*1.INTRODUCTION

Title: Traffic Simulation in C Using Graphics

In the realm of computer graphics and programming, creating visual representations of real-world scenarios provides an engaging platform for learning and experimentation. One such fascinating project is a traffic simulation, which serves not only as an educational tool but also as a prototype for understanding vehicular dynamics on roads. This project employs the Turbo C graphics library, leveraging its capabilities to create visually appealing graphics that simulate the movement of vehicles on a roadway.

The simulation includes various elements such as cars, road markings, and dynamic backgrounds that change according to user inputs. Users can interact with the simulation by controlling the traffic through simple keypress commands, allowing them to experience the effects of their decisions in a virtual environment. By modeling these interactions, the project aims to enhance understanding of programming concepts such as loops, conditional statements, and graphics manipulation in C.

Furthermore, this traffic simulation project is a gateway to explore more complex systems, such as traffic flow optimization, real-time traffic monitoring, and even autonomous vehicle simulations. The visual aspect, combined with interactive elements, makes this project not just an exercise in programming but also a fun and engaging way to appreciate the complexities of traffic systems.

This document details the implementation of the traffic simulation project, including the graphics used, the programming logic behind it, and the potential for future enhancements.

*2.KEY FEATURES :-

Key Features of the Traffic Simulation in C Using Graphics

1. **Realistic Vehicle Movement:**
   - Vehicles in the simulation can move at varying speeds, simulating real-world traffic dynamics. Users can control acceleration and deceleration to mimic driving behavior.
2. **User Interaction:**
   - The simulation allows users to control traffic flow through keyboard inputs. Users can start, stop, or change the direction of vehicles, offering an interactive experience.
3. **Dynamic Traffic Environment:**
   - The graphics display multiple vehicles moving in different lanes, with road markings and traffic signals, creating a realistic traffic scene.
4. **Collision Detection:**
   - Basic collision detection mechanisms are implemented to ensure that vehicles do not overlap or collide with each other, enhancing realism and user experience.
5. **Multiple Vehicle Types:**
   - The simulation includes different types of vehicles (e.g., cars, trucks) with varying sizes and speeds, allowing users to understand the diversity in traffic.
6. **Traffic Signals:**
   - Traffic lights change color, controlling the flow of vehicles. Users must obey these signals, simulating real traffic rules.
7. **Visual Effects:**
   - The project utilizes the Turbo C graphics library to create engaging visual effects, including road textures, background scenery, and animated vehicles.
8. **Scalability:**
   - The simulation can be expanded to include additional features such as pedestrian movement, weather effects, or

more complex road systems, providing a foundation for further development.

9. **Educational Tool:**
   - The project serves as an educational platform for understanding programming concepts, graphics manipulation, and the principles of traffic flow.
10. **Performance Metrics:**
    - The simulation can be extended to display performance metrics, such as the number of vehicles passing through an intersection or average wait times, allowing users to analyze traffic conditions.

*3.HARDWARE REQUIREMENTS :-

- **Workstations/PCs**:
  - **Processor**: Intel Core i5 or i7, 2.0 GHz or higher.
  - **RAM**: 8 GB minimum (16 GB for smoother performance).
  - **Storage**: 250 GB SSD.
  - **Display**: 19-inch or larger LCD/LED screen.
  - **Operating System**: Windows 10 or 11, Linux (Ubuntu or Fedora), or macOS.
  - **Network Connectivity**: Wired (Ethernet) or Wireless (Wi-Fi) capable.
- **Laptops/Tablets** (for mobile staff):
  - **Processor**: Intel Core i5 or i7, 1.6 GHz or higher.
  - **RAM**: 8 GB minimum.
  - **Storage**: 128 GB SSD.
  - **Display**: 13-inch or larger.
  - **Operating System**: Windows, Linux, or Android/iOS (if using tablets).

*4.SOFTWARE REQUIREMENTS :-

- **Operating System**:
  - Server: **Windows Server 2016+**, **Linux** (Ubuntu/CentOS).
  - Client: **Windows 10/11**, **Linux**, or **macOS**.
- **Database**:
  - **MySQL**, **PostgreSQL**, **SQL Server**, or **Oracle**.
- **Web Server**:
  - **Apache**, **Nginx**, or **IIS**.
- **Programming Languages**:
  - Backend: **PHP**, **Node.js**, **Python**, **Java , c , with Turbo C**
  - Frontend: **HTML5**, **CSS3**, **JavaScript** (**React.js**, **Angular.js**, or **Vue.js**).
- **Middleware** (optional):
  - **REST API** or **GraphQL** for system integrations

*5.ALGORITHM :-

Algorithm:

1. Start the program.

2. Initialize graphics mode:

   - Detect the graphics driver and mode.

   - Initialize the graphics system using `initgraph()`.

3. Set background color to black.

4. Display Instructions:

   - Display the instructions using `outtextxy()` for the user to press 'L', 'H', 'T', or 'P' and '1' to quit.

   - Display author information.

5. Loop indefinitely until a key is pressed (`while(1)` loop):

   - Draw shapes and elements on the screen:

     - Car Body:

       - Draw ellipses, lines, and mirrors for the car's front, rear, wheels, and side mirrors.

       - The shapes are created using `line()` and `ellipse()` functions.

     - Wheels:

       - Draw two sets of wheels using `ellipse()` and additional shapes for details.

6. Handle Key Press Events:

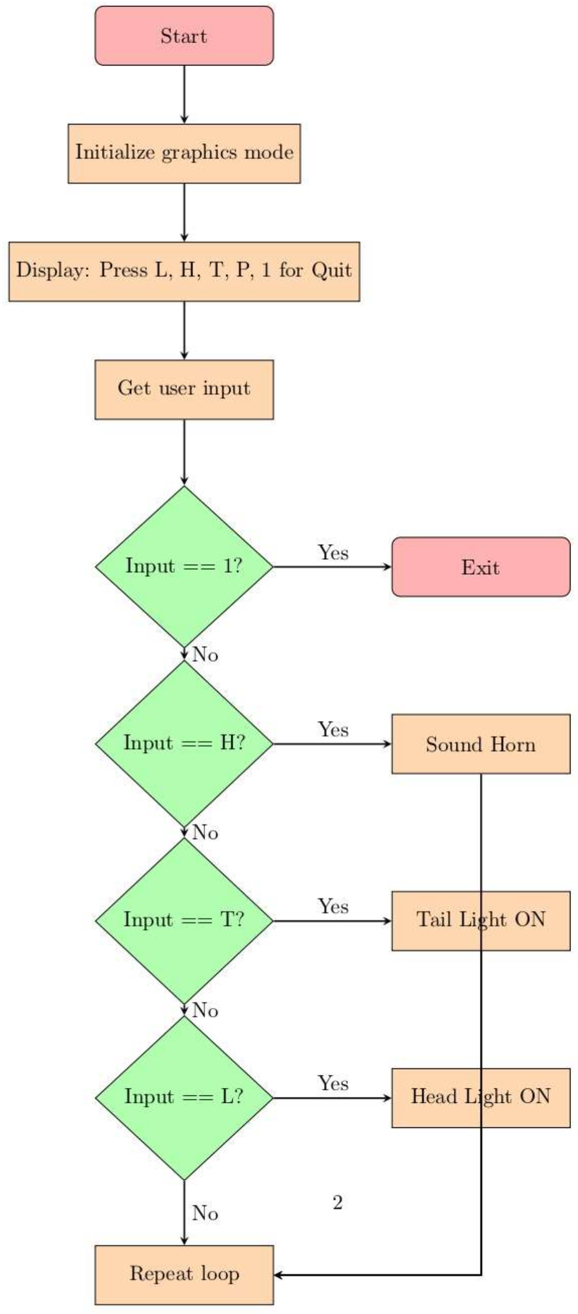
- If '1' is pressed:
  - Exit the program.
- If 'H' is pressed**:
  - Play a horn sound using `sound()` and `delay()`.
  - Stop the sound using `nosound()`.
- If 'T' is pressed:
  - Perform an animation by changing colors on various parts of the car (use `floodfill()` to fill shapes).
  - Randomize colors for an animated effect.
- If 'L' is pressed:
  - Perform a lighting effect using `floodfill()` to fill car lights and road elements with color.
  - Animate parts of the car and the road with delays for a smooth effect.
- If 'P' is pressed:
  - Perform an animation for raising and lowering a rectangle on the screen, simulating a lifting door or other moving object.
  - Display text for lights and horn.

7. Handle Final Shapes:

- After processing key events, draw a circle at `(300, 100)` as a decorative element.
- Wait for a keypress using `getch()`.

8. End the program.

*6.FLOWCHART :-



Start
Initialize graphics mode
Display: Press L, H, T, P, 1 for Quit
Get user input
Input == 1?
Yes
Exit
No
Input == H?
Yes
Sound Horn
No
Input == T?
Yes
Tail Light ON
No
Input == L?
Yes
Head Light ON
No
2
Repeat loop

*7.MAIN CODE /PROGRAM :-

```
#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

#include<process.h>


void main()

{

	int gd=DETECT,gm;

	initgraph(&gd,&gm,"c:\\turboc3\\bgi");

	int c=12;

	setbkcolor(0);

	int t;

	while(1)

	{

		settextstyle(2,0,5);

		outtextxy(100,10,"Press L,H ,T,P");

		outtextxy(100,30,"Press 1 for Quit");
outtextxy(110,50,"Designed By Karan Aher");

as:

setcolor(13);

ellipse(380,127,20,152,130,35);

/////////////////////////////rear//////////////////////////


line(490,109,560,142);

line(560,142,569,142);

line(569,142,582,102);

line(582,102,620,92);

line(593,132,617,125);


line(617,124,627,96);

line(620,92,628,97);


line(472,86,602,96);

line(501,113,575,121);

line(443,77,475,80);


line(443,77,432,93);
line(475,80,472,85);


line(593,132,593,137);

line(593,137,600,141);

line(600,141,600,185);

line(600,185,608,192);

line(608,192,608,234);

line(608,234,586,253);

line(586,253,577,248);


///     mirror

line(263,112,363,127);

line(193,160,263,112);

line(193,160,220,170);

line(220,170,280,180);

line(280,180,320,185);

line(320,185,363,127);

///////sidemirror

line(340,194,460,169);

line(460,169,519,152);
```

```
/////////////wheel1

ellipse(302,281,320,77,26,45);

ellipse(290,277,65,162,40,45);

ellipse(278,288,144,212,31,45);


/////////////wheel2

ellipse(302+260,229,328,87,26,45);

ellipse(290+280-7,277-50+2,90,162,40,45);

ellipse(278+270,288-50,144,215,27,45);

 b=0;

int v=0;


/////////

ellipse(302+250+v,227+b,295,90,29,41);

ellipse(302+234+v,231+b,245,306,50,40);


ellipse(302+248+v,229+b,0,360,21,30);


ellipse(302+247+v,229+b,0,360,8,10);
```

```
setfillstyle(6,11);


line(546+v,201+b,546+v,220+b);

line(551+v,201+b-2,551+v,220+b);


line(546+v,238+b,546+v,257+b);

line(551+v,238+b+2,551+v,257+b+2);



line(530+v,225+b,541+v,225+b);

line(530+v,230+b,541+v,230);


line(557+v,225+b,570+v,225+b);

line(557+v,230+b,570+v,230+b);




line(563+v,206+b,552+v,222+b);

line(534+v,246+b,543+v,232+b);
line(566+v,210+b,556+v,223+b);

line(536+v,250+b,544+v,238+b);


line(536+v,207+b,546+v,222+b);

line(532+v,213+b,542+v,224+b);


line(556+v,235+b,566+v,247+b);

line(551+v,237+b,563+v,253+b);


////////////////////////

v=-260;

b=56;

ellipse(302+233+v,221+b,260,60,49,51);

ellipse(302+243+v,224+b,0,360,28,35);

ellipse(300+245+v,223+b,0,360,10,12);


ellipse(285+249+v,239+b,210,260,30,33);

b=45;

v=v-4;

line(546+v,201+b,546+v,220+b+2);
line(551+v,201+b,551+v,220+b+2);

b=b+8;

line(546+v,238+b,546+v,257+b+4);

line(551+v,238+b,551+v,257+b+4);

v=v-2;

line(530+v-6,225+b,541+v,225+b);

line(530+v-6,230+b,541+v,230+b);

v=v+5;

line(557+v,225+b,570+v+3,225+b);

line(557+v-1,230+b,570+v+3,230+b);



b=b-5;

v=v-5;

line(565+v+3,206+b,552+v+4,222+b-2);

b=b+15;


line(534+v,246+b,543+v+3,232+b-5);

b=b-10;

line(566+v+7,210+b-5,556+v+4,220+b);
line(536+v-5,250+b,544+v-2,238+b-4);



line(536+v,207+b-8,545+v,222+b-5);
line(531+v,212+b-8,542+v,224+b-2);

line(556+v,235+b,566+v+3,247+b+5);
line(551+v,237+b,563+v+2,253+b+3);

///////////////lights
ellipse(199,250,144,345,18,8);
line(185,245,206,230);
ellipse(223,234,340,110,8,5);
line(230,237,217,252);
line(206,230,220,229);

/////////////////////////////
line(90,223,152,236);
line(152,236,137,254);
line(90,223,90,242);
```

```
ellipse(240,270,104,136,100,60);

ellipse(185,237,120,160,100,60);

ellipse(80,221,357,134,10,10);


line(152,236,168,228);

/////////////////////////////

line(435,116,440,171);

///////////////////////////hp

line(134,185,196,160);

line(214,212,318,185);

////////////////light


ellipse(166,247,99,330,8,8);

ellipse(171,243,310,129,7,7);

putpixel(174,250,13);

putpixel(173,251,13);

putpixel(164,239,13);

putpixel(165,238,13);
```

```
////////road////////////////////

setcolor(13);

line(1,430,639,300);

line(1,445,639,315);


line(1,210,93,194);

line(1,195,194,158);


line(520,90,639,71);

line(478,86,639,56);


int c=0;


line(10,194+c,10,208+c);

line(40,189+c,40,204+c);

line(70,183+c,70,198+c);

line(100,176+c,100,190+c);

line(130,170+c,130,177+c);

line(160,166+c,160,168+c);

line(190,160+c,190,161+c);
line(190+120,156+c-25,190+120,170+c-25);

line(190+150,156+c-30,190+150,170+c-30);

line(190+180,156+c-37,190+180,170+c-36);

line(190+210,156+c-42,190+210,170+c-42);

line(190+240,156+c-48,190+240,170+c-48);

line(190+270,156+c-55,190+270,170+c-54);

line(190+300,156+c-61,190+300,170+c-61);

 line(190+330,78+c+10,190+330,89+c+13);


line(190+360,72+c+11,190+360,85+c+13);

line(190+390,67+c+10,190+390,81+c+10);

line(190+420,62+c+8,190+420,76+c+10);

line(190+449,57+c+8,190+449,71+c+8);






////////////////road


setcolor(12);        /////////////////////////////1
```

```
line(1,310,25,306);

line(6,318,30,315);

line(1,310,6,318);

line(25,306,30,314);

int k,m;

k=13*45+19;

m=16*(-8);

                                                    //2

setcolor(12);


line(605,310-128,629,306-128);

line(610,318-128,634,315-128);

line(605,310-128,610,318-128);

line(629,306-128,634,314-128);


setcolor(12);    //////////////////////////////////3

k=45;

floodfill(527,86,13);

floodfill(587,71,13);
```

```
setfillstyle(1,14);

floodfill(64,303,12);


setfillstyle(9,4);

floodfill(302+248,230,13);

floodfill(302+248+v,230+b,13);


delay(20);

setfillstyle(1,14);


floodfill(200,250,13);


floodfill(170,250,13);

 floodfill(80,230,13);


 delay(20);

setfillstyle(1,0);


floodfill(200,250,13);
floodfill(170,250,13);

 floodfill(80,230,13);

}         }

if(t=='p')

{

int n=0;

while(!kbhit())

{

if(n<=60)

n++;

setcolor(0);

rectangle(1+1,-10,90-1,-12+n);

delay(14);

setcolor(12);

rectangle(1,-10,90,-10+n);

if(n==60)

{

outtextxy(10,10,"L-LIGHTS");

outtextxy(10,20,"H-HORN");

//outtextxy(10,30,"T-AllOY");
```

```
 delay(400);

 }

}

setcolor(0);

rectangle(1,-10,90,-10+n);

rectangle(1,-10,90,-11+n);

outtextxy(10,10,"L-LIGHTS");

outtextxy(10,20,"H-HORN");

//outtextxy(10,30,"T-AllOY");


}


}


circle(300,100,3);

nosound();

getch();

}
```

*8.OUTPUT :-

- **Welcome to the Traffic Simulation!**
    - Experience the dynamics of urban traffic management right on your screen.
- **Control Your Vehicle:**
    - Use the keyboard to accelerate, decelerate, and steer your vehicle through the simulated environment.
- **Realistic Traffic Flow:**
    - Observe vehicles moving at varying speeds, simulating real-world driving conditions.
- **Traffic Signals in Action:**
    - Pay attention to changing traffic lights, as they dictate the flow of vehicles at intersections.
- **Dynamic Road Environment:**
    - Navigate through a busy road filled with multiple vehicles and realistic road markings.
- **Collision Avoidance:**
    - Stay vigilant! The simulation includes collision detection to prevent accidents.
- **Various Vehicle Types:**
    - Encounter different vehicles, from cars to trucks, each with unique characteristics.
- **Interactive Learning Tool:**
    - Enhance your understanding of programming concepts and traffic dynamics through this engaging simulation.
- **Visual Appeal:**
    - Enjoy vibrant graphics and animated movements that bring the traffic scenario to life.
- **End of Simulation:**
    - Thank you for participating! Analyze your driving patterns and improve your skills for a safer journey.

*9.CONCLUSION :-

Conclusion of Traffic Simulation in C Using Graphics

The traffic simulation project provides a practical and engaging way to understand traffic flow dynamics and programming concepts through a visual and interactive approach. By simulating real-world scenarios with vehicles, traffic signals, and collision avoidance mechanisms, the project enhances problem-solving skills and fosters a deeper appreciation for the complexities of traffic management systems. Additionally, it serves as a valuable tool for both learning and entertainment, offering insights into how urban traffic operates in a controlled digital environment. Overall, this simulation is a stepping stone for aspiring programmers to explore graphical programming and real-time systems.