TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
Pulchowk Campus

Department of Electronics and Computer Engineering

Mini Project Report

On

POP-LOON (A C-Programming Project)

By

Name – Pratik Luitel	Roll No. 426
Name – Pujan Budhathoki	Roll No.428
Name – Sajil Awale	Roll No.436
Name – Shrey Niraula	Roll No.443

Kathmandu, Nepal

2073

Acknowledgement

We would like to take the time to thank the Department of Electronics and Computer Engineering, IOE Pulchowk Campus for providing us this platform to broaden the horizon of our understanding of the subject.

Suffice to say, we would also like to thank, from the bottom of our hearts our teachers for all the valuable co-operation which was vital for this idea to be conceived. They are the ones who guided and encouraged us every step of the way, and we hope that this co-operation will continue throughout our academic journey.

Also, we express our sincerest gratitude to everyone who had the slightest psychological impact upon us, including our friends for encouraging us and bringing out the best of us thanks to the healthy competition that was built on the foundation of friendship.

Abstract

The C- language is a perfect entry point on the journey to becoming a programmer of high calibre. As first year students, we can really launch our engineering journey forward by studying and applying the concepts of C Programming whenever appropriate. This project aims to do just that, by applying all the major topics that have been covered in the C programming course of the first semester in a small game titled Pop-Loons in which the user has the task of popping the bubbles that appear on the screen with their mouse button once they enter inside a rectangular target area displayed on the screen. The project also has a mini game that follows the same concept; but aims to find out the reaction time of the user/player on the basis of how fast the button is clicked when a bubble appears on the screen. All the topics of the course have been covered in the project, ranging from functions, pointers to file handling. The output of the project has also been largely according to our desire and design, as it has succeeded in fulfilling all the criteria that we put forward before typing out the code for it – including the popping of the bubbles, the score system (including high score storing) and also the reaction time mini game. Hence, it can be concluded that this project has largely been a success, and that it has fulfilled its purpose of imparting the basic knowledge of programming, and also building a sense of teamwork in us.

Table of Contents

1. Introduction

1.1. Introduction to the Project

The project is a simple game which was designed using our knowledge of C programming. The game is named "**Pop-Loons**" and serves as an entertaining means to broaden our understanding of the subject.

The objective of the game is to "pop" the balloons moving from left to right on the screen by clicking on them when they enter the rectangular target box; and pop them as close to the centre as possible in order to achieve the highest number of points. The moving balloons are of basically two sizes – big and small, and the player needs to click either the left mouse button or right mouse button depending on the size of the balloon to be popped. In the current version, the left mouse button pops the smaller balloon and right mouse button pops the larger balloon. The inability of the player to click on the balloon or wrong clicking of the mouse button will result in the ending of the game. The score is based on how close the player clicks the balloon to its center. Maximum points are awarded to clicking on the exact center.

Furthermore, to increase the difficulty level of the game, we have developed the system of increasing the movement speed of the balloon gradually as the game progresses. The target box also keeps on getting smaller, serving as an entertaining challenge to the player.

1.2. Team Members, Course of Action

The team members who contributed to the project are as follows:

i. SajilAwale (073BEX436)

ii. Shrey Niraula (073BEX443)

iii. PujanBudhathoki (073BEX428)

iv. Pratik Luitel (073BEX426)

The course of action for this project was mind bogglingly simple: we took out a few hours every week to discuss the project and make additions to the ideas and the code among ourselves. In times when we weren't discussing the project among ourselves, we conversed through social media and e-mail, furthering our understanding of the concepts along the way and implementing them when we had those weekly meetings. We started the project back in January, and it was a slow and steady process from there on in, taking us about two months to complete.

1.3. Objectives of the Project

The major objectives of the project are as follows:

i. To impart in us the practical knowledge of C programming language, its concept and execution.

ii. To test our skill of implementation of theoretical knowledge and give us a taste of carrying out a full-fledged project.

iii. To instill in us the experiences of group work and develop in us a sense of team-spirit.

2. Theory on topics used on project:

As our project comprises of various elements that are related to different topics on c, a brief description on these topics is a must.
Some of these topics are given below:

2.1. Array:

Arrays are aspect of C programing that often shows up when it would be convenient to have one name for a group of variables of the same type. The member of the array can easily be accessed by the numerical index under looping. Arrays are essentially a way to store many values under the same name. We can make an array out of any data-type from integer, float type, string to even pointer and structures.

General Syntax:
TypearrayName[arraySize];
This is called a single-dimensional array

Array can be of many dimensions as far as possible according to the necessity. To increase the dimension of the array, we can simply add another square bracket at the end and mention its size

i.e type array Name[array size][array size]…;
Similar to the fundamental data type, we can also initialize the array as;
intArray[]={24,455,-22,9};

2.2. Structures:

Unlike arraythatallows us to hold several data items of the same kind, structureenables us to hold data items of different kinds, making it combined data type.

Structures are generally used to represent a record. Let's suppose we want to keep track of books in a library, then we can have several related items of different data types such as:
Title (string)
Author (string)
Book ID (integer)
Price (floating value)

Under such condition, use of structure enables us to access these members easily. To declare the structure templates, general syntax is:
struct [structure tag]
{
member definition;
member definition;
….

} [one or more structure variables];

For example:
Struct book
{ char Title[100];
char Author[100];
intBook_ID;
float Price;
}book1;

To access the member of the structure, we use dot operator (.) or arraowpointer (->)
i.eprintf("%d", book1.Book_ID);
or
printf("%d", book1->Book_ID);

2.3. Pointers:

Pointers are derived type of data type which is used to hold the address of the variable which may be integer, float type, string, even the array and structure i.e basically everything. It means that there can be pointer to integer, pointer to float, pointer to the array and even pointer to pointer, i.e.they "point" to locations in memory.

Syntax
<variable_type> *<name>;
For example, we can declare a pointer that stores the address of an integer with the following syntax:
int *points_to_integer;

In above syntax, * is called the dereference operator, which is used to point to the value that pointer points to.
Similarly, there is reference operator (&) or also called as address of operator, which is used to pass the address of the particular variable.

```
i.eint a, *p;
    *ptr=&a;
     printf("%d",  ptr)
```

output
653002

Here, at first, integer variable is declared and pointer to integer is also declared then in next line, the memory address of the variable 'a' is passed to the pointer variable. When printing the value of the ptr, memory location of the 'a' is printed on the screen as shown above.

Effective use of the pointer can optimize a program, make it to run faster or use less memory that it would otherwise.By passing the address of the variable,

we can actually modify the actual arguments from some other function. This saves the compiler to copy the value each time from called function to the calling function resulting to faster processing with less memory consumption.

2.4. Functions:

A function is a self-contained block of statements that perform a coherent task of some kind.Every C program can be thought of asa collection of these functions.
Function can be of two major types; Library function and User defined Function.

i. Library Functions: They are inbuilt functions in C whose parameter and name cannot be altered by the programmer. They are called in function by including the appropriate header file.
   Examples are: printf, scanfunderstdio.h
   pow, sqrt under math.hetc.
ii. User Defined Function: They are programmer made function to perform certain task of the user. This function can be included in header file and can be called by incluing the header file.

For example:

```
#include <stdlib.h>   /* Include rand () */
int a = rand(); /* rand is a standard function that all compilers have */
```

This is an example of predefined function.
Now for user-defined functions, let's look at the program below

```
#include <stdio.h>
intmult ( int x, int y );
void main()
{
int x;
int y;
printf( "Please input two numbers to be multiplied: " );
scanf( "%d", &x );
scanf( "%d", &y );
printf( "The product of your two numbers is %d\n", mult( x, y ) );
getchar();
}
intmult (int x, int y)
{
return x * y;
}
```

This program contains a user defined function called mult() that takes values of two integers as arguments and returns their product.

2.5. File Handling:

Through file handling, one can perform operations like create, modify, delete etc on system files. File represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a readymade structure.

In C language, we use a structure pointer of file type to declare a file.
FILE *fp;
Where, FILE is built in structure under the header file stdio.h

C provides a number of functions that helps to perform basic file operations. Following are some of the functions,

Function	**description**
fopen()	Create a new file or open an existing file
fclose()	Closes a file
putc()	Writes a character to a file
fscanf()	Reads a set of data from a file
fprintf()	Writes a set of data to a file
getw()	Reads a integer from a file
putw()	Writes a integer to a file
fseek() and so on.	Set the position of file pointer to desire point

General Syntax:
*fp = FILE *fopen(const char *filename, const char *mode);

Here filename is the name of the file to be opened and mode specifies the purpose of opening the file. Some of the modes are:

Mode	**Description**
r	Opens a text file in reading mode
w	Opens or create a text file in writing mode.
a	Append mode to append the character from the end of file
r+	Reading mode followed by writing
w+	Writing mode followed by reading

2.6. Malloc:

In C, malloc is a subroutine for performing dynamic memory allocation. malloc is part of the standard library and is declared in the stdlib.h header.
Many implementations of malloc are available, each of which performs differently depending on the computing hardware and how a program is written. Performance varies in both execution time and required memory. The pointer to memory allocated using malloc must eventually be passed to the free subroutine to deallocate the memory in order to avoid memory leaks.
Its function prototype is
void *malloc(size_t size);
here, it allocates size bytes of memory. If the allocation succeeds, a pointer to the block of memory is returned. This pointer is guaranteed to be suitably aligned to any type (including struct and such), otherwise a NULL pointer is returned. Memory allocated via malloc is persistent: it will continue to exist, even after the program leaves the scope which called the allocation, until the program terminates

or the memory is explicitly deallocated by the programmer. This is achieved by use of the free subroutine. Its prototype is
void free(void *pointer);
which releases the block of memory pointed to by pointer.

2.7. graphics.h

graphics.his a non-standard header initially provided by Borland compiler package as their add-on goodies.It'sheader file which contains graphical functions.C graphics using graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more.

2.8. rand()

The C library function intrand(void) returns a pseudo-random number in the range of 0 to *RAND_MAX*.RAND_MAX is a constant whose default value may vary between implementations but it is granted to be at least 32767.
Syntax:
int rand(void);

2.9. stdbool.h

The header stdbool.h in the C Standard Library contains four macros for a Boolean data type.Under it, there is bool that expands to _bool. The main use of it is to minimize the size by taking only 1 bit.

2.10. dos.h

dos.h in c: dos.h header file of c language contains functions for handling interrupts, producing sound, date and time functions etc. Some of the functions under dos.h are: delay, getdate,gettime, nosound and so on.

2.11. time.h

The time.h header defines four variable types, two macro and various functions for manipulating date and time. It contains some of the library variable, macros, and library functions such as: time_t(library variable), NULL(macro), clock_t clock(void)(library function)

2.12. setactivepage(int page) and setvisualpage(int page)

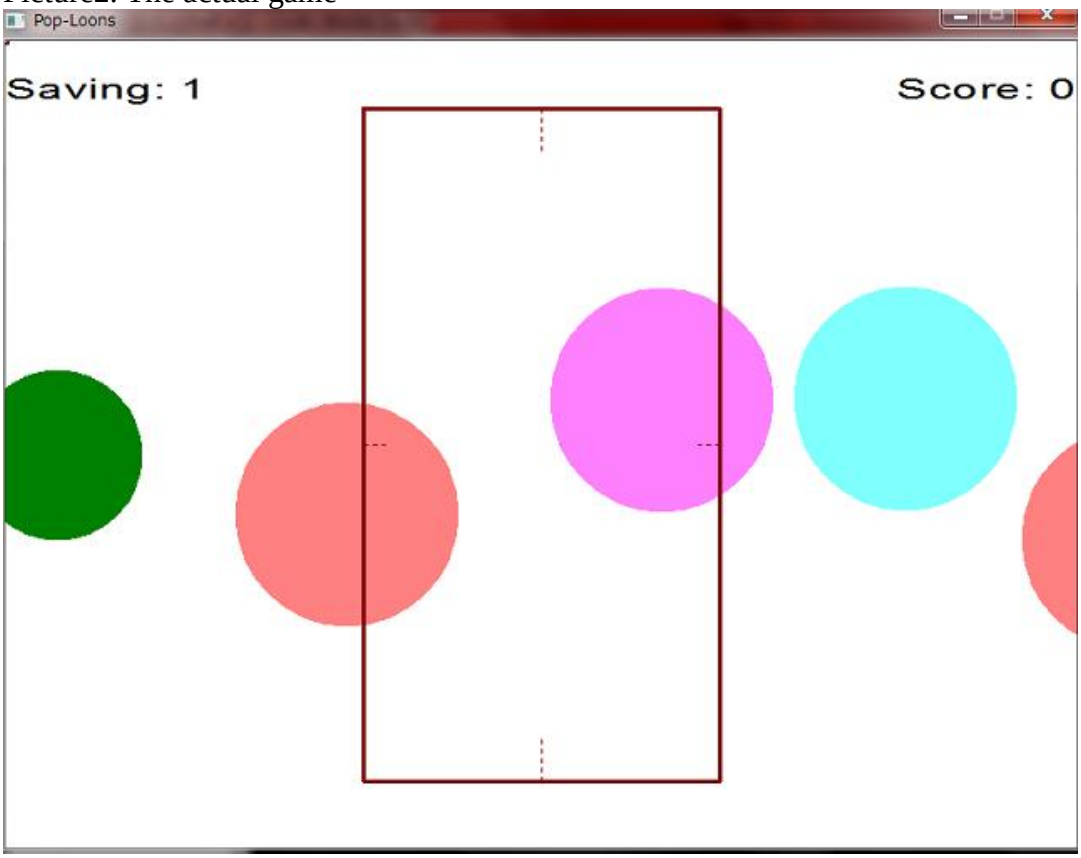
setactivepage makes page the active graphics page. All subsequent graphics output will be directed to that graphics page and setvisualpage makes page the visual graphics page. They are used to prevent the buffering or blinking of the graphics in graphics.h

3. Output

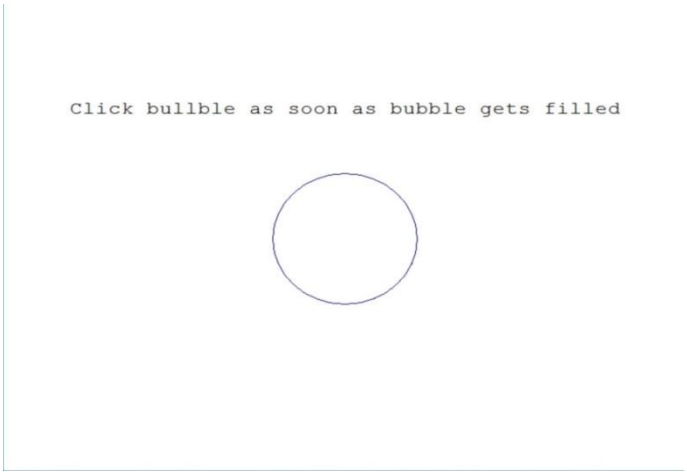
The output was largely to our desire, and the screenshots from the final output are attached below:

Picture1. Home screen (Menu)

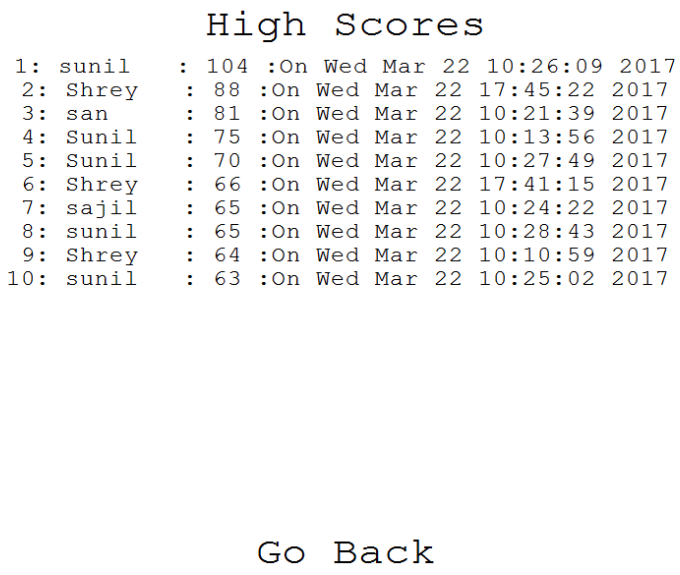
Picture2. The actual game



Picture3. The reaction time mini-game



Picture4:Highscore Screen



4. Working mechanics

In the project we have multiple functions with their own purpose. OS first calls the main function which initializes the graphics drivers and graphics mode.‘int check_h_m_h_s_i_f()’ function is called which returns the number of high score in the high score file and create the file incase the file has not been created. Then the main function enters infinite loop, where it first calls ‘int welcome()’

‘int welcome()’ prints out the dashboard of the game and returns specific integer depending on the click detected at the area. This function call more functions like ‘void b_but(int x,int y,char*)’ that takes x,y coordinates and string to be printed and outputs them on screen,’ bool button(int y, char *button_name)’ which takes only y coordinate of buttons placed on screen (it calculates x coordinates by itself) and name of button and returns a Boolean to signify if there is click. This function is inside a infinite loop with delay until any of the button is clicked. The y coordinate of the each button is calculated and returned by the ‘int ycalc(int nt,int n,int font=8,int txt_size=5)’,where nt and n are the total number of button to arrange and n is the specific number of button of whose y coordinates to be calculated.

Depending on the returned value of the ‘int welcome()’ , switch statement works and calls other function within the block. When 0 is returned which is for case label(value) of new game, ‘void new_game()’ is called. Inside it is declaration of an array named ‘sp’ i.e pointers that point to the typedef structure named ‘s’ containing all the necessary information about a single bubble. Also it declares another variable named ‘inf1’ of typedef structure named inf, i.e additional

information about the state of game. Then is initializes all the necessary variables controlling the state of the game, declares the variable i (int) and GAP where 'i' is unique identifier of each bubble and GAP is the distance between two centers of two consecutive bubble. The main loop, which we call 'i' loop starts then in which we malloc memory as required for only one s type structure store the returned address to sp[i], also we assign the radius, component of x coordinate, bubble counter, etc of the bubble. But note that there is an offset to some quantities in the assignment we do. This has to do with processing value of GAP, uncertainty of x coordinate, etc. This loop has a nested for loop which we call 'frame' loop,ie it generates a frame of the game.

In the 'frame' loop, we assign GAP according to the bubble size of two consecutive bubble. This value is randomized by calling int rnd(int ll, int hl) which returns integer in the range ll and hl.(lower limit and higher limit). The assignment happens only once for each bubble which then faces if statement that checks if it's time to create another new bubble. If it is not the time to create, then it continues animating the bubble. The only thing to do to create a new bubble is to increment the value of i/indi which we do by breaking from frame loop. After this is calling of 'bool bu_gen(s **sp,inf *inf1)', which generates the complete frame of bubbles in the game.

'bool bu_gen()' takes all the information about the bubbles and state of game as it's arguments. It takes major decision of the games like incrementing scores and gameover, decrementing savings etc. Thus it is the dominating function throughout the game. At the beginning it sets the double buffering. As the function is called for each frame/motion we can check different state of game and update it to variables too in the function. Inside this function is for loop that generates all the bubbles. Note that it does not generate all the bubbles ever created but the once siginificant to animate by using the state of the game variables, i.e saver1 and indi. 'saver1' is the no of bubbles out of screen and indi is the no of bubbles that has ever been created. More inside the for loop is the calculation of the x and y coordinates of the center of the bubble.'curr_x' and 'curr_y' are two variables in the s type structure that gives the component of the x coordinate of the centre and y centre respectively. The other two cy and cx are the uncertainties of the x and y coordinate of centre which has separate function for their calculations. These uncertainties are initlized to zero previously, hence these quantities are added to curr_x and curr_y before generating bubble. After the bubble has been generated cx and cy are calculated for the next frame. The basic of our jitter free motion is accomplished by the functions 'void y_uncer_val(s **sp,int i)' and 'void x_uncer_val(s **sp,int i)' which calculate cy and cx . y_uncer_val(), has assigns the random number within a range (-2*CY_RANGE,2*CY_RANGE), where CY_RANGE to variable dy. The upcoming if condition checks whether the bubble will lie within the target. If the bubble will lie on target box, range of random number is limited to such that it can't pass edge of the target box and the value of dy is then assigned to the cy. Similar to that but not symmetrical is x_uncer_val(), where natural flow of the bubble with 'c' can't be ignored but has to be included in the comparison. It checks if the dx which is initialized to random range (-2*CX_RANGE, 2*CX_RANGE) makes the bubble overlap with other bubble; i.e to its left or right.

The range if limited to only the side where there is no chance of overlap and finally dx is assignment to cx. Note that because this is a loop this value is only utilized in the next run of that particular bubble.

Game over is performed by the function void over() displaying game over is by calling a function display_over()function. It game overs if any criteria for game over is satisfied.

As per rule the mouse pointer must be within the target box. But for further checking condition, we will need to know exactly which bubble is on the mouse pointer, basically a function returning the i^{th} value of bubble that is being hovered is ideal, that is what 'int check_w_b(sp,inf1)'does. Inside this function is for loop from saver2 to indi where 'saver2' is the no of bubbles out of target box as it flows from left to right. This function measures the distance between each of the bubble center from mouse pointer, if the distance is less than the radius of that bubble, it means the bubble is on the mouse pointer so the function returns it's 'i' value.

Further this function also resets inf1.ssh, i.e stop scoring high. This is a Boolean like variable whose value at the beginning was initialized to 0 (i.e scoring up is possible). But on clicking on any bubble its value is incremented stopping the score to not ramp up when clicking continuously with no reason. This function has a static variable 'static int prev_i' which stores 'i' value just before it is returned and just before that it checks to see if the current 'i' value matches the previous 'i' value; if it does then ssh is not reset to 0 enabling the score to rise up depending on other condition. This function also returns -1 if no bubble lie on that region or white/invisible bubble lie there. Note that this function also check another state defining parameter; i.e 'inf1.prev_click'. This is a boolean that has its value 1 if mouse is continuously clicked and 0 if the continuous clicking has stopped.

Back to the 'bugen()' where it calls the 'int check_w_b()' and stores the returned value to t_bvariable whih stands for the_bubble. First we have if statement that tests if there is any click and the t_b is -1(No bubbles there) and no continous click then we get contidion of game over. But if there is click and ssh is equal to 0 (reset condition) then it further tests with nested if statement, if there is left click when the radius of the t_b th bubble is sr/small and fill is solid(i.e 1) or the bubble is big(br) and the fill is hatched and other secondary test are true then it 's time to disappear the bubble by colouring the bubble white and rewarding with score defined as per the function 'int UP_SCORE(s **sp,inf *inf1, int t_b)' which takes all the information of bubbles, state of game and t_b th bubble.

In the function, it first measures the distance between the mouse pointer and centre of that bubble using distance formula and this function returns the score. If the player click at the exact centre user will be awarded with the score equal to the radius of the bubble, otherwise score is divided by the distance from center and is provided to the user. At exact center, player is also awarded with increment of saving value. Here, 'saving' is number of bubbles that can be ignored or skipped which can be #defined. For this increment of saving we call 'int saving_keeper(int change,s **sp,inf *inf1)' which takes in the change it needs to make to the saving, here saving is a static int variable initialised to START_SAVING which is the #defined. In fact this function has a for loop of its own from 'saver2' to indi in which it checks to see if any new bubble has passed the target box; if so the saver2

is updated as it increments by 1. Inside as the nested if statement we check if the bubble is not invisible/white then the value of saving is to decrement by 1. This function then displays the saving value to graphics screen and returns to the calling function. After all that returning the increment score back to the bugen(), it has to call ‘int score_keeper(int change)’ which is similar to ‘saving_keeper’ but it does not have any for loops and is much simpler. The return value of UP_SCORE is fed to the score_keeper function and this function values the score by adding the change argument to the existing parameter static int sc_n that has been initialised to 0 and finally it displays the score to the screen.

Back to the bugen(), we have another similar opportunity to reward, with another else if statement that tests to see if there is right click when the radius of the t_b th bubble is br/big and fill is solid(i.e 1) or the bubble is small(sr) and the fill is hatched and other secondary test are true then it ’s time to disappear the bubble by colouring the bubble white and rewarding with score much more similar to previous and finally prev_click is set to 1.

If none of the condition is true then the player must have wrong clicked the bubble so it’s game over. At the end of bu_gen() is swapping the page of double buffer followed by the delay which is #defined to FRAME_DEl. This function returns 0 if game was over during the frame and 2 if no game over has occurred back to new game() being tested. If the game is over few parameters have to be reset e.g. score, saving, width of target box,etc also it has to free the unfreed malloc space with ‘free_rem_sp(sp,inf1.saver1,i+1)’ that has not yet been freed by the primary freeing statement in coming lines, and finally return to calling function which is main, otherwise continue with the game.

Followed by all these, there is function ‘void target_box()’ which displays the target box on the screen, the only thing is that this function first calls score_keeper() function any it sets the width of the target box with a mathematical function such that the width is smaller as the score keeps growing. Next statement call scorekeeper with argument 0, i.e just for viewing score plus updating the score value on screen. Just below is a significant for loop which we call increment loop that increment all the parameters needed for the bubbles to be moving in its flow. But how much should it be incremented in that go. This step value of increment determines the speed of the flow so it has to be processed which is what ‘int step_process()’ returns. In this function is another mathematical function whose value is directly proportional to the score. At the ending of the frame loop is a if statement that tests if the bubble with i value equal to saver1 has passed the screen, if so free up the memory allocated for that bubble and increment the value of saver1. Finally this end the new_game function and back to main function where the switch is ‘break’ causing it to loop to the beginning of the loop, i.e calling of welcom(), where it waits for a click.

Let’s consider this time the user clicks on Reflex Game causing welcom() to return 2 calling rxn_game()

Rxn_game() is a simple function, where it first outputs quick instruction above. This game has two loops where one of them is nested. The outer j loop is for numbers of bubble/fills and inner loop with delay 1 millisecond is to measure the reaction time of each clicks. As it enter j loop it outputs a circle with just the outline. Then it checks if there is any click if there is then freeze the program at

that point. This action is important because of the reason that we are humans and we click for some more than instant. The placing of this freezing statement might seem suspicious in the first run but the note that this is a loop and program will execute this over again in which this freezing is extremely significant. Then there is a delay that has random argument passed to it such that user can't trace pattern of timing of arrival time the of filled circle. Good or bad but the library function used to detect the click does work independent of the graphics window but it's value freezes during the delay. Think about it as Bitwise OR, if you click at the delay interval the value achieved after that is true no matter what. This makes too easy for our purpose, we want the player to disqualify if he clicks vary randomly to get lower reaction time. Hence we display "OUT OUT !!!" if that is what we detect any then return to main function which redirects to welcom() screen. Else we enter the i loop where we first display a filled circle, just below we check if we detect any click any if it is with in the circle if it is true we break the i loop. But most probably it does not break in the first run else the run-time is 0 the is not possible. At the ending is the delay of 1 millisecond as said before. the i value gives the total time in millisecond which after the loop ends sums ups to variable called sum previously initialised to 0. With the help we repeat this process for RXN_BUBB times which is #defined for longer lasting game and accurate measurement. Finally we display the reaction time which is achieved by dividing the sum by RXN_BUBB. Finally we return to main function —> welcom().

Concider this time user clicks on the "Instructions", welcom() returns 3 which redirects to 'bool instruction()'. Inside this function has an array of strings of instruction to output.In a for loop it calls b_but() to output all of then given the y coordinate which is achieved by adding text height of each with few additional numbers as it initialises. However x coordinate is easy to achieve which is to align the text to centre of screen. It also has a button placed at the bottom called "Go back" It is placed and operated completely by button() function which waits and return 1 if click is detected on it. If the button is clicked then this function returns true/1 to main function. This return value is check in the main function, if so the continue statement is executed that makes flow to the beginning of loop and calls welcom().

One of the thing not mentioned as the program flows was the high score algorithm. Back track the program flow when the condition of game over met. All the alternative of game over called 'void over(int score,char reason[])'. This function has a variable called bkcolor(int) which is initialised to 0 that is black colour in terms of colour. Then calls 'bool is_it_h_s(int score)' which returns 1 if somehow the current score achieved is one of the high score. If the return is 1 the 'bkcolor' is set to 5 that is red colour. There is two way the score achieved is high score, either the no of score saved in the file is less than TOP_SC which is #defined or if the score is larger than the smallest high score. Smallest high score is the last high score after it is sorted in descending order. A function 'bool D_h_s_sort()' arranges the high score in descending order as it overwrites with rb+ mode.The descending order is done with the bubble sort algorithm. It works similar to sorting array, the only difference is that you will need care about the file

pointer any where it points. The number of high score in the file is counted and returned by ‘int check_h_m_h_s_i_f()’ as said in the beginning lines. Then the Game Over is displayed, just after that is a function named ‘bool save_h_s_iff(int score)’ which saves the high score in two ways is called.

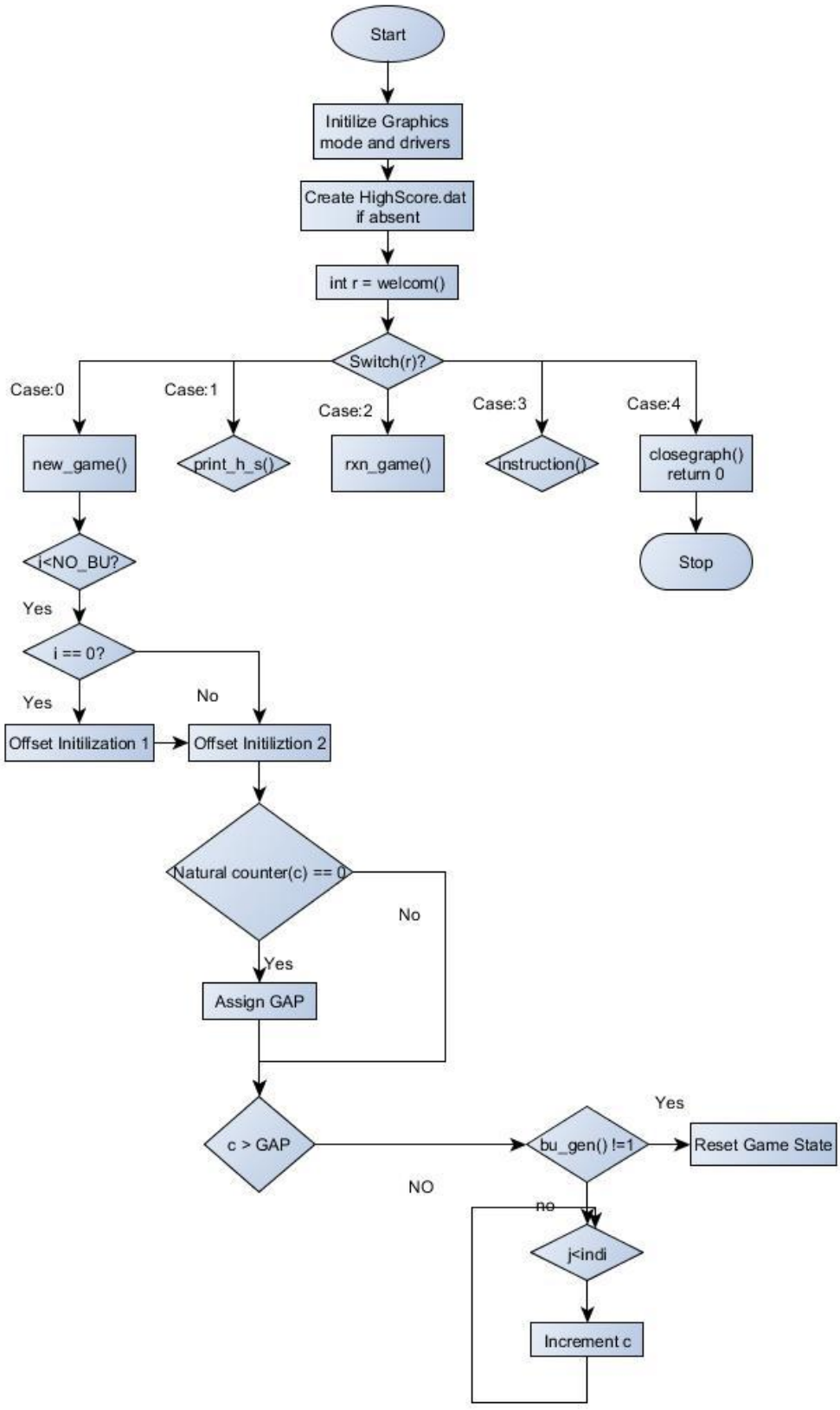
In ‘save_h_s_iff(int score)’ it declares a h_s_d typedef structure named h that has name, score and date_t as members. We open the high score data file in rb+ mode. Note the complication of this mode in problem encountered. But the main objective to open the file in this mode is to overwrite the high score file if we had to. Then we assign score and date_t to its members in advance. For the first method we check if number of high score is less than TOP_SC, if so then inside the block we call ‘void want_ur_name(h_s_d *h) ’ inside of which we close the graphics window and turn to text mode where we input the name of the player and store down to the name member and finally bring back the graphics mode.Back to the save_h_s_iff(), we ‘fseek’ the file to the end to continue writing. At this pointer we write the h to file. The other way to write is in the else block where we ‘fseek’ the file to -sizeof(h_s_d) from the end. That is we put the pointer just before the last high score which is the lowest after it is sorted and the h is overwritten to it.

Now, let’s concider that user clicks on high score button then ‘bool print_h_s()’ is called. In this function we open the file in rb mode.Along it we check if there is 0 high score data, if so we display that it looks as if it’s your first time. Then we sort the high score in descending order by calling D_h_s_sort(). In a for loop we take in data of each high score data store it to buffer then display it in formatted text using b_but again. Also it has a go back button so button() is called for the operation which waits till it is clicked which if detected print_h_s() returns 1 to main where it check for that return and continue the loop.

The last button on welcome screen is Exit. If this button is found clicked the welcom() returns -1, which when detected by the switch closes graphics screen and return 0 from the main that is the game exits out.

Last but most is the random number generator. The heart of this game lies there as it is the most frequently called function. So stronger the random number generator stronger the game and weaker it is so the game. Calling rand() in c does not generate a lots of random number in short time and one execution. So we seed the rand() by srand(time(NULL)) but return of time(NULL) does not change within a second and new random number can’t be generated within a second. So we have ‘ud_rnd()’ function which is xor shift random number generator. Random number generated by this function is number of call dependent so ud_rnd() can’t be used as random number generator nor rand() can be used as it is time dependent that does not change within a second. So what we did is that we combined time(NULL) and us_rand() to make seed value which is changing regardless of the time nor call but it make one hard core random number generator.

5. Block Diagram



6. Problems Encountered and possible enhancements

While developing the project, we came across the many problems such as memory management, double buffering, randomization, and code for mouse click. Although some of these problems were resolved, there still persists some of the problem such as continuousmemory increment and vast CPU usage which eventually freezes the game.
This problem has been tried to minimize lot by us by various ways, but we are still unable to find the core root of the problem.

In course of increasing the speed the of the balloon, we first applied the idea of reducing the delay time. But reducing thedelay time caused the system to pause for that certain moment due to which the unnecessary frame has been created and unnecessary memory usage has been increased and thus our game froze after 20-25 balloons.

To solve this problem, we changed the idea of the speeding the balloon which is by incrementing the value of the counter as per the score obtained. Because of this idea, we could prevent the unnecessary memory use. The number of the balloons could be increased to around50; however, there still exists the freezing problem, which we are still trying to solve on.

Dynamic memory allocation by the use malloc() function simplified our program increasing the program efficiency by freeing the memory of out-screen balloons.

Although the game was largely according to our desire, various other ideas were also encountered which sadly could not be implemented in the specified time, but are duly noted as follows:

i. Solving the freezing problems

This problem had to do with leaking graphics memory that froze the graphics window. But we are afraid that this might not only be the problem as we experiment that leaking graphics program that doesn't freeze whatsoever. In graphics.h closegraph() should have deallocated the memory but it didnot. There seems to be functions for allocating and deallocating graphics memory in graphics.h but sadly it wasnot implemented in windows version.

7. Conclusion

Hence, the requirements of the project were met, and we were able to create a fully functioning game that met all the objectives that were put forward and most of all, we were able to develop a team spirit and were able to implement the concepts that we learnt in class to our practical lives.

5. References/Bibliography

1. https://stackoverflow.com
2. http://www.cprogramming.com
3. http://tutorialspots.com

1. APPENDIX

1.1 SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <math.h>
#include <stdbool.h>
//for using bool datatype
#define TITLE "Pop-Loons"
#define SCREEN_WIDTH 800
//1360
#define SCREEN_HEIGHT 600
//700
#define TARGET_HEIGHT 500
#define xcen -79                    // x
cordinate center of ellipse at initial
point
#define sr 65                       //
radius of smaller balloon
#define br 85                       //
radius of bigger radius
#define NO_BU 1000
//currently this is the variable that
controls how many bubble will be
produced
#define H_S_LOCATION
"HighScore.dat"    //high score file
data
#define B_DEL 400
//button delay
#define INF_DEL 300
//delay for infinte loop i.e. provide
resting time for CPU
#define TOP_SC 10
//Number of high scores to be saved
#define FRAME_DEL 50
//i.e 20 fps
#define CY_RANGE 5
//is no of y or y/2 pixel uncertain
between any two frame of the game
#define CX_RANGE 2
//is no of x or x/2 pixel uncertain
between any two frame of the game
#define RNX_BUBB 7
//how many no of Rxn time
expirement to be performed
#define START_SAVING 5
#define GRAPH_INIT
initwindow(SCREEN_WIDTH,SCR
EEN_HEIGHT,TITLE)
typedef struct screen               //a
structure forstoring information of a
bubble
{
   int c;                           // c for
counters i.e i,j,k,l,m
   int cx;                          //for
uncertinity of x
   int cy;                          // for
uncertinity of y
   int curr_y;                      //current
y position of center of bubble
   int curr_x;                      //current
component of x position of center of
bubble
   int asize;                       //for
storing the size (radius)of bubble in
the screen
   unsigned int color:4;            // 0
to 2^4-1  = 15
   unsigned int fill:4;
} s;
typedef struct score_data
//structure for high score dat file
{
   char name[51];
//name of high score doer
   unsigned int score;
   char date_t[31];                 // date
and time of highscore created
} h_s_d;
typedef struct addtional_info
//structure to store the additional
information about state of game
{
```

```
    int indi;                       //
indicates how many bullunes are on
memory/have been created
    int saver1;                     // for
CPU efficency + (how many
balloons have gone out-1)
    int saver2;
    int ssh;                        // ssh -->
stop scoring high
    bool prev_click;                //for
score pani na badhaune out ni
nagaraune

} inf;

//essential prototype area
void over(int score,char *reason);
int score_keeper(int change);
int check_w_b(s **sp, inf *inf1);
int rnd(int ll,int hl);
int rd();
int step_process();
bool bu_gen(s **sp,inf *inf1);
void target_box();
s *allocate_the_memory();
int welcom();
bool c_fptr(FILE *fptr);
int check_h_m_h_s_i_f();

//essential globalvariables
int ycen = SCREEN_HEIGHT/2;
int oTARGET_WIDTH  =
SCREEN_WIDTH/3;

void want_ur_name(h_s_d *h)
//this function takes in the h_s_d
address + closes graphical window
and inputs name
{
    closegraph();
    puts("Enter your name for High
Score\n");
    fflush(stdin);
    scanf("%s",(h->name));
    puts("\nThanks, You can press
Enter to return to Gaming");
    fflush(stdin);
    getchar();
    GRAPH_INIT;
}
bool D_h_s_sort()
//returns 0 if sucessfull
{
    int struct_size, i, j;
    h_s_d h1, h2;
    struct_size = sizeof(h1);
    FILE * fptr =
fopen(H_S_LOCATION,"rb+");
    if(c_fptr(fptr))   return 1;
    fseek(fptr, 0, SEEK_END);
//go to end of file
    int file_size = ftell(fptr);
    rewind(fptr);                   //go to
begining of file
    for (i = 0; i < file_size; i +=
struct_size)
    {
        for (j = 0; j < file_size -
struct_size; j += struct_size)
        {
            fread(&h1, struct_size, 1,
fptr);          //rem that poniter itself
moved by 1 high score data so no
need to move it manually
            fread(&h2, struct_size, 1,
fptr);
            if (h1.score < h2.score)
            {
                fseek(fptr, -(struct_size *
2), SEEK_CUR);
                fwrite(&h2, struct_size, 1,
fptr);
                fwrite(&h1, struct_size, 1,
fptr);
                fseek(fptr, -struct_size,
SEEK_CUR);
            }
            else
            {
                fseek(fptr, -struct_size,
SEEK_CUR);
            }
        }
        rewind(fptr);
```

```
    }
    fclose(fptr);
    return 0;
}

bool save_h_s_iff(int score)
//save high score if and only if
{
    h_s_d h;
    time_t the_time = time(NULL);
    FILE *fptr =
fopen(H_S_LOCATION,"rb+");
    if(c_fptr(fptr))   return 0;
    h.score = score;
    strcpy(h.date_t,ctime(&the_time));

    if(check_h_m_h_s_i_f() <
TOP_SC)
    {
        want_ur_name(&h);
        fseek(fptr,0,SEEK_END);
//to continue writing
        if(fwrite(&h,sizeof(h),1,fptr) <
1)
        {
            perror("Could not save high
score " H_S_LOCATION);
            fclose(fptr);
            return 0;
        }
        fclose(fptr);
        return 1;              //this function
returns 1 if successfull
    }
    else
    {
        want_ur_name(&h);
        fseek(fptr,-
sizeof(h),SEEK_END);
        fwrite(&h, sizeof(h), 1, fptr);
        fclose(fptr);
        return 1;
    }
    fclose(fptr);
    return 0;
}

bool is_it_h_s(int score)
{
    FILE *fptr =
fopen(H_S_LOCATION,"rb");
    h_s_d h1;
    if(c_fptr(fptr))   return 0;
    if(check_h_m_h_s_i_f() <
TOP_SC)
    {
        fclose(fptr);
        return 1;
    }
    D_h_s_sort();
    fseek(fptr,-
sizeof(h1),SEEK_END);
    fread(&h1,sizeof(h1),1,fptr);
    if(h1.score < score)
    {
        fclose(fptr);
        return 1;
    }
    fclose(fptr);
    return 0;
}
void free_rem_sp(s **sp, int
saver1,int iPlus1)
{
    int i;
    for(i=saver1; i<=iPlus1; i++)
    {
        free(sp[i]);
    }
    free(sp);
}
bool c_fptr(FILE *fptr)
{
    if(fptr == NULL)
    {
        perror("Coulnot Open
"H_S_LOCATION);
        return 1;
    }
    return 0;
}
void b_but(int x,int y, char str[],int
color)
{
```

```
   setcolor(color);
   outtextxy(x,y,str);
}
int check_h_m_h_s_i_f()
//check how many high score data is
in the file + it creates the file
H_S_LOCATION.dat if absent
{
   int i;
   h_s_d temp;
   FILE *fptr =
fopen(H_S_LOCATION,"ab");      //
create the file H_S_Location.dat if
absent
   if(c_fptr(fptr))   exit(1);
//checks fptr
   fclose(fptr);                        // if
there file, append it then again open
it after closing
   fptr =
fopen(H_S_LOCATION,"rb");
   if(c_fptr(fptr))   exit(1);
   for(i=0;
fread(&temp,sizeof(temp),1,fptr) ==
1; i++);  // do nothing loop just
counts the score + fread return the
number of data red successfully +
second opton of this code is to use
fseek and ftell divid by sizeof temp
fclose(fptr);
   fclose(fptr);
   return i;                            // no
of high scores is returned
}
void over(int score,char reason[])
{
   char g_over[] = "GAME OVER
!!!";
   char sc[15];
   int bkcolor = 0; //black
   bool localbol = is_it_h_s(score);
   if(localbol)    bkcolor = 4; //red
   setbkcolor(bkcolor);
   settextstyle(3,0,8);
   outtextxy((getmaxx()/2)-
(textwidth(g_over)/2),(getmaxy()/2)-
(textheight(g_over)/2),g_over);
   sprintf(sc,"Score is %d",score);
   settextstyle(3,0,2);
   outtextxy((getmaxx()/2)-
(textwidth(reason)/2),(getmaxy()/2)-
3*(textheight(reason)),reason);
   settextstyle(3,0,5);
   outtextxy((getmaxx()/2)-
(textwidth(sc)/2),(getmaxy()/2)+(text
height(sc)/1.5),sc);
   delay(3000);
   if(localbol)
      save_h_s_iff(score);
   return;
}
int saving_keeper(int change,s
**sp,inf *inf1)
{
   static int saving =
START_SAVING;    //saving in
numbers
   int i;
   char savin[16];
   saving += change;
   for(i=inf1->saver2; i<inf1->indi;
i++)
   {
      if((((sp[i]->curr_x +sp[i]->c) -
(SCREEN_WIDTH+oTARGET_WI
DTH)/2)) > (sp[i]->asize))
      {
         (inf1->saver2)++;
         if((sp[i]->color)!= 15)
saving--;
      }
   }
   sprintf(savin,"Saving:
%d",saving);
   settextstyle(4,0,3);

b_but(0,textheight(savin),savin,BLA
CK);
   return saving;
}
int score_keeper(int change)
{
   static int sc_n=0;       //score in
numbers
```

```
   char score[16];
   sc_n += change;
   sprintf(score,"Score: %d",sc_n);
   settextstyle(4,0,3);
   b_but(getmaxx()-
textwidth(score),textheight(score),sc
ore,BLACK);
   setcolor(WHITE);
   return sc_n;
}
int UP_SCORE(s **sp,inf *inf1,int
t_b)       //this function return the
increment score accoring to distance
of mouse pointer from center
{
   int d;                        //distance
   int score;
   d = sqrt(pow((sp[t_b]->curr_x
+sp[t_b]->c) - (mousex()),2) +
pow((sp[t_b]->curr_y) -
(mousey()),2));
   if(d == 0)
   {
      saving_keeper(1,sp,inf1);
      return ((sp[t_b]->asize));
   }
   score = (sp[t_b]->asize)/(d);
   return score;
}
int check_w_b(s **sp, inf *inf1)
{
   int i;
   static int prev_i;
   for(i=(inf1->saver2); i<(inf1-
>indi); i++)
   {
      int t = sp[i]->asize;      //max
distance
      int d = sqrt(pow((sp[i]->curr_x
+sp[i]->c) - (mousex()),2) +
pow((sp[i]->curr_y) - (mousey()),2));
//distance between pointer and
bubble center
      if(d<=t)
      {
         if(prev_i != i)
            inf1->ssh = 0;      //reset the
stop score high(ssh) if new bullune is
on the target
         if(sp[i]->color != 15)
            return (prev_i=i);
         if(sp[i]->color == 15 &&
inf1->prev_click == 1)
            return (prev_i=i);
      }
   }

   return -1;         // i.e no bullunes
lie at that position
}
long ud_rnd()
{
   static long a=123456879,
b=356789879, c=578762099;
   long t;
   a ^= a << 13;
   a ^= a >> 17;
   a ^= a << 5;
   t = a;
   a = b;
   b = c;
   c = t ^ a ^ b;
   return c;
}
int rnd(int ll,int hl) // for generating
random numbers from lower limit ll,
to higher limit hl
{
   long raw_t = ((int)time(NULL) +
abs(ud_rnd()));
   srand(raw_t);
   return (rand()%(hl-ll+1) + ll);
}
int rd()
{
   if(rnd(0,1))    return br;
   else           return sr;
}
int step_process()
{
   int late, score = score_keeper(0);
   late = (12*score)/(sr) + 7;
```

```
   if(score >100 && score < 150)
late = rnd(25,35);
   if(score > 149)     late =
rnd(30,50);
   return late;
}
void y_uncer_val(s **sp,int i)
{
   int dy = rnd(-
2*CY_RANGE,2*CY_RANGE);
//randomize y cordinates;
   if((sp[i]->curr_y + dy) >
((SCREEN_HEIGHT+TARGET_HE
IGHT)/2 - sp[i]->asize))       // edi
taget box bhanda mathi gayo bhane
tellai count nagarna ko lagi tyo value
tala arkai assign  // over do cases
       dy = -rnd(0,CY_RANGE);
   if((sp[i]->curr_y + dy) <
((SCREEN_HEIGHT-
TARGET_HEIGHT)/2 + sp[i]-
>asize))        // overdo cases
       dy = rnd(0,CY_RANGE);
   sp[i]->cy = dy;
}
void x_uncer_val(s **sp,int i)
{
   int dx;
   dx = rnd(-
2*CX_RANGE,2*CX_RANGE);
   if(i==0)
   {
      sp[i]->cx =
rnd(0,2*CX_RANGE);
      return;
   }
   if(((sp[i-1]->curr_x+sp[i-1]->c)-
(sp[i]->curr_x +sp[i]->c+ dx))   <
(sp[i]->asize + sp[i-1]->asize))  dx =
-rnd(0,CX_RANGE);
   if(((sp[i]->curr_x +sp[i]->c+ dx)-
(sp[i+1]->curr_x+sp[i+1]->c))   <
(sp[i]->asize + sp[i+1]->asize))  dx =
rnd(0,CX_RANGE);   //over do cases
   sp[i]->cx = dx;
}
void display_over(char *reason,bool
page)
{
   setvisualpage(page);
   cleardevice();
   over(score_keeper(0),reason);
}
bool bu_gen(s **sp,inf *inf1)
{
   int i;
   bool b_t_r;     //value is 1 when
within target box   // is boolen
variable for knowing if the mouse is
witnin the target box
   static bool page=0;
   setactivepage(page);//drawing
page
   setvisualpage(1-page);  //display
page
   cleardevice();
   setlinestyle(0,0,2);

if(!(GetAsyncKeyState(VK_LBUTT
ON) ||
GetAsyncKeyState(VK_RBUTTON)
))
       inf1->prev_click = 0;
//reset prev_click if continousclick
has stoped
   for(i=(inf1->saver1); i<(inf1-
>indi); i++)
   {
       setcolor(WHITE);
       setfillstyle(sp[i]->fill,sp[i]-
>color);
       sp[i]->curr_y += (sp[i]->cy);
//set current y position
       sp[i]->curr_x += (sp[i]->cx);
//set current x position
       fillellipse(sp[i]->curr_x +sp[i]-
>c,sp[i]->curr_y,sp[i]->asize,sp[i]-
>asize);// bubbles
       y_uncer_val(sp,i); //randomize y
cordinates
       x_uncer_val(sp,i);    //randomize
x cordinates
   }
```

```
   setfillstyle(1,WHITE);
   setcolor(RED);
   fillellipse(mousex(),mousey(),3,3);
// mouse center pointer
   if(saving_keeper(0,sp,inf1) < 1)
   {
      display_over("Out of
Saving",page);
      return 0;
   }
   b_t_r =
((mousex())>=((SCREEN_WIDTH-
oTARGET_WIDTH)/2) &&
(mousex())<=((SCREEN_WIDTH+o
TARGET_WIDTH)/2) &&
(mousey())>=(ycen-
TARGET_HEIGHT/2) &&
(mousey())<=(ycen+TARGET_HEI
GHT/2));
   int t_b = check_w_b(sp,inf1);
//check_which_bubble lies in the
target area and store it's i
value(identifier) of bubble to t_b


if((GetAsyncKeyState(VK_LBUTT
ON) ||
GetAsyncKeyState(VK_RBUTTON)
) && t_b == -1 && inf1->prev_click
== 0)
   {
      display_over("No Balloon
there",page);
      return 0;
   }


if((GetAsyncKeyState(VK_LBUTT
ON) ||
GetAsyncKeyState(VK_RBUTTON)
)&&((inf1->ssh)++) == 0)
   {

if(GetAsyncKeyState(VK_LBUTTO
N)&& (((sp[t_b]->asize == sr) &&
(sp[t_b]->fill == 1)) || ((sp[t_b]-
>asize == br) && (sp[t_b]->fill ==
5))) && b_t_r && inf1-
>prev_click==0)
      {
         setcolor(sp[t_b]->color = 15);
//disappear the bubble(white)

score_keeper(UP_SCORE(sp,inf1,t_
b));    //score up
          inf1->prev_click = 1;
//after clicking correct, turn on the
booloen to freeze game state until
he/sheleaves the click button
      }


      else
if(GetAsyncKeyState(VK_RBUTTO
N) && (((sp[t_b]->asize == br) &&
(sp[t_b]->fill == 1)) || ((sp[t_b]-
>asize == sr) && (sp[t_b]->fill ==
5))) && b_t_r && inf1-
>prev_click==0)
      {
         setcolor(sp[t_b]->color = 15);
//disappear the bubble

score_keeper(UP_SCORE(sp,inf1,t_
b));
          inf1->prev_click = 1;
//after clicking correct, turn on the
booloen to freeze game state until
he/sheleaves the click button
      }
      else if(!b_t_r)
      {
         display_over("Your cursor is
outside target box",page);
         return 0;
//this function returns 0 if game is
over
      }

      else if(inf1->prev_click==0)
      {
         display_over("Wrong Click
for ballon",page);
         return 0;
      }
```

```
    }
    page = 1-page;
    delay(FRAME_DEL);
    return 1;                          //it
returns 1 if nothing happened
}
void target_box()
{
    int score = score_keeper(0);
    if(oTARGET_WIDTH >
SCREEN_WIDTH/10)
oTARGET_WIDTH =
SCREEN_WIDTH/3 - score;
    setcolor(RED);
    setlinestyle(0,0,3);
    rectangle((SCREEN_WIDTH-
oTARGET_WIDTH)/2, ycen-
TARGET_HEIGHT/2,(SCREEN_W
IDTH+oTARGET_WIDTH)/2,ycen+
TARGET_HEIGHT/2);
    setlinestyle(1,0,1);
    line(SCREEN_WIDTH/2,ycen-
TARGET_HEIGHT/2,SCREEN_WI
DTH/2,ycen-
TARGET_HEIGHT/2+TARGET_H
EIGHT/16);      //vertical line
obserable

line(SCREEN_WIDTH/2,ycen+TAR
GET_HEIGHT/2,SCREEN_WIDTH
/2,ycen+TARGET_HEIGHT/2-
TARGET_HEIGHT/16);
//vertical line obserable
    line(SCREEN_WIDTH/2-
oTARGET_WIDTH/2,ycen,SCREE
N_WIDTH/2-
oTARGET_WIDTH/2+oTARGET_
WIDTH/16,ycen);      //horizontal
line

line(SCREEN_WIDTH/2+oTARGE
T_WIDTH/2,ycen,SCREEN_WIDT
H/2+oTARGET_WIDTH/2-
oTARGET_WIDTH/16,ycen);
//horizontal line
}
s *allocate_the_memory()
{
    s *ptr = (s *)malloc(sizeof(s));
    if(ptr == NULL)
    {
        puts("Malloc failed");
        exit(1);
    }
    return ptr;
}
void new_game()
{
    GRAPH_INIT;
    s *sp[NO_BU+1];       // array to
store address  ,+1 is not for
terminating/ null char but is for error
free in comming lines
    inf inf1;
    inf1.ssh = inf1.saver1 =
inf1.prev_click= inf1.saver2 = 0;
    int i,GAP;
    setbkcolor(WHITE);
    for(i=0; i<NO_BU; i++)
    {
        if(i==0)
        {
            sp[i] =
allocate_the_memory();
            sp[i]->asize=rd();
//rd return radius of big or small
bullune
            sp[i]->curr_x=xcen;
            sp[i]->c=0;
        }
        sp[i+1] =
allocate_the_memory();  //No_BU
+1 balloon ko lagi ko size allocation
        sp[i]->fill = (rnd(1,100)%17 ==
0) ? 5 : 1;     //5 is stripe pattern of
some balloon in setfillstyle
        for(sp[i+1]->c=0,sp[i+1]-
>asize=rd(),sp[i]-
>color=rnd(0,14),inf1.indi=i+1,sp[i]-
>cy = 0,sp[i]->cx = 0,sp[i]-
>curr_y=ycen,sp[i+1]-
>curr_x=xcen;;)     //i is the identifier
of each bubblne here
        {
```

```
        int j;
        //runs only 1 time for one
bullune
        if(sp[i]->c==0)
        {
           if(sp[i]->asize == sp[i+1]-
>asize)   // if consecutive bubble are
of same size
           {
              if(sp[i]->asize == sr)
                 GAP = rnd(2*sr,
3*sr);      // max gap between
consecutive small bubble is 1.5 times
min GAP
              else
//if(s1.asize[i] == br)
                 GAP = rnd(2*br,
3*br);      // max gap between
consecutive big bubble is 1.5 times
min GAP
           }
           else
              GAP = rnd(sr+br,
2*sr+br);     // max limit is approx
1.5 times min vlaue for current value
        }
        if((sp[i]->c)>(GAP))     //c
for counter  //do not generqate until
minimum gap size is there ...baloon
is created when c for each balloon
increases till gap size
           break;              //after
break, index i increase
        if(bu_gen(sp, &inf1) != 1)
        {

free_rem_sp(sp,inf1.saver1,i+1);
//free all the memory remainig to free
up
           score_keeper(-
score_keeper(0));       //reset the
score to 0 for new game
           oTARGET_WIDTH =
SCREEN_WIDTH/3;       //reset the
obserbable target width backto initial

saving_keeper(START_SAVING-
saving_keeper(0,sp,&inf1),sp,&inf1)
;  //reset saving
              return;
// goback tocalling function i.e main
        }
        target_box();
//display target box
        score_keeper(0);
// display current score...vale of score
remaining the same
        for(j=(inf1.saver1);
j<(inf1.indi); j++)    //all the
necessery increments for making all
booluns moving //inf1.saver1=0
initially
        {
           (sp[j]->c)+=
step_process();          //sp[j]->c
controls natural speed i,e
accelearatingfrom left to right
        }
        if((xcen+sp[inf1.saver1]->c)
> getmaxx()+br)    //when balloon
goes out of screen and that distance
is equalt to br, then free the space
and saver value  is increased
        {
           free(sp[(inf1.saver1)]);
           inf1.saver1++;
        }
      }
   }
   getch();
   closegraph();
}
int ycalc(int nt,int n,int font=8,int
txt_size=5)
{
   int total_heit,y,ygap=10,txt_heit;
   settextstyle(font,0,txt_size);
   txt_heit = textheight("A");
   total_heit = nt*txt_heit + (nt-
1)*ygap;
   y = (SCREEN_HEIGHT/2 -
total_heit/2) +(n-1)*(txt_heit +
ygap);
   return y;
```

```
}
bool button(int y, char *but_nam)
{
   int txt_color;
   bool b_bool; //black by default
   settextstyle(8,0,5);
   b_bool = mousex() >
(SCREEN_WIDTH/2 -
textwidth(but_nam)/2) && mousex()
< (SCREEN_WIDTH/2 +
textwidth(but_nam)/2) &&
(mousey()) > (y) && (mousey()) <
(y+textheight(but_nam));
   txt_color = b_bool ? 9:0;   //green
or black
   b_but(SCREEN_WIDTH/2 -
textwidth(but_nam)/2, y, but_nam,
txt_color);
   if(b_bool &&
GetAsyncKeyState(VK_LBUTTON)
)
   {
      b_but(SCREEN_WIDTH/2 -
textwidth(but_nam)/2, y, but_nam,
12);      //red
      delay(B_DEL);

if(!GetAsyncKeyState(VK_LBUTT
ON))                        //extra
protection
         return 1;
//1 for successfulclick not click or
not on text
   }
   return 0;
//0 for unsuccessfulclick
}
int welcom()
{
   GRAPH_INIT;
   char title[] = TITLE;
   char we[][21] = {"Sajil
Awale","Shrey Niraula","Pratik
Luitel","Pujan Budhathoki"};
   int sep;
   int i, sum=0;
   int nt = 4;     //no of buttons
   // names
   {
      setbkcolor(WHITE);// for text
backgound color
      settextstyle(8,0,1);
      sep = (SCREEN_WIDTH -
textwidth(we[0]) -textwidth(we[1]) -
textwidth(we[2]) -
textwidth(we[3]))/5;    //
(screenwidth-total name length)/5
      for(i=0; i<4; i++)
//access string array by loop
      {

b_but(sum+sep,SCREEN_HEIGHT-
1.5*textheight(we[0]),we[i],i+2);
//x=sum+sep,
y=SCREEN_HEIGHT-
2*textheight(we[0]), i+2=color
         sum
+=(sep+textwidth(we[i]));
//sum allocates space eqaul to
sep+name length
      }
   }
   floodfill(SCREEN_WIDTH-
1,SCREEN_HEIGHT-1,WHITE);
// for white background
   // for title
   {
      settextstyle(3,0,7);
      b_but(SCREEN_WIDTH/2-
textwidth(title)/2,SCREEN_HEIGHT
/10,title,rnd(0,14));
   }
   for(char ne_gam[] = "New
Game",high_sc[] = "High
Score",close[] = "Exit",inst[] =
"Instructions",mini_game[] =
"Reflex Game";;)
   {
      //buttons
      if(button(ycalc(nt,1),ne_gam))
return 0;                  //New game
button
```

```
      if(button(ycalc(nt,4),high_sc))
return 1;                  //high score
button
      if(button(SCREEN_HEIGHT-
2*textheight(close),close))     return -
1; // exit button

if(button(ycalc(nt,2),mini_game))
return 2;                  //rxn game
      if(button(ycalc(nt,3),inst))
return 3;                  //instruction
button
      delay(INF_DEL);
   }
}
bool print_h_s()
{
   GRAPH_INIT;
   FILE *fptr;
   h_s_d h1;
   int i;
   char h_s_s[]= "High Scores";
   fptr =
fopen(H_S_LOCATION,"rb");
   cleardevice();
   setcolor(5);
   floodfill(SCREEN_WIDTH-
1,SCREEN_HEIGHT-1,WHITE);
// for white background
   setbkcolor(WHITE);
   if(c_fptr(fptr))    return 0;
   if(check_h_m_h_s_i_f() == 0)
   {
      char nofile[] = "Looks like it's
you first time";
      b_but(SCREEN_WIDTH/2-
textwidth(nofile)/2,SCREEN_HEIG
HT/2-textheight(nofile)/2,nofile,0);
   }
   // title i.e High score
   if(D_h_s_sort())    puts("High
score couldnot be sort");
//sort highscore in descending order
   settextstyle(8,0,5);
   b_but(SCREEN_WIDTH/2-
textwidth(h_s_s)/2,textheight(h_s_s)/
2,h_s_s,0);
   settextstyle(8,0,3);
   for(i=0;
fread(&h1,sizeof(h1),1,fptr) == 1;
i++)    //fread returns 1 if they were
sucessful
   {
      char buffer[300];
      sprintf(buffer,"%2d: %-7s : %d
:On
%s",i+1,h1.name,h1.score,h1.date_t);
      b_but(SCREEN_WIDTH/2-
textwidth(buffer)/2,2*textheight(h_s
_s)+(i+1)*textheight(buffer),buffer,0
);
   }
   fclose(fptr);
   for(char g_b_w_s[] = "Go Back";;)
   {
      if(button(SCREEN_HEIGHT-
1*textheight(g_b_w_s),g_b_w_s))
return 1;  //go back to welcome
screen
      delay(INF_DEL);
   }
   return 0;
}
void rxn_game()
{
   char mess[21] = "OUT OUT !!!";
   char ins[] = "Click bullble as soon
as bubble gets filled";
   int sum= 0,fill_color;
   GRAPH_INIT;
   floodfill(SCREEN_WIDTH-
1,SCREEN_HEIGHT-1,WHITE);
   setbkcolor(WHITE);
   settextstyle(8,0,3);
   b_but(SCREEN_WIDTH/2-
textwidth(ins)/2,SCREEN_HEIGHT/
5,ins,0);
   for(int j=0; j<RNX_BUBB; j++)
   {
      int i;
      setcolor(1);
      setfillstyle(1,15);
```

```
fillellipse(SCREEN_WIDTH/2,SCR
EEN_HEIGHT/2,br,br);
      fill_color = rnd(0,14);

while((GetAsyncKeyState(VK_LBU
TTON)))   delay(20);
      delay(rnd(1000,5000));

if((GetAsyncKeyState(VK_LBUTT
ON)))
      {
         //out out
         setbkcolor(15);
         settextstyle(3,0,5);
         b_but(SCREEN_WIDTH/2 -
textwidth(mess)/2,SCREEN_HEIGH
T/2 - textheight(mess)/2, mess, 0);
         delay(3000);
         return;
      }
      for(i=0;; i++)
      {
         setcolor(fill_color);
         setfillstyle(1,fill_color);
//COLOR(R,G,B)

fillellipse(SCREEN_WIDTH/2,SCR
EEN_HEIGHT/2,br,br);// bubbles

if(GetAsyncKeyState(VK_LBUTTO
N))
         {

if((sqrt(pow((SCREEN_WIDTH/2) -
(mousex()),2) +
pow((SCREEN_HEIGHT/2) -
(mousey()),2))) < br)
               break;
         }
         delay(1);
      }
      sum += i;
   }
   sprintf(mess,"Rxn time %.2f
ms",(float)sum/RNX_BUBB);
   setbkcolor(fill_color);
   settextstyle(8,0,5);
   b_but(SCREEN_WIDTH/2 -
textwidth(mess)/2,SCREEN_HEIGH
T/2 - textheight(mess)/2, mess, 15);
   delay(3000);
   return;
}
bool instruction()
{
   char inst[] = "Instructions",
line[][61] = {"Place mouse pointer
inside Targetbox", "Click Left mouse
button for Small Bubble","Click
Right mouse button for Big
Bubble","Reverse is true if bubble is
hatched","Click it as close to the
center of Bubble", "Game is Over
when you misclick it","Game is Over
if Saving is zero","Saving decreases
if bubble is ignored","Bonus Saving,
if center of bubble is clicked","Rxn
time is game to measure reflex
time"};
   int i;
   GRAPH_INIT;
   setbkcolor(WHITE);
   floodfill(SCREEN_WIDTH-
1,SCREEN_HEIGHT-1,WHITE);
// for white background
   settextstyle(8,0,5);
   b_but(SCREEN_WIDTH/2-
textwidth(inst)/2,textheight(inst)/2,in
st,0);
   settextstyle(8,0,3);
   for(i=0; i<10; i++)
   {
      char buffer[71];
      sprintf(buffer,"%2d: %-
45s",i+1,line[i]);
      b_but(SCREEN_WIDTH/2-
textwidth(buffer)/2,2*textheight(buff
er)+(i+2)*textheight(buffer),buffer,0)
;
   }
   for(char g_b_w_s[] = "Go Back";;)
   {
```

```
        if(button(SCREEN_HEIGHT-
1*textheight(g_b_w_s),g_b_w_s))
return 1;  //go back to welcome
screen
        delay(INF_DEL);
    }
    return 0;
    getch();
}
int main()
{
    int gd = DETECT, gm;
    check_h_m_h_s_i_f();        // to
create file if not created
    for(;;)
    {
        int r = welcom();       // r is the
returned value of mouse position by
function welcome
        closegraph();
        switch(r)
        {
        case 0:
            new_game();
            closegraph();
            break;
        case 1:
            if(print_h_s()==1)
            {
                closegraph();
                continue;
            }
            break;
        case 2:
            rxn_game();
            closegraph();
            break;
        case 3:
            if(instruction()==1)
            {
                closegraph();          //
closegraph not done in function
                continue;
            }
            break;
        case -1:
            closegraph();
                return(0);    //when exit is
clicked on welcome screen
        }
    }
    closegraph();
    return 0;
}
```