**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КОМПЬЮТЕРНАЯ ИГРА «ТАНКИ 2D»

**ТЕКСТ ПРОГРАММЫ**

КП.ПО6.200125-04 12 00

## Листов 22

|  |  |
| --- | --- |
| Руководитель | П.А. Кочурко |
| Выполнил | Д. М. Мартынович |
| Консультант |  |
| по ЕСПД | П.А. Кочурко |
|  |  |

2022

**СОДЕРЖАНИЕ**

Battle\_city.pro

#-------------------------------------------------

#

# Project created by QtCreator 2022-08-21T10:20:47

#

#-------------------------------------------------

QT += core gui \

multimedia

QT += widgets

QT += core5compat

TARGET = Battle\_City

TEMPLATE = app

# The following define makes your compiler emit warnings if you use

# any feature of Qt which as been marked as deprecated (the exact warnings

# depend on your compiler). Please consult the documentation of the

# deprecated API in order to know how to port your code away from it.

DEFINES += QT\_DEPRECATED\_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.

# In order to do so, uncomment the following line.

# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

client.cpp \

gamemenu.cpp \

main.cpp \

bullet.cpp \

player.cpp \

gameloop.cpp \

enemy.cpp \

score.cpp

HEADERS += \

bullet.h \

client.h \

gamemenu.h \

player.h \

gameloop.h \

enemy.h \

score.h

RESOURCES += \

res.qrc

win32:CONFIG(release, debug|release): LIBS += -L$$PWD/libs/about/build/release/ -lAbout.1.0.0

else:win32:CONFIG(debug, debug|release): LIBS += -L$$PWD/libs/about/build/debug/ -lAbout.1.0.0

else:unix: LIBS += -L$$PWD/libs/about/build/ -lAbout.1.0.0

INCLUDEPATH += $$PWD/libs/about/About

DEPENDPATH += $$PWD/libs/about/build

FORMS += \

client.ui

bullet.h

#ifndef BULLET\_H

#define **BULLET\_H**

#include <QGraphicsRectItem>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QTimer>

#include <QObject>

#include "player.h"

#include "enemy.h"

#include "gameloop.h"

#include <QAudioOutput>

class **Player**;

class **Enemy**;

class **GameLoop**;

class **Bullet** : public QObject, public QGraphicsPixmapItem

{

Q\_OBJECT

private:

QTimer \***timer**;

int **direct**;

int \*\***map**;

Player \***pl**;

Enemy \***enemy**;

int **flag**;

QMediaPlayer \***bullet\_sound**;

public:

**Bullet**(int **direct**, int \*\***map**, Player \***pl**, int **flag**);

int \*\***ft\_get\_map**();

int **ft\_check\_hit**(int **x**, int **y**);

public slots:

void **move**();

};

#endif // BULLET\_H

Bullet.cpp

#include "bullet.h"

#include <QTimer>

#include "player.h"

#include "gameloop.h"

#include "score.h"

#include <QGraphicsScene>

#include <iostream>

Bullet::**Bullet**(int **direct**, int \*\***map**, Player \***pl**, int **flag**) : QObject() , QGraphicsPixmapItem()

{

// draw a bullet

this->map = map;

this->direct = direct;

this->flag = flag;

this->enemy = pl->ft\_get\_enemy();

if (this->direct == 1)

setPixmap(QPixmap(":/pics/shot.png"));

if (this->direct == 2)

setPixmap(QPixmap(":/pics/shot\_d.png"));

if (this->direct == 3)

setPixmap(QPixmap(":/pics/shot\_r.png"));

if (this->direct == 4)

setPixmap(QPixmap(":/pics/shot\_l.png"));

this->timer = new QTimer();

this->pl = pl;

connect(timer, SIGNAL(timeout()), this, SLOT(move()));

timer->start(5);

}

void Bullet::**move**()

{

float **x\_p**;

float **y\_p**;

int **check\_x**;

int **check\_y**;

if (this->direct == 1)

{

x\_p = x();

y\_p = y() - 5;

}

if (this->direct == 2)

{

x\_p = x();

y\_p = y() + 5;

}

if (this->direct == 3)

{

x\_p = x() + 5;

y\_p = y();

}

if (this->direct == 4)

{

x\_p = x() - 5;

y\_p = y();

}

check\_x = int(x\_p / 64);

check\_y = int(y\_p / 64);

setPos(x\_p, y\_p);

if (this->map[check\_y][check\_x] == 9 && this->flag == 1)

{

this->map[check\_y][check\_x] = 0;

this->enemy->ft\_spawn();

this->enemy->setPos(this->enemy->ft\_get\_x() \* 64, this->enemy->ft\_get\_y() \* 64);

this->map[this->enemy->ft\_get\_y()][this->enemy->ft\_get\_x()] = 9;

this->pl->ft\_update\_map(this->*map*);

this->pl->ft\_get\_score()->ft\_increase(1);

scene()->removeItem(this);

delete this->timer;

delete this;

return ;

}

else if (this->map[check\_y][check\_x] > 0 && this->map[check\_y][check\_x] != 9)

{

if (this->map[check\_y][check\_x] == 5)

{

this->pl->setPixmap(QPixmap(":/pics/colorstone.png"));

this->pl->clearFocus();

this->pl->ft\_get\_score()->ft\_increase(2);

}

if (this->map[check\_y][check\_x] != 3)

{

this->map[check\_y][check\_x] = 0;

this->pl->ft\_update\_map(this->*map*);

QGraphicsPixmapItem \*\***grid**;

grid = this->pl->ft\_get\_grid();

grid[check\_y][check\_x].setPixmap(QPixmap(":/pics/colorstone.png"));

}

scene()->removeItem(this);

delete this->timer;

delete this;

return ;

}

else if ((check\_x == this->pl->ft\_get\_y() && (check\_y == this->pl->ft\_get\_x())

&& this->flag == 0))

{

this->pl->setPixmap(QPixmap(":/pics/colorstone.png"));

this->pl->clearFocus();

scene()->removeItem(this);

this->pl->ft\_get\_score()->ft\_increase(2);

delete this->timer;

delete this;

return ;

}

}

Client.h

#ifndef CLIENT\_H

#define **CLIENT\_H**

#include <QMainWindow>

#include <QTcpSocket>

#include <QMessageBox>

#include <QDebug>

#include <QJsonDocument>

#include <QJsonObject>

#include <QJsonParseError>

#include <QFileInfo>

#include <QDateTime>

#include <QString>

#include <QFile>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Client**; }

QT\_END\_NAMESPACE

class **Client** : public QMainWindow

{

Q\_OBJECT

public:

**Client**(QWidget \***parent** = nullptr);

~**Client**();

QTcpSocket \***socket**;

QByteArray **data**;

QJsonDocument **doc**;

QJsonParseError **docError**;

void **createLib**();

QDateTime **getLibVersion**();

bool **complexData**;

int **requireSize**;

QString **libName**;

QByteArray **libBytes**;

public slots:

void **sockReady**();

void **sockDisc**();

private slots:

void **on\_disconnectFromServer\_clicked**();

void **on\_downloadFromServer\_clicked**();

void **on\_connectToServer\_clicked**();

private:

Ui::Client \***ui**;

};

#endif // CLIENT\_H

Client.cpp

#include "client.h"

#include "ui\_client.h"

Client::**Client**(QWidget \***parent**)

: QMainWindow(*parent*)

, ui(new Ui::Client)

{

ui->setupUi(this);

socket = nullptr;

complexData = false;

}

Client::~**Client**()

{

delete ui;

}

void Client::**sockDisc**()

{

socket->deleteLater();

}

void Client::**sockReady**()

{

if(socket->*waitForConnected*(500))

{

socket->*waitForReadyRead*(500);

if(!complexData)

{

data = socket->readAll();

}

else

{

libBytes.append(socket->readAll());

qDebug() << "Client is downloading packets..\n";

qDebug() << "Downloaded Size = " << libBytes.size() << " Need size = " << requireSize;

if(libBytes.size() == requireSize)

{

qDebug() << "Client downloads the .dylib";

createLib();

qDebug() << "Dylib is created in File System";

libBytes.clear();

complexData = false;

}

}

doc = QJsonDocument::fromJson(data, &*docError*);

if(docError.errorString().toInt() == QJsonParseError::NoError)

{

if((doc.object().value("type").toString() == "connect") &&

(doc.object().value("status").toString() == "yes"))

{

QMessageBox::information(this, "Информация", "Соединение установлено");

}

else if((doc.object().value("type").toString() == "info") &&

(doc.object().value("resp").toString() == "dylib"))

{

//QMessageBox::information(this, "Информация", "Соединение не установлено");

requireSize = doc.object().value("size").toInt();

libName = doc.object().value("libname").toString();

QDateTime **serverLibVersion** = QDateTime::fromString(doc.object().value("version").toString());

QDateTime **currentLibVersion** = getLibVersion();

qDebug() << "Client gets the info (libname, libsize, libversion) from server";

if (serverLibVersion > currentLibVersion)

{

complexData = true;

socket->write("{\"type\":\"download\", \"params\":\"dylib\"}");

socket->*waitForBytesWritten*(500);

qDebug() << "Client sends a request (download) to server";

}

else

{

qDebug() << "Dylib is not needed to updating";

QMessageBox::information(this, "Информация", "Обновлений не требуется");

}

}

}

data.clear();

}

}

void Client::**on\_disconnectFromServer\_clicked**()

{

if (socket != nullptr){

QMessageBox::information(this, "Информация", "Клиент отключился от сервера");

delete socket;

}

socket = nullptr;

qDebug() << "Client disconnected";

}

void Client::**on\_downloadFromServer\_clicked**()

{

qDebug() << "Client sends a request (get info) to server";

if ((socket != nullptr) && (socket->isOpen()))

{

qDebug() << "Client sends a request (get info) to server";

socket->write("{\"type\":\"recInfo\", \"resp\":\"dylib\"}");

socket->*waitForBytesWritten*(500);

}

else

{

QMessageBox::information(this, "Информация", "Клиент не подключен к серверу");

}

}

void Client::**on\_connectToServer\_clicked**()

{

QMessageBox::information(this, "Информация", "Попытка соединиться с сервером");

qDebug() << "Client sends a request (connect) to server";

socket = new QTcpSocket(this);

connect(socket, SIGNAL(readyRead()), this, SLOT(sockReady()));

connect(socket, SIGNAL(disconnected()), this, SLOT(sockDisc()));

socket->*connectToHost*("127.0.0.1", 5555);

}

QDateTime Client::**getLibVersion**()

{

QFileInfo **lib**("//Users//ikrut0nardo//Desktop//MyProjects//Qt//ОСИСП//Tanks-master//libs//about//build//" + libName);

QDateTime **time** = lib.lastModified();

return time;

}

void Client::**createLib**()

{

QFile::remove("//Users//ikrut0nardo//Desktop//MyProjects//Qt//ОСИСП//Tanks-master//libs//about//build//" + libName);

QFile **lib**("//Users//ikrut0nardo//Desktop//MyProjects//Qt//ОСИСП//Tanks-master//libs//about//build//" + libName);

if (lib.*open*(QIODevice::WriteOnly))

{

lib.write(libBytes);

}

lib.*close*();

}

Enemy.h

#ifndef ENEMY\_H

#define **ENEMY\_H**

#include <ctime>

#include <QTimer>

#include <cstdlib>

#include "player.h"

#include "bullet.h"

#include "gameloop.h"

class **Player**;

class **Bullet**;

class **GameLoop**;

class **Enemy** : public QObject, public QGraphicsPixmapItem

{

Q\_OBJECT

public slots:

void **move**();

void **shoot**();

private:

int \*\***map**;

int **pos\_x**;

int **pos\_y**;

int **direct**;

Player \***pl**;

public:

**Enemy** (int \*\***map**);

int **ft\_get\_x**();

int **ft\_get\_y**();

void **ft\_spawn**();

void **ft\_set\_pl**(Player \***pl**);

Player \***ft\_get\_pl**();

int **ft\_get\_direct**();

};

#endif // ENEMY\_H

Enemy.cpp

#include "enemy.h"

#include "gameloop.h"

#include "bullet.h"

#include "player.h"

#include "score.h"

#include <iostream>

class **GameLoop**;

class **Enemy**;

class **Bullet**;

class **Player**;

void Enemy::**move**()

{

srand(time(0));

this->direct = rand() % 4 + 1;

if (this->direct == 4)

{

this->setPixmap(QPixmap(":/pics/enemy\_l.png"));

if (this->map[this->pos\_y][this->pos\_x - 1] == 0)

{

this->map[this->pos\_y][this->pos\_x] = 0;

this->pos\_x--;

this->setPos(x() - 64, y());

this->map[this->pos\_y][this->pos\_x] = 9;

}

}

if (this->direct == 3)

{

this->setPixmap(QPixmap(":/pics/enemy\_r.png"));

if (this->map[this->pos\_y][this->pos\_x + 1] == 0)

{

this->map[this->pos\_y][this->pos\_x] = 0;

this->setPos(x() + 64, y());

this->pos\_x++;

this->map[this->pos\_y][this->pos\_x] = 9;

}

}

if (this->direct == 2)

{

this->setPixmap(QPixmap(":/pics/enemy\_d.png"));

if (this->map[this->pos\_y + 1][this->pos\_x] == 0)

{

this->map[this->pos\_y][this->pos\_x] = 0;

this->setPos(x(), y() + 64);

this->pos\_y++;

this->map[this->pos\_y][this->pos\_x] = 9;

}

}

if (this->direct == 1)

{

this->setPixmap(QPixmap(":/pics/enemy.png"));

if (this->map[this->pos\_y - 1][this->pos\_x] == 0)

{

this->map[this->pos\_y][this->pos\_x] = 0;

this->setPos(x(), y() - 64);

this->pos\_y--;

this->map[this->pos\_y][this->pos\_x] = 9;

}

}

}

Enemy::**Enemy** (int \*\***map**) : QObject(), QGraphicsPixmapItem()

{

this->direct = 2;

this->map = map;

srand(time(0));

ft\_spawn();

}

int Enemy::**ft\_get\_x**()

{

return (this->pos\_x);

}

int Enemy::**ft\_get\_y**()

{

return (this->pos\_y);

}

void Enemy::**ft\_spawn**()

{

int **choice**;

choice = rand() % 3 + 1;

if (choice == 3)

{

this->pos\_x = 1;

this->pos\_y = 11;

}

if (choice == 1)

{

this->pos\_x = 11;

this->pos\_y = 1;

}

if (choice == 2)

{

this->pos\_x = 11;

this->pos\_y = 11;

}

}

void Enemy::**ft\_set\_pl**(Player \***pl**)

{

this->pl = pl;

}

Player \*Enemy::**ft\_get\_pl**()

{

return (this->pl);

}

void Enemy::**shoot**()

{

Bullet \***bull**;

bull = new Bullet(this->direct, this->*map*, this->*pl*, 0);

if (this->direct == 1 || this->direct == 2)

bull->setPos(x() + 30, y());

if (this->direct == 3)

bull->setPos(x(), y() + 30);

if (this->direct == 4)

bull->setPos(x(), y() + 30);

scene()->addItem(*bull*);

}

int Enemy::**ft\_get\_direct**()

{

return (this->direct);

}

Gameloop.h

#ifndef GAME\_LOOP\_H

#define **GAME\_LOOP\_H**

#define **MAP\_SIZE** 13

#define **WIDTH** 1132

#define **HEIGHT** 832

#include <cstdlib>

#include <ctime>

#include <iostream>

#include <QKeyEvent>

#include <QGraphicsView>

#include <QWidget>

#include <QGraphicsScene>

#include <QGraphicsPixmapItem>

#include "player.h"

#include "enemy.h"

#include "bullet.h"

#include "score.h"

#include <QBrush>

#include <QTimer>

#include <QColor>

class **Player**;

class **Enemy**;

class **Bullet**;

class **Score**;

class **GameLoop** : public QGraphicsView

{

private:

int \*\***map**;

QGraphicsScene \***scene**;

QGraphicsView \***viewer**;

QGraphicsPixmapItem \*\***grid**;

Score \***score**;

Player \***player**;

Enemy \***enemy**;

QTimer \***timer\_move**;

QTimer \***timer\_shoot**;

public:

int \*\***ft\_get\_map**();

void **ft\_update\_map**(int \*\***map**);

void **ft\_print\_map**();

void **ft\_set\_headquarters**();

void **ft\_add\_obstacle**();

void **ft\_make\_map**();

void **ft\_destroy\_map**();

**GameLoop**();

~**GameLoop**();

};

#endif // GAME\_LOOP\_H

Gameloop.cpp

#include <QGraphicsScene>

#include <QGraphicsView>

#include <QBrush>

#include <QTimer>

#include <QColor>

#include <QGraphicsRectItem>

#include <QMediaPlayer>

#include <QGraphicsScene>

#include "player.h"

#include "enemy.h"

#include "gameloop.h"

#include "score.h"

int \*\*GameLoop::**ft\_get\_map**()

{

return (this->map);

}

void GameLoop::**ft\_update\_map**(int \*\***map**)

{

int **i**;

int **j**;

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

this->map[i][j] = map[i][j];

j++;

}

i++;

}

}

void GameLoop::**ft\_print\_map**()

{

int **i**;

int **j**;

i = 0;

std::cout << "\t\tGAME MAP\n\n";

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

std::cout << this->map[i][j] << " ";

j++;

}

std::cout << "\n";

i++;

}

}

void GameLoop::**ft\_set\_headquarters**()

{

int **i\_1**;

int **i\_2**;

int **j\_1**;

int **j\_2**;

j\_1 = MAP\_SIZE / 2 - 1;

j\_2 = j\_1 + 2;

i\_1 = MAP\_SIZE - (MAP\_SIZE / 4);

while (j\_1 <= j\_2)

{

this->map[i\_1][j\_1] = 6;

j\_1++;

}

while (i\_1 < MAP\_SIZE - 1)

{

this->map[i\_1][j\_1 - 1] = 6;

i\_1++;

}

i\_2 = MAP\_SIZE - (MAP\_SIZE / 4);

j\_1 = MAP\_SIZE / 2 - 1;

while (i\_2 < MAP\_SIZE - 1)

{

this->map[i\_2][j\_1] = 6;

i\_2++;

}

this->map[MAP\_SIZE - 2][MAP\_SIZE / 2] = 5;

}

void GameLoop::**ft\_add\_obstacle**()

{

int **i**;

int **j**;

int **counter**;

srand(time(0));

counter = 0;

while (counter < 25)

{

i = 2 + rand() % (MAP\_SIZE - 1 - 2 + 1);

j = 2 + rand() % (MAP\_SIZE - 1 - 2 + 1);

if (this->map[i][j] == 0 && i != j)

{

this->map[i][j] = 1;

this->map[i][MAP\_SIZE - j - 1] = 1;

counter++;

}

}

}

void GameLoop::**ft\_make\_map**()

{

int **i**;

int **j**;

i = 0;

this->map = new int\*[MAP\_SIZE];

while (i < MAP\_SIZE)

{

this->map[i] = new int[MAP\_SIZE];

i++;

}

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

if (i == 0 || i == MAP\_SIZE - 1

|| j == 0 || j == MAP\_SIZE - 1)

this->map[i][j] = 3;

else

this->map[i][j] = 0;

j++;

}

i++;

}

ft\_set\_headquarters();

ft\_add\_obstacle();

this->map[1][1] = 0;

this->map[1][MAP\_SIZE - 2] = 0;

this->map[MAP\_SIZE - 2][1] = 0;

this->map[MAP\_SIZE - 2][MAP\_SIZE - 2] = 0;

}

void GameLoop::**ft\_destroy\_map**()

{

int **i**;

i = 0;

while (i < MAP\_SIZE)

{

delete this->map[i];

i++;

}

delete [](this->map);

}

GameLoop::**GameLoop**() : QGraphicsView()

{

this->scene = new QGraphicsScene();

this->viewer = new QGraphicsView();

this->viewer->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

this->viewer->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

this->viewer->setScene(this->*scene*);

this->viewer->setFixedSize(1132, 832);

this->scene->setSceneRect(0, 0, 1132, 832);

ft\_make\_map();

this->score = new Score();

this->score->setPos(872, 50);

this->player = new Player();

this->player->ft\_create\_map(*map*);

this->player->ft\_spanw();

this->player->setPixmap(QPixmap(":/pics/player.png"));

this->player->setFlag(QGraphicsItem::ItemIsFocusable);

this->player->setPos(player->ft\_get\_x() \* 64, player->ft\_get\_y() \* 64);

this->player->setFocus();

this->scene->setBackgroundBrush(QBrush(QColor(0x009096a0)));

int **i**;

int **j**;

i = 0;

this->grid = new QGraphicsPixmapItem \*[MAP\_SIZE];

while (i < MAP\_SIZE)

{

this->grid[i] = new QGraphicsPixmapItem [MAP\_SIZE];

i++;

}

i = 0;

this->map = player->ft\_player\_map();

this->enemy = new Enemy(*map*);

this->enemy->setPixmap(QPixmap(":/pics/enemy\_d.png"));

this->enemy->setPos(this->enemy->ft\_get\_x() \* 64, this->enemy->ft\_get\_y() \* 64);

this->player->ft\_set\_enemy(this->*enemy*);

this->player->ft\_set\_score(this->*score*);

this->enemy->ft\_set\_pl(this->*player*);

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

if (this->map[i][j] == 3)

this->grid[i][j].setPixmap(QPixmap(":/pics/greystone.png"));

if (this->map[i][j] == 5)

this->grid[i][j].setPixmap(QPixmap(":/pics/eagle.png"));

if (this->map[i][j] == 1)

this->grid[i][j].setPixmap(QPixmap(":/pics/greystone.png"));

if (this->map[i][j] == 6)

this->grid[i][j].setPixmap(QPixmap(":/pics/redbrick.png"));

if (this->map[i][j] == 0 || map[i][j] == 9)

this->grid[i][j].setPixmap(QPixmap(":/pics/colorstone.png"));

this->grid[i][j].setPos(j \* 64, i \* 64);

this->scene->addItem(&(this->*grid*[i][j]));

j++;

}

i++;

}

this->scene->addItem(this->*player*);

this->scene->addItem(this->*enemy*);

this->player->ft\_set\_grid(this->*grid*);

this->grid = this->player->ft\_get\_grid();

this->timer\_move = new QTimer();

QObject::connect(this->timer\_move, SIGNAL(timeout()), this->enemy, SLOT(move()));

this->timer\_move->start(450);

this->timer\_shoot = new QTimer();

QObject::connect(this->timer\_shoot, SIGNAL(timeout()), this->enemy, SLOT(shoot()));

this->timer\_shoot->start(550);

this->scene->addItem(this->*score*);

this->viewer->show();

}

GameLoop::~**GameLoop**()

{

int **i**;

i = 0;

while (i < MAP\_SIZE)

{

delete this->map[i];

i++;

}

delete []this->map;

i = 0;

while (i < MAP\_SIZE)

{

delete this->grid[i];

i++;

}

delete []this->grid;

delete this->scene;

delete this->viewer;

delete this->score;

delete this->player;

delete this->enemy;

delete this->timer\_move;

delete this->timer\_shoot;

}

Gamemenu.h

#ifndef GAMEMENU\_H

#define **GAMEMENU\_H**

#include <QMainWindow>

#include <QPushButton>

#include <QVBoxLayout>

class **gameMenu**:public QWidget

{

Q\_OBJECT

public:

**gameMenu**();

QWidget \***window**;

QPushButton \***button1**;

QPushButton \***button2**;

QPushButton \***button3**;

public slots:

void **startGame**();

void **about**();

void **setMenu**();

};

#endif // GAMEMENU\_H

Gamemenu.cpp

#include "gamemenu.h"

#include "QtCore/qobjectdefs.h"

#include "gameloop.h"

#include "about.h"

#include "client.h"

gameMenu::**gameMenu**()

{

this->window = new QWidget();

button1 = new QPushButton("start");

button2 = new QPushButton("about");

button3 = new QPushButton("check for update");

QVBoxLayout \***layout** = new QVBoxLayout(*window*);

layout->addWidget(*button1*);

layout->addWidget(*button2*);

layout->addWidget(*button3*);

QFont **font**("Times New Roman");

this->window->setFont(font);

this->window->resize(400,500);

this->window->show();

connect(button1,SIGNAL(released()),this,SLOT(startGame()));

connect(button2,SIGNAL(released()),this,SLOT(about()));

connect(button3,SIGNAL(released()),this,SLOT(setMenu()));

}

void gameMenu::**startGame**()

{

GameLoop \***gm**;

gm = new GameLoop;

}

void gameMenu::**about**()

{

About \***about** = new About();

}

void gameMenu::**setMenu**()

{

window->close();

Client \***w** = new Client();

w->setWindowTitle("Update");

w->show();

}

Player.h

#ifndef PLAYER\_H

#define **PLAYER\_H**

#include <QGraphicsRectItem>

#include <QGraphicsPixmapItem>

#include <QGraphicsItem>

#include <QKeyEvent>

#include <QDebug>

#include <QMediaPlayer>

#include <QAudioOutput>

//#include "gameloop.h"

//#include "enemy.h"

//#include "bullet.h"

#include "score.h"

class **Enemy**;

class **Bullet**;

class **GameLoop**;

class **Score**;

class **Player** : public QGraphicsPixmapItem

{

private:

int **x\_pos**;

int **y\_pos**;

int **up**;

int **down**;

int **left**;

int **right**;

int \*\***map**;

QMediaPlayer \***bullet\_sound**;

QGraphicsPixmapItem \*\***grid**;

Enemy \***enemy**;

Score \***score**;

QAudioOutput \***audioOutput**;

public:

void ***keyPressEvent***(QKeyEvent \***event**);

int **ft\_get\_x**();

int **ft\_get\_y**();

void **ft\_set\_x**(int **x**);

void **ft\_set\_y**(int **y**);

void **ft\_spanw**();

void **ft\_create\_map**(int \*\***map**);

void **ft\_update\_map**(int \*\***map**);

int \*\***ft\_player\_map**();

int **ft\_rotate\_left**();

int **ft\_rotate\_right**();

int **ft\_rotate\_up**();

int **ft\_rotate\_down**();

void **ft\_set\_grid**(QGraphicsPixmapItem \*\***grid**);

QGraphicsPixmapItem \*\***ft\_get\_grid**();

**Player**();

void **ft\_set\_enemy**(Enemy \***e**);

Enemy \***ft\_get\_enemy**();

void **ft\_incr\_score**();

Score \***ft\_get\_score**();

void **ft\_set\_score**(Score \***score**);

};

#endif // PLAYER\_H

Player.cpp

#include "player.h"

#include "gameloop.h"

#include "bullet.h"

#include "score.h"

#include <QGraphicsScene>

#include <QDebug>

#include <QSoundEffect>

Player::**Player**()

{

this->up = 0;

this->down = 1;

this->left = 0;

this->right = 0;

this->bullet\_sound = new QMediaPlayer();

this->bullet\_sound->setSource(QUrl("qrc:/sound/gun-gunshot-01.mp3"));

}

void Player::***keyPressEvent***(QKeyEvent \***event**)

{

if (event->key() == Qt::Key\_Left)

{

this->setPixmap(QPixmap(":/pics/player\_l.png"));

this->left = 1;

this->right = 0;

this->up = 0;

this->down = 0;

if (this->map[this->x\_pos][this->y\_pos - 1] == 0)

{

this->y\_pos--;

setPos(x() - 64, y());

}

}

if (event->key() == Qt::Key\_Right)

{

this->setPixmap(QPixmap(":/pics/player\_r.png"));

this->left = 0;

this->right = 1;

this->up = 0;

this->down = 0;

if (this->map[this->x\_pos][this->y\_pos + 1] == 0)

{

setPos(x() + 64, y());

this->y\_pos++;

}

}

if (event->key() == Qt::Key\_Down)

{

this->setPixmap(QPixmap(":/pics/player.png"));

this->left = 0;

this->right = 0;

this->up = 0;

this->down = 1;

if (this->map[this->x\_pos + 1][this->y\_pos] == 0)

{

setPos(x(), y() + 64);

this->x\_pos++;

}

}

if (event->key() == Qt::Key\_Up)

{

this->setPixmap(QPixmap(":/pics/player\_u.png"));

this->left = 0;

this->right = 0;

this->up = 1;

this->down = 0;

if (this->map[this->x\_pos - 1][this->y\_pos] == 0)

{

this->x\_pos--;

setPos(x(), y() - 64);

}

}

if (event->key() == Qt::Key\_Space)

{

int **direct**;

// create a bullet

if (this->up)

direct = 1;

if (this->down)

direct = 2;

if (this->right)

direct = 3;

if (this->left)

direct = 4;

Bullet \***bullet**;

bullet = new Bullet(direct, this->*map*, this, 1);

if (direct == 1 || direct == 2)

bullet->setPos(x() + 30, y());

if (direct == 3)

bullet->setPos(x(), y() + 30);

if (direct == 4)

bullet->setPos(x(), y() + 30);

scene()->addItem(*bullet*);

// play bullet sound

// audioOutput=new QAudioOutput;

// bullet\_sound->setAudioOutput(audioOutput);

// bullet\_sound->setSource(QUrl("qrc:/sound/gun-gunshot-01.mp3"));

// audioOutput->setVolume(50);

// bullet\_sound->play();

}

}

void Player::**ft\_update\_map**(int \*\***map**)

{

int **i**;

int **j**;

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

this->map[i][j] = map[i][j];

j++;

}

i++;

}

}

int Player::**ft\_get\_x**()

{

return (this->x\_pos);

}

int Player::**ft\_get\_y**()

{

return (this->y\_pos);

}

void Player::**ft\_set\_x**(int **x**)

{

this->x\_pos = x;

}

void Player::**ft\_set\_y**(int **y**)

{

this->y\_pos = y;

}

void Player::**ft\_spanw**()

{

int **i**;

int **j**;

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

while (j < MAP\_SIZE)

{

if (this->map[i][j] == 0)

{

this->x\_pos = i;

this->y\_pos = j;

return ;

}

j++;

}

i++;

}

}

void Player::**ft\_create\_map**(int \*\***map**)

{

int **i**;

int **j**;

this->map = new int \*[MAP\_SIZE];

i = 0;

while (i < MAP\_SIZE)

{

j = 0;

this->map[i] = new int [MAP\_SIZE];

while (j < MAP\_SIZE)

{

this->map[i][j] = map[i][j];

j++;

}

i++;

}

}

int \*\*Player::**ft\_player\_map**()

{

return (this->map);

}

void Player::**ft\_set\_grid**(QGraphicsPixmapItem \*\***grid**)

{

this->grid = grid;

}

QGraphicsPixmapItem \*\*Player::**ft\_get\_grid**()

{

return (this->grid);

}

void Player::**ft\_set\_enemy**(Enemy \***e**)

{

this->enemy = e;

}

Enemy \*Player::**ft\_get\_enemy**()

{

return (this->enemy);

}

Score \*Player::**ft\_get\_score**()

{

return (this->score);

}

void Player::**ft\_set\_score**(Score \***score**)

{

this->score = score;

}

Score.h

#ifndef SCORE\_H

#define **SCORE\_H**

#include <QGraphicsItem>

#include <QGraphicsTextItem>

class **Score** : public QGraphicsTextItem

{

public:

**Score**(QGraphicsItem \***parent** = 0);

void **ft\_increase**(int **flag**);

private:

int **score**;

};

#endif // SCORE\_H

Score.cpp

#include "score.h"

#include "enemy.h"

#include "player.h"

#include "gameloop.h"

#include "bullet.h"

#include <QFont>

class **Enemy**;

class **GameLoop**;

class **Bullet**;

class **Player**;

Score::**Score**(QGraphicsItem \***parent**) : QGraphicsTextItem(*parent*)

{

// set the score

this->score = 0;

// draw text

setPlainText("SCORE: " + QString::number(this->score) + "\nSPACEBAR TO SHOOT\nARROWS TO MOVE\n");

setDefaultTextColor(QColor("white"));

setFont(QFont("Helvetica", 15));

}

void Score::**ft\_increase**(int **flag**)

{

if (flag == 1)

{

this->score += 10;

setPlainText("SCORE: " + QString::number(this->score) + "\nSPACEBAR TO SHOOT\nARROWS TO MOVE\n");

}

if (flag == 2)

{

setPlainText("YOU LOSE\nYour score = "+QString::number(this->score));

setFont(QFont("Helvetica", 60));

setPos(300, 400);

}

}

Main.cpp

#include <QApplication>

#include "gameloop.h"

#include "gamemenu.h"

int **main**(int **argc**, char \***argv**[])

{

QApplication **a**(*argc*, *argv*);

//GameLoop \*gl;

//gl = new GameLoop();

gameMenu \***m** = new gameMenu();

return a.exec();

}