A Project report on

DESIGN OF GAME USING C GRAPHICS

In

Advanced Academic Centre

By

C. PRANEETH REDDY (14241A1213)

N. VENKAT REDDY (14241A1243)

K. SRI HARSHA (14241A1226)

Under the esteemed guidance of

DR. Y. VIJAYALATA

HOD of IT

DEPARTMENT OF INFORMATION & TECHNOLOGY

Gokaraju Rangaraju Institute of Engineering and Technology

(Autonomous)

Gokaraju Rangaraju Institute of Engineering and Technology

(Autonomous)

Department of information & technology

CERTIFICATE

This is to certify that the project report titled “Design Of Game Using C Graphics” is a bonafide work done by C. Praneeth Reddy, K. Sri Harsha, N. Venkat Reddy bearing the roll numbers 14241A1213, 14241A1226, 14241A1243, in partial fulfilment of the requirements for the award of the certificate in advanced academic centre and submitted to the Department Of Information Technology, Gokaraju Rangaraju Institute Of Engineering And Technology, Hyderabad.

This work was not submitted earlier at any other University or Institute for the award of any certificate.

Dr. Y. Vijayalata	**Dr. S. Rama Murthy**
(Head of Department)	**(Vice Principal)**

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our guide **Dr. Y. Vijayalata, Head of Department, Information Technology**, Gokaraju Rangaraju Institute of Engineering & Technology College. We express our sincere thanks for her encouragement, suggestions and support, which provided to impetus and paved the way for the successful completion of the project work and we feel very fortunate to have this opportunity to study under her supervision.

We are grateful to **Dr. S. Rama Murthy, Vice Principal** and the members of the Advance Academic Centre for their valuable suggestions.

We express our sincere thanks to **Prof. P. S. Raju,** Professor and Director of Gokaraju Rangaraju Institute of Engineering & Technology College, and **Dr. Jandhyala N Murthy,** Principal of Gokaraju Rangaraju Institute of Engineering & Technology College, for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad, for their kind cooperation throughout.

C. PRANEETH REDDY
K. SRI HARSHA
N. VENKAT REDDY

DECLARATION

This is to certify that the project titled **DESIGN OF GAME USING C GRAPHICS** is a bonafide work done by us in partial fulfilment of the requirements for the award of the certificate and submitted to the Advanced Academic Centre, Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad.

We also declare that this project is a result of our own effort and this has not been submitted at any other university or institute for the award of any certificate.

C.Praneeth Reddy	**D.S.PrasanthVarma**	**P.BharathChandra**
chpraneeth1539@gmail.com	srikacharla@gmail.com	venkat.narahari96@gmail.com
+91 96180 30665	+91 73868 01391	+91 99855 35643

ABSTRACT

Games have been a medium of entertainment for a long time. From the time of Tetris and Mario to Call of Duty and Dota 2, games have taken a huge leap in terms of their graphic content. Gaming on mobiles has been prevalent from the later stages of mobile infancy itself. Of all the famous mobile games, SNAKE is one of the most popular ones. The game consists of a snake which grows in size by eating food. We are trying to recreate the experience of the SNAKE game through graphics using the C language. In this case the SNAKE grows in size whenever it touches a square. If the snake touches itself it dies and the game ends.

INDEX

LIST OF FIGURES & SCREENSHOTS

1. INTRODUCTION

Snake is a general name for a video game where the player manoeuvres a line which grows in length, with the line itself being a primary obstacle. There is no standard version of the game. The concept originated in the 1976 arcade game blockade, and its simplicity has led to many implementations (some of which have the word *snake* or *worm* in the title). After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in the Snake concept as it found a larger audience.

Snake is an older classic video game. It was first created in late 70s. Later it was brought to PCs. In this game the player controls a snake. The objective is to eat as much food as possible. Each time the snake eats food, its body grows. The snake must avoid the walls and its own body. This game is sometimes called *Nibbles*.

Development

The size of each of the joints of a snake is 20px. The snake is controlled with the cursor keys. Initially, the snake has one joint. If the game is finished, the "Game Over" message is displayed in the middle of the board.

In the second variant, a sole player attempts to eat objects by running into them with the head of the snake. Each object eaten makes the snake longer, so manoeuvring is progressively more difficult.

2. FEATURES USED

Basics in C

Data structures, loops, conditional statements, file accessing methods, dynamic memory allocation, function calls, GUI.

Why only Single linked list

The game requires us to increase the size of the snake every time the head coincides with the food. Hence we need to create and allocate new memory dynamically every time this happens. In order to achieve this, linked lists is the most efficient method. Using linked lists we store the x and y coordinates of each block in two nodes and store the location of the next node using a pointer.

As the head of the snake changes its position, the block preceding it needs to acquire the coordinates of the head and so on. Using single linked list we traverse to the end and update the coordinates every time the head changes position. This gives the illusion of the tail following the head and the whole transition makes the movement look smooth.

3. STRUCTURE OF C

1. The documentation section consists of a set of comment lines giving the name of the program, date and other details which the programmer would like to use later.
2. The link section provides instructions to the compiler to link functions from the system library.
3. The definition section defines all the symbolic constants.
4. There are some variables that are used in more than one function. Such variables are called Global variables and are declared in the global declaration section that is outside all the functions. This section also declares all the user-defined functions.
5. Every c program must have one main() function section. This section contains two parts:

   1. Declaration part and 2. Execution part.

   The declaration declares all the variables used in the executable part. There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. ({ }). The program execution begins at the opening brace and ends at the closing brace. All statements within the main section should end with a semicolon (;).
6. The subprogram section contains all the user-defined functions that are called in the main function. User-defined functions are generally placed immediately after the main function, although they may appear in any order.
7. All sections except the main section may be absent when they are not required.

4. HEADER FILES

- **conio.h:** Declares various functions used in calling the DOS console I/O routines.
- **dos.h:** Defines various constants and gives declarations needed for DOS and 8086-specific calls.
- **graphics.h:** Declares prototypes for graphic functions.
- **stdio.h:** Defines types and macros needed for the standard I/O package defined in Kernighan and Ritchie and extended under UNIX system V. Defines the standard I/O pre-defined streams stdin, stdout, stdprn and stderr and declares stream level I/O routines.
- **stdlib.h**: Declares several commonly used routines: conversion routines, search or sort routines and other miscellany.

Header files containing their respective functions

1. **conio.h:** clrscr, getch, putch.
2. **dos.h:** delay, sound, int86x.
3. **stdlib.h:** time, exit.
4. **stdio.h:** fopen, fclose, fgetc, fprintf, fscanf, getc, putc, putchar, fseek.
5. **graphics.h:** closegraph, circle, getmaxy, settextstyle, setcolor, restorecrtmode, putpixel, outtextxy, initgraph, getmaxx.

clrscr

- Clears text mode window.
- Declaration: void clrscr(void);
- Remarks: Clrscr clears the current text window and places the cursor in the upper left-hand corner (t position 1, 1).

getch

- getch gets a character from the console but does not echo to the screen.
- Declaration: int getch(void);
- Remarks: getch reads a single character directly from the keyboard, without echoing to the screen.

putch

- Outpts character to the text window on the screen.
- Declaration: int putc(int ch);
- Remarks: putch outputs the character c to the current text window. It is a text-mode function that performs direct video output to the console.

delay

- Suspends execution for interval (milliseconds)
- Declaration: void delay (unsigned milliseconds);
- Remarks: With a call to delay, the current program is suspended from execution for the specified by the argument milliseconds. It is not necessary to make a calibration call to delay before using it. Delay is accurate to one millisecond.

sound

- Sounds turns the PC speaker on at the specified frequency.
- Declaration: void sound (unsigned frequency);
- Remarks: Sound turns on the PC's speaker at a given frequency. Frequency specifies the frequency of the sound in hertz (cycles per second).

int86x

- General 8086 software interrupt interfaces.
- Declaration: int int86x (int intno, union REGS *inregs, union REGS *outregs, struct SREGS *segregs);
- Remarks: Int86x execute an 8086 software interrupt specified by the argument intno. With int86x, you can invoke an 8086 software interrupt that takes a value of DS different from the default data segment, and/or takes an argument in ES.

time

- Time gets time of the day
- Declaration: time_time(time_t *timer);
- Remarks: Time gives the current time, in seconds, elapsed since 00:00:00 GMT, January 1, 1970. It stores that value in the location *timer, provided that timer is not a null pointer.

exit

- exit terminates the program.
- Declaration: void exit(int status);
- Remarks: Exit terminates the calling process. Before termination, exit does the following:
- Closes all files.

fopen

- Fopen opens a stream.
- Declaration: FILE *fopen(const char *filename, const char *mode);
- Remarks: Fopen opens a file and associate a stream with it. It returns a pointer that identifies the stream in subsequent operations.

fclose

- Closes a stream
- Declaration: int fclose(FILE *stream);
- Remarks: fclose closes the named stream. All buffers associated with the stream are flushed before closing.

fgetc

- fgetc gets a character from a stream
- Declaration: int fgetc(File *stream);
- Remarks: fgetc returns the next character on the named input stream. It is a function version of the getc marco.

fprintf

- fprintf sends formatted output to a stream
- Declaration: int fprintf(FILE *stream,const char *format [,argument,…]);

fscanf

- Scans and formats input from a stream.
- Declaration: int fscanf(FILE *stream, const char *format [, address, …]);

getc

- getc is a macro that gets one character from a stream
- Declaration: int getc(FILE *stream);
- Remarks: getc returns the next character on the given input stream and increments the stream's file pointer to point to the next character.

putc

- putc is a macro that outputs a character to a stream.
- Declaration: int putc(int c, FILE *stream);
- Remarks: putc outputs the character given by c to the stream given by stream.

putchar

- putchar is a macro that outputs a character on stdout.
- Declaration: int putchar(int c);
- Remarks: putchar is a macro defined as putc(c, stdout) putchar puts the character given by c on the output stream stdout.

fseek

- Repositions the file pointer of a stream.
- Declaration: int fseek(FILE *stream, long offset, int whence);
- Remarks: fseek sets the file pointer associated with a stream to a new position.

closegraph

- Shuts down the graphics system.
- Declaration: void far closegraph(void);
- Remarks: closegraph deallocates all memory allocated by the graphics system. It then restores the screen to the mode it was in before you called initgraph.

circle

- Circle draws a circle.
- Declaration: void far circle(int x, int y, int radius);
- Remarks: circle draws a circle in the current drawing color.

getmaxx

- Returns maximum x screen coordinate.
- Declaration: int far getmaxx(void);
- Remarks: getmaxx returns the maximum x value(screen-relative) for the current graphics driver and mode.

getmaxy

- Returns maximum y screen coordinate.
- Declaration: int far getmaxy(void);
- Remarks: getmaxy returns the maximum y value(screen-relative) for the current graphics driver and mode.

settextstyle

- Sets the current text characteristics.
- Declaration: void far settextstyle(int font, int direction, int charsize);
- Remarks: settextstyle sets the text font, the direction in which text is displayed, and the size of the characters.

setcolor

- setcolor sets the current drawing color.
- Declaration: void far setcolor(int color);
- Remarks: setcolor sets the current drawing color to color, which can range from o to getmaxcolor.

restorecrtmode

- Restore screen mode to pre-initgraph setting.
- Declaration: void far restorecrtmode(void);
- Remarks: restorecrtmode restores the original video mode detected by initgraph.

putpixel

- It plots a pixel at a specified point.
- Declaration: void far putpixel(int x, int y, int color);
- Remarks: putpixel plots a point in the color defined by color at (x,y).

outtextxy

- outtextxy displays a string at the specific location (graphics mode).
- Declaration: void far outtextxy(int x, int y, char far *textstring);
- Remarks: outtextxy displays a text string in the viewpoint at the position (x,y).

initgraph

- Initializes the graphics system.
- Declaration: void far initgraph(int far *graphdriver, int far *graphmode, char far *pathodriver);
- Remarks: To start the graphics system you must first call initgraph.

5. GLOBAL DECLARATIONS

FILE *fp;

File pointer in order to traverse in file, print and retrieve data.

struct snk

{

int cx,cy;

int d;

struct snk *l;

};

Structure for linked list to store the coordinates.

int s=0,ch,w=150;

s variable to store score.

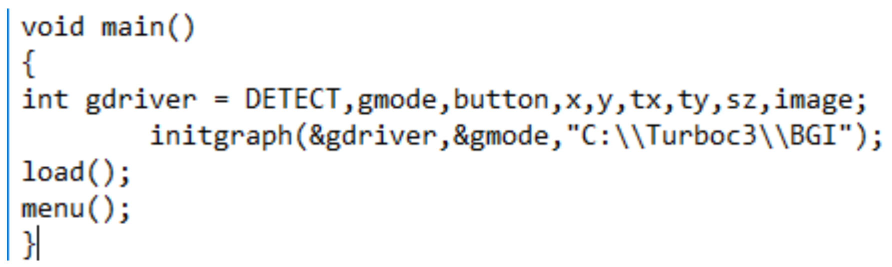
ch to store the retrieved data from the file.

w to change the speed of the snake.

6. FLOW OF CODE

main()

1. The function and the graphics driver is initialized using the initgraph function.

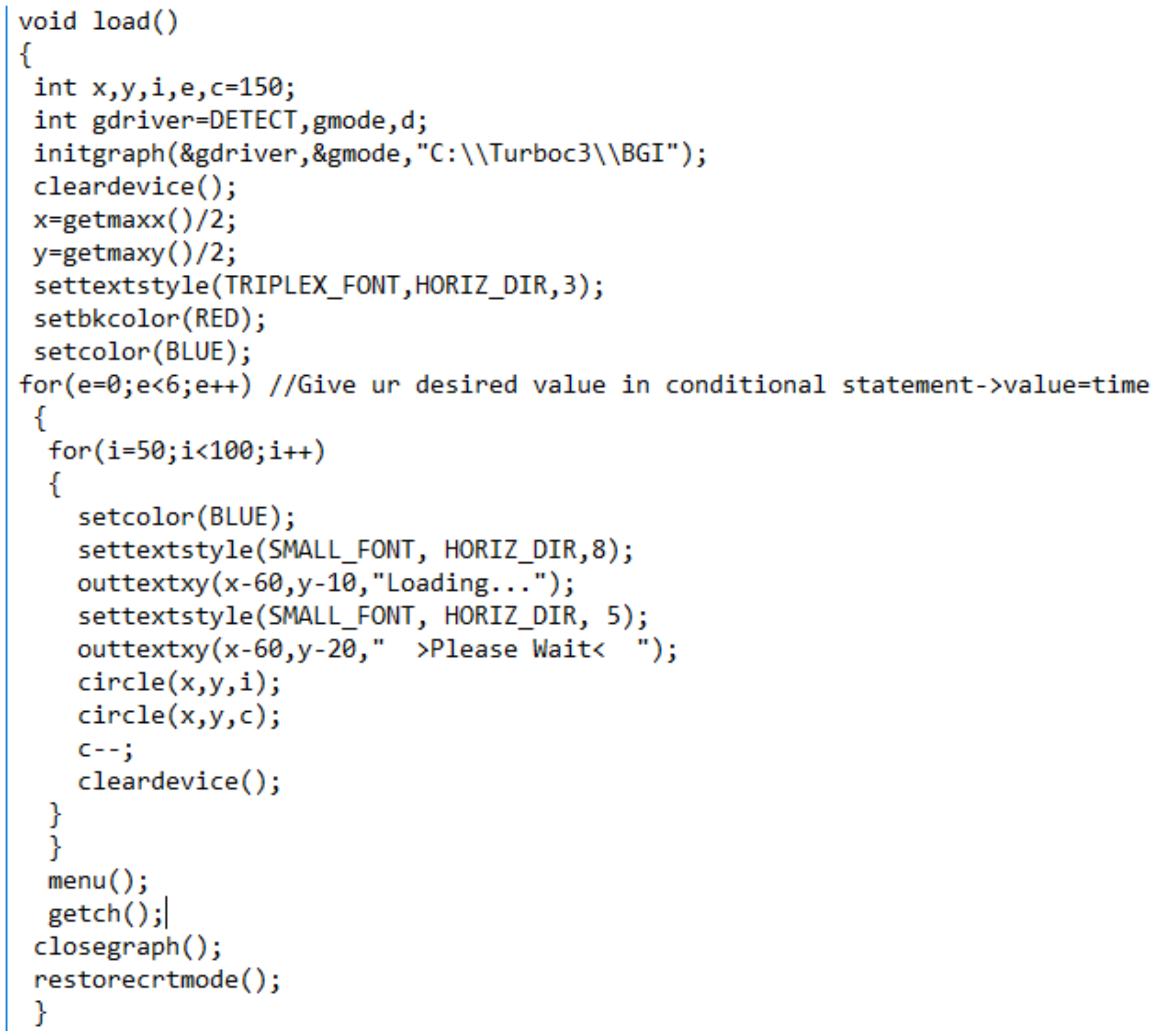


```
void main()
{
int gdriver = DETECT,gmode,button,x,y,tx,ty,sz,image;
        initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");
load();
menu();
}
```

2. Then after the graphics is initialized it calls the load function and the control transfers to the load function.

load()

1. As the control is transferred to the load function, a new screen is appeared where it shows the loading screen with flashing curves and text.
2. The screen is cleared using the clear device function.
3. Then the for loop starts executing and the loading text and the curves are printed on the screen and the for loop creates an illusion of flashing curves and text.
4. After the for loop is done executing the graphics driver is closed using the closegraph function and is reinitialised to the previous mode by the restorecrtmode.



```
void load()
{
 int x,y,i,e,c=150;
 int gdriver=DETECT,gmode,d;
 initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");
 cleardevice();
 x=getmaxx()/2;
 y=getmaxy()/2;
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
 setbkcolor(RED);
 setcolor(BLUE);
for(e=0;e<6;e++) //Give ur desired value in conditional statement->value=time
 {
  for(i=50;i<100;i++)
  {
    setcolor(BLUE);
    settextstyle(SMALL_FONT, HORIZ_DIR,8);
    outtextxy(x-60,y-10,"Loading...");
    settextstyle(SMALL_FONT, HORIZ_DIR, 5);
    outtextxy(x-60,y-20,"  >Please Wait<  ");
    circle(x,y,i);
    circle(x,y,c);
    c--;
    cleardevice();
  }
  }
  menu();
  getch();
 closegraph();
 restorecrtmode();
 }
```

5. The control is then transferred back to the main function.

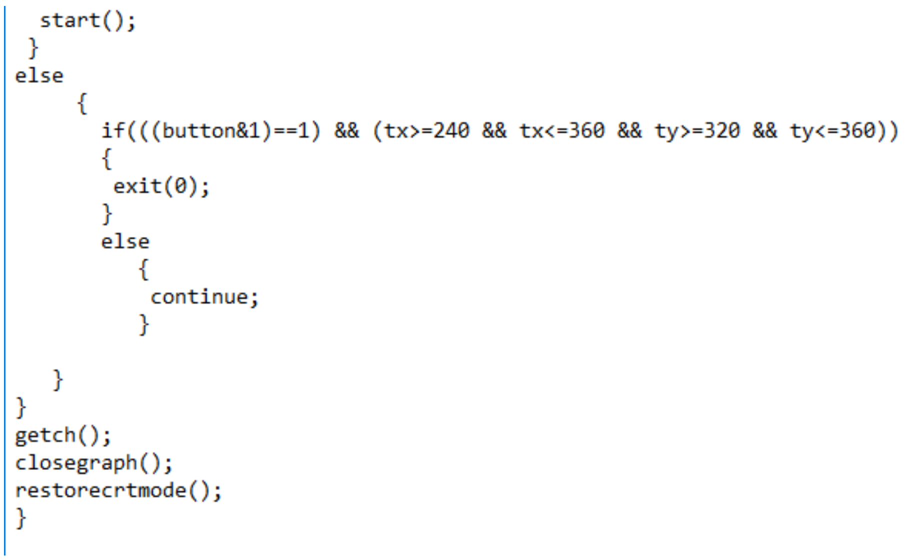
menu()

1. As the control is back to the main function after the execution of load function the menu function is called and the control is transferred to the same.

```
void menu()
{
int gdriver = DETECT,gmode,button,x,y,tx,ty,sz,image;
initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");
getmaxx();
getmaxy();
setbkcolor(RED);
setcolor(WHITE);
outtextxy(225,120,"!!!!!SNAKE GAME!!!!!");
outtextxy(220,460,"Copyrights AAC GRIET");
rectangle(480,380,640,478);
outtextxy(490,390,"DEVELOPERS:");
rectangle(240,200,360,240);
outtextxy(500,400,"1)HARSHA");
outtextxy(500,410,"2)PRANEETH");
outtextxy(500,420,"3)VENKAT");
outtextxy(480,440,"Mentor:Dr.Vijaylatha");
outtextxy(560,450,"(HOD)");
fp=fopen("highscore.txt","r");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\t\t\t\t\n\n\n\t\t       highscore is");
while((ch=fgetc(fp))!=EOF)
putchar(ch);
fclose(fp);
outtextxy(285,220,"PLAY");    /*playoption*/
rectangle(230,260,370,300);
rectangle(240,320,360,360);
outtextxy(285,340,"Quit");
rectangle(240,320,360,360);    /*quitoption*/
showmouseptr();
while(1)
{
getmousepos(&button,&tx,&ty);
if(((button&1)==1) && (tx>=240 && tx<=360 && ty>=200 && ty<=240))
 {
  start();
 }
```

2. The graphics driver is initialized again using the initgraph function and the coordinates of the two extreme coordinates is found using the getmaxx and getmaxy functions.
3. The background is set to RED color and the text color is set to WHITE.

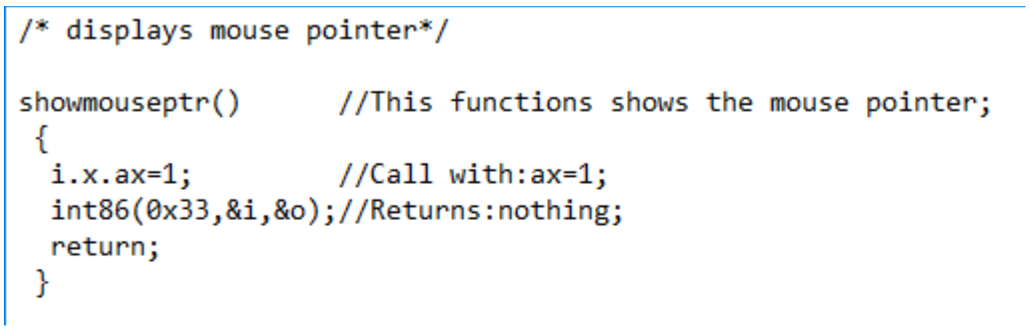
4. Then the text and borders in the menu are printed using outtextxy and rectangle functions respectively.
5. The high score is acquired from the highscore file using the file functions.
6. The showmouseptr function displays the location of the mouse pointer on the screen.
7. The getmousepos function returns the coordinates of the mouse pointer.
8. If the left mouse button is pressed and the coordinates of the mouse pointer are inside the rectangle containing play option. If this happens the control is transferred to the start() function.



```
  start();
 }
else
     {
       if(((button&1)==1) && (tx>=240 && tx<=360 && ty>=320 && ty<=360))
       {
        exit(0);
       }
       else
          {
           continue;
          }

   }
}
getch();
closegraph();
restorecrtmode();
}
```

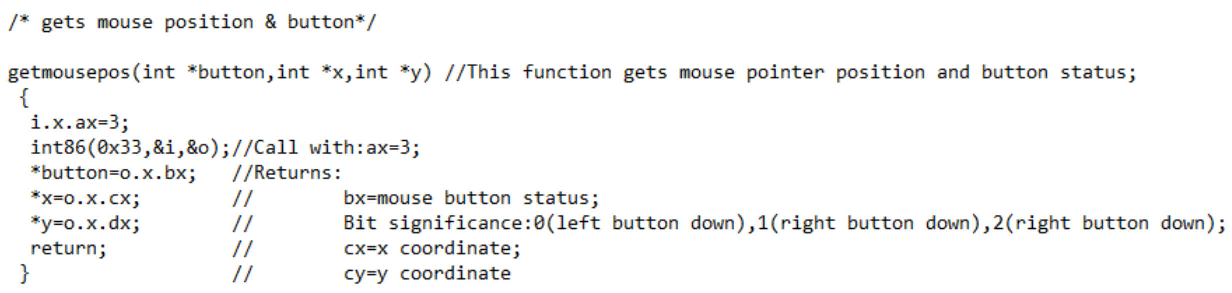
9. If the above condition fails it checks the next condition which is left button is pressed and the coordinates of the mouse pointer are inside the rectangle containing quit option. If this happens the control is transferred to exit() which exits the game.
10. If both the conditions fail, the steps 7 to 9 are performed again using an infinite while loop.

showmouseptr()



```
/* displays mouse pointer*/

showmouseptr()      //This functions shows the mouse pointer;
 {
  i.x.ax=1;         //Call with:ax=1;
  int86(0x33,&i,&o);//Returns:nothing;
  return;
 }
```

getmousepos()



```
/* gets mouse position & button*/

getmousepos(int *button,int *x,int *y) //This function gets mouse pointer position and button status;
 {
  i.x.ax=3;
  int86(0x33,&i,&o);//Call with:ax=3;
  *button=o.x.bx;   //Returns:
  *x=o.x.cx;        //        bx=mouse button status;
  *y=o.x.dx;        //        Bit significance:0(left button down),1(right button down),2(right button down);
  return;           //        cx=x coordinate;
 }                  //        cy=y coordinate
```

Mouse can be used in text mode and also in graphics mode and usually it is used in graphics mode. So the function
'initgraph()' is used to change from text to graphics mode. 'DETECT' is a macro defined in 'graphics.h' which requests 'initgraph()' to automatically determine which graphics driver to load in order to switch to the highest resolution graphics mode(here it is 640x420px).

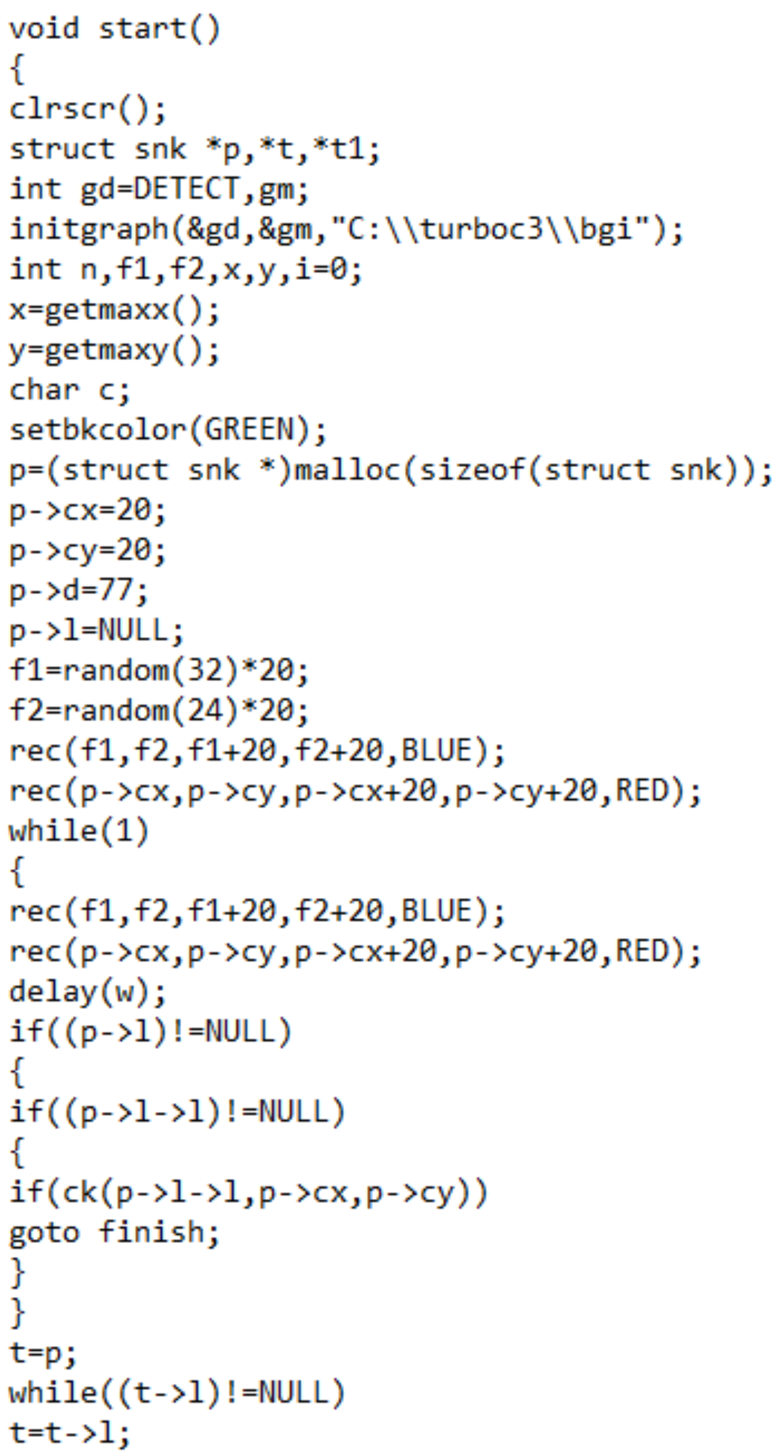
Then initgraph() sets up the numeric values of graphics driver and graphics mode chosen in the variables gd and gm respectively.

Here we are assuming that the driver files are in the directory 'C:\Turboc3\BGI', hence the path passed to initgraph() is 'C:\\Turboc3\\BGI'.
Once we are into graphics mode we have called the functions 'getmaxx()' and 'getmaxy()' to obtain the maximum x and y coordinates in the current graphics mode.

In the above functions one function shows a code on to display a mouse pointer ('showmouseptr()') and the other to get the mouse position i.e. it's coordinates and also its button status ('getmousepos()').The way these functions process is given in the comment lines in the code.

start()

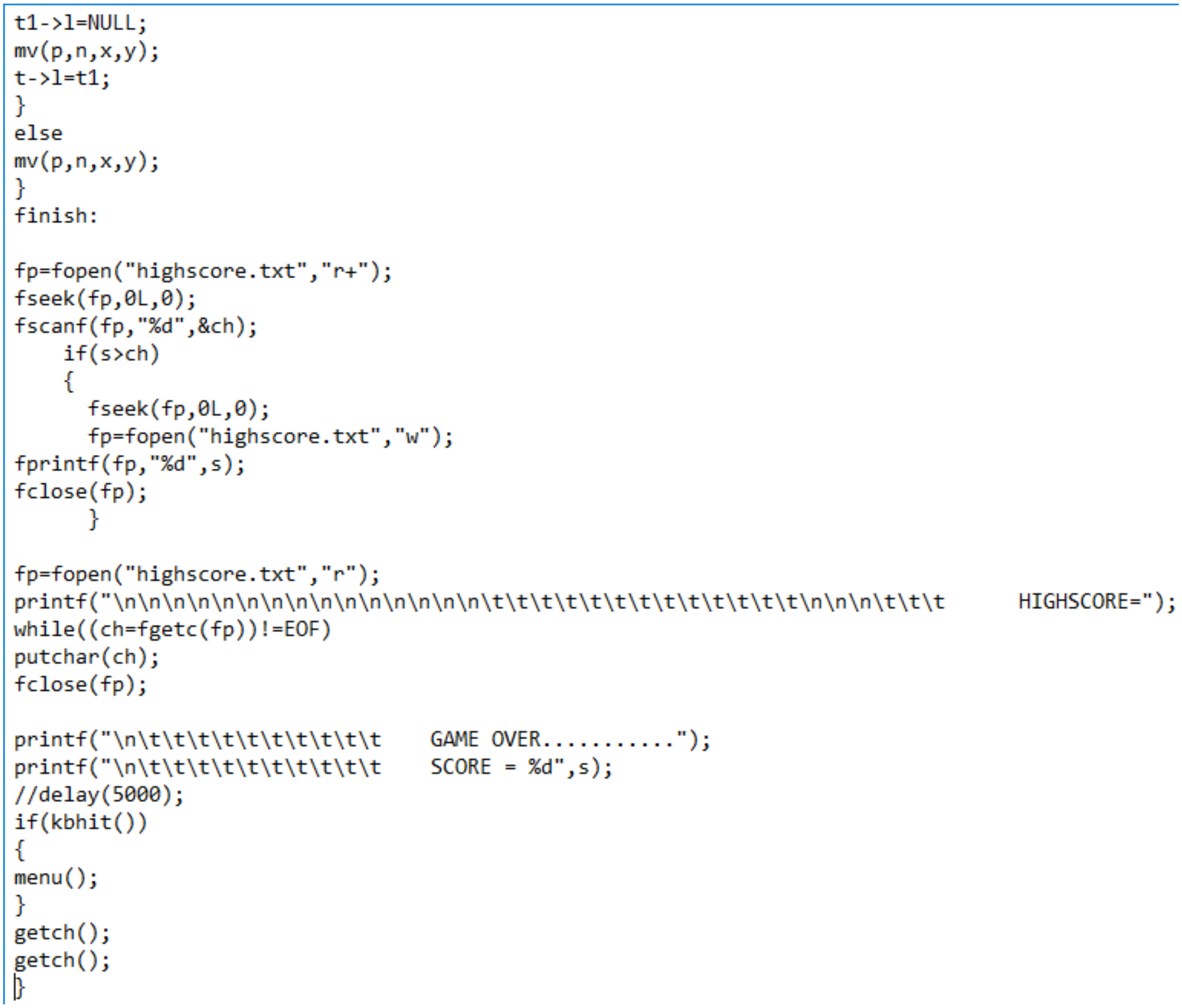
1. In the menu function inside the infinite while loop if the first condition is true then the control is transferred to start() function.
2. Two structure pointer variables p and t are declared under the structure snk.
3. The graphics driver is initialized again using the initgraph function and the coordinates of the two extreme coordinates is found using the getmaxx and getmaxy functions.
4. Then the main gameplay background is set to green color.
5. Memory is allocated to the structure pointer p using malloc function and it is initialized to coordinates (20, 20) and the direction is set to right using d data member of the structure variable p.



```
void start()
{
clrscr();
struct snk *p,*t,*t1;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
int n,f1,f2,x,y,i=0;
x=getmaxx();
y=getmaxy();
char c;
setbkcolor(GREEN);
p=(struct snk *)malloc(sizeof(struct snk));
p->cx=20;
p->cy=20;
p->d=77;
p->l=NULL;
f1=random(32)*20;
f2=random(24)*20;
rec(f1,f2,f1+20,f2+20,BLUE);
rec(p->cx,p->cy,p->cx+20,p->cy+20,RED);
while(1)
{
rec(f1,f2,f1+20,f2+20,BLUE);
rec(p->cx,p->cy,p->cx+20,p->cy+20,RED);
delay(w);
if((p->l)!=NULL)
{
if((p->l->l)!=NULL)
{
if(ck(p->l->l,p->cx,p->cy))
goto finish;
}
}
t=p;
while((t->l)!=NULL)
t=t->l;
```

6. Two random numbers are stored in f1 and f2 which are used as coordinates to display food rectangle using rec function. The color of the rectangle is set to blue.

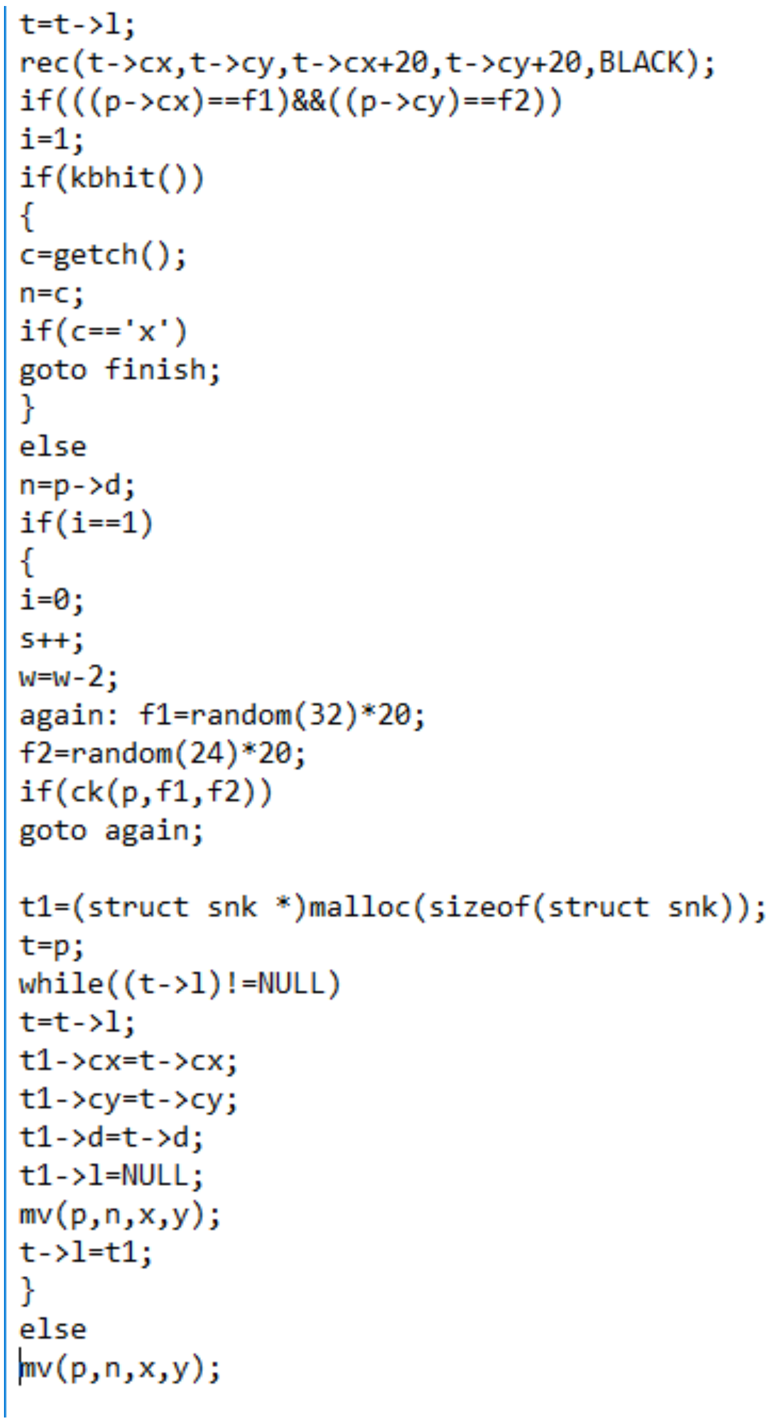
7. The rectangle of the snake is printed by passing the coordinates and the RED color through the rec function.
8. The delay function is used with a variable in order to increase the speed of the snake as the game proceeds.
9. The conditions check whether the length of the snake is at least two rectangles long and the last condition transfers the control to **ck** function which checks if the snakes head coincides with any part of the body. If the condition returns true then the control is transferred to the finish tag using the goto function.
10. The p structure variable is copied into a temporary structure variable t to print the snake but this time with black color which gives the impression of the snake disappearing.
11. If the coordinates of the snakes head coincide with the coordinates of the food **i** variable is initialized to 1 for further use.
12. The kbhit function returns true when a key is pressed.
13. When the user presses a key this condition becomes true and the value of the key is stored in the c variable.
14. If the user presses 'x' then the program is skipped to the finish tag using goto function.
15. Else if he presses any other key and if the value of **i** initialized before is 1 then it is reinitialized to zero and the score variable s is increased by 1 and the speed variable w is decreased by a value of two random variables are stored in f1 and f2 variables which are used in printing the food rectangle.
16. The p structure variable and f1 and f2 variables are passed into the **ck** function which returns 1 when the head touches the food. If this becomes true then the control transfers back to again tag and the random variables are stored again to check for coincidence.



```
t1->l=NULL;
mv(p,n,x,y);
t->l=t1;
}
else
mv(p,n,x,y);
}
finish:

fp=fopen("highscore.txt","r+");
fseek(fp,0L,0);
fscanf(fp,"%d",&ch);
    if(s>ch)
    {
      fseek(fp,0L,0);
      fp=fopen("highscore.txt","w");
fprintf(fp,"%d",s);
fclose(fp);
      }

fp=fopen("highscore.txt","r");
printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\t\t\t\t\n\n\n\t\t\t      HIGHSCORE=");
while((ch=fgetc(fp))!=EOF)
putchar(ch);
fclose(fp);

printf("\n\t\t\t\t\t\t\t\t\t\t    GAME OVER...........");
printf("\n\t\t\t\t\t\t\t\t\t\t    SCORE = %d",s);
//delay(5000);
if(kbhit())
{
menu();
}
getch();
getch();
}
```

17. Memory is allocated to a new structure variable t1 using malloc function and p structure variable is inserted into it and it is traversed till the end in the while loop.
18. The last coordinates of t are stored in the t1 variable and the mv function is called to add another node into the snake and update the values.
19. The t variable is moved by another node and the t1 variables are stored in it.

20. If **i** is not equal to 1 then the else part is executed where the coordinates of p are updates by calling the mv function.
21. Inside the finish tag the current highscore is compared with the previously stored highscore and if it is more it is stored inside the file.



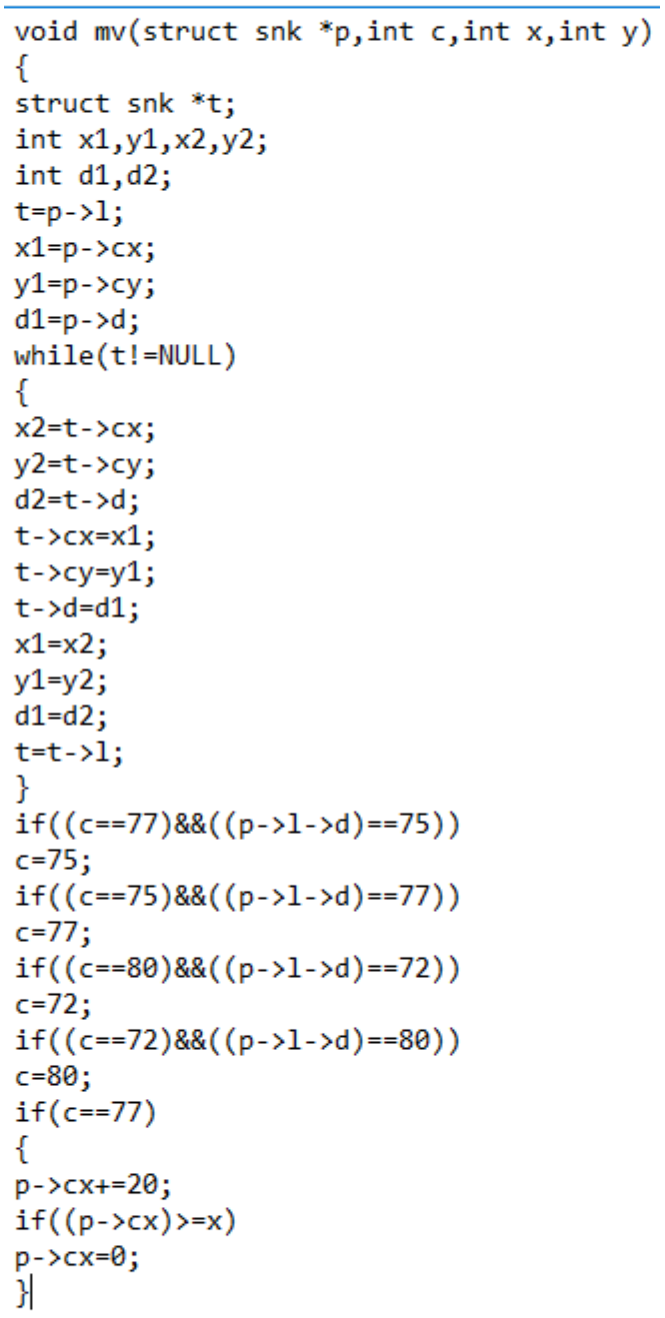
```
t=t->l;
rec(t->cx,t->cy,t->cx+20,t->cy+20,BLACK);
if(((p->cx)==f1)&&((p->cy)==f2))
i=1;
if(kbhit())
{
c=getch();
n=c;
if(c=='x')
goto finish;
}
else
n=p->d;
if(i==1)
{
i=0;
s++;
w=w-2;
again: f1=random(32)*20;
f2=random(24)*20;
if(ck(p,f1,f2))
goto again;

t1=(struct snk *)malloc(sizeof(struct snk));
t=p;
while((t->l)!=NULL)
t=t->l;
t1->cx=t->cx;
t1->cy=t->cy;
t1->d=t->d;
t1->l=NULL;
mv(p,n,x,y);
t->l=t1;
}
else
mv(p,n,x,y);
```

ck()

1. The ck function accepts a snk structure variable and two integer variables as parameters.
2. A temporary snk structure pointer variable is created in which the address of the structure variable passed as a parameter is stored.
3. An integer variable f is created and is initialized to 0.
4. Inside the while loop an if condition checks if the coordinates stored in the t structure variable coincide with the integer parameters passed into the function and if the condition becomes true the value of f variable is changed to 1.
5. The while loop is executed till it reaches to the end of the linked list or when the f value changes to 1.
6. After the while loop is exited the value of f is returned to the calling function.

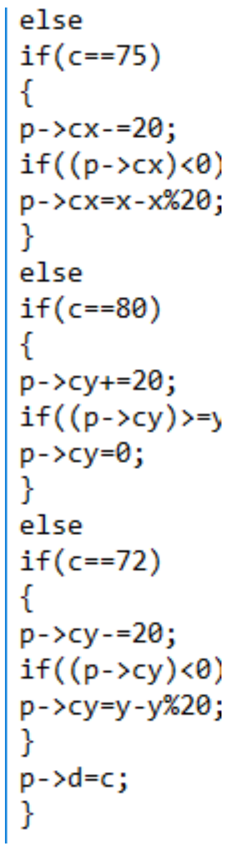
mv()

1. The mv function accepts a snk structure variable as the parameter along with 3 integer variables.
2. A temporary snk structure pointer variable is created which stores the value of the passed structure variable from the second node.
3. 6 integer variables x1, x2, y1, y2, d1, d2 are created. The coordinates stored in the first node of the passed structure variable are stored in x1 and y1 variables and the direction d value is stored in d1.



```
void mv(struct snk *p,int c,int x,int y)
{
struct snk *t;
int x1,y1,x2,y2;
int d1,d2;
t=p->l;
x1=p->cx;
y1=p->cy;
d1=p->d;
while(t!=NULL)
{
x2=t->cx;
y2=t->cy;
d2=t->d;
t->cx=x1;
t->cy=y1;
t->d=d1;
x1=x2;
y1=y2;
d1=d2;
t=t->l;
}
if((c==77)&&((p->l->d)==75))
c=75;
if((c==75)&&((p->l->d)==77))
c=77;
if((c==80)&&((p->l->d)==72))
c=72;
if((c==72)&&((p->l->d)==80))
c=80;
if(c==77)
{
p->cx+=20;
if((p->cx)>=x)
p->cx=0;
}
```

4. Inside the while loop, the coordinates and the direction variable d in the t temporary structure variable are stored inside x2, y2 and d2 respectively. Then the values stored in x1, y1 and d1 are copied into t and are assigned with values in x2, y2 and d2 right after.
5. The while loop is continued till the end of the t linked list.

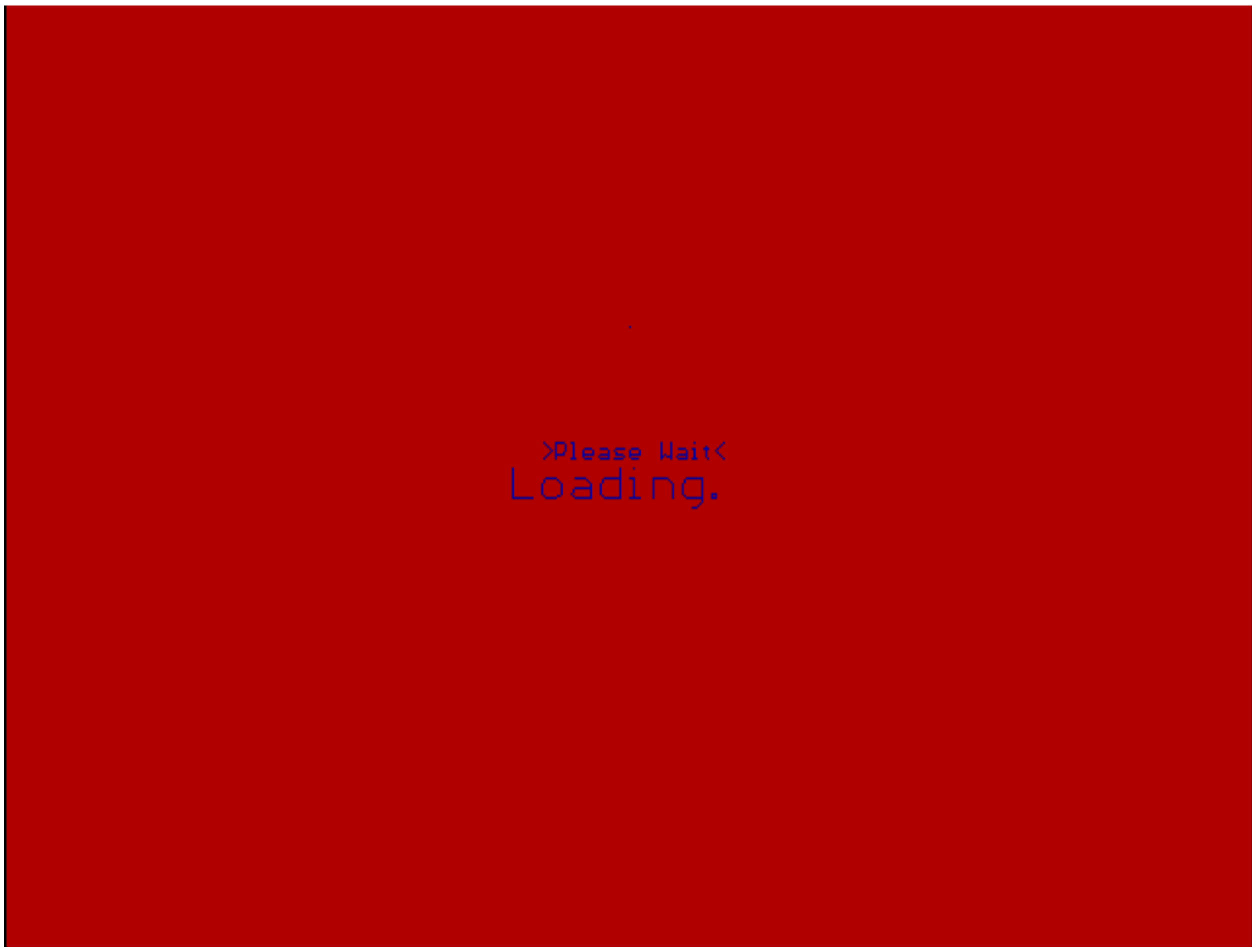


```
else
if(c==75)
{
p->cx-=20;
if((p->cx)<0)
p->cx=x-x%20;
}
else
if(c==80)
{
p->cy+=20;
if((p->cy)>=y
p->cy=0;
}
else
if(c==72)
{
p->cy-=20;
if((p->cy)<0)
p->cy=y-y%20;
}
p->d=c;
}
```

6. Now the pressed key condition is checked through the passed c integer variable which stores the ASCI value of the key pressed and the value of direction and the coordinates in the structure variable are changed accordingly.

7. OUTPUT

This is the loading screen displayed when the program gets executed.



```
>Please Wait<
Loading.
```

Fig 7.1 LOADING SCREEN

As the loading screen disappears, the menu screen is displayed as follows. When the program is executed for the first time high score is not printed as the file containing highscore is initially set to end of file.

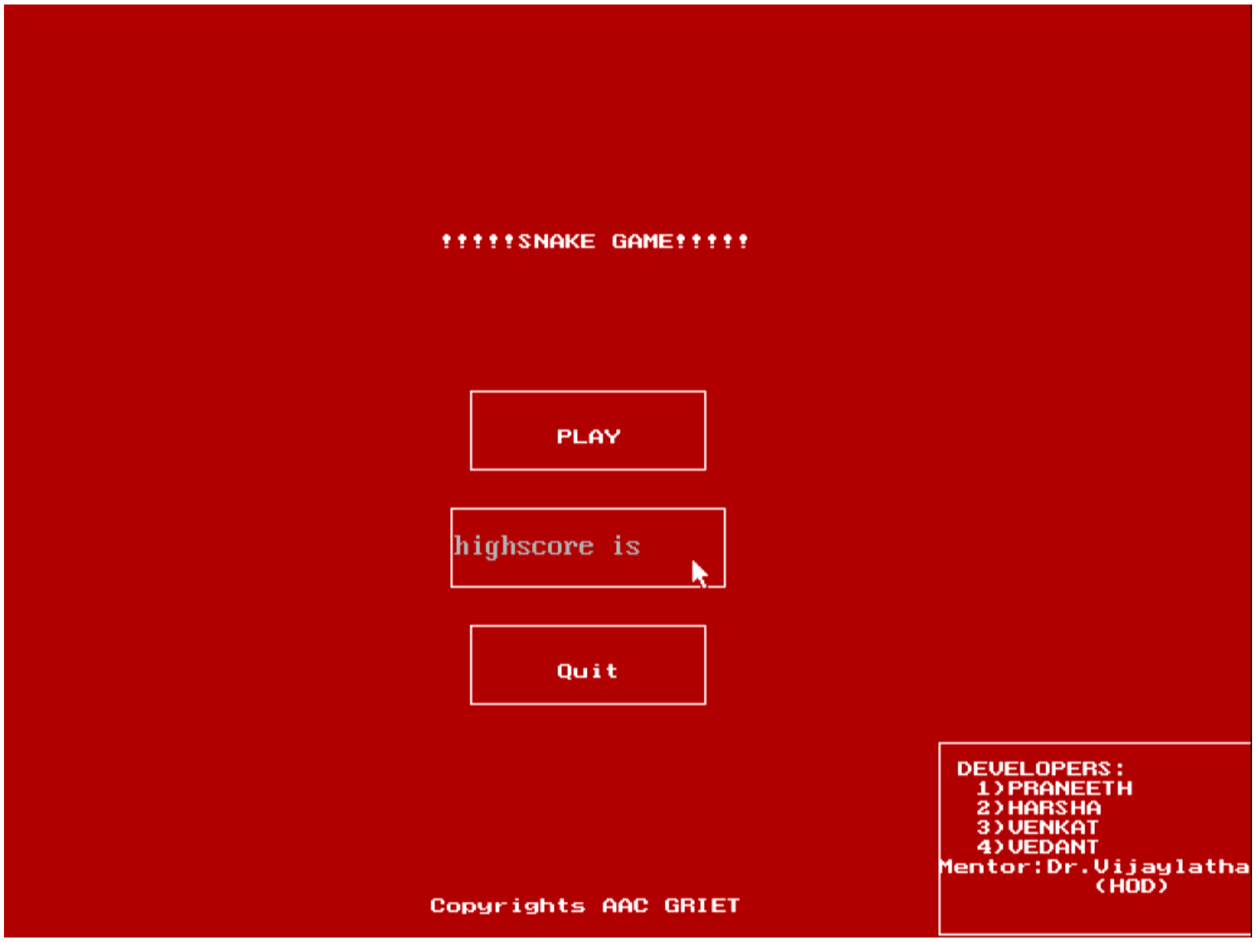


Fig 7.2 MENU SCREEN

This is the game play screen displayed when the user selects play option from menu. The red box which is the snake moves continuously towards right as the initial movement of the snake is set to right and the blue box which is the food, is initially displayed at the position.

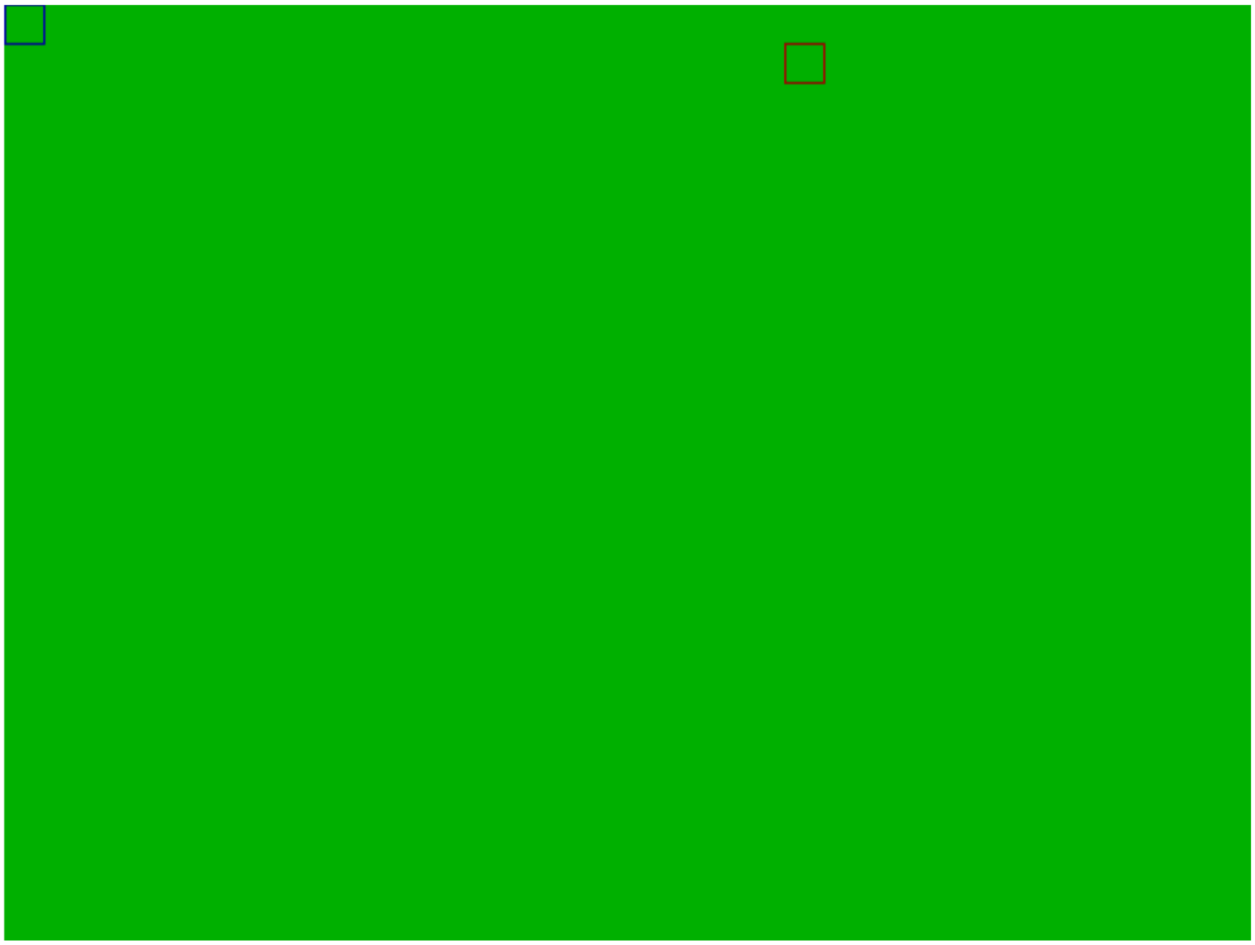


Fig 7.3 GAME PLAY

This is the game over screen displayed when the user moves the snake in the opposite direction.

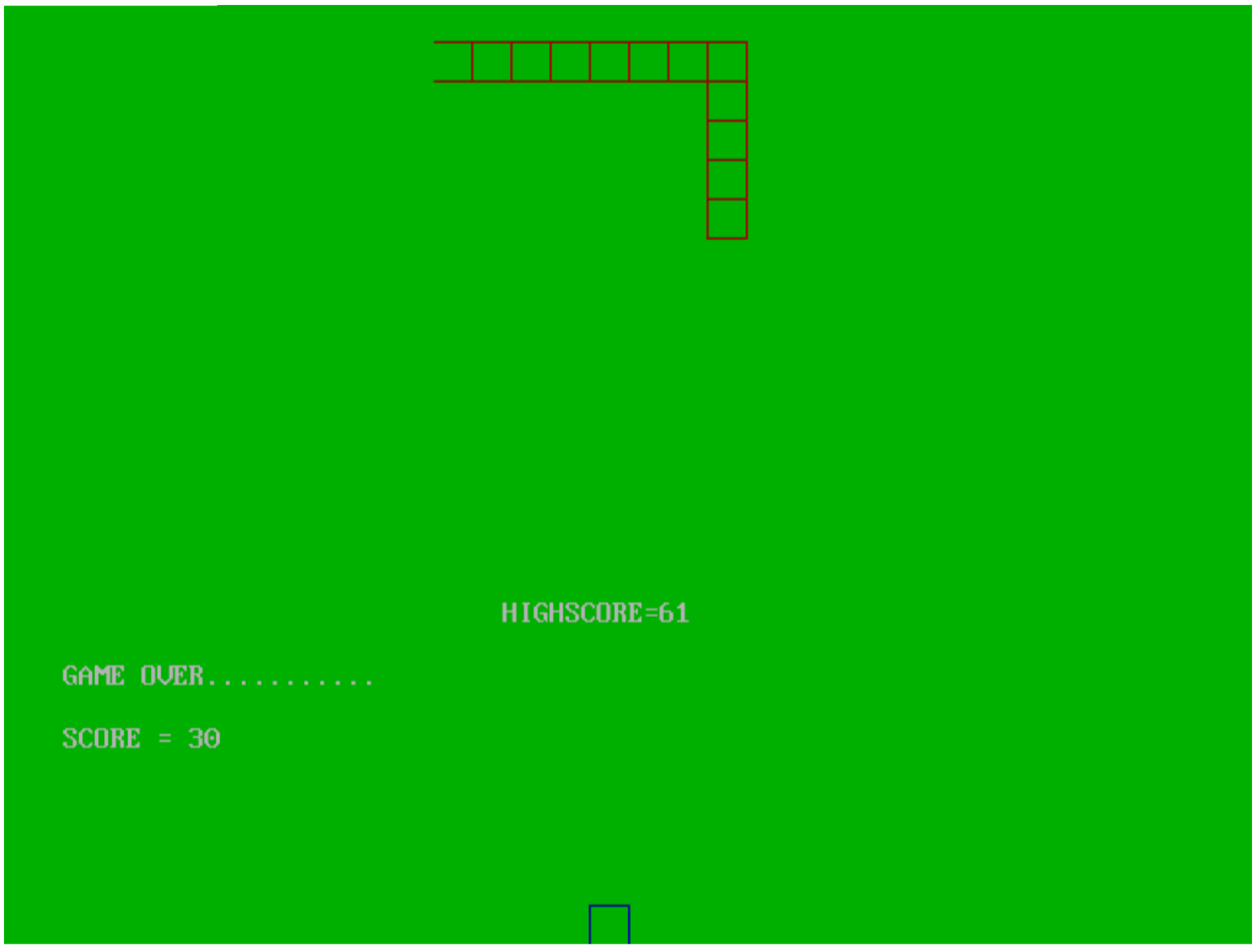
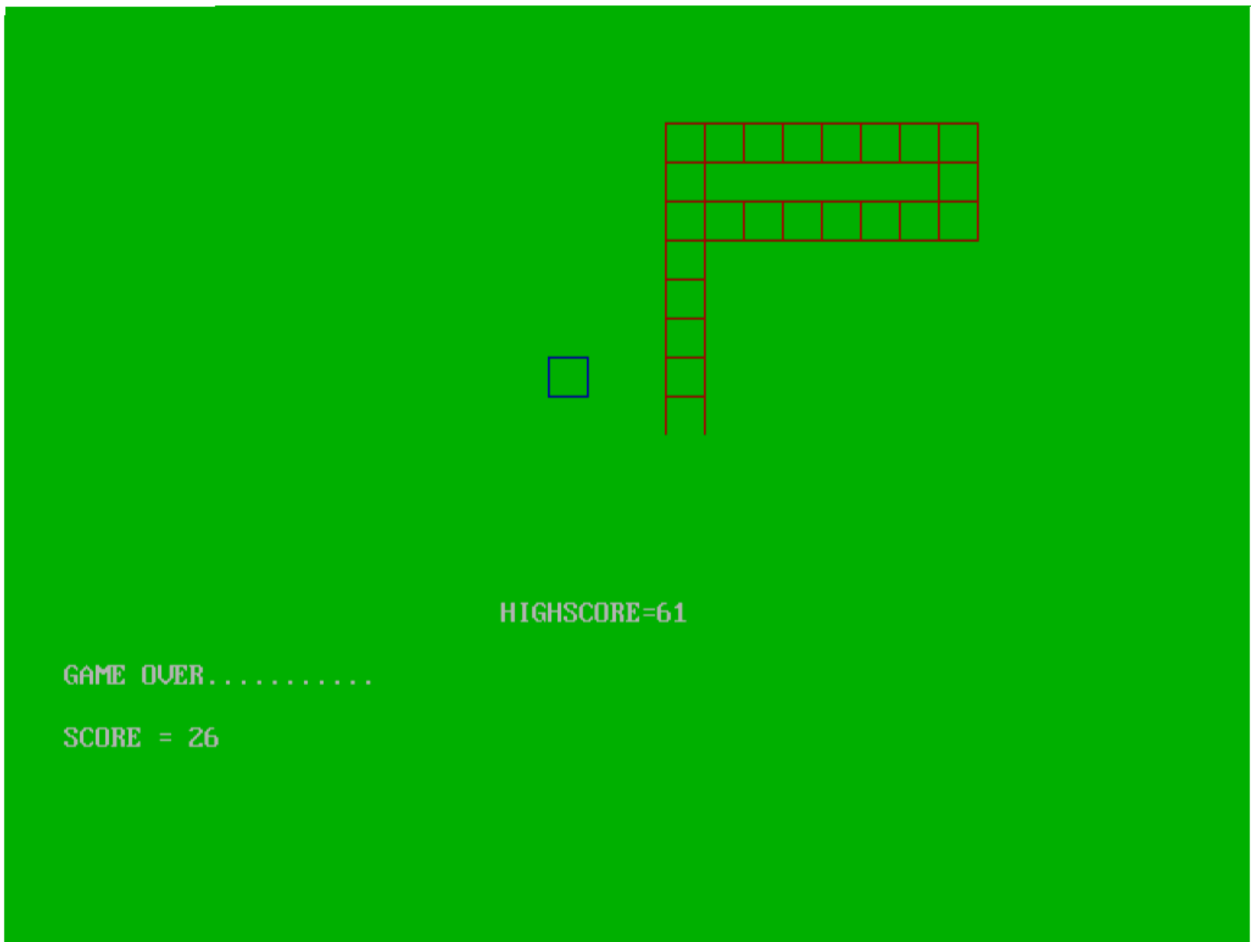


Fig 7.4 GAME OVER - I

This is the game over screen displayed when the snake touches its tail.



HIGHSCORE=61

GAME OVER...........

SCORE = 26

Fig 7.5 GAME OVER - II

As soon as the game is over the user is redirected to the menu screen with updated high score.

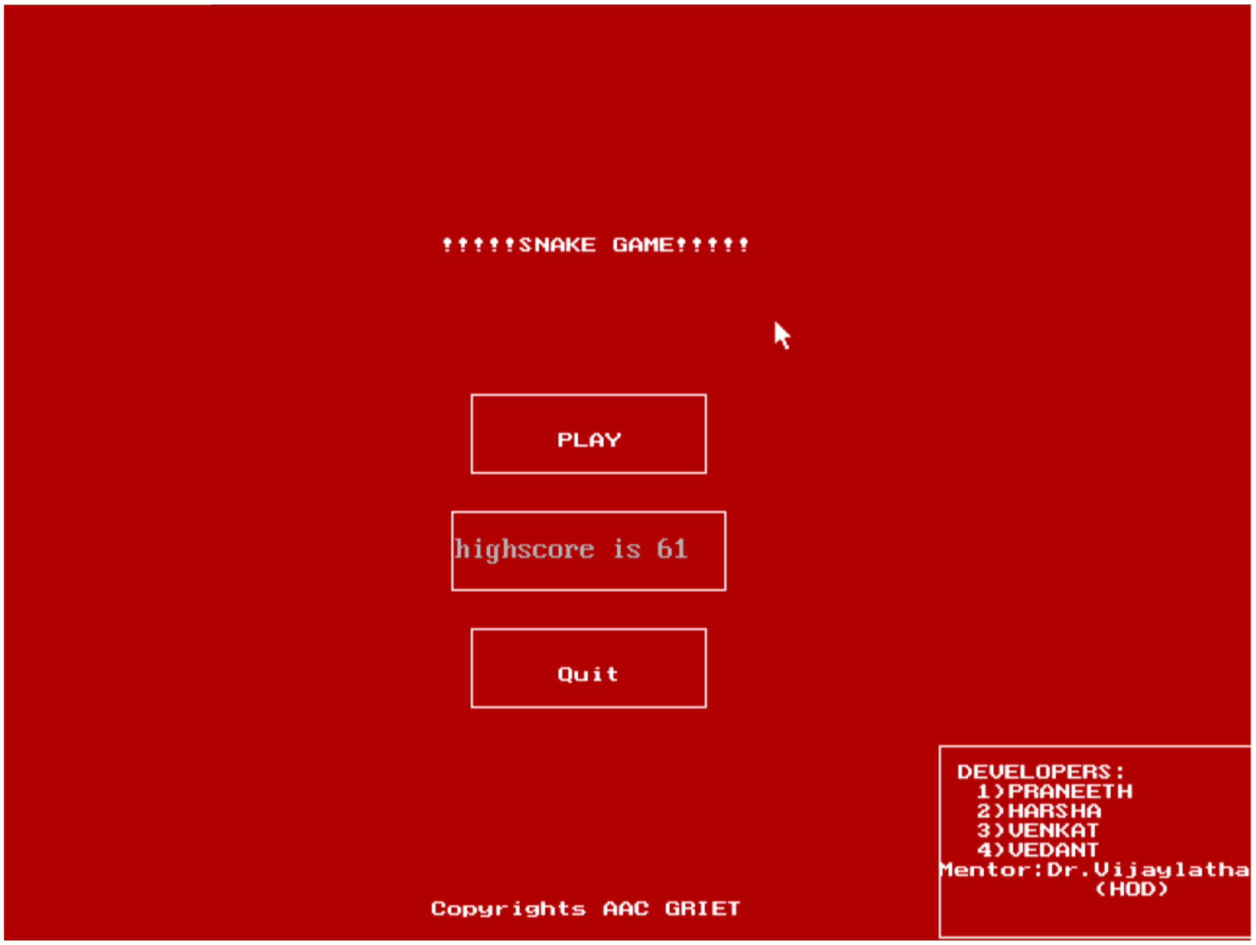


Fig 7.6 HIGHSCORE MENU

8. CONCLUSION

Hence a basic game like snake can be developed using data structures and graphics in c. This can be made better using in terms of graphics using visual C++ or java and other similar modern languages.

Future Implementations:

Using the concept of Data Structures, the same game can be implemented using any programming language. The game can be taken to next level by including some extra features like levels, mazes, stop watch, power-up’s etc.,

9. REFERENCES

Text books Referred:

1. Graphics under C by **Yeshwanth Kanethkar**

Sites Referred:

1. www.geekforgeeks.com

2. www.tutorialspoint.com