Computer Science 2 Project Report : GTA Milestone 1

Omar Almekhlafy

Khaled Hamza

Abdelrahman Alhamahmy

Table of Content

Introduction

This game includes: a player (Franklin), two enemies, powerups, bullets, a game map, and some rules and objectives. They are all implemented using OPP on QT. The game goes as follows:

1. Franklin starts near his house on the game board
2. Franklin moves around trying to collect bullets to kill enemies
3. The two enemies move randomly across the board trying to get Franklin to die
4. Franklin kills an enemy by collecting two bullets
5. Franklin dies when his health (3 hearts) are taken away by enemies touching him
6. A powerup allows Franklin to kill an enemy using 1 bullet instead of 2
7. A powerup lasts for 5 seconds for Franklin
8. Winning is defined as the death of all enemies
9. Losing is defined as Franklin's loss of all his health

The board map is designed to be a maze, and there are obstacles which neither Franklin nor the enemies can bypass. The following sections of this report will show how all of these functions are implemented.

Explanation of Classes and Functions

Enemies

```
#ifndef ENEMY_H
#define ENEMY_H
#include <QGraphicsPixmapItem>
#include <QGraphicsScene>
#include <QKeyEvent>
#include <QList>
#include <QImage>
#include <QMediaPlayer>
#include <QAudioOutput>
#include <QTimer>

class Enemy1: public QObject, public QGraphicsPixmapItem
{
private:
    int row, column;
    int data[12][12];
    int health;
    QTimer *timerenemy;
public:
    bool alive=true;
    Enemy1(int boardData[12][12]);
    void losehealth();
    int gethealth();
    void set_row(int r);
    void set_column(int c);
    void reset_health();
public slots:
    void movingenemy();
};

#endif // ENEMY_H
```

Fig. 1. Enemy Class Header File

The enemies within this game (which are two) are implemented using the functions above. Their function within the game could be summarized as: 1. Moving around 2. Their death signal winning the game 3. They try to take all of Franklin's 3 lives. Within the game, they have a starting position, movement, and health.

In Fig. 1 we see the variables and members involved in the health of the enemy: health (integer variable), alive (boolean variable), losehealth (void function), gethealth (a getter for the integer variable health), and reset_health (void function). Starting out with the variables: health is used to identify what is the object's lifetime. In the constructor in Fig. 2 we can see that it starts out as 2, and the function losehealth() decreases the health by one as Fig. 3 suggests. The gethealth() functions as a normal getter for purposes later in the code. The reset_health() function is to retrieve the enemy's lost health whenever the game resets after Franklin has been shot by an enemy. The enemy starts out alive and dies later (bool alive=true; in Fig. 1).

```
Enemy1::Enemy1(int boardData[12][12])
{
    // Set Image
    QPixmap image("C:/Users/wifi/Downloads/enemy1.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);

    // Set Position
    row = 10;
    column = 6;
    setPos(50 + column * 50, 50 + row * 50);

    // Set data Array
    for (int i = 0; i < 12; i++)
        for (int j = 0; j < 12; j++)
            data[i][j] = boardData[i][j];
    //set health
    health = 2;
}
```

Fig. 2 Enemy Constructor

When it comes to movement, the board data (2-d array) itself, which is later defined in the main.cpp file, is taken as a parameter in the constructor to define a domain within which the movement is possible. The game map is made to correspond to the data variable in the class and movement becomes on the map using the for-loop in the constructor as Fig. 2 shows. In Fig. 2, we can see that the enemy is assigned to be in the 10th row, and the 6th column and then positioned as an image over the game map. The enemy later moves when the player moves. As Fig. 4 shows, the movement is random. There are always 4 possible movements(up, down, left,

or right), hence the variable n is to take a random number from 0 to 3, and according to it a movement is made.

```
void Enemy1::losehealth(){
    health--;
}
int Enemy1::gethealth(){
    return health;
}
void Enemy1::set_row(int r){
    row=r;
}
void Enemy1::set_column(int c){
    column=c;
}
void Enemy1::reset_health(){
    health=2;
}
```

Fig. 3 Some Enemy class functions

```
void Enemy1::movingenemy ()
{
        int n = rand()%4;

        if (n==0 && data[row-1][column]>=0)
        {
            row--;// this is not setting a position
             setPos(50+(50*column),50+(50*row));
            // connect(timertom,SIGNAL(timeout()), this,SLOT(dt()) );
        }
            else if (n==1 && data[row+1][column]>=0)
        {
            row++;
              setPos(50+(50*column),50+(50*row));
                 //connect(timertom,SIGNAL(timeout()), this,SLOT(ut()) );
        }
        else if ((n==2 )&& (data[row][column+1]>=0))
    {
            column++;
           setPos(50+(50*column),50+(50*row));
              //connect(timertom,SIGNAL(timeout()), this,SLOT(rt()) );
    }
    else if ( n ==3 &&data[row][column-1]>=0)
   {
        column--;
         setPos(50+(50*column),50+(50*row));
            //connect(timertom,SIGNAL(timeout()), this,SLOT(lt()) );
   }}
```

Fig. 4 The enemy's random movement

As of now, all the functions not involving the player are completed and all the aspects of the enemy are covered as per the requirements of the milestone 1 submission.

Bullets

The bullet class is one of the simplest classes in this game. It just constructs a distinct object that has a position in the map and a certain image. It is not involved with the player or the game other than being there waiting for the player to collect it. The further usage of this class comes later when it is passed into the constructor of the player in the main.cpp where the rules of the game are implemented. Fig. 5 and 6 show the details of this class.

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

class Bullet :  public QGraphicsPixmapItem
{
public:
    Bullet();
};

#endif // BULLET_H
```

Fig 5. Bullets header file

```
#include "bullet.h"

Bullet::Bullet()
{
    QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/bullet.png");
    image = image.scaledToWidth(40);
    image = image.scaledToHeight(30);
    setPixmap(image);

    // Set Position
    setPos(50 + 1 * 50, 50 + 1 * 50);
}
```

Fig 6. Bullets .cpp file

Powerup, Health, and Door

These classes are similar to the bullet class, and only come to be useful when they are passed as parameters in the player constructor in the main.cpp. The following figures display the details of these classes. The door class is also implemented in the same exact manner.

```
#ifndef POWERUP_H
#define POWERUP_H

#include <QGraphicsPixmapItem>

class Powerup : public QGraphicsPixmapItem
{
public:
    Powerup();
};

#endif // POWERUP_H
```

Fig. 7. Powerup class header file

```
#include "powerup.h"

Powerup::Powerup()
    {
        // Set Image
       QPixmap image("C:/Users/wifi/OneDrive/Documents/GTA/projectresourse/powerupcoin.png");
        image = image.scaledToWidth(50);
        image = image.scaledToHeight(50);
        setPixmap(image);
        // Set Position
        setPos(50 + 10 * 50, 50 + 5 * 50);
    }
```

Fig. 8. Powerup class .cpp file

```
#ifndef HEALTH_H
#define HEALTH_H
#include <QGraphicsPixmapItem>

class Health : public QGraphicsPixmapItem
{
public:
    Health();
private:

};

#endif // HEALTH_H
```

Fig. 9. Health class header file

```
#include "health.h"
#include <QLabel>

Health::Health()
{
// set picture
    QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/heart.png");
    image = image.scaledToWidth(40);
    image = image.scaledToHeight(40);
    setPixmap(image);

    // Set Position
    setPos(50 + 1 * 50, 50 + 13 * 50);
}
```

Fig. 10. Health class .cpp file

Win/Loss

This class is done in a very similar way to the classes above, however, it is particularly different because it takes a boolean variable as a parameter in its constructor as Fig. 11 shows. This allows this class to output a winning or a loss screen if according to this parameter. Fig. 12 shows this in detail.

```
#ifndef WINLOSS_H
#define WINLOSS_H
#include <QGraphicsPixmapItem>

class WinLoss : public QGraphicsPixmapItem
{
public:
    WinLoss(bool wincon);
};

#endif // WINLOSS_H
```

Fig. 11. WinLoss class header file

```
#include "winloss.h"

WinLoss::WinLoss(bool wincon)
{
    if (wincon == true){
    QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/winScreen (2).jpeg");
    image = image.scaledToWidth(750);
    image = image.scaledToHeight(660);
    setPixmap(image);
   setPos( 1, 1 );
    }
    else  {
        QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/lossScreen.png");
        image = image.scaledToWidth(750);
        image = image.scaledToHeight(660);
    setPixmap(image);
   setPos(1 , 1 );
    }
```

Fig. 12. WinLoss class .cpp file

Player

This class is the most important class in this game. It binds together all the elements in relation to the player and to the game later in the main.cpp. The player like the enemies have: 1. Movement 2. Health. 3. Position. Unlike the enemies, the player has: 4. Powerups 5. Bullets. The header file in Fig. 13 shows all these variables and functions and they will be explained shortly.

```
class Player : public QObject, public QGraphicsPixmapItem
{
    Q_OBJECT
private:
    int row, column;
    int data[12][12];
    int health;
    bool alive = true;
     int timecount;
       int i=0; // for timer on bottom right of display
    Enemy1 *enemy1;
    Enemy2 *enemy2;
    Health * htprr1,* htprr2,* htprr3;
    Bullet * bul1,* bul2,* bul3,* bul4;
    Powerup * pow1, * pow2;
    Door *door;
    WinLoss *win, *loss;
    QGraphicsScene *sptr;
    QGraphicsView *vptr;
    QGraphicsTextItem *text[5];
    QMediaPlayer * music = new QMediaPlayer();
    QAudioOutput * musicaudio = new QAudioOutput();
    QTimer *t;
    QTimer *dt;

public:
    Player(int boardData[12][12], Enemy1* ptrenemy1, Enemy2* ptrenemy2, Powerup* fptr,
    QGraphicsScene *sptrr , Health* hptr1,Health* hptr2, Health* hptr3,QGraphicsView *viewptr, Bullet *b1,
    Bullet *b2, Bullet *b3, Bullet *b4, Powerup* pu1, Powerup *pu2,  Door* dr);
    int gethealth();
    bool powered = false;
    bool getstatus();
    void setstatus();
    int decreasehealth();
    void sethealth(int hp);
    void time();



public slots:
    void keyPressEvent(QKeyEvent* event);
    void empower();
    void nopower();
   void resetpos();
   void shoot();
   void norm();
   void shootpower();
   void reducecount();
};
```

Fig. 13. Player class header file

Starting out with the constructor shown in Fig. 15, we can see that the player's health is set to equal 3, and 3 pointers to health objects (graphic hearts) are passed in the constructor corresponding to the health of the player. Together they are the actual and the graphic constituents of the health of the player. The position is then set in the middle of the 12x12 game map, where the row and the column private variables shown in Fig. 13 are each set to 6. The position is then set graphically and the data variable is made to correspond to the board data which will be designed in main.cpp. The scene, view, bullets, powerups, win and loss and all other important constituents of the game. Timers which will later be used for the application of the powerup.

There is no need to show the code for movement, it is simple and straightforward. As each of the arrow keys are pressed Franklin moves to the corresponding direction. Maybe a sample would suffice, as does Fig. 14, where Franklin's movement left and right are shown. We can see the image flips to the left if Franklin moves to the left as in line 118 in contrast to the default transformation done in all other keys (tr.scale(1,1)). Also, Franklin's image changes when he is powered and the same transformation is applied to it.

```
else if (event->key() == Qt::Key_Left && data[row][column - 1] >= 0)
{ // changing character model when moving to left
     if (powered != true){
    QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/nerddefault.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    QTransform tr;
        tr.scale(-1, 1);
      image= image.transformed(tr);
       setPixmap(image);
       player->setAudioOutput(audioOutput);
       player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/projectres
       audioOutput->setVolume(50);
       player->play();
    column--;
     }
     if (powered == true){
         QPixmap image("C:/Users/wifi/OneDrive/Documents/projectresourse/nerdbuff.png");
         image = image.scaledToWidth(50);
         image = image.scaledToHeight(50);
         QTransform tr;
             tr.scale(-1, 1);
           image= image.transformed(tr);
            setPixmap(image);
            player->setAudioOutput(audioOutput);
            player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/proje
            audioOutput->setVolume(50);
            player->play();
         column--;
}
}
```

Fig. 14. Player movement.

```
Player::Player(int boardData[12][12], Enemy1* ptrenemy1, Enemy2 *ptrenemy2, Powerup* fptr, QGraphicsScene *sptrr,
Health* hptr1,Health* hptr2, Health* hptr3,QGraphicsView *viewptr,
Bullet *b1, Bullet *b2, Bullet *b3, Bullet *b4, Powerup* pu1, Powerup *pu2, Door* dr)
{
    health = 3;
    // Set Image
    QPixmap image("C:/Users/wifi/Downloads/My project-1(1).png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);
    // Set Position
    row = 6;
    column = 6;
    setPos(50 + column * 50, 50 + row * 50);
        sptr = sptrr;
        htprr1 =hptr1;
        htprr2 = hptr2;
        htprr3 = hptr3;
        enemy1 = ptrenemy1;
        enemy2 = ptrenemy2;
        vptr= viewptr;
        bul1=b1;
        bul2=b2;
        bul3=b3;
        bul4=b4;
        pow1= pu1;
        pow2= pu2;
        win = new WinLoss(true);
        loss= new WinLoss(false);
        door = dr;
        timecount =5;
        for (int f = 0; f < 5; f++){
            text[i]= new QGraphicsTextItem;
        }
    // Set data Array
    for (int i = 0; i < 12; i++)
        for (int j = 0; j < 12; j++)
            data[i][j] = boardData[i][j];
    //set timer
t= new QTimer(this);
dt= new QTimer(this);

//set music theme
    music->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/GTA/projectresourse/sound/gametheme2.mp3"));
    musicaudio->setVolume(5);
    music->setAudioOutput(musicaudio);
    music->play();

   }
```

Fig. 15. Player constructor

Collisions are handled next with each of the elements. Franklin can collide with: 1. Enemy 2. Powerup 3. Bullet. Each of those collisions has different consequences. If we take the first collisions as we see in Fig. 16, the decreasehealth function is called within the sethealth function to decrease the player's health. There are 3 cases that follow, which is either the health becomes 2, 1, or 0. If it is 2, the reset function (which will be explained later) and a heart is removed, and the relevant music is played. If it is 1, the reset function is called and a heart is removed and the relevant music is played. If it is 0, the last heart is removed and the loss screen is shown while the loss music is played.

```
{ QList<QGraphicsItem*> enemies = collidingItems();
      for (int i = 0; i < enemies.size(); i++)
         {
             if (typeid(*enemies[i]) == typeid(Enemy1) || typeid(*enemies[i])== typeid(Enemy2)){

                 sethealth(decreasehealth());
                       if ( health==2)
                            {
                                sptr->removeItem(htprr3);
                                player->setAudioOutput(audioOutput);
                                player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documen
                                audioOutput->setVolume(50);
                                player->play();
                             resetpos();
                            }

                       else if (health == 1)
                            {
                                sptr->removeItem(htprr2);
                                player->setAudioOutput(audioOutput);
                                player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documen
                                audioOutput->setVolume(50);
                                player->play();
                              resetpos();
                            }
                       else {
                           sptr->removeItem(htprr1);
                           player->setAudioOutput(audioOutput);
                           player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/pr
                           audioOutput->setVolume(50);
                           player->play();
                           sptr->addItem(loss);
                           music->stop();
                           player->setAudioOutput(audioOutput);
                           player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/pr
                           audioOutput->setVolume(50);
                           player->play();
             }
      }}}
```

Fig. 16. Collisions with the enemy

If the collision happens with a powerup item, Fig. 17 shows us what happens. First of all, the item is removed, and the boolean variable of the player named powered becomes true, so all movements will be using the powered image. Then a function called time() is called to end the powerup effect after some time. This function, as shown in Fig. 19, calls on the function empower(), which will be discussed shortly, and uses QTimer to start a singleshot timer that calls the function nopower() to end the effects of the empower function after 5 seconds and sets the powered boolean variable back to false. The empower function, shown in Fig. 19, sets Franklin's image to the buffed one, and then a timer is started the decrements the displayed count representing the five seconds using another that is not a singleshot timer, one which calls on the function that reduces the time count every second as shown in the function. After the five seconds of the first timer the nopower() function is called and sets the image back to the original one and reverts the boolean variable powered to false again. The second timer (the one responsible for the count) is stopped when the timecount variable is decremented from 0 to -1 as is shown in Fig. 18 in the reducecount() function that also removes the text.

```
{ QList<QGraphicsItem*> items = collidingItems();
    for (int i = 0; i < items.size(); i++)
    {
        if (typeid(*items[i]) == typeid(Powerup)){
            scene()->removeItem(items[i]);
            powered=true;
            time();
            }
    }}
```

Fig. 17. Collision with powerup item

```
void Player::reducecount(){
    sptr->removeItem(text[i]);
    text[i] = sptr->addText(QString::number(timecount));
    text[i]->setDefaultTextColor(QColorConstants::White);
    text[i]->setPos(50+11*50, 50 +13 * 50);
    text[i]->setTextWidth(50);
    timecount--;
    if (timecount ==-1){
        dt->stop();}
}
```

Fig. 18. reducecount() function of class player

```
void Player::empower(){
    i=0;
    timecount = 5;
    QMediaPlayer *player = new QMediaPlayer;
         QAudioOutput *audioOutput = new QAudioOutput;
    QPixmap image ("C:/Users/wifi/OneDrive/Documents/projectresourse/nerdbuff.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);
    player->setAudioOutput(audioOutput);
    player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/projectresou
    audioOutput->setVolume(50);
    player->play();
    powered=true;
    dt->start(1000);
    dt->setSingleShot(false);
    connect(dt, SIGNAL(timeout()), this, SLOT (reducecount()));
}


int Player::gethealth(){
    return health;
}
bool Player::getstatus(){
    return alive;
}
void Player::setstatus(){
    alive = false;
}
int Player::decreasehealth(){
    return --health;
}
void Player::sethealth(int hp){
    health = hp;
}

void Player::time(){
      empower();
t->start(5000);
t->setSingleShot(true);
connect(t, SIGNAL(timeout()), this, SLOT (nopower()));
}
void Player::nopower(){
    QPixmap image ("C:/Users/wifi/OneDrive/Documents/projectresourse/nerddefault.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);
    powered = false;
}
```

Fig. 19. Some functions of the class player

Collisions with bullets have the most important consequences in this game as shown in Fig. 21. First of all, the shoot() function is called which creates sound for shooting as Fig. 22 shows, and the image of Franklin changes to a shooting image. After half a second, the function norm() is called which returns Franklin to his original shape. Then the bullet graphic is removed and enemy loses health if health is 2, then his health decreases by 1. Otherwise, the enemy is fully removed and the boolean (alive) belonging to the object is set to false so that the bullets could be directed at the other enemy now.

If the second enemy dies a door shows and if the player collides with this door a certain music is played and the winning screen shows as Fig. 20 shows.

```
{ QList<QGraphicsItem*> doorc = collidingItems();
    for (int i = 0; i < doorc.size(); i++)
    {
        if (typeid(*doorc[i]) == typeid(Door)){
            sptr->addItem(win);
            music->stop();
            player->setAudioOutput(audioOutput);
            player->setSource(QUrl::fromLocalFile("C:/Us
            audioOutput->setVolume(50);
            player->play();
            }
    }}
```

Fig. 20. Collision with door

```
{QList<QGraphicsItem*> bullets = collidingItems();
       for (int i = 0; i < bullets.size(); i++)
       {
           if (typeid(*bullets[i]) == typeid(Bullet) && enemy1->alive==true){
               shoot();
               t->start(500);
               t->setSingleShot(true);
               connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                scene()->removeItem(bullets[i]);
                    if(enemy1->gethealth()!=1){
                        enemy1->losehealth();
                        if(powered == true){
                            shootpower();
                            t->start(500);
                            t->setSingleShot(true);
                            connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                            sptr->removeItem(enemy1);
                            enemy1->alive=false;
                            enemy1->losehealth();
                            enemy1->losehealth();
//                            sptr->addItem(win);
                        }
                    }
                    else {
                        sptr->removeItem(enemy1);
                        enemy1->alive=false;
                        }
            }
           else
           {
               if (typeid(*bullets[i])== typeid(Bullet))
               {
                   shoot();
                   t->start(500);
                   t->setSingleShot(true);
                   connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                   QPixmap image ("C:/Users/wifi/OneDrive/Documents/projectresou
                   image = image.scaledToWidth(50);
                   image = image.scaledToHeight(50);
                   setPixmap(image);
                   scene()->removeItem(bullets[i]);
                   if(enemy2->gethealth()!=1)
                   {
                       shoot();
                       t->start(500);
                       t->setSingleShot(true);
                       connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                       enemy2->losehealth();
                       if(powered == true)
```

```
                {                       shootpower();
                    t->start(500);
                    t->setSingleShot(true);
                    connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                    sptr->removeItem(enemy2);
                    enemy2->losehealth();
                    enemy2->losehealth();
                    sptr->addItem(door);

                }
            }
            else
            {
                shootpower();
                t->start(500);
                t->setSingleShot(true);
                connect(t, SIGNAL(timeout()), this, SLOT (norm()));
                sptr ->removeItem(enemy2);
                sptr->addItem(door);
                sptr->removeItem(htprr1);
                sptr->removeItem(htprr2);
                sptr->removeItem(htprr3);
            }
        }
    }
}
```

Fig. 21. Collision with bullet items

```
void Player::shoot(){
    QMediaPlayer *player = new QMediaPlayer;
         QAudioOutput *audioOutput = new QAudioOutput;
    QPixmap image ("C:/Users/wifi/OneDrive/Documents/projectresourse/nerdshootnobuff.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);
    player->setAudioOutput(audioOutput);
    player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/projectresourse/s
    audioOutput->setVolume(50);
    player->play();
}
void Player::shootpower(){
    QMediaPlayer *player = new QMediaPlayer;
    QAudioOutput *audioOutput = new QAudioOutput;
    QPixmap image ("C:/Users/wifi/OneDrive/Documents/projectresourse/nerdbuffshoot.png");
    image = image.scaledToWidth(50);
    image = image.scaledToHeight(50);
    setPixmap(image);
    player->setAudioOutput(audioOutput);
    player->setSource(QUrl::fromLocalFile("C:/Users/wifi/OneDrive/Documents/projectresourse/s
    audioOutput->setVolume(50);
    player->play();
}
```

Fig. 22. More functions for player class

Explanation of Design

In this section, the main.cpp is going to be discussed and explained in detail. The first step is to set up the QGraphicsView and QGraphicsScene which will later be made into the game map, which takes us to the next step. The text file showing the map is opened and read and stored into the stream variable which is later put in a nested loop and all the content of stream is put into temp, while the two dimensional array stores the values of temp now that they are turned into integers as Fig. 23 shows.

```
QApplication a(argc, argv);
QGraphicsView *view= new QGraphicsView;
QGraphicsScene *scene= new QGraphicsScene;

view->setFixedSize(800, 700);
view->setWindowTitle("GTA");
QBrush brush(Qt::black);
view->setBackgroundBrush(brush);

QFile file("C:/Users/wifi/Downloads/mapaux.txt"); // parse the text file
file.open(QIODevice::ReadOnly);
QTextStream stream(&file);
int boardData[12][12];
QString temp;
for (int i = 0; i < 12; i++)
    for (int j = 0; j < 12; j++)
    {
        stream >> temp;
        boardData[i][j] = temp.toInt();
    }
```

Fig. 23. Setting up the game map

Later, we as designers find images and allocate them to specific numbers into the data board two dimensional array using the setPixmap functions in a nested for-loop as Fig. 24 shows. Later, all the objects are declared, positioned, and added to the scene and later included in the declaration of the player in its constructor as Fig. 25 shows.

```
QGraphicsPixmapItem boardItems[12][12]; // creating board
for (int i = 0; i < 12; i++)
    for (int j = 0; j < 12; j++)
    {
        // Set Image

        if(((i == 0) &&  (j > 7 || j == 0))){
            boardItems[i][j].setPixmap(treeskyeImage); //set pavement tile next to buildings

        }
        else if((i ==2) && (j > 2 && j < 10)){
             boardItems[i][j].setPixmap(pavementImage); //set pavement tile next to buildings

         }
        else if((i ==4 && j==6)){
            boardItems[i][j].setPixmap(stopsignImage); //set pavement tile next to buildings
        }
        else if((i ==6 && (j < 4 && j > 0))){
            boardItems[i][j].setPixmap(coneImage); //set pavement tile next to buildings
        }
        else if((i ==9 && j==3)){
            boardItems[i][j].setPixmap(carsImage); //set pavement tile next to buildings
        }
        else if((i ==10 && (j==3 || (j > 3 && j < 5)))){
            boardItems[i][j].setPixmap(carsImage); //set pavement tile next to buildings
        }
        else if((i ==3 || i==4) && (j==3)){
            boardItems[i][j].setPixmap(carsImage); //set pavement tile next to buildings
        }
        else if(i ==4 && j > 5){
            boardItems[i][j].setPixmap(building1pav); //set pavement tile next to buildings
        }
        else if((i == 5) && (j==6)){
            boardItems[i][j].setPixmap(houseImage); //set pavement tile next to buildings
        }
        else if((i == 6) && (j==6)){
            boardItems[i][j].setPixmap(grassImage); //set pavement tile next to buildings
        }
        else if (boardData[i][j] == -1 && (!((i == 0) &&  (j > 7))))
            boardItems[i][j].setPixmap(treeImage);
        else if (boardData[i][j]== 1)
            boardItems[i][j].setPixmap(roadImage);
        else if (boardData[i][j]== -2)
            boardItems[i][j].setPixmap(buildingImage1);
        else if (boardData[i][j]== -3)
            boardItems[i][j].setPixmap(buildingImage2);
        else if(boardData[i][j]== -4){
            boardItems[i][j].setPixmap(pavementImage); //set pavement tile next to buildings
        }
```

Fig. 24. Assigning creates QPixmaps to the 2-dimensional QGraphicsPixmapItem game board

```
Player player(boardData, &enemy1, &enemy2, &powerup1, scene, &heart1, &heart2, &heart3,
              view, &bullet1, &bullet2, &bullet3, &bullet4, &powerup1, &powerup2, &door);
scene->addItem(&player);



WinLoss win(true), loss(false);
player.setFlag(QGraphicsPixmapItem::ItemIsFocusable);
player.setFocus();

view->setScene(scene);
view->show();
return a.exec();
```

Fig. 25. Declaring player and finishing up the program in main.cpp

Finally the game looks as shown in Fig. 26 and the next section will describe the logistics of this project as far as milestone 1 is concerned.

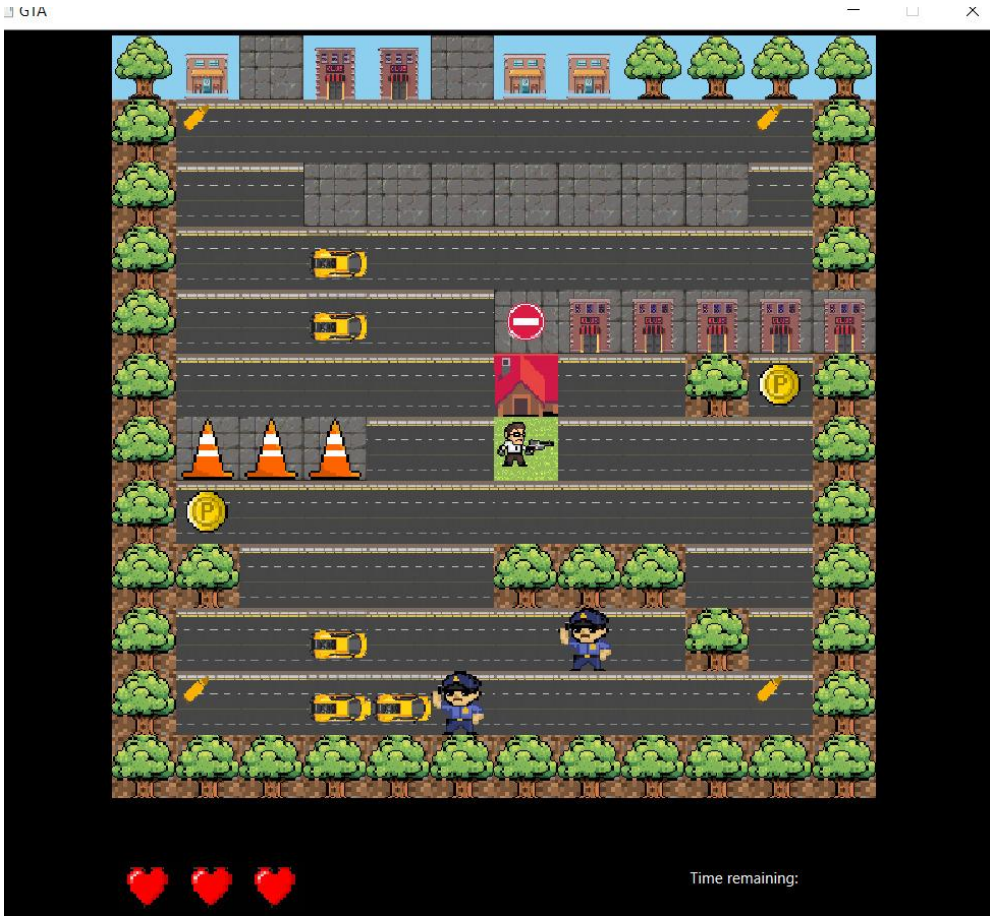

Fig. 26. The Game's Final Look

Work Distribution and Meet-ups

Meeting Time	Attendance	Tasks attempted
Second week after the project was posted	Omar	Set up map+Classes+sounds and music+main.cpp
	-	
	-	
Saturday and Sunday 19 and 20/Nov	Omar	Enemy movement+Powerup+Bullets+Player health+health Display
	Khaled	Timer+Door+Powerup+Bullets
	Abdelrahman	Follow up with what we have wrote
Monday 21/Nov	Omar	Timer + win and loss screens
	Khaled	Timer + win and loss screens
	-	
Tuesday 22/Nov	Omar	Fine Tuning
	Khaled	Report+Reviewing the code and checking for mistakes
	Abdelrahman	New map design (might be used in milestone 2)

General Participation and knowledge of code:

Omar Almekhlafy	All code
Khaled Hamza	All code
Abdelrahman Elhamahmy	Map Designing