```
#include <graphics.h>
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <process.h>
#include <dos.h>
#include <time.h>
#include <fstream.h>

char  *MONTHS [] =
  {
  "Jan", "Feb", "March", "April", "May", "June",
  "July", "Aug", "Sep", "Oct", "Nov", "Dec"
  };/*This string array is created to put the name of the months of the year
      corresponding to the month name.*/

struct highscores
{
    int date;
    char  month[5];
    int year;
    int stored_score;
}high_score;

class DODGER
{
    private:
    int POSX,POSY;
    int OBS_POSX_1,OBS_POSY_1;
    int OBS_POSX_2,OBS_POSY_2;
    int maxx,maxy;
    int crash;

    protected:
    int score;

    public:
    DODGER()
    {
        crash = 1;
        score = 0;
    }
    void car();
    void grass();
    void Master();
    void obstacle(int,int);
    void clearobject(int,int);
    void direction();
    int level();
};

void DODGER::Master()
{
    maxx = getmaxx();
    maxy = getmaxy();
    POSX = maxx/2;
    POSY = maxy-50;
    grass();
    car();
    /*OBSTACLE CONTROL AND DODGER MOVEMENT*/
    int i = 0;
    crash = 1;
    while(crash!=0)
    {
        OBS_POSX_1 = 190+random(maxx-370);
        OBS_POSY_1 = 30;
        while(OBS_POSY_1<=maxy)
        {
            obstacle(OBS_POSX_1,OBS_POSY_1);
            delay(level());
            clearobject(OBS_POSX_1,OBS_POSY_1);
            OBS_POSY_1+=5;
            gotoxy(10,10);
            cout<<"SCORE:"<<score;
            score++;
            switch(level())
            {
                case 10:
                gotoxy(10,8);
                cout<<"LEVEL:1";
                break;

                case 8:
                gotoxy(10,8);
                cout<<"LEVEL:2";
                break;

                case 6:
                gotoxy(10,8);
                cout<<"LEVEL:3";
                break;
            }
            while(kbhit())
            {
                direction();
            }
            if(POSX-20<=OBS_POSX_1+20&&POSX+20>=OBS_POSX_1-20&&POSY-15<=OBS_POSY_1+40&&POSY
+40>=OBS_POSY_1-15)
            {
                setcolor(RED);
                settextstyle(DEFAULT_FONT, HORIZ_DIR, 6);
                outtextxy(getmaxx()/2,50,"CRASH");
                delay(2000);
                crash = 0;
                break;
            }
        }
        i++;
    }
}

void DODGER::car()
{
    /* our polygon array */
    int poly[8];
    /*HEAD*/
    setcolor(RED);

    poly[0] = POSX-10;
    poly[1] = POSY-15;

    poly[2] = POSX+10;
    poly[3] = POSY-15;

    poly[4] = POSX+20;
    poly[5] = POSY;

    poly[6] = POSX-20;
    poly[7] = POSY;

    /* set fill pattern */
    setfillstyle(SOLID_FILL, RED);

    /* draw a filled polygon */
    fillpoly(4, poly);

    /*MAIN BODY*/

    poly[0] = POSX-20;
    poly[1] = POSY;

    poly[2] = POSX+20;
```

```
poly[3] = POSY;

poly[4] = POSX+20;
poly[5] = POSY+40;

poly[6] = POSX-20;
poly[7] = POSY+40;

/* set fill pattern */
setfillstyle(SOLID_FILL, RED);

/* draw a filled polygon */
fillpoly(4, poly);

/*WINDSCREEN*/

poly[0] = POSX-8;
poly[1] = POSY-10;

poly[2] = POSX+8;
poly[3] = POSY-10;

poly[4] = POSX+15;
poly[5] = POSY-2;

poly[6] = POSX-15;
poly[7] = POSY-2;

/* set fill pattern */
setfillstyle(SOLID_FILL, CYAN);

/* draw a filled polygon */
fillpoly(4, poly);

/*LEFT WINDOW*/

poly[0] = POSX-20;
poly[1] = POSY+10;

poly[2] = POSX-15;
poly[3] = POSY+10;

poly[4] = POSX-15;
poly[5] = POSY+30;

poly[6] = POSX-20;
poly[7] = POSY+30;

/* set fill pattern */
setcolor(LIGHTGRAY);
setfillstyle(SOLID_FILL,LIGHTGRAY);

/* draw a filled polygon */
fillpoly(4, poly);

/*RIGHT WINDOW*/

poly[0] = POSX+15;
poly[1] = POSY+10;

poly[2] = POSX+20;
poly[3] = POSY+10;

poly[4] = POSX+20;
poly[5] = POSY+30;

poly[6] = POSX+15;
poly[7] = POSY+30;

/* set fill pattern */
setcolor(LIGHTGRAY);
setfillstyle(SOLID_FILL, LIGHTGRAY);
```

```
    /* draw a filled polygon */
    fillpoly(4, poly);

    /*TAIL*/
    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL, BROWN);
    pieslice(POSX,POSY+40,0,180,10);
}

void DODGER::grass()
{
    int poly[8];
    /*LEFT GRASS*/
    poly[0] = 0;
    poly[1] = 0;

    poly[2] = 160;
    poly[3] = 0;

    poly[4] = 160;
    poly[5] = maxy;

    poly[6] = 0;
    poly[7] = maxy;

    setcolor(GREEN);
    setfillstyle(SOLID_FILL,GREEN);

    fillpoly(4,poly);

    /*RIGHT GRASS*/
    poly[0] = maxx-160;
    poly[1] = 0;

    poly[2] = maxx;
    poly[3] = 0;

    poly[4] = maxx;
    poly[5] = maxy;

    poly[6] = maxx-160;
    poly[7] = maxy;

    setcolor(GREEN);
    setfillstyle(SOLID_FILL,GREEN);

    fillpoly(4,poly);

    /*ROAD*/
    poly[0] = 160;
    poly[1] = 0;

    poly[2] = maxx-160;
    poly[3] = 0;

    poly[4] = maxx-160;
    poly[5] = maxy;

    poly[6] = 160;
    poly[7] = maxy;

    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL,LIGHTGRAY);

    fillpoly(4,poly);


    setcolor(WHITE);
    settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
    outtextxy(10,10,"CONTROLS:");
    outtextxy(10,20,"\'A\'->LEFT");
    outtextxy(10,30,"\'D\'->RIGHT");

    /*BORDER LINE*/
    setcolor(BLACK);
    line(160,0,160,maxy);
    line(maxx-160,0,maxx-160,maxy);
}

void DODGER::obstacle(int posx,int posy)
{
    /* our polygon array */
    int poly[8];
    /*HEAD*/

    poly[0] = posx-10;
    poly[1] = posy-15;

    poly[2] = posx+10;
    poly[3] = posy-15;

    poly[4] = posx+20;
    poly[5] = posy;

    poly[6] = posx-20;
    poly[7] = posy;

    setcolor(YELLOW);
    /* set fill pattern */
    setfillstyle(SOLID_FILL, YELLOW);

    /* draw a filled polygon */
    fillpoly(4, poly);

    /*MAIN BODY*/

    poly[0] = posx-20;
    poly[1] = posy;

    poly[2] = posx+20;
    poly[3] = posy;

    poly[4] = posx+20;
    poly[5] = posy+40;

    poly[6] = posx-20;
    poly[7] = posy+40;

    /* set fill pattern */
    setfillstyle(SOLID_FILL, YELLOW);

    /* draw a filled polygon */
    fillpoly(4, poly);

    /*WINDSCREEN*/

    poly[0] = posx-8;
    poly[1] = posy-10;

    poly[2] = posx+8;
    poly[3] = posy-10;

    poly[4] = posx+15;
    poly[5] = posy-2;

    poly[6] = posx-15;
    poly[7] = posy-2;

    /* set fill pattern */
    setfillstyle(SOLID_FILL, CYAN);

    /* draw a filled polygon */
    fillpoly(4, poly);
    /*LEFT WINDOW*/

    poly[0] = posx-20;
    poly[1] = posy+10;

    poly[2] = posx-15;
    poly[3] = posy+10;

    poly[4] = posx-15;
    poly[5] = posy+30;

    poly[6] = posx-20;
    poly[7] = posy+30;

    /* set fill pattern */
    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL,LIGHTGRAY);

    /* draw a filled polygon */
    fillpoly(4, poly);

    /*RIGHT WINDOW*/

    poly[0] = posx+15;
    poly[1] = posy+10;

    poly[2] = posx+20;
    poly[3] = posy+10;

    poly[4] = posx+20;
    poly[5] = posy+30;

    poly[6] = posx+15;
    poly[7] = posy+30;

    /* set fill pattern */
    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL, LIGHTGRAY);

    /* draw a filled polygon */
    fillpoly(4, poly);

    /*TAIL*/
    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL, BROWN);
    pieslice(posx,posy+40,0,180,10);
}

void DODGER::clearobject(int posx,int posy)
{
    int poly[8];

    poly[0] = posx-20;
    poly[1] = posy-20;

    poly[2] = posx+20;
    poly[3] = posy-20;

    poly[4] = posx+20;
    poly[5] = posy+40;

    poly[6] = posx-20;
    poly[7] = posy+40;

    setcolor(LIGHTGRAY);
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    fillpoly(4,poly);
}

void DODGER::direction()
{
    char dir = getch();

    switch(dir)
    {
    case 'a':
        if(POSX-15<=182)
        {}
        else
        {
            clearobject(POSX,POSY);
            POSX-=15;
            car();
        }
        break;

    case 'd':
        if(POSX+15>=maxx-182)
        {}
        else
        {
            clearobject(POSX,POSY);
            POSX+=15;
            car();
        }

        break;

    case 'e':
        exit(0);
    }
}

int DODGER::level()
{
    if(score <= 750)
    {
    return 10;  //level 1
    }
    else if(score <= 1000)
    {
    return 8;   //level 2;
    }
    else
    {
    return 6;   //level 3
    }
}

class INTRODUCTION:public DODGER
{
    public:
    char input_choice;
    char k;

    void first_page();
    void selection();
    void store_highscore();
    void display_highscore();
};

void INTRODUCTION::first_page()
{
    int i;
   /* request auto detection */
   int gdriver = DETECT, gmode, errorcode;
   int midx, midy;
   int radius = 100;

   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "C:\\turboc3\\bgi");

   /* read result of initialization */
```

```c
errorcode = graphresult();
if (errorcode != grOk)  /* an error occurred */
{
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1); /* terminate with an error code */
}

midx = getmaxx() / 2;
midy = getmaxy() / 2;

k = ' ';

while(k!='e'||input_choice!='s')
{
cleardevice();
i=-200;
setcolor(RED);
settextstyle(DEFAULT_FONT, HORIZ_DIR, 6);
outtextxy(midx-145,midy+i,"DODGER");
setcolor(WHITE);
rectangle(midx-160,midy+i-10,midx+150,midy+i+50);
settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
setcolor(BROWN);
outtextxy(midx-145,midy+i+60,"EXCESSIVE ADRENALINE!!");
settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
setcolor(GREEN);
outtextxy(midx-145,midy+i+112,"1.) PLAY GAME");
setcolor(WHITE);
rectangle(midx-160,midy+i+137,midx+150,midy+i+100);
setcolor(CYAN);
outtextxy(midx-145,midy+i+142,"2.) QUIT GAME");
setcolor(WHITE);
rectangle(midx-160,midy+i+167,midx+150,midy+i+100);

settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
outtextxy(midx-150,midy+i+225,"press \'a\' to accept input");

int y = 160;
setcolor(BLACK);
outtextxy(120,y+20,"-->");
setcolor(YELLOW);
outtextxy(120,y,"-->");
 k = ' ';

while(k!='a')
{
     k = getch();
     input_choice = k;
     switch(k)
     {
         case 'w':
         y = 160;
         setcolor(BLACK);
         outtextxy(120,y+20,"-->");
         setcolor(YELLOW);
         outtextxy(120,y,"-->");
         break;

         case 's':
         y = 180;
         setcolor(BLACK);
         outtextxy(120,y-20,"-->");
         setcolor(YELLOW);
         outtextxy(120,y,"-->");
         break;

         case 'a':
         break;

         case 'e':
            closegraph();
            exit(0);
            break;
        }

    }
    //selection();
    if(y==160)
    {
        Master();
        cleardevice();
        settextstyle(DEFAULT_FONT, HORIZ_DIR, 3);
        setcolor(GREEN);
        outtextxy(175,50,"GAME OVER!!!");
        setcolor(CYAN);
        outtextxy(160,80,"Scores (Last 5)");
        store_highscore();
        display_highscore();
        delay(2000);
        settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
        outtextxy(160,300,"Press any key to continue");
        getch();
        score = 0;
    }

    else if(y==180)
    break;
    }
   /* clean up */
   closegraph();
}

void INTRODUCTION::selection()
{
    if(input_choice=='w')
    {
        Master();
        cleardevice();
        settextstyle(DEFAULT_FONT, HORIZ_DIR, 3);
        setcolor(GREEN);
        getch();
    }
}

void INTRODUCTION::store_highscore()
{
    fstream highscore_file("HIGHSCORE.DAT",ios::binary|ios::in|ios::out|ios::ate);
    time_t      rawtime;
    struct tm* timeinfo;
    time( &rawtime );
    timeinfo = localtime( &rawtime );
    high_score.date = timeinfo->tm_mday;
    strcpy(high_score.month,MONTHS[ timeinfo->tm_mon ]);
    high_score.year = timeinfo->tm_year + 1900;
    high_score.stored_score = score;
    highscore_file.write((char *)&high_score,sizeof(highscores));
    highscore_file.close();
}

void INTRODUCTION::display_highscore()
{
    gotoxy(1,11);
    fstream highscore_file("HIGHSCORE.DAT",ios::binary|ios::in|ios::out|ios::ate);
    textcolor(BLACK);
    textcolor(WHITE);
    highscore_file.seekg(highscore_file.tellg()-5*sizeof(highscores));
    while(!highscore_file.eof())
    {
        highscore_file.read((char *)&high_score,sizeof(highscores));
        cout<<high_score.date<<high_score.month<<high_score.year<<'\t'<<high_score.
stored_score<<endl;
    }
    highscore_file.close();
}

void main()
{
    INTRODUCTION dodger;
    high_score.date = 0;
    strcpy(high_score.month,"");
    high_score.year = 0;
    high_score.stored_score = 0;
    dodger.first_page();
}
```