VIVEKANAND EDUCATION SOCIETY'S
INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

COMPUTER GRAPHICS (CG)

Mini Project Report

On

CAR DASH (A CAR GAME)

Submitted in partial fulfilment of the requirements of
Second Year Computer Engineering

By
Kedar Kharde – D7B – 29
Parthesh Pawar – D7B – 47

Supervisor:
Mrs. Pallavi Saindane

DEPARTMENT OF COMPUTER ENGINEERING V.E.S
INSTITUTE OF TECHNOLOGY

2019-20

Abstract:

Graphic programming in C is used for drawing various geometrical shapes, colouring an object with different colours, patterns and simple animation programs.

Car dash is a (car) game implemented using a graphics library in C. A car dash has a simple gameplay in which the user has to avoid the obstacles by jumping over it.

With each successful jump score is incremented. After a successful jump a new obstacle arrives and the player has to avoid it for continuing the game. For an unsuccessful jump game gets over and score is displayed. Player by pressing 'X' can exit the game in between the gameplay.

Table of Contents:

Introduction :

Car dash is a (car) game implemented using graphics library in C. Objects such as car and obstacles are drawn using functions in graphics.h header file.

Control guide is shown on the screen. By pressing 'space' the player can make a jump and by pressing 'X' player can quit the game. Initially score is 0 after successful jump over obstacle score is updated on screen. With each unsuccessful jump game gets over, displaying the last updated score. New obstacles arrive after a successful jump and the player has to avoid it for continuing the game.
Game starts by displaying the project name and visuals of the game. This is achieved by calling startpage() function in main function.

After displaying the starting page a while(1) loop starts executing in the main function. The loop calls the car() function with every iteration, the loop breaks when the player exits the game by pressing 'X' or when the player makes an unsuccessful jump and the game gets over. In car function obstacle is translated horizontally over the road line. If left x coordinate of obstacle and front end of car have same x value it is an unsuccessful jump and game is over.

For checking successful jumps there are 2 variables obsline and jumpline. Obsline is initialized with a y-coordinate of the top edge of the obstacle. Jumpline is initialized with y-coordinate at which car touches the road. When player presses ‘SPACE’ for jump car is translated in y-axis. Also the jumpline is translated in y axis. If car is on road jumpline is greater than obsline and when jump is executed jumpline is less than obsline so car can easily cross obstacle. If jump is successful score is incremented and updated on screen.

Requirement Analysis:

- Software : Turbo C
- Hardware : RAM should be of size 2GB or higher.

Implementation:

1. Starting Page

Explanation:

Startpage() function is first executed and will appear before starting the actual game. It displays name of the game and gives idea about visuals of a game.

- settextstyle() function:

  Syntax: void settextstyle(int font , int direction , int font_size);

  The header file graphics.h contains settextstyle() function which is used to change the way in which text appears. Using it we can modify the size of text, change direction of text and change the font of text. Direction=0 for horizontal direction.

- setfillstyle() and floodfill() function:

  Syntax: void setfillstyle(int pattern, int color);
  void floodfill(int x, int y, int border_color);

  The header file graphics.h contains setfillstyle() function which sets the current fill pattern and fill color. floodfill() function is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area.

CODE:

```
void startpage()
{
  cleardevice();
  settextstyle(8,0,2);
  outtextxy(10,20,"1.Name : Kedar Kharde , Roll No : 29");
  outtextxy(10,60,"2.Name : Parthesh Pawar , Roll No : 47");
  settextstyle(8,0,5);
  outtextxy(250,(ymax/2),"ROAD DASH");
  line(0,(ymax/2)+60,xmax,(ymax/2)+60);

  /*TYRE*/
  setfillstyle(SOLID_FILL,RED);
  circle(30,(ymax/2)-i+40,20);
  circle(90,(ymax/2)+40-i,20);
  floodfill(30,(ymax/2)-i+40,WHITE);
  floodfill(90,(ymax/2)-i+40,WHITE);

  /*CAR BODY*/
```

```
  setfillstyle(SOLID_FILL,YELLOW);
  rectangle(0,ymax/2-i,120,(ymax/2)+20-i);
  floodfill(5,(ymax/2)+5,WHITE);

  /*WINDOWS*/
  setfillstyle(SOLID_FILL,GREEN);
  line(60,ymax/2-i,60,(ymax/2)-30-i);
  line(20,(ymax/2)-30-i,100,(ymax/2)-30-i);
  line(20,(ymax/2)-30-i,0,ymax/2-i);
  line(100,(ymax/2)-30-i,120,ymax/2-i);
  floodfill(30,(ymax/2)-5,WHITE);
  floodfill(75,(ymax/2)-15,WHITE);

  delay(5000);
  cleardevice();

}
```

2. Implementation of Game:

Explanation:

- kbhit() function is present in conio.h and used to determine if a key has been pressed or not. To use kbhit function in your program you should include the header file "conio.h". If a key has been pressed then it returns a non-zero value otherwise returns zero.

- Car function starts game execution. Game will terminate either if we quit the game by pressing 'X' on keyboard or if we run into obstacle and game gets over.

- **STEP 1**:
  First for loop iterates over j=0 to j=xmax which is used for translating obstacle horizontally, where xmax contains maximum x-coordinate for current graphics mode and driver.
  kbhit() function check whether player has pressed any keyboard input either for jumping or for quitting the game. If player pressed 'SPACE', jump variable is set to 1 which is initially 0. If 'X' is pressed car function returns 2 to main function and execution of code stops. An obstacle is drawn by rectangle function. Obsline variable is initialize to y-coordinate of top edge of obstacle. A global variable 'rectx' is initialize to left x coordinate of obstacle, which is decremented with every iteration for translating obstacle.

- **STEP 2**:
  Another condition check whether jump is equal to 1, i.e. it check whether 'SPACE' is pressed. If that is the case then 'i' which is global variable is set to 60 otherwise zero. While drawing car 'i' is subtracted for every iteration, if 'SPACE' is pressed then 'i' is 60 and car is translated in y-axis which simulates jump.

- **STEP 3**:
  After setting 'i' jumpline variable is initialize to y-coordinate of intersection point of car and road. Now when rectx is in range from 120 to 130, it checks whether jumpline is less than obsline if this condition satisfies then car can successfully pass the obstacle. When jump is successful car remains in jump postion and obstacle is translated along road which simulates jumping action of car. After each successful jump score is incremented and updated on screen.

- **STEP 4:**
  If rectx is not in range from 120 to 130 then car is drawn and obstacle is translated. Car is coloured using floodfill() and setfillstyle() functions. If ‘i’ is equal to 60 then seed point is changed accordingly.

- **STEP 5:**
  At end resetting ‘jump’, ‘jumpline ’and ‘i’ variables i.e. setting ‘jump’ and ‘i’ to zero and ‘jumpline’ to y-coordinate of intersection point of car and road.

CODE:

```
int car()
{
  settextstyle(0,0,2);
  for(j=0;j<xmax;j++)
  {
 //STEP 1
    if(kbhit())
    {
          ch=getch();
          if(ch==' ')
          {
           jump=1;
          }
    else if(ch=='x')
          {
          return 2;
          }
    }
     outtextxy(0,30,"Enter X to Quit");
     outtextxy(0,50,"Enter space to Jump");
     sprintf(str,"Score: %d",score);
     outtextxy(0,70,str);
     rectx=xmax-200-j; // x coordinate of obstacle
     line(0,(ymax/2)+60,xmax,(ymax/2)+60); //Road
     /*OBSTACLE*/
     setfillstyle(SOLID_FILL,BROWN);
     rectangle(rectx,(ymax/2)+40,(xmax-200)+30-j,(ymax/2)+60);
     floodfill(rectx+10,(ymax/2)+50,WHITE);

     obsline=(ymax/2)+40; //comparing for jump, top side of obstacle
    //STEP 2
    if(jump==1)
    {
     i=60; //Jump by translating in Y direction
    }
    else
    {
     i=0;
    }
    //STEP 3
    jumpline=(ymax/2)+60-i; //comparing to obsline for successful jump
```

```
if(rectx>=120 && rectx<=130) //if jump when obstacle is in given range its successfull jump
{
      if(rectx==120 && jumpline>obsline) //For unsuccessful jump
      {
        return 3;
      }

      if(jumpline<obsline)      //For successful jump
      {
          while(rectx!=10)     //Till obstacle is overcome
          {
                  if(rectx==15) //score increment as obstacle is overcame
                  {
                    k=0;
                    score++;
                    i=0;
                    return 0;
                  }
         //getting x-coordinates before jump
         jumpx1=xmax-200-j;
         jumpx2=(xmax-200)+30-j; // only x coordinate of obstacle changes when jump

          line(0,(ymax/2)+60,xmax,(ymax/2)+60); //Road
           /*TYRE*/
           setfillstyle(SOLID_FILL,RED);
           circle(30,(ymax/2)-i+40,20);
           circle(90,(ymax/2)+40-i,20);

           floodfill(30,(ymax/2)-20,WHITE);
           floodfill(90,(ymax/2)-20,WHITE);
          /*CAR BODY*/
          setfillstyle(SOLID_FILL,YELLOW);
          rectangle(0,ymax/2-i,120,(ymax/2)+20-i);

          floodfill(5,(ymax/2)-i+15,WHITE);
         /*WINDOWS*/
         setfillstyle(SOLID_FILL,GREEN);
         line(60,ymax/2-i,60,(ymax/2)-30-i);
         line(20,(ymax/2)-30-i,100,(ymax/2)-30-i);
         line(20,(ymax/2)-30-i,0,ymax/2-i);
         line(100,(ymax/2)-30-i,120,ymax/2-i);

         floodfill(30,(ymax/2)-70,WHITE);
         floodfill(75,(ymax/2)-70,WHITE);
         /*OBSTACLE*/
         setfillstyle(SOLID_FILL,BROWN);
         rectangle(jumpx1-k,(ymax/2)+40,jumpx2-k,(ymax/2)+60);

         floodfill(rectx+10,(ymax/2)+50,WHITE);
         /*text*/
         outtextxy(0,30,"Enter X to Quit");
         outtextxy(0,50,"Enter space to Jump");
         sprintf(str,"Score: %d",score);
         outtextxy(0,70,str);
```

```
            k++;
            rectx--;      //continued after successful jump
            delay(10);
            cleardevice();
          }

        }
}
            //STEP 4
            /*TYRE*/
             setfillstyle(SOLID_FILL,RED);
             circle(30,(ymax/2)-i+40,20);
             circle(90,(ymax/2)+40-i,20);
            if(i==60) //If jump then filling coordinates also changes
            {
                  floodfill(30,(ymax/2)-20,WHITE);
                  floodfill(90,(ymax/2)-20,WHITE);
            }
            else
            {
                 floodfill(30,(ymax/2)-i+40,WHITE);
                 floodfill(90,(ymax/2)-i+40,WHITE);
            }

            /*CAR BODY*/
            setfillstyle(SOLID_FILL,YELLOW);
            rectangle(0,ymax/2-i,120,(ymax/2)+20-i);
            if(i==60)
            {
                floodfill(5,(ymax/2)-i+15,WHITE);
            }
            else
            {
                 floodfill(5,(ymax/2)+5,WHITE);
            }

           /*WINDOWS*/
           setfillstyle(SOLID_FILL,GREEN);
           line(60,ymax/2-i,60,(ymax/2)-30-i);
           line(20,(ymax/2)-30-i,100,(ymax/2)-30-i);

           line(20,(ymax/2)-30-i,0,ymax/2-i);
           line(100,(ymax/2)-30-i,120,ymax/2-i);
           if(i==60)
           {
           floodfill(30,(ymax/2)-70,WHITE);
           floodfill(75,(ymax/2)-70,WHITE);
           }
           else
           {
            floodfill(30,(ymax/2)-5,WHITE);
            floodfill(75,(ymax/2)-15,WHITE);
           }
```

```
//STEP 5
        if(i==60) //Resetting jumpline , jump and i
        {
            jumpline=(ymax/2)+60;
            i=0;
            jump=0;
        }

        delay(25);
        cleardevice();
 }
        return 0; //returns 0 so countinue in while loop
}
```

3. Main function:

Explanation:

- initgraph() function:

Syntax: void initgraph(int *graphicsDriver, int *graphicsMode, char*driverDirectoryPath);

Initgraph() method of graphics.h library initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode.

Graphics driver is a pointer to an integer specifying the graphics driver to be used. It tells the compiler that what graphics driver to use or to automatically detect the drive. In our program we have use DETECT macro of graphics.h library that instruct compiler for auto detection of graphics driver.
Graphics mode is a pointer to an integer that specifies the graphics mode to be used. If gdriver is set to DETECT, then initgraph sets gmode to the highest resolution available for the detected driver.

driverDirectoryPath specifies the directory path where graphics driver files (BGI files) are l ocated. If directory path is not provided, then it will search for driver files in current working directory directory.

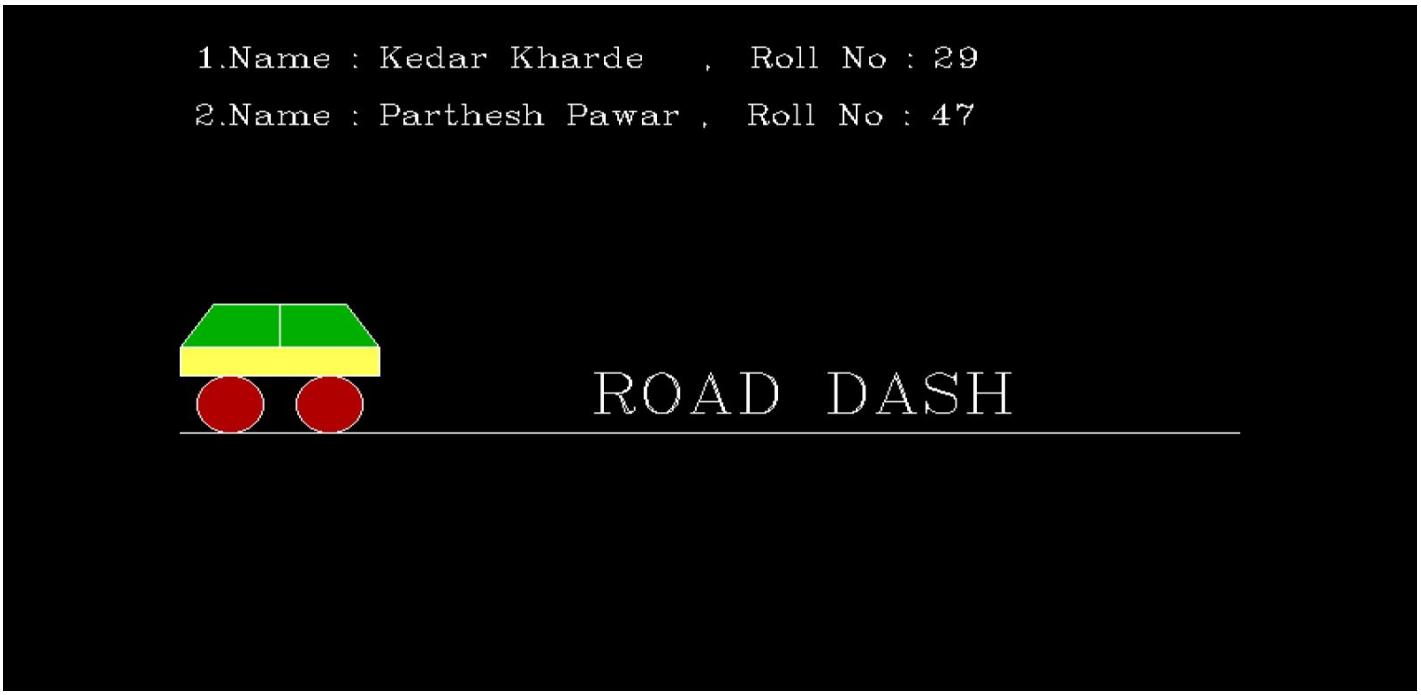
- After initializing graphics system startpage() function is called which displays starting page.
- After displaying starting page while(1) loop starts to execute which calls car() function, if car function returns 2 i.e. 'X' key is pressed and game is quit. If function returns 3 then due unsuccessful jump function is returned game get over.
- If function do not return 2 or 3 then loop continues to execute and calls car() function till one of above two cases occur.
- After while(1) loop closegraph() function closes the graphics mode.

CODE:

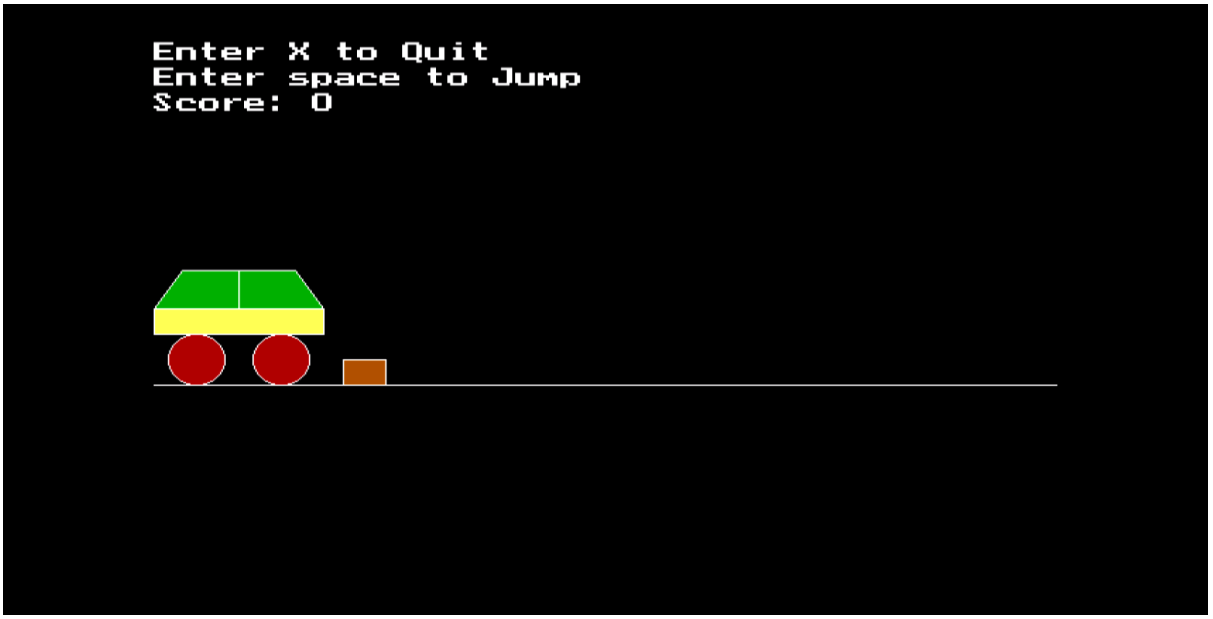
```
void main()
{

/* request auto detection*/
 int gdriver= DETECT , gmode , errorcode;

 /* intialize local variables and graphics*/
 initgraph(&gdriver , &gmode ,"C:\\TURBOC3\\BGI");
 /* read result of intialization*/
 errorcode=graphresult();

 /* an error has occured*/
 if(errorcode!=grOk)
 {printf("Graphics error: %s",grapherrormsg(errorcode));
  printf("Enter any key to halt:");
  getch();
  exit(1);
 }

  xmax=getmaxx();
  ymax=getmaxy();
  startpage(); //starting page

 while(1)
 {
     para=car();
     if(para==2) //Quit page
     {
     settextstyle(8,0,5);
     setcolor(YELLOW);
     outtextxy(300,200,"End");
     delay(1000);
     cleardevice();
     break;
     }

     else if(para==3) //Game over page
     {
     cleardevice();
     settextstyle(8,0,2);
     setcolor(RED);
     outtextxy(290,200,"GAME OVER");
     setcolor(YELLOW);
     outtextxy(300,250,str);
     delay(10000);
     cleardevice();
     break;
     }
 }
 closegraph();
}
```

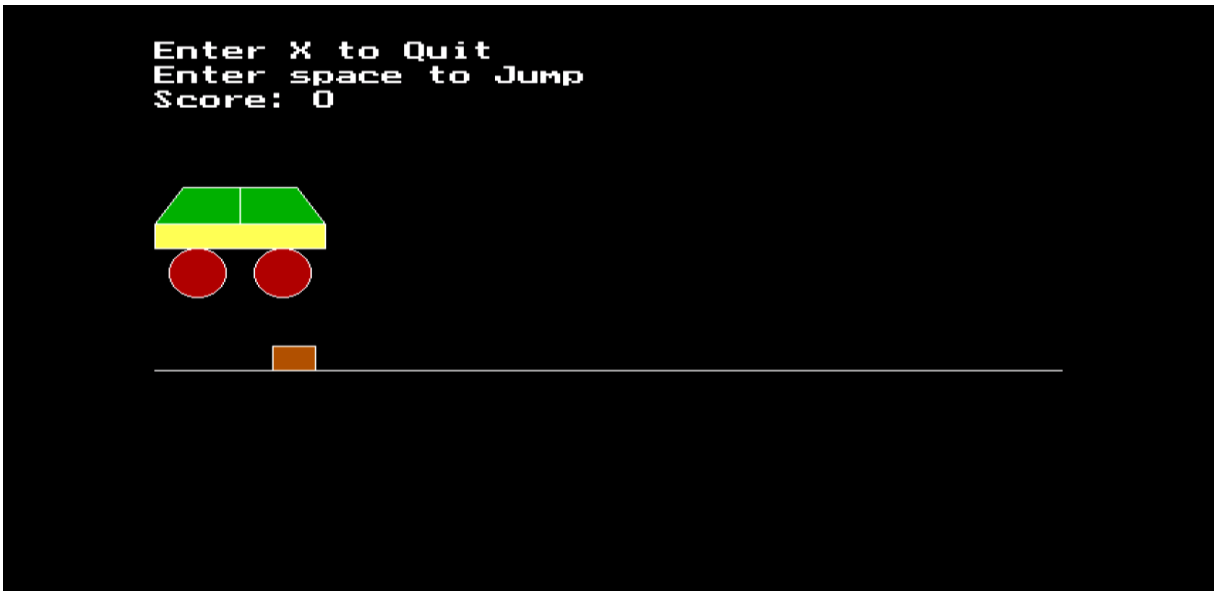
Output (SNAPSHOTS):

Start Page :

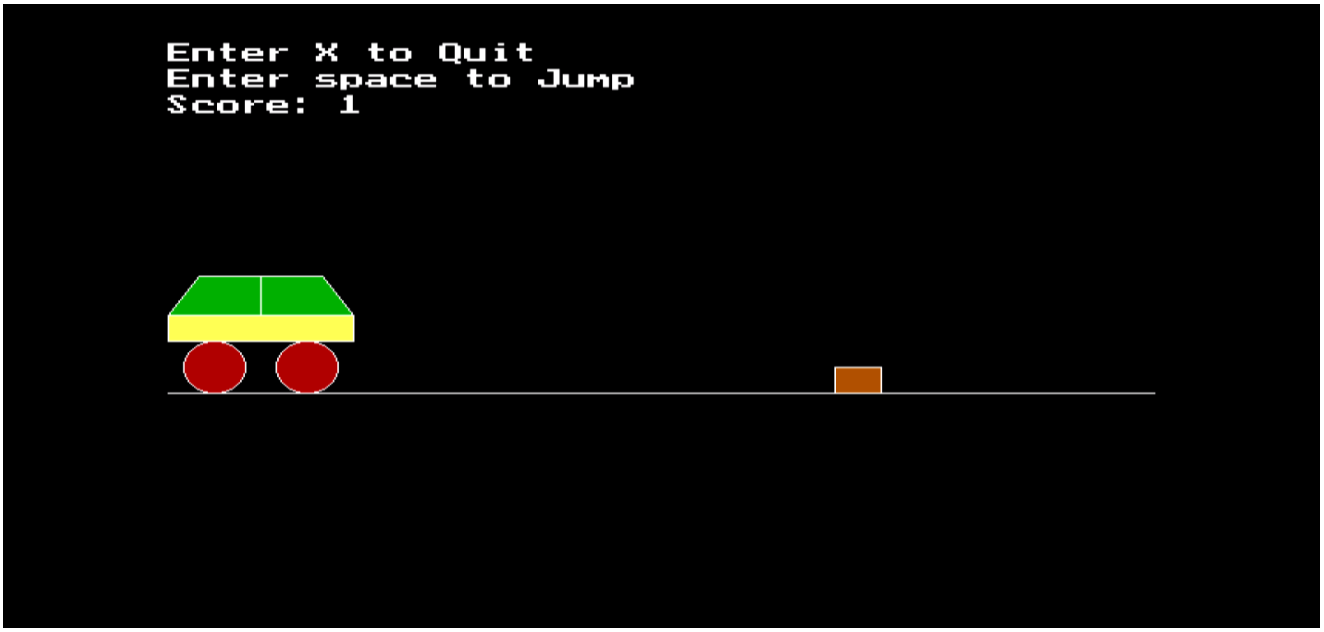


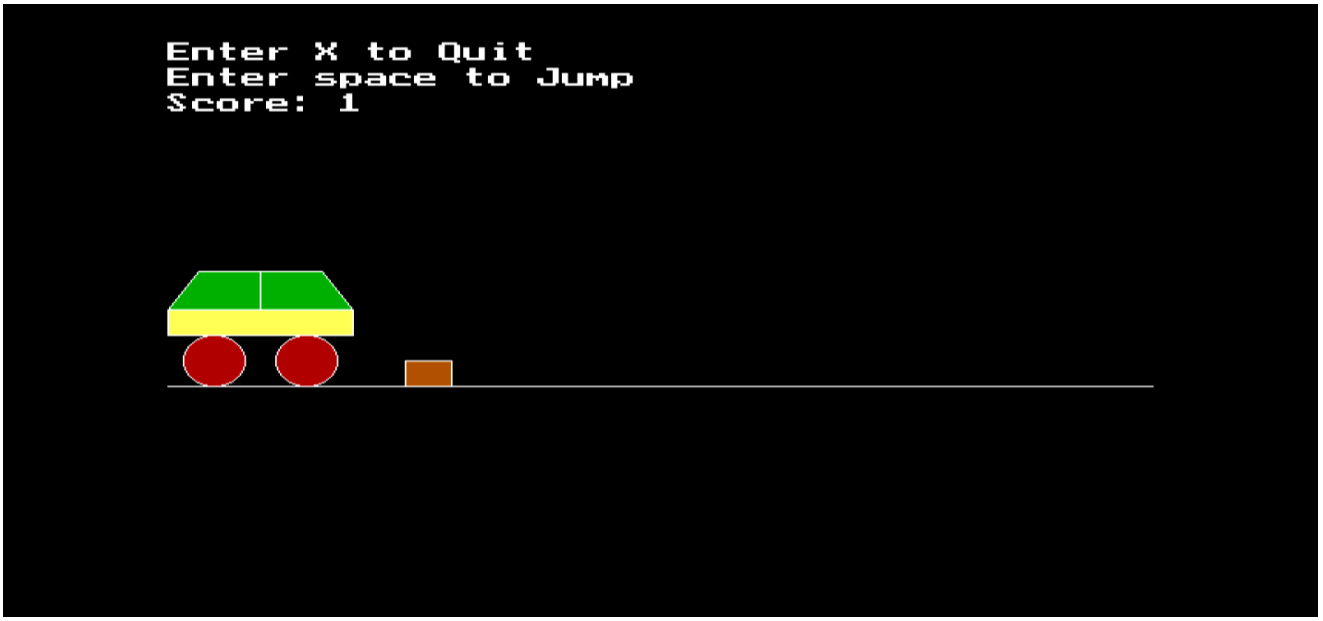
Game Starts :

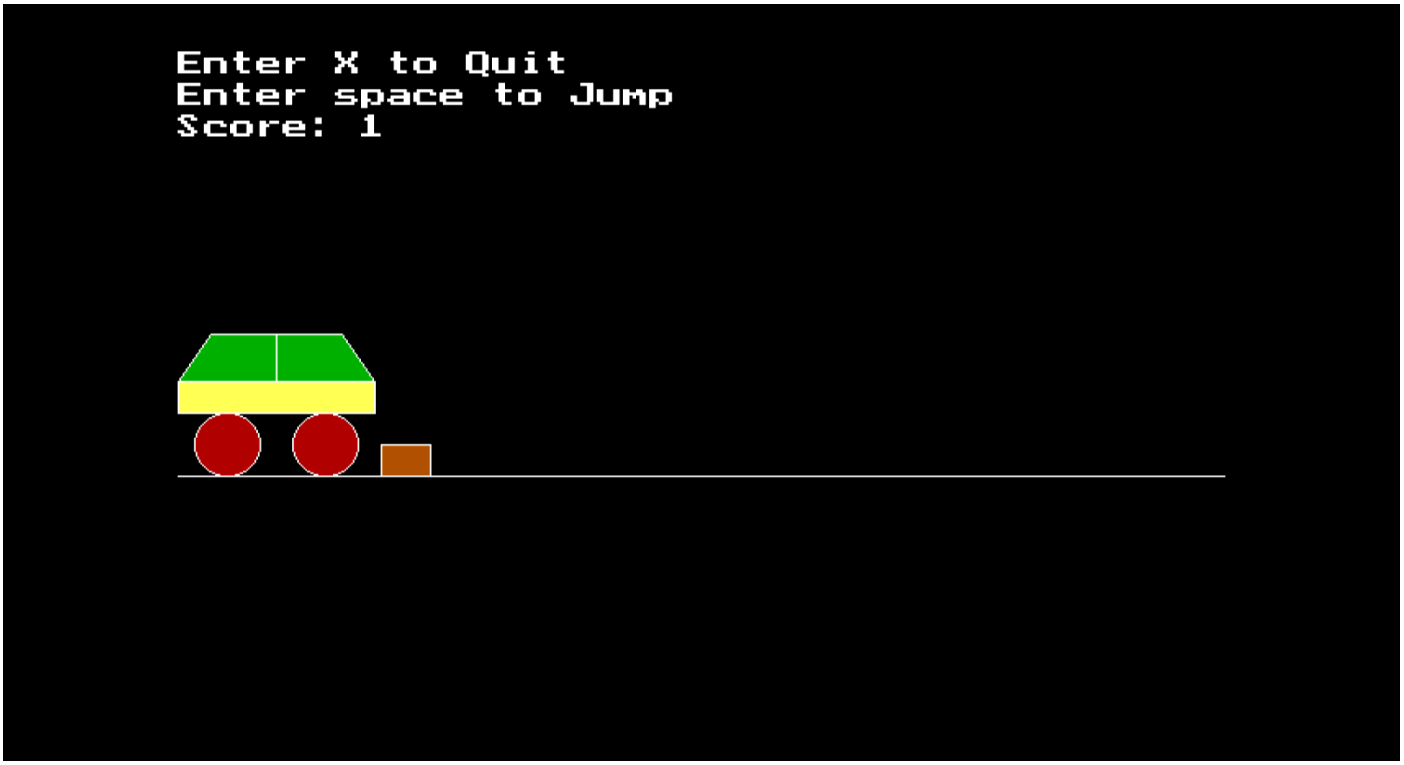

Enter X to Quit
Enter space to Jump
Score: 0

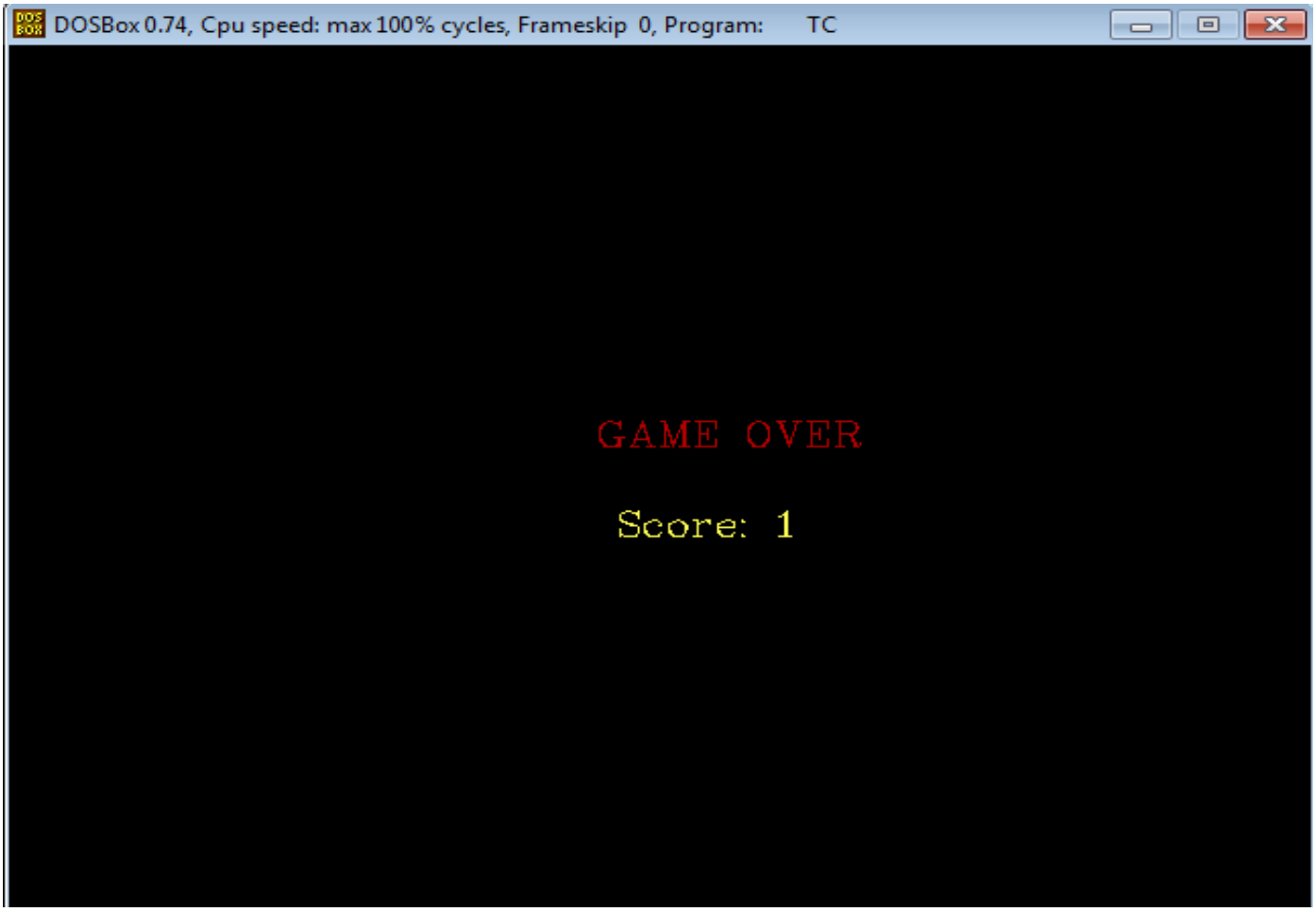

Enter X to Quit
Enter space to Jump
Score: 0


Enter X to Quit
Enter space to Jump
Score: 0

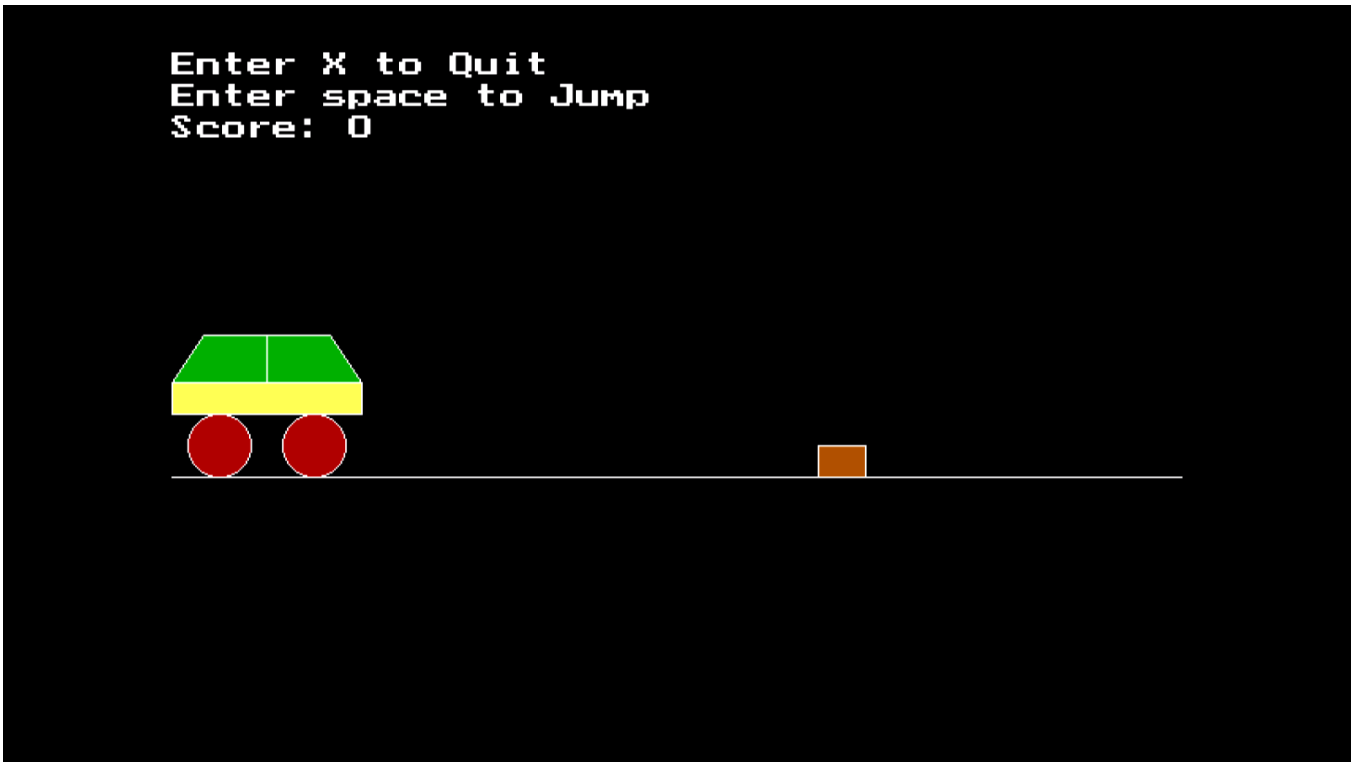

Enter X to Quit
Enter space to Jump
Score: 1

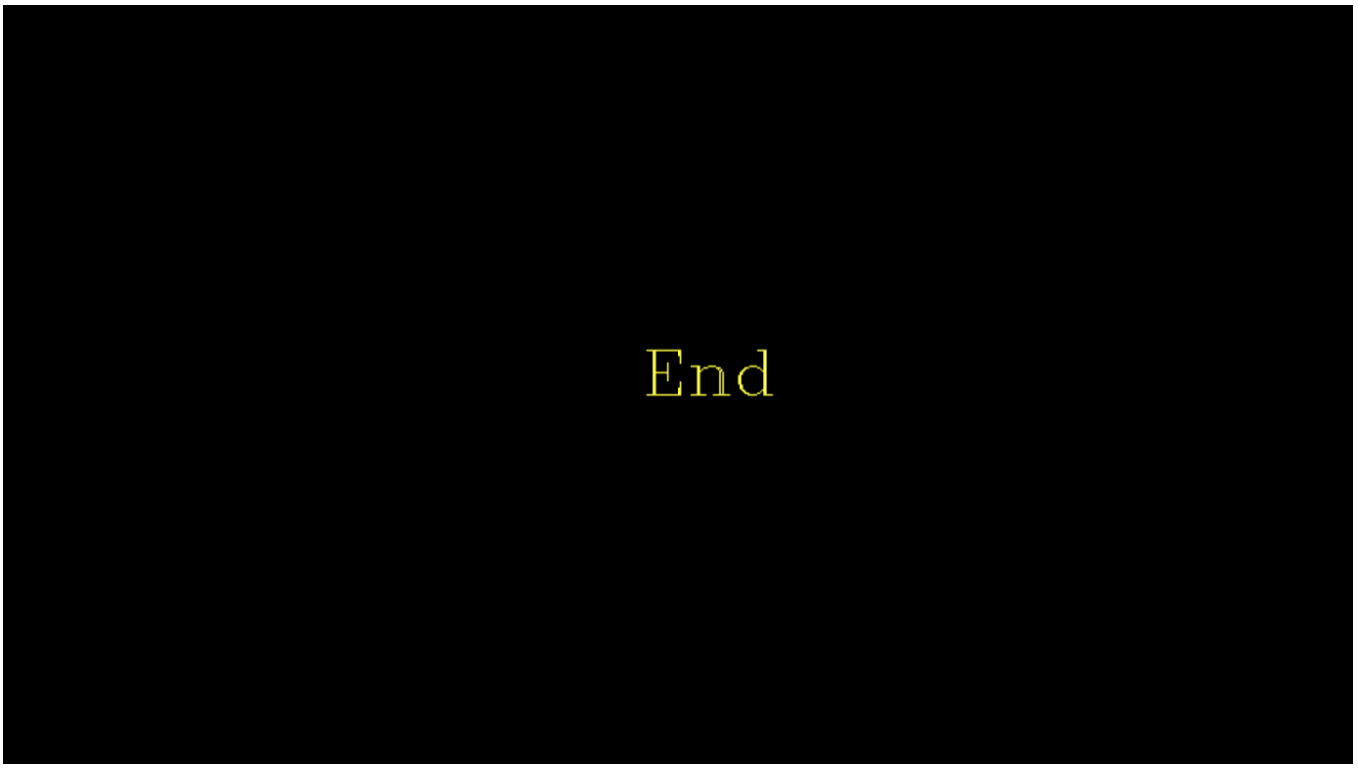

Enter X to Quit
Enter space to Jump
Score: 1


Enter X to Quit
Enter space to Jump
Score: 1


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
GAME OVER
Score: 1

Quitting game by pressing X :





Conclusion:

Hence, with help of graphics.h library we have implemented a car game in Turbo c. In this game we used graphics.h to draw a car and obstacle and using floodfill() function we filled the colour. We have set the specific controls for playing game such as ‘SPACE’ key is used for jumping and by pressing ‘X’ key you exit the gameplay in between. Based on input given by player we decide the position of a car and change the output. The gameplay is easy to understand, a player have to avoid the obstacle. After each successful jump over obstacle score is incremented and for unsuccessful jump game gets over and score is displayed. A player has to avoid the each obstacle for continuing the game. A player has control to exit the game in between the gameplay.