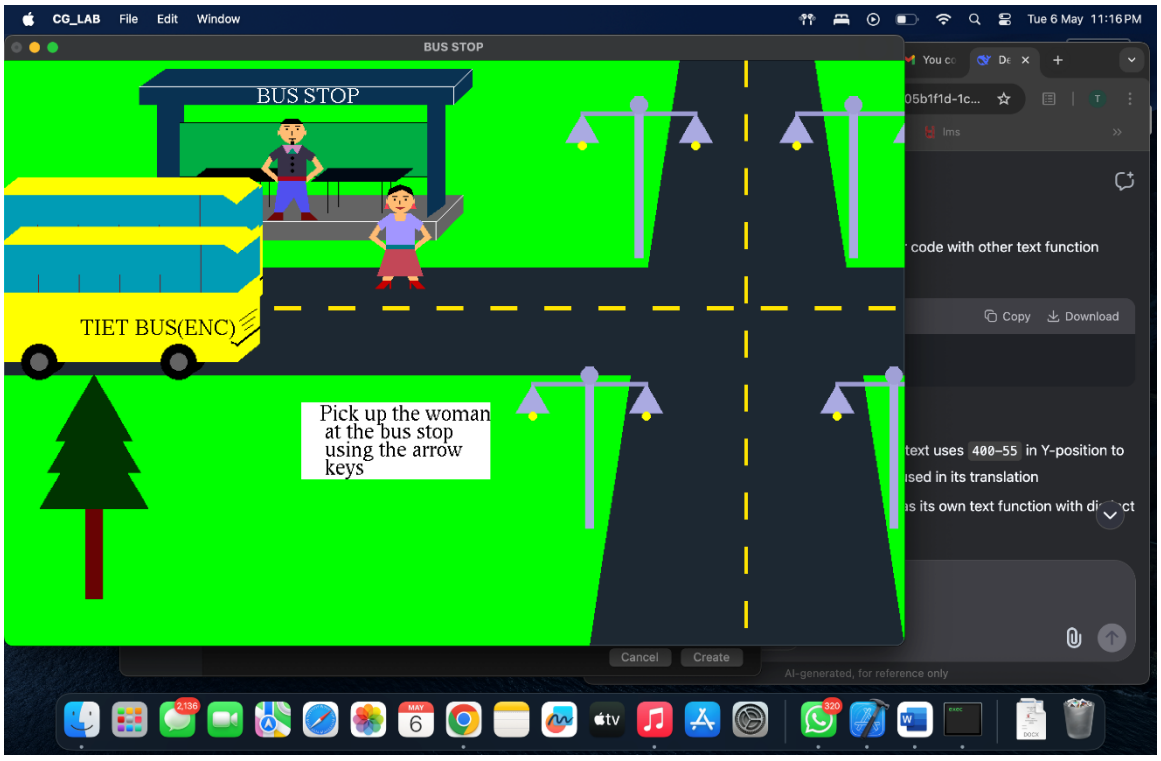
```
    glEnd();
    //hands
    glBegin(GL_POLYGON);
    glColor3ub(255, 191, 128);
    glVertex2i(565, 468);
    glVertex2i(575, 453);
    glVertex2i(567, 454);
    glVertex2i(562, 462);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(575, 453);
    glVertex2i(556, 438);
    glVertex2i(556, 445);
    glVertex2i(567, 454);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(515, 468);
    glVertex2i(505, 453);
    glVertex2i(513, 454);
    glVertex2i(518, 462);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex2i(505, 453);
    glVertex2i(524, 438);
    glVertex2i(524, 445);
    glVertex2i(513, 454);
    glEnd();
    // belt
    glBegin(GL_POLYGON);
    glColor3ub(10, 120, 130);
    glVertex2i(556, 445);
    glVertex2i(524, 445);
    glVertex2i(524, 440);
    glVertex2i(556, 440);
    glEnd();


    /// leg
```

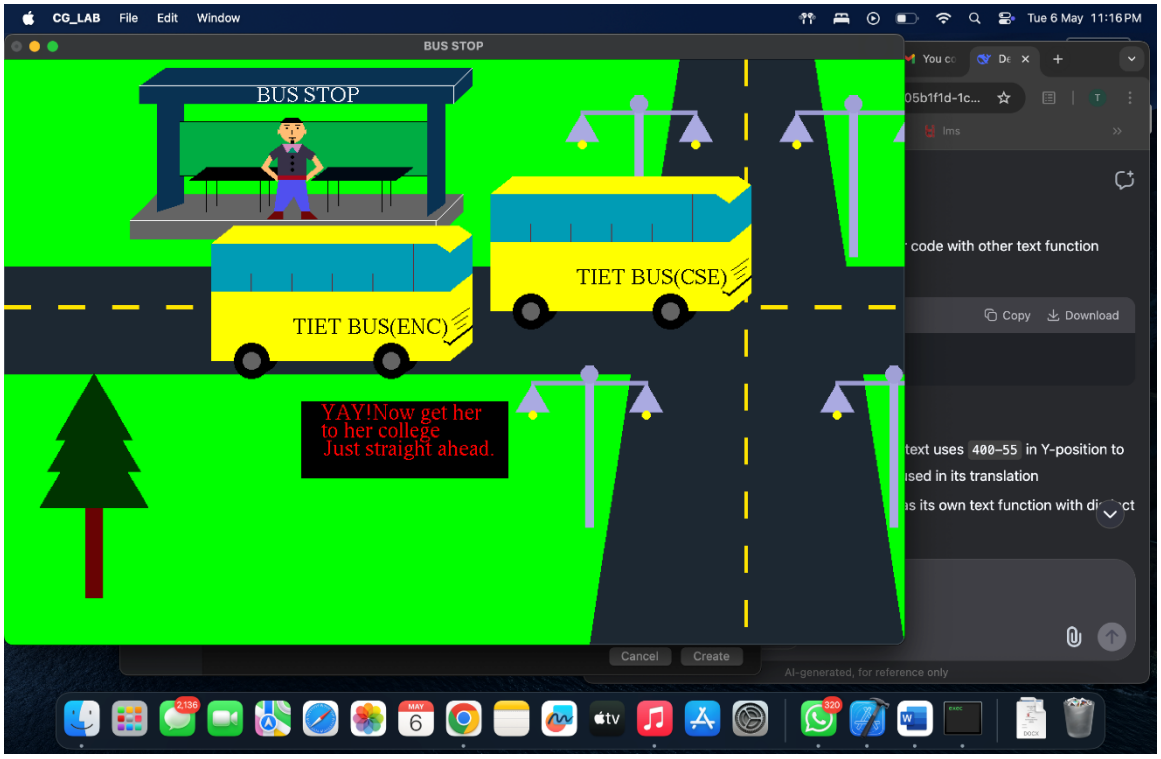
```
847         glVertex2i(520, 405);
848         glVertex2i(530, 405);
849         glVertex2i(533, 438);
850         glVertex2i(550, 405);
851         glVertex2i(560, 405);
852         glEnd();
853         //skirt
854         glBegin(GL_POLYGON);
855         glColor3ub(180, 80, 90);
856         glVertex2i(524, 440);
857         glVertex2i(556, 440);
858         glVertex2i(566, 410);
859         glVertex2i(514, 410);
860         glEnd();
861         //shoe left
862         glBegin(GL_POLYGON);
863         glColor3ub(180, 0, 0);
864         glVertex2i(530, 405);
865         glVertex2i(530, 396);
866         glVertex2i(528, 396);
867         glVertex2i(528, 404);
868         glVertex2i(522, 396);
869         glVertex2i(512, 396);
870         glVertex2i(520, 405);
871
872         glEnd();
873         //shoe right
874         glBegin(GL_POLYGON);
875         glColor3ub(180, 0, 0);
876         glVertex2i(550, 405);
877         glVertex2i(550, 396);
878         glVertex2i(552, 396);
879         glVertex2i(552, 404);
880         glVertex2i(558, 396);
881         glVertex2i(568, 396);
882         glVertex2i(560, 405);
883         glEnd();
884
```

```
889     void man()
890     {
891         glColor3ub(0, 0, 0);
892         glPushMatrix();
893         glTranslatef(540 - 220, 495 + 76, 0);
894         glutSolidTorus(1, 10, 100, 90);
895         glPopMatrix();
896         glColor3ub(255, 191, 128);
897         glPushMatrix();
898         glTranslatef(540 - 220, 495 + 76, 0);
899         glutSolidTorus(7, 7, 100, 90);
900         glPopMatrix();
901         glColor3ub(0, 0, 0);
902         glBegin(GL_LINES);
903         glVertex2i(540 - 220, 495 + 76);
904         glVertex2i(540 - 220, 490 + 76); //nose
905         glVertex2i(531 - 220, 500 + 76);
906         glVertex2i(537 - 220, 500 + 76);//eyebrow
907         glVertex2i(543 - 220, 500 + 76);
908         glVertex2i(549 - 220, 500 + 76);//eyebrow
909         glEnd();
910         //ear right
911         glBegin(GL_POLYGON);
912         glColor3ub(255, 191, 128);
913         glVertex2i(540 - 14 - 220, 494 + 1 + 76);
914         glVertex2i(540 - 14 - 220, 490 + 1 + 76);
915         glVertex2i(538 - 14 - 220, 489 + 1 + 76);
916         glVertex2i(538 - 14 - 220, 495 + 1 + 76);
917         glEnd();
918         //ear left
919         glBegin(GL_POLYGON);
920         glColor3ub(255, 191, 128);
921         glVertex2i(554 - 220, 495 + 76);
922         glVertex2i(556 - 220, 496 + 76);
923         glVertex2i(556 - 220, 491 + 76);
924         glVertex2i(554 - 220, 490 + 76);
925         glEnd();
926         //hair
927         glBegin(GL_POLYGON);
```

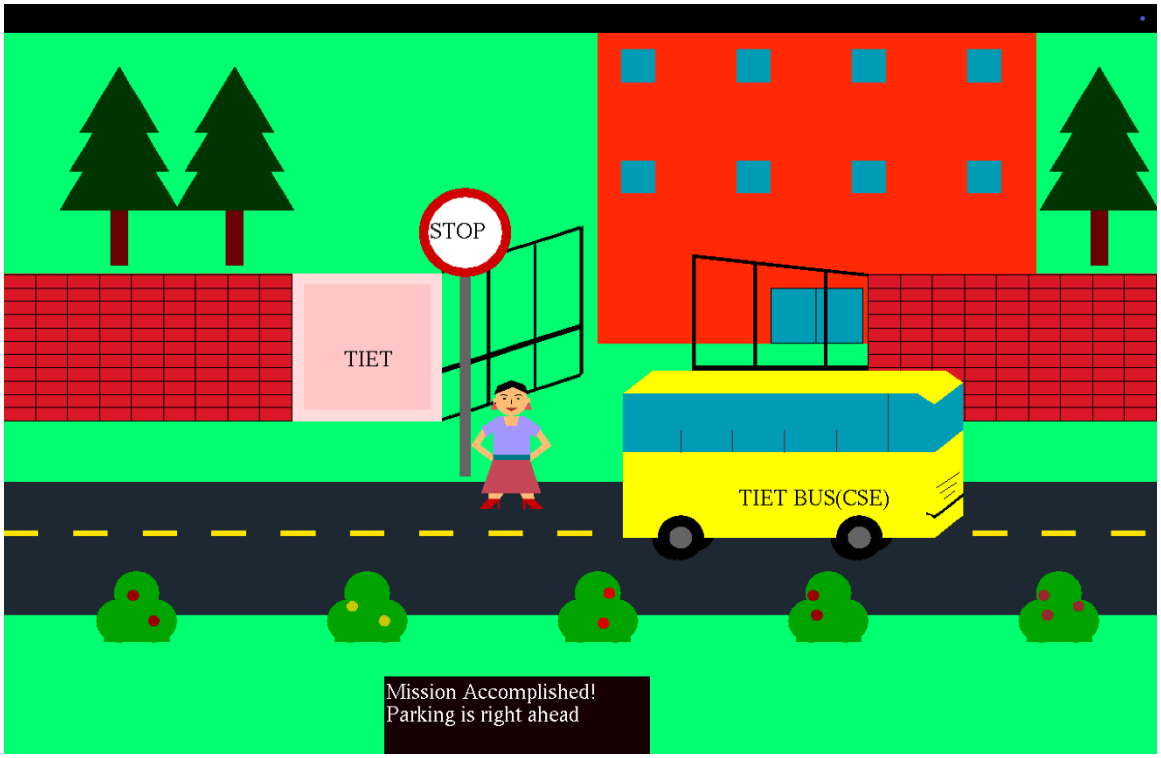
```
// belt
glBegin(GL_POLYGON);
glColor3ub(150, 12, 30);
glVertex2i(556 - 220, 445 + 76);
glVertex2i(524 - 220, 445 + 76);
glVertex2i(524 - 220, 440 + 76);
glVertex2i(524 - 220, 440 + 76);

glVertex2i(556 - 220, 440 + 76);
glEnd();
// collar
glBegin(GL_POLYGON);
glColor3ub(200, 140, 110 + 76);
glVertex2i(529 - 220, 480 + 76);
glVertex2i(551 - 220, 480 + 76);
glVertex2i(546 - 220, 470 + 76);
glVertex2i(534 - 220, 470 + 76);
glEnd();

glBegin(GL_TRIANGLES);
glColor3ub(20, 140, 110);
glVertex2i(540 - 220, 477 + 76);
glVertex2i(545 - 220, 470 + 76);
glVertex2i(535 - 220, 470 + 76);
glEnd();

// buttons
glColor3ub(0, 0, 0);
glPushMatrix();
glTranslatef(540 - 220, 465 + 76, 0);
glutSolidTorus(1, 1, 100, 90);
glPopMatrix();
glPushMatrix();
glTranslatef(540 - 220, 458 + 76, 0);
glutSolidTorus(1, 1, 100, 90);
glPopMatrix();
glPushMatrix();
glTranslatef(540 - 220, 451 + 76, 0);
glutSolidTorus(1, 1, 100, 90);
glPopMatrix();
glPushMatrix();
glTranslatef(540 - 220, 451 + 76, 0);
glutSolidTorus(1, 1, 100, 90);
glPopMatrix();

/// pant
glBegin(GL_POLYGON);
glColor3ub(80, 80, 230);
glVertex2i(555 - 220, 440 + 76);
glVertex2i(525 - 220, 440 + 76);
glVertex2i(520 - 220, 405 + 76);
glVertex2i(530 - 220, 405 + 76);
glVertex2i(533 - 220, 438 + 76);
glVertex2i(550 - 220, 405 + 76);
glVertex2i(560 - 220, 405 + 76);
glEnd();
//shoe left
glBegin(GL_POLYGON);
glColor3ub(100, 10, 10);
glVertex2i(530 - 220, 405 + 76);
glVertex2i(530 - 220, 396 + 76);
glVertex2i(512 - 220, 396 + 76);
glVertex2i(520 - 220, 405 + 76);

glEnd();
//shoe right
glBegin(GL_POLYGON);
glColor3ub(100, 10, 10);
glVertex2i(550 - 220, 405 + 76);
glVertex2i(550 - 220, 396 + 76);
glVertex2i(568 - 220, 396 + 76);
glVertex2i(560 - 220, 405 + 76);
glEnd();
```

SCREENSHOTS






BUS STOP
TIET BUS(ENC)
TIET BUS(CSE)
YAY!Now get her
to her college
Just straight ahead.


STOP
TIET
TIET BUS(CSE)
Mission Accomplished!
Parking is right ahead

REFERENCES

- Angel, E. (2008). *Interactive computer graphics: A top-down approach using OpenGL* (5th ed.). Addison-Wesley.
- Cats, O. (2021). *Public transport planning with smart card data*. CRC Press. https://doi.org/10.1201/9780429265419
- Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1994). *Computer graphics: Principles and practice* (2nd ed.). Addison-Wesley.
- Hearn, D., Baker, M. P., & Carithers, W. (2014). *Computer graphics with OpenGL* (4th ed.). Pearson.
- Kilgard, M. J. (1996). *OpenGL utility toolkit (GLUT) programming API*. Silicon Graphics.
- OpenGL. (n.d.). *OpenGL documentation*. https://www.opengl.org/documentation/
- Parent, R. (2012). *Computer animation: Algorithms and techniques* (3rd ed.). Morgan Kaufmann.
- Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL programming guide: The official guide to learning OpenGL* (8th ed.). Addison-Wesley.