TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DESKTOP ENVIRONMENT – UNIVERSAL (DE-U)

A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE PRACTICAL COURSE ON OBJECT ORIENTED PROGRAMMING [CT 451].

SUBMITTED BY:

1. **ANJU CHHETRI (PUL076BEI005)**
2. **ASHMA YONGHANG (PUL076BEI007)**

SUBMITTED TO:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

NOVEMBER 2020 A.D

Abstract

The main objective behind this project is to mock user interface for our desktops. These interfaces has a basic features of setting password in the login interface and in the consecutive desktop interface it has simple demonstration of to-do-list and aeroplane combat game. We also have time display in each of our interfaces. Basically in the aeroplane combat game our one fighter plane is to shoot enemy fighter plane. For the shooting function we take the keyboard input(player can shoot using space bar) and for the left/right movement of our fighter plane also we take input from keyboard(left key for left and right key for right). This way we have tried to see this project from more of a perspective of fun-to-do project where we try to explore the dimensions of game logics and interfaces side by side implementing cpp knowledge we have gained so far. This whole thing is achieved by using qt framework and exploiting the concepts of inheritance(to make custom classes for making objects like our fighter plane and enemy fighter plane, bullet, score display ,health display, gamretc.).

Keywords: user interface, aeroplane combat game, fun-to-do project etc.

ACKNOWLEGMENT

We would like to express our special thanks of gratitude to our teacher Bikal Adhikari who provided us such a wonderful opportunity to extend the boundaries our knowledge in the field pf programming. We are highly indebt for his guidance and supervision, as well as for providing necessary information regarding this project.

We would also like to express our gratitude to our families for supporting and encouraging us during this time.

Table of Contents

List of Figures

ABBREBIATIONS

GNOME: GNU Network Object Model Environment

GNU: GNU's Not Unix

GUI: Graphical User Interface

IDE: Integrated Development Environment

KDE: K Desktop Environment

MOC: Meta Object Compiler

UI: User Interface

CHAPTER 1: INTRODUCTION

1.1 Background

In today's date mankind and technology have come closer than ever before. Nowadays technology has created a great impact in everyone's life. Laptop, computer and mobile phones are some typical example of today's technology and if we give a closer look to those wonderful devices, we find some features really similar among them. One of such features is their Graphical User Interface (GUI). GUI has enabled people of all age, education and profession to make the best use of their devices. Imagine our interaction with those devices without any GUI, feels impossible right?

Windows is one of the most popular operating system in today's date and one of its most prominent features is its GUI. Its GUI is so user friendly that people from every profession and age find it really effective and productive. Technology is the future and a better technology requires a better interface. Desktop and laptops are an important part of our future and Desktop environment makes interaction with such devices possible for everyone. Desktop environment is a collection of software which provides Graphical user interface and runs on top of operating system. KDE and GNOME are some example of desktop environment.

1.2 Objective

Its main objective are enlightened below:

1. To provide an easy-to-use interface to the user.
2. To make our knowledge of C++ programming language better.
3. To understand what happens under the hood of any GUI program.

1.3 Limitations

1. User cannot customize the environment as per their need.
2. Availability of less widgets and application for user.

CHAPTER TWO: PROBLEM ANALYSIS

2.1 System Requirement

2.1.1 Software Requirement

For the development of this program Qt framework is used. Though its coding can be done in any IDE like visual studio due to the efficiency and faster code execution of code in Qt's own IDE this project was purely created using QT's own tools. Windows was used as its OS platform. Its source code is portable across any Qt creator 4.13.2 version under MinGW 64-bit compiler.

2.1.2 Hardware Requirement

This program is economically feasible program so it doesn't require a lot of hardware resources. A laptop with x64-based processor and following specifications are enough to run this program in Qt.

1) 256 MB of RAM
2) 500 MHz CPU, 1 GHz preferred for 60-FPS velvet-smooth UI
3) OpenGL ES 2.0 support *

2.2 Feasibility Analysis

2.2.1 Economic Feasibility

Economic feasibility analysis is the most commonly used method for determining the efficiency of a new project. It is also known as cost analysis. It helps in identifying profit against investment expected from a project. Cost and time are the most essential factors involved in this field of study. This program is economically highly feasible program since it is written in C++ programming language Qt framework. so economically it is highly satisfying with minimum time consumption.

2.2.2 Technical Feasibility

Technical feasibility study is the complete study of the project in terms of input, processes, output, fields, programs and procedures. It is a very effective tool for long term planning and trouble shooting. The technical feasibility study should most essentially support the financial information of an organization. It is technically well managed program whose source code can be easily modified for better improvement and for debugging any error.

2.2.3 Operational Feasibility

Operational feasibility refers to the measure of solving problems with the help of a new proposed system. It helps in taking advantage of the opportunities and fulfills the requirements as identified during the development of the project. It takes care that the management and the user support the project.it is very easy to use simply a person who understand English language can use this program.

Chapter three: REVIEW OF RELATED LITERATURES

The **history** of video **games** goes as far back as the early 1950s, when academic computer scientists began designing simple **games** and simulations as part of their research or just for recreation. At M.I.T. in the 1960s, professors and students played **games** such as 3D tic-tac-toe and Moon Landing[1].

Qt has also been exploited for developing some of the games like qt-nethack, sudoku,Bubble-chains, piperwar,iQ puzzle, dust-racing 2D,GNUDoQ,Attal : Lords of doom,Hedgewars,Gottet,JAG,Construqtor,OpenTowerDefence,BattlefleetBox Poker,qArkanoid etc[2].

Our aeroplane combat game is based on some of the basic features of qt and some properties of cpp like inheritance. This is our meagre endeavour to plant the seed in our mind,of how games use the logics.

As we dive in the history of desktop interface,we come to see, Engelbart's work directly led to the advances at Xerox PARC. Several people went from SRI to Xerox PARC in the early 1970s. In 1973, Xerox PARC developed the Alto personal **computer**. It had a bitmapped screen, and was the first **computer** to demonstrate the **desktop** metaphor and graphical **user interface** (GUI)[3].

Rather than exploiting the details we attempt to see how qt can fulfill the purpose of creating those interfaces at the most convenient level as these are our initial approaches to understand those interfaces explicitly with qt[4].

Chapter Four: ALGORITHM DEVELOPMENT, FLOWCHART and TREE DIAGRAM

3.1 ALGORITHM

An algorithm is a set of instructions designed to perform a specific task. By using highly-efficient algorithms, developers can ensure their programs run as fast as possible and use minimal system resources. Of course, not all algorithms are created perfectly the first time. Therefore, developers often improve existing algorithms and include them in future software updates. Algorithms for the first version of DE-U are written below.

3.1.1 Algorithm for Login Window

STEP 1: Start
STEP 2: Enter your username and password
STEP 3: username=”project” and password =”HelloWorld” ?
    If yes go to STEP 4
    If no go to STEP 5
STEP 4: Desktop Window
STEP 5: Try again?
    If yes go to STEP 2
    If no go to STEP 6
STEP 6: End

3.1.2 Algorithm for Desktop Window

STEP 1: Start
STEP 2: Choose between To-do-list and Aero Shoot
    If To-do-list go to STEP 3
    If Aero-shoot game go to STEP 4
    If log out go to STEP 6
    If Restart go to STEP 5
STEP 3: Open To-do-list
STEP 4: Open Aero Shoot game
STEP 5: Login Window
STEP 6: End

3.1.3 Algorithm for Game Window

STEP 1: Start
STEP 2: Initialize Score=0 and Health =3
STEP 3: Shoot enemy
STEP 4: collision occurs?
If yes go to STEP 5
If no go to STEP 7
STEP 5: Score = Score+1
STEP 6: Go to STEP 3
STEP 7: Health = Health -1
STEP 8: Health <1?
If yes go to STEP 9
If no go to STEP 3
STEP 9: Display Game Over
STEP 10: Play again?
If yes go to STEP 2
If no go to STEP 11
STEP 11: End

3.2 FLOWCHART

A **flowchart** is a graphical representation of decisions and their results mapped out in individual shapes that were first developed by Herman Goldstine and John Von Neumann in the 1940s. Flowcharts can provide a step-by-step diagram for mapping out complex situations, such as programming code or troubleshooting problems with a computer. After the completion of algorithms flowchart were designed. Flowchart for following windows are shown below:

1. Login Window

2. Desktop Window

3. Game Window

3.2.1 Flowchart of Login Window

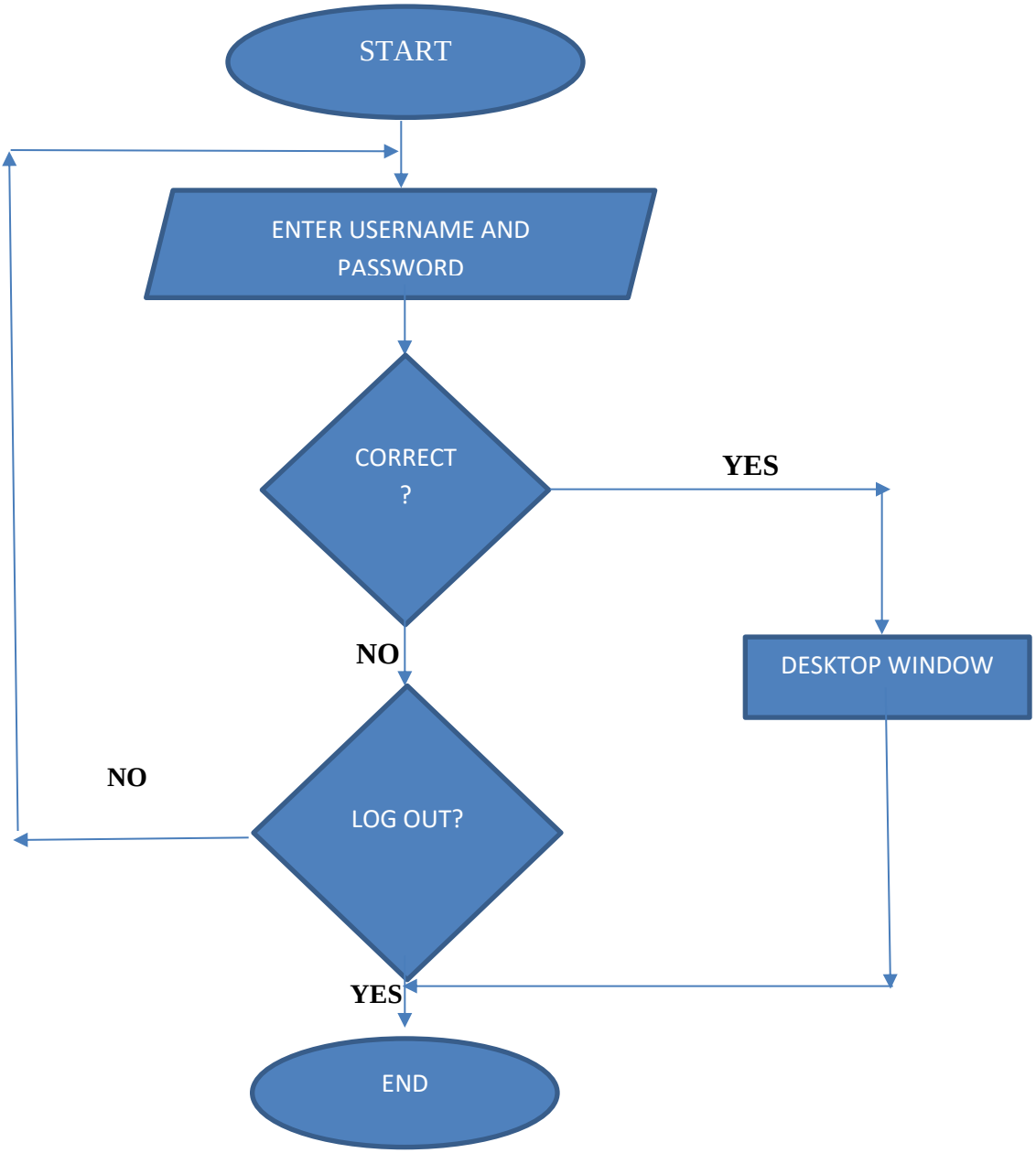


FIG: 1.1

3.2.2 Flowchart for Desktop Window:

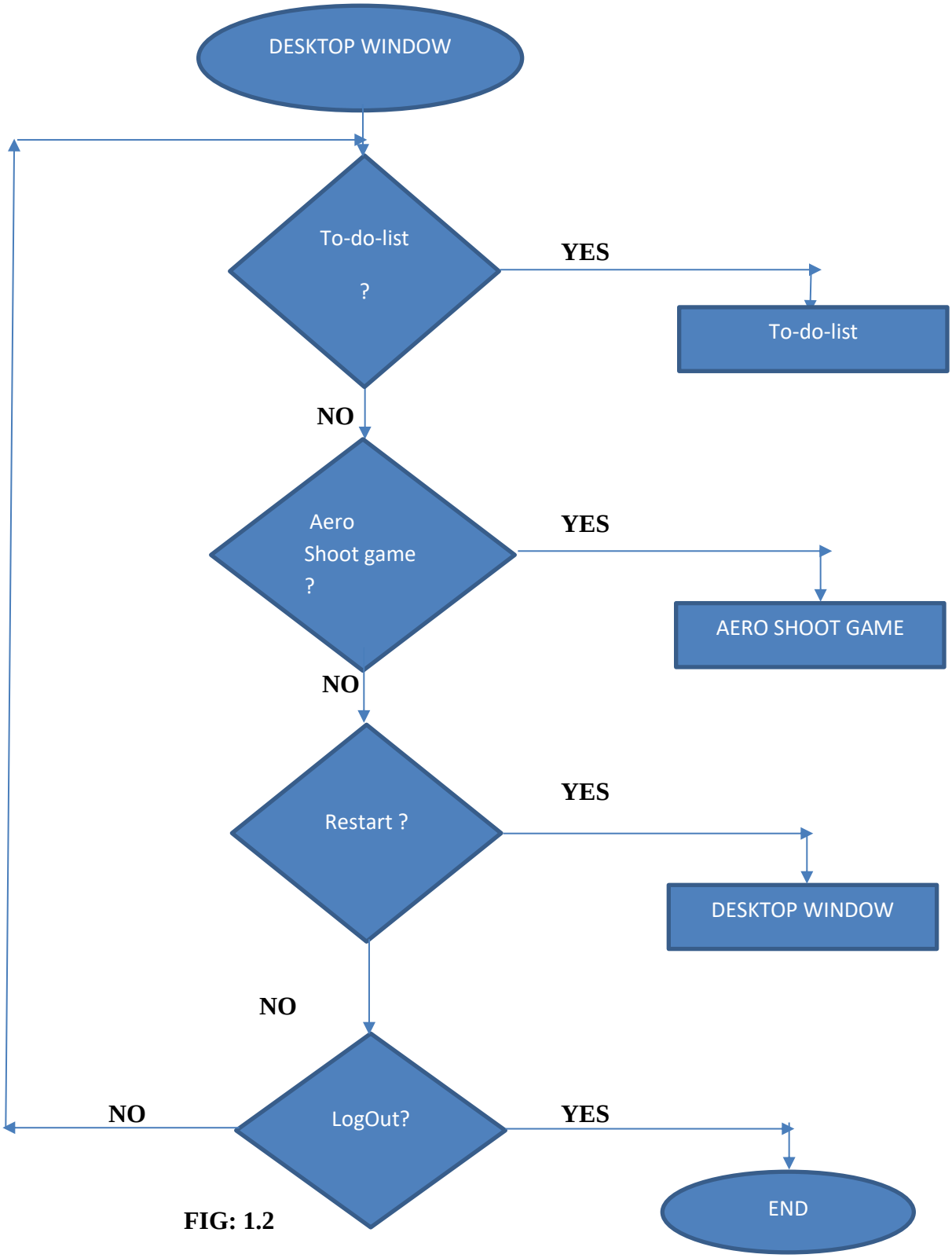
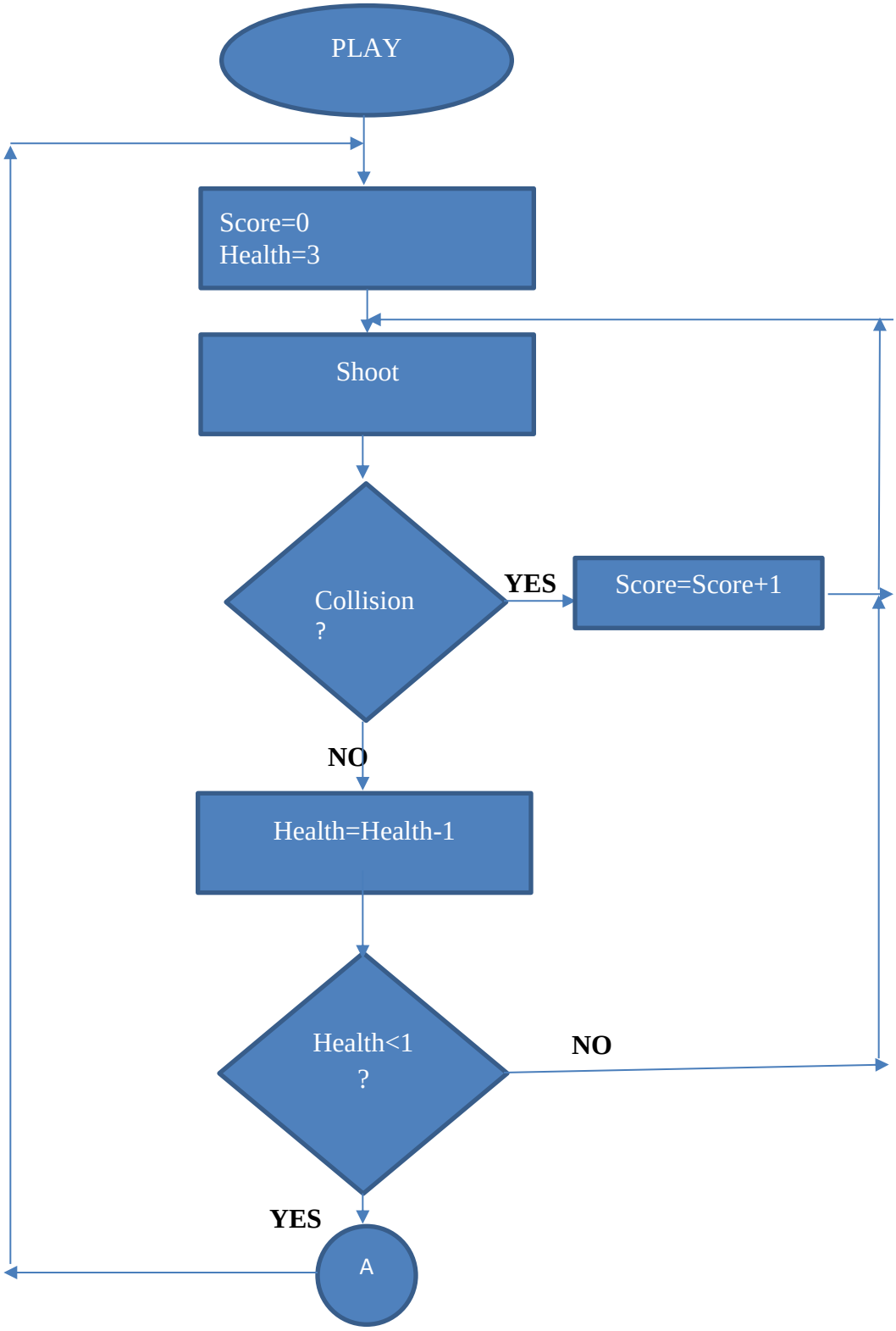


FIG: 1.2

3.2.3 Flowchart of Aero Shoot:



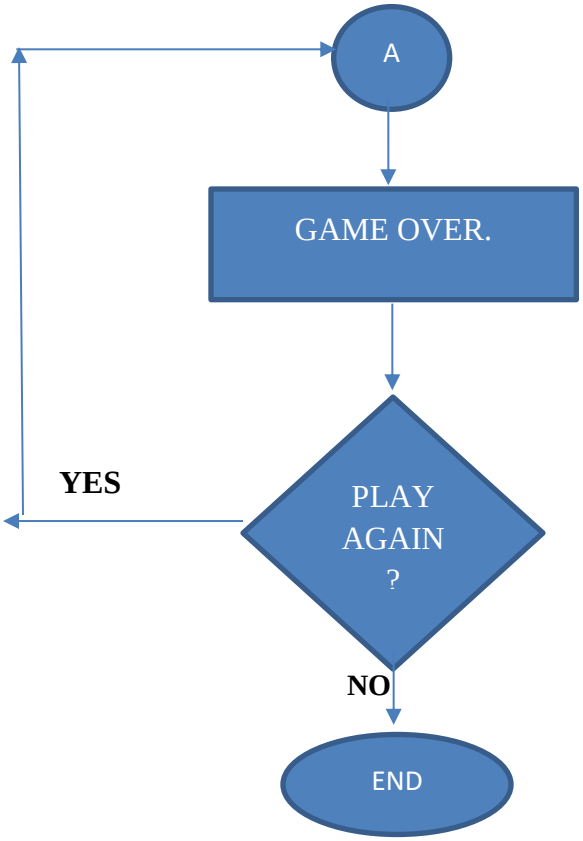


FIG: 1.3

3.3 Inheritance tree diagram

3.3.1 User Interface

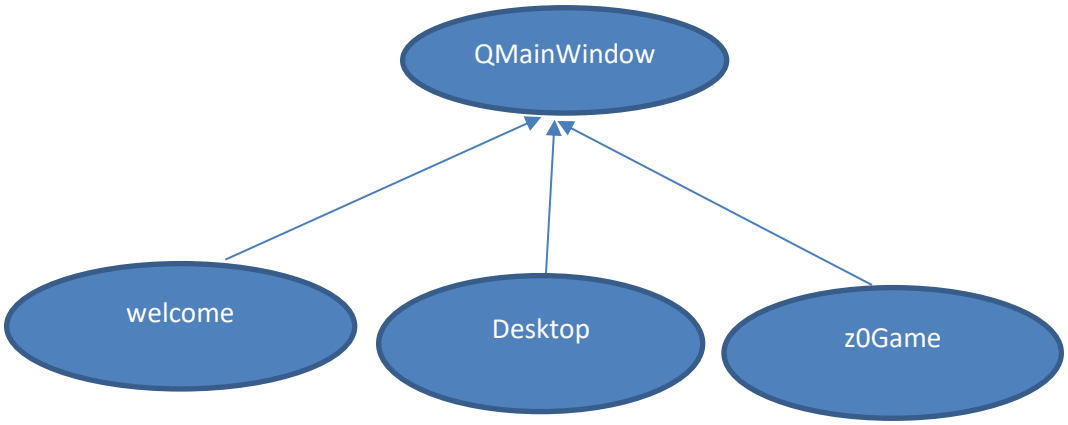


FIG: 2.1

3.3.2 Classes in Game

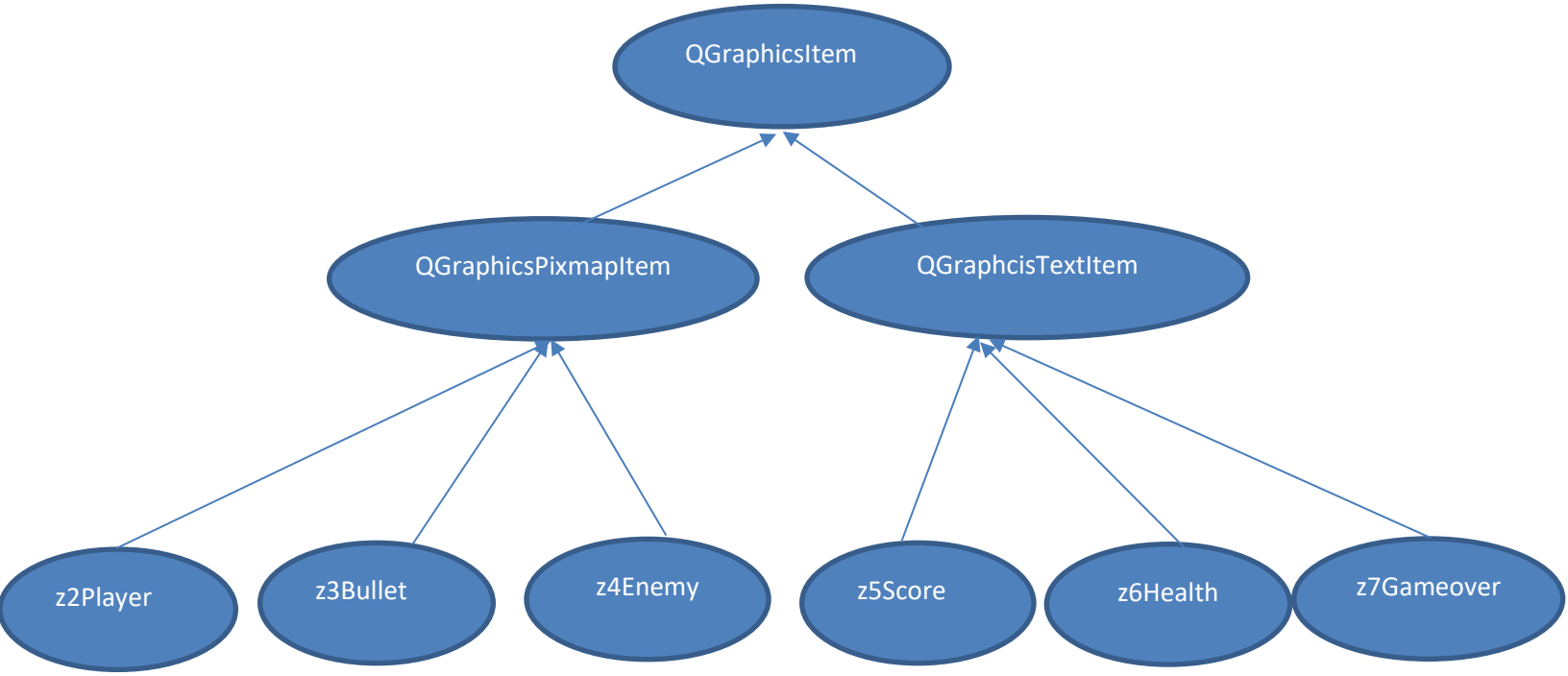


Fig: 2.2

CHAPTER FIVE: IMPLEMANTATION AND CODING

4.1 Implementation

4.1.1 Introduction

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

SCREENSHOTS:

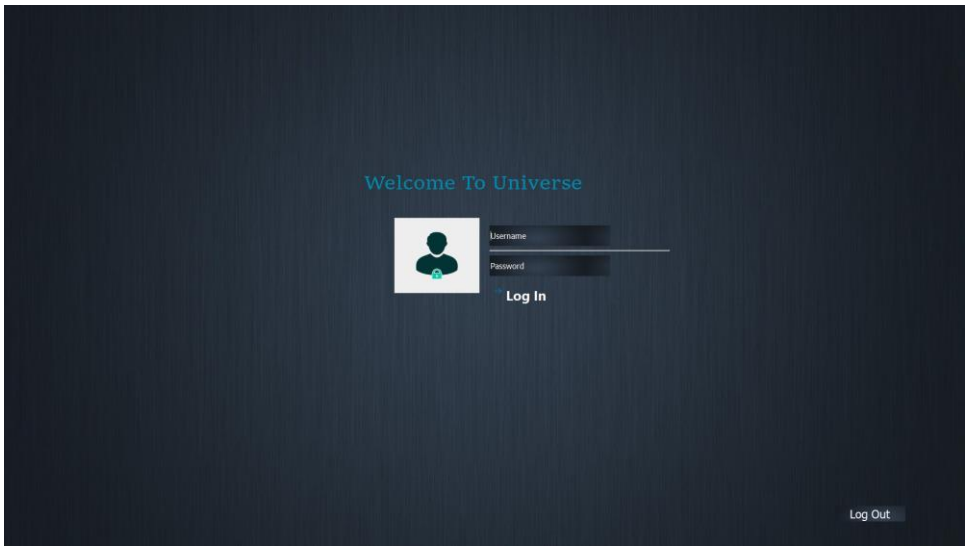


Fig: 3.1
LOGIN WINDOW

This the first UI window this program. It asks for username and password with and user and if correct it takes them to the next UI window.

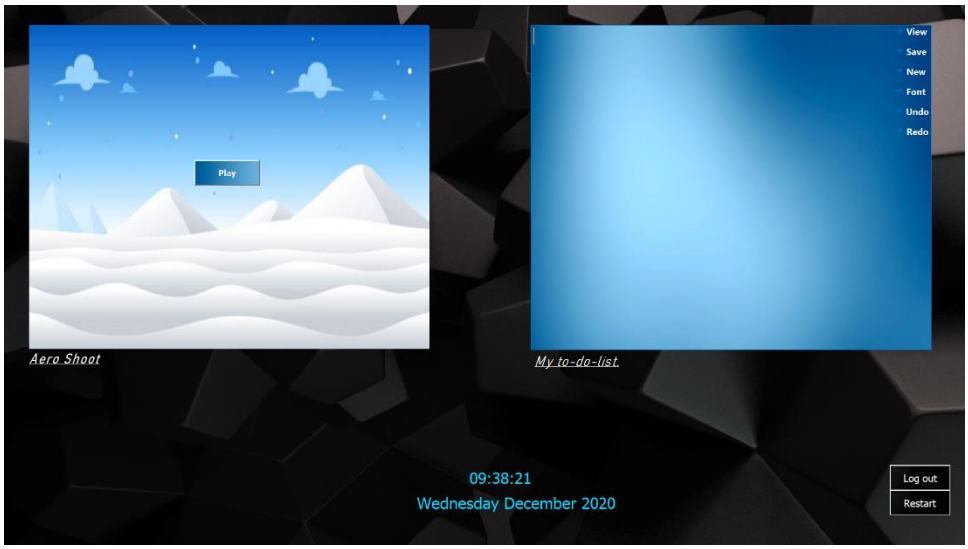


FIG: 3.2
HOME SCREEN

This the Second UI window where the user is taken to. Here the user has option to create their own to do list or play aero shoot game. User can also log out or go to login window as per their wish.

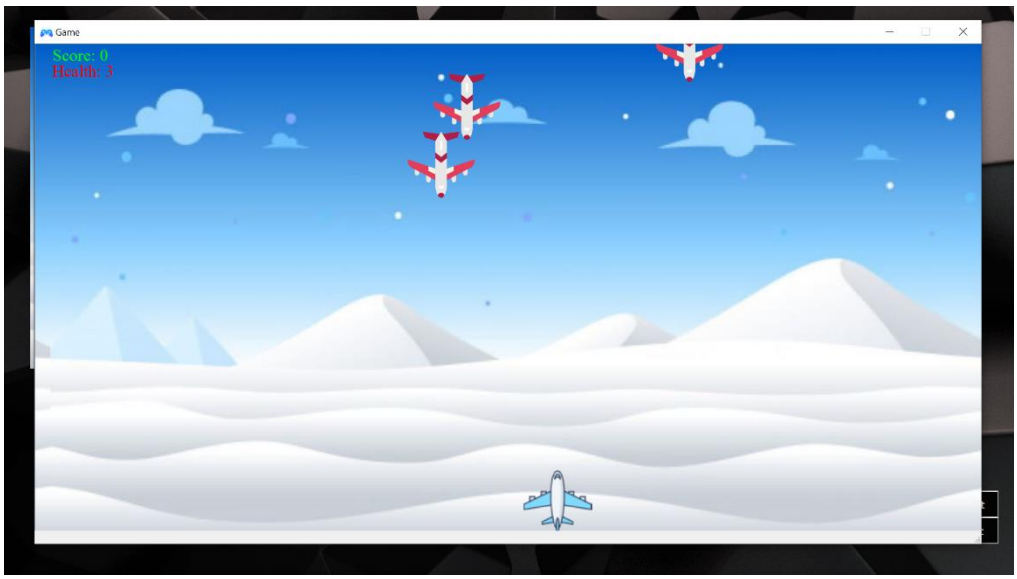


FIG: 3.3
GAME WINDOW

If the user decides to play aero shoot game then this is the third UI window that is presented to them. Here player has to shoot the coming airplanes and the score and health are displayed on the screen.

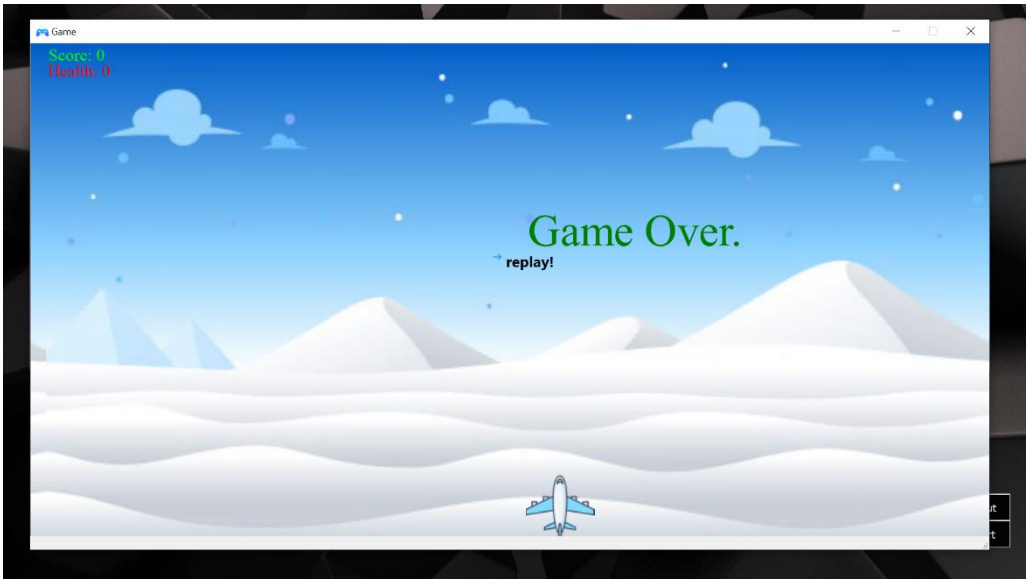


FIG: 3.4
GAME OVER

In the game over screen user has two option either they can play again or press ctrl+q (or press X button) to exit the game window.

4.2 CODING OF THE PROJECT:

The program is written using Qt framework. Qt uses a command line tool called qmake tool that parses these project files in order to generate "makefiles", files that are used by compilers to build an application. This project file is created and changed by Qt creator itself, unless absolute necessary the user do not need to change the contents in this file.

Minimal code found in .pro file.

QT = core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

SOURCES += main.cpp

Core gui is the library and SOURCES show the list of .cpp files present in the program.

Here the files can be divided into 4 files.

1. Header file (.h file)
2. Source file (.cpp file)
3. Forms (.ui file)
4. Resource file (.qrc file)

Namespace, class and function (optionally pointers) here are declared first in Header file and later defined in their respective source file. All the necessary graphics item are created in .ui file and as per the requirement coded in the source file. The object of the graphics item in .ui file is created in a header file in a separate folder by Qt itself. Resource files contain images and audio used in the program.

The very first program of this project is main.cpp. Main.cpp of Qt starts the event loop. Event loop here waits for keyboard input, network traffic, timer events etc. and responds accordingly.

Code Snippet of main.cpp:

```
#include <QApplication>
int  main(int argc, char *argv[]){
QApplication a(argc, argv);
/* code here*/
return a.exec();
}
```

Here a.exec() starts the event loop of the program. main.cpp here calls the welcome.cpp using a pointer. welcome.cpp is responsible for showing login interface. All the files of this program are at divided into two parts header and source part. All the class and function declaration are made in header file.

Code snippet of EXAMPLE.h file

```
#ifndef EXAMPLE_H
#define EXAMPLE_H
#include <QMainwindow>
Qt_BEGIN_NAMESPACE
Namespace Ui{class example}
Qt_END_NAMESPACE
Class example: public QMainwindow{
Public:
Private:
Private slots:
}
#endif
```

In this same way all the header files are defined in Qt. Once all the declaration is made in

welcome.h file function definition is done in source file. All the necessary pushButton, labels, textEdit are first created in the welcome.ui file using Qt designer by using drag and drop. Once they tools are arranged in welcome.ui file then using the concept of signal and slot they function definition is done in welcome.cpp. welcome.ui consist of 2 buttons i.e login button and logout button, a label for login picture and 2 lineEdit for username and password.

Code snippet for signal and slot function:

In welcome.h:

Private slots:
void on_commandLinkButton_view_clicked();

In welcome.cpp:

void welcome:: on_commandLinkButton_view_clicked(){
/ Code Here*/*
}

Signal and slots are special concept used by Qt. A macro must be defined in header file i.e Q_OBJECT , it invokes the meta-object compiler (moc) which then convert the signal and slots in the required c++ code which is then compiled by g++ compiler. User must insert the username and password in the textEdit which is then checked by the program. If correct then the user is taken to desktop window. If not the required message is displayed in the status bar. User has the option to log out from the window using logout button. A message box is displayed for confirmation. In this login window an event handler is written for resize event.

Code snippet for event handler:

void resizeEvent(QResizeEvent initial window){}*

Here, Qwidget has void resizeEvent(QResizeEvent * ptr) function, so when an user defines this function they override this function in parent class. In Qt is a event handler is created then every action regarding that event will first go to that event handler rather then the program. In this program event handler is use to maintain the background image size with respect to the size of main window. After the login is completed in welcome window the user is taken to the desktop window when the password and username are correct. Desktop window consist of one header file, one ui file and one source file. These window consist of 3 main components to-do-list , game window and time.

To-do-list:
It a simple version of notepad where user can read, write and append to the file. It consist of one plainTextEdit and six commandLinkButton. Each commandLinkButton is associated with read, write, save, font , undo and redo function. Its main purpose it to allow the user to create the list of things that they want to do. User can read from the file, write to the file and clear the file completely. Each commandLinkButton is connected to a slot for doing their work.

Code snippet for font button:

```
void Desktop:: on_commandLinkButton_font_clicked(){
bool to_check;
QFontDialog::getFont(&to_check, this)
if(to_check){
ui->plainTextEdit_List->setFont(font)
}
else
        return;
}
```

This function checks if the font button is clicked or not. If clicked than it opens the font dialog box from Qt and assign the chosen font the plainTextEdit. Here plainTextEdit_List is an object of plainTextEdit class which calls its setFont function. Each button is assigned with a shortcut key. For example user can press ctrl+f to open the font dialog box.

Code snippet for short key:

```
QShortcut *for_list_font=new QShortcut(QKeySequence("ctrl+f"), this);
Connect(for_list_font,SIGNAL(activated()),this,SLOT(on_commandLinkButton_font_clicked()));
```

This code is first compiled by moc and then g++. In connect function when key sequence ctrl+f is pressed a activated() signal is generated and then function in slot is called. In this way code for font button is connected with the key sequence. In similar all other buttons are defined and a short key is assigned to them. Read, save and new button are all related to opening a text file in resource file or from the local device and reading or writing to or from the file.

Game Window :

This part of desktop window contains a UI window of its own. It has one .ui file, seven header files and seven source files. The main window where all the items are added is z0game.ui. This UI consist of one graphicsview and one push button. All the items related to game are added to scene in graphicsview.

Code snippet for adding to scene:

```
QGraphicsScene *scene=new  QGraphicsScene;
z2player *player=new z2player();
ui ->graphicsview ->adddscene(scene);
scene ->addItem(player);
```

This game has one player, player has to shoot another airplane coming from the opposite side. As the player shoots airplane his score increases which can be seen in on the screen, similarly if any enemy airplane goes out of scene then the health decreases. Its source file are:

1) z0game.cpp
   This is the main source file of this game. It has one class z0Game under which all the pointer related to player and scene are created. Background music and background image are also created in this class constructor. This class inherits form QMainWindow.

2) z2player.cpp
   z2player.cpp has one class z2Player, in which player is created and given the keyboard input. To take the keyboard input from the user a eventhandler had to be written.

   Code snippet of eventhandler for left move:

```
Void z2player::keypressevent(QkeyEvent *event){
        If(event key()==Qt::Key_Left){
                If(pos().x()>0){
                        setPos(x()-10,y());
}}}
```

Whenever a key is pressed in Qt it sends its signal to the focus widget, after receiving the signal that a keyevent has occurred Qt checks if there is any event handler in program. If yes then all the input are send to the eventhandler which then checks the code to see the key required by the player is pressed or not. If pressed then Qt executes the required code. In this way right move and left move are also defined in this class.

3) z3bullet.cpp
   The class of this source file is used to create a bullet whenever a space key is pressed. The constructor of z3Bullet class is responsible for the creation of a bullet. KeyPressEvent in z2player creates a new z3bullet pointer whenever a space key is pressed. Then the constructor in z3Player class creates a move and class for z3player::move() function which is responsible for its vertical upward motion. move() function also if the bullet produced has collided with the enemy or using typeid() and if collided then it calls for health function in z6health to decrease health.

   Code snippet for bullet collision:

```
        QList <QGraphicsItem *>collision_items= collidingItems()
for(int i=0,n=colliding_items.size();i<n ;i++){
if(typeid(*(colliding_items[i]))==typeid(z4Enemy)){
/*code*/
}}
```

   Music after a bullet is created is a produced in from this file. multimedia library of Qt is used for this.

4) z4enemy.cpp
From z0Game class spawn() function is called in a certain interval of time from z0Player class which then creates an enemy pointer and add it to the scene(). The constructor of z4Enemy creates enemy and calls the public slot move().In this slot the bullet is made to move in vertical downward motion. Here if the airplane goes out of scene then the pointer is deleted to free the memory.

5) z5score.cpp
This source file has one class z5Score which inherits from QGraphicsTextItem. QGraphicsTextItem is use to display text in graphics window. Here a private member score is created which contains the present score of player and z0Game is made it friend class so that z0Game can access the score data after the game is over.

Code snippet to display score:

```
Z5Score::z5Score(QGraphicsItem *parent): QGraphicsTextItem(parent){
        score=0;
        setPlainText(QString("Score  :") + QString::number(score));
        setDefaultTextColor(Qt::green);
        setFont(QFont("times",16));
}
```

6) z6health.cpp
This source is similar to z5score.cpp. It also inherits from QGraphicsTextItem and has a private member health. This class is used to display health on screen and update the health whenever enemy airplane goes out of screen. Its code definition is similar to that of z5Score class.

7) z7gameover.cpp
This is the final source file which contains one class z7GameOver. This class pointer is created in z4Enemy class whenever an enemy goes out of screen. Its constructor calls getHealth() function from z6Health class which checks if health is less one. If yes then it disables the player and makes a play again commandLinkButton visible. If the play again commandLinkButton is clicked then is again calls the z0Game class and starts the game again.

By calling appropriate function in appropriate time all the seven source files are integrated to make one game. All the source are has one of its own header file except z0game.cpp which has one .ui file linked to it.

Time:

In the center of the lower part of the Desktop Window time and date are displayed. For this QTime library of Qt is used. currentTime() from QTime is used here to get the current which is then converted into QString and added to label. For this to happen a private slot time_delay() is created.

Code of time_delay():

```
void Desktop::time_delay(){
QTime current_time = QTime::currentTime();
QString string_time = current_time.toString("hh:mm:ss");
ui ->label_time ->setText(string_time);
}
```

Two pushButton are added in the lower left corner of the Desktop window. These two pushButton are named as Logout and Restart and performs the same task as indicated by their name. When the Logout button is clicked then by using QApplication library the desktop window is destroyed and if Restart button is clicked then a pointer to the Login Window is called which open the Login Window.

CHAPTER SIX: RESULTS AND DISCUSSION

After building of our project as we run executable, the login interface opens where user has various options to choose whether he wants to login to desktop window or logout. Everytime for the right username and password entered, opens a desktop window while for wrong password or username for three consecutive attempt, leads to logout. In the desktop window user can see two options either to use notepad or play game. For features like new, save, font, undo, redo, read notepad is facilitated with respective buttons.
If user wants to play ,he should click on the play button. As the user clicks on it game window pops up where the user can play by pressing keyboard keys like left_key to move player's plane left, right_key to move player's plane right and space bar to shoot bullets towards enemy plane.
After gameover user can replay by clicking on replay button and he can quit using shortkey(ctrl+Q).

CHAPTER SEVEN: CONCLUSION

Thus, Desktop Environment – Universal(DE-U) was completed successfully. Using this user can create a list in to-do-list and change it as per their requirement similarly if bored they can play Aero shoot game. It is easy to use and anyone with knowledge of C++ programming language can change its source code and modify it as per their need. Though there were some problem in the initial days due to lack of C++ and Qt framework knowledge but the proper determination towards this project has not only made our knowledge of C++ programming more strong but also helped us to know what happens under the hood of any GUI program .

The completion of this project has made us realize us the power of inheritance, polymorphism, operator overloading etc in the programming world. Here, a simple creation of pushButton by creating an object of QPushButton class represents the power of object oriented programming language and how it can be linked to the objects of real world. This project has made us familiar with the important aspects of GUI world like event loop, active window focused window etc. In a nutshell, this project has successfully helped us achieve our objectives which we had set before starting this project.

REFERENCES

1. History of games(https://en.wikipedia.org/wiki/History_of_video_games)
2. Qt based games(https://wiki.qt.io/Qt_Based_Games)
3. History of gui(https://en.wikipedia.org/wiki/History_of_the_graphical_user_interface)
4. Qt for beginners (https://wiki.qt.io/Qt_for_Beginners)

APPENDIX (Source Code)

1. main.cpp

```
#include "welcome.h"
#include "desktop.h"
#include <QApplication>
welcome *w; //creating a global welcome pointer
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);//creating a QApplication object
     w =new welcome;
     w->showFullScreen(); //calling the welcome window in fullscreen mode
    return a.exec(); //starting the eventloop.
}
```

2. welcome.h

```
#ifndef WELCOME_H
#define WELCOME_H

#include <QMainWindow>
#include <desktop.h>
QT_BEGIN_NAMESPACE
namespace Ui { class welcome; }
QT_END_NAMESPACE

class welcome : public QMainWindow
{
    Q_OBJECT //macro declaration for meta object compiler

public:
    welcome(QWidget *parent = nullptr);
    ~welcome();
    void resizeEvent(QResizeEvent *initial_window);
    void loginpicture();
    void text_username();
    void text_password();
    Desktop *goDesktop;
private slots:  //slots compiled using MOC
    void on_pushButton_out_clicked();
```

```
    void on_commandLinkButton_clicked();

private:
    Ui::welcome *ui;
};
#endif // WELCOME_H
```

3. welcome.cpp

```
#include "welcome.h"
#include "ui_welcome.h"
#include <QPixmap>
#include<QLineEdit>
#include <QMessageBox>
welcome::welcome(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::welcome)
{
    ui->setupUi(this); //sending the mainwindow pointer to ui_welcome.h which create the ui
window widgets
    loginpicture();   //setting login picture,username and password
    text_username();
    text_password();
    ui->label_welcome->setStyleSheet("QLabel {color : rgb(4, 136, 172);}");
    ui->commandLinkButton->setStyleSheet("color : rgb(255, 255, 255);font-size: 27px;");
    ui->pushButton_out->setStyleSheet("color: rgb(255, 255, 255);background-
image:url(:/Images/Images/textedit.jpg);border: rgb(4,136,180);font-size:23px");
    ui->statusbar->setStyleSheet("font-size: 22px;");
}

welcome::~welcome()
{
    delete ui;
}

void welcome::resizeEvent(QResizeEvent *initial_window) //event handler for fullscreen mode.
{

    QPixmap background_color(":/Images/Images/background.jpg");
    background_color=background_color.scaled(this->size(),Qt::IgnoreAspectRatio);
    QPalette for_login;
    for_login.setBrush(QPalette::Background,background_color);
    this->setPalette(for_login);
}
```

```
void welcome::loginpicture()
{
   QPixmap login(":/Images/Images/login1_resize.jpg");
   ui->userpicture->setPixmap(login.scaled(ui->userpicture-
>size(),Qt::KeepAspectRatioByExpanding));

}

void welcome::text_username()
{
   QPalette p;
   p.setColor(QPalette::PlaceholderText,Qt::white);
   ui->lineEdit_username->setPlaceholderText("Username");
   ui->lineEdit_username->setStyleSheet("color: white;background-
image:url(:/Images/Images/textedit.jpg);border: rgb(4,136,180);");
   ui->lineEdit_username->setPalette(p);

}

void welcome::text_password()
{
   QPalette p;
   p.setColor(QPalette::PlaceholderText,Qt::white);
   ui->lineEdit_password->setPlaceholderText("Password");
   ui->lineEdit_password->setStyleSheet("color: white;background-
image:url(:/Images/Images/textedit.jpg);border: rgb(4,136,180);");
   ui->lineEdit_password->setPalette(p);

}


void welcome::on_pushButton_out_clicked() //creating a messagebox for login interface
{
QMessageBox::StandardButton option;
option= QMessageBox::question(this,"Confirm","Log out from this
device?",QMessageBox::Yes|QMessageBox::No);
if(option==QMessageBox::Yes){
   QApplication::quit();}
}

void welcome::on_commandLinkButton_clicked()//checking the username and password entered
by the user
{
   if(ui->lineEdit_username->text()=="project" && ui->lineEdit_password-
>text()=="HelloWorld") {
       this->hide();
```

```
        goDesktop=new Desktop;
        goDesktop->showFullScreen();
    }
    else if(ui->lineEdit_password->text()=="" && ui->lineEdit_password->text()==""){
        ui->statusbar->showMessage("Please enter your username and password.",4000);
        ui->statusbar->setStyleSheet("color:white;");
    }
    else{
        ui->statusbar->showMessage("Either username or password wrong.",4000);
        ui->statusbar->setStyleSheet("color:white;");
    }
}
```

4. desktop.h

```
#ifndef DESKTOP_H
#define DESKTOP_H

#include <QMainWindow>

namespace Ui {
class Desktop;
}

class Desktop : public QMainWindow
{
    Q_OBJECT //macro declaration for meta object compiler

public:
    void resizeEvent(QResizeEvent *DesktopResize);
    explicit Desktop(QWidget *parent = nullptr);
    ~Desktop();
    void to_do_list_button();

private slots: //slots compiled using MOC
    void time_delay();

    void on_commandLinkButton_view_clicked();

    void on_commandLinkButton_append_clicked();

    void on_commandLinkButton_new_clicked();

    void on_commandLinkButton_font_clicked();
```

```
    void on_commandLinkButton_undo_clicked();

    void on_commandLinkButton_redo_clicked();

    void on_pushButton_gameplay_clicked();

    void on_pushButton_desktop_log_out_clicked();

    void on_pushButton_restart_clicked();

private:
    Ui::Desktop *ui;
};

#endif // DESKTOP_H
```

5. desktop.cpp

```
#include "desktop.h"
#include "ui_desktop.h"
#include <QTimer>
#include<QTime>
#include <QMessageBox>
#include<QTextStream>
#include <QFontDialog>
#include <QShortcut>
#include"z0game.h"
#include <welcome.h>

z0Game *game; //creating a global game pointer
extern QMediaPlayer *music; //accessing the global pointer declared in another program
extern welcome *w;
Desktop::Desktop(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Desktop)
{
    ui->setupUi(this);
    ui->label_time->setStyleSheet("color : rgb(14,215,255);font-size: 32px;");//7,182,191
    ui->label_date->setStyleSheet("color: rgb(14,215,255);font-size : 32px;");//120,124,123
   ui->pushButton_gameplay->setStyleSheet("background-
image:url(:/Images/Images/Desk2.jpg);font-weight:bold;color:rgb(255,255,255);");
    QTimer *time=new QTimer();
    connect(time,SIGNAL(timeout()),this,SLOT(time_delay())); //connecting the inbuild function
timeout() with program function time_delay()
    time->start(500);  //accessing the time_delay() funciton in every 0.5seconds
```

```
    QDate current_date=QDate::currentDate();
    QString string_date=current_date.toString("dddd MMMM yyyy");
    ui->label_date->setText(string_date);//setting the string_date in label
    to_do_list_button();
    ui->pushButton_desktop_log_out->setStyleSheet("background-
image:url(:/Images/Images/inner_window.png);color:rgb(255,255,255);font-size:20px");
    ui->pushButton_restart->setStyleSheet("background-
image:url(:/Images/Images/inner_window.png);color:rgb(255,255,255);font-size:20px");

}
Desktop::~Desktop()
{
    delete ui;
}
void Desktop::resizeEvent(QResizeEvent *DesktopResize) //event handler for fullscreen mode.
{
    QPalette p;
    QPixmap desktopback(":/Images/Images/inner_window.png");
    desktopback=desktopback.scaled(this->size(),Qt::IgnoreAspectRatio);
    p.setBrush(QPalette::Background,desktopback);
    this->setPalette(p);
}


void Desktop::to_do_list_button()
{
    /*here we are setting the stylesheet for commandlink button and creating  shortcut
    key for all the buttons. each activated() function is calling its respective program function
    when the shortkey is pressed.*/
    ui->commandLinkButton_append->setStyleSheet("color: rgb(255,255,255);");
    ui->commandLinkButton_font->setStyleSheet("color: rgb(255,255,255);");
    ui->commandLinkButton_new->setStyleSheet("color: rgb(255,255,255);");
    ui->commandLinkButton_view->setStyleSheet("color: rgb(255,255,255);");
    ui->commandLinkButton_undo->setStyleSheet("color: rgb(255,255,255);");
    ui->commandLinkButton_redo->setStyleSheet("color: rgb(255,255,255);");
    QPalette plaintextedit;
    plaintextedit.setColor(QPalette::Text,Qt::white);
    ui->plainTextEdit_list->setPalette(plaintextedit);
    QShortcut *for_list_view=new QShortcut(QKeySequence("ctrl+w"),this);

connect(for_list_view,SIGNAL(activated()),this,SLOT(on_commandLinkButton_view_clicked()))
;
    QShortcut *for_list_append=new QShortcut(QKeySequence("ctrl+s"),this);

connect(for_list_append,SIGNAL(activated()),this,SLOT(on_commandLinkButton_append_click
ed()));
```

```
    QShortcut *for_list_new=new QShortcut(QKeySequence("ctrl+n"),this);

connect(for_list_new,SIGNAL(activated()),this,SLOT(on_commandLinkButton_new_clicked()));
    QShortcut *for_list_font=new QShortcut(QKeySequence("ctrl+f"),this);

connect(for_list_font,SIGNAL(activated()),this,SLOT(on_commandLinkButton_font_clicked()));
    QShortcut *for_list_undo=new QShortcut(QKeySequence("ctrl+u"),this);

connect(for_list_undo,SIGNAL(activated()),this,SLOT(on_commandLinkButton_undo_clicked())
);
    QShortcut *for_list_redo=new QShortcut(QKeySequence("ctrl+r"),this);

connect(for_list_redo,SIGNAL(activated()),this,SLOT(on_commandLinkButton_redo_clicked()))
;

}
void Desktop::time_delay()
{
 QTime current_time=QTime::currentTime();
 QString string_time=current_time.toString("hh:mm:ss");
 ui->label_time->setText(string_time);
}

void Desktop::on_commandLinkButton_view_clicked()
{
 /*creating a slot for view button in to_do_list.*/
QFile for_view("C:/Users/anju/OneDrive/Desktop/DE-U/DE-U/Images/to_do_list.txt");
if(!for_view.open(QFile::ReadOnly|QFile::Text)){
    QMessageBox::information(this,"error","to_do_list.txt file does not exist.");
}
QTextStream to_read(&for_view);
QString text=to_read.readAll();//conversion from QtextStream to QString.
ui->plainTextEdit_list->setPlainText(text);
for_view.close();
}

void Desktop::on_commandLinkButton_append_clicked()
{
QFile append("C:/Users/anju/OneDrive/Desktop/DE-U/DE-U/Images/to_do_list.txt");
if(!append.open(QFile::WriteOnly| QFile::Text)){
QMessageBox::information(this,"error", "to_do_list.txt file does not exist.");
}
QTextStream to_append(&append);
QString text= ui->plainTextEdit_list->toPlainText();//conversion from plaintext to QString.
to_append<<text; //writing in file
to_append.flush(); //clearing the buffer
```

```
append.close();
}
void Desktop::on_commandLinkButton_new_clicked()
{
ui->plainTextEdit_list->clear();
QFile for_new("C:/Users/anju/OneDrive/Desktop/DE-U/DE-U/Images/to_do_list.txt");
for_new.resize(0); // clearing the content of text file.
if(!for_new.open(QFile::Text|QFile::WriteOnly| QFile::Truncate)){
   QMessageBox::information(this,"error", "to_do_list.txt file does not exist.");
}
QTextStream renew(&for_new);
QString text=ui->plainTextEdit_list->toPlainText();
renew<<text;
renew.flush(); //clearing the buffer
for_new.close();//closing the file
}


void Desktop::on_commandLinkButton_font_clicked()
{
bool to_check;
QFont font=QFontDialog::getFont(&to_check, this);// calling the getfont function from
QfontDialog namespace
if(to_check){
   ui->plainTextEdit_list->setFont(font);//setting the font in plaintextedit
}
else
   return;
}

void Desktop::on_commandLinkButton_undo_clicked()
{
   ui->plainTextEdit_list->undo();//calling the undo function using plainTextEdit_list object
}

void Desktop::on_commandLinkButton_redo_clicked()
{
   ui->plainTextEdit_list->redo();//calling the redo function using plainTextEdit_list object
}


void Desktop::on_pushButton_gameplay_clicked()
{
   /*on play button clicked creating a game pointer to call another ui window*/

   game=new z0Game;
```

```
    game->move(50,30); //setting its poistion in window
    game->setFixedSize(1800,950);
    game->show();

}

void Desktop::on_pushButton_desktop_log_out_clicked()
{
    //creating a messagebox for confirmation for logout button.
    QMessageBox::StandardButton confirm;
    confirm=QMessageBox::question(this,"Confirm","Log out from this
device",QMessageBox::Yes|QMessageBox::No);
    if(QMessageBox::Yes==confirm){
    QApplication::quit(); //exit the whole window
}}

void Desktop::on_pushButton_restart_clicked()
{
    //creating a messagebox for confirmation for restart button.

    QMessageBox::StandardButton restart;

restart=QMessageBox::question(this,"Confirm","Restart?",QMessageBox::Yes|QMessageBox::
No);
    if(restart==QMessageBox::Yes){
    this->hide(); //hiding the current window.
    w->showFullScreen();} //opening the login window in fullscreenmode
}
```

6. z0game.h

```
#ifndef Z0GAME_H
#define Z0GAME_H

#include <QMainWindow>
#include<QGraphicsScene>
#include<QTimer>
#include "z2player.h"
#include "z5score.h"
#include"z6health.h"
#include "z7gameover.h"
#include <QTimer>
#include <QCloseEvent>
namespace Ui {
class z0Game;
}
```

```
class z0Game : public QMainWindow
{
    Q_OBJECT

public:
    explicit z0Game(QWidget *parent = nullptr);
    ~z0Game();
    QGraphicsScene *scene;
    z2Player *player;
    z5Score *score;
    z6Health *health;
    QTimer *timer;
    void closeEvent(QCloseEvent *close);
void make_replay_visible();

public slots:
    void deletecalled();
    void on_replay_clicked();

private:
    Ui::z0Game *ui;
};

#endif // Z0GAME_H
```

7. z0game.cpp

```
#include "z0game.h"
#include "ui_z0game.h"
#include <QMediaPlayer>
#include <QImage>
#include<QBrush>
#include<QShortcut>
z7GameOver *gameover;
QMediaPlayer *music;

z0Game::z0Game(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::z0Game)
{
    ui->setupUi(this);
    this->setWindowTitle("Game");
    setWindowIcon(QIcon(":/Game/Game/joystick.png"));
    ui->replay->setStyleSheet("color: rgb(0,0,0);font-size: 27px;font-weight:bold;");
```

```
    ui->statusbar->setStyleSheet("color: rgb(0,0,0);font-size: 15px;font-weight:bold;");
    ui->statusbar->showMessage("Press ctrl+t to exit the game.",5000);
    QShortcut *f=new QShortcut(QKeySequence("ctrl+q"),this);//shortcut key for window
termination
    connect(f,SIGNAL(activated()),this,SLOT(deletecalled()));//calling deletecalled when inbuild
fucntion activated is called

    scene = new QGraphicsScene();
    scene->setSceneRect(0,0,1800,950); //setting a scene window
    ui->graphicsView-
>setBackgroundBrush(QBrush(QImage(":/Game/Game/background1.jpg")));
    ui->replay->setVisible(false);

    ui->graphicsView->setScene(scene);//setting the scene in grahicsview
    ui->graphicsView->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    ui->graphicsView->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    //setFixedSize(800,600);

    player = new z2Player(); //creating a player pointer
    player->setPixmap(QPixmap(":/Game/Game/player1.png"));
    player->setPos(1800/2,800); //setting the position of player in window

    player->setFlag(QGraphicsItem::ItemIsFocusable); //setting window focus in player object
pointer
    player->setFocus();

    score = new z5Score();
    scene->addItem(score); //adding score to scene
    health = new z6Health();
    health->setPos(health->x(),health->y()+30);
    scene->addItem(health);
    scene->addItem(player);

    timer = new QTimer();
    QObject::connect(timer, SIGNAL(timeout()), player, SLOT(spawn()));
    timer->start(2000);

    music = new QMediaPlayer;
    music->setMedia(QUrl("qrc:/Game/Game/gameMusic.mp3"));/*playing the music*/
    music->play();
}

z0Game::~z0Game()
{
    delete ui;//deleting the ui when the z0Game class is destroyed
}
```

```
void z0Game::closeEvent(QCloseEvent *close)//event handler for window close
{
   delete music;
}

void z0Game::make_replay_visible()
{
   ui->replay->setVisible(true);
}




void z0Game::deletecalled()
{
   delete this;

}


void z0Game::on_replay_clicked()
{


   scene->removeItem(gameover);

   score->score = 0;
   score->scoreupdate();
   health->health = 3;
   health->healthupdate();
   player->setEnabled(true);
   music->play();
   ui->replay->setVisible(false);
}
```

8. z2player.h

```
#ifndef Z2PLAYER_H
#define Z2PLAYER_H

#include <QGraphicsPixmapItem>
#include <QObject>
#include "z3bullet.h"
```

```
#include "z4enemy.h"
#include <QMediaPlayer>

class z2Player:public QObject ,public QGraphicsPixmapItem{
   Q_OBJECT


QMediaPlayer *bulletsound = new QMediaPlayer;
public:
z2Player();
   void keyPressEvent(QKeyEvent * event);

public slots:
   void spawn();


};

#endif // Z2PLAYER_H
```

9. z2player.cpp

```
#include "z2player.h"
#include<QKeyEvent>
#include <QGraphicsScene>
#include "z3bullet.h"


z2Player::z2Player()
{
   bulletsound->setMedia(QUrl("qrc:/Game/Game/GunFiring.mp3"));
}

void z2Player::keyPressEvent(QKeyEvent *event)
{
  /*event handler for keypressevent*/
if(event->key()==Qt::Key_Left)
{
   if(pos().x()>0){
   setPos(x()-10,y());} //move towards left
}
else if(event->key()==Qt::Key_Right)
{
   if(pos().x()+10<1800){
   setPos(x()+10,y());}//move towards right
```

```
}
else if(event->key()==Qt::Key_Space)
{
   z3Bullet *bullet = new z3Bullet();
   bullet->setPos(x()+50,y());//setting the bullet positioin
   scene()->addItem(bullet);


   if(bulletsound->state() == QMediaPlayer::PlayingState)
   {
      bulletsound->setPosition(0); /*if song is playing play it from the begining*/
   }
   else if(bulletsound->state() == QMediaPlayer::StoppedState)
   {
      bulletsound->play();
   }
}

}
void z2Player::spawn() //creating a enemy and adding it to scene
{
 z4Enemy *enemy = new z4Enemy();
 scene()->addItem(enemy);
}
```

10. z3bullet.h

```
#ifndef Z3BULLET_H
#define Z3BULLET_H
#include "QGraphicsPixmapItem"
#include <QObject>
class z3Bullet: public QObject,public QGraphicsPixmapItem
{
   Q_OBJECT
public:
   z3Bullet();
public slots:
   void move();
};

#endif // Z3BULLET_H
```

11. z3bullet.cpp

```
#include "z3bullet.h"
#include<QTimer>
```

```
#include<QGraphicsScene>
#include<QList>
#include "z4enemy.h"
#include "z0game.h"
#include <QMediaPlayer>
extern z0Game *game;

z3Bullet::z3Bullet()
{
setPos(x(),y());
setPixmap(QPixmap(":/Game/Game/bullet.png"));
QTimer *timer = new QTimer(this);
connect(timer, SIGNAL(timeout()),this, SLOT(move()));
timer->start(50);/*call move function in every 50ms*/

}

void z3Bullet::move()
{
   /*create a list of graphicsitem*/
 QList <QGraphicsItem *> colliding_items = collidingItems();
 for(int i=0, n=colliding_items.size(); i<n; i++)
 { /*if colliding item with bullet is of type z4enemy condition is true*/
    if(typeid(*(colliding_items[i]))==typeid (z4Enemy))
    {
        QMediaPlayer *collision_song=new QMediaPlayer;
        collision_song->setMedia(QUrl("qrc:/Game/Game/collision.mp3"));
        collision_song->play();
        game->score->increase();
        scene()->removeItem(colliding_items[i]);
        scene()->removeItem(this);

        delete colliding_items[i];//delete the item from list
        delete this; //delete enemy.
        return;

    }
 }
 setPos(x(),y()-10); //move vertically upward
 if (pos().y() + 50 < 0){
    scene()->removeItem(this);
    delete this;//if bullet goes out of screen delete it from memory
 }
}
```

12. z4enemy.h

```
#ifndef Z4ENEMY_H
#define Z4ENEMY_H
#include<QGraphicsPixmapItem>
#include <QObject>
#include "z7gameOver.h "
class z4Enemy:public QObject,public QGraphicsPixmapItem
{
   Q_OBJECT

public:
   z4Enemy();

public slots:
   void move();
};

#endif // Z4ENEMY_H
```

13. z4enemy.cpp

```
#include "z4enemy.h"
#include<stdlib.h>
#include<QGraphicsScene>
#include<QTimer>
#include"z0game.h"
extern z0Game *game;
extern z7GameOver *gameover;

z4Enemy::z4Enemy()
{
int random_num = rand()%1700; //creating a enemy inside window only
setPos(random_num,-150);

setPixmap(QPixmap(":/Game/Game/airplane1.png"));
setTransformOriginPoint(45,45); //changing the enemy's origin point and rotating it by 180
degree
setRotation(180);
QTimer *timer= new QTimer(this);
connect(timer,SIGNAL(timeout()),this,SLOT(move()));
timer->start(70);
}

void z4Enemy::move()
```

```
{
   setPos(x(),y()+5);
   if(game->health->getHealth() <=0)
   {

      if(pos().y()>=-130)
      {
         delete this; //if health is <0  stopping the production of enemy
      }

   }
   if (pos().y()>(950-45))
   {
      game->health->Decrease();
       gameover = new z7GameOver; //if health is <0 display gameover
       scene()->addItem(gameover);
      scene()->removeItem(this);
      delete this;
   }
}
```

14. z5score.h

```
#ifndef Z5SCORE_H
#define Z5SCORE_H
#include<QGraphicsTextItem>


class z5Score: public QGraphicsTextItem
{
public:
   z5Score(QGraphicsItem *parent = 0);
    void increase();
    void scoreupdate();
private:
    int score;
    friend class z0Game;
};

#endif // Z5SCORE_H
```

15. z5score.cpp

```
#include "z5score.h"
```

```
#include<QGraphicsTextItem>
#include<QFont>
z5Score::z5Score(QGraphicsItem *parent):QGraphicsTextItem(parent)
{
score = 0;
setPlainText(QString("Score: ") + QString::number(score)); //display score on screen
setDefaultTextColor(Qt::green);
setFont(QFont("times",16));
}

void z5Score::increase()
{
   score++;
   setPlainText(QString("Score: ")+ QString::number(score));
}

void z5Score::scoreupdate()
{
   setPlainText(QString("Score: ")+ QString::number(score));
}
```

16. z6health.h

```
#ifndef Z6HEALTH_H
#define Z6HEALTH_H
#include <QGraphicsTextItem>

class z6Health: public QGraphicsTextItem
{
public:
   z6Health(QGraphicsItem *parent=0);
   void Decrease();
   int getHealth();
   void healthupdate();
private:
   int health;
   friend class z0Game;
};

#endif // Z3HEALTH_H
```

17. z6health.cpp

```
#include "z6health.h"
#include"QFont"
z6Health::z6Health(QGraphicsItem *parent):QGraphicsTextItem(parent)
{
health = 3;
setPlainText(QString("Health: ")+QString::number(health)); //dispaly health on screen
setDefaultTextColor(Qt::red);
setFont(QFont("times", 16));
}

void z6Health::Decrease()
{
   health--;
   setPlainText(QString("Health: ")+QString::number(health));
}

int z6Health::getHealth()
{
   return health;
}

void z6Health::healthupdate()
{
   setPlainText(QString("Health: ")+QString::number(health));
}
```

18. z7gameover.h

```
#ifndef Z7GAMEOVER_H
#define Z7GAMEOVER_H
#include<QGraphicsTextItem>

class z7GameOver: public QGraphicsTextItem
{
public:
   z7GameOver(QGraphicsItem *parent=0);

};

#endif // Z7GAMEOVER_H
```

19. z7gameover.cpp

```
#include "z7gameover.h"
#include "z0game.h"
extern z0Game *game;
extern QMediaPlayer *music;
z7GameOver::z7GameOver(QGraphicsItem *parent):QGraphicsTextItem(parent)
{
    int k=game->health->getHealth();
    if(k==0){ //checking for health

        setPlainText("Game Over.");
        setFont(QFont("times",42));
        setPos(900,300);
        setDefaultTextColor(Qt::darkGreen);
        game->player->setEnabled(false); //stoppping the player from taking keyboard input
        game->make_replay_visible();//make replay button visible
        music->stop();


    }

}
```