**Computer Graphics Mini Project**

**MAZE RUNNER**

-Bhavesh khubnani ,31

& Dhiren Motwani ,43

-D7C

# 

# Introduction

Maze Runner,Objective of this game is to collect all of the 5 coins inside the maze and solve the maze in as little time as possible.

**Working**

When user starts the game ,program displays the maze and all the coins.The exit gate is closed and only opens after all the 5 coins have been collected,this is achieved by using a counter which checks if the current position is equal to the coin position ,if true then it increments the coin counter,same is reflected on screen.

Out of 5 coins 1 coin is generated Randomly and 4 are fixed points. This is achieved by generating new coordinates using builtin Rand() and srand() which generate random number and random seed number.

For input ASCII values of arrow keys were compared with the input values. If input was equal to ASCII value of down key then the position would be changed if it is a valid position.

A position is valid if it does not cross the maze walls.This is done by adding +10 to current position in the direction of input.If the new position is not intersecting the maze walls(boundary) then current position is changed to new position.Position is checked by comparing the New Position color with boundary color,if it matches then new position is invalid and message is displayed saying **“INVALID INPUT”** and input is taken again using getch() function.

Input is taken using while loop which terminates when user reaches the exit after collection all 5 coins.

Maze is drawn using line() function of the graphics.h helper file.

**LIBRARIES USED:**

**A.Graphics.h**

C graphics using graphics.h functions or WinBGIM (Windows 7) can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in Turbo C compiler you can make graphics programs, animations, projects, and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them. Following is a list of functions of graphics.h header file. Every function is discussed with the arguments it needs and its description.

### FUNCTIONS USED:

* **Cleardevice**: cleardevice function clears the screen in graphics mode and sets the current position to (0,0). Clearing the screen consists of filling the screen with current background color.
* **Closegraph**: closegraph function closes the graphics mode, deallocates all memory allocated by graphics system and restores the screen to the mode it was in before you called initgraph.
* **Getcolor**: getcolor function returns the current drawing color.Declaration : int getcolor();
* **Getpixel:** getpixel function returns the color of pixel present at location(x, y).
* **Line:** line function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line.The code given below draws a line.Declaration: void line(int x1, int y1, int x2, int y2);
* **Outtextxy:** outtextxy function display text or string at a specified point(x,y) on the screen.Declaration: void outtextxy(int x, int y, char \*string);
* **Circle:** Circle function is used to draw a circle with center (x,y) and third parameter specifies the radius of the circle. The code given below draws a circle.Declaration: void circle(int x, int y, int radius);
* **Putpixel:** putpixel function plots a pixel at location (x, y) of specified color. Declaration: void putpixel(int x, int y, int color);
* **Floodfill:** Floodfill function is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area.(x, y) is any point on the screen if (x,y) lies inside the area then inside will be filled otherwise outside will be filled,border specifies the color of boundary of area. To change fill pattern and fill color use setfillstyle. Code given below draws a circle and then fills it.
* **Rectangle:**Declaration: void rectangle(int left, int top, int right, int bottom); Rectangle function is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle. left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner. The code given below draws a rectangle.
* **Setviewport:** Setviewport function sets the current viewport for graphics output.Declaration: void setviewport(int left, int top, int right, int bottom, int clip);

**B. TIME.H**

The time.h header defines four variable types, two macro and various functions for manipulating date and time.

## 

## 

## Library Variables :

Following are the variable types defined in the header time.h −

|  |  |
| --- | --- |
| **Sr.No.** | **Variable & Description** |
| 1 | size\_t  This is the unsigned integral type and is the result of the sizeof keyword. |
| 2 | clock\_t  This is a type suitable for storing the processor time. |
| 3 | time\_t is  This is a type suitable for storing the calendar time. |
| 4 | struct tm  This is a structure used to hold the time and date. |

**CODE :**

#include<time.h>

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

void maze(int,int);

char a[50];

int f1=0,f2=0,f3=0,f4=0,f5=0;

int c=0,g,l;

int f11=1,f22=0,f33=0,f44=0,f55=0;

clock\_t before;

clock\_t diff;

char z;

int tt=0;

void main()

{

char ck[100];

/\* request auto detection \*/

int gdriver = DETECT, gmode, errorcode;

char ch=0,h;

int x,y;

/\* initialize graphics and local variables \*/

initgraph(&gdriver, &gmode, "C://TURBOC3//BGI ");

p:

cleardevice();

before=clock();

l=0;

srand(time(0));

f11=f22=f33=f44=f55=f1=f2=f3=f4=f5=h=0;

setaspectratio(1600,1280);

c=0;

x=80;

y=40;

maze(x,y);

ch=0;

/\* read result of initialization \*/

errorcode = graphresult();

/\* an error occurred \*/

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

while(ch!=27) //escape=27

{

if(l==0)

{

u:

g=(rand())%10;

g=g\*10;

if(getpixel(50+g,50+g)==3)

goto u;

l=l+1;

}

ch=getch();

if((x==120&&y==280)&&(f11<3))

{ f1=1;

c=c+1;

f11++;

}

if((x==100&&y==100)&&(f33<3))

{ f3=1;

c=c+1;

f33++;

}

if((x==(50+g)&&y==(50+g))&&(f22<3))

{ f2=1;

c=c+1;

f22++;

}

if((x==200&&y==320)&&(f44<3))

{ f4=1;

c=c+1;

f44++;

}

if((x==210&&y==140)&&(f55!=2))

{ f5=1;

c=c+1;

f55++;

}

if((x==80&&y==40)&&(ch!=80)) //only down

{

outtextxy(350,350,"INVALID INPUT");

continue;

}

switch(ch)

{

case 72: //up

if(getpixel(x,y-10)==3)

{

outtextxy(35,300,"INVALID INPUT");

continue;

}

else

y=y-10;

break;

case 80: //down

if(getpixel(x,y+10)==3)

{

outtextxy(350,200,"INVALID INPUT");

continue;

}

else y=y+10;

break;

case 77: //right

if(getpixel(x+10,y)==3)

{

outtextxy(350,200,"INVALID INPUT");

continue;

}

else

x=x+10;

break;

case 75: //left

if(getpixel(x-10,y)==3)

{

outtextxy(350,200,"INVALID INPUT");

continue;

}

else x=x-10;

break;

}

cleardevice();

maze(x,y);

if(x==310)

{

s:

h=getch();

outtextxy(350,200,"YOU WIN");

diff =-(before-clock());

sprintf(ck,"Time taken %d seconds %d milliseconds\n",

diff/10,diff%10);

outtextxy(350,180,ck);

outtextxy(350,220,"PRESS R TO PLAY AGAIN OR Q TO EXIT");

if(h=='q'||h=='Q')

{

break;}

else if(h=='r'||h=='R')

{goto p;}

else

{goto s; } //stop timer

}

}

getch();

closegraph();

}

void maze(int x,int y)

{

setcolor(WHITE);

outtextxy(400,100,"COLLECT ALL 5 COINS TO WIN");

setcolor(3);

line(50,50,70,50);

line(90,50,290,50);

line(290,50,290,290);

line(290,310,290,370);

line(50,370,290,370);

line(50,50,50,370);

line(70,70,70,90);

line(50,90,70,90);

line(90,90,270,90);

line(90,70,90,90);

line(50,110,250,110);

line(270,90,270,230);

line(270,230,150,230);

line(150,230,150,150);

line(150,150,90,150);

line(250,110,250,150);

line(250,150,200,150);

line(200,150,200,130);

line(200,130,70,130);

line(70,130,70,310);

sprintf(a,"THE NUMBER OF COLLECTED COINS IS %d",c/2);

outtextxy(150,450,a);

line(70,310,150,310);

line(150,310,150,350);

line(150,350,250,350);

line(250,350,250,310);

line(250,310,170,310);

line(170,310,170,330);

line(170,330,230,330);

line(90,150,90,290);

line(90,290,290,290);

line(170,130,170,170);

line(270,290,270,350);

rectangle(170,170,250,210);

line(70,50,70,30);

line(90,50,90,30);

line(290,290,310,290);

line(290,310,310,310);

if(c<10)

line(290,290,290,310);

if(f1!=1)

circle(120,280,3);

if(f2!=1)

circle(50+g,50+g,3);

if(f3!=1)

circle(100,100,3);

if(f4!=1)

circle(200,320,3);

if(f5!=1)

circle(210,140,3);

setcolor(WHITE);

outtextxy(65,15,"ENTRY");

outtextxy(315,297,"EXIT");

setcolor(YELLOW);

circle(x,y,3);

setfillstyle(1,2);

floodfill(x,y,14);

}

**OUTPUT :**



