DOCUMENTATIE

Fighting Cars

Maxim Daniel-Gabriel

Mocanu Alin

Mocanu Sorin

Serbana George-Theodor

Profesor: Stefanescu Alin

CUPRINS

Aceasta constă din următoarele părți:

Descriere in limbaj natural a proiectului

Proiectul se constituie intr-un joc. Acesta incepe, desigur, cu inceputul: doua persoane intra in joc se conecteaza unul cu celalalt, fiind pe dispositive diferite (laptop/personal computer). Este de tipul “care pe care”.

Fiecare jucator are o masina pe care o controleaza, miscandu-se in sus, jos, stanga, dreapta. Mai are optiunea de a lansa gloante. Scopul final este de a il nimeri pe adversar cu gloantele respective. Astfel, la reusita, i se scade un punct de viata celui lovit. Totusi, nu este foarte usor, deoarece cand primul jucator il tinteste pe al doilea, pentru a il invinge, si celalalt il poate tinti. Este nevoie de mare atentie.

Pe parcursul jocului sunt adaugate anumite functionalitati: apar cadouri! Inimioara iti adauga o viata, ceasul iti mareste viteza de deplasare, iar bilele negre te ajuta sa ii scazi viata adversarului cu 2 puncte, nu cu 1 cum este normal.

Jocul se termina cand unul dintre jucatori nu mai are puncte de viata.

Cerinte in limbaj natural

- Doua persoane sa se conecteze una cu cealalta, una ca si server, cealalta ca si guest, si sa porneasca jocul.
- Prin apasarea de taste, se poate modifica pozitia masinii in sus, jos, stanga, dreapta
- Prin space se trage cu glont catre adversar si il va afecta, scazandu-i viata
- Pe parcursul jocului apar diverse cadouri care le va oferi masinilor mai multe vieti, viteza mai mare si gloante pentru a-l afecta mai puternic pe adversar cand il loveste. Masina trebuie sa se plaseze pe coordonatele bonusurilor pentru a si le insusi.
- Jocul se termina cand unul dintre jucatori nu mai are nicio viata, nicio inimioara.

Scheme in Z

Codul asupra caruia sunt create schemele:

```
18  void PacketManager::setParent(QObject* parent)
19  {
20      mParent = parent;
21  }
```

SetParent cu parametrii [mParent] [parent]

SetParent

Player Player' parent1?:parent parent2!:mParent
parent1? ϵ parent_get\dom parent_set parent_set'=parent_set ∪ $\{parent1 ? \rightarrow$ parent2!$\}$ parent_get'=parent_get

Player cu parametrii [Parent] [mParent]

Player

parent_get: $\mathbb{P}$ parent parent_set: Parent $\nrightarrow$ mParent
dom parent_set ⊆ parent_get

Diagrama UML cazuri de utilizare

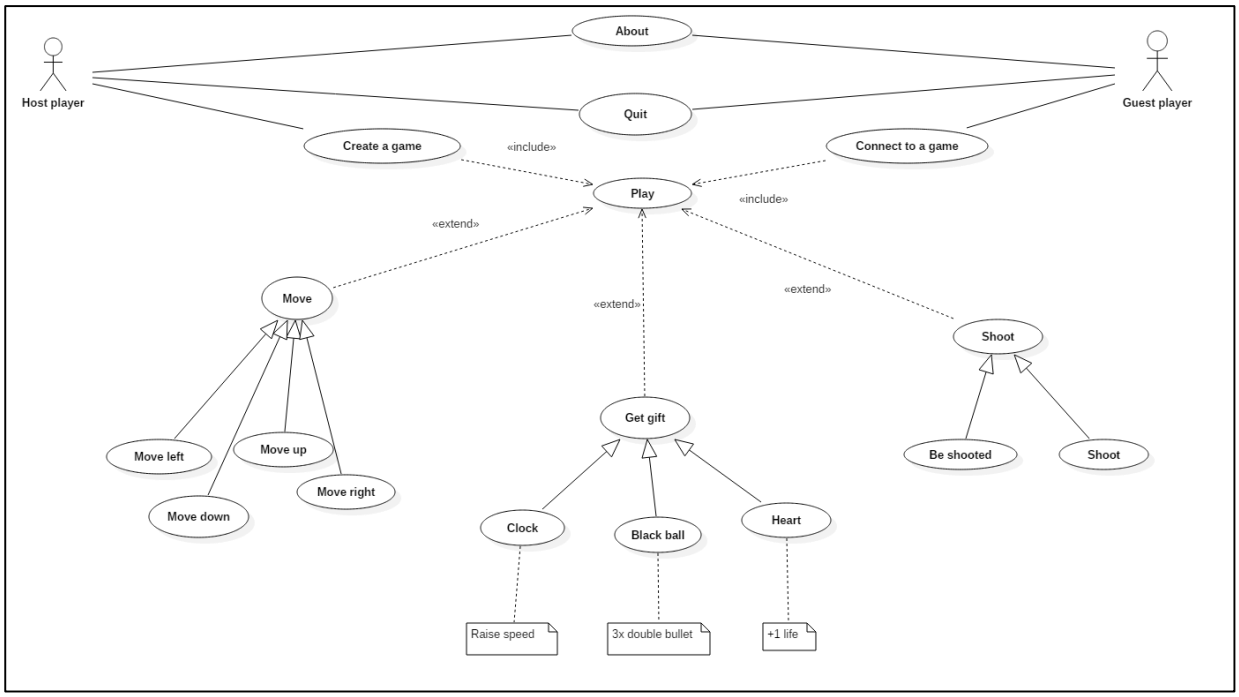


Diagrama UML de stari

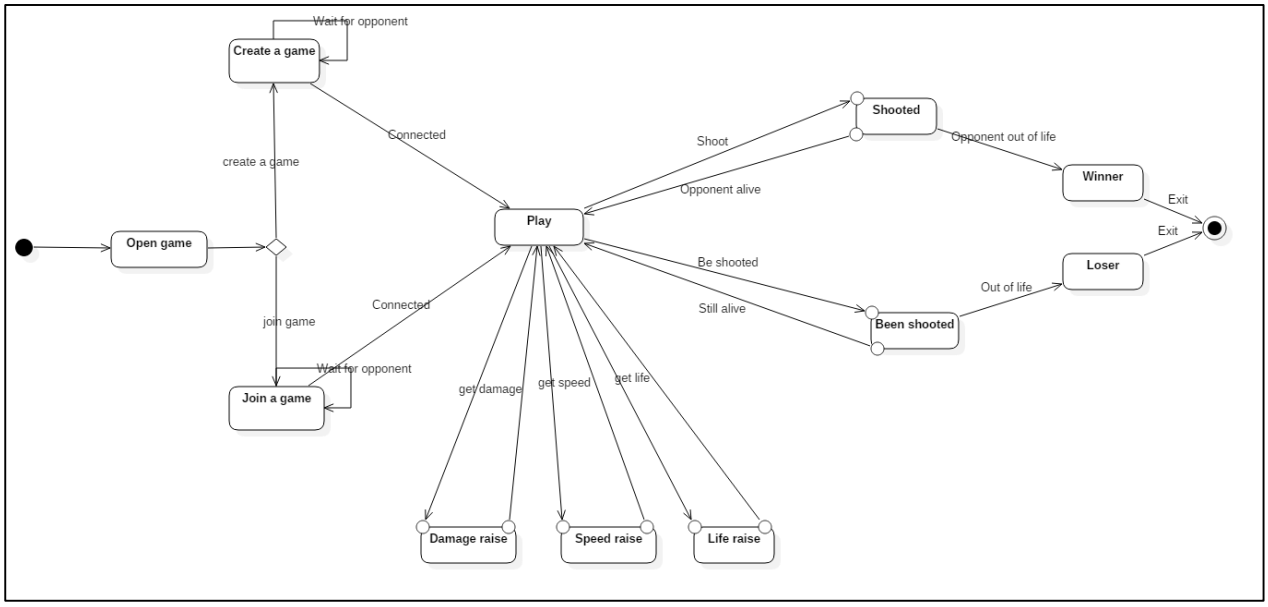


Diagrama UML de clase

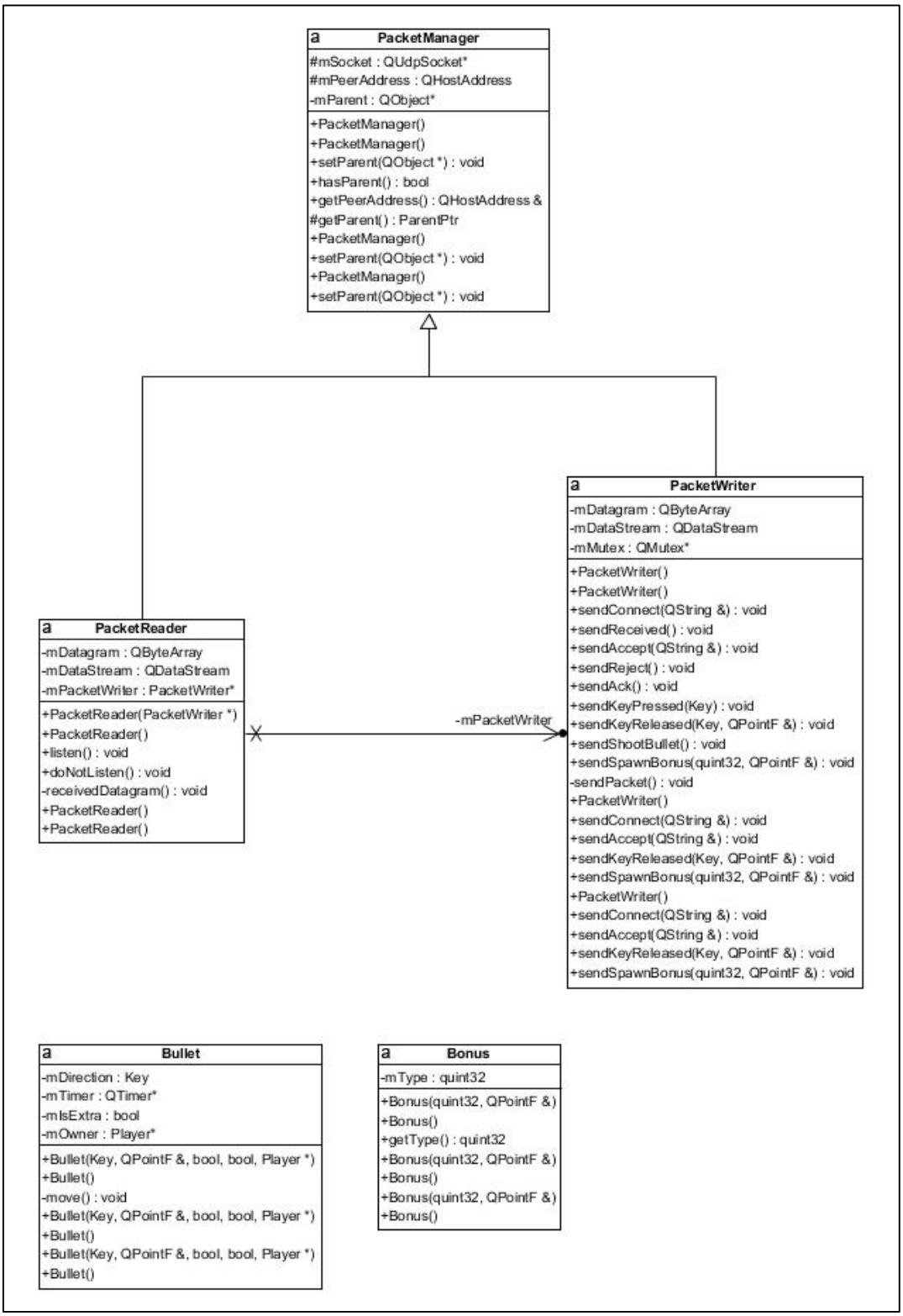
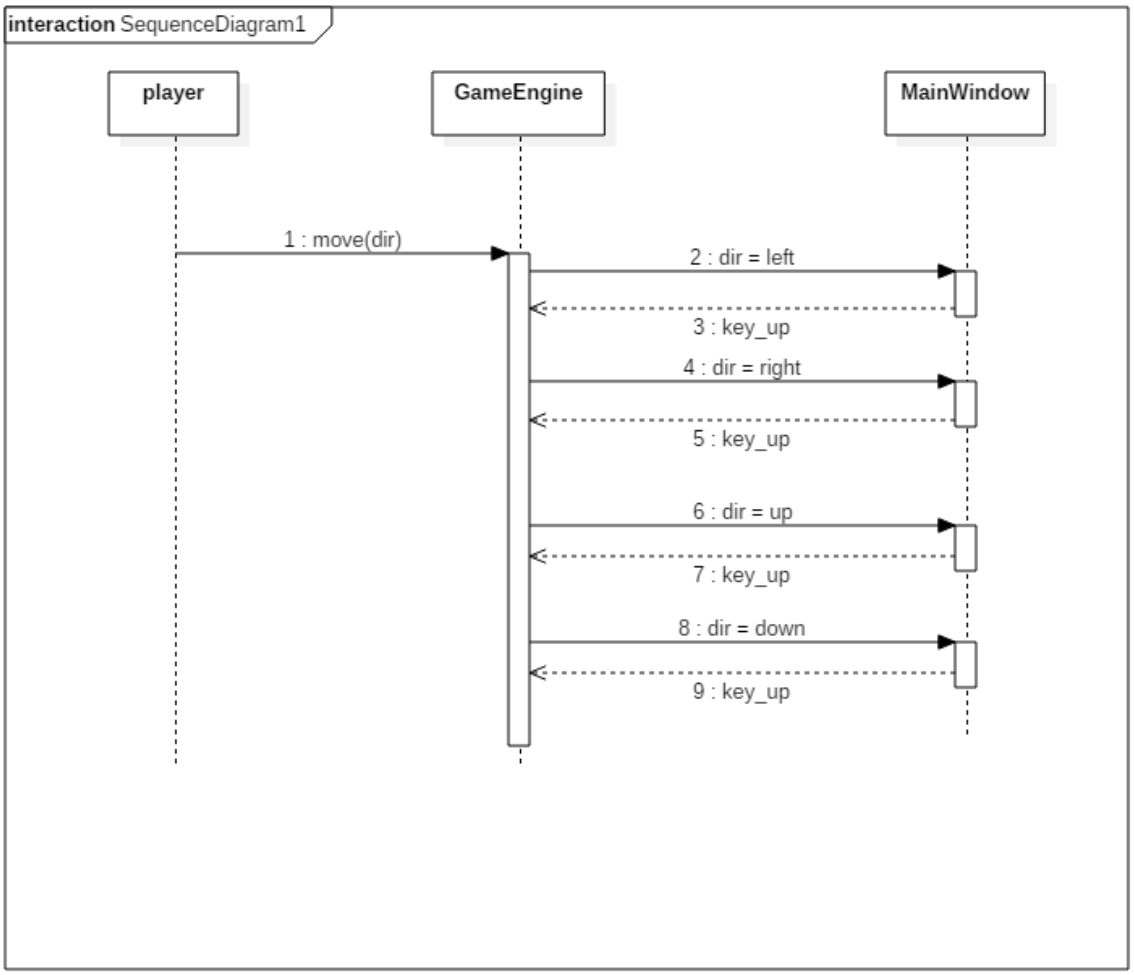


Diagrama UML de secvente



Teste

Am efectuat 4 teste pe diferite clase din program. Am creat clasa Test, cu header si cu sursa, si acolo am implementat testele. Acestea se refera la diverse clase din proiect.

Header-ul este urmatorul:

test.h — testRandomGenerator(): void

```
#ifndef TEST_H
#define TEST_H
#include <QObject>
#include <QtTest/QTest>
#include "Network/PacketManager.h"
#include "Network/PacketReader.h"
#include "GameEngine/GameEngine.h"

class Test:
        public QObject
{
public:
    Q_OBJECT
    Test();
private slots:
    void testPacketManagerSetParent();
    void testPacketReaderStartListening();
    void testPacketReaderStopListening();
    void testRandomGenerator();
};

#endif // TEST_H

```

testPacketManagerSetParent() testeaza codul urmator din clasa PacketManager.

```
void PacketManager::setParent(QObject* parent)
{
    mParent = parent;
}
```

testPacketReaderStartListening() testeaza codul urmator din clasa PacketReader.

```
void PacketReader::listen()
{
    mSocket->bind(QHostAddress::Any, PEER_PORT);
}
```

testPacketReaderStopListening() testeaza codul urmator din clasa PacketReader.

```
void PacketReader::doNotListen()
{
    mSocket->disconnectFromHost();
}
```

testRandomGenerator testeaza codul urmator din clasa GameEngine.

```
quint32 GameEngine::getRandomBetween(quint32 min, quint32 max) const
{
    return qrand() % ((max + 1) - min) + min;
}
```

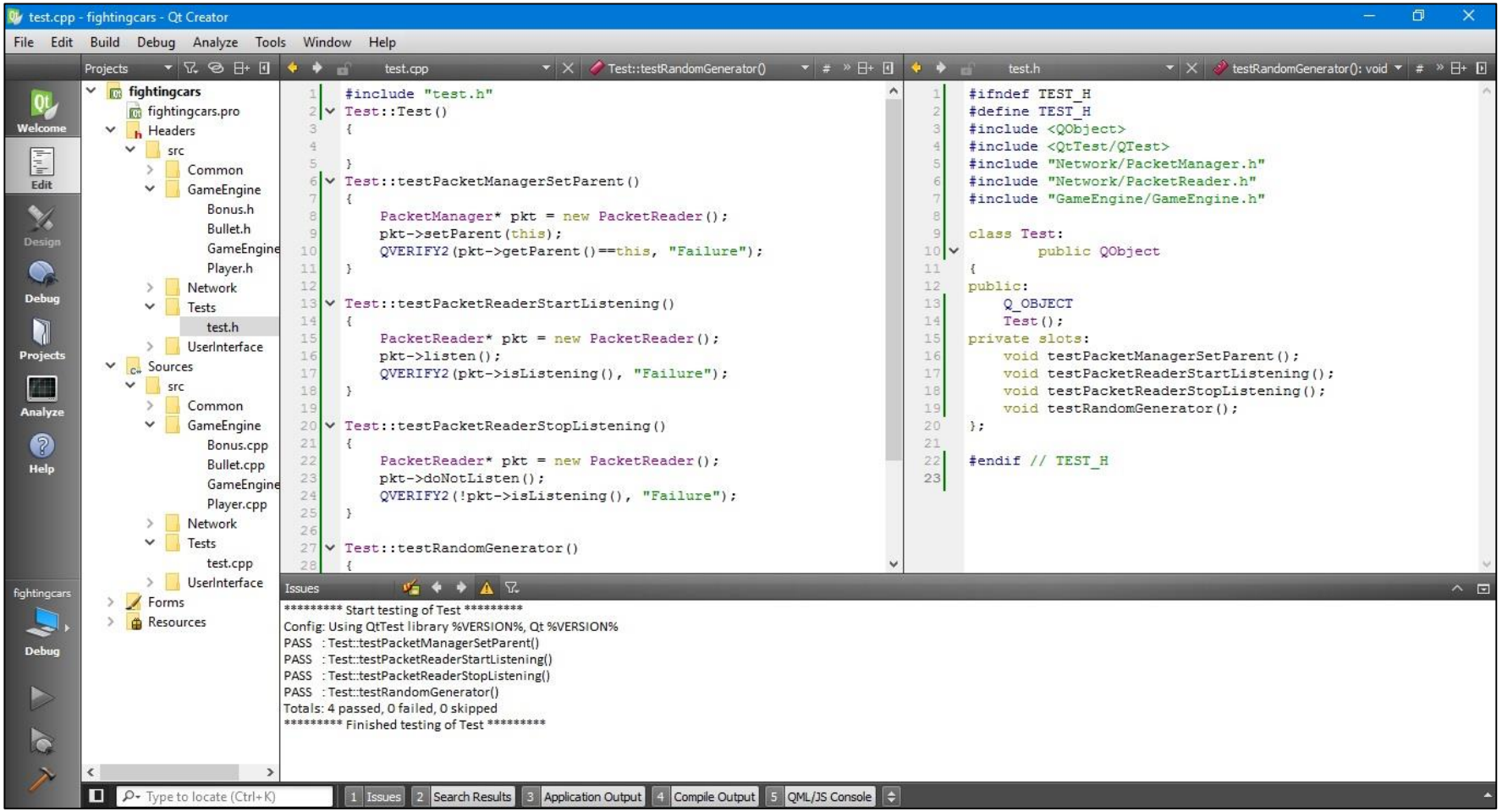
Codul sursa cu implementarile acestor teste este urmatorul:

test.cpp — Test::testRandomGenerator()

```
#include "test.h"
Test::Test()
{

}
Test::testPacketManagerSetParent()
{
    PacketManager* pkt = new PacketReader();
    pkt->setParent(this);
    QVERIFY2(pkt->getParent()==this, "Failure");
}

Test::testPacketReaderStartListening()
{
    PacketReader* pkt = new PacketReader();
    pkt->listen();
    QVERIFY2(pkt->isListening(), "Failure");
}

Test::testPacketReaderStopListening()
{
    PacketReader* pkt = new PacketReader();
    pkt->doNotListen();
    QVERIFY2(!pkt->isListening(), "Failure");
}

Test::testRandomGenerator()
{
    quint32 minim=1,maxim=100;
    GameEngine* ge = new GameEngine();
    quint32 value=ge->getRandomBetween(minim,maxim);
    QVERIFY2(1<=value && value<=100, "Failure");
}

```

Rezultatul acestei testari este urmatorul:

```
********* Start testing of Test *********
Config: Using QtTest library %VERSION%, Qt %VERSION%
PASS   : Test::testPacketManagerSetParent()
PASS   : Test::testPacketReaderStartListening()
PASS   : Test::testPacketReaderStopListening()
PASS   : Test::testRandomGenerator()
Totals: 4 passed, 0 failed, 0 skipped
********* Finished testing of Test *********
```


test.cpp - fightingcars - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
fightingcars
fightingcars.pro
Headers
src
Common
GameEngine
Bonus.h
Bullet.h
GameEngine
Player.h
Network
Tests
test.h
UserInterface
Sources
src
Common
GameEngine
Bonus.cpp
Bullet.cpp
GameEngine
Player.cpp
Network
Tests
test.cpp
UserInterface
Forms
Resources
test.cpp
Test::testRandomGenerator()
#include "test.h"
Test::Test()
{
}
Test::testPacketManagerSetParent()
{
PacketManager* pkt = new PacketReader();
pkt->setParent(this);
QVERIFY2(pkt->getParent()==this, "Failure");
}
Test::testPacketReaderStartListening()
{
PacketReader* pkt = new PacketReader();
pkt->listen();
QVERIFY2(pkt->isListening(), "Failure");
}
Test::testPacketReaderStopListening()
{
PacketReader* pkt = new PacketReader();
pkt->doNotListen();
QVERIFY2(!pkt->isListening(), "Failure");
}
Test::testRandomGenerator()
{
test.h
testRandomGenerator(): void
#ifndef TEST_H
#define TEST_H
#include <QObject>
#include <QtTest/QTest>
#include "Network/PacketManager.h"
#include "Network/PacketReader.h"
#include "GameEngine/GameEngine.h"
class Test:
public QObject
{
public:
Q_OBJECT
Test();
private slots:
void testPacketManagerSetParent();
void testPacketReaderStartListening();
void testPacketReaderStopListening();
void testRandomGenerator();
};
#endif // TEST_H
Issues
********* Start testing of Test *********
Config: Using QtTest library %VERSION%, Qt %VERSION%
PASS : Test::testPacketManagerSetParent()
PASS : Test::testPacketReaderStartListening()
PASS : Test::testPacketReaderStopListening()
PASS : Test::testRandomGenerator()
Totals: 4 passed, 0 failed, 0 skipped
********* Finished testing of Test *********
Welcome
Edit
Design
Debug
Projects
Analyze
Help
fightingcars
Debug
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
5 QML/JS Console

Idei pentru viitor

- sunete
- muzica
- testare
- campionat