**BALLOON SHOOTING**

*A report submitted in partial*



Department of Computer Science and Engineering

KL University, Vaddeswaram, Guntur Dt.

2013-14

.

**ABSTRACT**

This is a simple balloon shooting game made in C language. In this game, there are number of balloons, a bow and arrows for shooting the balloons. The balloons rise from the lower portion of the screen. The bow is put on left side which can move up and down. If you shoot a balloon you will get 20 points. There are total three levels in our game. There are 10 balloons and 6 arrows in first level. If you get a minimum score of 100, you will be eligible for entering in second level. If you not get this minimum score, you are not qualified for second level and game will be over and you are looser. There are 8 balloons and 6 arrows in second level. If you get a minimum score of 180, you will be eligible for entering in third level. In third level there are 6 balloons and 6 arrows.. If you completed all the three levels, you will be winner. The up and down arrow key are used for positioning the bow. The right arrow key will be used for shooting.

Modules

There are 4 modules in our project

Module 1:

**This module guides the user the controls of the game.**

**Module 2:**

**This module instructs the user different levels of the game.**

**Module 3:**

**This module is about the score the player gains and the level of difficulty he passes through each level.**

**Module 4:**

**This module displays the final score the player made at the end of the game.**

**TABLE OF CONTENTS**

**1.INTRODUCTION 7**

**2.PROBLEMS DEFINITION 13**

**3.REQUIREMENTS 15**

**4.IMPLEMENTATION 17**

**5.OUTPUT SCREEN SHOTS 37**

**6.CONCLUSION 40**

**7.REFERENCES 41**

**1.INTRODUCTION**

In computing, **C** is a general-purpose programming language initially developed by Dennis Ritchie between 1969 and 1973 at AT&T bell laboratories. Its design provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, most notably system software like the Unix computer operating system

C is one of the most widely used programming languages of all time, and there are very few computer architectures for which a C compiler does not exist.

Many later languages have borrowed directly or indirectly from C, including C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Python, and Unix's C Shell. The most pervasive influence on these languages (excluding Python) has been syntactical, and they tend to combine the recognizable expression and statement syntax of C with underlying type systems, data models, and semantics that can be radically different. C++ started as a pre-processor for C and is currently nearly a superset of C

Before there was an official standard for C, many users and implementors relied on an informal specification contained in a book by Dennis Ritchie and Brian Kernighan; that version is generally referred to as "K&R" C. In 1989 the American National Standards Institute published a standard for C (generally called "ANSI C" or "C89"). The next year, the same specification was approved by the International Organization for Standardization as an international standard (generally called "C90"). ISO later released an extension to the internationalization support of the standard in 1995, and a revised standard (known as "C99") in 1999. The current version of the standard (now known as "C11") was approved in December of 2011.

Facts about C:-

* C was invented to write an operating system called UNIX.
* C is a successor of B language which was introduced around 1970
* The language was formalized in 1988 by the American National Standard Institute (ANSI).
* By 1973 UNIX OS almost totally written in C.
* Today C is the most widely used System Programming Language.
* Most of the state of the art software have been implemented using C
* Easy to learn
* Structured language
* It produces efficient programs.

It can handle low-level activities.

* It can be compiled on a variety of computers.

Why to use C?

C was initially used for system development work, in particular the programs that make-up the operating system. C was adoped as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

* Operating Systems
* Language Compilers
* Assemblers
* Text Editors
* Print Spoolers
* Network Drivers
* Modern Programs
* Data Bases
* Language Interpreters
* Utilities

Design:-

C is an imperative (procedural) language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. C was therefore useful for many applications that had formerly been coded in assembly language, such as in system programming.

Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with few changes to its source code.

The language has become available on a very wide range of platforms, from embedded microcontroller

 Supercomputers.

Characteristics of C language:-

Like most imperative languages in the ALGOL tradition, C has facilities for structured programming and allows lexical variable scope and [recursion](http://en.wikipedia.org/wiki/Recursion), while a static type system prevents many unintended operations. In C, all executable code is contained within subroutines, which are called "functions" (although not in the strict sense of functional programming). Function parameters are always passed by value. Pass-by-reference is simulated in C by explicitly passing [pointer](http://en.wikipedia.org/wiki/Pointer_(computer_programming)) values. C program source text is [free-format](http://en.wikipedia.org/wiki/Free-form_language), using the [semicolon](http://en.wikipedia.org/wiki/Semicolon) as a [statement](http://en.wikipedia.org/wiki/Statement_(programming)) terminator and [curly braces](http://en.wikipedia.org/wiki/Curly_braces#Curly_brackets_.7B_.7D) for grouping blocks of statements.

The C language also exhibits the following characteristics:

* There is a small, fixed number of keywords, including a full set of [flow of control](http://en.wikipedia.org/wiki/Flow_of_control) primitives: [for](http://en.wikipedia.org/wiki/For_loop), [if/else](http://en.wikipedia.org/wiki/Conditional_(programming)), [while](http://en.wikipedia.org/wiki/While_loop), [switch](http://en.wikipedia.org/wiki/Switch_statement), and [do/while](http://en.wikipedia.org/wiki/Do_while_loop). There is basically one [namespace](http://en.wikipedia.org/wiki/Namespace), and user-defined names are not distinguished from keywords by any kind of [sigil](http://en.wikipedia.org/wiki/Sigil_(computer_programming)).
* There are a large number of arithmetical and logical operators, such as +, +=, ++, &, ~, etc.
* More than one [assignment](http://en.wikipedia.org/wiki/Assignment_(computer_science)) may be performed in a single statement.
* Function return values can be ignored when not needed.
* Typing is [static](http://en.wikipedia.org/wiki/Static_typing), but [weakly enforced](http://en.wikipedia.org/wiki/Weak_typing): all data has a type, but implicit conversions can be performed; for instance, characters can be used as integers.
* [Declaration](http://en.wikipedia.org/wiki/Declaration_(computer_programming)) [syntax](http://en.wikipedia.org/wiki/C_syntax) mimics usage context. C has no "define" keyword; instead, a statement beginning with the name of a type is taken as a declaration. There is no "function" keyword; instead, a function is indicated by the parentheses of an argument list.
* User-defined (typedef) and compound types are possible.
  + Heterogeneous aggregate data types ([struct](http://en.wikipedia.org/wiki/Struct_(C_programming_language))) allow related data elements to be accessed and assigned as a unit.
  + Array indexing is a secondary notion, defined in terms of pointer arithmetic. Unlike structs, arrays are not first-class objects; they cannot be assigned or compared using single built-in operators. There is no "array" keyword, in use or definition; instead, square brackets indicate arrays syntactically, e.g. month
  + [Enumerated types](http://en.wikipedia.org/wiki/Enumerated_type) are possible with the enum keyword. They are not tagged, and are freely interconvertible with integers.
  + [Strings](http://en.wikipedia.org/wiki/String_(computer_science)) are not a separate data type, but are conventionally implemented as [null-terminated](http://en.wikipedia.org/wiki/Null-terminated_string) arrays of characters.
* Low-level access to computer memory is possible by converting machine addresses to typed [pointers](http://en.wikipedia.org/wiki/Pointer_(computer_programming)).
* Procedures (subroutines not returning values) are a special case of function, with an untyped return type void.
* Functions may not be defined within the lexical scope of other functions.
* Function and data pointers permit *ad hoc* [run-time polymorphism](http://en.wikipedia.org/wiki/Type_polymorphism).
* A [pre-processor](http://en.wikipedia.org/wiki/C_preprocessor) performs [macro](http://en.wikipedia.org/wiki/Macro_(computer_science)) definition, [source code](http://en.wikipedia.org/wiki/Source_code) file inclusion, and [conditional compilation](http://en.wikipedia.org/wiki/Conditional_compilation).
* There is a basic form of [modularity](http://en.wikipedia.org/wiki/Modular_programming): files can be compiled separately and [linked](http://en.wikipedia.org/wiki/Linker_(computing)) together, with control over which functions and data objects are visible to other files via static and extern attributes.
* C is a case sensitive programming language. It means in C *printf* and *Printf* will have different meanings.
* C has a free-form line structure. End of each C statement must be marked with a semicolon.
* Multiple statements can be one the same line.
* White Spaces (i.e. tab space and space bar) are ignored.
* Statements can continue over multiple lines.

C does not include some features found in newer, more modern high-level languages, including [object orientation](http://en.wikipedia.org/wiki/Object-oriented_programming) and [garbage collection](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)).

Token:-The smallest individual unit in program is called as a token. The C tokens are classified as keywords, identifiers, constants, operators, strings and some special symbols.

### Character set:-

The basic C source character set includes the following characters:

* Letters: a–z, A–Z, \_
* Digits: 0–9
* Punctuation: ~! @ # % ^ & \* ( ) - + = : ; " ' <> , . ? | / \ { } [ ]
* Whitespace characters: space, horizontal tab, vertical tab, form feed, newline.

Keywords and Identifiers:-In C every word is classified into either a keyword or an identifier. Keywords have fixed meaning. There are 32 keywords.

|  |  |  |  |
| --- | --- | --- | --- |
| Auto  break  case  char  const  continue  default  do | double  else  enum  extern  float  for  goto  if | Int  long  register  return  short  signed  sizeof  static | struct  switch  typedef  union  unsigned  void  volatile  while |

Identifiers:-Identifiers are user defined names consisting of sequence of letters and digits used to refer the names of the variables, functions etc.

Libraries:-The C programming language uses libraries as its primary method of extension. In C, a library is a set of functions contained within a single "archive" file. Each library typically has a header file, which contains the prototypes of the functions contained within the library that may be used by a program, and declarations of special data types and macro symbols used with these functions. In order for a program to use a library, it must include the library's header file, and the library must be linked with the program, which in many cases requires compiler flags The most common C library is the C standard library, which is specified by the ISO and ANSI C standards and comes with every C implementation. This library supports stream input and output, memory allocation, mathematics, character strings, and time values. Several separate standard headers specify the interfaces for these and other standard library facilities.

C-Program Structure:-A C program basically has the following form:

* Pre-processor Commands
* Functions
* Variables
* Statements & Expressions
* Comments

Preprocessor Commands: - These commands tell the compiler to do preprocessing before doing actual compilation. Like *#include <stdio.h>* is a preprocessor command which tells a C compiler to include stdio.h file before going to actual compilation.

Functions:-Functions are main building blocks of any C Program. Every C Program will have one or more functions and there is one mandatory function which is called main() function. This function is prefixed with keyword int which means this function returns an integer value when it exits. This integer value is returned using return statement.

The C Programming language provides a set of built-in functions. In the above example printf () is a C built-in function which is used to print anything on the screen.

Variables:- Variablesare used to hold numbers, strings and complex data for manipulation

**2.PROBLEM DEFINITON**

This chapter gives the information of the header files and the functions used in this project. The header files used are stdio.h, conio.h,dos.h

**STDIO.H**: The C programming language provides many standard library functions for file input and output. These functions make up the bulk of the C standard library header <stdio.h>.It’s full form is Standard Input Output. This header file is used to give the input and output functions to the program. Some of the functions that are included in the stdio library are:-

printf - prints the formatted byte/matter

scanf - reads the formatted byte/matter

**CONIO.H**:**conio.h** is a C header file used mostly by MS-DOS compilers to provide console input/output. It’s full form is Console Input Output.This header declares several useful library functions for performing "console input and output" from a program the member functions that are included in the library are:-

Getch - Reads a character directly from the console without buffer, and without echo

Clrscr - Clears the screen.

**DOS.H:**

 Dos.h header file of c language contains functions for handling interrupts, producing sound, date and time functions etc. It is borland specific and works in turbo c compiler.Some of the functions that are present in dos.h are:-

[delay](http://www.programmingsimplified.com/c/dos.h/delay) sleep

[getdate](http://www.programmingsimplified.com/c/dos.h/getdate) sound

[gettime](http://www.programmingsimplified.com/c/dos.h/gettime)

[nosound](http://www.programmingsimplified.com/c/dos.h/nosound)

[setdate](http://www.programmingsimplified.com/c/dos.h/setdate)

The non-repetitive statements that are used in our program are switch statement and simple if statement. The repetitive statement that is used is do while loop.

Simple If Statement:-the simple if statement is a conditional control statement used to check the condition. If the condition is true ,the statement will execute otherwise no statement will execute.

Syntax:- if(condition)

Statement;

Switch Statement:-The purpose of a switch statement is to allow the value of a variable or expression to control the flow of program execution through a multiway branch. The main reasons for using a switch include improving clarity, by reducing otherwise repetitive coding, and also offering the potential for faster execution through easier compiler optimization in many cases.

Syntax:- switch(expression)

{

Case ‘constant identifier 1’:

Statement;

Break;

Case ‘constant identifier 2’:

Statement;

Break;

.

.

.

Default:

Default statement;

}

**3.** SOFTWARE AND HARDWARE SPECIFICATIONS

Computer Hardware: Refers to a set of physical components of the computer and its related devices. These include RAM, Hard drive, etc…

Computer software: Include the operating system and all the utilities that enable the function to perform.

Software and hardware configuration of our system is:

SOFTWARE**:**

Operating System: windows 7 professional N.

Language: C language.

Software: Turbo C 3.0(DOSBOX)

HARDWARE**:**

Processor: AMD Phenon™IIX3170 processor 2.60Hz.

Installed Memory: 4.00GB (3.46GB usable).

System type: 32-bit operating system.

Pen and Touch: No pen or touch input is available for this display.

Hard Disk: 320GB

4.IMPLEMENTATION

# include "graphics.h"

# include "conio.h"

# include "stdio.h"

# include "stdlib.h"

# include "dos.h"

#define ARROW\_SIZE 7

#define BALLOON\_SIZE 7

Intflag\_arrow=0, flag\_balloon=1,count\_arrow=6,count\_balloon=10;

void \*balloon,\*bow,\*arrow,\*burst;

void \*clear\_balloon,\*clear\_burst;

void draw\_balloon(int ,int );

void draw\_burst ( int x, int y );

void draw\_ bow(int x,int y);

void draw\_arrow(int x, int y);

void shoot(int \*x, int \*y);

int testkeys();

void fly(int \*x, int \*y);

void start();

void main()

{

int maxx = getmaxx() ;

int maxy = getmaxy() ;

int p=400,q=300,m=100,n=100,x=m,y=n,key,score=0,finish=0,level=1,l\_flag=1;

char score1[5],ch,cnt\_ball[5],char\_level[2];

int gmode = DETECT, gdriver , area ;

initgraph ( &gmode, &gdriver, " " ) ;

setbkcolor(1);

start();

rectangle ( 0, 0, maxx, maxy - 10 ) ;

draw\_burst(200,300);

area=imagesize(0,0,32,24);

burst=malloc(area);

getimage(200-16,300-12,200+16,300+12,burst);

putimage(200-16,300-12,burst,XOR\_PUT);

draw\_balloon(p,q);

area=imagesize(p-4\*BALLOON\_SIZE,q-5\*BALLOON\_SIZE,p+4\*BALLOON\_SIZE,q+7\*BALLOON\_SIZE);

balloon=malloc(area);

getimage(p-4\*BALLOON\_SIZE,q-5\*BALLOON\_SIZE,p+4\*BALLOON\_SIZE,q+7\*BALLOON\_SIZE,balloon);

putimage(p-4\*BALLOON\_SIZE, q-5\*BALLOON\_SIZE, balloon, COPY\_PUT);

draw\_arrow(x ,y );

area = imagesize(x, y-ARROW\_SIZE, x+(6\*ARROW\_SIZE), y+ARROW\_SIZE);

arrow=malloc(area);

getimage(x, y-ARROW\_SIZE, x+(6\*ARROW\_SIZE), y+ARROW\_SIZE,arrow);

putimage(x, y-ARROW\_SIZE,arrow,XOR\_PUT);

draw\_bow(x,y);

area=imagesize(x+25,y-65,x+66,y+65);

bow=malloc(area);

getimage(x+25,y-65,x+66,y+65,bow);

if ( balloon == NULL || burst == NULL || bow == NULL )

{

printf( "Insufficient memory... Press any key " ) ;

getch() ;

closegraph() ;

restorecrtmode() ;

exit( 0 ) ;

}

while (!finish)

{

settextstyle(8,0,1);

setusercharsize(4,4,3,3);

outtextxy(getmaxx()/2-100,5,"LEVEL : ");

itoa(level,char\_level,10);

setfillstyle(0,0);

bar(getmaxx()/2+40,15,getmaxx()/2+70,45);

outtextxy(getmaxx()/2+50,5,char\_level);

rectangle(5,360,250,460);

if( flag\_balloon && count\_balloon>0 )

fly( &p, &q );

else

{

q = 400;

flag\_balloon = 1;

}

if( kbhit() )

{

key = testkeys();

if(key==77)

flag\_arrow = 1;

}

if( key == 27 )

break;

if (key == 80 &&!flag\_arrow)

{

x=125;

putimage(x,y-65,bow,XOR\_PUT);

if(y<300)

y+=25;

putimage(x,y-65,bow,XOR\_PUT);

draw\_bow(x-25,y);

key=0;

}

if (key == 72 &&!flag\_arrow)

{

x=125;

putimage(x,y-65,bow,XOR\_PUT);

if(y>70)

y-=25;

putimage(x,y-65,bow,XOR\_PUT);

draw\_bow(x-25,y);

key=0;

}

if(count\_arrow > 0 && count\_balloon > 0)

{

if(score==100 && l\_flag==1)

{

level=2;

count\_balloon=8;

count\_arrow=6;

l\_flag=2;

}

if(score==180 && l\_flag==2)

{

level=3;

count\_balloon=6;

count\_arrow=6;

l\_flag=0;

}

if( key == 77 || flag\_arrow)

{

shoot(&x,&y);

draw\_bow(m,y);

if(x>(p-12) && x<(p+12) && y>(q-15) && y<(q+25))

{

putimage(p-16,q-12,burst,COPY\_PUT);

sound(1500);

delay(800);

nosound();

putimage(p-16,q-12,burst,XOR\_PUT);

count\_balloon--;

settextstyle(10,0,1);

setusercharsize(30,70,20,70);

outtextxy(20,380,"BALLOONS LEFT:");

setfillstyle(0,0);

bar(200,370,230,400);

itoa(count\_balloon,cnt\_ball,10);

outtextxy(200,380,cnt\_ball);

flag\_balloon=0;

score+=10;

itoa(score,score1,10);

setfillstyle(0,0);

bar(190,getmaxy()-50,230,getmaxy()-30);

setcolor(RED);

outtextxy(20,getmaxy()-50,"SCORE : ");

outtextxy(190,getmaxy()-50,score1);

}

key=0;

}

}

else

{

clearviewport();

setbkcolor(9);

setcolor(10);

settextstyle(4,0,7);

setusercharsize(120,50,120,40);

outtextxy(getmaxx()/2-220,getmaxy()/2-180,"GAME OVER");

settextstyle(8,0,1);

setusercharsize(50,60,40,50);

if(count\_arrow<=0)

outtextxy(getmaxx()/2-100,getmaxy()/2-70,"NO MORE ARROWS");

if(count\_balloon<=0)

outtextxy(getmaxx()/2-120,getmaxy()/2-70,"NO MORE BALLOONS");

outtextxy(getmaxx()/2-120,getmaxy()/2-20,"YOUR SCORE IS : ");

itoa(score,score1,10);

outtextxy(getmaxx()/2+150,getmaxy()/2-20,score1);

if(level==1)

outtextxy(getmaxx()/2-220,getmaxy()/2+20,"YOU REQUIRE TO PRACTICE MORE");

if(level==2)

outtextxy(getmaxx()/2-70,getmaxy()/2+20,"WELL PLAYED");

if(level==3)

outtextxy(getmaxx()/2-220,getmaxy()/2+20,"YOU ARE AN EFFICIENT SHOOTER");

outtextxy(getmaxx()/2-30,getmaxy()/2+50,"(Q)UIT");

settextstyle(1,0,1);

setusercharsize(30,65,30,60);

while( getch() != 'q');

finish=1;

break;

}

}

free(bow);

free(arrow);

free(balloon);

closegraph();

}

void draw\_balloon(int x,int y)

{

setcolor(RED);

setfillstyle(1,RED);

fillellipse(x,y,3\*BALLOON\_SIZE,4\*BALLOON\_SIZE);

line(x,y+4\*BALLOON\_SIZE,x,y+6\*BALLOON\_SIZE);

}

void draw\_burst ( int x, int y )

{

setlinestyle(0,0,1);

line ( x - 16, y - 12, x - 10, y - 2 ) ;

line ( x - 10, y - 2, x - 16, y ) ;

line ( x - 16, y, x - 10, y + 2 ) ;

line ( x - 10, y + 2, x - 16, y + 12 ) ;

line ( x - 16, y + 12, x - 6, y + 2 ) ;

line ( x - 6, y + 2, x, y + 12 ) ;

line ( x, y + 12, x + 6, y + 2 ) ;

line ( x + 6, y + 2, x + 16, y + 12 ) ;

line ( x - 16, y - 12, x - 6, y - 2 ) ;

line ( x - 6, y - 2, x, y - 12 ) ;

line ( x, y - 12, x + 6, y - 2 ) ;

line ( x + 6, y - 2, x + 16, y - 12 ) ;

line ( x + 16, y - 12, x + 10, y - 2 ) ;

line ( x + 10, y - 2, x + 16, y ) ;

line ( x + 16, y, x + 10, y + 2 ) ;

line ( x + 10, y + 2, x + 16, y + 12 ) ;

}

void draw\_bow(int x,int y)

{

setcolor(RED);

setlinestyle(0,0,1);

line(x+32,y-49,x+32,y+49);

setlinestyle(0,0,3);

arc(x,y,300,60,60);

arc(x+34,y-56,100,220,6);

arc(x+34,y+56,140,260,6);

}

void shoot(int \*x, int \*y)

{

char cnt\_arrow[5];

putimage(\*x, \*y-ARROW\_SIZE, arrow, COPY\_PUT);

delay(3);

putimage(\*x, \*y-ARROW\_SIZE, arrow, XOR\_PUT);

\*x+=ARROW\_SIZE;

if (\*x>590)

{

\*x=155;

flag\_arrow=0;

count\_arrow--;

settextstyle(10,0,1);

setusercharsize(30,70,20,70);

outtextxy(20,400,"ARROWS LEFT :");

setfillstyle(0,WHITE);

bar(200,400,220,425);

itoa(count\_arrow,cnt\_arrow,10);

outtextxy(200,400,cnt\_arrow);

}

}

void draw\_arrow(int x, int y)

{

setlinestyle(0,0,2);

moveto(x, y);

linerel(6\*ARROW\_SIZE, 0);

linerel(-2\*ARROW\_SIZE, -1\*ARROW\_SIZE+1);

linerel(0, 2\*ARROW\_SIZE-1);

linerel(2\*ARROW\_SIZE, -1\*ARROW\_SIZE);

}

int testkeys()

{

union REGS ii, oo ;

ii.h.ah = 0 ;

int86 ( 22, &ii, &oo ) ;

/\* if ascii code is not 0 \*/

if ( oo.h.al )

return ( oo.h.al ) ; /\* return ascii code \*/

else

return ( oo.h.ah ) ; /\* return scan code \*/

}

void fly(int \*x, int \*y)

{

int x1;

char cnt\_ball[5];

putimage(\*x-4\*BALLOON\_SIZE, \*y-5\*BALLOON\_SIZE, balloon, COPY\_PUT);

delay(20);

putimage(\*x-4\*BALLOON\_SIZE, \*y-5\*BALLOON\_SIZE, balloon, XOR\_PUT);

\*y-=BALLOON\_SIZE;

if(\*y<= 20)

{

\*y=400;

x1=450+rand()%150;

\*x=x1;

count\_balloon--;

settextstyle(10,0,1);

setusercharsize(30,70,20,70);

outtextxy(20,380,"BALLOONS LEFT:");

setfillstyle(0,0);

bar(200,370,230,400);

itoa(count\_balloon,cnt\_ball,10);

outtextxy(200,380,cnt\_ball);

}

}

void start()

{ float octave[7] = { 130.81, 146.83, 164.81, 174.61, 196, 220, 246.94};

setbkcolor(3);

settextstyle(7,0,0);

outtextxy(10,400," PRESS ANY KEY TO CONTINUE....");

settextstyle(1,0,0);

setcolor(4);

setusercharsize(25,15,20,4);

outtextxy(85,120,"BALLOON SHOOTING");

while( !kbhit() )

{

sound( octave[ random(7) ]\*4 );

delay(300);

nosound();

getch();

clearviewport();

rectangle(1,1,638,478);

settextstyle(3,0,1);

setusercharsize(50,30,50,30);

outtextxy(150,10,"INSTRUCTIONS");

setbkcolor(10);

settextstyle(1,0,1);

setusercharsize(40,70,20,20);

outtextxy(10,70,"1. You can play three levels.");

outtextxy(10,110,"2. You can move the bow UP and DOWN with the help of arrow keys.");

outtextxy(10,150,"3. Press right arrow key to shoot the arrow.");

outtextxy(10,190,"4. You score 20 points every time you shoot the balloon.");

outtextxy(10,230,"5. First level has 6 arrows and 10 balloons.");

outtextxy(10,270,"6. You require to score 100 points to enter the second level.");

outtextxy(10,310,"7. Second level has 6 arrows and 8 balloons.");

outtextxy(10,350,"8. You require to score 200 points to enter the third level.");

outtextxy(10,390,"9. Third level has 6 arrows and 6 balloons.");

settextstyle(7,0,1);

outtextxy(150,430,"PRESS ANY KEY TO CONITINUE");

}

getch();

setusercharsize(1,1,1,1);

settextstyle(0,0,0);

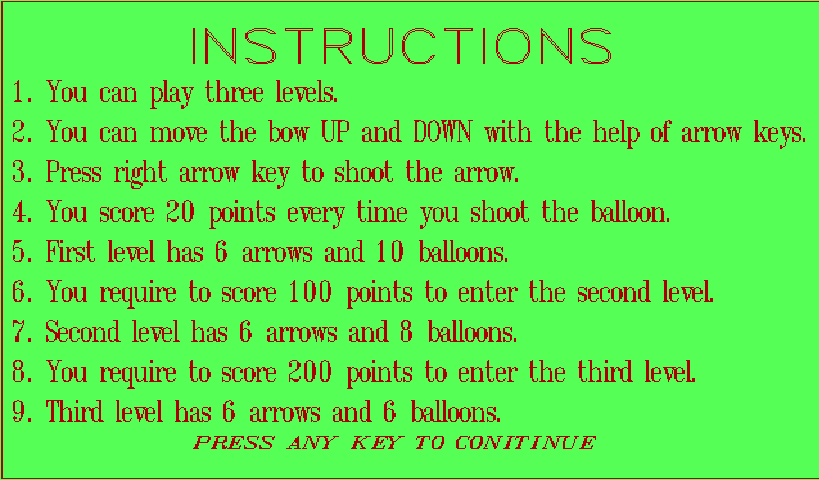
setbkcolor(10);

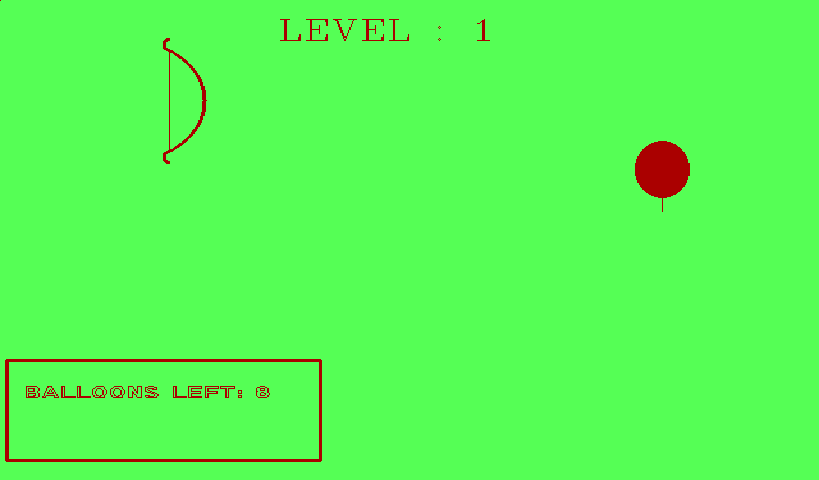
clearviewport();

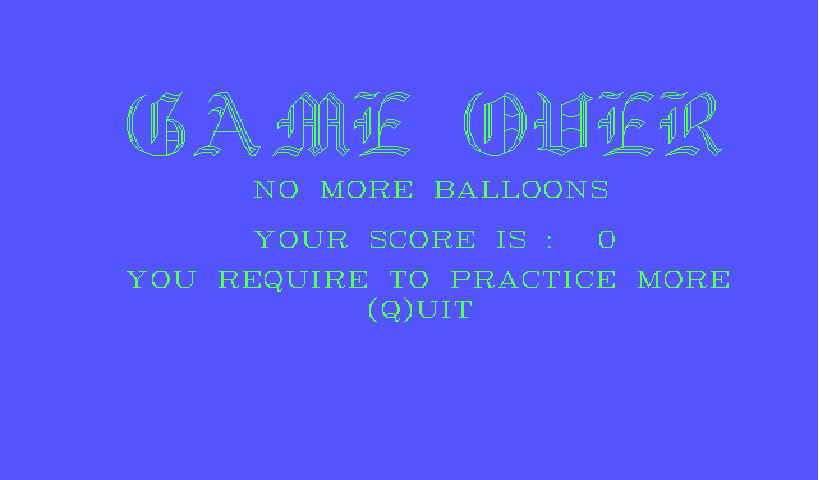
}

**5.SCRREN SHOTS**





****

****

**6.conclusion**

This project has enabled us to create a balloon shooting game which involves creating a graphics game using C language consisting of different levels.

This program simulates normal concepts in C language which makes ourselves strong with fundamentals and other important concepts.

This project is fortified with efficient graphic tools that enable the balloon shooting game to manage easily and effectively.

**7.REFERENCES**

WEB REFERENCES:

<http://www.sourcecodesworld.com/source/show.asp?ScriptID=785>

BOOK REFERENCES:

* C language by E.BALA GURUSWAMY