VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

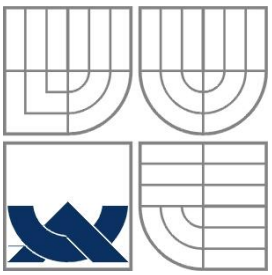
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SÍŤOVÁ HRA: HON NA PONORKU

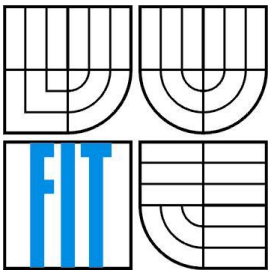
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ KUDRNA
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SÍŤOVÁ HRA: HON NA PONORKU

NETWORK GAME: DESTROY THE SUBMARINE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ KUDRNA
AUTHOR

VEDOUCÍ PRÁCE — ING. IGOR SZÖKE
SUPERVISOR

BRNO 2010

Abstrakt

Bakalářská práce popisuje návrh a implementaci 2D hry Hon na ponorku. V této hře ovládá každý hráč svoji ponorku a snaží se zničit své protivníky. Hra umožňuje hru více hráčů přes počítačovou síť i u jednoho PC. Aplikace je vyvíjena v programovacím jazyku C++ s pomocí Qt frameworku. Jedná se o multiplatformní aplikaci.

Abstract

The bachelor´s thesis describes the design and the implementation of the 2D network game “Destroy the submarine”. In this game each player controls his submarine and tries to destroy the submarines of his opponents. The game can be played by more players via the computer network or on one PC. The application is developed in the C++ programming language with the help of the Qt framework. It is a multi-platform application.

Klíčová slova

síťová hra, Qt framework, ponorka, 2D, multiplatformní

Keywords

network game, Qt f, submarine, 2D, multi-platform

Citace

Kudrna Jiří: Síťová hra: Hon na ponorku, bakalářská práce, Brno, FIT VUT v Brně, 2010

Síťová hra: Hon na ponorku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Igora Szökeho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Kudrna
17. 5. 2010

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu Ing. Igoru Szökemu, za pomoc a rady při tvorbě této práce.

Obsah

1 Úvod

Cílem práce bylo vytvoření 2D síťové hry s názvem Hon na ponorku. Součástí práce byla volba vhodného rozhraní pro 2D grafiku (s ohledem na podporu různých operačních systémů) navržení vhodného komunikačního protokolu a naimplementování hry.

Práce zahrnuje návrh pravidel, grafického provedení a síťové komunikace. Z důvodu požadavku multiplatformní hry byl zvolen programovací jazyk C++ a Qt framework 4.6, který toto zajišťuje.

V kapitole číslo 2 je vytyčena představa o hře - její pravidla a možná vylepšení.

Třetí kapitola seznamuje se základy multiplatformního Qt frameworku, pomocí kterého je hra vytvořena, popisuje tvorbu GUI, signály, sloty, třídu QWidget a QLayout i její odvozené třídy QBoxLayout, QGridLayout, QFormLayout a QStackedLayout.

Kapitola 4 je věnována návrhu síťové komunikace, konkrétně výběru transportního protokolu a návrhu komunikačního protokolu.

Kapitola 5 se zabývá samotnou implementací celé hry. Od jednotlivých tříd, přes kompletní hrací prostředí až po síťovou komunikaci.

V závěrečné, 6. kapitole je pak celá práce zhodnocena.

V práci se vyskytují názvy tříd, metod, signálů a dalších objektů implementovaných ve hře. Odlišení těchto názvů je realizováno fontem `psacího stroje`.

2 Návrh hry

V této kapitole si ukážeme představu o tom, jak bude hra vypadat a jaká bude mít pravidla.

2.1 Pravidla hry

Hlavním cílem hry bude eliminace protivníků. Neuvažujeme o hře v týmech. Do jedné hry se budou moci zapojit maximálně 4 hráči. Hra bude mít dva herní módy.

Prvním bude *hra na čas*, při které se odpočítává nastavený čas a každému hráči se započítává počet zneškodněných nepřátel. Hráč, který má po vypršení času největší skóre, vyhrál.

Druhým typem hry bude *hra na životy*. V této hře je nastaven hráčům stejný počet životů. Tyto životy se postupně při každém zásahu odečítají. Pokud hráč dosáhne hodnoty životů rovné nule, je vyřazen ze hry. V tomto módu vyhrává hráč, který vydrží nejdéle.

2.2 Vybavení ponorek

Každá ponorka bude vybavena dvěma druhy zbraní. První zbraní jsou neřízená torpéda. Jejich počet bude neomezený. Jejich používání bude však ovlivněno dobou nabíjení 4 torpédových komor. Další základní zbraní bude řízené torpédo. Při jeho vystřelení se zaměří na nepřítele a bude se na něho navádět. Po uplynutí nabíjecího času, můžeme vystřelit další.

2.3 Herní plocha

Celá herní plocha bude odkryta a bude mít pevně dané rozměry. Projíždění okrajem mapy nebude povoleno. Dalšími objekty ve hře budou překážky umístěné na mapě. Tyto překážky budou nezničitelné a budou poskytovat hráčům ochranu před střelami.

2.4 Tvorba map

Tvorba map by měla být jednoduchá nejen pro tvůrce hry, ale i pro hráče. Možným řešením by mohl být zápis struktury mapy do XML[1] souboru. Tento soubor by obsahoval údaje o pozici překážek v mapě, jejich velikost a také startovní pozice hráčů.

2.5 Hra Hot-Seat

Dalším plánovaným vylepšením, krom tvorby vlastních map, by mohla být hra více hráčů na jednom počítači. Tento druh hraní byl populární hlavně v dřívějších dobách, kdy měl doma každý pouze jeden počítač a počítače nebyly připojeny k počítačové síti. I přes dnešní možnost hrát s někým, kdo sedí u PC na druhé straně světa, má toto hraní u jednoho stroje stále své kouzlo. Proto jsem se rozhodl tuto možnost do hry přidat.

[1] Extensible Markup Language, v překladu rozšiřitelný značkovací jazyk.

3 Qt framework

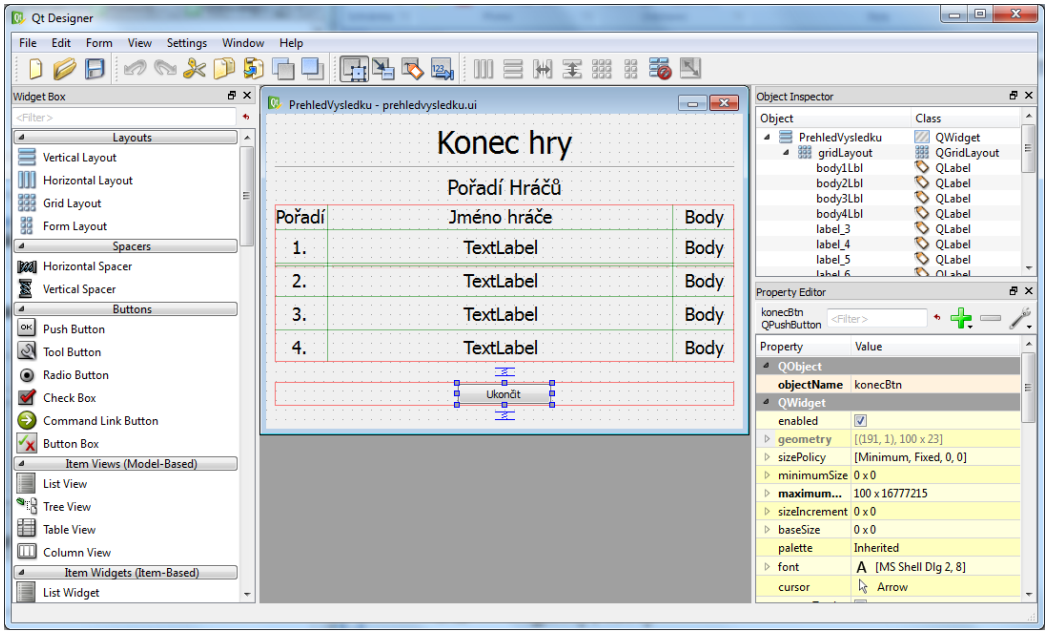
Qt je multiplatformní všestranný framework vyvíjený firmou Trolltech od roku 1991. V roce 1998 bylo zvoleno Qt jako hlavní knihovna pro implementaci prostředí KDE pro Linux. Nyní tuto firmu vlastní NOKIA, která uvolnila framework pod licencí LGPL. Do té doby byl framework šířen pod licenci GPL. Mezi známé aplikace vyvíjené za pomoci Qt patří Skype, Opera, Google Earth a mnoho dalších.

Jak už bylo řečeno, jedná se multiplatformní framework. S jeho pomocí můžeme vyvíjet aplikace nejen pro GNU/Linux a Windows, ale i pro Mac OS, Embedded Linux a zařízení s operačním systémem Symbian. Hlavním programovacím jazykem je C++. Qt ovšem podporuje i jiné jazyky, například Python (PyQt), Ruby, C# a Java.

Následující podkapitoly obsahují další informace o tomto frameworku.

3.1 Tvorba GUI

Pro snadnou tvorbu GUI je k dispozici nástroj Qt Designer, umožňující snadnou tvorbu uživatelského rozhraní metodou drag&drop. Také můžeme rovnou propojit námi naprogramované signály a metody s ovládacími prvky. Pokud bychom si ovšem chtěli napsat celé grafické rozhraní sami, je to možné. Použití tohoto nástroje je ovšem rozhodně rychlejší a pohodlnější. Všechny vytvořené GUI návrhy jsou uloženy v souborech s příponou ui.



Obrázek 3.1: Ukázka okna QtDesigner

Na obrázku 3.1 vidíme okno pro návrh GUI. Uprostřed se nachází okno, jehož návrh právě probíhá. Vlevo je seznam widgetů a layoutů, které můžeme pro návrh použít. Vpravo nahoře seznam všech objektů, které jsme už vložili do našeho okna. Objekty jsou v tomto seznamu uspořádány podle toho, jakému layoutu náleží. Okno pod tímto seznamem nám ukazuje hodnoty vlastností právě zvoleného prvku.

3.2 Signály, sloty

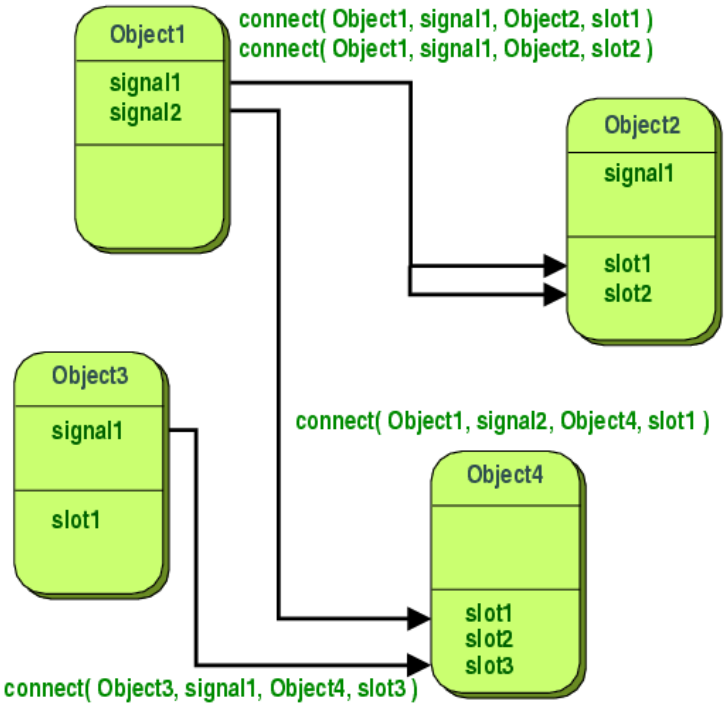
Jedná se o druh komunikace mezi objekty a jednu z vlastností odlišující Qt od ostatních frameworků. Pokud chceme ve své vlastní třídě používat signály nebo sloty, musíme do definice třídy vložit makro `Q_OBJECT`, který nám toto umožní.

Signál je metoda, která nevrací hodnotu a ani nemá tělo. Má pouze prototyp. Signál, jak už jeho název napovídá, slouží ostatním objektům k signalizaci vzniku nějaké události. Pro jeho odeslání stačí použít příkaz `emit` spolu s názvem signálu. Jako při volání metody.

Slot je metoda, kterou můžeme provázat se signálem. Tento signál poté spouští danou metodu.

```
Class MojeTlacitko: public QPushButton
{
Q_OBJECT
…
signals:
    void stisknuto();
public slots:
    void schovat(bool schovat);
}
```

Zdrojový kód 3.1: Použití makra Q_OBJECT



Obrázek 3.2: Ukázka propojení objektů

Obrázek 3.2 je převzat z nápovědy programu Qt Creator [5]. Ukazuje nám, jakým způsobem dochází k propojení signálu a slotu určitých objektů. K propojení slouží metoda `connect`. Tato metoda je statickou metodou třídy `QObject`, která je základní třídou všech objektů v Qt. První dva parametry při volání této metody definují signál, na který se má reagovat a objekt vysílající tento signál. Třetím parametrem je příjemce signálu. Čtvrtý parametr může být signál i slot. Pokud se jedná o slot, je při přijetí zachytávaného signálu tento slot zavolán. Jedná-li se o signál, je tento signál vyvolán tímto objektem.

3.3 Třída QWidget

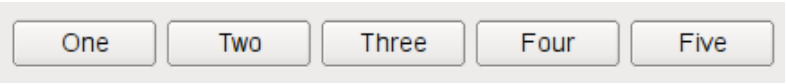
Tato třída je základní třídou všech GUI prvků. Jde tedy o základní stavební kámen. Pokud chceme vytvořit vlastní ovládací prvek, postačí nám k tomu vytvoření vlastní třídy, která bude potomkem `QWidget`. Od této chvíle může být naše třída použita v návrhu uživatelského rozhraní nebo jako samostatné okno. Widgety lze skládat do složitějších za pomoci ostatních jednodušších widgetů. Konstruktoru třídy `QWidget` může být předán ukazatel na jiný widget jako parametr parent. Pokud tento parametr nenastavíme nebo ho nastavíme na 0, stává se widget samostatným oknem. Pokud mu ovšem předáme ukazatel na existující widget, stane se součástí tohoto widgetu.

3.4 Třída QLayout

`QLayout` je základní třídou pro správu geometrických prvků v GUI návrhu. Jedná se o abstraktní třídu, od které jsou odvozeny všechny ostatní třídy (layouty) spravující rozložení prvků v uživatelském rozhraní. Pomocí těchto tříd můžeme pozicovat všechny grafické prvky a ostatní widgety a ovlivňovat jejich chování při změnách velikosti nadřazeného okna.

QBoxLayout

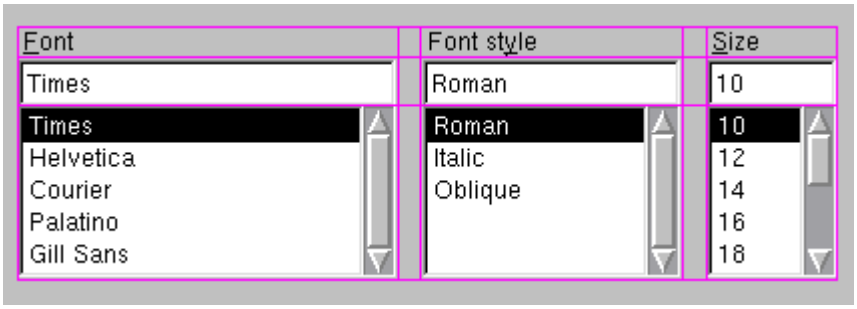
Jednoduchý layout, který řadí prvky horizontálně nebo vertikálně. Pomocí příslušných metod můžeme ovlivnit zarovnávání vložených prvků.



Obrázek 3.3: Horizontální QBoxLayout (převzato z [5])

QGridLayout

Všechny prvky jsou uchycovány do mřížky. Každé buňce můžeme nastavit minimální rozměry a buňky mohou existovat i bez widgetu uvnitř. Stačí do buňky vložit prvek zvaný spacer. Jde o třídu `QSpacerItem`, která zajišťuje prázdné místo v layoutu. Stačí zvolit rozměr spaceru a vložit. Pokud chceme, můžeme nastavit, aby spacer zabíral co nejvíce místa a roztahoval svoji velikost.



Obrázek 3.4: QGridLayout (převzato z [5])

QFormLayout

Layout je určen pro tvorbu formulářů. Obsahuje dva sloupce. První pro popisek (label) a druhý pro objekt, do kterého se zadávají vstupní data (např.: textové pole).

Obrázek 3.5: QFormLayout (převzato z [5])

QStackedLayout

Třída `QStackedLayout` poskytuje zásobník více widgetů, z nichž je v jeden čas viditelný pouze jeden.

4 Teorie síťové komunikace

Zadání práce obsahuje požadavek na navžení vlastního komunikačního protokolu. Nejprve musíme zvolit, přes který síťový protokol bude naše aplikace komunikovat, a poté navrhnout samotný komunikační protokol aplikace.

4.1 Výběr síťového protokolu

Pro naši aplikaci musíme zvolit vhodný transportní protokol. V úvahu připadá UDP nebo TCP. Každý z těchto protokolů má své výhody i nevýhody.

TCP protokol vytváří spojení, na kterém nám protokol garantuje spolehlivé doručování a doručení dat ve správném pořadí. Spolehlivým doručováním rozumíme kontrolu přijatých dat pomocí kontrolního součtu a znovu odeslání dat při jejich nedoručení příjemci. Nevýhodou tohoto protokolu jsou větší rozměry paketů díky větší hlavičce. Ta musí obsahovat informace pro zajištění výše zmíněných vlastností protokolu.

UDP je jednodušší protokol a žádné spojení nevytváří, je tedy nespojový. Neposkytuje žádné záruky na doručení diagramu a ani na to, zda bude zachováno pořadí doručení. Výhodou je jeho rychlost. Neobsahuje tak velkou hlavičku, jak je tomu u TCP protokolu, jeho diagramy mají tudíž menší velikost.

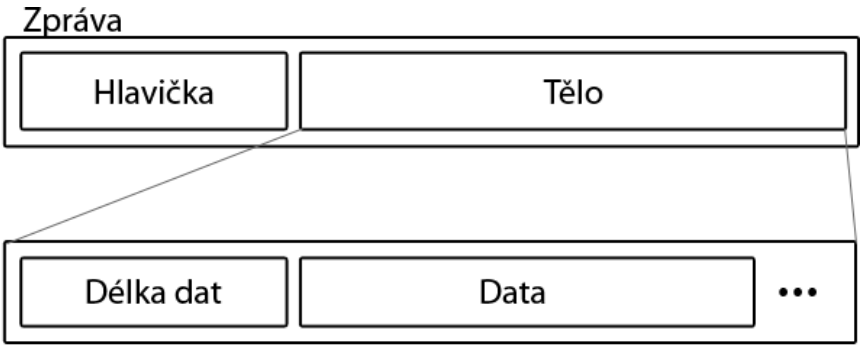
Po zvážení všech pro a proti jsem zvolil protokol TCP. Důvodem volby byla jeho schopnost hlídat pořadí příchozích dat a moje zkušenosti s programováním TCP serveru v prostředí Qt z předmětu ICP[2].

4.2 Návrh aplikačního protokolu

K vytvoření aplikačního protokolu můžeme využít jazyk XML. Výhodou této implementace je lehká čitelnost zpráv, jednoduchost vytváření a zpracování. Zpráva informující o připojení nového hráče s nickem Jirka vypadá například takto: „<novyHrac>Jirka</novyHrac>“. Je nutné ovšem používat zkrácené názvy paketů pro úsporu přenášených dat.

V práci byla zvolena jiná metoda vytváření zpráv. Místo XML prvků je využito identifikačního kódu zprávy umístěného v hlavičce. Tento kód nám řekne, o jakou zprávu jde a jaká data, v jakém uspořádání můžeme očekávat v jejím těle. Kód zprávy je číslo. Tato čísla můžeme dělit do skupin podle významu. Například kódy zpráv zajišťující chatovací služby budou z intervalu 100 až 200, pro navazování spojení se použije interval 50 až 100, atd. Nevýhodou oproti XML je těžší vytváření zpráv a nutnost znát význam každého kódu pro rozpoznání paketu. U XML přístupu nám název tagu naznačil, o jakou zprávu se jedná. Zde máme k dispozici pouze číslo, které nám bez překladu nic neříká.

[2] Seminář C++



Obrázek 4.1: Struktura zprávy

5 Implementace

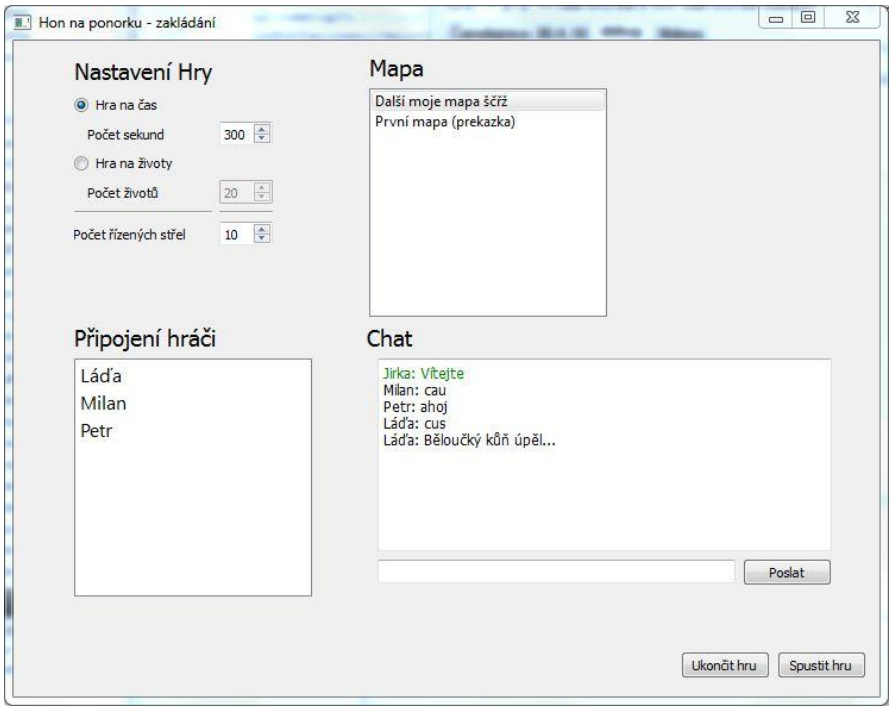
V této kapitole si představíme tvorbu grafického uživatelského rozhraní, několik vytvořených tříd potřebných pro hru a námi vytvořené widgety. Dále se podíváme na tvorbu map, hry hot-seat a síťové hry. A na konci si popíšeme tvorbu síťové komunikace a problémy s ní spojené.

5.1 GUI

Uživatelské rozhraní této hry je vytvořeno pomocí nástroje Qt Designer, o kterém jsme se zmiňovali v podkapitole 3.1. Nejprve ovšem proběhly pokusy o napsání kódu rozhraní ručně. Tento postup byl však neefektivní a pomalý. Proto bylo nakonec využito tohoto nástroje.

S jeho pomocí bylo sestaveno jednoduché grafické rozhraní ze základních ovládacích prvků. Na místa určená pro widgety, které ještě nebyly naprogramovány, se umístili zástupci, na které se později dané komponenty připojily. Mezi widgety, které se musely udělat, patří přehled život/bodů, stav munice a chatovací okno.

Ukázka, jak vypadá výsledek návrhu GUI pro zakládání síťové hry lze vidět na obrázku 5.1.



Obrázek 5.1: Ukázka GUI

5.2 Herní objekty

V této podkapitole se zaměříme na objekty, které používáme přímo ve hře - herní objekty a widgety.

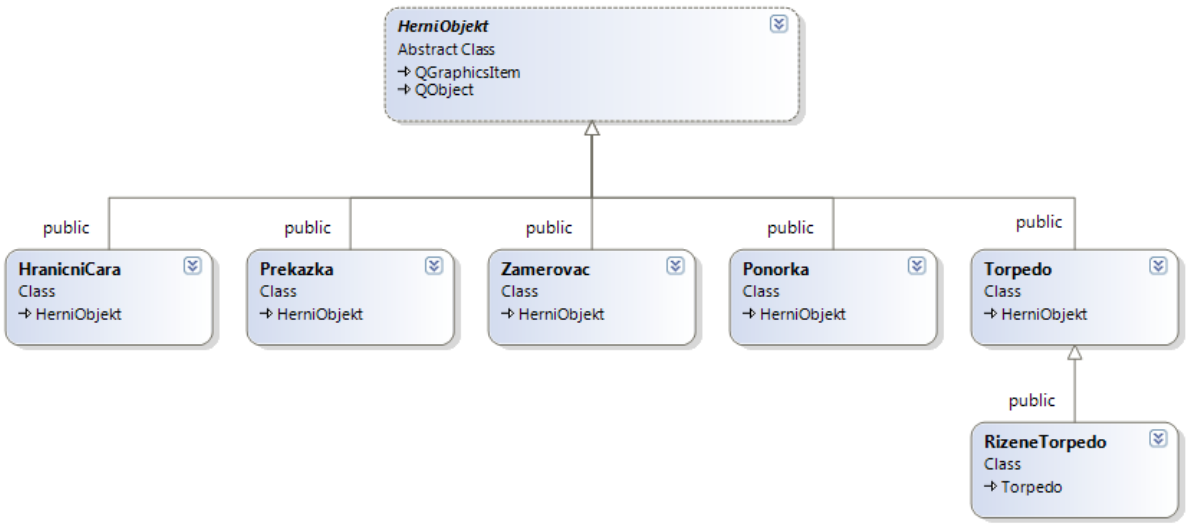
5.2.1 Třída HerniObjekt

Jedná se o abstraktní třídu určenou k odvození ostatních herních prvků. Sama třída je odvozena od Qt třídy `QGraphicsItem`. To naší třídě umožňuje, aby byla použita jako grafický objekt v `QGraphicsScene` (dále jen GS). GS je vrstva umožňující správu velkého množství 2D objektů. Pro jejich vizualizaci se musí použít třída `QGraphicsView`, která umožní zobrazení celé scény, nebo jen její výseče.

Třída obsahuje tři virtuální metody:

- `void zasazen()`
  Tato metoda je určena k implementaci reakce objektu na zasáhnutí jiným objektem.
- `void zasahnul(HerniObjekt *)`
  Reakce objektu při zasáhnutí objektu, jako parametr je předán ukazatel na zasažený objekt.
- `TypObjektu identifikace()`
  Identifikuje, o jaký objekt se jedná. Zda jde o ponorku, střelu či zeď.

Třída `QGraphicsItem`(dále GI) obsahuje další virtuální metody, které je třeba v odvozených třídách implementovat. První z nich je metoda `paint`. Tato metoda určuje grafické znázornění objektu. Další metodou je `advance`. Pokud jsou GI umístěny v GS a je volán slot `advance` u GS, je i potom volána tato metoda u všech těchto grafických objektů. Volání této metody je provedeno dvakrát, nejprve s parametrem 0, který říká, že se blíží změna scény. Podruhé s parametrem 1, který umožní objektům pohyb.



Obrázek 5.2: Graf dědičnosti třídy HerniObjekt

5.2.2 Třída Ponorka

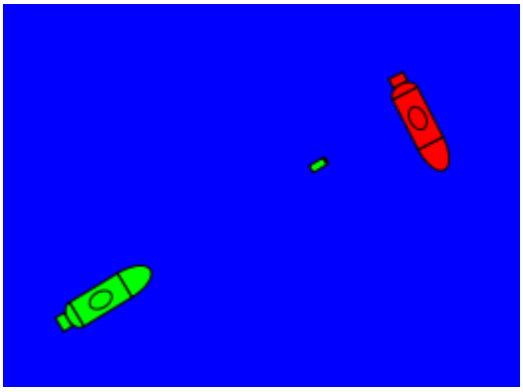
Třída je odvozena od třídy `HerniObjekt` a doiplementovává výše zmíněné metody. `Ponorka` je přímo ve hře zkonstruována z jednoduchých geometrických tvarů. Nejedná se tedy o obrázek načítaný do hry. Provedení ponorky zachycuje obrázek 5.3.

Pohyb ponorky po scéně je realizován v metodě `advance`. Třída obsahuje strukturu `bool` proměnných, které se nastavují v závislosti na stisknutých klávesách. Tato struktura se poté v metodě

`advance` vyhodnotí a provede se posun ponorky přepočtem lokální souřadnice na souřadnici v herní ploše.

Při střelbě torpéd jako první zkontrolujeme, zda je torpédo k dispozici. Pokud ano, vytvoříme nový objekt od třídy `Torpedo` na souřadnicích získaných opět přepočtem z lokálních souřadnic ponorky. Jako další zjistíme úhel dráhy střely. K tomuto nám pomůže metoda `uhelNatoceni`. Metoda vrací hodnotu úhlu natočení ponorky vzhledem k hrací ploše. Poté už jen stačí torpédo vložit do scény pomocí metody `addItem` a vyšleme signál oznamující vystřelení torpéda. Nastavíme časovač, kterým simulujeme čas potřebný k znovunabití torpédové komory.

Dále ve třídě `Ponorka` najdeme statickou proměnnou `pocetPonorek` (slouží k zjištění počtu vytvořených ponorek) a proměnné pro uchování startovní pozice ponorky. Ty jsou potřebné pro vrácení ponorky na výchozí pozici po jejím zničení.



Obrázek 5.3: Vzhled ponorek

5.2.3 Třída Torpedo, RizeneTorpedo

Jedná se o jednoduché třídy. Třída `Torpedo` je základní třídou pro `RizeneTorpedo`. Jedná se o stejné grafické objekty. Jediným rozdílem je pohyb. Torpédo se pohybuje po přímce. Od bodu vytvoření v zadaném úhlu. Po nárazu do objektu je torpédo odstraněno ze scény a dealokováno.

Řízené torpédo musí dostat ukazatel na cíl. Neboli ukazatel na `HerniObjekt`. Pokud je vytvořeno a tento ukazatel je mu předán, vypočítává si při každém překreslení obrazu úhel k cíli. Pokud je tento úhel v rozmezí ±90°, upraví torpédo směr k cíli. A to o ±4° podle vypočítaného úhlu. Pokud je vypočítaný úhel mimo udávané rozmezí, torpédo bude pokračovat aktuálním směrem do té doby, dokud nenastane kolize s jiným objektem.

Získání cíle probíhá prostřednictvím třídy `Zamerovac`, jejíž instanci má každá ponorka. Jedná se o grafický objekt, který při zavolání metody `zamer` vykreslí do mapy neviditelnou čáru od pozice ponorky ve směru střelby. Dále projde seznam kolizních objektů a najde první objekt, který je ponorka. Jeho adresu poté vrátí. Pokud nedojde k najití ponorky, vrátí se `NULL`.

5.2.4 Herní plocha

Herní plocha se skládá ze dvou částí. Z vrstvy, která uchovává a zpracovává grafické objekty. Tuto vrstvu zastupuje třída `QGraphicsScene`. A vrstvy, jež tuto datovou vrstvu zobrazuje uživateli. Zobrazení probíhá přes třídu `QGraphicsView`. Tato třída poskytuje pohled na určitý výřez scény.

V naší implementaci je vytvořena třídu `HerniPlocha`. Jedná se o potomka již zmiňované `QGraphicsView`. Obsahuje navíc metodu pro nastavení zachytávání událostí z klávesnice

a definuje tělo dvou virtuálních metod `keyPressEvent` a `keyReleaseEvent`. Tyto metody jsou volány při stisku jakékoliv klávesy na klávesnici. Zároveň je jim jako parametr předán kód stisknuté klávesy. Pokud je kód klávesy shodný s některou klávesou určenou pro ovládání ponorky, je příslušné ponorce nastaven příznak označující tuto událost.

Dále tato třída obsahuje metodu `nastavOkraje(int sirka,int vyska)`. Parametry této metody jsou rozměry herní plochy, po které se mohou ponorky pohybovat. Kolem této plochy se pomocí třídy `HranicniCara`, která je také potomkem třídy `HerniObjekt`, postaví barikády. Tyto barikády znemožňují hráčům plout mimo jim vytyčený prostor. Z nutnosti identifikovat při kolizi objekt, se kterým kolize nastala, musela být pro tyto hranice vytvořena zvláštní třída. Nebylo možné použít základní `QGraphicsItem`.

5.3 Vytvořené widgety

V této kapitole si představíme několik widgetů, které jsme museli pro hru vytvořit. Většinou jde o zabalení několika základních prvků do většího widgetu a přidání obslužných metod.

5.3.1 Časomíra

Pro mód „Hra na čas“ bylo třeba vytvořit widget, kterému předáme počet sekund do konce hry, tyto sekundy nám poté zobrazí ve formátu mm:ss[3] (viz obrázek 5.4). Pro tento účel jsme vytvořili třídu (widget) `Odpocet` obsahující slot `zobrazCas(int sekundy)`. Tomuto slotu posíláme pomocí signálu údaj o zbývajících vteřinách. Slot tento údaj přepočítá na minuty a sekundy a zobrazí v požadovaném formátu.

5.3.2 Informace o hráči

Dalším potřebnou věcí ve hře je zobrazení hrajících hráčů a jejich skóre. K tomuto účelu je vytvořena třída `InformaceOHraci`. Třída umožňuje nastavení popisku, který se zobrazí nad číselnou hodnotou. Tuto hodnotu lze taky nastavit. V našem případě představuje popisek místo pro hráčův nick[4] a číselná hodnota je jeho počet životů či skóre, to už záleží na druhu hry, jaká se právě hraje. Vzhled widgetu lze vidět na obrázku 5.6.

5.3.3 Přehled zbraní

Pro zobrazení počtu zbývající munice, vybrané zbraně a stavu nabíjení torpédových komor je v projektu třída `PrehledZbrani`. Pro zachytávání signálů má tato třída několik slotů. Slot `torpedoVystreleno`, jenž nastaví nápis „nabíjím…“ u použité torpédové komory. `torpedoPripraveno`, nastavující u nabité komory nápis „nabito“. `rizenaVystrelane`, která nastaví u řízené střely příznak, že je komora nabíjena a odečte jednu řízenou střelu. K tomuto slotu je nutný slot `rizenaPripravena`, který příznak nastaví zpět na „nabito“. Posledním slotem je `prepniZbran`. To přepíná zvýraznění kolem vybrané zbraně. Zvýraznění je realizováno zelenou barvou textu (viz obrázek 5.5).

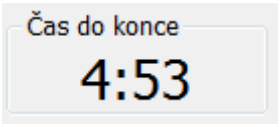
[3] minuty:sekundy.

[4] herní přezdívka hráče

5.3.4 Chat

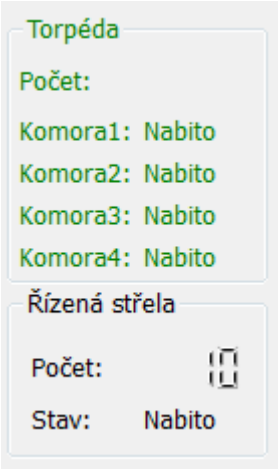
Widget obsahuje dvě textová okna a jedno tlačítko. První okno je víceřádkové pro zobrazení příchozích zpráv. Druhé je určené k odesílání vzkazů. Odeslání se provede kliknutím na tlačítko odeslat nebo stisknutím klávesy ENTER. Zprávy se zobrazují ve tvaru [nick: text zprávy]. Vzkazy napsané přímo hráčem budou zvýrazněné zelenou barvou textu. Vzhled tohoto widgetu je vidět na obrázku 5.1 v kapitole 5.1.

5.3.5 Ukázka widgetů



Obrázek 5.4: Widget Odpocet

Obrázek 5.6: Widget InformaceOHraci



Obrázek 5.5: Widget PrehledZbrani

5.4 Tvorba map

Jak už jsme zmiňovali dříve, rozhodli jsme se umožnit hráčům vlastní mapy. Implementaci této vlastnosti ukazuje tato kapitola.

Třída HerniMapa

Prvním krokem bylo vytvoření třídy představující jednu herní mapu. Třídu jsme pojmenovali `HerniMapa` a podědili jsme ji od `QListWidgetItem`. Tato dědičnost nám dovoluje použít instance třídy `HerniMapa` jako položky ve widgetu `QListWidget`, což je z uživatelova pohledu seznam položek s možností výběru.

Třída obsahuje dva kontejnery `QMap`. Jeden je určen pro uložení startovních pozic hráčů a druhý pro startovní natočení ponorek. Oba mají jako klíč ID hráčů. Dále třída uchovává seznam všech překážek, které se v mapě vyskytují. Překážky mají svoji vlastní třídu se jménem `Prekazka`, ta obsahuje informaci o rozměrech, pozici a barvě překážky. Dále `HerniMapa` uchovává název mapy a jméno souboru, z kterého byla načtena.

Metod má několik. Jedná se hlavně o metody pro načtení hodnot do výše zmíněných proměnných a o jejich zpětné získání. Překážky vkládáme do třídy po jedné. Při požadavku na jejich vrácení, vrátíme seznam ukazatelů na překážky.

Reprezentace mapy pomocí XML

Pro jednoduchou tvorbu map jsme zvolili zápis informací o mapě do XML souboru. Struktura tohoto souboru je jednoduchá a měla by umožnit vytvoření mapy každému, kdo ovládá práci s textovým editorem. Stačí tento XML soubor sepsat a uložit do adresáře Mapy.

Celá definice mapy je uvnitř tagu <mapa>. Jméno mapy, které je zobrazeno v seznamu map uvádíme do tagu <jmeno>. Startovní pozice hráče je uvnitř tagu <pozice id="x">, kde atribut id určuje, pro jaké ID hráče bude tato pozice použita. Pozici a úhel natočení zapisujeme pomocí tagů <x>, <y> a <uhel>. Hodnoty mezi těmito tagy jsou celé čísla a představují souřadnice a úhel ve stupních. Překážky jsou definovány uvnitř tagu <objekt>. Opět jsou použity tagy <x>, <y> pro určení souřadnice objektu. Novými tagy jsou <w>, <h> pro určení šířky a výšky překážky a <barva>, kterým můžeme zvolit barvu překážky. Barva se zadává ve formátu #RRGGBB, kde každou barevnou složku představuje číslice v šestnáctkové soustavě. Ukázku XML souboru s definicí mapy najdeme v příloze.

Načítání map z XML

Celé zpracování probíhá metodou SAX[5]. To umožňuje proudové zpracování dokumentu XML. U tohoto zpracování se rozdělí dokument na počáteční tagy, koncové tagy a na obsahy tagů, atd. . Při procházení dokumentu jsou volány metody podle toho, do jaké skupiny nalezený prvek spadá.

Načtení informací z XML souboru a jejich uložení do objektu typu `HerniMapa` má na starosti naše třída `XMLParser`. Tato třída je odvozena od `QXmlDefaultHandler`, což je základní třída pro obsluhu XML dokumentů. V naší třídě bylo nutné naimplementovat obsah virtuálních metod zděděných od `QXmlDefaultHandler`. Mezi ty, které se starají o načítání hodnot z XML souboru, patří `startElement`, `endElement` a `characters`. Metoda `startElement` se v našem případě stará pouze o načtení ID startovní pozice. Všechny ostatní hodnoty jsou ukládány pomocí metody `endElement`, ta zkontroluje, zda metoda `characters` načetla správná data a ty poté uloží do příslušné proměnné v již vytvořeném objektu třídy `HerniMapa`. Po úspěšném projití celého souboru je vrácen ukazatel na nově vytvořený objekt s mapou. Celá tato operace je provedena v metodě `nactiMapu`, které je jako parametr předán název souboru. Metoda si vytvoří objekt `HerniMapa`, pokusí se otevřít soubor a pokud se otevření zdaří, je soubor postupně načítán za pomoci třídy `QXmlSimpleReader` dokud není celý rozparsován nebo není nalezena chyba. Nakonec metoda vrátí ukazatel na načtenou mapu nebo v případě chyby hodnotu NULL.

5.5 Hra HotSeat

Jako rozšíření byla zvolena implementace hry více hráčů na jednom počítači. Jedná se o hru maximálně dvou hráčů na jedné klávesnici.

5.5.1 Třída HraHotSeat

`HraHotSeat` je třída představující okno, ve kterém se odehrává celá hra. Obsahuje ukazatele na oba hrající hráče, zajišťuje časování pro překreslení snímku scény, zobrazuje okno s výsledky, zapíná a pozastavuje hru, nastavuje velikost herní plochy, vytváří hráče, zakládá hru a obsahuje metodu pro přidání překážky do hry.

[5] Simple API for XML

Pokud chceme hrát hru, musíme jako první věc vytvořit instanci této třídy. Poté předáme pomocí metody `nastavVelikostPlochy`, nastavíme šířku a výšku hrací plochy. Dále máme na výběr, jestli chceme založit hru na čas nebo hru na životy. Pokud hru na čas, voláme metodu `zalozHruNaCas` s parametry představující délku hrací doby v sekundách a počet řízených střel. Jestliže chceme hrát hru na životy, zavoláme metodu `zalozHruNaZivoty`. Zde opět jako parametr uvedeme počet řízených střel. Druhým parametrem už nebudou sekundy, ale počet životů. Poté do hry přidáme oba hráče pomocí metod `vytvorHrace1` a `vytvorHrace2`, kterým jako parametry předáme barvu, nick a startovní pozici a natočení získanou z herní mapy. Posledním krokem je načtení všech překážek do herního světa. K tomuto použijeme metody `pridejPrekazku`, která přidává do hry postupně všechny překážky ze seznamu překážek mapy. A jako poslední stačí už jen zapnout hru metodou `zapnoutHru`. Zapnutí hry probíhá nastavením časovače `casovacSnimku` a jeho spuštěním. Tento časovač zajišťuje překreslování herní scény.

Po ukončení hry nastane zastavení časovače `casovacSnimku` a výsledky se předají třídě `PrehledVysledku`. Ta se postará o jejich zobrazení v novém okně. Vidíme zde pořadí hráče, jeho nick a počet získaných bodů.

5.5.2 Třída HotSeatForm

`HotSeatForm` obsahuje ukazatel na třídu `HraHotSeat` a pomocí tohoto ukazatele tuto třídu (její instanci) spravuje. Poskytuje nám grafické rozhraní pro nastavení parametrů hry a napsání jmen hráčů. Toto rozhraní bylo stejně jako všechna ostatní vytvořeno pomocí nástroje Qt Designer. Pokud zvolíme spuštění hry, vezmou se všechna data z formulářů a využijí se pro sestavení hry tak, jak je napsáno v kapitole 5.5.1.

5.6 Síťová hra

Hlavní třídou při hraní po síti je `SitovaHra`. Jde o třídu zobrazující okno s herní plochou, údaje o hráčích a zbývající čas. Dále umožňuje přidávat hráče a překážky do hry, spouštět a zastavovat hru, nastavovat parametry hry, odebrat hráče ze hry, nastavit velikost hrací plochy. Je velice podobná třídě `HraHotSeat`, kterou popisuje kapitola 5.5.1. Některé metody má s touto třídou shodné. Například `zalozHruNaCas`, `zalozHruNaZivoty`, `zapnoutHru` atd. . Obsahuje ale také nové metody potřebné pro hru po síti a napojení na klientskou nebo serverovou část aplikace, z které bude dostávat informace o dění ve hře.

Spouštění hry probíhá stejně jako je tomu u třídy `HraHotSeat`. Pouze spoluhráči se nepřidávají do hry pomocí metody `vytvorHrace2`, ale pomocí `vytvorMPHrace` s parametry ID[6] hráče a nick. Vytvoření hráče 1 voláním metody vytvorHrace1, tedy hráče sedícího u PC, požaduje stejné parametry. Barva hráče je zde automaticky přiřazena dle hráčova ID. Pozice hráčů a překážek jsou získány stejně jako u hot-seat hry z mapy.

Dalším rozdílem oproti hře hot-seat je přítomnost nových signálů a slotů – signál `poziceHrace`, slot `zpracujPrijatouPozici` a slot `novaAkce`. Pomocí signálů informujeme server, případně klienta o událostech, které hráč ve hře provedl. Tyto události jsou poté převedeny na zprávy a odeslány ostatním hráčům. Pomocí slotů zpracováváme přijaté události o činnosti ostatních hráčů.

[6] Identifikační číslo hráče.

5.7 Síťová komunikace

Tato kapitola je věnovaná implementaci síťové komunikace. Implementace vychází z teorie, kterou jsme si popsali v kapitole 4. Případné změny zde budou zmíněny.

5.7.1 Serverová část

Serverová část programu se skládá ze dvou hlavních částí. Z části, která zpracovává nová připojení, přijímá a odesílá data. A části zpracovávající přijatá data a zobrazující formulář pro založení síťové hry, v kterém zakládající hráč nastaví parametry hry a vybere herní mapu.

Příjem a odeslaní dat

Třída představující server se jmenuje `TcpServer` a je zděděna od Qt třídy `QTcpServer`. To nám umožňuje přijímat TCP připojení na námi definovaném portu a adrese pomocí metody `listen()`. Po jejím zavolání začne server naslouchat a v případě příchozího požadavku je volána metoda `incomingConnection(int socketDescriptor)`. V této metodě definuje obsluhu této události. Parametr `socketDescriptor` je identifikátor příchozího připojení a je použit pro vytvoření socketu.

V naší třídě `TcpServer` je po zjštění žádosti o nové připojení vytvořen v metodě `incomingConnection` nový TCP socket pomocí třídy `QTcpSocket` a je mu přiděleno příchozí připojení pomocí metody `setSocketDescriptor`, kde je jako parametr použit již výše zmíněný `socketDescriptor`. Dále je vytvořeno vlákno, které dostane ukazatel na nově vytvořený socket. Hlavní povinností vlákna je kontrolovat, zda se na socketu nenacházejí data. Pokud zde data jsou, vyšle vlákno signál `priselPacket(QTcpSocket *,QByteArray)`. Parametry tohoto signálu jsou ukazatel na socket, z kterého data pochází, a vlastní přijatá data. Tento signál přijímá server a posílá ho dále. Ukazatel na socket i ukazatel na vlákno jsou uloženy v příslušných proměnných pro pozdější práci a dealokaci.

Další funkcí `TcpServeru` je odesílání dat. K tomu nám slouží dvě přetížené metody. První metodou je `odesliPacket`, tato metoda odesílá námi zadaná data pouze jednomu příjemci. Určit příjemce můžeme pomocí ID hráče nebo ukazatelem na socket, pomocí kterého s hráčem komunikujeme. Druhá metoda `odesliVsemPacket` odesílá data všem příjemcům nebo všem krom hráče definovaného zadaným socketem.

Zpracování dat (a založení hry)

O zpracování dat a zobrazení formuláře pro založení hry se stará třída `SitovaHraZakladani`. Také zajišťuje spuštění a ukončení hry.

Tato třída obsahuje instanci výše zmíněné třídy `TcpServer` s názvem `server` pro příjem a odeslání dat. Signál obsahující přijatá data, který vysílá `server`, je provázán se slotem `zpracujPacket`. Tento slot kontroluje identifikační kód zpráv a zprávy s jednoduchou obsluhou zpracuje sám. Pro ostatní volá příslušné obslužné metody.

Jako příklad si uvedeme první metodu, která je po připojení nového hráče volána, jedná se o `zpracujREQPacket`. Volání této metody, jak už název napovídá, je reakcí na příchozí `REQ` paket, neboli žádost o připojení do hry. Nejdříve se ověří, zda už není dosaženo maximálního počtu hráčů ve hře. Pokud je ve hře pro hráče místo, je mu přiřazeno identifikační číslo ID a to je odesláno žadateli v `ACK` paketu. Hráčův nick a id jsou uloženy do seznamu hráčů. Dále je nově připojenému

hráči odeslán seznam již připojených hráčů a ostatním hráčům informace o připojení nového hráče do hry. Po odeslání informací o hráči následuje odeslání informací o zakládané hře (typ hry, počet životů/času, jméno mapy). Nakonec je ve formuláři překreslen obsah widgetu `listHracu`, který má na starosti zobrazovat seznam připojených hráčů. Od této doby už je hráč členem hry a může s ostatními hráči chatovat pomocí widgetu na chat. Pokud ovšem už pro hráče není ve hře místo, je o tom informován `NACK` paketem.

Kontrola dostupnosti hráče

Při hraní hry může dojít k výpadku spojení s některým z hráčů. Je vhodné tuto situaci zachytit a ošetřit. V naší aplikaci odhalujeme pomocí odesílaní paketu PING.

Při připojení hráče do hry je vytvořen časovač, který je vázán na tohoto hráče. Časovač je spuštěn s hodnotou 2*`INTERVALPING`[7] při zapnutí hry. Pokud časovač není do této doby resetován, je volán signálem `timeout` slot `nedoselPing`, který odpojí hráče ze hry a rozešle informaci o odpojení hráče i ostatním hráčům. Na straně hráč běží po zapnutí hry také časovač, ale s délkou intervalu `INTERVALPING`, tedy poloviční dobou oproti časovači na serveru. Po každém vypršení tohoto času je časovač resetován a na server je odeslán paket `PING`. Ten tam resetuje časovač hlídající připojení hráče. Kontrola pomocí `PING` paketů končí až s dohráním hry.

Nastavení parametrů hry

Nastavení parametrů vytvářené hry probíhá pomocí vytvořeného uživatelského rozhraní. Nejdříve jsme vyzváni k zadání našeho nicku a portu, na kterém hra bude naslouchat. Po úspěšném zabrání portu se dostaneme do okna, kde můžeme nastavit, jaký typ hry budeme hrát, počet životů, délku hry, počet řízených střel a herní mapu, na které se bude hra odehrávat. Při změně jakéhokoliv nastavení se ihned změna odešle všem připojeným hráčům. Pokud by hráč nedisponoval mapou, která byla vybrána na serveru, je tato informace odeslána serveru pomocí chatovací zprávy a sám hráč je o této okolnosti upozorněn nápisem v místě, kde se nachází za normálních okolností jméno zvolené mapy.

Spuštění hry

Spustit síťovou hru může pouze zakladatel hry a to stisknutím tlačítka „Spustit hru“. Po stisku tohoto tlačítka dojde k odeslání paketu `START`. Poté následuje vytvoření hry a nastavení všech herních parametru. Nakonec se hra spustí.

5.7.2 Klientská část

Stejně jako serverová část se i tato dělí na dvě třídy – na třídu obstarávající připojení, odesílání a příjem dat. A na třídu, která obstarává zpracování dat a zobrazování příslušných oken uživateli.

Příjem a odesílání dat

Pro práci se síťovým připojením je k dispozici třída `TcpClient`. Jedná se o třídu obsahující jeden socket a jedno vlákno. A metody pro připojení k serveru, odpojení od serveru a odeslání paketu.

Pro připojení k založené hře se používá metoda `ustanovSpojeni`, kde jako parametry zadáváme adresu počítače a port, na kterém hra naslouchá. Poté se metoda pokouší o navázání spojení. Pokud k navázání nedojde do pěti sekund, je vráceno FALSE a pokus o spojení je ukončen.

[7] Časová konstanta v milisekundách definovaná v souboru sitovahra.h

Pokud ovšem k navázání dojde, je vytvořeno nové vlákno. Toto vlákno je určeno stejně jako v serverové části ke kontrole příchozích dat. Pokud tedy přijdou data, jsou odeslána spolu s adresou socketu pomocí signálu do třídy `TcpClient`. Odtud je signál `priselPacket` vyslán dále. Dále zde najdeme metodu `odesliPacket`, která slouží k odeslání dat, a metodu `odpojit`. Ta ukončí běh vlákna, odpojí nás od serveru a vytvořený socket odstraní.

Zpracování dat a připojení ke hře

O zpracování přijatých dat se stará třída `SitovaHraKlient`. Zároveň zajišťuje zobrazení uživatelských oken, spuštění a ukončení hry.

Třída `SitovaHraKlient` obsahuje instanci třídy `TcpClient` pro připojení k serveru a následnému příjmu a odesílání dat. Signál `priselPacket` je opět provázán se slotem, který má na starost zpracování příchozích paketů. Tímto slotem je `zpracujPacket`. Slot zkontroluje identifikační číslo paketu a podle něj volá příslušné metody na zpracování zbytku paketu. Některé pakety zpracovává sám, ale pouze ty jednoduché.

Po úspěšném připojení k serveru obdrží hráč seznam již připojených hráčů a hodnoty herních parametrů. Tyto údaje vidí hráč zobrazené v okně. Pokud na serveru nastane změna, jsou mu ihned nové parametry poslány a ihned zobrazeny. Dále v tomto okně může využívat chat pro komunikaci s ostatními připojenými. Hra je spuštěna až po přijetí paketu `START`. Poté se hra vytvoří, nastaví a spustí.

5.7.3 Tvorba paketů

Jednotlivé pakety jsou vytvořeny jako bytové pole `QByteArray`. Data jsou do pole nahrávána jako proud dat pomocí třídy `QDataStream`. Proměnné této třídy stačí předat ukazatel na bytové pole a poté do něho můžeme nahrávat data pomoci operátoru <<.

Přiklad vytvoření paketu

```
QByteArray SitovaHraZakladani::vytvorChatPacket(QByteArray
text){
    QByteArray block;
    QDataStream out(&block, QIODevice::ReadWrite);
    out.setVersion(QDataStream::Qt_4_6);
    out << (int) 0;
    out << (int) MESG;
    out << (int) this->idHrace;
    out << text;
    out.device()->seek(0);
    out << (int)(block.size() - sizeof(int));
    return block;
}
```

Zdrojový kód 5.1: Vytvoření paketu pro chat

Jako ukázku vytvoření paketu si popíšeme metodu pro vytvoření paketu pro chat, která vrací bytové pole obsahující kompletní paket připravený k odeslání (viz Zdrojový kód 5.1).

Nejprve vytvoříme proměnnou `block` a proměnnou `out`, které předáme adresu proměnné `block`. Dále upřesníme, jakou budeme používat verzi `QDataStream`. Poté postupně nahráváme další data v pořadí prázdný `int`, identifikační číslo paketu, ID hráče a text zprávy. Nakonec přesuneme ukazatel pozice v proměnné `out` na začátek. Zjistíme velikost uložených dat a tuto velikost zapíšeme do `out`. Zde díky posunu na začátek přepíše tato hodnota námi vloženou nulu,

která nám jenom držela místo pro nahrání této hodnoty a zajistí tak, že bude údaj o velikosti na začátku paketu. Všechny ostatní pakety jsou vytvářeny stejným postupem. Z kódu je zřejmé, že je ve stavbě paketu oproti návrhu v kapitole 4.2 změna. V konečné implementaci je na prvním místě v paketu umístěna velikost paketu, poté kód paketu a nakonec data.

Načtení informací probíhá obdobně. Stačí znát pouze strukturu paketu a poté načítat postupně všechny hodnoty z paketu do vytvořených proměnných pomocí operátoru >>.

5.7.4 Herní komunikace

Další důležitou částí síťové komunikace je samotná komunikace jednotlivých her během hraní. Tím myslíme ohlašování naší pozice ostatním hráčům a jejich informování, pokud dojde k výstřelu. Při teoretickém návrhu jsme si určili, že každá hra bude mít svůj časovač, který bude zajišťovat odesílání pozice ponorky s úhlem natočení v pravidelných intervalech. Pokud by došlo k výstřelu, byla by tato událost oznámena speciálním paketem.

Tento princip komunikace se podařilo naimplementovat a testovat. Při testech se ukázalo, že způsob komunikace nebyl vhodně zvolen. Pokud byl zadán interval zasílání moc krátký, kolem 1/20 sekundy, byl pohyb ponorky po hrací ploše viditelně trhavý. Důvodem mohl být velký přísun dat, který program nestačil zpracovávat. Při intervalu zasílání na 1/10 sekundy nebyl pohyb opět plynulý. Tentokrát ovšem z nedostatku dat.

Částečné řešení problému

Po zjištění problému, jsme se rozhodl zkusit navrhnout a naimplementovat nový způsob komunikace mezi hrami. Nový způsob už nebyl závislý na časovači. V předchozím návrhu byly ponorky neovládané hráčem přesouvány pouze na pozice, které byly přijaty po síti. V novém návrhu jsou těmto ponorkám posílány pouze informace, jakou akci provedl vzdálený hráč. Ponorka na akci reaguje stejně jako by tento hráč seděl u PC a ovládal ponorku pomocí klávesnice. Jedná se tedy o jakési vzdálené ovládání. Ponorce tedy přijde například informace o tom, že její majitel aktivoval pohyb dopředu. Ponorka tuto informaci zpracuje nastavením odpovídajícího příznaku. Tedy stejně jako by tato ponorka byla ovládána klávesnicí. Poté se o pohyb ponorky stará sama ponorka. Ukončení pohybu končí opět obdržením zprávy informující ponorku, že byla akce ukončena. Pro střelbu jsou stejně jako v předchozím návrhu použity jiné pakety.

Při testování se ovšem ukázaly nepřesnosti pohybu vlivem zpožďování paketů s informací o ukončení akce. Protože ponorka u hráče zastaví okamžitě, ale na vzdálených PC zastaví až po doručení paketu, který ji informuje o tom, že hráč ukončil akci. Ovšem díky již zmíněné prodlevě mezi skutečným ukončením akce a informováním ostatních hráčů o této skutečnosti, je ponorka u ostatních hráčů zastavena o několik pixelů dále. Tato odchylka závisí na délce trvání doručení paketu. Tyto odchylky se nám podařilo eliminovat pomoci synchronizačního paketu. Tento paket je odesílán při ukončení jakékoliv akce a obsahuje směr ponorky a souřadnice, na kterých došlo k přerušení akce (pohybu). Jedná se tedy vlastně o paket, který byl v předchozím řešení používán pro informování ostatních o pohybu ponorky. Reakcí na tento paket je upravení pozice ponorky na tyto souřadnice.

Pokud se tedy ponorka pohybuje vpřed a obdrží informaci o tom, že má pohyb ukončit, ihned tak učiní. Poté obdrží synchronizační paket, ten jí řekne, kde měla správně ponorka pohyb přerušit a na tuto pozici ji přesune. Tento pohyb (v tomto případě zpět) hráč většinou ani nezaregistruje. Jedná se o malé doladění pozice ponorky. Pokud by ovšem byla doba cestování paketu od jednoho hráče k druhému velká, byly by tyto úpravy polohy znatelné.

Nejprve se během testů zdálo vše v pořádku. Ovšem poté se ukázalo, že při častém měnění směru dochází k rozsynchronizování hry. Ponorka při těchto manévrech začne přeskakovat z místa na místo. Toto chování je nejspíše způsobeno již zmíněným zpožďováním synchronizačních paketů. Špatné chování jsme se pokoušeli odstranit přidáním synchronizačních paketů i před začátek zatáčecích úkonů, změnami v kódu i kombinací obou metod, ale bezvýsledně.

Návrh dalšího způsobu komunikace či vylepšení se nepodařilo nalézt, a tak je síťová hra hratelná pouze s tolerováním popsaného nedostatku v plynulosti pohybu ponorky.

6 Závěr

Obsah práce odpovídá jejímu zadání, což je vypracování multiplatformní 2D síťové hry s názvem Hon na ponorku. A to včetně návrhu pravidel hry, grafického zpracování a síťové komunikace přes vlastní komunikační protokol.

Cílem hry je eliminace protivníků. Do jedné hry se mohou zapojit maximálně čtyři hráči, kteří mají možnost volby ze dvou herních módů; hrou na čas a hrou na životy. Zpracování umožňuje hráčům jednoduché vytváření vlastních map pomocí XML souborů. V těchto souborech mohou pomocí jednoduchého zápisu definovat startovní pozice a umístění, rozměry a barvu překážek. Při hraní mají na výběr ze dvou druhů zbraní – neřízených torpéd a řízených torpéd, které pronásledují zaměřenou ponorku.

Hlavním přínosem práce pro mě bylo bližší seznámení se s propracovaným frameworkem Qt a praktické ověření získaných vědomostí. Dále zkušenost s tvorbou takto velkého projektu, v němž se více než u malých projektů projeví nutnost plánovat celý postup dopředu.

Hra splnila požadavek na multiplatformnost a byla úspěšně spuštěna jak na systému Windows (Windows 7 32b), tak Linux (Ubuntu 10.04 32b).

Vedle jmenovaného vylepšení hry o možnost tvorby vlastních map pomocí XML souboru byla jako další rozšíření úspěšně implementována hra dvou hráčů na jednom počítači, takzvaná hra hotseat.

Nedostatkem zpracování jsou problémy se synchronizací polohy ponorky při zatáčení u síťové hry. Vzniklý problém nebrání používání hry.

Dalším krokem v projektu by mělo být vylaďění problémové části síťové komunikace. Pokud by se povedlo nedostatek odstranit, mohlo by být dalšími kroky vytvoření aplikace představující herní server. K tomuto serveru by se hráči připojovali a mohli zakládat hry viditelné pro ostatní hráče. To by umožňovalo hru přes internet hráčům, kteří nemají veřejnou IP adresu. Server by mohl také obsahovat statistiky jednotlivých hráčů a vytvářet z nich žebříčky.

Literatura

1. **Ludačka, Radek.** QT framework - pomocník programátora. *swmag.cz.* [Online] 15. říjen 2009. [Citace: 10. květen 2010.] http://www.swmag.cz/546/qt-framework-pomocnik-programatora/.
2. **Davison, Andrew.** *Programování dokonalých her v Javě.* [překl.] Petr Krejčí. Brno : Computer Press, a.s., 2006. ISBN 80-7226-944-5.
3. **Blanchette, Jasmin a Summerfield, Mark.** *C++ GUI programming with Qt4.* Stoughton : Prentice Hall, 2006. ISBN 0-13-187249-4.
4. **Prata, Stephen.** *Mistrovství v C++.* [překl.] Sokol Boris a Vozák David. 3. aktualizované vydání. Brno : Computer Press, a.s., 2007. ISBN 978-80-251-1749-1.
5. **Nokia Corporation.** *Qt Reference Documentation.* [Qt Creator Help] 2009.

Seznam příloh

Příloha A

Obsah přiloženého DVD

- /SRC – Zdrojové kódy programu.
- /BIN/WIN – Spustitelná hra pro Windows (přeloženo ve Windows 7 32b).
- /DOC – Programová dokumentace vygenerovaná pomocí nástroje Doxygen.
- /BP – Bakalářská práce ve formátu PDF a DOCX.
- /MANUAL – Návod k překladu a spuštění hry.

Příloha B

Struktura komunikačních zpráv

REQ

ID zprávy (int)	hráčův nick (string)

– Žádost o připojení ke hře.

ACK

ID zprávy (int)	ID hráče (int)

– Přijmutí žádosti (REQ) na připojení ke hře. Informuje hráče o jemu přiděleném ID.

NACK

ID zprávy (int)

– Informuje hráče o zamítnutí jeho žádosti (REQ) o přijetí do hry.

INFO

ID zprávy (int)	druh hry (int)	počet času/zivotů (int)	počet střel (int)

– Informuje připojené hráče o nastavení zakládané hry.

HINFO

ID zprávy (int)	ID hráče (int)	nick hráče (string)

– Informuje hráče o připojení nového spoluhráče a nově připojeného o stávajících hráčích.

DISC

ID zprávy (int)	ID hráče (int)

– Oznamuje odpojení hráče podle ID.

MESG

ID zprávy (int)	ID hráče (int)	text zprávy (string)

– Chatovací zpráva. ID zde označuje odesilatele.

NOVAPOLOHA

ID zprávy (int)	ID hrače (int)	souřadnice x (int)	souřadnice y (int)	úhel (int)

– Určen k posílání přesné polohy hráče. V textu též zmiňován jako synchronizační paket.

START

ID zprávy (int)

– Zpráva odstartující hru.

PING

ID zprávy (int)	ID hráče (int)

– Zpráva informující o dostupnosti hráče.

NOVAAKCE

ID zprávy (int)	ID hráče (int)	typ akce (int)	začátek/konec akce (bool)

– Oznamuje ostatním, že hráč započal nebo ukončil nějakou akci.

MAPAVYBRANA

ID zprávy (int)	jméno souboru s mapou (string)

– Rozesílá informaci o zvolené mapě.

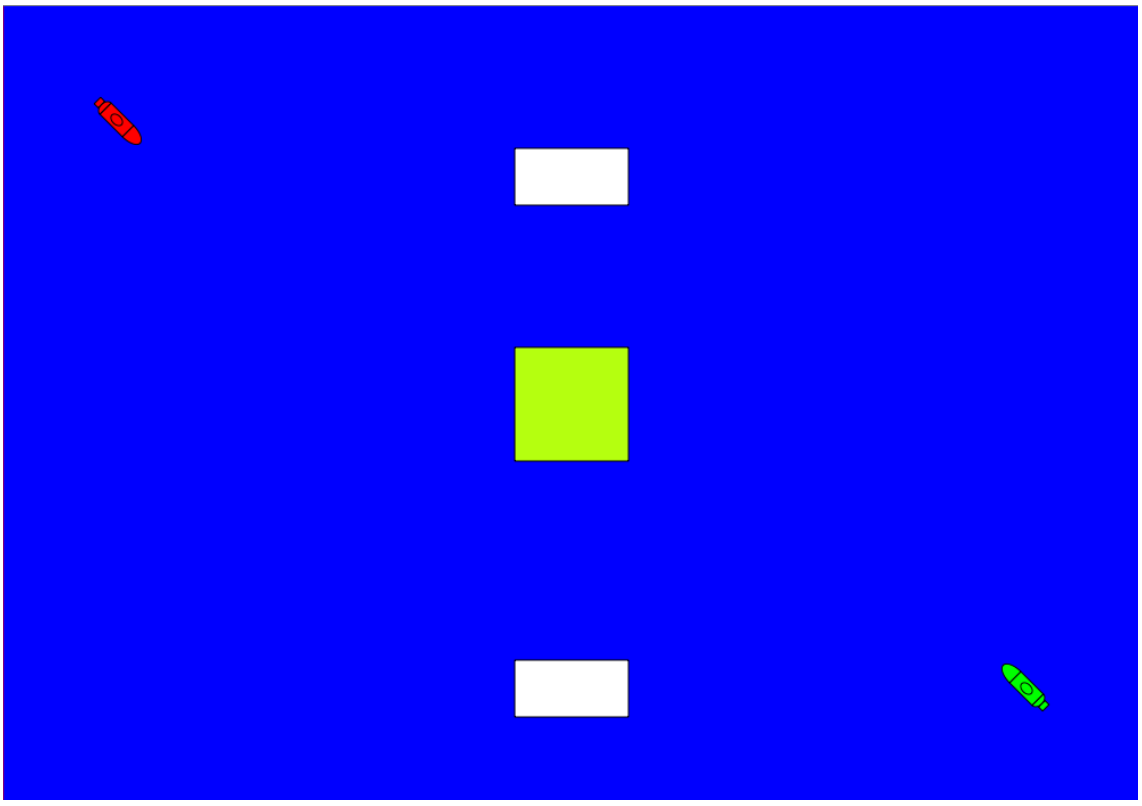
Všechny uvedené těla zpráv mají před sebou ještě hlavičku obsahující informaci o délce těla zprávy. Tato informace je typu `int`.

Příloha C

Ukázka XML souboru s mapou

```
<mapa>
  <jmeno>Mapa2</jmeno>
  <pozice id="0">
    <x>100</x>
    <y>100</y>
    <uhel>-45</uhel>
  </pozice>
  <pozice id="1">
    <x>900</x>
    <y>600</y>
    <uhel>135</uhel>
  </pozice>
  <pozice id="2">
    <x>900</x>
    <y>100</y>
    <uhel>45</uhel>
  </pozice>
  <pozice id="3">
    <x>100</x>
    <y>600</y>
    <uhel>-135</uhel>
  </pozice>
…(pokračování vpravo)…
```

```
…
<objekt>
  <x>450</x>
  <y>300</y>
  <w>100</w>
  <h>100</h>
  <barva>#B5FF0F</barva>
</objekt>
<objekt>
  <x>450</x>
  <y>125</y>
  <w>100</w>
  <h>50</h>
  <barva>#FFFFFF</barva>
</objekt>
<objekt>
  <x>450</x>
  <y>575</y>
  <w>100</w>
  <h>50</h>
  <barva>#FFFFFF</barva>
</objekt>
</mapa>
```



Výsledný vzhled navržené mapy