VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

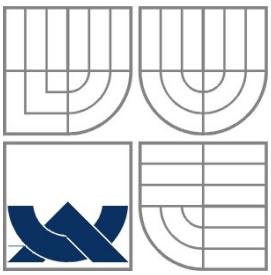
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

POČÍTAČOVÁ HRA VE 2D PRO LINUX

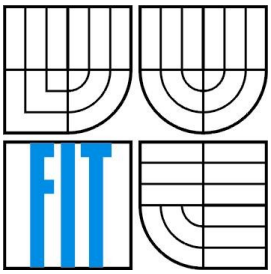
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAROSLAV ŠEVČÍK
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

POČÍTAČOVÁ HRA VE 2D PRO LINUX

COMPUTER GAME IN 2D FOR LINUX

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV ŠEVČÍK

VEDOUCÍ PRÁCE
SUPERVISOR

MGR. ADAM ROGALEWICZ, Ph. D.

BRNO 2009

Abstrakt

Práce se zabývá tvorbou umělé inteligence pro počítačem řízené prvky a jejich provedení do podoby počítačové hry. V práci jsou dále uvedeny typické postupy při řešení některých problémů rozhodování umělé inteligence a jejich implementace do funkční aplikace.

Abstract

This Bachelor's thesis is focused on artificial intelligence for computer controlled objects, represented by computer game. There are contained typical procedures to handle some behavior problems of intelligent objects and finally their implementation into fully functional application.

Klíčová slova

počítačová hra, umělá inteligence, algoritmy chování, hra na hrdinu (RPG)

Keywords

computer game, artificial intelligence, behavior algorithms, role playing game (RPG)

Citace

Jaroslav Ševčík: Počítačová hra ve 2D pro Linux, bakalářská práce, Brno, FIT VUT v Brně, 2009

Počítačová hra ve 2D pre Linux

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Adama Rogalewicza, Ph. D
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Ševčík
14.5.2009

Poděkování

Tímto bych rád poděkoval vedoucímu mé práce, panu Mgr. Adamu Rogalewiczovi, Ph. D za , jeho podporu a odborní pomoc při tvorbě této práce.

Obsah

1 Úvod....3
2 Návrh hry....4
2.1 Postavy....4
2.1.1 Hráčska postava (PC)....4
2.1.2 Nehráčska postava (NPC)....4
2.1.3 Povolania NPC....5
2.2 Herný svet....6
2.3 Pravidlá hry....6
2.3.1 Pravidlá boja....6
2.3.2 Zlepšovanie postavy....8
2.3.3 Špeciálne schopnosti....8
3 Umelá inteligencia....9
3.1 Metódy hľadania najkratšej cesty....9
3.1.1 Dantzigov algoritmus....9
3.1.2 Metóda Greedy Search....10
3.2 Algoritmy správania....11
3.3 Návrh GUI....13
4 Implementácia....15
4.1 QT....15
4.1.1 QT Image IO Extension Library....15
4.1.2 Signály a sloty v QT....15
4.2 Implementácia GUI....16
4.2.1 Trieda PlayGround....16
4.2.2 Trieda Layout....17
4.3 Implementácia zmeny správania....18
4.4 Komunikačný protokol....19
4.4.1 Protokol pre rozhovor....19
4.4.2 Protokol pre herné úlohy....20
4.5 Pohyb po hracej ploche....22
4.5.1 Pohyb hráča....22
4.5.2 Pohyb NPC....23
4.5.3 Prekonanie prekážok....23
4.5.4 Prechod oblastí....24

4.6 Bojový systém....25
4.6.1 Súboj....25
4.6.2 Smrť postavy....26
4.6.3 Zlepšovanie postavy....26
4.7 Implementačné problémy....27
5 Záver....29
Literatúra....30
Zoznam príloh....31
Príloha 1....32

1 Úvod

Umelá inteligencia je obor, ktorý sa v súčastnej dobe nachádza v mnohých vedných smeroch. Tento obor bol vytvorený na základe tvrdení, že ľudské konanie môže byť tak presne popísané, že sa dá simulovať strojom [1].

V oblasti informačných technológií sa umelá inteligencia využíva vo viacerých vetvách. Medzi najrozšírenejšie vetvy patria rozpoznávanie obrazu, reči, riadenie robotov a v neposlednom rade, stále rozvinutejšie odvetvie, hranie hier. Táto práca popisuje návrh a implementáciu počítačovej hry. V tejto hre bude hráč stretávať rôzne postavy, ktoré mu budú poskytovať rozlišné možnosti rozhovoru, budú reágovať na hráčove činy a svojím správaním budú dotvárať celkovú atmosféru hry. Úlohou hráča bude vyriešiť problémy, ktoré mu nastolí samotný dej hry. Niektoré problémy sa dajú vyriešiť jednoduchými rozhovormi, no iné len súbojmi „na život a na smrť“.

V kapitole 2 je rozobratý celkový návrh hry. V jednotlivých podkapitolách sú uvedené typy postáv ovládaných počítačom, herný svet a pravidlá hry. Kapitola 3 obsahuje informácie o rôznych algoritmoch využívaných v oblasti umelej inteligencie. Jedná sa o používané a funkčné algoritmy, ktoré boli použité aj pri tvorbe tejto práce. V kapitole 4 je popísaná implementácia programu. Sú to vlastné algoritmy, ktoré hra využíva k tomu, aby sa postavy vo svete správali podľa logického očakávania. Nachádza sa tu celkový popis tvorby samotného programu. Taktiež obsahuje návrh grafického užívateľského rozhrania a základný popis použitej grafickej knižnice.

2 Návrh hry

Po zvážení všetkých pre a proti som sa rozhodol vytvoriť hru typu RPG [5] (Role Playing Game alebo, voľne preložené, hra na hrdinu). K tomuto typu hry patrí veľká príprava pravidiel, ktorými sa bude samotná hra riadiť. Súčasťou tejto kapitoly je základný popis typov postáv a pravidiel hry. Implementačné detaily sú zaradené do kapitoly 4.

2.1 Postavy

V hre sú dva rôzne typy postáv. Prvý typ je taký, kde postavu ovláda hráč a nazýva sa hráčska postava (PC). Druhý typ je postava ovládana počítačom (NPC). V tejto kapitole si tieto pojmy priblížime a detailnejšie popíšeme.

2.1.1 Hráčska postava (PC)

Jedná sa o užívateľom ovládanú postavu [6]. Ovládanie pohybu prebieha pomocou klikania myši.
Hráč bude mať možnosť zdokonalovať svoju postavu počas hrania príbehu či už plnením rôznych úloh, alebo bojom s počítačom riadenými protivníkmi. Počas hrania hry bude stretávať rôzne postavy, ktoré budú dopĺňať hráčovu predstavu o svete, v ktorom sa jeho postava nachádza.

Hráčska postava sa od nehráčskej líši aj rôznym množstvom atribútov a možnosťou sa voľne pohybovat po hracej ploche. Na úvod má hráč pevne stanovené atribúty, ktoré má ale možnosť zmeniť počas hry.

2.1.2 Nehráčska postava (NPC)

NPC [7] (Non Player Character) je počítačom riadená postava. Má vlastnú umelú inteligenciu a reaguje na podnety spôsobené hráčom podľa preddefinovaných pravidiel. Môže mať priateľské alebo nepriateľské správanie.

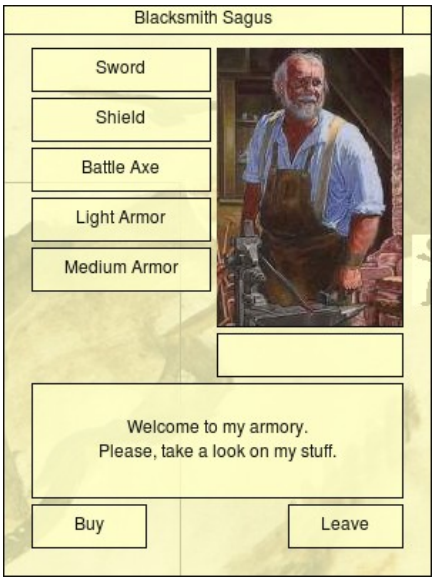
Priateľské správanie – postava sa správa k hráčovi ako k neznámemu človeku. Takéto správanie majú napríklad obchodníci, strážci mesta a podobne. Hra však umožňuje toto správanie zmeniť konaním rôznych skutkov zo strany hráča.

Nepriateľské správanie – postava s týmto typom správania okamžite na hráča útoči, ak sa hráč dostatočne blízko priblíži. Toto správanie má napríklad divá zver. Rovnako ako hráčska postava, má svoje špecifické štatistiky, ktoré sa líšia v závislosti od jej povolania.

2.1.3 Povolania NPC

V hre môže hráč naraziť na rôzne druhy NPC. Každá takáto postava má svoje unikátne postavenie v hre a s hráčom komunukujú rôznými spôsobmi.

- **Obchodník -** Postava obchodníka je dôležitá pre hráča predovšetkým kvôli tomu, že poskytuje hráčovi rôzne zbrane a zbroje. Ceny vecí, ktoré ponúka sa líšia v závislosti od samotnej veci. Čím je vec silnejšia, tým je vyššia aj jej cena.
- **Liečitel –** Ak hráč utrpí v boji zranenia, nevylieči sa automaticky. V takom prípade musí vyhľadať pomoc liečiteľa, ktorý jeho zranenia vylieči.
- **Strážca -** Jedná sa o silného bojovníka, ktorý chráni osadu pred nepriateľmi. Niektorí strážci dokážu za určitý obnos zvýšiť hráčove bojové schopnosti.
- **Sedliak –** Sedliaci sú rádoví obyvatelia mesta. Nijako nevynikajú v boji, ale môžu hráčovi zadať úlohu, ktorou môže získať užitočnú odmenu.
- **Nemŕtvy -** Hlavný nepriateľ hráča. Ak sa hráč dostane do blízkosti nemŕtveho, bez váhania na neho zaútočí.
- **Boss -** Najsilnejší nepriateľ v hre. Po jeho porazení končí hra.



Obrázok 1: Obchodník

Obrázok 2: Strážca

2.2 Herný svet

Ďaľšou dôležitou informáciou pri návrhu hry je herný svet. Ide o prostredie v ktorom je dej zasadený. Obsahuje hernú zápletku, prostredie a z toho vychádzajúce herné možnosti. Herný svet tejto hry je zasadený do zvláštneho pochmúrneho prostredia, odkiaľ hráč hľadá východisko. Herný konflikt hráčovi načrtne už samotné spustenie hry prostrednícvyvom krátkeho intra.

Vo svete bude stretať rôzne nezvyčajné bytosti, ale aj úplne bežných obyvateľov mesta. Hráč začína ako pútnik, ktorý po dlhšej dobe príde k malej neznámej dedinke. Pod vidinou jedla a vody sa rozhodne ísť jej smerom. Ako však zistí, táto dedinka má svoje problémy, ktoré môže vyriešiť len on. A tu sa začína samotná hra.

2.3 Pravidlá hry

Poslednou časťou fungujúcej hry je súbor pravidiel. Pravidlá musia byť vyvážené a ich príprava vyžaduje veľa pozornosti. Postava musí mať šancu hru dokončiť bez ohľadu na skúsenosť hráča.

2.3.1 Pravidlá boja

Zbrane - Pomáhajú hráčovi v boji proti svojim nepriateľom. Čím ťažšia zbraň, tým väčšie škody spôsobuje, ale o to je pomalšia. V hre je hráčovi k dispozícií viacero druhov zbraní. Hodnoty, ktorými ovplyvňujú jeho útočné čislo a rýchlosť útoku sú popísané v následujúcej tabuľke.

Typ:	Zranenie:	Rýchlosť
malé	3	350
jednoručné	5	600
obojručné	12	1100

Tabuľka 1: Zbrane

Obrázok 3: Ukážka zbraní

Brnenie - Chráni svojho majiteľa pred útokmi slabších zbraní. Hodnota jej obrany sa pripočíta k hráčovej schopnosti „Obrana“. Čím väčšiu má hráč obranu, tým ťažším cielom sa stáva pre svojich nepriateľov. Podobne ako u zbraní, aj tu je viacero druhov vybavenia, z ktorých si môže hráč vybrať. Dostupné zbroje sú uvedené v tabuľke.

Zbroj:	**Obrana:**
ľahká (kožená)	5
stredná (krúžková)	10
ťažká (plátová)	15

Tabľka 2: Zbroje

Obrázok 4:Ukážky zbrojí

Štíty - Štíty poskytujú aktívnu obranu, čo má za následok zvýšenie šance hráča na vykrytie nepriateľovho útoku. Čím väčší štít si hráč zabezpečí, tým bude jeho úspešnosť vykryť ranu a teda nedostať žiadne zranenie.

Typ:	**Obrana:**
malé	3
stredné	5
veľké	7

Tabuľka 3: Štíty

2.3.2 Zlepšovanie postavy

Hra poskytuje hráčovi svoju postavu zdokonaliť počas jej priebehu a to dvomi spôsobmi:

- **V boji**
- **U trénera**

V boji má hráč pri každom úspešnom zásahu nepriateľa šancu na zlepšenie svojej útočnej schopnosti. Pod pojmom úspešný zásah sa rozumie zásah, ktorý spôsobil nepriateľovi zranenie. Obrannú schopnosť získava naopak hráč po každej úspešne vykrytej rane, čo znamená po každej rane, ktorá mu nespôsobila zranenie.

V hre sa vyskytujú aj špeciálne postavy, ktoré dokážu hráčove schopnosti zvýšiť. Toto zvyšovanie vlastností je možné len do určitej hranice. Každý tréner, ktorý už hráča nedokáže nič naučiť, mu túto skutočnosť oznámi a odmietne mu poskytovať naďalej svoje služby.

2.3.3 Špeciálne schopnosti

Počas zdokonalovania svojej postavy má možnosť hráč získať jednu zo špeciálnych schopností. Tieto schopnosti pomáhajú hráčovi v boji podľa toho, akým štýlom hráč trénuje svoju postavu. Existujú tri špeciálne schopnosti v hre:

- **Berserker Rage** – túto schopnosť môže hráč získať len v prípade, že jeho útočná schopnosť je oveľa väčšia ako obranná. Jedná sa o silnú schopnosť, ktorá hráčovi umožňuje každým zásahom zraniť svoj ciel za dvojnásobok pôvodnej hodnoty. Jedná sa o pasívnu schopnosť, ktorá má vždy určitú šancu na úspech pri každom úspešnom zásahu.
- **Balancing Act** – túto schopnosť hráč získa, ak je jeho útočná a obranná schopnosť vyrovnaná. Dáva svojmu majiteľovi možnosť znížiť súperov počet životov na percentuálnu hodnotu svojich maximálnych životov. To znamená, že ak má hráč 50% životov, je schopný jednou ranou znížiť súperové životy o polovicu. Opäť sa jedná o pasívnu schonosť s určitou šancou na úspech.
- **Aegis Stance** – obranná schopnosť, ktorú hráč získa len v prípade, že hráčova obrana dosiahne velmi vysokú hodnotu a pritom bude mať pomerne nízky útok. Aktivuje sa len v prípade, že hráčov počet životov klesne pod 50%. Keď je táto schonosť aktivovaná, dostáva útočník zranenie naviac každým útokom na hráča.

3 Umelá inteligencia

Aby sa počítač dokázal správať tak, ako sa od neho očakáva, musíme mu toto správanie presne definovať. V tejto kapitole si priblížime známe algoritmy v oblasti umelej inteligencie, ktoré boli použité pri riešení tejto práce. Jedná sa o metódy hľadania najkratšej cesty, použité pre inteligenté prekonávanie prekážok na hracej ploche a algoritmus zmeny správania, ktorým sa riadia nehráčske postavy. Táto kapitola sa preto sústredí na tieto dve časti.

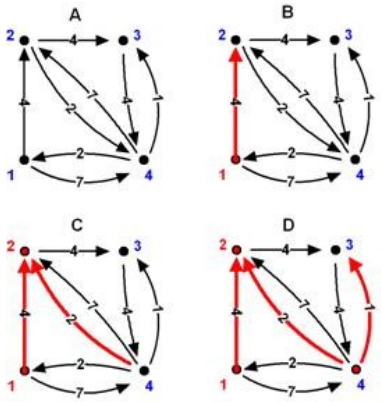
3.1 Metódy hľadania najkratšej cesty

Tieto metódy sa používajú v prípade, že chceme, aby sa nami navrhnutý systém pohyboval čo najefektívnejšie. V súčastnosti existuje rada rôznych spôsobov, ako tohto cieľa dosiahnúť. Pre tento projekt boli najvhodnejšie tieto 2 metódy:

1. **Dantzigov algoritmus [2]**
2. **Metóda Greedy search [3]**

3.1.1 Dantzigov algoritmus

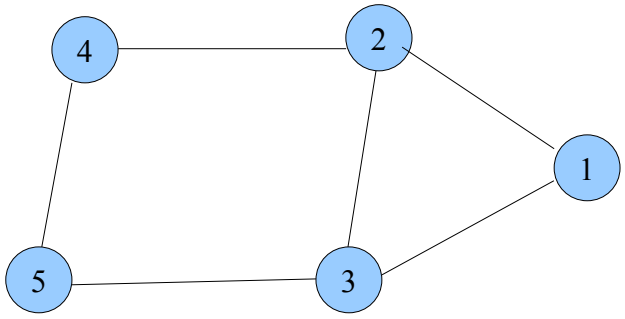
Jednou z osvedčených a používaných metód pre riešenie problému najkratšej cesty je Dantzigov algoritmus [2]. Jeho princíp spočíva v tom, že určíme hodnotu jednotlivým cestám a na základe tejto hodnoty sa počíta najkratšia vzdialenosť. Hodnota sa volí podľa potreby buď ako vzdialenosť, cena alebo časová náročnosť na prekonanie tejto hrany. Program hľadá možné kombinácie a vyberie najkratšiu z nich.



Obrázok 5: Dantzigov algoritmus

3.1.2 Metóda Greedy Search

Metóda Greedy Search [3] ohodnocuje uzly heuristickou funkciou, tj. odhadovanou cenou z daného uzlu do cieľového uzlu. K expanzii vyberá uzol s najmenším ohodnotením. Najpoužívanejšia heuristika je priama vzdiaľenosť medzi bodmi a to aj v prípade, že medzi nimi neexistuje priame spojenie.



Obrázok 6: Greedy Search

Počiatočný uzol:	**Vzdialenosť k uzlu 1:**	**Vzdialenosť k uzlu 2:**
1	-	30
2	30	-
3	35	35
4	60	45
5	70	40

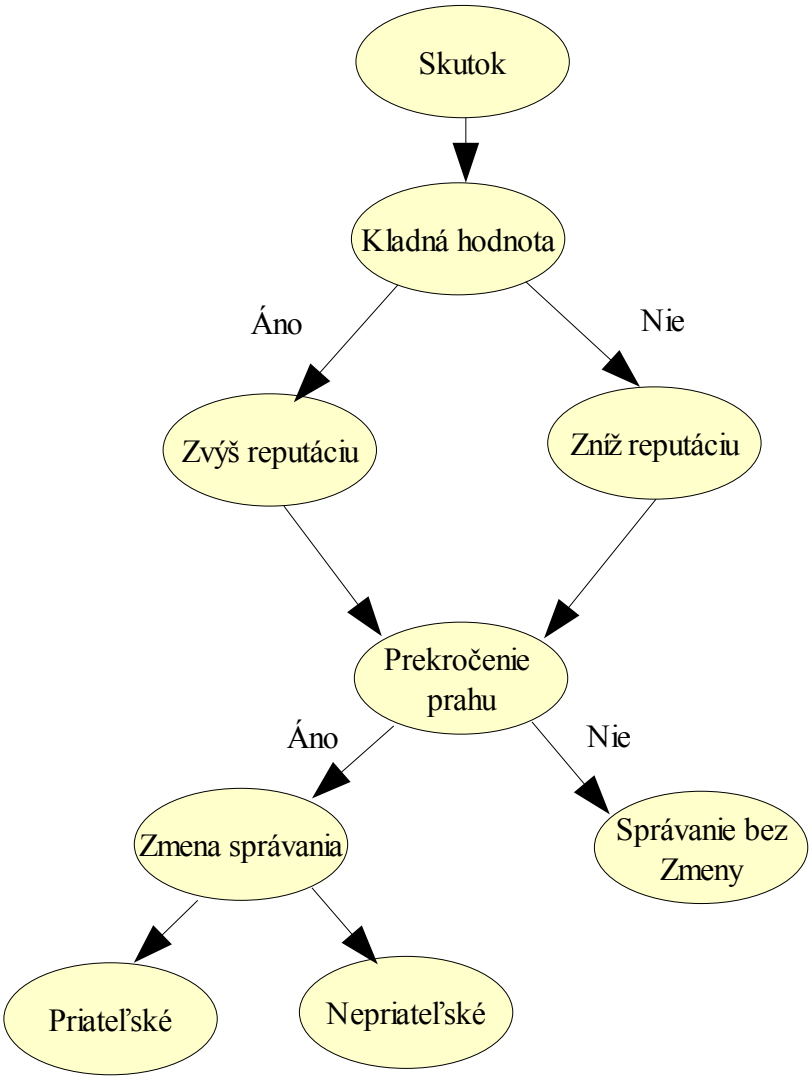
Tabuľka 4: Vzdialenosti k uzlom 1,2

Túto tabuľku vypočíta algoritmus, ktorý odhadne priamu vzdialenosť od požadovaného uzla ku všetkým ostatným. V tabuľke su zobrazené odhadnuté vzdialenosti len k uzlom 1 a 2. Metóda Greedy Search je používaná v situáciách, keď máme známy počet uzlov a ak tento počet nieje veľký, pretože pri väčšom počte uzlov stráca táto metóda efektivitu, kvôli väčšej časovej náročnosti.

3.2 Algoritmy správania

Každý inteligentný systém potrebuje presne určit svoje správanie. Aby sme dosiahli požadovaného správania, musíme navrhnúť algoritmy, ktoré budú systém reprezentovať.

Pre riešený projekt sme navrhli algoritmus riadenia založený na následujúcich princípoch. Každá postava má svoje atribúty, medzi ktorými je aj údaj o správaní sa voči hráčovi (reputácia), ktorý rozhodne, či sa bude postava správať k hráčovi priateľsky alebo nepriateľsky. Každým skutkom hráča sa táto hodnota v závislosti od jeho závažnosti rôzne mení. Popis tohto správania je daný následujúcim grafom.



Obrázok 7: Graf algoritmu správnia

Popis grafu:

Vstup:

Vstupom do systému je skutok hráča. Podľa závažnosti skutku, sa priradí určitá hodnota, s ktorou sa ďaľej pracuje. Medzi takéto skutky sa počíta napríklad porazenie nepriateľa, či splnenie úlohy. Viac informácií je uvedených v pravidlách hry v kapitole 2.3.

Výstup:

Výstupom je informácia, či sa skúmanému objektu zmení typ správania, alebo ostane bez zmeny, na základe čoho sa spustia príslušné algoritmy popísané v kapitole 4.

Priebeh:

1. na vstup príde hodnota.
2. ak je táto hodnota kladná, zvýši sa reputácia daného objeku voči hráčovi, alebo naopak, ak je záporná, tak sa táto hodnota zníži.
3. skontroluje sa, či nedošlo k prekročeniu nastaveného prahu. Ak k tejto udalosti nedošlo, v systéme sa neudeje žiadna zmena. Ak hodnota prekročila prah, nastane zmena správania.
4. Ak je rozdiel táto hodnota kladná, zmení sa správanie na priateľské, ak je záporná, naopak na nepriateľské.

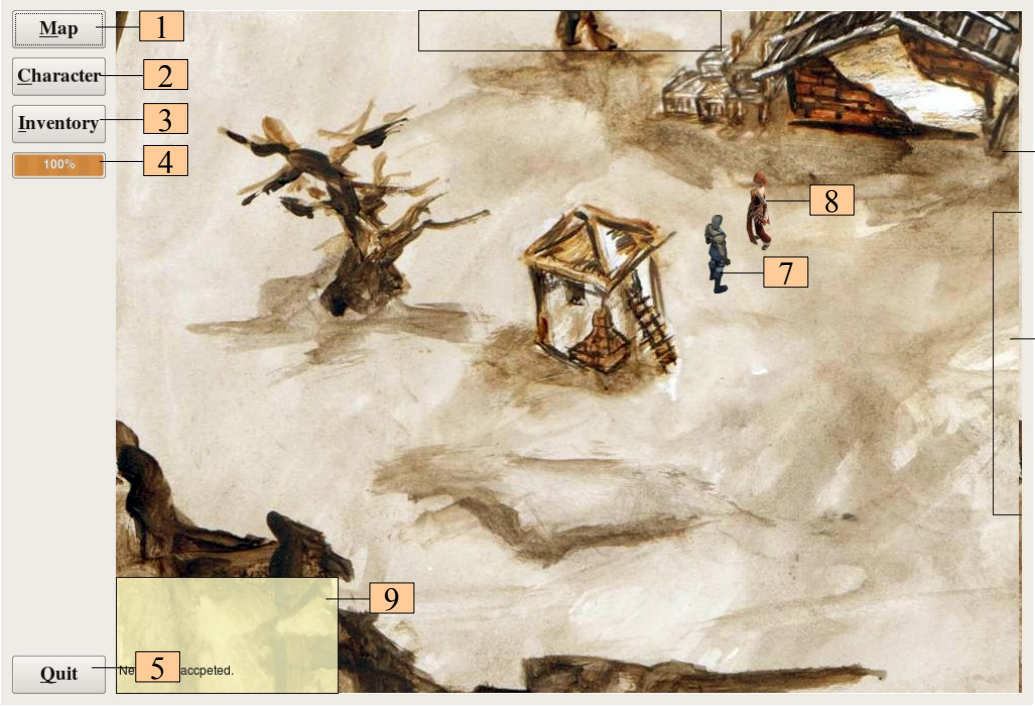
3.3 Návrh GUI

Aby mohol komunikovať počítač s užívateľom a naopak, je vhodné zabezpečiť uživateľské rozhranie. Pre vytvorenie grafického rozhrania sa používajú špeciálne knižnice. Jednou z takýchto knižníc je QT [4], v ktorom je projekt vytvorený.

Rozhranie každej hry by malo spĺňať následujúce požiadvky:

1. **prehľadnosť**
2. **jednoduchosť**
3. **možnosť intuitívneho ovládania**

Navrhnuté rozhranie obsahuje panel s herným menu a hlavné okno, v ktorom sa bude samotná hra odohrávať. Panel s herným menu obsahuje položky pre rýchlejšiu a interaktívnejšiu prácu s programom. Obsahuje informácie o postave, mapu hernej plochy a herný inventár. Na uvedenom obrázku je konečné GUI hry.

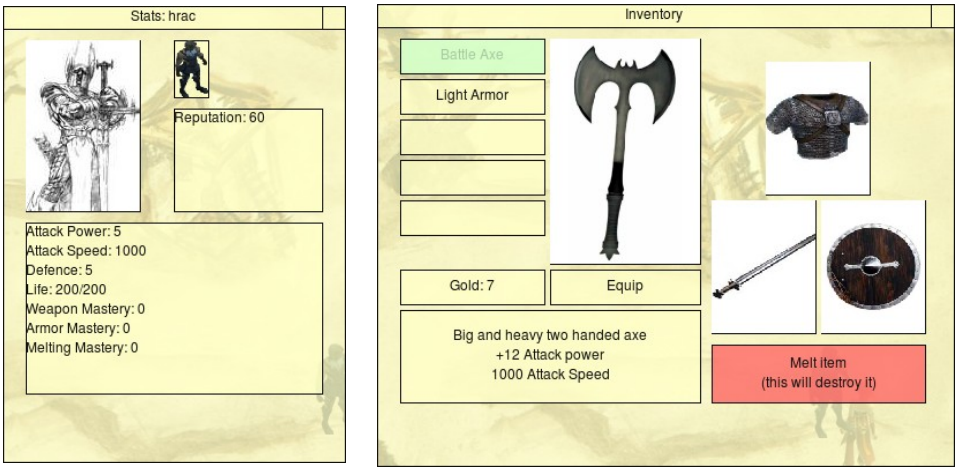


Obrázok 8: Grafické uživateľské rozhranie

Legenda:

1. **Map** – po kliknutí na toto tlačítko sa na hracej ploche objaví okno s mapou hernej plochy a momentálnou pozíciou hráča.

2. **Character** – toto tlačítko po kliknutí zobrazí informácie o postave, kde hráč nájde štatistiky svojej postavy, informácie o svojich schopnostiach a reputácii vo svete
3. **Inventory** – inventár je miesto, kde hráč uchováva svoje predmety. Má limitovný počet položiek a dovoľuje hráčovi jednotlivé veci používať, či ich taviť, z čoho získa nejaký peňažný obnos.
4. **Ukazateľ života** – Ukazuje momentálny stav života hráčskej postavy. Tento počet je prepočítavaný na percentuálnu hodnotu a hráč teda vidí koľko života mu ostáva.
5. **Tlačítko Quit** – tlačítko je určené na vypnutie hry. Po jeho stlačení sa hra okamžite vypne.
6. **Hracia plocha** – na hracej ploche sa odohráva celá hra. Hráč sa v nej môže pohybovať a komunikovať so svojím okolím.
7. **Hráčska postava –** postava ovládaná hráčom
8. **Nehráčska postava–** postava ovládená počítačom
9. **Okno s udalosťami** – v tomto okne sa zobrazujú všetky dôležité udalosti v hre. Môže sa tu zobrazovať napríklad správa o prijatí novej úlohy, informácie o zranení a pod.
10. **Prechod oblastí –** ak hráčska postava príde do jednej z týchto oblastí, dôjde k presunu do inej časti mapy, čo v tomto prípade znamená premiestnenie sa do inej časti mesta.



Obrázok 9: Okno Character (vľavo) a okno Inventory (vpravo)

4 Implementácia

V tejto kapitole sú obsiahnuté základné implementačné algoritmy a popisy ich funkčnosti. Celý projekt je naprogramovaný pomocou jazyka C++. Uživateľské rozhranie je vytvorené pomocou knižnice QT [4]. Každý objekt, ktorý využíva sloty a signály musí byť definovaný ako objekt knižnice QT, čoho dosiahneme pomocou makra Q_OBJECT [8]. Týmto makrom aktivujeme prjímanie a odosielanie signálov z/do tejto triedy. Signály sú základný komunikačný prvok s ostatnými triedami a môžu posielať potrebné údaje do slotov, ktoré tieto údaje dokážu spracovať ako štandardná funkcia jazyka C++.

4.1 QT

Qt je C++ GUI toolkit pre vývoj pod systémom Unix/Win32. Qt je vyvíjané firmou Troll Tech v Nórsku. Vývoj Qt bol zahájený v roku 1992 a o dva roky neskôr bol založený Troll Tech. O pár rokov neskor táto firma vydáva rozšírenia pre QT. Pre túto prácu je dôležité predovšetkým rozšírenie QT Image IO Extension Library. QT toolkit je distribuovaný buď pod komerčnou licenciou za $1290, alebo pod tzv. free licenciou, pre vývoj nekomerčných aplikácií.

4.1.1 QT Image IO Extension Library

Knižnica rozšiřujúca QT o prácu s rôznymi obrazovými formátmi. Momentálne podporuje JPG a PNG. Dôvodom existencie tejto knižnice je snaha o možnosť dynamického pridávánia podpory pre obrazové formáty bez nutnosti meniť zdrojový kód QT. Aplikácia vyžadujúca dodatočné obrazové formáty môže použiť funkciu **qInitImageIO().**

4.1.2 Signály a sloty v QT

GUI komponenty vytvorené pomocou QT spolu nekomunikujú automaticky po ich vytvorení. Je nutné ich prepojiť pomocou signálov a slotov, ktoré nám zabezpečia komunikáciu.

Myšlienka je následujúca: pomocou špecifikácie prístupu a slova **slots:** objekt definuje sloty. Potom pomocou slova **signals:** definuje signály. Signál nesmie mať návratovú hodnotu. Pomocou funkcie **QObject::connect** prepojíme signál určitého objektu so slotom iného objektu, čím nám vznikne komunikácia medzi týmito dvoma objektami.

4.2 Implementácia GUI

Základným kameňom GUI je trieda PlayGround v súboroch playground.cpp a playground.h, viz. Kapitola 4.2.1. Táto trieda obsahuje informácie o tom, ako sa jednotlivé objekty vykreslujú na hracej ploche.

Ďaľšou dôležitou triedou je trieda Layout (viz. 4.2.2). Táto trieda obsahuje celkové rozloženie objektov v programe. Má na starosti zobrazovanie herného menu a aj samotnej hracej plochy. Definuje veľkosť okna a pozíciu jednotlivých objektov tak, aby výsledné rozmiestnenie spĺňalo očakávané požiadavky (jednoduchosť, prehľadnosť a možnosť intuitívneho ovládania).

4.2.1 Trieda PlayGround

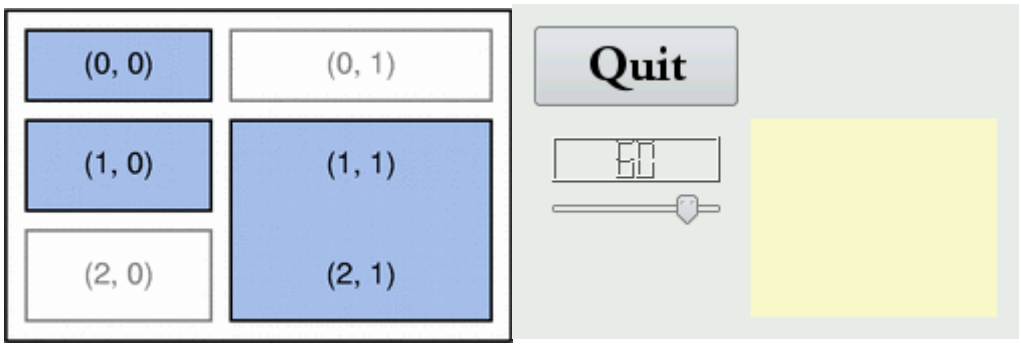
Trieda plne využíva implementačných možností knižnice QT. Je definovaná ako samostatný widget, ktorý sa vykresluje podľa definovaných pravidiel. Najskôr sa vykreslí pozadie podľa identifikačného čísla oblasti, v ktorej sa nachádza hráčska postava. Toto číslo je unikátne pre každú oblasť a tvorí ho súčet jej vertikálnej a horizontálnej zložky hernej plochy. Bližšie si túto metódu indexovania popíšeme v následúcej kapitole.

Vykreslovanie objektov

Objekty na hracej ploche sa zobrazujú pomocou triedy QPainter. Každý objekt musí mať definovanú plochu do ktorej sa môže vykresliť.Existuje viacero metód tejto triedy, no využívame len tie, ktoré sú potrebné na vyrkeslovanie geometrických tvarov a obrázkov. Pri každej zmene objektu sa musí použiť funkcia update(), ktorá aktivuje udalosť paintEvent a tá prekreslí plátno s novými atribútami.
Je to veľmi často využívaná funkcia, hlavne v prípade, keď sa jedná o aplikáciu bežiacu v reálnom čase.

4.2.2 Trieda Layout

Táto trieda nám zabezpečuje rozloženie kompotnent v samotnom programe. Obsahuje všetky komponenty, ktoré vidí užívateľ pri chode programu. Na usporiadanie samotných komponent využíva triedu QGridLayout, ktorá na základe súradnicového systému dokáže jednotlivé objektu usoriadať do mriežky. Táto trieda nieje samostatný widget, ale úplne odlišný objekt, ktorý dokáže spravovať potomkov akéhokoľvek widgetu [9]. Na obrázku je zobrazená funkčnosť a využitie tejto triedy.



Obrázok 10: definícia pomocou QGridLayout(vľavo) a skutočný výstup

4.3 Implementácia zmeny správania

Na základe algoritmu z kapitoly 3.2.1 bol implementovaný algoritmus, ktorý má za úlohu počítačom ovládaným objektom povedať, ako sa majú k hráčovi správať. Ako prvá sa zisťuje hodnota reputácie, čo je vlastne celé číslo od 0 do 100 a vyskypuje sa ako u PC, tak u NPC. Každé NPC má v sebe zabudovaný vnútorný časovač a po jeho vypršaní skontroluje, či sa nezmenila reputácia u hráča. Ak nastane zmena, prepočíta, či rozdiel hodnôt reputácií PC a NPC nepresahuje hodnotu 50. Ak túto hodnotu presahuje, nastaví sa správanie na agresívne. Ak sa hráč priblíži do určitej vzdialenosti k postave s týmto typom správania, automaticky na neho zaútočí.

Celý algoritmus je implementovaný ako súbor podmienok a cyklov a riadi ho trieda PlayGround. V skratke by sa dal popísať následujúcim pseudokódom:

```
if (rozdiel reputacie >= 50)  nastav agresivne spravanie;
{
      //nastavime dosah agresie
      if (vzdialenost k hracovi < = 250) zautoc na hraca;
}
else nastav priatelske spravanie.
```

Príklad 1: Pseudokód zmeny správania

Pri každom type správania má hráč iné možnosti ako na NPC reágovať. Ak je priateľské, môže začať rozhovor a získať informácie, ktoré mu poskytuje. Všetky rozhovory sú tvorené komunikačným protokolom, vytvoreným práve preto, aby spĺňali požiadavky hry. Tento protokol je popísaný v následujúcej kapitole. Ak je správanie nepriateľské, nemá až toľko možností, ale môže na postavu zaútočiť, čím spustí bojové algoritmy a výpočty.

4.4 Komunikačný protokol

Každá postava má definovaný rozhovor, ktorý prebieha na základe komunikačnáho protokolu.Tento protokol riadi reakciu na hráčové odpovede. Celý protokol je definovaný v triede Chat. Funguje tak, že hráč zadá odpoveď, tá sa rozkóduje a vyhľadajú sa v nej kľúčové slová. Ak sa také slovo najde, postupuje sa ďalej podľa presných pravidiel.

4.4.1 Protokol pre rozhovor

Program obsahuje dva protokoly. Ten prvý používajú všetky NPC, s ktorými je možné viesť rozhovor. Obsahuje svoje kľúčové slová, na základe ktorých sa rozhoduje, ako ďaľej v rozhovore pokračovať.

Kľúčové slová pre rozhovor sú následujúce:

- **end**
- **next** [end|quest]
- **karma**<hodnota>
- **cash**<hodnota>
- **atk**<hodnota>
- **def**<hodnota>
- **life**<hodnota>
- **heal**<hodnota>

end – zavolá funkciu, ktorá okamžite ukončí rozhovor a zrúší zobrazovanie okna s rozhovorom.
next - zavolá funkciu, ktorá načíta ďalšie údaje potrebné pre rozhovor. Môže obsahovať voliteľné parametre, ktoré presne určia, čo sa po načítaní parametrov má stať. Parameter end ukončí po načítaní textu rozhovor. Parameter quest povie programu, že ďalší text má načítať z iného súboru.
karma – zmení hráčovu reputáciu v hre o hodnotu udávanú v povinnom poli
cash – pripočíta alebo odráta sumu peňazí z hráčovho konta. Množstvo závisí od hodnoty udanej za týmto kľúčovým slovom
atk – zvýši/zniži hodnotu hráčoveho útoku o danú hodnotu
def – zvýši/zniži hodnotu hráčovej obranu o danú hodnotu. Kľúčové slová atk a def používajú postavy, ktoré majú úlohu trénera.

life – zvýši maximálny počet hráčových životov o hodonotu udanú v povinnom parametri
heal – vylieči hráčove zranenia o danú hodnotu. Toto kľúčové slovo je používané postavami s úlohou liečiteľa.

Pre lepšie pochopenie protokolu, uvádzam príklad:

```
9
Hello traveler. I have here one problem. I am on duty and I have
something, I have to do asap. Can you help me? I can pay 10 gold.
That is interesting, please, tell me more.|next|quest|
No, I am not interested|next|end|
Did you change your decision?
Yes, I have more time now. I can help you, tell me more|quest|
No, I don't have time for that right now.|end|
Quest|1
questAnsw1
questAnsw2
```

Príklad 2: Komunikačný protokol

Tento text obsahuje rozhovor postavy strážca Balth s hráčom. Po začatí rozhovoru sa načíta celý súbor do pamäti a rozkóduje sa podľa protokolu. Prvý riadok súboru je číslo, ktoré nám označuje počet riadkov rozhovoru. Následuje úvodný text a dve odpovede hráča. Vždy musí následovať táto trojica údajov, aby dokázal komunikačný algoritmus správne rozkódovať text určený pre NPC a hráča. Text NPC je označený modou farbou.

Na konci každého textu sú oddelené kľúčové slová zvislou čiarou. Každé kľúčové slovo a jeho parametry musia byť ohraničené týmito čiarami z oboch strán.

4.4.2 Protokol pre herné úlohy

Po obdržaní parametru **quest** sa začína riadiť rozhovor pomocou iného protokolu. Tento protokol dokáže využívať aj predchádzajúci protokol, ale obsahuje aj ďalšie kľúčové slová, ktoré definujú stav úlohy, ktorá je pre hráča pripravená.

Protokol je následujúci:

- **quest**<hodnota> [complete]
- **take**<hodnota>
- **isTaken**<hodnota>

quest – kľúčové slovo quest v tomto protokole plní inú funkciu ako v protokole pre rozhovor. Po jeho prijatí sa načíta rozhovor pre úlohu identifikovanú hodnotou v povinnom paramtetri. Voľitelným parametrom tohto slova je údaj complete, ktorý nastaví úlohu ako splnenú.
take – vyskytuje sa výhradne v odpovediach hráča a pomocou hodnoty identifikuje číslo úlohy, ktorú hráč prijal. Prijatie úlohy znamená u niektorých NPC zmenu rozhovoru.
isTaken – kontroluje, či už hráč danú úlohu neprijal.Po prijatí sa zavolá funkcia, ktorá úlohu vyhľadá a skontroluje príznak taken základe identifikačného čísla z povinného parametru . Potom vráti jeho hodnotu (true/false) funkcii, ktorá ju ďaľej spracuje.

Podľa tohto a predchádzajúceho protokolu sa riadia všetky hráčske úlohy v hre. Príklad využitia protokolu pre správu herných úloh je následujúci.

```
Please deliver this letter to Raggra. And please, hurry.
Please, hurry...
Thanks, here is your reward.
Nice to meet you again.

OK, I will take it to her.|take|1|
I'm workin' on it.
*Take sack with gold coins*|cash|10|
Nice to meet you too.
```

Príklad 3: Protokol pre herné úlohy

Súbor s úlohou je rozdelený do 8 riadkov, pričom prvé štyri obsahujú text určený pre NPC a druhá štvorica obsahuje text určený pre hráča.. Vždy sa v rozhovore načíta jeden riadok z každej štvorice.

Prvý riadok štvorice sa objaví, keď začne úloha. Program rozkóduje text a objaví, že sa tu vyskytuje kľúčove slovo **take,** ktoré spracuje a označí úlohu s identifikačným číslom 1 ako prijatú. Jedná sa teda o zadanie úlohy pre hráča. Druhá dvojica sa zobrazí, ak úloha ešte nieje ukončená, ale už je prijatá. Postava nabáda hráča, aby na úlohu nezabudol. Následujúca dvojica sa objaví v prípade, ak je úloha hotová a hráč si ide pre svoju odmenu. V tomto prípade využíva predchádzajúceho protokolu, ktorý zabezpečí, že hráčovi sa na konto pripíš 10 zlatiek. Posledná dvojica sa objaví, ak hráč už dostal svoju odmenu a znovu začal rozhovor s touto postavou. Postava hráča poznáva a správa sa voči nemu priateľsky, pretože jej hráč v minulosti pomohol.

4.5 Pohyb po hracej ploche

Postava musí byť schopná dostať sa na miesto, ktoré hráč označil. V hre je implementovaných niekoľko algoritmov, ktoré zabezpečujú presun postáv po hracej ploche. V tejto kapitole si priblížime niektoré z nich.

4.5.1 Pohyb hráča

Hráč sa pohybuje pomocou kliknutia na hraciu plochu ľavým tlačítkom myši. Ak neklikol na statický objekt (budova, prekážka), tak sa začne vypočítavať cesta, ktorú postava musí prejsť. Zákaldnou triedou, ktorá riadi presun a jeho zobrazovanie je trieda PlayGround. Pohyb začne tým, že sa spustí vnútorný časovač objektu Character. Každým jeho tikom sa načíta pozícia hráča na mape a porovnáva sa s cieľovou pozíciou. Algoritmus vypočítava každý následujúci krok postavy na základe odpočtu vertikálnej a horizontálnej zložky oboch súradníc.Podľa tohto výpočtu sa rozhoduje, ktorým smerom sa postava pohne. Algoritmus je možné zapísať následujúcim pseudokódom:

```
X = horizontalna_zlozka_pozicie_postavy
Y = vetikalna_zlozka_pozicie_postavy
Xf  =  horizontalna_zlozka_konecnej_pozicie
Yf  =  vertikalna_zlozka_konecnej_pozicie
if (X < Xf) presun_doprava()
if (X > Xf) presun_dolava()
if (X == Xf) presun_po_vertikalnej_osi()
if (Y < Yf) presun_dole()
if (Y > Yf) presun_hore()
if (Y == Yf)presun_po_horizontalnej_osi()
```

Príklad 4: Pseudokód pohybu postavy.

4.5.2 Pohyb NPC

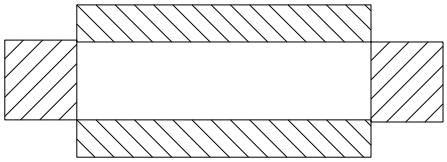
Pohyb počítačom riadenej postavy funguje na rovnakom princípe ako u hráča, s tým rozdielom, že sa konečná pozícia nezadáva kliknutím myši, ale algoritmom, ktorý určí jednu z možných pozícií na hracej ploche. Postava sa môže riadiť jedným z dvoch rôznych spôsobov.

Prvý vyberá náhodne súradnice, kam sa postava presunie. Vynecháva pritom súradnice, ktoré sú obsiahnuté v jednotlivých statických objektoch na hracej ploche. Druhým spôsobom je nastavenie pevných bodov, po ktorých sa bude postava pohybovať. Postava vždy dojde k jednému bodu, načíta ďalší bod, ak taký existuje a presunie sa k nemu. Toto správanie je použité predovšetkým u postáv, ktoré majú za úlohu strážiť nejakú časť územia.

4.5.3 Prekonanie prekážok

Po tom, ako sme definovali pohyb postáv k svojmu cieľu bolo nutné navrhnúť algoritmus, ktorý riadi správanie postavy, ktorá pri svojom pohybe narazí na prekážku.Každá postava má v sebe zabudovanú sondu, ktorá zisťuje existenciu objektov vo svojom okolí.

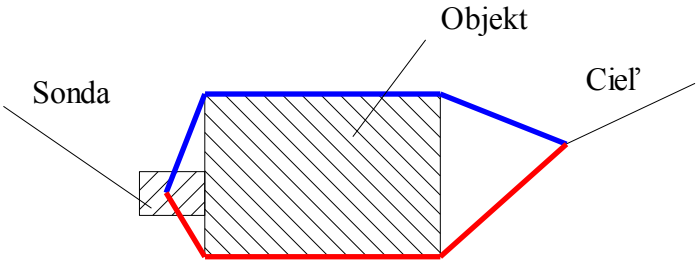
Sonda sa skladá zo štyroch objektov, ktoré snímajú objekty zo všetkých štyroch strán. Aby nedochádzalo k problémom pri lineárnom pohybe objektov, musela byť zostavená tak, aby sa v jednom momente dotýkala snímaného objektu vždy len jedna strana sondy.



Obrázok 11: Sonda okolitých objektov

Týmto tvarom sondy eliminujeme dotyk dvoch strán naraz pri posune nahor a nadol. Pri posune do strán je tu možnosť, že k takému dotyku dojde a preto bolo treba túto možnosť eliminovať priamo v algoritme, ktorý automaticky vypočíta ďalší krok a okamžite ho vykoná.

Takáto sonda sa nachádza vždy pri nohách postavy a ak sa jedna z jej strán dotkne objektu, začne sa počítať cesta okolo objektu. Pri výpočte sa počíta najkratšia cesta k cieľu pomocou Dantzigovho algoritmu [2]. Ohodnotia sa hrany vedúce po obvode obchádzaného objektu až k cieľu. Hodnotami sú vzdialenosti medzi jednotlivými bodmi prekážky. Výpočet prebieha ako súčet všetkých týchto hodnôt a vytvorí sa tabuľka. Od tohto momemntu sa algoritmus správa ako metóda Greedy search, pretože poznáme vzdialenosti od jednoho bodu k ostatným.

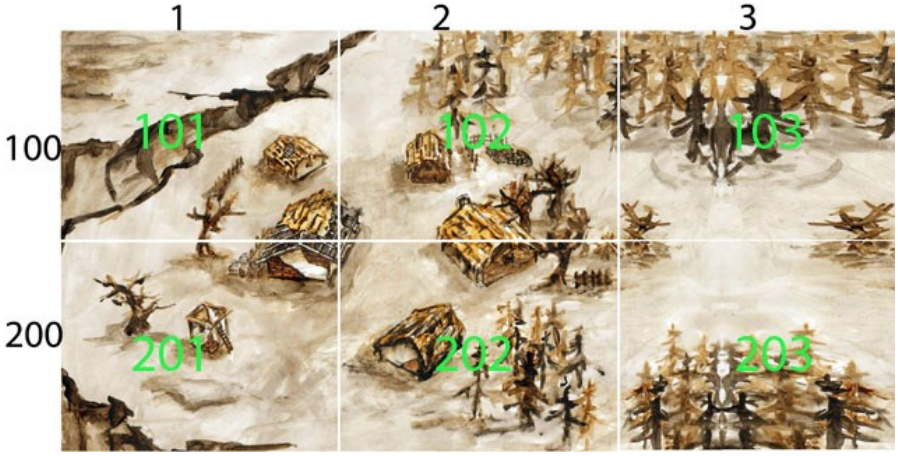


Obrázok 12: Obchádzanie objektov

Na obrázku je znázornené akým spôsobom funguje algoritmus hľadania najkratšej cesty. Najskôr sa vypočíta vrchná cesta (znázornená modrou farbou) pomocou algoritmu popísaného vyššie a potom spodná cesta (červená farna). Porovnaním týchto hodôt sa objekt rozhodne, ktorou stranou ísť. Program vyberá ako najvhodnejšiu cestu s najnižším ohodnotením.

4.5.4 Prechod oblastí

Pod pojmom prechod oblastí sa rozumie celkové prekreslenie hracej plochy, načítanie údajov o objektoch a zmenu pozície hráčskej postavy. Prekreslenie hracej plochy znamená načítanie správneho pozadia a vzhľadu postáv podľa dopredu definovaných pravidiel. Nová hracia plocha so sebou prináša aj mnoho nových objektov, ktoré majú rôzne atribúty. Na priradenie prechodu oblastí sú vytvorené špeciálne algoritmy, ktoré ho riadia. Ako prvé bolo nutné navrhnúť systém, ktorým by sa jednotlivé oblasti dali indexovať tak, aby medzi sebou vytvárali nejaké pravidlo.Výsledný systém reprezentuje následujúci obrázok



Obrázok 13: Indexovanie lokácií

Pri prechode lokácií sa používa algoritmus, ktorý kontroluje predchádzajúcu a momentálnu lokáciu. Každý smer má inú hodnotu. Prechod doprava má hodnotu 1, prechod doľava -1, prechod dole 100 a nakoniec prechod hore -100. Na základe súčtu momentálnej lokácie a tohto prechodu jednoznačne identifikujeme lokáciu, ktorá sa má vykresliť. Objekty prislúchajúce do tejto lokácie sú definované už priamo v jej vlastnej funkcii.

4.6 Bojový systém

Ďaľšou dôležitou položkou v hre sú súboje. Každý takýto súboj sa riadi určitými algoritmami, ktoré počítajú výsledky súboja. Do súboja sa započítavajú rôzne atribúty zúčastnených postáv. Súboj začne tým, ze nehráčska postava zaútočí na hráča. Spustí sa algoritmus startFight(), ktorý je súčasťou triedy Character. Ten v sebe obsahuje časovač, ktorý vždy po upynutí časovej konštanty odošle signál svojej nadradenej triede so všetkými potrebnými atribútami k výpočtu.

4.6.1 Súboj

Súboj začína len v prípade, že má postava nastavené nepriateľské správanie. Tohto správania dosiahne podľa algoritmu popísaného v kapitole 4.3. V momente ako sa dostane postava na dotyk k hráčovi, začne sa fáza súboja. V tejto fáze sa započítavajú obranné a útočne hodnoty zúčastnených strán. Pri každom útoku na protivníka si hádže postava pomyselnou 10 stennou kockou, implementovanou pomocou generátora náhodných čísel. Výsledok tohto hodu sa pripočíta k celkovej útočnej sile. Ďalšia položka, ktorá sa pripočíta k útoku je zbraň, ktorú má postava v rukách. Jednotlivé útočné bonusy sú uvedené v tabuľkách zbraní.

Keď máme útočnú silu, potrebujeme spočítať obranu cieľa útoku. Podobne ako pri útočnej sile, aj tu sa hádže kockou, tentokrát 12 stennou. Výsledná hodnota je rovná súčtu hodu, základnej obrany a obrannej schopnosti zbroje, ktorú má postava momentálne na sebe. Takto získané čísla položíme do jednoduchej rovnice.

ZRANENIE = ÚTOČNÉ ČÍSLO – OBRANNÉ ČÍSLO

Rovnica: Výpočet zranenia

Týmto výpočtom získame hodnotu, ktorá bude odrátana obráncovi z jeho momentálneho množstva životov. Informácia o každom zásahu sa vyskytuje v okne udalostí, ktoré je umiestnené v ľavom spodnom rohu hracej plochy. Po každom výpočte sa odošle signál do rodičovskej triedy a tá

túto informáciu pomocou triedy eventWindow zobrazí. Percentuálny počet aktuálnych životov u hráča sa zobrazuje pod tlačítkami hlavnej hernej ponuky pomocou triedy QProgressBar. Počet životov nepriateľa sa zobrazuje vo vrchnej časti hracej plochy ako červený obdĺžnik.

4.6.2 Smrť postavy

Po tom, ako klesne počet životov postavy na 0, spustia sa algoritmy, ktoré túto informáciu spracujú. Najskôr sa zmení hráčova reputácia v závislosti od toho, akú postavu zabil. Ak zabil postavu s reputáciou nad 50 bodov, tak klesne a naopak ak pod 50 bodov, tak vzrastie. Ďalšia zmena, ktorá nastane je zisk majetku mrtvej postavy. Po smrti sa jej majetok automaticky zmení na zlatky, ktoré sa hráčovi pripíšu do inventára.

Aby nedochádzalo k neustálemu oživovaniu postáv pri prechode do lokácií, sa musí uložiť informácia o ich stave a pozícii. Pri načítaní novej lokácie prebehú algoritmy, ktoré načítajú stav a pozíciu postáv ešte predtým, ako ich paintEvent prekreslí.

Ak umrie hráč, presunie sa do špeciálnej lokácie, kde sa bude môcť oživiť. Samotné oživenie je algoritmus, ktorý zvýši hráčove momentálne zdravie nad 0 a premiestni ho do mesta. Ak sa vráti naspäť do lokácie, kde umrel, postavy tam budú mať opäť plný počet životov.

4.6.3 Zlepšovanie postavy

Pod pojmom zlepšovanie postavy sa, z hľadiska implementačného, rozumie skupina funkcií, ktoré majú za úlohu vyhodnotiť, kedy sa zvýši nejaká bojová schopnosť. Algoritmus je založený na tom, že sa náhodne vygeneruje číslo v rozpätí od 0 po 100. Ak je toto číslo väčšie ako zadaná hodnota (v tomto prípade 98), navýši sa daná atribúta. Kedy sa ktoré atribúty navyšujú je uvedené v samotnom návrhu, v kapitole 2.3.2. Po dosiahnutí určitej hranice sa táto šanca zníži na 0% a teda schopnosť nie je možné ďaľej navyšovať.

4.7 Implementačné problémy

Popri vývoji aplikácie bolo nutné prejsť niekoľko testov funkčnosti. Jednalo sa predovšetkým o rôzne testy pohybu postáv ovládaných počítačom, zmeny ich správania a reakcie na hráčové odpovede v rozhovore. Popri týchto testoch sa objavovali rôzne problémy, ktoré požadovali riešenie. V tejto kapitole si pripomenieme niektoré z vážnejších problémov a ich riešenie.

1. **Pohyb postavy –** prvým problémom, ktorý sa vyskytol pri testovaní bol problém s obchádzaním objetov na mape. Pôvodný algoritmus pracoval s dotykom objektu, v ktorom je vykreslená hráčska postava s rovnakým objektom u statického objektu. Počas testovania sa táto metóda stala nevyhovujúcou, pretože nebolo možné rozoznať smer dotyku. To viedlo k vytvoreniu „nárazníkov“ po okrajoch obdĺžnika, ktorý sa nachádza okolo každého objektu. Táto metóda sa zdala byť dostatočne dobrá, dokedy sa nevyskytli komplikácie s dotykom dvoch nárazníkov s jedným objektom. To viedlo postupným testovaním až ku konečnej verzii sondy, ktorá je uvedená v kapitole 4.5.3. Aj táto sonda však mala svoje problémy, ale tie sa vyriešili jednoduchými algoritmamy, ktoré objekt automaticky presunú o dva kroky v smere pohybu postavy. Týmto krokom preskočíme kritickú oblasť a postava pokračuje vo svojom pohybe ďalej.

2. **Schopnosť vyhľadať cieľ počítačom riadenej postavy** – Hneď po implementácií pohybových algoritmov pre počítačom riadené postavy nastal jeden zásadný problém. Ak si postava zvolila bod, ktorý bol v jednom z objektov, zasekla sa s tým, že nemohla svojho cieľu dosiahnúť. Tento problém bol vyriešený vytvorením algoritmu, ktorý zosníma všekty statické objekty a ak si postava vyberie cieľ, krorý sa vyskytuje v jednom z týchto objektov, nespustí sa pohybový algoritmus, ale zvolí iný bod, dokedy nebude platný pre pohyb.

3. **Súboje** – ani súboje sa nevyskytli bez problémov. Prvým z problémov bolo, že postava, ktorá dostávala záporne zranenie sa tým vlastne liečila. Tento problém bol pomerne jednoducho odstránený. V neskorších fázach, keď už mala hra viacero lokácií došlo k problému, kedy sa po návrate naspäť do lokácie mŕtve postavy samy oživili, aj keď to nebolo žiadané. Musel sa teda vytvoriť algoritmus, ktorý je v skratke popísaný v kapitole 4.6.2.

4. **Rozhovory** – rozhovory boli pomerne problematické hneď po vytvorení prvej verzie. Nastával problém, kedy jedna postava hovorila text inej postavy, pretože všetky rozhovory

boli pevne stanovené v zdrojovom kóde. Tento problém bol vyriešený vytvorením samostatnej triedy Chat, ktorá ovládala rozhovory pre každú postavu zvlášť. Zmenilo sa načítanie textu zo zdrojového kódu na načítanie textu zo súboru v adresárovej štruktúre na pevnom disku. Všetko fungovalo do momentu, keď sa od počitača požadovalo, aby rozoznával text odpovede, ktorú si hráč zvolil. Bolo teda nutné navrhnúť funkčný komunikačný protokol, s ktorým bude riadiaca pracovať. Výsledný protokol sa postupne upravoval až do podoby, ktorý je uvedený v kapitole 4.4.

5. **Prechod lokáciami** – Pri prechode lokáciami nastával problém predovšetkým s objektami, ktoré sa vykreslovali v predchádzajúcej oblasti. Objekty sa síce nevykreslili, ale ostali uložené v pamäti a postava teda obchádzala neviditeľné prekážky, alebo na hráča útočili neviditeľní nepriatelia.Tento problem sa musel teda vyriešiť vytvorením funkcie, ktorá všetky objekty resetovala do pôvodného stavu. Tu však nasáva problém, ktorý je popísaný v odstavci treťom odstavci tejto kapitoly.

Po objavení všetkých väčších aj menších problémoch sa zistené chyby podarilo odstrániť. Program testovali ďalší dvaja ľudia a nič závažnejšie vo funkčnosti neobjavili, takže by sa hra mala vyskytovať vďaka väčšiemu počtu testov bez závažných problémov, ktoré by mali za následok nemožnosť dohrania hry.

5 Záver

Táto práca mala pre mňa veľký prínos, pretože okrem toho, že som sa naučil pracovať s prepracovanou knižnicou QT, zistil som, čo ma na informatike baví najviac. Je to vytváranie umelej inteligencie, dodávanie strojom „život“. Tento projekt plánujem ešte naďalej rozširovať a vytvoriť z neho plnohodnotnú freeware hru nielen na scéne systému Linux.

Ďaľšími plánovanými rozšíreniami sú vytvorenie komplentých grafických animácií postáv a niektorých objekov. V neposlednom rade je v pláne rozšírenie herného sveta, množstvo postáv, koré bude hráč stretávať, pridanie vchodov do budoviek a vytvorenie ich interiérov. Dodať svetu budovy typu hospoda, nevestinec, či kostol, ktoré by ho okorenili podľa chutí každého hráča. Taktiež je už naplánovaná prepracovaná tvorba a vývoj postavy [10].

Na záver by som chcel ešte spomenúť ľudí, ktorí mi pomohli pri tvorbe programu svojími kresbami a hernými nápadmi. Sú to ľudia Pavla Osičková, Michal Butko a Peter Strečanský.

Literatúra

[1] McCarthy, John; Minsky, Marvin; Rochester, Nathan; Shannon, Claude, 1955, A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence
http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html (05.2009)
[2] Geospatial Analysis - a comprehensive guide. 2nd edition © 2006-2008 de Smith, Goodchild, Longley
http://www.spatialanalysisonline.com/output/html/Dantzigalgorithm.html (05.2009)
[3] doc. Ing. František Zbořil, Ph.D., Základy umělé inteligence, Studijní opora, 2006
[4] Jaroslav Dorňák, seminář z Linuxu, 1998
http://atrey.karlin.mff.cuni.cz/seminar/OLD/Papers97/Qt/ (05.2009)
[5] Wikipedia.org, Role-playing game
http://en.wikipedia.org/wiki/Role-playing_game (05.2009)
[6] Wikipedia.org, Player Character
http://en.wikipedia.org/wiki/Player_character (05.2009)
[7] Wikipedia.org, Non-player Character
http://en.wikipedia.org/wiki/Non_player_character (05.2009)
[8] Trolltech, Meta Object System, 2005
http://doc.trolltech.com/3.3/metaobjects.html (05.2009)
[9] Trolltech, Qt Tutorial, 2008
http://doc.trolltech.com/4.3/tutorial-t8.html (05.2009)
[10] Jaroslav Ševčík, Peter Strečanský, Vyvrženci: Krv anjelov, Kniha pravidiel, 2008
http://www.stud.fit.vutbr.cz/~xsevci37/ISP/ (05.2009)

Zoznam príloh

Príloha 1. Príručka pre spustenie a kompiláciu programu

Príloha 2. DVD obsahujúce zdrojové kódy, technickú prácu v elektronickej podobe, manuál k hre a preložené binárne súbory pre systém Linux 32bit a 64bit.

Príloha 1

Spustenie programu - Program je možné spustiť priamo pomocou predkompilovaného súboru. Kompilácia je vykonaná na 32 a 64 bitových architektúrach systému Linux. Program spustíme buď pomocou príkazovej riadky zadaním príkazu ./ibp, alebo pomocou grafického rozhrania, poklikaním na spúštací súbor „ibp“.

Kompilácia – program je možné skompilovať na systéme linux. Musí mať však prístup k štandardným knižniciam jazyka C++ a knižniciam QT4. Nižšia verzia knižnice QT nedokáže tento projekt skompilovať správne. Po tom, ako si zabezpečíme prístup k týmto knižniciam, následuje jednoduchá kompilácia, ktorú QT poskytuje. Celá kompilácia je riadená programom qmake, ktorý sa inštaluje spolu s knižnicou QT. Stačí teda napísať následujúce príkazy do príkazovej riadky v tomto poradí:

```
qmake -project
qmake
make
```

Ak sa v systéme nachádza viacero verzií programu qmake, musíme vybrať tú, ktorá patrí verzii QT4.

Testované PC zostavy:

1. Program bol testovaný na následujúcich počítačoch:
   AMD Athlon X2 64bit, 5600+
   6GB RAM
   GeForce 9800GT, 512 MB, PCIe
   Systém: Ubuntu 8.04 64bit

2. Intel Pentium 4, 1800MHz
   1GB RAM
   GeForce 8500, 256MB, AGP
   Systém: Ubuntu 8.10 32bit

Na oboch zostavách bežal program rovnako rýchlo a neboli zistené žiadne veľké vytaženia hardwarových komponent. Program bol otestovaný aj cez vzdialené pripojenie k serveru merlin.fit.vutbr.cz a taktiež fungoval s rovnakým výsledkom.