**University of Dhaka**

**Department of Computer Science and Engineering**

**CSE – 1211**

**LAB PROJECT: FINDING AHIRAF**

Submitted to:

Dr. Muhammad Asif Hossain Khan

Associate Professor

Ms. Anna Fariha

Lecturer

Submitted by:

Saad Ahmed Akash (AE-05) Md. Al-Zihad (AE-35)

Date-of-Submission: December 4th, 2014

**Table of Contents**

**Contents Page**

**1. Introduction 03**

**2. Project Name 03**

**3. Game objective 03**

**4. Project Outline 03**

**5. Main Features 03**

**6. Special Features 05**

**7. Additional Features 05**

**8. Graphical Enhancements and Coding 05**

**9. Coding Challenges 06**

**10. Graphical Interface 07**

**11. Coding Algorithm 10**

**12. Main Game Interface 12**

**13. Project Layout 13**

**14. Source Code 15**

**15. Conclusion 64**

**16. References 64**

**Introduction:**

The aim of the given project was to learn the way to implement basic level C programming into real life. There’s no alternative to learning coding other than leaning its applications and implementation. Implementation of what we learned in the C programming lab course was the aim of the project.

**Project name:**

Finding Ahiraf

**Game objective:**

To provide users with a funny experience, specially the fighting game-lovers will hopefully enjoy the game too much. The game contains a full duration of a certain amount of time with three different levels. To give user an experience of a complete 2D fighting game is the main objective of the game. It raises one’s ability to react in spontaneous situations as the enemy pops out in different positions approaching the user from different directions. So the game raises the spontaneity of the user.

**Project outline:**

A 2-D fighting game of single player with different levels to fight against many obstacles and enemies. It contains multiple features along with extra feature which are added later to make the game more exciting. The objective is to find a person named Ahiraf. There is a story outline which is displayed by high quality graphics at the beginning of each levels of this game. The story pop-ups as well as the increase of difficulty of hardness along with the progress of levels enhance its charm. The navigation and movement of the player will be controlled by the input from keyboard by the user while the navigation and movement with the spontaneity of the evil will be determined by using the basic level of “artificial intelligence” of computer.

**Main features:**

1. **Main Menu:**
2. Play Now Tab : When the user put a mouse click on this Tab, the game starts.
3. How To Play Tab : This tab shows the adequate game controls to the user.
4. Credit Tab : This tab includes the information of the game makers and the credits.
5. Exit Game Tab : User can simply exit the game by clicking on this tab.
6. **Story development:**

There’s a great enjoyable story of a classic tale of a kingdom, the king and the princess along with her one secret admirer. The sudden invasion of another cruel king and the change of the flow of occurrences really make the game an exciting one. The dialogues between people are also included to make things more realistic and exciting.

Every level starts with different story and different scenario which has made this game more alluring to the user. The dialogue boxes as well as the story dashboards have taken it to another level.

1. **The movement of the characters:**

It has been ensured that both the hero and the evil can move in both directions. We had to write over 1600 co ordinates for drawing the figures including both forward and backward directions. To make the opposite direction, the concept of rotation of the points in respect of an axis has been used in the code.

1. **The fighting positions of the hero and the evils:**

To make it a complete fighting game, different fighting positions have been introduced. The fighting positions are as follows:

* 1. The punching position
  2. The kicking position
  3. The jumping position
  4. The jump and kicking position

1. **The life bar:**

The different life bars of hero and evil have been introduced for the indication of the respective duration and state of the game at a certain point. The increase and decrease of life bars also vary from different conditions which has made this game as an improbable game too.

1. **Introduction of levels:**

This game provides the users with different levels having properties of different level of difficulties. The atmosphere also changes as the gradual change of the main storyline.

1. **Game over and Game loading functions**:

These functions are built and called at the proper places in the code so that whenever the life bar of the hero finishes the game comes to an end and shows a graphical representation like the actual game.

1. **Variance of difficulties:**

The gradual increase of difficulties has been arranged such in the order of Easy, Medium and Advanced respectively. These are introduced as the gradual development of levels and time.

**Special features:**

1. **Artificial intelligence:**

Implementation of artificial intelligence in the game has made the game of a single people against computer. The basic level of “artificial intelligence” of computer determines the navigation and movement of the evil. We set up the range in the code of the evil so that it seems that the evil sees the hero in a certain range and only after seeing, it starts to move.

1. **The spontaneity of the evil:**

The evil reacts differently in different situations. For a simple example, if the hero is standing in the forward direction, the evil also turns to him and then starts to fight back. Then again in the opposite scenario, the evil will turn automatically as per the position of hero and will carry out respective actions.

1. **Appearance of evils from arbitrary position:**

When each evil is dead, there comes another evil but from arbitrary position. For this, we had to use the function which generates random numbers. But at the same time, we also had to notice so that these points remain in the certain range of the game window.

**Additional features:**

1. **Leader board:**

We also added a leader board which will be shown at the end of the game if the user finishes the game by passing through all three levels.

1. **Exit option:**

The user can also exit the game whenever he/she wants by simply clicking on the exit option displayed in the Main Menu tab.

1. **Change of the appearance of evil:**

The appearance of evil will also change from level to level, this has been done by passing multiple parameter through the user defined functions using the new created header file “mystyle.h”

**Graphical enhancements and coding:**

* 1. **The animation of the sage :**

The whole story of a kingdom is told by an old sage, where it is seen the sage is telling the story to the user. The work of animation is done by an efficient application of C coding.

* 1. **The animation of the story dashboard:**

The animation of the appearance of the story in the story dashboard has also been included flawlessly to keep the continuation for the visual comfort. The uses of proper loop arguments and loop limits have enhanced the graphics part too much.

* 1. **The uses of delay() function:**

The proper uses of arguments of delay() functions have been used and mo dified many times just to reduce the problems faced to establish a great graphics as well as an enjoyable game.

* 1. **The editing of dashboard:**

This game includes a special story line which will enhance the game’s charm to the users. This story is about a kingdom, its king & the princess. The invasion of the vicious king of the neighboring kingdom, the exciting flow of occurrences has been manifested by the perfect work of graphics.

* 1. **The appearance of story:**

The full story and the dialogues had to appear in the particau box area of the dashboard. So we had to take the arguments of readimage(); function very carefully so that not to exceed the limit.

**Coding challenges:**

1. **Division into modules:**

It was a huge project, it should be well organized. In order to keep the whole code well arranged, the entire code must be divided into modules. But in this case, handling each of the modules and passing parameters among these functions and the main functions are not easy tasks either.

1. **Implementation of artificial intelligence:**

Since we proposed a 2D game which will be played between computer and a user, we had to browse a lot to know about the Implementation of artificial intelligence in the BGI (Borland Graphics Interface).

1. **Drawing different fighting positions by coding:**

As BGI doesn’t support Bitmap image as input, we had to draw images manually by taking each and every co ordinates continuously to create these shapes of both hero and evil.

1. **Screen limit:**

The hero and the evil were not to be moved out of the screen. So we had to put necessary limits in the code to debug.

1. **Flickering:**

BGI displays too much flickering when any image is taken as input as a background. So the diagrams had to be drawn manually and then convert them in respect of only two points just to reduce flickering, which was not an easy task at all.

1. **Maintaining the speed:**

When too much delay() function is used in order to reduce flickering, it slows down the game. So to maintain the speed and reduce flickering at the same time was also a challenge when coding.

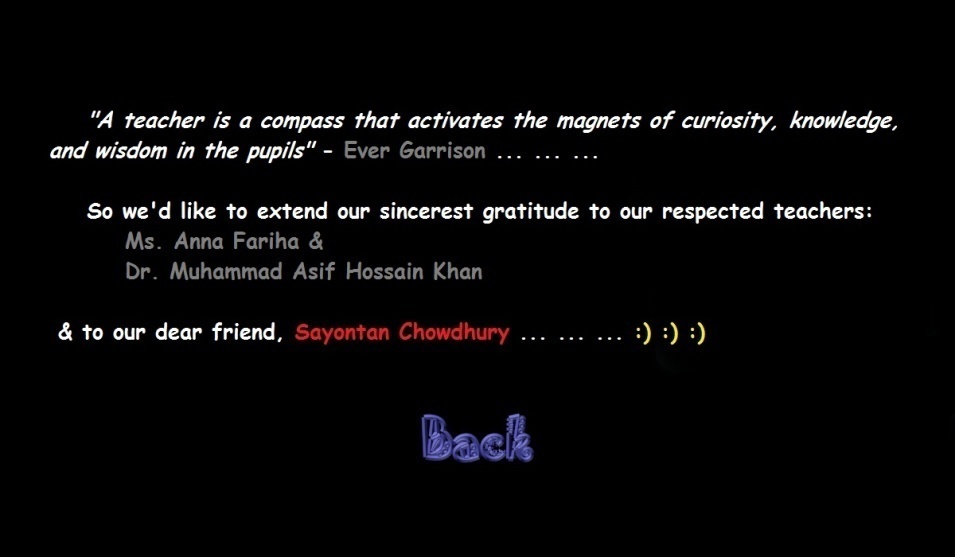
**Graphical interface:**

1. Main Menu:

****

1. Credit window:





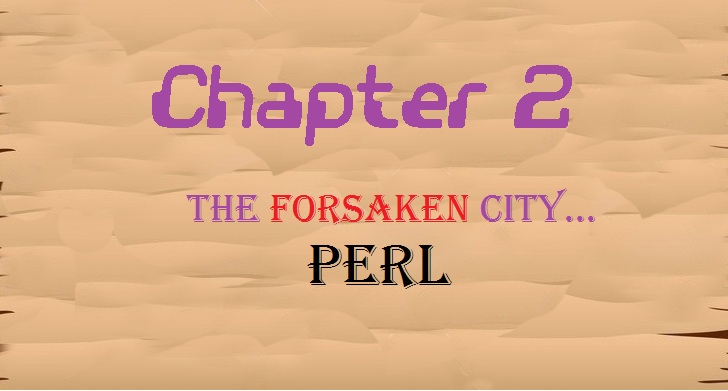
1. How to Play:



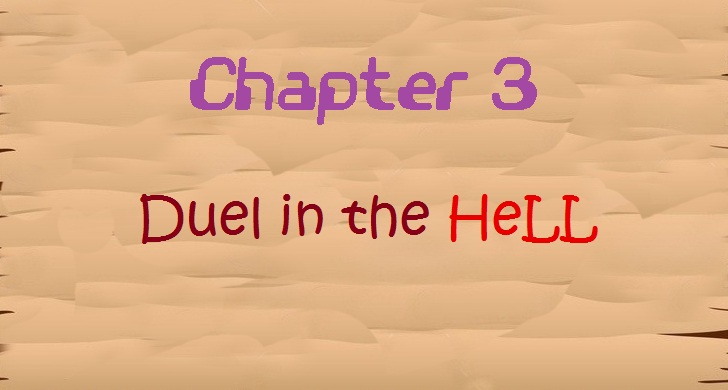
1. Story:
2. Chapter 1:



1. Chapter 2:

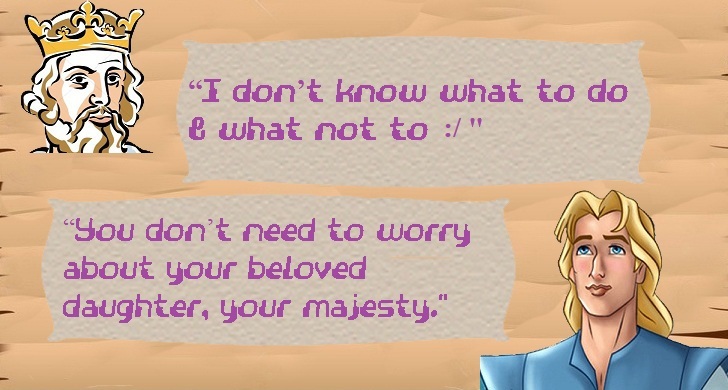


1. Chapter 3:

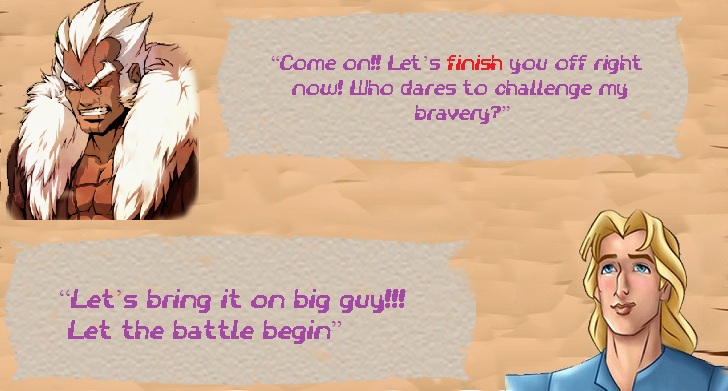


1. Dialogue box:

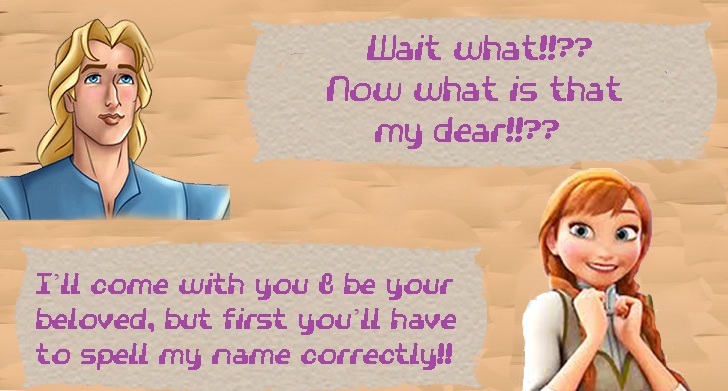
Dialogue between king and the hero-



Dialogue between the evil and the hero-



1. The end:



**Coding algorithm:**

1. Use of extra header file:

To divide the whole code into many modules, we had to use many user-defined functions. To assemble them, a header file “mystyle.h” has been created in Visual C++.

1. User-defined functions:
2. int printmenu():prints the menu window
3. int printcredit():prints the credit window
4. int printhowtoplay():prints the howtoplay window
5. void gameover():executes the end of levels and return to the menu
6. void herostillpositionplus(int localx, int localy): prints the forward state of hero
7. void herostillpositionminus(int localx, int localy): prints the backward state of hero
8. void herofightpositionplus( int localx,int localy): prints the forward punching state of hero
9. void herofightpositionminus( int localx,int localy): prints the backward punching state of hero
10. void evilpositionstillplus(int localx, int localy, int lv2): prints the forward state of evil
11. void evilpositionstillminus(int localx, int localy, int lv3): prints the backward state of evil
12. void evilpositionhitplus( int localx,int localy, int lv): prints the forward punching state of evil
13. void evilpositionhitminus( int localx,int localy, int lv1): prints the backward punching state of evil
14. void herokickminus( int localx,int localy): prints the backward kicking state of evil
15. void herokickplus( int localx,int localy): prints the backward kicking state of evil
16. int level1(int levelnum): organizes every level and passes all co ordinates among defined functions
17. void story1(): prints the story before 1st level and controls the animation of the wise man or the storyteller
18. void story2():prints the story before 2nd level and controls the animation of the wise man or the storyteller
19. void story3():prints the story before 3rd level and controls the animation of the wise man or the storyteller
20. void endstory():prints the finishing story after 3rd level and the controls the animation of the wise man or the storyteller
21. void gameload(): loads the game, creates a animated sphere and prints our name
22. Functions of the header file “graphics.h”:

**void cleardevice (void):**

**Refreshes the graphical screen.**

**void circle (int x, int y, int radius):**

**This function draws a circle.4**

**void initwindow (int width, int height):**

**The function initializes the graphics system by opening a graphics window of the specified size.**

**void delay (int millisec):**

**The function pauses the computation for the the specified number of milisec**

**void fillellipse (int x, int y, int xradius, int yradius):**

**Draws an ellipse using (x,y) as a center point and xradius and yradius and vertical axes, and fills it with the current fill color and fill pattern.**

**void fillpoly (int numpoints, int \*polypoints):**

**It draws the outline of a polygon with numpoints points in the current line style and color (just as drawpoly does), then fills the polygon using the current fill pattern and fill color.**

**void settextstyle (int font, int direction, int charsize):**

**This function the style, direction and size of a text.**

**void outtextxy (int x, int y, char \*textstring):**

**This function shows message on a given coordinate.**

**void readimage(int x,int y,int x2,int y2):**

**This function takes only JPEG and BITMAP image.**

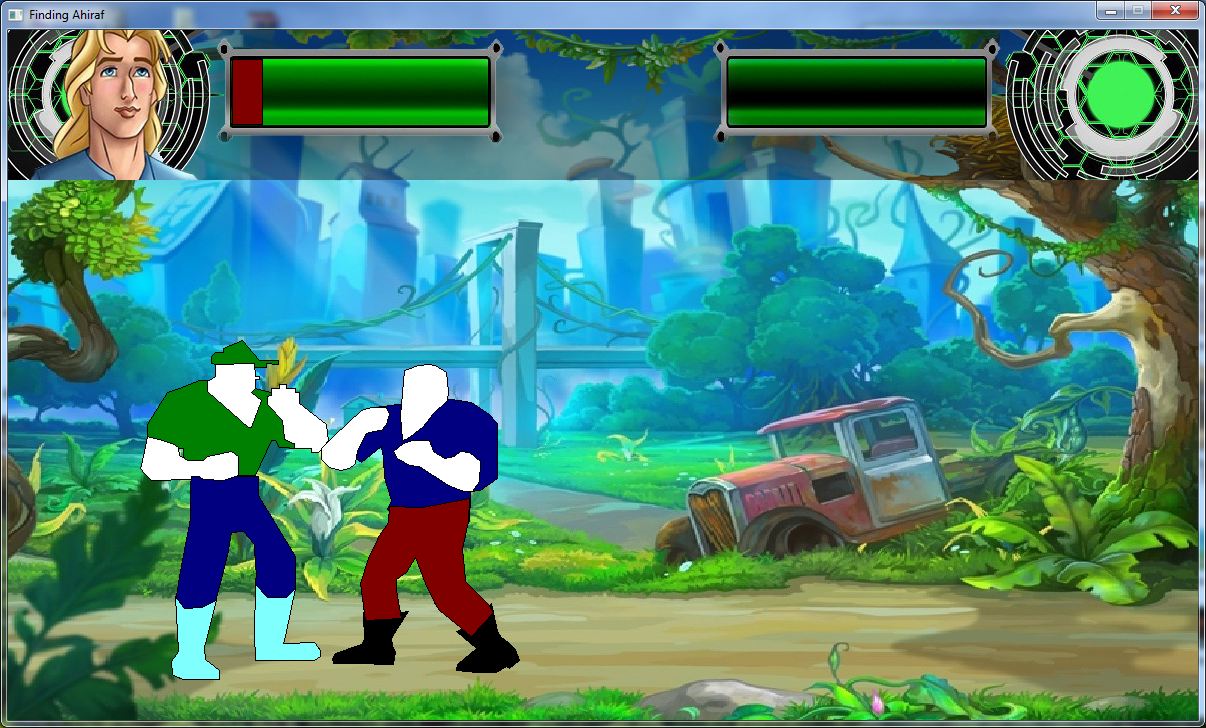
**int kbhit (void);**

**The function returns true (nonzero) if there is a character in the input buffer ready to read. Otherwise it returns false. In order to work, the user must click in the graphics window.**

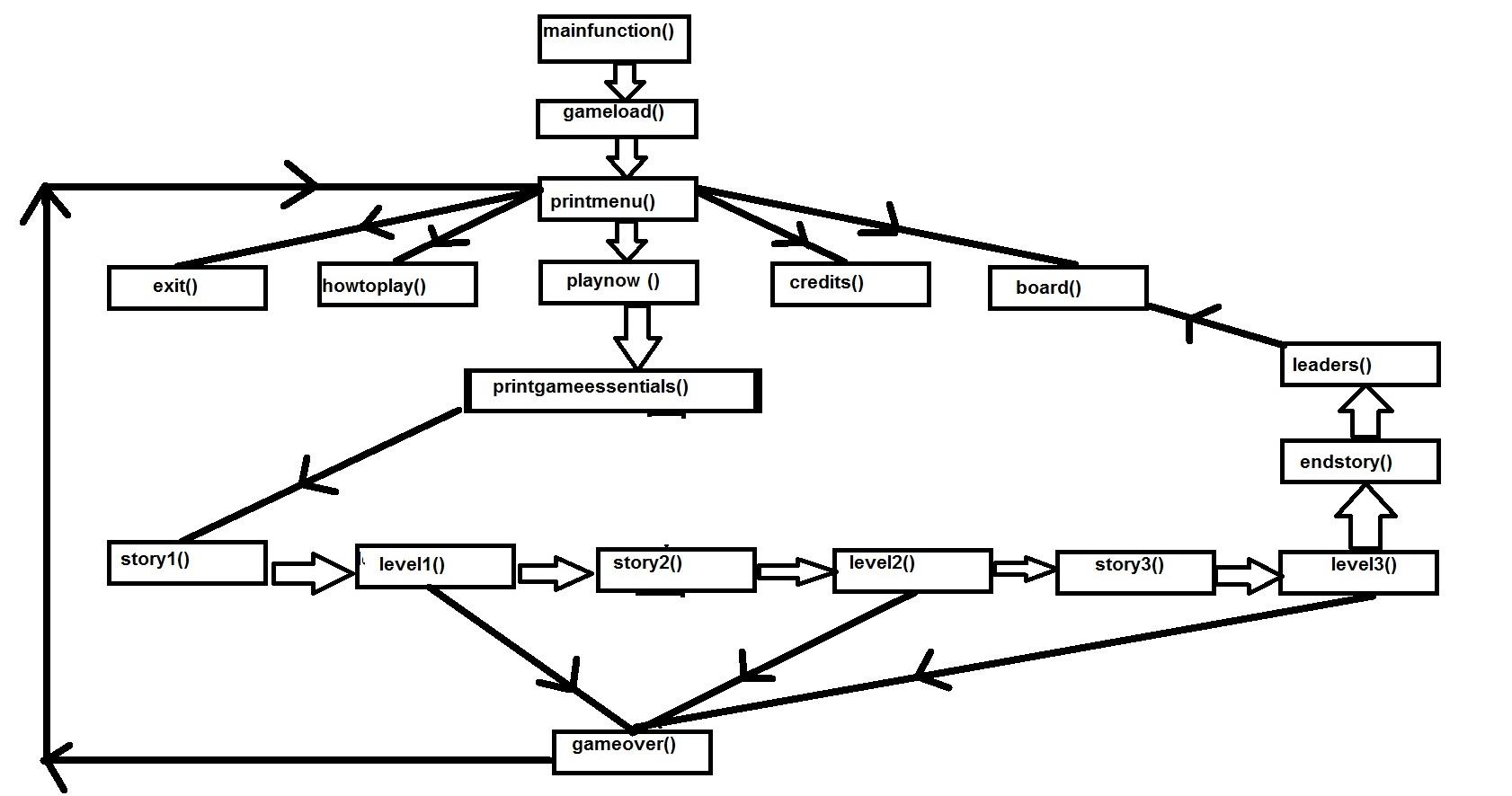
**void setfillstyle (int pattern, int color):**

**This function imposes the color and pattern of filling.**

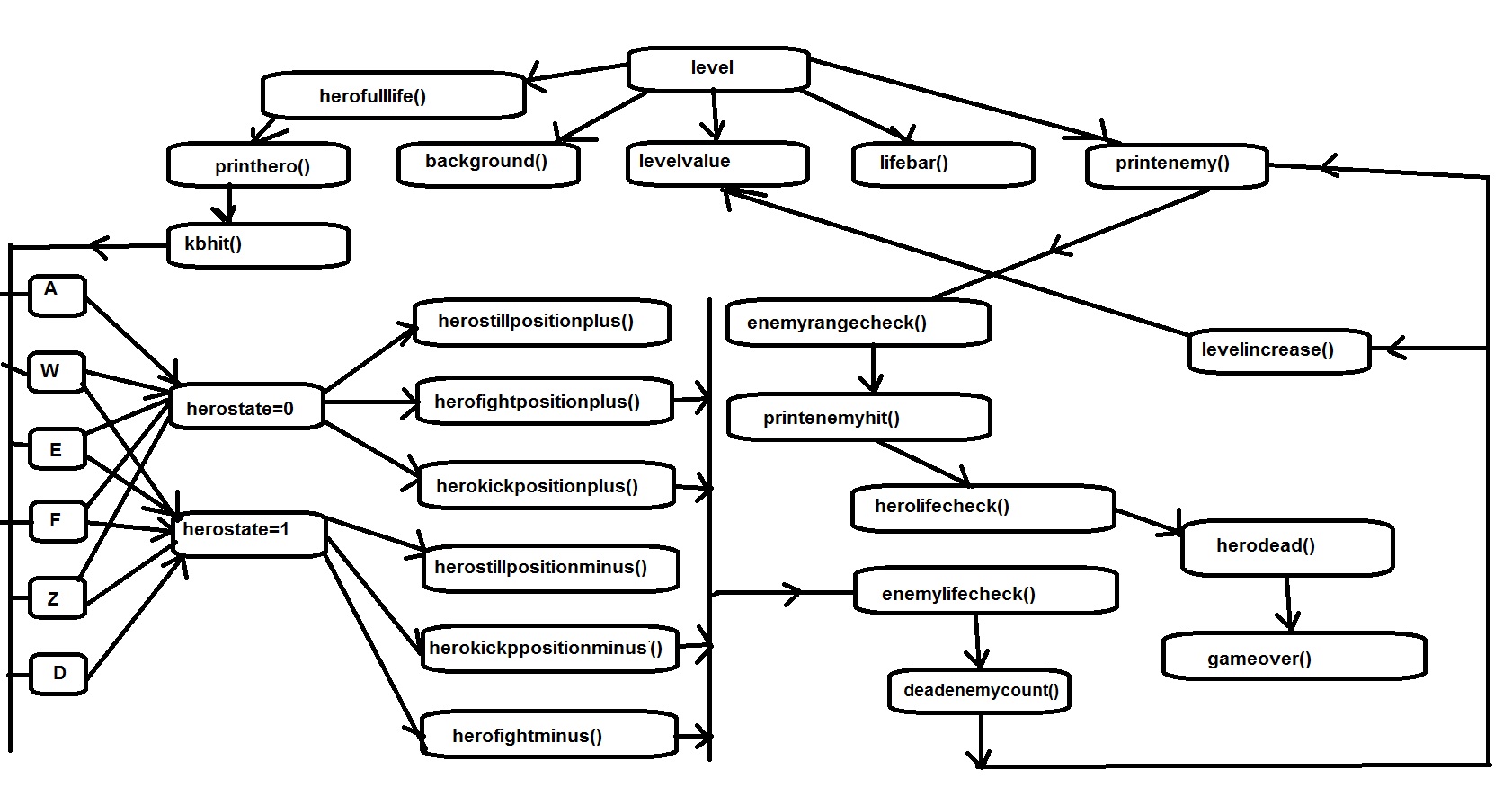
**Main game interface:**



**Project Layout:**



Flow chart 1: The inter relation of functions of the entire game



Flow chart 2: The inter relation and order of functions of levels

**Source Code:**

Mainfunction.cpp:

#include "graphics.h"

#include "mystyle.h"

int heroglobalpositionx=0, heroglobalositiony=450,enemyglobalpositionx=1200, enemyglobalpositiony;

int herolifebarchangex=550,herolifebarchangey=450,enemylifebarchangex,

enemylifebarchangey;

// this function actually prints the game like things. such as: loading game, game over , try again etc.

// added this function with a plan that it will help us to create multiple levels of our game

// we will call the level1().. function from this printgameessential function

int printgameessentials()

{

delay(1200);

int level;

int ghorardim;

ghorardim=story1 ();

level=level1(1); // level1 function will return 0 or 1, it's the main game function

if (level==0)

{

gameover();

delay(10);

return 1;

}

ghorardim=story2();

level=level1(2);

if (level==0)

{

gameover();

delay(10);

return 1;

}

ghorardim=story3();

level=level1(3);

if (level==0)

{

gameover();

delay(10);

return 1;

}

endstory();

return board();

}

**// The rest of the code is omitted by Anna Fariha**

**Conclusion:**

In spite of the tremendous hard work, we had a lot of fun during this project as well as learned the proper implementation of basic C code for animation, graphics and a wide variety of tasks. As our first attempt to make such a game, we hope that users will also enjoy playing this game the same way as we did while making it. We also look forward to make more games.

**Reference:**

1. C The Complete Reference (4th Edition) By Herbert Schildt
2. Programming in ANSI C By E.Balaguruswami
3. Let Us C By Yashwant Kanetkar
4. http://www.stackoverflow.com
5. <http://www.cs.colorado.edu>
6. <http://www.programmingsimplified.com>
7. <http://en.wikipedia.org/>
8. <http://www.mycplus.com/tutorials/c-programming-tutorials/graphics-programming>
9. <http://cprogrammingexpert.com/C/Tutorial/graphics.aspx>