**Introduction**

This is a car game wherein the objective is to guide a car without hitting the oncoming traffic or going off-road. The implementation of said idea uses the 16-bit library offered by Turbo C++ to create the required graphics.

Without the use of more advanced environments like Open GL, we have made this game as aesthetically appealing as possible. The screen is treated as a two dimensional array and our car as well as the enemy cars are confined to move within this array.

Each iteration moves the enemy cars downwards, thus emulating motion. The player’s car can be moved about by using the arrow keys (bios.h is included to input commands). If at any time, there are two cars at the position within the array, it means that the player has hit an enemy car and he loses the game. Also, if he goes out of the array, he has crashed.

The game offers the facility of creating an account in one’s name. A player’s account has his username and also his highest score till date (which is updated after each game). The account is accessed by a user-defined password.

A leaderboard has also been incorporated as a lucrative feature. All the players who have an account are listed, in the descending order of their highest respective scores.

There is also a ‘How to Play’ page with full instructions.

Having acknowledged the numerous limitations of an antiquated graphics environment, our program has, by and by, satisfactorily worked around the associated problems.

**System Requirements**

The newer versions of the Windows Operating system don’t support a DOS environment, primarily due to the incompatibility of a 32 or 64 bit processor in working with the routines designed for 16 bit compilers/processors.

Consequently, the DOS environment is emulated using a DOS BOX.

There are two versions available: one for 32-bit systems and one for 64-bit systems. The appropriate version is to be downloaded and installed.

Turbo C++ is also to be installed.

As is evident, there are no computational or processing lower bounds as part of the system requirements as our program has a relatively low frame rate and small memory requirements.

HEADER FILES AND FUNCTIONS UNDER THEM

#include<stdio.h> :-

int printf(const char\* \_format);

int remove(const char\* \_path);

int rename(const char\* \_oldname,const char\* \_newname);

#include<dos.h> :-

void delay(unsigned \_milliseconds);

#include<process.h> :-

void exit(int\_status);

#include<time.h> :-

time\_t time(time\_t \*\_timer);

#include<conio.h> :-

int getch(void);

void gotoxy(int \_x,int \_y);

int kbhit(void);

void clrscr(void);

#include<stdlib.h> :-

void malloc(int \_size);

int rand(void);

int srand(unsigned \_seed);

#include<graphics.h> :-

void bar(int left,int top,int right,int bottom);

void outtextxy(int x,int y,char \*string);

void fillellipse(int x,int y,int xradius,int yradius);

void fillpoly(int num,int \*polypoints);

unsigned int imagesize(int left,int top,int right,int bottom);

void floodfill(int x,int y,int border);

void closegraph(void);

void setcolor(int colour);

void setfillstyle(int pattern,int colour);

void pieslice(int x,int y,int stangle,int endangle,int radius);

void putimage(int left,int top,void \*ptr,int op);

void settextstyle(int font,int direction,int charsize);

void clearviewport(void);

#include<bios.h> :-

int bioskey(int value);

#include<fstream.h> :-

stream\_object.open(“file name”,(file mode));

int eof();

istream& read((char\*)&buf,int sizeof(buf));

ofstream& write((char\*)&buf,int sizeof(buf));

FUNCTIONS

void drawPath();

void drawCar(int row,int col,int last, void \*test);

void Erasebottom(int c);

void enemy(int row,int col);

void play();

void create();

void create2();

CLASSES

Class Menu :-

Data members :

char name[20];

int pass;

int hscore;

Member functions:

void create();

int login(int temp);

void score();

int retscore();

void display();

void hiscore();

int showpass();

void sethigh();

void insert();

**Limitations**

As the code has been written using the antiquated Turbo C++ graphics environment, associated limitations on the resolution and availability of features are to be noted.

The cars in our game move in a 2-D array and this motion is not fluid. Essentially, the car is deleted and then pasted onto its new position. This lack of fluidity in transition is inherent to the way that we have designed the logic of the game.

On a related note, are the limitations on the sensation of motion: the speed of our game is modulated by a change in the frame rate of the entire screen, i.e. all elements are updated simultaneously.