Jatiya Kabi Kazi Nazrul Islam University

Trisal,Mymensingh-2220,Bangladesh

Dept. of Computer Science & Engineering

Course Name : Computer Graphics

Course Code : CSE -403

Report on : A car game using Graphics in C.

Submitted To :

DR. A. H. M. KAMAL

Professor

Dept. of CSE, JKKNIU

Submitted By :

ISRAQ AHMED ANIK

Roll No : 15102013

Reg : 4088

Table of Contents

Abstract

Graphics are visual images or designs on some surface, such as a wall, canvas, screen, paper, or stone to inform, illustrate, or entertain. Computer graphics has enabled the visualization of different object across different plains called displays using Pixels. In this project the use of computer graphics are explored, and a simple game mechanic is implemented using the *grpahics.h* header file in C programming language. Car games has existed since the beginning of computer gaming, and it's nothing new but visualization of some basic elements of graphics.

CHAPTER 1: Introduction

In this project we are going to design a basic car game ,the car can be controlled by the arrow keys , the car will be controllable within a confined box, where an object will travel from the top to the bottom of the box, it will appear each time from a different point to increase the difficulty of the game .

The game should have different levels , and as score increases so does the level , changing the color of the track and speeding up the movement of the car after each level .

CHAPTER 2: Background

Graphics programming in C used to drawing various geometrical shapes(rectangle, circle, eclipse etc), use of mathematical function in drawing curves, coloring an object with different colors and patterns and simple animation programs like jumping ball and moving cars. Using the existing functions in the *grpahics.h* header like *circle(), rectangle()* we are going to simulate a moving car on a track and colliding objects.

The program initializes the graphics system by loading the passed graphics driver then changing the system into graphics mode. It also resets or initializes all graphics settings like color, palette, current position etc, to their default values. The graphics drivers on the computer is initialized using *initgraph* method of *graphics.h* library.

grpahics.h header allows for 16 colors to use ranging form BLACK to WHITE . Each of them has a code (integer value) assigned to them.

People throughout the years tried different ways to simulate moving object like cars using computer graphics. This method only uses the resources of the *grpahics.h* header file. To implement car movement, we are going to use the arrow keys to move the car from left to right and up to down.

CHAPTER 3: Methodology

In this portion of the report we are going to discuss about the construction and the design of the code that resulted in the game. The whole game is reliant on math and simple object co-ordinate logics.

3.1: Designing the Road

To design the road, I have implemented a simple rectangle using the simple rectangle command to define the area of the game and two lines to define the road.

```
rectangle(50,50,400,400);
line(150,50,150,400);
line(300,50,300,400);
```

3.2: Simulating Road Movement

To simulate the road moving I have drawn two lines one big and one small, and take a variable **down** that will increase exponentially and that variable to the **y** value of those line to simulate that the line is moving downwards.

```
//big lines
 line(x+k,down,150,down);
 line(300,down,y-k,down);
 down=down+20;
 //small lines
 line(x+k+50,down,150,down);
 line(300,down,y-k-50,down);
```

3.3: Defining the Car

For the car I have previously defined four variable x1,x2,y1,y2 as the parameter of the car , and use the command *rectangle()* to draw the car.

```
rectangle(x1,y1,x2,y2);
```

3.4: Defining the Obstacle

For this I wanted a simple obstacle going from top to the bottom of the road , and it appeared at any random point of the street , so we decided on a circle which's x value is random and y value is increasing by a constant . If the y value is less than or equal to the end point of the road it's reset to the starting point of the road .

```
int cx= (rand() % (290 - 160 + 1)) + 160;
int cy= 60;

if(cy<380){
        circle(cx,cy,10);
        cy=cy+15;
}
if(cy>380){
        cy=60;
         cx= (rand() % (290 - 160 + 1)) + 160;
}
```

Here 380 is the end y value of the road.

3.5: Implementing Car movement

To implement car movement I am going to use the arrow keys . In a keyboard each keys has different ascii values , and the arrows are :

Character	ASCII
Up Arrow	72
Down Arrow	80
Left Arrow	75
Right Arrow	77

So , if we use the *kbhit()* function we can check for a key press, and if those key press are any of the arrow keys we do translation according to the direction of the arrow .

Up Arrow	Down Arrow	Left Arrow	Right Arrow

`if (key==72){`
`y1=y1-20;`
`y2=y2-20;`
`if(y1<=50){`
`y1=50;`
`y2=90;`
`}`
`}`
	`if(key==80){`
	`y1=y1+20;`
	`y2=y2+20;`
	`if(y1>=360){`
	`y1=360;`
	`y2=400;`
	`}`
	`}`
		`if(key==75){//left`
		`x1=x1-20;`
		`x2=x2-20;`
		`if(x1<=150){`
		`x1=150;`
		`x2=180;`
		`}`
		`}`
			`if(key==77){`
			`x1=x1+20;`
			`x2=x2+20;`
			`if(x1>=270){`
			`x1=270;`
			`x2=300;`
			`}`
			`}`

3.6: Implementing the Collision Mechanic

The car and the obstacle will collide only if the car touches the object circle , or the circle resides inside the car rectangle parameter , so we can implement a very simple logic:

```
if((cx+10>=x1)&&(cy+10>=y1)&&(cx-10<=x2)&&(cy-10<=y2)){
```

3.7: Implementing the Score, Level, High score and Death count

For score, level and death count I have used three variables called **score**, **level**, **death** respectively. Whenever the road progresses the score increase by one, the level increases by one(1) each time score increases by 20 . And once the car and obstacle collide with each-other the death count increases and checks for High score, if high sore is smaller than current score than it saves current score as High score.

3.8: Implementing Color Changing and Speeding Up Each Level

So we know that The *gaphics.h* header supports 16 different color, so we can implement a system that changes color each time and speeds up the player level goes up. To do that we take a variable **i** for color and set it on a loop for each level change from **1** to **15** color value , excluding **0** as 0 is the color value for the color Black .
For speed we are going to take a variable **del** which will work as delay , and as the level progresses the delay decreases , resulting in the simulation of more speed in game . The exact part of the code:

```
if(score%20==1){
 level++;
 del=del-30;
 if(del<40)del=40; //for delay

 //for color changing
 if(i>14)i=1;
 i++;
 setcolor(i);
 }
```

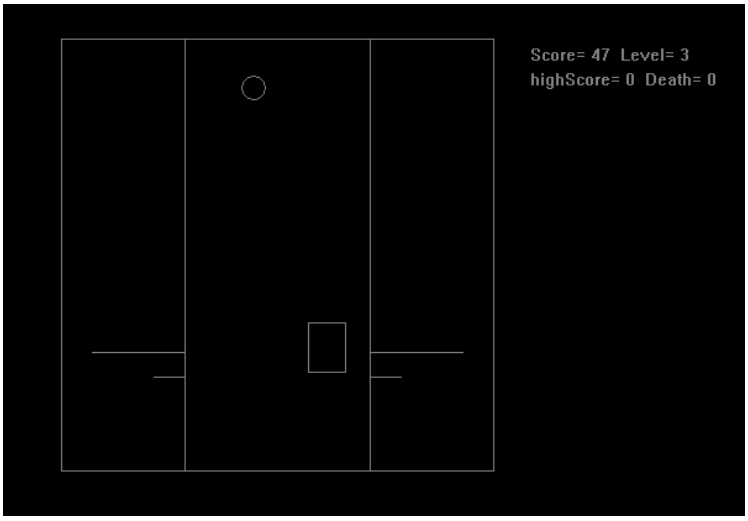
CHAPTER 4: Code

```
#include<graphics.h>
#include<process.h>
#include<dos.h>
#include<iostream>
#include<stdio.h>

using namespace std;

int main()
{
   int hscore=0;
   int death=0;
   start:
   int i=1;
   int gdriver = DETECT, gmode ,errorcode;
   initgraph (&gdriver, &gmode, "");
   int down=50;
   int x1=200,y1=280,x2=230,y2=320;//car parameters
   int score=1;
   int level=1;
   char msg[128];
   char msg2[128];
   int cx= (rand() % (290 - 160 + 1)) + 160;
   int cy= 60;
   int del=250;//for delay
   while(1)
   {
      rectangle(50,50,400,400);
      line(150,50,150,400);
      line(300,50,300,400);
      sprintf(msg, "Score= %d  Level= %d", score,level);
      outtextxy(430, 55, msg);
      sprintf(msg2, "highScore= %d  Death= %d", hscore,death);
      outtextxy(430, 75, msg2);
      char key;

      //moving car
      if(kbhit()){
         key=getch();
         if(key==77){//right
            x1=x1+20;
            x2=x2+20;
            if(x1>=270){
               x1=270;
               x2=300;
            }
         }
         if(key==75){//left
            x1=x1-20;
            x2=x2-20;
            if(x1<=150){
               x1=150;
               x2=180;
            }
         }
         if (key==72){//up
            y1=y1-20;
            y2=y2-20;
            if(y1<=50){
               y1=50;
               y2=90;
            }
         }
         if(key==80){//down
            y1=y1+20;
            y2=y2+20;
            if(y1>=360){
               y1=360;
               y2=400;
            }
         }
         if(key==27){
            break;
         }
      }
      rectangle(x1,y1,x2,y2);
      //side lines
      int b=2;
      int x=50,y=400;
      int k=25;
      //big lines
      line(x+k,down,150,down);
      line(300,down,y-k,down);
      down=down+20;
      //small lines
      line(x+k+50,down,150,down);
      line(300,down,y-k-50,down);


      k=k+25;
      if(k>=100){
         k=25;
      }
      down=down+20+b;

      if(down+20>400){
         b=b++;
         down=50+b;

         //color change
         i=i+1;
      }
      if(cy<380){
         circle(cx,cy,10);
         cy=cy+15;
      }
      if(cy>380){
         cy=60;
          cx= (rand() % (290 - 160 + 1)) + 160;
      }
      //collision mechanic
      if((cx+10>=x1)&&(cy+10>=y1)&&(cx-10<=x2)&&(cy-
10<=y2)){
         death++;
         if(score>hscore){
            hscore=score;
         }
         outtextxy(cx, cy, "YOU DIED");
         goto start;
      }
      delay(del);
      cleardevice();
      score++;
      if(score%20==1){
         level++;
         del=del-30;
         if(del<40)del=40;

         //for color changing
         if(i>14)i=1;
         i++;
         setcolor(i);
      }
   }

   printf("High score is = %d\n",hscore);
 return 0;

}
```

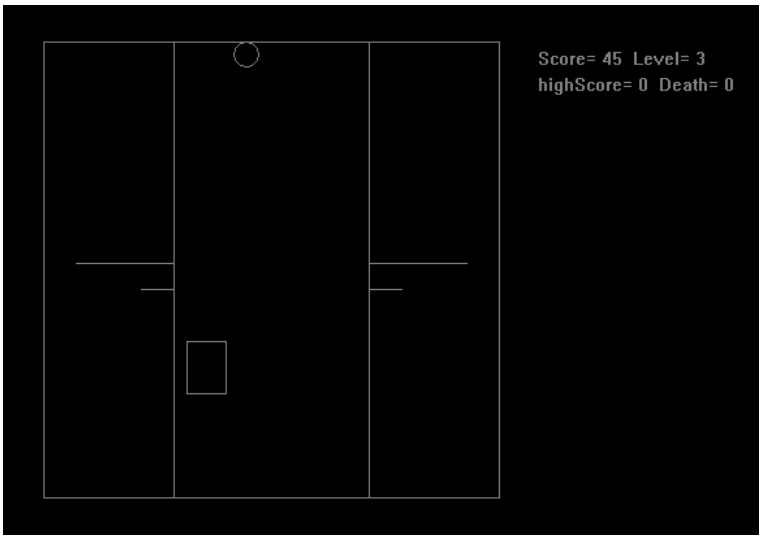
CHAPTER 5: Testing

In this chapter we are going to test the functionality of the arrow key functions and collision mechanic implementations.

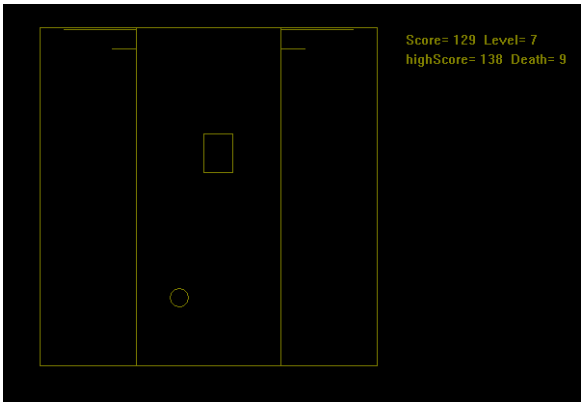
Now If we press the right arrow key we can see that the car moves right :

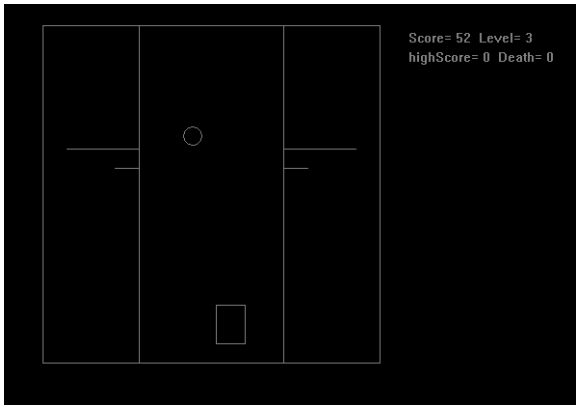


If we press the left arrow key we can see that the car moves left :

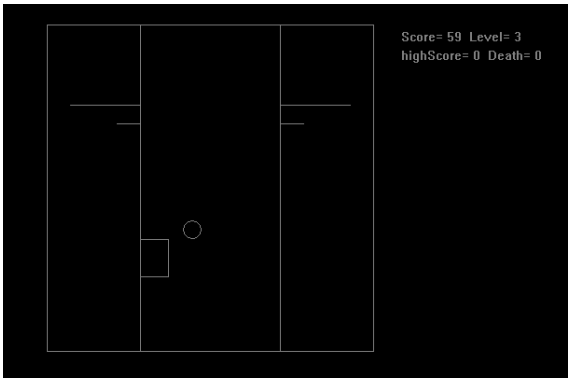


Same goes for pressing up and down :

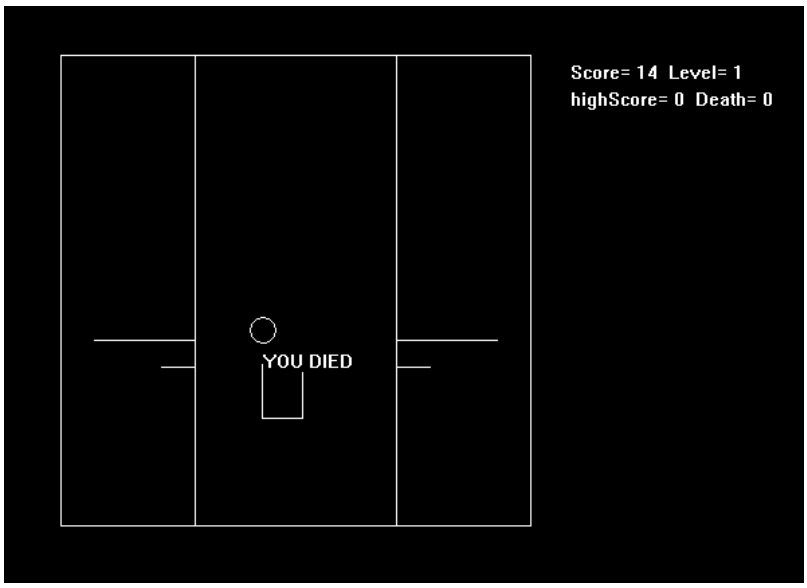




and the car can't move beyond the boundary :

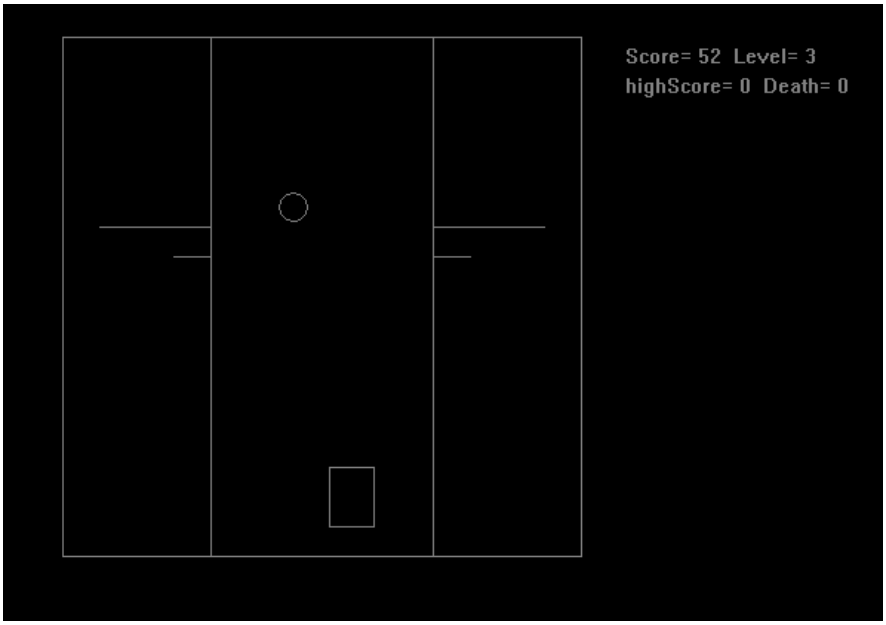


And if Collision happens the game gives you a "You Died" warning and increase the death counter while starting the new game :



CHAPTER 6: Result

In this project we learnt about the objects of *graphics.h* header and how to create an interactive game using the elements of the header , resulting in a basic yet interactive game .With features like score , Level , Color changing , Death count and High score.



CHAPTER 7: Limitation

The key hold of any arrow key takes way too many inputs to output simultaneously, so each input should be given one at a time, meaning you should press any arrow key once at a time.

CHAPTER 8: Conclusion

This was a fun project to work on, that taught me more about computer graphics and simple game mechanics. Overall we can say that the project was a success.