**2.2 MAZEFIN1.CPP**:-

This files has all necessary functions to be used in the main program and other parts.

**HEADER FILES:-**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

#include<math.h>

#include<fstream.h>

#include<string.h>

#include<dos.h>

**FUNCTIONS USED:-**

1. ***void start():***

This function has starting effects when the game starts. Almost every game has loading effect in different manners. So we created it in our style.

Figure 1 Figure 2

On execution of the function following steps are viewed.

* Two integer variables ‘c’ and ‘r’ are taken. This variables are taken for printing white bar & the curved line below as you can see in the figure. ‘r’ is initialized to 300.
* cleardevice(); function clears the screen and background color is set ‘blue’.
* One white rectangle is drawn in which the white bar will be displayed by increasing column size.
* One for loop is taken which will start from c=40 and increasing value on it by 5 until c=565 which is the dimension of white rectangle.
* Function bar needs four coordinates. Here row coordinates remain same as bar has same vertical length all the time. Horizontal length keeps on increasing with ‘c’ as it is one coordinate of white bar.
* The string “TIME TO PLAY THE GAME” is printed in the background color. So the text appears from the background color gradually.
* Variable ‘r’ is used for printing underlined curve. The value of ‘r’ varies as following formula:- “r=300+(5\*sin(c)) “ and it’s used in row coordinate of the center of white circle printed using ‘pieslice’ function. So the line looks like a sine wave.
* Final image looks like as shown in figure 2.

1. ***void Lmenu(int l,int f) and int menumove(int l):-***

This two functions are interconnected. This functions are used to print “LEVEL MENU” shown below in figure 1.



Figure 1

The Lmenu function has two parameters:- first is for how many levels are to be printed and second is the flag variable which determines what bar is to be printed blue or red for the “MAIN MENU” part. This function prints design for different layouts. The different layouts of the Lmenu function is shown below(figure 2 and 3). This layouts are created according to the fact that when user plays for the first time the other two levels are locked.

Figure 2 Figure 3

The function menumove takes one integer parameter about what the level is to be printed. And returns one parameter which is a flag value which determines when we hit Enter what “LEVEL” or “MAIN MENU” is selected which will be used for the further procedure. How this two functions communicates with each other is shown by block diagram chart shown below.

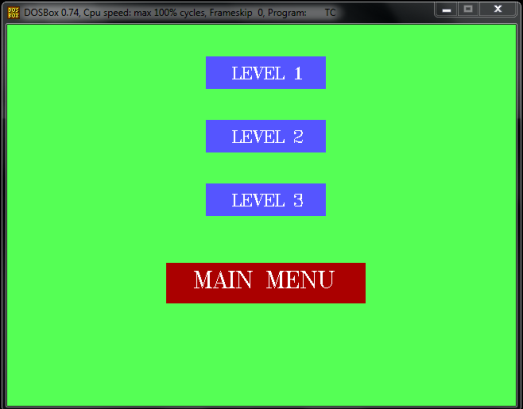
This is shown for when all levels are unlocked. All other layout follows same procedure.   Figure 3.2 Figure 3.4



Figure 3.3

void Lmenu(int l,int f)

{ if(l==1)

{ design1; //(figure2)

}

else if(l==2)

{ design2; //(figure 3)

}

else if(l==3)

{ design3; //(figure 1)

}

if(f==0)

{ figure 1;

}

else if(f==1)

{ figure 3.2;

}

else if(f==2)

{ figure 3.3;

}

else if(f==3)

{ figure 3.4;

}

}

Int menumove(int l)

{ int f=0;

char key;

if(l==3)

{ key=getch();

while(key!=27)

{ key=getch();

if(key==72)

{ if(f==0)

{ Lmenu(3,3);

f=3; }

else if(f==1)

{ Lmenu(3,0);

f=0; }

else if(f==2)

{ Lmenu(3,1);

f=1; }

else if(f==3)

{ Lmenu(3,2);

f=2; }

}

else if(key==80)

{ if(f==0)

{ Lmenu(3,1);

f=1;}

else if(f==1)

{ Lmenu(3,2);

f=2;}

else if(f==2)

{ Lmenu(3,3);

f=3;}

else if(f==3)

{ Lmenu(3,0);

f=0;}

}

else if(key==13)

{ break;} } }

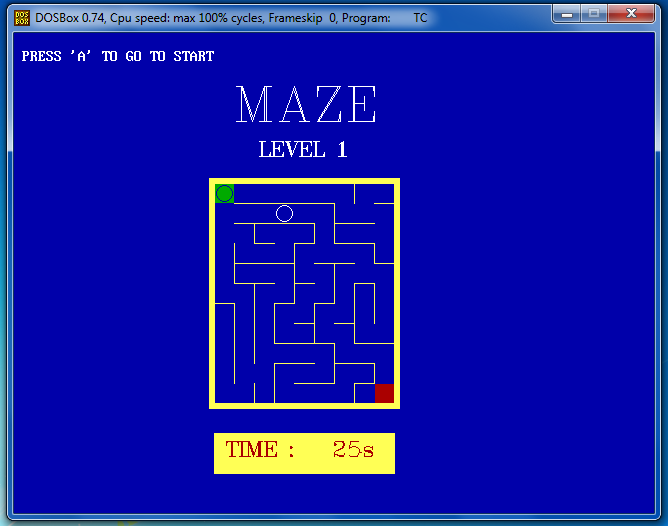
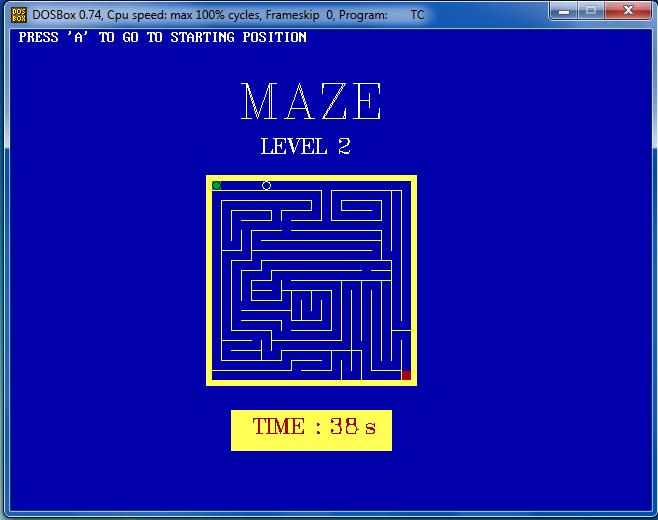
return f;

}

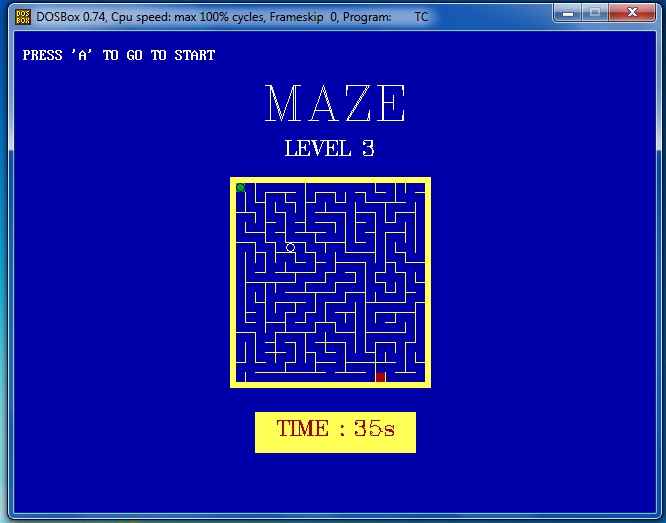
From the connectives you can see that only the design keeps changing for each movements of ‘up’ and down keys making this a dynamic menu. The locks drawn with some basic shapes shown in figure 2 and 3.

***3)int LEVEL1(); int LEVEL2(); & int LEVEL3();***

This three functions generates design on each level. The user plays the game in this functions i.e. he starts the ball moving from starting position to ending position. The time is calculated and it’s return to the calling function. The time calculated is the parameter for showing high scores i.e from less time to greater. The designs of all three levels are shown below.

LEVEL1 LEVEL2



LEVEL3

All three level funtions has same funtionality. The following steps are executed for all the three levels in same manner.

* Console screen is cleared with cleardevice(); and background color is set to blue.
* The maze design is drawn by line functions. These designs are taken from existing design we found from different resources. In drawing lines we need four parameters viz. start column ,start row, end column, end row. Horizontal lines are drawn with same start row and end row & the vertical line has same start column and end column parameters. These parameters are given in pixels.

example: line(c+120,r+10,c+140,r+10);

Here ‘c’ and ‘r’ are reference parameters for all lines drawn to design i.e. starting pixel of the outer rectangle. This are useful because we don’t have to remember pixel address of every line and the ball (i.e. cursor of maze) moves with reference to this two parameters.

* After design is drawn the clock starts. For showing the time period in the program header file ”time.h” has class clock\_t. We create three objects (or variables) of it st, rt, et. Each representing starting time, intermediate time and ending time respectively. When we write “st=clock();” the clock starts ticking and when another variable calling clock function is encountered the clock stops at that moment. To convert the time difference in integer variable displaying seconds, the “time.h” file has a **macro** “rc=(et-st)/CLK\_TCK”; ‘rc’ is an integer variable.
* After we start the clock while loop gets started of which the termination condition is the ‘esc’ key. Each time a variable defined ‘key’ takes a character from keyboard using ‘getch();’ function. The keys used in moving the bar are:-

1) Arrow keys:- To move the bar up, down, right, left.

2) ‘A’ or ‘a’:- To go to starting position.

3) Esc key:- To terminate the game without completing it if some fault occurs.

The pressed key is checked by the if else ladder according to **“ASCII”** values of the keys which are as follows:

UP:- 72 DOWN:-80 RIGHT:- 77 LEFT:-75

‘A’:- 65 ‘a’:- 97 Esc:- 27

* Two variables ‘c’ and ‘r’ are taken which are initialized to starting coordinates of the circle which moves in the entire maze. According to key pressed value of ‘r’ and ‘c’ is increase of decrease to change the position of ball. According to the basic functions of each key following actions are executed.

UP : Value of ‘r’ is decreased.

DOWN : Value of ‘r’ is increased.

RIGHT : Value of ‘c’ is increased.

LEFT : Value of ‘c’ is decreased.

* Before changing value of ‘c’ of ‘r’ values them in previous iteration of while checked by two types of conditions.

1) Border conditions: So that the ball doesn’t move outside of design.

2) Blocking conditions: So that ball can’t go up or down across horizontal lines & right or left across vertical lines. This conditions are put together in ‘if\_else ladder’ in the key checking blocks. This conditions is checked in a way if ball is between some position it can’t go to next position pressed by the user.

These blocking conditions are the main logic of the maze game. You can’t go across the lines or from outside border. When any of these condition is satisfied the control of that loop iteration using ‘goto’ statement. So the preceding statements won’t be executed which moves the circle.

For example:-

if((c>=c0+20 && c<c0+120) && r==r0+20)

{ goto ENTER; }

else if((c>=c0+160 && c<c0+180) && r==r0+20)

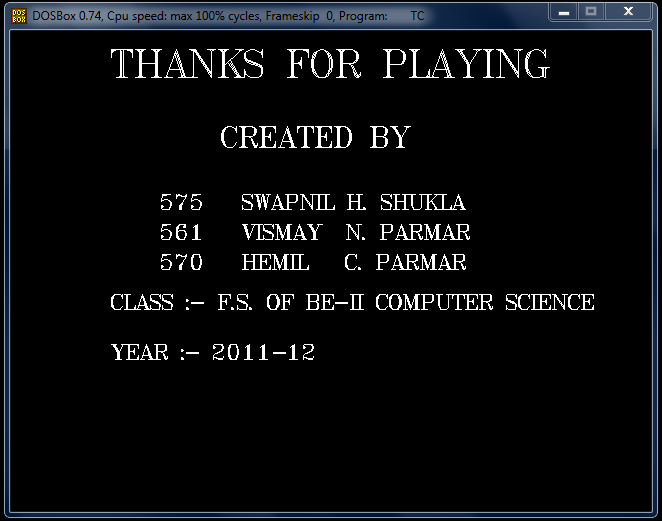
{ goto ENTER; }

This conditions will check whether the current position ball determined by ‘c’ and ‘r’ and is between columns ‘c0+20’ and ‘c0+120’ and the row is ‘r0+20’ then it will not allow ball to go up across the line which is between given two columns. All other conditions are written in same way .

* After any of conditions explained before is not satisfied then the value of ‘r’ and ‘c’ is changed according to pressed key and white circle is printed having coordinates center of the circle which are ‘c’ and ‘r’ and radius of the circle which is kept according to width between lines.
* Whenever above execution is done we have to erase the previous white circle otherwise all the white circle will be shown continuously. So two temporary variables ‘c1’ and ‘r1’ takes current values of ‘c’ and ‘r’ respectively. Then they are increased or decreased according to last position of circle. And black circle is printed exactly at the last position. So it looks like only one circle is moving dynamically. The last position of circle is determined using a **flag variable ‘f’** which has values:-1 for up, 2 for down, 3 for right, 4 for left, 6 for going to starting position i.e. for ‘a’ (or ‘A’) and 5 for ending the while loop. This flag values are checked using ‘if else ladder’ and then value of ‘c1’ and ‘r1’ is changed and black circle is printed. The values of ‘c1’ and ‘r1’ are changed in exactly opposite manner in which we increased or decreased them. For example using up key value of ‘r’ is decreased, so ‘r1’ takes value of ‘r’ and increases and then black circle is printed whose center will be at the last position determined by ‘c1’ and ‘r1’.
* In all iterations of while loop current time is displayed continuously using clock\_t variable ‘rt’. It determines the current time. Then the time difference between ‘st’ and ‘rt’ is calculated in seconds and stored in int variable. For displaying that time it’s converted into string using ‘itoa()’ function defined in header file “**stdlib.h”.**
* After the ball reaches ending position the message “YOU MADE IT” is displayed and while loop breaks.
* The end time is stored in variable ‘et’. The time difference between ‘st’ and ‘et’ is calculated in integer variable, it’s displayed and returned to the calling function which stores the time particularly for that level.

***4) void end();***

This function displays some messages when the game ends. These messages are displayed as credits in a film. This is shown in the image below.



**2.3 THEGAME.CPP:**

This is the driver program. It uses the definitions from the previous two files and contains the code for actual gameplay and high scores.

*CLASSES AND FUNCTIONS INCLUDED:-*

1) class Score:-

This class contains standard input function for inputting player’s name which is taken every time when any new player starts for first time. This class also contains one integer variable to save score of that player for different levels. Other functions are shown below.

* void gettime(int) :- This function takes int variable as parameter and stores it the variable made for saving the time taken in completing game.
* Int get\_time():- This function returns the time for that object.
* char\* get\_name():- This function returns the name of player which is a string.

2) void Gameon(Score &S):-

This function takes the current object as parameter. It runs the game for all levels with level locks and saving scores in the files. Following steps are executed.

* First there is a for loop which displays the Level menu explained above and takes the flag value of the cursor on which user has player has pressed ‘Enter’.
* According the flag value, level 1,2,3, or the Main Menu is opened using appropriate functions and conditions. When user selects one level for playing, for example ‘LEVEL1’, one file is opened in write mode, for this ‘LEVEL1.CPP’ opened, LEVEL1() function is called and the time taken in the completing level is stored in a int variable. This variable is sent to current object using ‘gettime(int t)’ function and the object is written in the file. This procedure for all levels and scores of all levels are stored in separate files so that they can be easily retrieved from corresponding file.
* After all levels are unlocked user can continue to play again and again until he hits MAIN MENU button. The scores will be saved with the same name.

3)void highscore(int ls):-

This function takes integer variable which according to corresponding level of the game taken from Level Menu. According to the level passed corresponding file is opened in read mode. First no. objects are calculated. Then according to no. of objects the values of scores and corresponding name are retrieved from the current object in loop and they are stored in integer array and character array (2D array) with one to one correspondence between them. This integer array is sorted in ascending scores with corresponding names are also sorted. Then **top 5** scores are displayed as shown below.



* **THE main() FUNCTION**

The procedure of main function is shown by block diagrams as shown below. All the functions used are explained above.

main()

menuinit();

int M=0;

while(M!=3)

M=mazemenu();

end();

else if(M==1)

{ cleardevice();

Lmenu(3,0);

hs=menumove(3);

switch(hs)

{ case 0 : highscore(1); break;

case 1 : highscore(2); break;

case 2 : highscore(3); break;

default : break; } }

else if(M==2)

{ break; }

if(M==0)

{ cleardevice();

S.input();

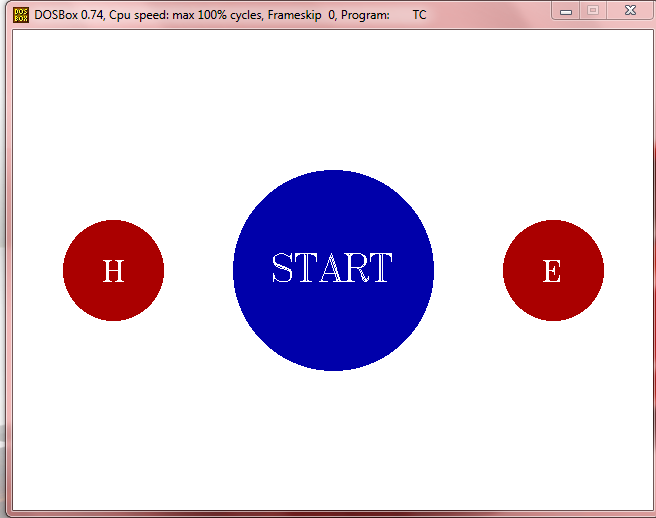
start();

Gameon(S);

}

3. SNAP SHOTS OF THE GAME

***3.1 MAIN MENU***



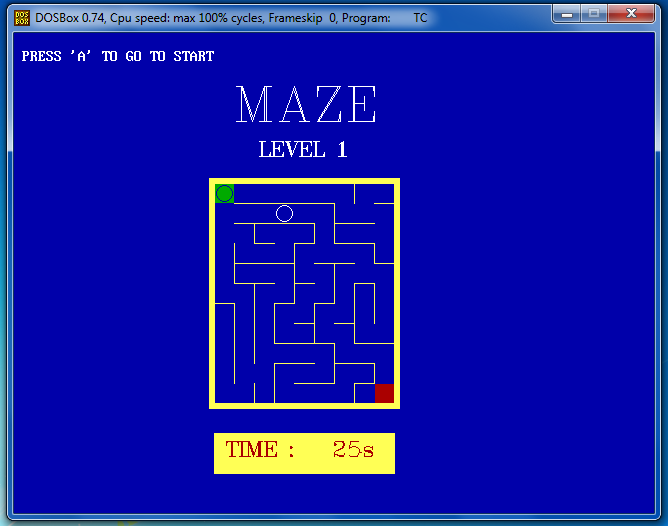
***3.2 START***



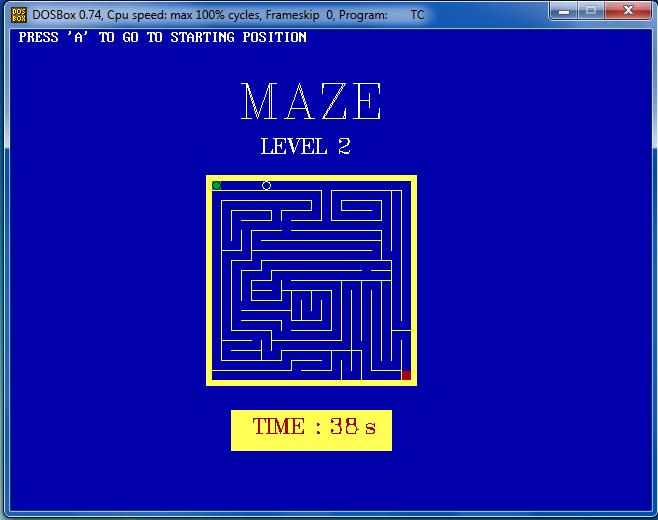
***3.3 LEVEL MENU WITH LEVEL LOCKS***



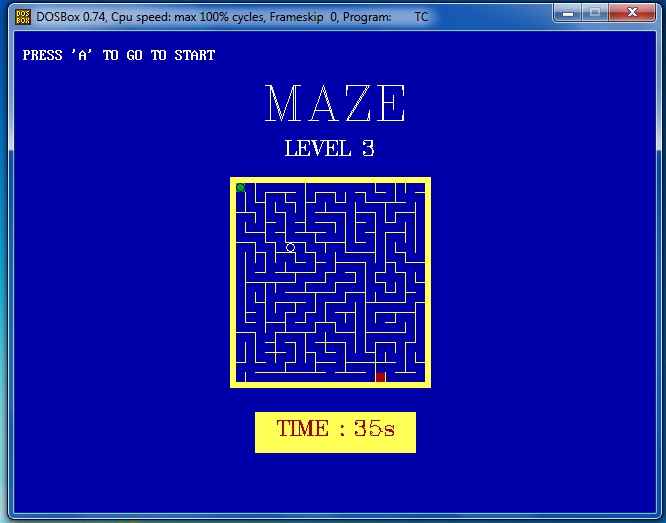
***3.4 LEVEL 1***



***3.5 LEVEL 2***



***3.6 LEVEL 3***



***3.7 HIGHSCORES***



***3.8 THE END***

