IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: B.L.D. Jayasiri	
Student Reference Number:10707228	

Module Code: SOFT336SL	Module Name: SOFT336SL Cross platform Application Development in C++
Coursework Title: QAliens-Killer mini game	
Deadline Date:18/01/2022	Member of staff responsible for coursework: Mr. Marius Varga
Programme: BSc(Hons)Software Engineering	

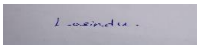
Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed : 

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

Overall mark _____% Assessors Initials _____ Date_____

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

CONTENT

QAliens-Killer

1)Introduction

QAliens-Killer is a simple shooting mini game. It is a fundamentally simple cross-platform Qt application. At one time, only one player can play this game. QAliens-Killer mini game was designed to run Windows10, Mac Os and Linux-ubuntu operating systems. This shooting game was totally developed using Qt libraries and C++ language.

In this game, player can score points by shooting aliens and at the begging of the game, player's health is equal to 3. When an alien safely moved from the bottom of the screen, the player's health decreases one each. You can find out more about this QAliens-Killer game under below topics.

2)Scope

Followings are the basic requirements of this game,

- Allow user to fire bullets using space bar of the computer.
- Allow user to hear a firing sound when a bullet release.
- Allow user to hear a background music within the time of playing the game.
- Allow user to move right side and left side of the game screen using left and right arrow keys for easily aiming to aliens.
- Allow user to end the game any time.
- Allow user to see their current score and current health level while playing the game.

The initial scope of what the project needs to fulfil is of completion. I hope to add some more features and functionalities to this game in future. it was discussed detailed under "Future Implements".

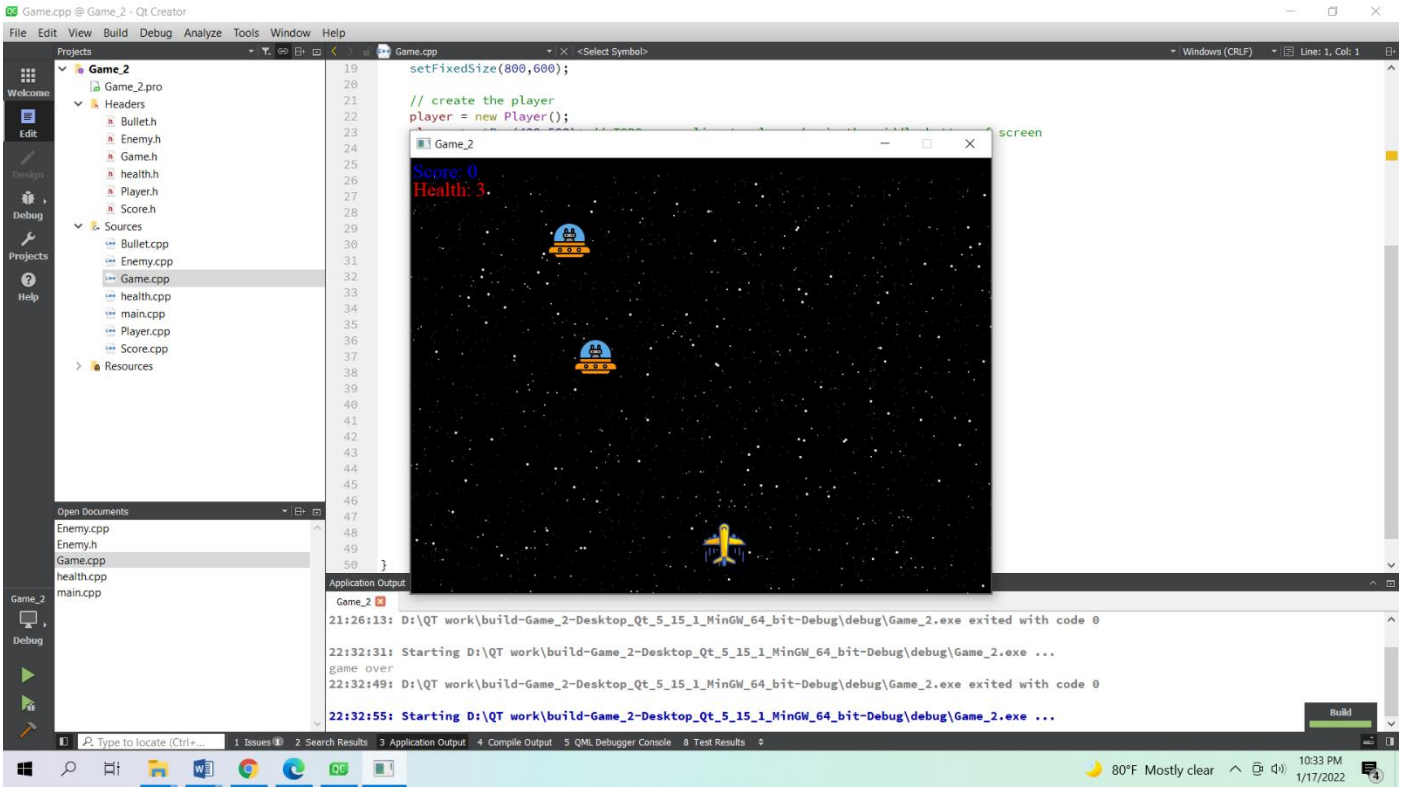
3)Installation

QAliens-Killer can be built and executed without needing to install external dependencies. It relies only on core Qt APIs. Therefore, the application can be run simply by extracting the source directory and loading the project (.pro) file before running the build/execute in Qt creator.

4) Supported Platforms

Currently, QAliens-Killer game has been tested and verified on windows 10, Mac OS and Linux-Ubuntu operating systems. Below are some screenshots to prove the above.

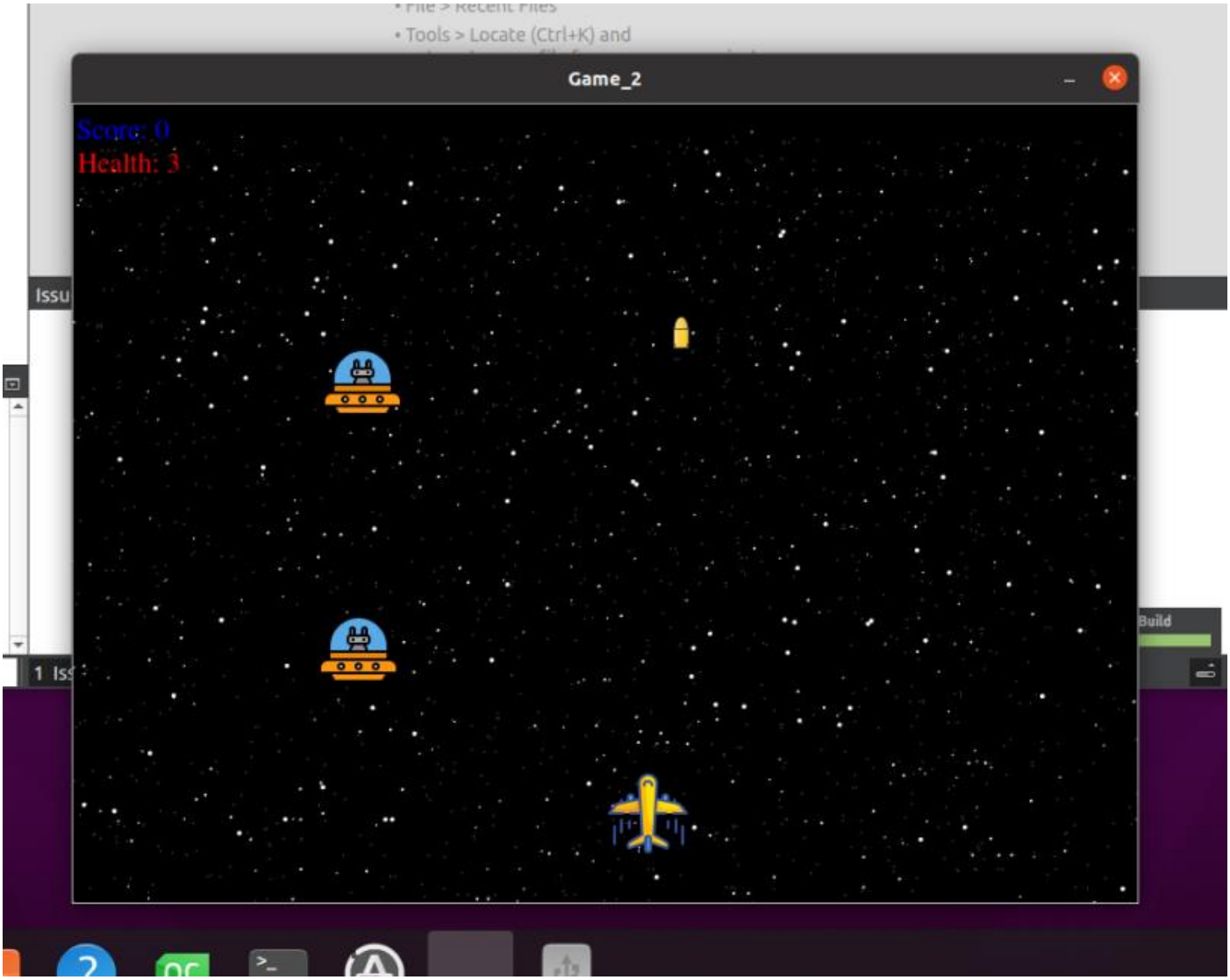
4.1)Windows 10


Game.cpp @ Game_2 - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
Projects
Game_2
Game_2.pro
Headers
Bullet.h
Enemy.h
Game.h
health.h
Player.h
Score.h
Sources
Bullet.cpp
Enemy.cpp
Game.cpp
health.cpp
main.cpp
Player.cpp
Score.cpp
Resources
setFixedSize(800,600);
// create the player
player = new Player();
Game_2
Score: 0
Health: 3
Open Documents
Enemy.cpp
Enemy.h
Game.cpp
health.cpp
main.cpp
Application Output
21:26:13: D:\QT work\build-Game_2-Desktop_Qt_5_15_1_MinGW_64_bit-Debug\debug\Game_2.exe exited with code 0
22:32:31: Starting D:\QT work\build-Game_2-Desktop_Qt_5_15_1_MinGW_64_bit-Debug\debug\Game_2.exe ...
game over
22:32:49: D:\QT work\build-Game_2-Desktop_Qt_5_15_1_MinGW_64_bit-Debug\debug\Game_2.exe exited with code 0
22:32:55: Starting D:\QT work\build-Game_2-Desktop_Qt_5_15_1_MinGW_64_bit-Debug\debug\Game_2.exe ...
80°F Mostly clear
10:33 PM
1/17/2022

4.2)Mac OS


mainwindow.ui - Qt Creator
Score: 0
Health: 3
p Emscripten SDK for WebAssembly? To do it later, select Options > Devices > WebAssembly.
Setup Emscripten SDK

4.3)Linux-Ubuntu



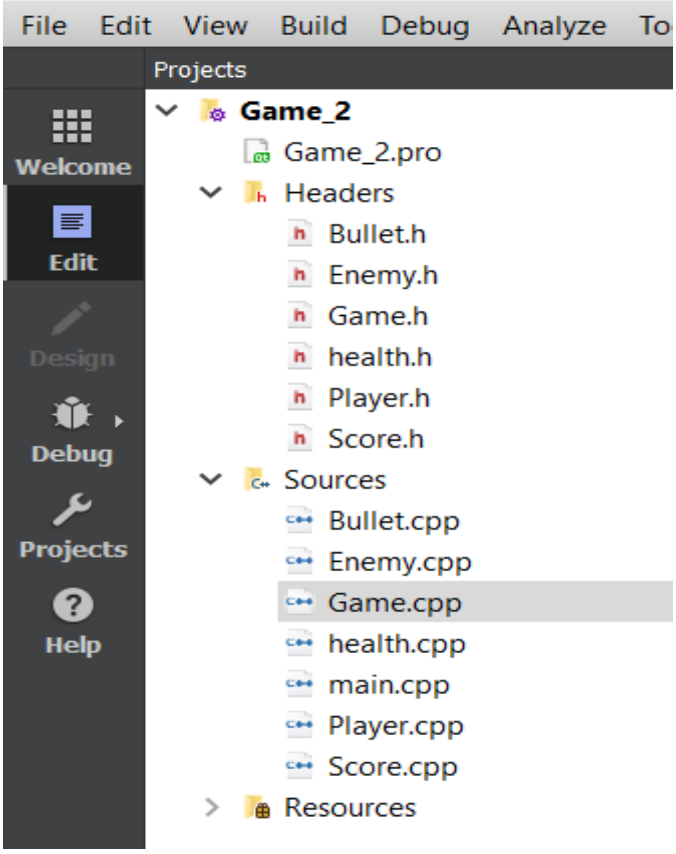
5)System Interaction

In this shooting game, there is only one window displaying throughout the game. It is the main Window/GUI of the QAliens-Killer game. The player's movements, enemies' movements and bullets' movements are displayed in this window. Current score and the current health level of the player is displayed top left corner of the window. The window has a fixed size of (800,600).

QGraphicsView, QWidget, **QGraphicsScene** libraries are used to developed this game.
QMediaPlayer library is used to play the background music and the firing sound of a bullet.

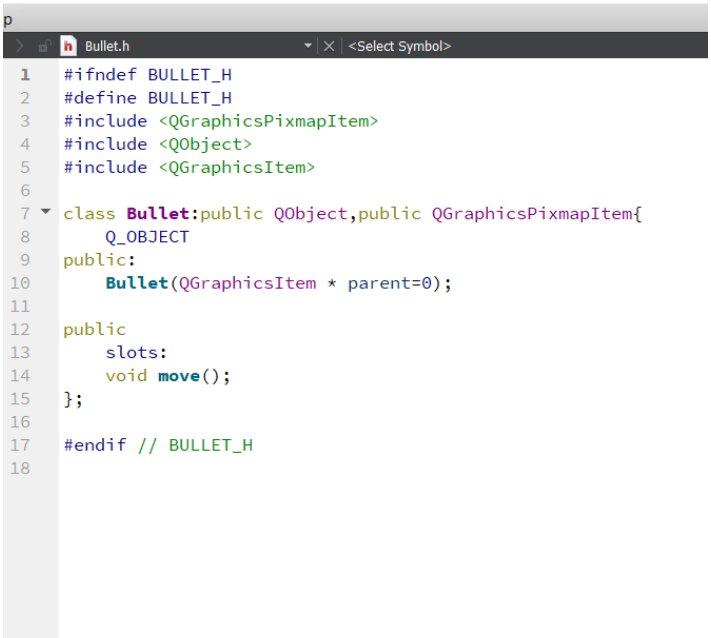
6)Code description

6.1)Project file



6.2)Headers

- Bullet.h




```
#ifndef BULLET_H
#define BULLET_H
#include <QGraphicsPixmapItem>
#include <QObject>
#include <QGraphicsItem>

class Bullet:public QObject,public QGraphicsPixmapItem{
    Q_OBJECT
public:
    Bullet(QGraphicsItem * parent=0);

public
    slots:
    void move();
};

#endif // BULLET_H
```

- Enemy.h

Enemy.h

```
#ifndef ENEMY_H
#define ENEMY_H

#include <QObject>
#include <QGraphicsPixmapItem>

class Enemy:public QObject,public QGraphicsPixmapItem{
    Q_OBJECT
public:
    Enemy(QGraphicsItem * parent=0);

public
    slots:
    void move();
};



#endif // ENEMY_H
```

- Game.h

Game.h

```
#ifndef GAME_H
#define GAME_H

#include <QGraphicsView>
#include <QWidget>
#include <QGraphicsScene>
#include "Player.h"
#include "Score.h"
#include "health.h"


class Game: public QGraphicsView{
public:
    Game(QWidget * parent=0);

    QGraphicsScene * scene;
    Player* player;
    Score * score;
    Health * health;
};



#endif // GAME_H
```

- Health.h

health.h

```
#ifndef HEALTH_H
#define HEALTH_H

#include <QGraphicsTextItem>

class Health: public QGraphicsTextItem{

public:
    Health(QGraphicsItem * parent=0);
    void decrease();
    int getHealth();
private:
    int health;

};

#endif // HEALTH_H
```

- Player.h

Player.h <Select Symbol>

```
#ifndef PLAYER_H
#define PLAYER_H
#include <QGraphicsPixmapItem>
#include <QObject>
#include <QGraphicsItem>
#include <QMediaPlayer>
#include <QPixmap>

class Player:public QObject, public QGraphicsPixmapItem{
    Q_OBJECT
public:
    Player(QGraphicsItem * parent=0);
    void keyPressEvent(QKeyEvent * event);
public slots:
    void spawn();
private:
    QMediaPlayer * bulletsound;
};
#endif // PLAYER_H
```

- Score.h

Score.h <Select Symbol>

```
#ifndef SCORE_H
#define SCORE_H

#include <QGraphicsTextItem>

class Score: public QGraphicsTextItem{

public:
    Score(QGraphicsItem * parent=0);
    void increase();
    int getScore();
private:
    int score;

};

#endif // SCORE_H
```

6.3)Sources

- Bullet.cpp

Bullet.cpp Bullet::move() -> void

```cpp
#include "Bullet.h"
#include <QTimer>
#include <QGraphicsScene>
#include <QList>
#include "Enemy.h"
#include "Game.h"

extern Game * game; // there is an external global object called game

Bullet::Bullet(QGraphicsItem *parent): QObject(), QGraphicsPixmapItem(parent)
{
    // drew bullet
    setPixmap(QPixmap(":/images/Resources/Images/bullet2.png"));

    //connect
    QTimer * timer = new QTimer();
    connect(timer,SIGNAL(timeout()),this,SLOT(move()));

    timer->start(50);

}

void Bullet::move()
{

    //if bullet collides with enemy, destroy both
    QList<QGraphicsItem *>colliding_items =collidingItems();
    for(int i=0, n= colliding_items.size(); i <n; ++i){
        if(typeid(*(colliding_items[i])) == typeid (Enemy)){    expression with
            //increase the score
            game->score->increase();

            //remove them both
            scene()->removeItem(colliding_items[i]);
            scene()->removeItem(this);
            //delete them both
            delete colliding_items[i];
            delete this;
            return;
        }
    }

    //move the bullet up
    setPos(x(),y()-10);

    if(pos(),y() <0){

        scene()->removeItem(this);
        delete this;


    }
}
```

- Enemy.cpp

Enemy.cpp | <Select Symbol>

```cpp
#include "Enemy.h"
#include <QTimer>
#include <QGraphicsScene>
#include <QList>
#include <stdlib.h>
#include "Game.h"

extern Game * game;

Enemy::Enemy(QGraphicsItem *parent): QObject(), QGraphicsPixmapItem(parent)
{
    //set Random Position
    int random_number =rand() % 700;
    setPos(random_number,0);
    // drew the enmey
     setPixmap(QPixmap(":/images/Resources/Images/Enemy.png"));

    //connect
    QTimer * timer = new QTimer();
    connect(timer,SIGNAL(timeout()),this,SLOT(move()));

    timer->start(50);

}

void Enemy::move()
{
    //move enmy down
    setPos(x(),y()+5);
    //destroy enemy when it goes out of the screen
    if(pos(),y()  >600){
```

```cpp
        //secrease the health
        game->health->decrease();


        scene()->removeItem(this);
        delete this;


    }
}

```

- Game.cpp

Game.cpp | Game::Game(QWidget *) -> void

```cpp
#include "Game.h"
#include <QTimer>
#include <QGraphicsTextItem>
#include <QFont>
#include "Enemy.h"
#include <QMediaPlayer>
#include <QImage>

Game::Game(QWidget *parent){
    // create the scene
    scene = new QGraphicsScene();
    scene->setSceneRect(0,0,800,600);
    setBackgroundBrush(QBrush(QImage(":/images/Resources/Images/background.jpg")));

    // remove scrollbars
    setScene(scene);
    setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    setFixedSize(800,600);

    // create the player
    player = new Player();
    player->setPos(400,500); // TODO generalize to always be in the middle bottom of screen
    // make the player focusable
    player->setFlag(QGraphicsItem::ItemIsFocusable);
    player->setFocus();
    // add the player to the scene
    scene->addItem(player);

    // create the score/health
        score = new Score();
        scene->addItem(score);
```

```
        health =new Health();
        health->setPos(health->x(),health->y()+25);
        scene->addItem(health);

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

    //play background music

    QMediaPlayer *music=new QMediaPlayer();
    music->setMedia(QUrl("qrc:/Sound/Resources/BackgroundMusic.mp3"));
    music->play();

    show();
}

```

- health.cpp

health.cpp* | Health::decrease() -> void

```
#include "health.h"
#include <QFont>
#include <QMessageBox>
#include <QDebug>
#include "Game.h"
Health::Health(QGraphicsItem *parent):QGraphicsTextItem(parent)
{
    //intialize the health to 0
    health=3;

    //draw the text
    setPlainText(QString("Health: " + QString::number(health)));//Health=3
    setDefaultTextColor(Qt::red);
    setFont(QFont("times",16));
}

void Health::decrease()
{

    health--;
    setPlainText(QString("Health: " + QString::number(health)));// Health

    if(health==0){
       qDebug() <<"game over";
    }
}

int Health::getHealth()
{
    return health;
}
```

- main.cpp

main.cpp | <Select Symbol>

```
#include <QApplication>
#include "Game.h"


Game * game;

int main(int argc, char *argv[]){
    QApplication a(argc, argv);

    game = new Game();
    game->show();

    return a.exec();
}

```

- Player.cpp

Player.cpp <Select Symbol>

```cpp
#include "Player.h"
#include <QGraphicsScene>
#include <QKeyEvent>
#include "Bullet.h"
#include "Enemy.h"


Player::Player(QGraphicsItem *parent): QGraphicsPixmapItem(parent){
    // set bullet sound
    bulletsound = new QMediaPlayer();
    bulletsound->setMedia(QUrl("qrc:/Sound/Resources/Gun Shot.mp3"));

    // set graphic
    setPixmap(QPixmap(":/images/Resources/Images/plane.png"));
}

void Player::keyPressEvent(QKeyEvent *event){
    // move the player left and right
    if (event->key() == Qt::Key_Left){
        if (pos().x() > 0)
        setPos(x()-10,y());
    }
    else if (event->key() == Qt::Key_Right){
        if (pos().x() + 100 < 800)
        setPos(x()+10,y());
    }
    // shoot with the spacebar
    else if (event->key() == Qt::Key_Space){
        // create a bullet
        Bullet * bullet = new Bullet();
        bullet->setPos(x()+45,y());
        scene()->addItem(bullet);
```

```cpp
        // play bulletsound
        if (bulletsound->state() == QMediaPlayer::PlayingState){
            bulletsound->setPosition(0);
        }
        else if (bulletsound->state() == QMediaPlayer::StoppedState){
            bulletsound->play();
        }

    }
}

void Player::spawn(){
    // create an enemy
    Enemy * enemy = new Enemy();
    scene()->addItem(enemy);
}

```

- Score.cpp

Score.cpp <Select Symbol>

```cpp
#include "Score.h"
#include <QFont>

Score::Score(QGraphicsItem *parent):QGraphicsTextItem(parent)
{
    //intialize the score to 0
    score=0;

    //draw the text
    setPlainText(QString("Score: " + QString::number(score)));
    setDefaultTextColor(Qt::blue);
    setFont(QFont("times",16));
}

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

}

int Score::getScore()
{
    return score;
}
```

7)Bugs

A small number of known bugs exist in the system as it stands. They are listed below:

- After the player's health level is 0, game is not over.it is still going forward.
- Sometimes player's health level get a minus value. (e.g.: after the zero, health level is decreasing like -1,-2,-3 etc.).

8)Future Implements

- To make a main menu for this game.
- To be able to improve this game with game over moment.
- To add a pause button and a start button to the game.
- To add a facility to look the highest score of the player.
- To improve the code, so that the enemy can shoot.