Snowman battle

Generováno programem Doxygen 1.6.3

Thu Apr 15 13:58:56 2010

Obsah

Kapitola 1

Rejstřík tříd

1.1 Hierarchie tříd

Zde naleznete seznam, vyjadřující vztah dědičnosti tříd. Je seřazen přibližně (ale ne úplně) podle abecedy:

Kapitola 2

Rejstřík tříd

2.1 Seznam tříd

Následující seznam obsahuje především identifikace tříd, ale nacházejí se zde i další netriviální prvky, jako jsou struktury (struct), unie (union) a rozhraní (interface). V seznamu jsou uvedeny jejich stručné popisy:

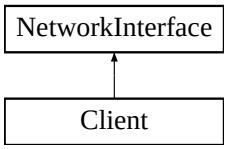
Kapitola 3

Dokumentace tříd

3.1 Dokumentace třídy Client

`#include "client.h"`

Diagram dědičnosti pro třídu Client



Signály

- void sendingHelloPacket ()

Veřejné metody

- Client (QHostAddress address, int port, QString name, NetworkInterface ∗const parent=0)
- virtual ∼Client (void)
- void send (QByteArray ∗message)
- int getNetworkID () const
- void setNetworkID (int networkID)
- void timerEvent (QTimerEvent ∗event)

3.1.1 Detailní popis

Client implementuje clientskou logiku. Je zapouzdren pod vzorem Strategy pod jednotne rozhrani s NetworkInterface a volani ji predava trida Network, ktera ho obsluhuje.

3.1.2 Dokumentace konstruktoru a destruktoru

3.1.2.1 Client::Client (QHostAddress *address*, int *port*, QString *name*, NetworkInterface *const *parent* = 0)

Vytvoreni klienta, ktery se pokusi pripojit na zvolenou adresu a port.

Parametry

address adresa, na ktere naslouchа server

port port, na kterem naslouchа server

3.1.2.2 Client::~Client (void) [virtual]

Destruktor uzavirajici socket

3.1.3 Dokumentace k metodám

3.1.3.1 int Client::getNetworkID () const [virtual]

Vrati ID v siti, ktera mu byla pridelena klientem.

Implementuje NetworkInterface.

3.1.3.2 void Client::send (QByteArray * *message*) [virtual]

Odeslani dat v poli znaku.

Parametry

message pole dat k odeslani

Implementuje NetworkInterface.

3.1.3.3 void Client::sendingHelloPacket () [signal]

Informuje o umyslu odeslat hello packet.

3.1.3.4 void Client::setNetworkID (int *networkID*) [virtual]

Nastavi klientovi jeho networkID

Parametry

networkID pridelovane ID

Implementuje NetworkInterface.

3.1.3.5 void Client::timerEvent (QTimerEvent * *event*)

Nastartuje timer posílají hello packety.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/client.h
- controller/network/client.cpp

3.2 Dokumentace třídy ClientThread

`#include "clientthread.h"`

Signály

- void newMessage (QByteArray ∗message)

Veřejné metody

- ClientThread (QTcpSocket ∗socket)
- ∼ClientThread ()
- void run ()

3.2.1 Detailní popis

Vytvori vlakno, ktere bude poslouchat nova prichozi data. Server: Vytvari pro kazdeho klienta zvlastni vlakno. Client: Vytvori si po pripojeni prave jedno vlakno, kde bude prijimat data od serveru.

3.2.2 Dokumentace konstruktoru a destruktoru

3.2.2.1 ClientThread::ClientThread (QTcpSocket ∗ *socket*)

Vytvori vlakno komunikujici s danym socketem.

Parametry

socket komunikacni socket klienta

3.2.2.2 ClientThread::∼ClientThread ()

Definuje ukonceni vlakna.

3.2.3 Dokumentace k metodám

3.2.3.1 void ClientThread::newMessage (QByteArray ∗ *message*) `[signal]`

Tento signal je vyemitovan vzdy, kdy jsou prijata nova data a predava je ke zpracovani.

Parametry

message pole prijatych dat

3.2.3.2 void ClientThread::run (void)

Definuje cinnost vlakna - pouze spusti smycku udalosti a dale jiz reaguje jen na signaly.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/clientthread.h
- controller/network/clientthread.cpp

3.3 Dokumentace třídy Game

`#include "game.h"`

Signály

- void **frameEnded** (void)
- void **shotCreated** (int shotID, int x, int y)
- void **wPackRemoved** (int wPackID)
- void **playerMoved** (int playerID, int x, int y)
- void **shotMoved** (int shotID, int x, int y)
- void **shotDestroyed** (int shotID)
- void **wPackCreated** (int wPackID, int x, int y, int type)

Veřejné metody

- Game (const int countOfPlayers, const int scoreToWin, QObject ∗const parent=0)
- virtual ∼Game (void)
- void quitGame (void)
- void generateValidCoordinates (const double sizeX, const double sizeY, MapObject ∗const object)
- void addShot (Shot ∗const shot)
- void removeWeaponPackage (WeaponPackage ∗const wPackage)
- void timerEvent (QTimerEvent ∗const event)
- bool isGameRunning (void) const
- int getScoreToWin (void) const
- QMutex ∗ getBigGameMutex (void) const

Chráněné metody

- virtual void run (void)

Friends

- class GameFacade

3.3.1 Detailní popis

Třída reprezentující hru samotnou, je to vlákno které obsluhuje pohyb objektů a kolizí mezi nimi

3.3.2 Dokumentace konstruktoru a destruktoru

3.3.2.1 Game::Game (const int *countOfPlayers*, const int *scoreToWin*, QObject ∗const *parent* = 0) [explicit]

Vytvoří novou hru

Parametry

countOfPlayers počet hráču, kteří budou hrát hru

parent určuje rodičovský objekt

3.3.2.2 Game::∼Game (void) `[virtual]`

Uvolňuje prostředky alokované pro potřeby hry

3.3.3 Dokumentace k metodám

3.3.3.1 void Game::addShot (Shot ∗const *shot*)

Přidá střelu do seznamu střel(bude s ní pohybovat)

Parametry

shot ukazatel na střelu, která bude do seznamu přidána

3.3.3.2 void Game::generateValidCoordinates (const double *sizeX*, const double *sizeY*, MapObject ∗const *object*)

Metoda, která slouží pro vyhledání náhodných souřadnic, do kterých může být umísteň objekt

Parametry

size velikost daného objektu v pixelech

object ukazatel na objekt, který se má umístit

3.3.3.3 QMutex ∗ Game::getBigGameMutex (void) const

Getr pro hlavní mutex

Návratová hodnota

ukazatel mutex, který slouží pro uzamykání hlavního vlákna

3.3.3.4 int Game::getScoreToWin (void) const

Getr pro skóre potřebné na výhru

Návratová hodnota

vrací, kolik je potřeba pro výhru zabít soupeřů

3.3.3.5 bool Game::isGameRunning (void) const

Getr sloužící ke zjištění zde toto má nastavenou řídící proměnou

Návratová hodnota

true, pokud hra běží

3.3.3.6 void Game::quitGame (void)

Ukončí smyčku(vlákno) detekce kolizí

3.3.3.7 void Game::removeWeaponPackage (WeaponPackage ∗const *wPackage*)

Odebere zbraň ze seznamu objektů (poté co jí hráč vezme)

Parametry

wPackage zbraň, jenž bude odebrána

3.3.3.8 void Game::run (void) `[protected, virtual]`

Metoda běhu vlákna, provede nezbytnou inicializaci a spustí smyčku událostí

3.3.3.9 void Game::timerEvent (QTimerEvent ∗const *event*)

Tato metoda se spouští každý frame a provádí deteci kolizí

Parametry

event časovač, který spustil tuhle metodu

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/game/game.h
- controller/game/game.cpp

3.4 Dokumentace třídy GameFacade

`#include "gamefacade.h"`

Veřejné sloty

- void newGame (const int countOfPlayers)
- void endGame (void)
- void pauseGame (void)
- void startMoveNorth (const int playerID)
- void **startMoveWest** (const int playerID)
- void **startMoveSouth** (const int playerID)
- void **startMoveEast** (const int playerID)
- void stopMove (const int playerID)
- void **shot** (const int playerID)
- void **changeWeapon** (const int playerID)
- void activatePlayer (const int playerID)
- void **deactivatePlayer** (const int playerID)

Veřejné metody

- GameFacade (QObject ∗const parent=0)
- virtual ∼GameFacade (void)

3.4.1 Detailní popis

Fasáda, která vytváří rozhraní pro komunikaci s herní logikou

3.4.2 Dokumentace konstruktoru a destruktoru

3.4.2.1 GameFacade::GameFacade (QObject ∗const *parent* = 0) `[explicit]`

Vytvoření fasády, která zastřešuje vnitřní logiku hry

Parametry

parent určuje rodičovský objekt

3.4.2.2 GameFacade::∼GameFacade (void) `[virtual]`

Destruktor ukončí aktuálně běžící hru

3.4.3 Dokumentace k metodám

3.4.3.1 void GameFacade::activatePlayer (const int *playerID*) `[slot]`

Sloty pro aktivování a deaktivování hráče

Parametry

playerID id hráče, jenž se má aktivovat nebo deaktivovat

3.4.3.2 void GameFacade::endGame (void) [slot]

Ukončí aktuální hru

3.4.3.3 void GameFacade::newGame (const int *countOfPlayers*) [slot]

Vytvoří novou hru se zadaným počtem hráčů

Parametry

countOfPlayers počet hráčů, kteří budou hrát

3.4.3.4 void GameFacade::pauseGame (void) [slot]

Pauzne (nebo odpauzne) aktuální hru

3.4.3.5 void GameFacade::startMoveNorth (const int *playerID*) [slot]

Sloty pro zahájení pohybu danným směrem

Parametry

playerID id hráče, který tuto akci má vykonat

3.4.3.6 void GameFacade::stopMove (const int *playerID*) [slot]

Sloty pro přerušení pohybu, střelbu a změnu zbraně

Parametry

playerID id hráče, který tuto akci má vykonat

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/game/gamefacade.h
- controller/game/gamefacade.cpp

3.5 Dokumentace třídy Globals

`#include "globals.h"`

Statické veřejné atributy

- static Window * mainWindow = NULL
- static Network * network = NULL
- static PacketParser * packetParser = NULL
- static PacketCreator * packetCreator = NULL
- static GameFacade * gameFacade = NULL
- static bool isGameRunning = false
- static int players = 0

3.5.1 Detailní popis

Speciální třída, který v sobě udržuje ukazatele instance všech singletonových tříd

3.5.2 Dokumentace k datovým členům

3.5.2.1 GameFacade * Globals::gameFacade = NULL [static]

Instance herní fasády

3.5.2.2 bool Globals::isGameRunning = false [static]

Priznak, zda jiz je hra odstartovana.

3.5.2.3 Window * Globals::mainWindow = NULL [static]

Instance hlavního okna

3.5.2.4 Network * Globals::network = NULL [static]

Instance síťového rozhraní

3.5.2.5 PacketCreator * Globals::packetCreator = NULL [static]

Vytvářeč paketů

3.5.2.6 PacketParser * Globals::packetParser = NULL [static]

Parser na pakety

3.5.2.7 int Globals::players = 0 [static]

Pocet hracu.

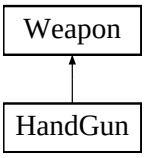
Dokumentace pro tuto třídu byla generována z následujících souborů:

- globals.h
- main.cpp

3.6 Dokumentace třídy HandGun

`#include "handgun.h"`

Diagram dědičnosti pro třídu HandGun



Veřejné metody

- HandGun (Player *const parent)
- void shot (void)
- void refill (void)

3.6.1 Detailní popis

Tato třída reprezentuje pistoli, kterou každý hráč používá jako základní zbraň

3.6.2 Dokumentace konstruktoru a destruktoru

3.6.2.1 HandGun::HandGun (Player *const *parent*) `[explicit]`

Konstruktor vytvoří danému hráči pistoli

Parametry

parent hráč, kterému zbraň patří

3.6.3 Dokumentace k metodám

3.6.3.1 void HandGun::refill (void) `[virtual]`

Implementace abstraktní metody předka (doplnění nábojů)

Implementuje Weapon.

3.6.3.2 void HandGun::shot (void) `[virtual]`

Implementace abstraktní metody předka (výstřel zbraně)

Implementuje Weapon.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/handgun.h
- model/handgun.cpp

3.7 Dokumentace třídy KeyboardHandler

`#include "keyboardhandler.h"`

Signály

- void upMove (void)
- void **downMove** (void)
- void **leftMove** (void)
- void **rightMove** (void)
- void stopMove (void)
- void shot (void)
- void changeWeapon (void)
- void pauseGame (void)

Veřejné metody

- KeyboardHandler (QObject ∗const parent)
- void handleKeyEvent (QKeyEvent ∗const event)

3.7.1 Detailní popis

Tato třída se napojí na grafický prvek a na něm odchytává herní klávesy a podle nich posílá signály

3.7.2 Dokumentace konstruktoru a destruktoru

3.7.2.1 KeyboardHandler::KeyboardHandler (QObject ∗const *parent*)

Konstruktor vytvoří "odchytavač" kláves

Parametry

parent grafický prvek, na který je tento objekt napojen

myID id lokálního hráče

3.7.3 Dokumentace k metodám

3.7.3.1 void KeyboardHandler::changeWeapon (void) `[signal]`

Toto je signál stisknutí talčítka pro změnu zbraně

3.7.3.2 void KeyboardHandler::handleKeyEvent (QKeyEvent ∗const *event*)

Tato metoda rozpozná daný key event a podle toho vyšle odpovídající signál

Parametry

event key event, který se má zpracovat

3.7.3.3 void KeyboardHandler::pauseGame (void) [signal]

Toto je signál stisknutí talčítka pro pozastavení hry

3.7.3.4 void KeyboardHandler::shot (void) [signal]

Toto je signál stisknutí talčítka pro střelbu

3.7.3.5 void KeyboardHandler::stopMove (void) [signal]

Tento signál je vyslán, když je puštěna pohybová klávesa

3.7.3.6 void KeyboardHandler::upMove (void) [signal]

Tyto signály jsou vyslány, když je stisknuta odpovídající směrová klávesa

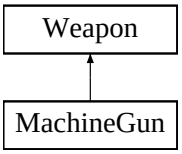
Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/gui/keyboardhandler.h
- controller/gui/keyboardhandler.cpp

3.8 Dokumentace třídy MachineGun

`#include "machinegun.h"`

Diagram dědičnosti pro třídu MachineGun



Veřejné metody

- MachineGun (Player ∗const parent)
- void shot (void)
- void refill (void)
- void timerEvent (QTimerEvent ∗const event)

3.8.1 Detailní popis

Třída reprezentující samopal

3.8.2 Dokumentace konstruktoru a destruktoru

3.8.2.1 MachineGun::MachineGun (Player ∗const *parent*) `[explicit]`

Konstruktor vytvoří danému hráči samopal

Parametry

parent hráč, kterému zbraň patří

3.8.3 Dokumentace k metodám

3.8.3.1 void MachineGun::refill (void) `[virtual]`

Implementace abstraktní metody předka (doplnění nábojů)

Implementuje Weapon.

3.8.3.2 void MachineGun::shot (void) `[virtual]`

Implementace abstraktní metody předka (výstřel zbraně)

Implementuje Weapon.

3.8.3.3 void MachineGun::timerEvent (QTimerEvent ∗const *event*)

Tato metoda slouží k tomu, aby ze samopalu správně vystřelily další střely

Parametry

event časovač, který spustil tuto metodu

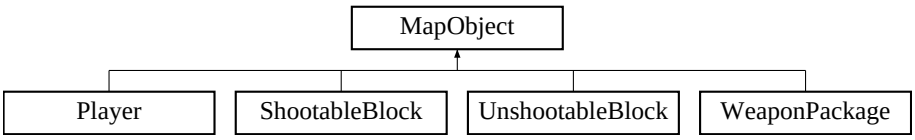
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/machinegun.h
- model/machinegun.cpp

3.9 Dokumentace třídy MapObject

`#include "mapobject.h"`

Diagram dědičnosti pro třídu MapObject



Veřejné metody

- MapObject (Game *const parent, const double x1=-100.0, const double y1=-100.0, const double x2=-100.0, const double y2=-100.0)
- virtual bool interactPlayer (Player *const player)=0
- virtual bool interactShot (Shot *const shot)=0
- double getX1 (void) const
- double **getY1** (void) const
- double **getX2** (void) const
- double **getY2** (void) const

Chráněné atributy

- Game * parentGame
- double x1
- double **y1**
- double x2
- double **y2**

Friends

- class Game

3.9.1 Detailní popis

Toto je abstraktní třída, která reprezentuje objekt na herní ploše

3.9.2 Dokumentace konstruktoru a destruktoru

3.9.2.1 MapObject::MapObject (Game *const *parent*, const double *x1* = `-100.0`, const double *y1* = `-100.0`, const double *x2* = `-100.0`, const double *y2* = `-100.0`)

Vytvoří objekt na mapě na konkrétních souřadnicích

Parametry

parent hra v rámci níž je tento objekt vytvořen

x1 x-ová souřadnice levého horního rohu objektu

y1 y-ová souřadnice levého horního rohu objektu

x2 x-ová souřadnice pravého dolního rohu objektu

y2 y-ová souřadnice pravého dolního rohu objektu

3.9.3 Dokumentace k metodám

3.9.3.1 double MapObject::getX1 (void) const

Getry pro jednotlivé souřadnice

3.9.3.2 virtual bool MapObject::interactPlayer (Player ∗const *player*) [pure virtual]

Metoda popisující jak se objekt zachová, když na něj vběhne hráč

Parametry

player hráč, který chce na místo objektu vejít

Návratová hodnota

vrací true, pokud hráč na místo objektu může vstoupit

Implementováno v Player, ShootableBlock, UnshootableBlock a WeaponPackage.

3.9.3.3 virtual bool MapObject::interactShot (Shot ∗const *shot*) [pure virtual]

Metoda popisující jak se objekt zachová, když do něj vletí střela

Parametry

shot střela, která vniká do objektů

Návratová hodnota

vrací true, pokud přes objekt může střela letět

Implementováno v Player, ShootableBlock, UnshootableBlock a WeaponPackage.

3.9.4 Dokumentace k datovým členům

3.9.4.1 Game∗ MapObject::parentGame [protected]

Ukazatel na hru, ke které tento objekt patří

3.9.4.2 double MapObject::x1 [protected]

Souřadnice pro levý horní roh

3.9.4.3 double MapObject::x2 [protected]

Souřadnice pravý dolní roh

Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/mapobject.h
- model/mapobject.cpp

3.10 Dokumentace třídy Network

```
#include "network.h"
```

Veřejné metody

- Network (NetworkInterface ∗strategy, QObject ∗const parent=0)
- void send (QByteArray ∗message)
- int **getNetworkID** () const
- void **setNetworkID** (int networkID)

3.10.1 Detailní popis

Network zaobaluje sitove rozhrani dle navrhoveho vzoru Strategy. Na vyber jsou zakladni dve moznosti - client a server, pricemz obe musi byt potomky tridy NetworkInterface.

3.10.2 Dokumentace konstruktoru a destruktoru

3.10.2.1 Network::Network (NetworkInterface ∗ *strategy*, QObject ∗const *parent* = 0) [explicit]

Vytvoreni sitoveho rozhrani jako server ci client.

Parametry

strategy ukazatel na zvoleny zpusob prace se siti

3.10.3 Dokumentace k metodám

3.10.3.1 void Network::send (QByteArray ∗ *message*)

Odeslani dat v poli znaku.

Parametry

pole znaku k odeslani

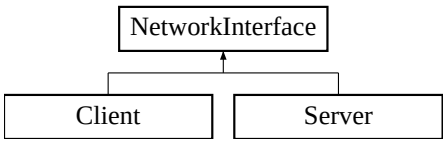
Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/network.h
- controller/network/network.cpp

3.11 Dokumentace třídy NetworkInterface

`#include "networkinterface.h"`

Diagram dědičnosti pro třídu NetworkInterface



Veřejné metody

- **NetworkInterface** (QObject ∗parent=0)
- virtual void send (QByteArray ∗message)=0
- virtual int getNetworkID () const =0
- virtual void setNetworkID (int networkID)=0

3.11.1 Detailní popis

Definuje spolecne rozhrani pro clienta a server.

3.11.2 Dokumentace k metodám

3.11.2.1 virtual int NetworkInterface::getNetworkID () const [pure virtual]

Vrati hodnotu ID v siti.

Implementováno v Client a Server.

3.11.2.2 virtual void NetworkInterface::send (QByteArray ∗ *message*) [pure virtual]

Odeslani dat v poli znaku.

Parametry

message pole znaku k odeslani

Implementováno v Client a Server.

3.11.2.3 virtual void NetworkInterface::setNetworkID (int *networkID*) [pure virtual]

Nastavi ID v siti.

Implementováno v Client a Server.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/networkinterface.h
- controller/network/networkinterface.cpp

3.12 Dokumentace třídy PacketCreator

`#include "packetcreator.h"`

Veřejné sloty

- QByteArray * assignID (int id)
- void assignName (int id, QString *name)
- void chooseMap (int mapID)
- void startGame (void)
- void startMoveNorth (void)
- void startMoveWest (void)
- void startMoveSouth (void)
- void startMoveEast (void)
- void pressShot (void)
- void pressChangeWeapon (void)
- void stopMove (void)
- void sendHelloPacket (void)
- void sendGameEnginePacket (void)
- void spawnPlayer (int playerID, int x, int y, int direction)
- void spawnWeaponPack (int weaponPackID, int x, int y, int type)
- void killPlayer (int playerID)
- void despawnWeaponPack (int weaponPackID)
- void changeWeapon (int playerID, int weaponType, int restOfAmmo)
- void createShot (int shotID, int x, int y)
- void movePlayer (int playerID, int x, int y)
- void moveShot (int shotID, int x, int y)
- void destroyShot (int shotID)
- void playerShots (int playerID)
- void incrementScore (int playerID)
- void quitGame (void)
- void pauseGame (void)
- void activatePlayer (int playerID)
- void deactivatePlayer (int playerID)
- void winPlayer (int playerID)
- void sendChatMessage (int playerID, QString *msg)

Veřejné metody

- PacketCreator (QObject *const parent=0)
- ~PacketCreator (void)

3.12.1 Detailní popis

Třída, která slouží k vytváření paketů a jejich posílání po síti

3.12.2 Dokumentace konstruktoru a destruktoru

3.12.2.1 PacketCreator::PacketCreator (QObject ∗const *parent* = 0) [explicit]

Konstruktor slouží k vytvoření "vytvářeče" paketů

Parametry

parent objekt, který je rodičem

3.12.2.2 PacketCreator::∼PacketCreator (void)

Destruror ruší alokované prostředky

3.12.3 Dokumentace k metodám

3.12.3.1 void PacketCreator::activatePlayer (int *playerID*) [slot]

Slot pro aktivování hráče

Parametry

playerID id hráče, který se má aktivovat

3.12.3.2 QByteArray ∗ PacketCreator::assignID (int *id*) [slot]

Slot pro přidělení packetu

Parametry

id id, které se stanici přidělí

3.12.3.3 void PacketCreator::assignName (int *id*, QString ∗ *name*) [slot]

Slot představovacího paketu

Parametry

id id hráče, kterému patří jméno
name ukazatel na jméno

3.12.3.4 void PacketCreator::changeWeapon (int *playerID*, int *weaponType*, int *restOfAmmo*) [slot]

Slot, do kterého se posílá informace o změny zbraně

Parametry

playerID id hráče, který si mění zbraň
weaponType typ zbraně, kterou si bere
restOfAmmo počet zbývajících nábojů v této zbrani

3.12.3.5 void PacketCreator::chooseMap (int *mapID*) [slot]

Slot pro výběr mapy (pro nás nezajímavé, protože máme jen jednu mapu)

Parametry

mapID překvapivě id mapy

3.12.3.6 void PacketCreator::createShot (int *shotID*, int *x*, int *y*) [slot]

Slot, do kterého se posílá informace o vytvoření střely

Parametry

shotID,id této střely

x x-ová souřadnice střely (její střed)

y y-ová souřadnice střely (její střed)

3.12.3.7 void PacketCreator::deactivatePlayer (int *playerID*) [slot]

Slot pro aktivování hráče

Parametry

playerID id hráče, který se má deaktivovat

3.12.3.8 void PacketCreator::despawnWeaponPack (int *weaponPackID*) [slot]

Slot, do kterého se posílá informace o zmizení zbraně

Parametry

weaponPackID id zbraně, která zmizí

3.12.3.9 void PacketCreator::destroyShot (int *shotID*) [slot]

Slot, do kterého se posílá informace o zničení střely

Parametry

shotID id střely, který se má zničit

3.12.3.10 void PacketCreator::incrementScore (int *playerID*) [slot]

Slot, do kterého se posílá informace o zvýšení skóre hráče

Parametry

playerID id hráče, kterému se zvýšilo skóre

3.12.3.11 void PacketCreator::killPlayer (int *playerID*) [slot]

Slot, do kterého se posílá informace o zabití hráče

Parametry

playerID id hráče, který byl zabit

3.12.3.12 void PacketCreator::movePlayer (int *playerID*, int *x*, int *y*) [slot]

Slot, do kterého se posílá informace o pohybu hráče

Parametry

playerID id hráče, který se bude přesouvat

x x-ová souřadnice místa, kam se má hráč přesunout (levý horní roh)

y y-ová souřadnice místa, kam se má hráč přesunout (levý horní roh)

3.12.3.13 void PacketCreator::moveShot (int *shotID*, int *x*, int *y*) [slot]

Slot, do kterého se posílá informace o pohybu střely

Parametry

shotID id střely, který se bude přesouvat

x x-ová souřadnice místa, kam se má střela přesunout (její střed)

y y-ová souřadnice místa, kam se má střela přesunout (její střed)

3.12.3.14 void PacketCreator::pauseGame (void) [slot]

Slot pro zapauzování hry

3.12.3.15 void PacketCreator::playerShots (int *playerID*) [slot]

Slot, do kterého se posílá informace o výstřelu hráče

Parametry

playerID id hráče, který střílí

3.12.3.16 void PacketCreator::pressChangeWeapon (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro změnu zbraně

3.12.3.17 void PacketCreator::pressShot (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro střelbu

3.12.3.18 void PacketCreator::quitGame (void) [slot]

Slot pro ukončení hry

3.12.3.19 void PacketCreator::sendChatMessage (int *playerID*, QString * *msg*) [slot]

Slot, kterým se odesílá chatová zpráva

Parametry

playerID id hráče, jenž posílá zprávu

msg ukazatel na zprávu, která se odešle

3.12.3.20 void PacketCreator::sendGameEnginePacket (void) [slot]

Slot, do kterého se posílá informace o tom, že je možné odeslat paket enginu hry

3.12.3.21 void PacketCreator::sendHelloPacket (void) [slot]

Slot pro odeslání hello paketu

3.12.3.22 void PacketCreator::spawnPlayer (int *playerID*, int *x*, int *y*, int *direction*) [slot]

Slot, do kterého se posílá informace o naspawnování hráče

Parametry

playerID id hráče, který se má spawnout

x x-ová souřadnice místa spawnutí (levý horní roh)

y y-ová souřadnice místa spawnutí (levý horní roh)

direction směr, kterým bude hráč otočený

3.12.3.23 void PacketCreator::spawnWeaponPack (int *weaponPackID*, int *x*, int *y*, int *type*) [slot]

Slot, do kterého se posílá informace o objevení zbraně na herní ploše

Parametry

weaponPackID id zbraně, která se objeví

x x-ová souřadnice místa objevení (levý horní roh)

y y-ová souřadnice místa objevení (levý horní roh)

type typ zbraně, která se objeví

3.12.3.24 void PacketCreator::startGame (void) [slot]

Slot startu hry

3.12.3.25 void PacketCreator::startMoveEast (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro pohyb doprava

3.12.3.26 void PacketCreator::startMoveNorth (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro pohyb nahoru

3.12.3.27 void PacketCreator::startMoveSouth (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro pohyb dolů

3.12.3.28 void PacketCreator::startMoveWest (void) [slot]

Slot, do kterého se posílá, že hráč stisknul klávesu pro pohyb doleva

3.12.3.29 void PacketCreator::stopMove (void) [slot]

Slot, do kterého se posílá, že hráč pustil klávesu pro pohyb

3.12.3.30 void PacketCreator::winPlayer (int *playerID*) [slot]

Slot pro výhru

Parametry

playerID id hráče, který vyhrál

Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/packetcreator.h
- controller/network/packetcreator.cpp

3.13 Dokumentace třídy PacketParser

`#include "packetparser.h"`

Veřejné sloty

- void parseAll (QByteArray *packets)

Signály

- void nameAssigned (int id, QString *name)
- void mapChoosed (int mapID)
- void gameStarted (void)
- void norhtKeyPressed (int playerID)
- void westKeyPressed (int playerID)
- void southKeyPressed (int playerID)
- void eastKeyPressed (int playerID)
- void shotKeyPressed (int playerID)
- void changeKeyPressed (int playerID)
- void moveKeyReleased (int playerID)
- void helloPacketAccepted (int senderID)
- void playerSpawned (int playerID, int x, int y, int direction)
- void weaponPackSpawned (int weaponPackID, int x, int y, int type)
- void playerKilled (int playerID)
- void weaponPackDespawned (int weaponPackID)
- void weaponChanged (int playerID, int weaponType, int restOfAmmo)
- void shotCreated (int shotID, int x, int y)
- void playerMoved (int playerID, int x, int y)
- void shotMoved (int shotID, int x, int y)
- void shotDestroyed (int shotID)
- void playerShoted (int playerID)
- void playersScoreIncremented (int playerID)
- void gameQuited (void)
- void gamePaused (void)
- void playerActivated (int playerID)
- void playerDeactivated (int playerID)
- void playerWon (int playerID)
- void chatMessageRecieved (int playerID, QString *msg)

Veřejné metody

- PacketParser (QObject *const parent=0)
- ~PacketParser (void)

3.13.1 Detailní popis

Tato třída slouží pro parsování paketů přicházejících ze sítě

3.13.2 Dokumentace konstruktoru a destruktoru

3.13.2.1 PacketParser::PacketParser (QObject ∗const *parent* = 0) [explicit]

Konstrutor vytvoří parser

Parametry

parent objekt, který je rodičem

3.13.2.2 PacketParser::∼PacketParser (void)

Destruktor zruší parser

3.13.3 Dokumentace k metodám

3.13.3.1 void PacketParser::changeKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu změn zbraně

Parametry

playerID id hráče, jenž klávesu stiskl

3.13.3.2 void PacketParser::chatMessageRecieved (int *playerID*, QString ∗ *msg*) [signal]

Signál přijetí chatové zprávy

Parametry

playerID id hráče, jenž poslal zprávu

msg ukazatel na zprávu, která přišla

3.13.3.3 void PacketParser::eastKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu pohybu doprava

Parametry

playerID id hráče, jenž klávesu stiskl

3.13.3.4 void PacketParser::gamePaused (void) [signal]

Signál zapauzování hry

3.13.3.5 void PacketParser::gameQuited (void) [signal]

Signál ukončení hry

3.13.3.6 void PacketParser::gameStarted (void) [signal]

Signál startu hry

3.13.3.7 void PacketParser::helloPacketAccepted (int *senderID*) [signal]

Signál přijatého hello packetu

Parametry

senderID id hráče, který paket poslal

3.13.3.8 void PacketParser::mapChoosed (int *mapID*) [signal]

Signál výběru mapy (pro nás nezajímavé, protože máme jen jednu mapu)

Parametry

mapID překvapivě id mapy

3.13.3.9 void PacketParser::moveKeyReleased (int *playerID*) [signal]

Signál, že hráč pustil klávesu pohybu

Parametry

playerID id hráče, jenž klávesu pustil

3.13.3.10 void PacketParser::nameAssigned (int *id*, QString * *name*) [signal]

Signál přidělení id

Parametry

id přidělené id Signál informující o tom, že danému id je přiděleno jméno (představovací paket)

id tomuto id se přidělí jméno

name ukazatel na jméno

3.13.3.11 void PacketParser::norhtKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu pohybu nahoru

Parametry

playerID id hráče, jenž klávesu stiskl

3.13.3.12 void PacketParser::parseAll (QByteArray * *packets*) [slot]

Zpracuje všechny pakety v zadanám poli bytů

Parametry

packets pole bytů, které se má naparsovat

3.13.3.13 void PacketParser::playerActivated (int *playerID*) [signal]

Signál aktivování hráče

Parametry

playerID id hráče, který se má aktivovat

3.13.3.14 void PacketParser::playerDeactivated (int *playerID*) [signal]

Signál deaktivování hráče

Parametry

playerID id hráče, který se má deaktivovat

3.13.3.15 void PacketParser::playerKilled (int *playerID*) [signal]

Signál zabití hráče

Parametry

playerID id hráče, který byl zabit

3.13.3.16 void PacketParser::playerMoved (int *playerID*, int *x*, int *y*) [signal]

Signál pohybu hráče

Parametry

playerID id hráče, který se bude přesouvat

x x-ová souřadnice místa, kam se má hráč přesunout (levý horní roh)

y y-ová souřadnice místa, kam se má hráč přesunout (levý horní roh)

3.13.3.17 void PacketParser::playerShoted (int *playerID*) [signal]

Signál výstřelu hráče (pro odečtení nábojů)

Parametry

playerID id hráče, který vystřelil

3.13.3.18 void PacketParser::playerSpawned (int *playerID*, int *x*, int *y*, int *direction*) [signal]

Signál naspawnování hráče

Parametry

playerID id hráče, který se má spawnout

x x-ová souřadnice místa spawnutí (levý horní roh)

y y-ová souřadnice místa spawnutí (levý horní roh)

direction směr, kterým bude hráč otočený

3.13.3.19 void PacketParser::playersScoreIncremented (int *playerID*) [signal]

Signál zvýšení skóre hráče

Parametry

playerID id hráče, kterému se zvýšilo skóre

3.13.3.20 void PacketParser::playerWon (int *playerID*) [signal]

Signál vítězství hráče

Parametry

playerID id hráče, který vyhrál

3.13.3.21 void PacketParser::shotCreated (int *shotID*, int *x*, int *y*) [signal]

Signál vytvoření střely

Parametry

shotID,id této střely

x x-ová souřadnice střely (její střed)

y y-ová souřadnice střely (její střed)

3.13.3.22 void PacketParser::shotDestroyed (int *shotID*) [signal]

Signál zničení střely

Parametry

shotID id střely, který se má zničit

3.13.3.23 void PacketParser::shotKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu střelby

Parametry

playerID id hráče, jenž klávesu stiskl

3.13.3.24 void PacketParser::shotMoved (int *shotID*, int *x*, int *y*) [signal]

Signál pohybu střely

Parametry

shotID id střely, který se bude přesouvat

x x-ová souřadnice místa, kam se má střela přesunout (její střed)

y y-ová souřadnice místa, kam se má střela přesunout (její střed)

3.13.3.25 void PacketParser::southKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu pohybu dolů

Parametry

playerID id hráče, jenž klávesu stiskl

3.13.3.26 void PacketParser::weaponChanged (int *playerID*, int *weaponType*, int *restOfAmmo*) [signal]

Signál změny zbraně

Parametry

playerID id hráče, který si mění zbraň

weaponType typ zbraně, kterou si bere

restOfAmmo počet zbývajících nábojů v této zbrani

3.13.3.27 void PacketParser::weaponPackDespawned (int *weaponPackID*) [signal]

Signál zmizení zbraně

Parametry

weaponPackID id zbraně, která zmizí

3.13.3.28 void PacketParser::weaponPackSpawned (int *weaponPackID*, int *x*, int *y*, int *type*) [signal]

Signál objevení zbraně na herní ploše

Parametry

weaponPackID id zbraně, která se objeví

x x-ová souřadnice místa objevení (levý horní roh)

y y-ová souřadnice místa objevení (levý horní roh)

type typ zbraně, která se objeví

3.13.3.29 void PacketParser::westKeyPressed (int *playerID*) [signal]

Signál, že hráč stisknul klávesu pohybu doleva

Parametry

playerID id hráče, jenž klávesu stiskl

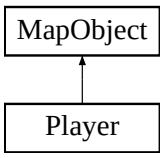
Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/packetparser.h
- controller/network/packetparser.cpp

3.14 Dokumentace třídy Player

`#include "player.h"`

Diagram dědičnosti pro třídu Player



Signály

- void **playerKilled** (int playerID)
- void **playerSpawned** (int playerID, int x, int y, int direction)
- void **playerShoted** (int playerID)
- void **weaponChanged** (int playerID, int weapon, int restOfAmmo)
- void **scoreIncremented** (int playerID)
- void **playerWon** (int playerID)
- void **playerActivated** (int playerID)
- void **playerDeactivated** (int playerID)

Veřejné metody

- Player (Game *const parent, const int id)
- bool interactPlayer (Player *const player)
- bool interactShot (Shot *const shot)
- void respawn (void)
- void tryMove (void)
- void backMove (void)
- void shot (void)
- void changeWeapon (void)
- void timerEvent (QTimerEvent *const event)
- void setActualWeapon (const int actualWeapon)
- void incrementScore (void)
- int getID (void) const
- Directions getDirection (void) const
- void **setDirection** (const Directions direction)
- bool isMoving (void) const
- void **setMoving** (const bool moving)
- bool isShoting (void) const
- void **setShoting** (const bool shoting)
- bool isSpawned (void) const
- bool isActive (void) const
- void **setActive** (const bool active)
- Weapon *const * getInventory (void) const
- Game * getParentGame (void) const

3.14.1 Detailní popis

Tato třída reprezentuje hráče v herní logice

3.14.2 Dokumentace konstruktoru a destruktoru

3.14.2.1 Player::Player (Game ∗const *parent*, const int *id*)

Vytvoření nového hráče (pouze na začátku hry)

Parametry

parent hra, v rámci níž je hráč vytvořen

3.14.3 Dokumentace k metodám

3.14.3.1 void Player::backMove (void)

Pokud se hráč na dané místo nemůže přesunout kvůli kolizi, tak jej tato metoda vrátí zpět

3.14.3.2 void Player::changeWeapon (void)

Změní aktuální zbraň hráče na další, která má náboje

3.14.3.3 Directions Player::getDirection (void) const

Getr a setr pro směr hráče

3.14.3.4 int Player::getID (void) const

Getr pro id

3.14.3.5 Weapon ∗const ∗ Player::getInventory (void) const

Getr pro inventář zbraní

3.14.3.6 Game ∗ Player::getParentGame (void) const

Getr vracející ukaztel na hru, v rámci níž je tento hráč vytvořen

3.14.3.7 void Player::incrementScore (void)

Metoda pro zvýšení skóre

3.14.3.8 bool Player::interactPlayer (Player ∗const *player*) [virtual]

Implementace abstraktní metody předka

Parametry

player zde nevyužit

Návratová hodnota

vrací false (hráč nemůže vstoupit na jiného hráče)

Implementuje MapObject.

3.14.3.9 bool Player::interactShot (Shot ∗const *shot*) [virtual]

Implementace abstraktní metody předka

Parametry

shot zde nevyužit

Návratová hodnota

vrací false (střela zabije hráče ale nepokračuje dále)

Implementuje MapObject.

3.14.3.10 bool Player::isActive (void) const

Getr a setr pro příznak aktivity

3.14.3.11 bool Player::isMoving (void) const

Getr a setr pro příznak, zda se hráč pohybuje

3.14.3.12 bool Player::isShoting (void) const

Getr a setr pro příznak, zda hráč střílí

3.14.3.13 bool Player::isSpawned (void) const

Getr pro příznak, zda je hráč na mapě

3.14.3.14 void Player::respawn (void)

Obnoví hráče na náhodné lokaci

3.14.3.15 void Player::setActualWeapon (const int *actualWeapon*)

Setr pro nastavení konkrétní zbraně jako aktuální (volá se po sebrání zbraně)

Parametry

actualWeapon pořadí zbraně, na kterou se má přepnout

3.14.3.16 void Player::shot (void)

Po zavolání této metody hráč vytřelí aktuální zbraní

3.14.3.17 void Player::timerEvent (QTimerEvent ∗const *event*)

Tato metoda je zavolána pět vteřin poté, co byl hráč zabit, a způsobí, že se hráč znovu spawne

Parametry

event časovač, který spustil tuto metodu

3.14.3.18 void Player::tryMove (void)

Tato metoda posune hráčovy souřadnice, aby se mohly zjistit kolize

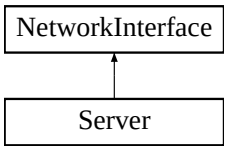
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/player.h
- model/player.cpp

3.15 Dokumentace třídy Server

`#include "server.h"`

Diagram dědičnosti pro třídu Server



Signály

- void sentMessage (QByteArray ∗message)
- void clientLags (int id)
- void clientIsBack (int id)

Veřejné metody

- Server (int port, int count, NetworkInterface ∗const parent=0)
- virtual ∼Server (void)
- void send (QByteArray ∗message)
- int getNetworkID () const
- void setNetworkID (int networkID)
- void timerEvent (QTimerEvent ∗event)

3.15.1 Detailní popis

Server implementuje serverovou logiku. Je zapouzdren pod vzorem Strategy pod jednotne rozhrani s NetworkInterface a volani ji predava trida Network, ktera ho obsluhuje.

3.15.2 Dokumentace konstruktoru a destruktoru

3.15.2.1 Server::Server (int *port*, int *count*, NetworkInterface ∗const *parent* = 0)

Vytvoreni serveru naslouchajiciho na zvolenem portu.

Parametry

port port, na kterem bude server naslouchat

count pocet hracu, kteri budou hrat

3.15.2.2 Server::∼Server (void) [virtual]

Desktruktor uzavirajici serverovy socket.

3.15.3 Dokumentace k metodám

3.15.3.1 void Server::clientIsBack (int *id*) [signal]

Vysle signal, ze lagly klient opet odpovedel a aktivuje ho

3.15.3.2 void Server::clientLags (int *id*) [signal]

Vysle signal s informaci, ze dany klient laguje.

3.15.3.3 int Server::getNetworkID () const [virtual]

Vrati ID server - vzdy 0.

Implementuje NetworkInterface.

3.15.3.4 void Server::send (QByteArray * *message*) [virtual]

Odesilani dat v poli znaku.

Parametry

message pole znaku k odeslani

Implementuje NetworkInterface.

3.15.3.5 void Server::sentMessage (QByteArray * *message*) [signal]

Vyemituje zpravu, ze odeslal packet, aby ho i server zpracoval.

3.15.3.6 void Server::setNetworkID (int *networkID*) [virtual]

Nastavi ID serveru - nic nemeni, je pouze kvuli jednotnemu pristupu pres Network.

Implementuje NetworkInterface.

3.15.3.7 void Server::timerEvent (QTimerEvent * *event*)

Casovana metoda, ktera bude kontrolovat, zda se klient nelagnul.

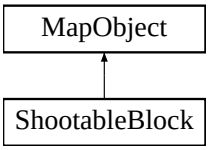
Dokumentace pro tuto třídu byla generována z následujících souborů:

- controller/network/server.h
- controller/network/server.cpp

3.16 Dokumentace třídy ShootableBlock

`#include "shootableblock.h"`

Diagram dědičnosti pro třídu ShootableBlock



Veřejné metody

- ShootableBlock (Game *const parent, const double x1, const double y1, const double x2, const double y2)
- bool interactPlayer (Player *const player)
- bool interactShot (Shot *const shot)

3.16.1 Detailní popis

Tato třída reprezentuje překážky, přes které lze střelit (např. voda)

3.16.2 Dokumentace konstruktoru a destruktoru

3.16.2.1 ShootableBlock::ShootableBlock (Game *const *parent*, const double *x1*, const double *y1*, const double *x2*, const double *y2*)

Vytvoří přestřelitelnou překážku na zadaných souřadnicích

Parametry

parent hra v rámci níž je tato překážka vytvořena

x1 x-ová souřadnice levého horního rohu překážky

y1 y-ová souřadnice levého horního rohu překážky

x2 x-ová souřadnice pravého dolního rohu překážky

y2 y-ová souřadnice pravého dolního rohu překážky

3.16.3 Dokumentace k metodám

3.16.3.1 bool ShootableBlock::interactPlayer (Player *const *player*) `[virtual]`

Implementace abstraktní metody předka

Parametry

player zde nevyužit

Návratová hodnota

vrací false (překážku nelze přejít)

Implementuje MapObject.

3.16.3.2 bool ShootableBlock::interactShot (Shot *const *shot*) [virtual]

Implementace abstraktní metody předka

Parametry

shot zde nevyužit

Návratová hodnota

vrací true (překážka je přestřelitelná)

Implementuje MapObject.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/shootableblock.h
- model/shootableblock.cpp

3.17 Dokumentace třídy Shot

`#include "shot.h"`

Veřejné metody

- Shot (const double x, const double y, const double angle, const double speed, Player ∗const parent)
- ∼Shot (void)
- void move (void)
- Player ∗ getOwner (void) const
- int getShotID (void) const
- double getX (void) const
- double **getY** (void) const

3.17.1 Detailní popis

Třída reprezentující střelu letící na hrací ploše

3.17.2 Dokumentace konstruktoru a destruktoru

3.17.2.1 Shot::Shot (const double *x*, const double *y*, const double *angle*, const double *speed*, Player ∗const *parent*)

Konstruktor vytvoří střelu se zadanými parametry

Parametry

parent ukazatel na hráče, který tuto střelu vytřelil

x x-ová souřadnice střely

y y-ová souřadnice střely

angle směr střely (zadaný v radiánech)

speed rychlost střely (jednotka pixel/frame)

3.17.2.2 Shot::∼Shot (void)

Destruktor zajišťuje, aby mohlo být znovu použité ID této střely

3.17.3 Dokumentace k metodám

3.17.3.1 Player ∗ Shot::getOwner (void) const

Getr pro vlastníka

3.17.3.2 int Shot::getShotID (void) const

Getr pro id střely

3.17.3.3 double Shot::getX (void) const

Getry pro souřadnice střely

3.17.3.4 void Shot::move (void)

Metoda pro pohyb o jeden frame

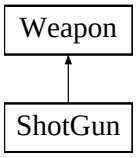
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/shot.h
- model/shot.cpp

3.18 Dokumentace třídy ShotGun

`#include "shotgun.h"`

Diagram dědičnosti pro třídu ShotGun



Veřejné metody

- ShotGun (Player ∗const parent)
- void shot (void)
- void refill (void)

3.18.1 Detailní popis

Třída reprezentující brokovnici

3.18.2 Dokumentace konstruktoru a destruktoru

3.18.2.1 ShotGun::ShotGun (Player ∗const *parent*) `[explicit]`

Konstruktor vytvoří danému hráči brokovnice

Parametry

parent hráč, kterému zbraň patří

3.18.3 Dokumentace k metodám

3.18.3.1 void ShotGun::refill (void) `[virtual]`

Implementace abstraktní metody předka (doplnění nábojů)

Implementuje Weapon.

3.18.3.2 void ShotGun::shot (void) `[virtual]`

Implementace abstraktní metody předka (výstřel zbraně)

Implementuje Weapon.

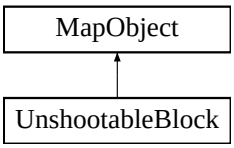
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/shotgun.h
- model/shotgun.cpp

3.19 Dokumentace třídy UnshootableBlock

`#include "unshootableblock.h"`

Diagram dědičnosti pro třídu UnshootableBlock



Veřejné metody

- UnshootableBlock (Game ∗const parent, const double x1, const double y1, const double x2, const double y2)
- bool interactPlayer (Player ∗const player)
- bool interactShot (Shot ∗const shot)

3.19.1 Detailní popis

Tato třída reprezentuje překážky, přes které nelze střelit (např. skála)

3.19.2 Dokumentace konstruktoru a destruktoru

3.19.2.1 UnshootableBlock::UnshootableBlock (Game ∗const *parent*, const double *x1*, const double *y1*, const double *x2*, const double *y2*)

Vytvoří přestřelitelnou překážku na zadaných souřadnicích

Parametry

parent hra v rámci níž je tato překážka vytvořena

x1 x-ová souřadnice levého horního rohu překážky

y1 y-ová souřadnice levého horního rohu překážky

x2 x-ová souřadnice pravého dolního rohu překážky

y2 y-ová souřadnice pravého dolního rohu překážky

3.19.3 Dokumentace k metodám

3.19.3.1 bool UnshootableBlock::interactPlayer (Player ∗const *player*) `[virtual]`

Implementace abstraktní metody předka

Parametry

player zde nevyužit

Návratová hodnota

vrací false (překážku nelze přejít)

Implementuje MapObject.

3.19.3.2 bool UnshootableBlock::interactShot (Shot ∗const *shot*) [virtual]

Implementace abstraktní metody předka

Parametry

shot zde nevyužit

Návratová hodnota

vrací false (překážka není přestřelitelná)

Implementuje MapObject.

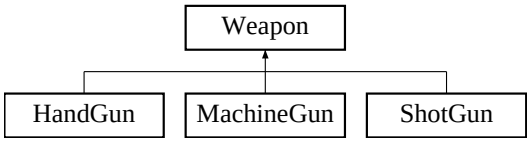
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/unshootableblock.h
- model/unshootableblock.cpp

3.20 Dokumentace třídy Weapon

`#include "weapon.h"`

Diagram dědičnosti pro třídu Weapon



Veřejné metody

- Weapon (Player ∗const parent)
- virtual void shot (void)=0
- virtual void refill (void)=0
- int getAmmo (void) const

Chráněné metody

- void findPointOfCreatingShots (double &x, double &y, double &angle)

Chráněné atributy

- int ammo
- Player ∗ owner

3.20.1 Detailní popis

Třída reprezentující zbraň, kterou nese hráč

3.20.2 Dokumentace konstruktoru a destruktoru

3.20.2.1 Weapon::Weapon (Player ∗const *parent*) `[explicit]`

Konstruktor vytvoří zbraň, která drží zadaný hrač

Parametry

parent hráč, který má tuto zbraň

3.20.3 Dokumentace k metodám

3.20.3.1 void Weapon::findPointOfCreatingShots (double & *x*, double & *y*, double & *angle*) `[protected]`

Pomocná metoda, která na základě parametrů hráče, zjistí, ve kterém bodu se objeví střela(y)

Parametry

x na místo, kam ukazuje tento ukazatel, se zapíše x-ové souřadnice bodu

y na místo, kam ukazuje tento ukazatel, se zapíše y-ové souřadnice bodu

angle na místo, kam ukazuje tento ukazatel, se zapíše úhel, pod kterým bude střela leťet

3.20.3.2 int Weapon::getAmmo (void) const

Getr pro počet zbývajících nábojů

Návratová hodnota

zbývající počet nábojů

3.20.3.3 virtual void Weapon::refill (void) [pure virtual]

Tato metoda doplní zbrani její náboje na plný počet

Implementováno v HandGun, MachineGun a ShotGun.

3.20.3.4 virtual void Weapon::shot (void) [pure virtual]

Tato metoda "naprosto nečekaně" provede to, že daná zbraň vystřelí

Implementováno v HandGun, MachineGun a ShotGun.

3.20.4 Dokumentace k datovým členům

3.20.4.1 int Weapon::ammo [protected]

Počet zbývajících nábojů ve zbrani

3.20.4.2 Player* Weapon::owner [protected]

Hráč, který má tuto zbraň

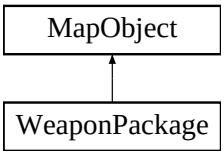
Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/weapon.h
- model/weapon.cpp

3.21 Dokumentace třídy WeaponPackage

`#include "weaponpackage.h"`

Diagram dědičnosti pro třídu WeaponPackage



Veřejné metody

- WeaponPackage (Game ∗const parent=0)
- ∼WeaponPackage (void)
- bool interactPlayer (Player ∗const player)
- bool interactShot (Shot ∗const shot)
- int getWeaponPackageID (void) const
- WeaponType getType (void) const

Chráněné atributy

- int weaponPackageID
- WeaponType type

Statické chráněné atributy

- static double weaponPackageSizeX = 70.0
- static double weaponPackageSizeY = 20.0
- static QBitArray idArray = QBitArray(256)

3.21.1 Detailní popis

Tato třída reprezentuje zbraň, která leží na hrací ploše a může být zvednuta hráčem

3.21.2 Dokumentace konstruktoru a destruktoru

3.21.2.1 WeaponPackage::WeaponPackage (Game ∗const *parent* = 0) [explicit]

Konstruktor vytvoří náhodnou zbraň na herní ploše (brokovnici nebo samopal)

Parametry

parent hra, v rámci níž je zbraň vytvořena

3.21.2.2 WeaponPackage::~WeaponPackage (void)

Destruktor zajišťuje, aby mohlo být znovu použité ID této zbraně

3.21.3 Dokumentace k metodám

3.21.3.1 WeaponType WeaponPackage::getType (void) const

Getr pro typ zbraně

3.21.3.2 int WeaponPackage::getWeaponPackageID (void) const

Getr pro id zbraně

3.21.3.3 bool WeaponPackage::interactPlayer (Player *const *player*) [virtual]

Implementace abstraktní metody předka

Parametry

player hráč, který dostane tuto zbraň

Návratová hodnota

vrací true (přes ležící zbraň lze přejít)

Implementuje MapObject.

3.21.3.4 bool WeaponPackage::interactShot (Shot *const *shot*) [virtual]

Implementace abstraktní metody předka

Parametry

shot zde nevyužit

Návratová hodnota

vrací true (ležící zbraň je přestřelitelná)

Implementuje MapObject.

3.21.4 Dokumentace k datovým členům

3.21.4.1 QBitArray WeaponPackage::idArray = QBitArray(256) [static, protected]

Toto pole udržuje záznamy o tom, která ID jsou pro zbraně použity

3.21.4.2 WeaponType WeaponPackage::type [protected]

Typ zbraně

3.21.4.3 int WeaponPackage::weaponPackageID [protected]

Jednoznačné ID zbraně

3.21.4.4 double WeaponPackage::weaponPackageSizeX = 70.0 [static, protected]

Šířka zbraně v pixelech

3.21.4.5 double WeaponPackage::weaponPackageSizeY = 20.0 [static, protected]

Výška zbraně v pixelech

Dokumentace pro tuto třídu byla generována z následujících souborů:

- model/weaponpackage.h
- model/weaponpackage.cpp

Index