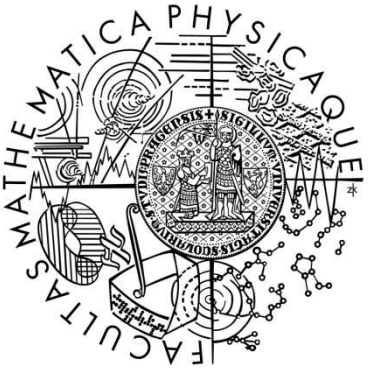
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Matyáš Brenner

Sector 66 – modulární desková hra

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek

Studijní program: Informatika

Studijní obor: Programování

Praha 2012

Děkuji vedoucímu práce Mgr. Pavlu Ježkovi za vedení, cenné rady, nápady, připomínky a čas, který mi poskytl v průběhu jejího vypracování.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V …..... dne …..... podpis

Název práce: Sector 66 – modulární desková hra

Autor: Matyáš Brenner

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek

Abstrakt:
Cílem této práce je navrhnout a implementovat deskovou hru Sector 66, která je svými pravidly založena na hře Quoridor od firmy Gigamic. Sector 66 je hra pro dva až čtyři hráče s 3D reprezentací scény. Je jí možné hrát po síti nebo proti počítači. Dále nabízí rozšiřitelnost zásuvnými moduly v podobě polí herního plánu a kouzel. Umožňuje vytvářet hru na základě předem připravených šablon, které do jisté míry modifikují pravidla.

Klíčová slova: XNA, WCF, plug-in, hra

Title: Sector 66 – A Modular Board Game

Author: Matyáš Brenner

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek

Abstract:
The goal of this thesis is to design and implement a board game Sector 66, which is based on the rules of Quoridor game designed by Gigamic company. Sector 66 is a game for two to four players with a 3D representation of the scene. It is possible to play it over a network or against the computer. It also offers expandability by plug-ins with fields of game plan and spells. It is possible to start the game based on prepared template, which can modify game rules in a certain way.

Keywords: XNA, WCF, plug-in, game

Obsah

1 Úvod

Tato kapitola obsahuje popis hry Quoridor a z ní vycházející Sector 66, který je hlavní náplní celé práce. Kapitola také popisuje zadání práce a nastiňuje problémy, které bude potřeba vyřešit.

1.1 Quoridor a Sector 66

Sector 66 je desková strategická hra navržená jako rozšíření hry Quoridor od firmy Gigamic, který je popsán níže.

Quoridor

Quoridor, ze kterého tato práce vychází, je desková hra pro dva nebo čtyři hráče hraná na čtvercovém herním plánu o rozměrech 9 x 9 políček, mezi kterými jsou širší spáry umožňující stavění zábran. Hráči začínají na středovém políčku na stranách herního plánu proti sobě. Cílem hry je přejít herní plán rychleji než soupeř. Hráči mají v rámci svého tahu na výběr ze dvou akcí. Buď posunou figurku v povoleném směru, který je daný vzájemnou pozicí hráčů a rozmístěním zábran, nebo postaví zábranu, která ztěžuje pohyb figurek po herním poli.

Obrázek 1 zobrazuje počáteční rozmístění hráčů a zábrany, které ještě nejsou umístěny do herního plánu.

Obrázek 1. Počáteční rozmístění hráčů ve hře Quoridor

Kompletní pravidla a popis hry Quoridor je možné najít na internetových stránkách firmy Gigamic (1).

Sector 66

Sector 66 vychází z Quoridoru a je jeho počítačovou implementací s upravenými pravidly. Sector 66 oproti Quoridoru přinese možnost hraní po síti, použití políček se speciálními efekty a sesílání kouzel. Kouzla budou herní objekty, které se budou náhodně objevovat v souladu s nastavením hry po herním plánu. Poté, co hráč postaví svoji figurku na pole, kde se kouzlo objeví, dojde k jeho přesunu do inventáře hráče. Ten bude moci kouzlo použít ve svém libovolném dalším tahu místo posunu figurkou, nebo postavení zábrany. Kouzlo může ovlivnit buď figurku hráče, figurky ostatních hráčů, pole herního plánu, nebo zábrany. Daný typ kouzla má vždy stejný efekt. Každé kouzlo je možné použít jednou a poté je odstraněno z inventáře hráče, který má kapacitu omezenou pravidly na čtyři položky.

Sector 66 také umožňuje začít hrát hru z jiného, téměř libovolného, počátečního rozestavění.

Kompletní pravidla hry Sector 66 je možné najít v příloze 1.

Hra by měla pravidla aktivně vynucovat a zvýrazňovat možné tahy hráče, který je na tahu.

1.2 Zadání práce

Grafika a uživatelské rozhraní

V souladu se zadáním by implementace hry Sector 66 měla využívat 3D grafiku k zobrazení vlastního herního plánu, figurek a animaci jejich pohybu. Figurky mohou mít složitější tvar než ostatní herní objekty. Aplikace by zároveň neměla používat žádné pokročilé grafické efekty, aby ji bylo možné spustit i na starších, méně výkonných počítačích. Aby hra běžela plynule a zajistila se nižší zátěž procesoru, bude nutné využít akceleraci grafické karty.

Pro menu a ovládání hry budou použity běžné 2D ovládací prvky jako tlačítka a podobně.

Lokalizace

Aplikace by měla být lokalizovatelná a měla by umožňovat tvorbu překladů pomocí jednoduché utility.

Zásuvné moduly

Sector 66 by měl podporovat zásuvné moduly v podobě kouzel a políček.

Pro jednoduchou tvorbu zásuvných modulů by měly být v projektu připraveny třídy a rozhraní, která stačí rozšířit do požadované podoby.

Umělá inteligence

Součástí řešení by měla být umělá inteligence, která by umožňovala hrát hru proti počítači.

Síťová komunikace a server hry

Sector 66 by mělo být možné hrát po síti. K tomu by měl být vytvořen dedikovaný server, který by umožňoval komunikaci mezi hráči, Tento server by měl být schopný hostovat více her současně. Pro případ výpadků spojení by v sobě měl nést informace o běžících hrách.

Šablony

Vzhledem k tomu, že Sector 66 přináší oproti Quoridoru mnoho nových možností, bude nutné určit, jak se bude daná hra hrát. K tomu by měla sloužit šablona hry, která by měla určovat, co je ve hře povoleno.

Měla by umožnit nastavit, jestli bude možné používat kouzla, jaká kouzla to budou, jak často se budou generovat nová a s jakou pravděpodobností. Měl by jít omezit maximální a minimální počet hráčů ve hře, také měnit z jakých políček bude složen herní plán a kolik zábran budou moci hráči postavit. Také by tato šablona měla umožnit postavit zábrany do hry již při jejím spuštění. V neposlední řadě pak na jakém poli hráči budou začínat a kde budou jejich cílová pole. Na základě použitých zásuvných modulů budou zajištěny ty, bez kterých se nepodaří šablonu načíst.

Šablony by mělo být možné vytvářet, upravovat a zakládat podle nich hry.

Pro jednoduchou manipulaci s nimi by měl být součástí řešení editor šablon.

1.3 Cíle práce

V následujících bodech je shrnuto zadání této práce.

1. Vybrat vhodný grafický framework pro účely tohoto projektu
2. Vytvořit aplikaci reprezentující hru s aktivním vynucováním pravidel a zvýrazňováním možných tahů
3. Vytvořit mechanismus lokalizace a utility umožňující tvorbu překladů
4. Realizovat mechanismus pro načítání zásuvných modulů
5. Vytvořit server hry a navrhnout síťovou komunikaci
6. Realizovat umělou inteligenci
7. Vytvořit editor šablon hry
8. Naprogramovat ukázkové zásuvné moduly do hry

2 Analýza – Návrh

Následující text rozšiřuje kapitolu 1 o analýzu závislou na vybraných technologiích, zdůvodňuje jejich výběr a poskytuje alternativy. V této kapitole jsou popisovány pouze ty problémy, se kterými bylo počítáno v době návrhu projektu. V kapitole 3 jsou popisovány problémy, které vznikly až v průběhu implementace.

2.1 Platforma

Projekt bude vyvinut v souladu se zadáním na platformě .NET 4.0. Mohl by být vyvinut i na starším .NETu 3.5, ale jak se ukáže po volbě grafického frameworku, verze 4.0 je nutná, protože XNA 4.0 starší .NET nepodporuje.

Jako jazyk, ve kterém je projekt naprogramován, byl zvolen C# z důvodu jeho rozšířenosti a pohodlného použití. V případě potřeby cílené optimalizace bude část projektu možné napsat v jazyce C++/CLI, což nepoškodí konzistenci kódu ve smyslu zachování jedné platformy. Vzhledem k tomu, že C++/CLI a C# jsou oba jazyky .NETu, nevznikl by žádný problém s kompatibilitou ani připojováním optimalizovaného kódu do projektu.

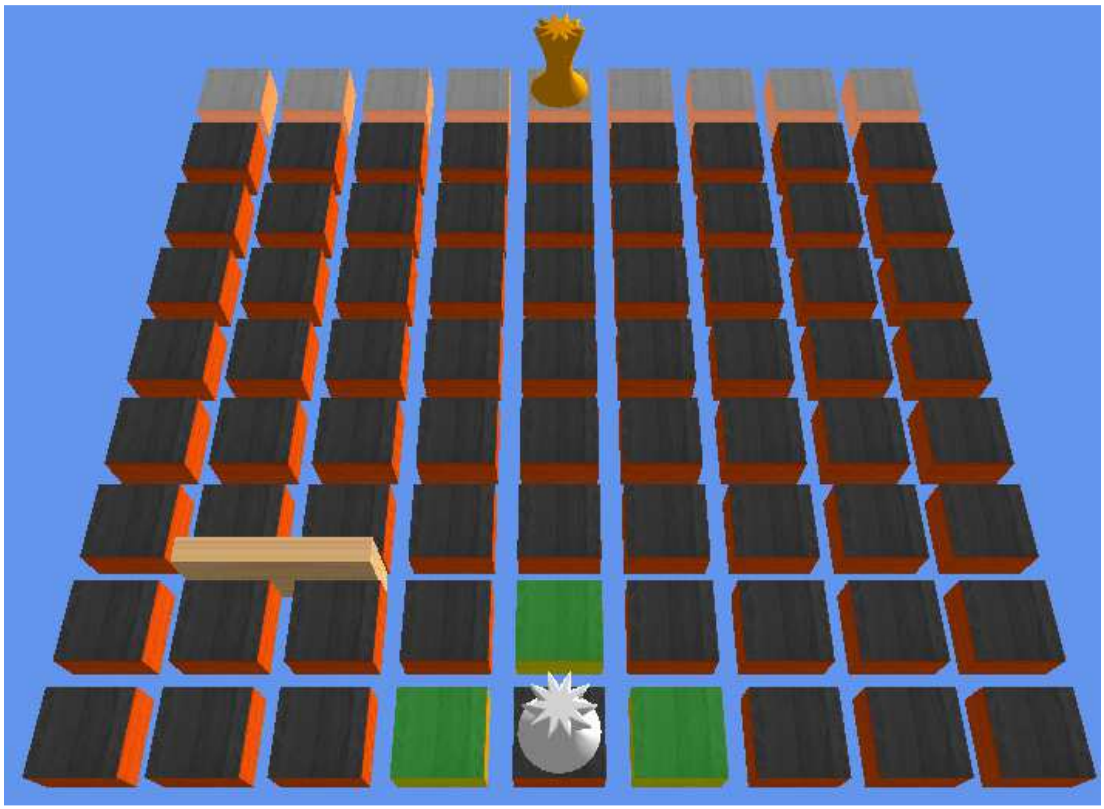
Výkon jazyka C# by ale měl být pro projekt dostačující a cílené optimalizace proto pravděpodobně nebude nutné použít.

2.2 Grafický framework

Tato kapitola se věnuje výběru vhodného grafického frameworku pro potřeby projektu a je řešením bodu 1 z kapitoly 1.3.

Jak bylo uvedeno v úvodu, v aplikaci bude nutné použít hardwarově akcelerovanou grafiku, aby byl zajištěn její plynulý běh. Proto bude nutné vybrat vhodný grafický framework, který by práci s grafickou kartou umožňoval. Vzhledem k figurkám hráčů, které mohou mít složitější tvar, by mělo být možné do hry načíst model figurky vytvořený v některém z běžných softwarů pro 3D modelování, jako například Blender nebo Maxon Cinema 4D.

Obrázek 2 ukazuje představu o zobrazení hry bez ovládacích prvků.



Obrázek 2. Představa o grafice hry Sector 66

Vzhledem k povaze aplikace by tento framework měl být k dispozici zdarma. Komerční produkty, jako například Source Engine nebo Unreal Engine nebudou do srovnání zahrnuty.

Grafické frameworky je možné rozdělit do několika úrovní abstrakce. Nejníže na této pomyslné stupnici by byly ty, které umožňují přímý, nebo téměř přímý přístup k hardwaru a mají buď malé množství podpůrných knihoven pro práci s maticemi texturami a podobně, nebo dokonce žádné. Mezi ně patří DirectX a OpenGL.

O něco výše leží frameworky podobné XNA, které oproti nižší úrovni nabízí rozsáhlé knihovny pro práci s vektorovým počtem, maticemi, texturami a podobně. Ještě ale nenabízí správu grafických objektů, jako plnohodnotné grafické enginy, které tvoří třetí úroveň na pomyslné stupnici.

Vzhledem k relativní jednoduchosti hry z grafického hlediska, správa grafických objektů nebude potřeba. Použití frameworků třetí úrovně abstrakce by do kódu aplikace zavleklo složité mechanismy, které by neměly rozumné využití ani návratnost v podobně jednodušší správy hry.

Frameworky nejnižší úrovně by pravděpodobně vyžadovaly doprogramování knihoven pro práci s vektorovým počtem a proto se také nejeví jako vhodná volba.

Poslední čtvrtou vrstvu, již mimo navrženou stupnici, tvoří plnohodnotné herní enginy, které vedle grafického framework také obsahují správu herních objektů, podporu pro fyziku hry nebo podporu pro umělou inteligenci.

2.2.1 DirectX

DirectX je knihovna od firmy Microsoft poskytující aplikační rozhraní, které umožňuje ovládat grafické karty. Je relativně nízkoúrovňový a neposkytuje programátorovi takovou podporu, jako některý ze systémů pro tvoru her. Velká část těchto systémů ale DirectX vnitřně využívá.

Pro použití DirectXu by bylo nutné využít wrapper knihovnu, jako například SlimDX nebo SharpDX, která by zpřístupnila funkce frameworku v řízením prostředí .NETu.

Pro DirectX se dají najít různé tutoriály, které usnadňují programátorovi pochopení jeho používání. Většina nalezených ale nebyla aktuální a byla napsaná pro .NET 3.5.

Testovací projekt se nepodařilo zkompilovat, pokud v cestě ke zdrojovým kódům byly speciální znaky jako mezery nebo znaky s diakritikou.

Z uvedených důvodů DirectX nebyl vybrán jako vhodné řešení pro potřeby tohoto projektu.

2.2.2 OpenGL

Podobně jako DirectX je OpenGL nízkoúrovňový framework pro práci s grafickou kartou a pro jeho použití by také bylo nutné využít wrapper, který by zpřístupnil jeho funkce z prostředí .NETu. Pravděpodobně nejrozšířenější je open sourcový OpenTK, který obaluje funkce OpenGL, OpenCL a OpenAL. Navíc poskytuje základní podporu pro vektorový počet a podobně.

OpenGL je relativně rozšířené pro práci s grafikou a tím pádem není problém najít tutoriály, různé rady a fóra.

OpenGL nemá k dispozici vstupy, a proto je s nimi nutné pracovat stejně, jako u okenních aplikací. To mimo jiné znesnadňuje případný přenos na jinou platformu, jako například XBox.

OpenGL nebylo vybráno jako vhodné řešení kvůli své nízkoúrovňovosti a snaze o zachování jedné platformy. Vzhledem k .NETu budou preferována řešení firmy Microsoft.

2.2.3 XNA

XNA je platforma pro programování her v jazycích C# a Visual Basic od firmy Microsoft podporující nejen Microsoft Windows, ale také XBox 360, Zune a Windows Phone. Další platformy by mohlo nabídnout prostředí MonoGame (2), což je open sourcová implementace frameworku XNA.

XNA nabízí o něco vyšší míru abstrakce než DirectX a OpenGL. Také přináší vyšší podporu v podobě knihoven pro vektorový počet, práci s texturami a podobně. Podle nastavení XNA vnitřně využívá DirectX 9 nebo modernější DirectX 10.

XNA využívá pro operace s grafikou, modely i jinými daty předkompilovaný formát XNB, který umožňuje rychlejší práci za cenu nutné kompilace vstupních souborů. XNA ale umožňuje načtení i nepředkompilovaných data.

Hlavním rozdílem mezi DirectX a XNA je, že XNA používá pro vše, co se má vykreslit, efekt. Efekt je typicky popsán krátkým zdrojovým kódem v jazyce HSLS definující pixel a vertex shader. Shadery říkají grafické kartě, co má dělat se sadou bodů, trojúhelníků a textur, které dostane k vykreslení. Umožňuje dělat transformace nejen na úrovní kódu aplikace, ale právě i v efektu, což přenáší zátěž z procesoru na grafickou kartu, která je na vektorové výpočty lépe optimalizovaná než procesory. Tento mechanismus ostatní frameworky vyšší míry abstrakce také využívají.

Nevýhodu XNA je, že je to uzavřený framework, který neumožňuje náhled do zdrojových kódů. Proto je těžké určit jeho efektivitu.

Další nevýhodou XNA je, že neobsahuje žádné připravené struktury pro vytváření grafického rozhraní. Pokud programátor chce používat tlačítka a podobné standardní ovládací prvky, nezbývá mu, než si je sám naprogramovat, nebo použít jednu z existujících knihoven třetích stran.

Pro XNA existuje velké množství tutoriálů. Na internetových stránkách Microsoftu (3) se dají najít menší aplikace a další tutoriály usnadňující jeho pochopení.

Tento framework, v aktuální verzi 4.0, byl vybrán jako vhodné řešení pro tento projekt, protože je dobře dostupný a zdokumentovaný, stále se vyvíjí a představuje vhodnou míru abstrakce, která rozumně odděluje aplikaci od hardwaru a nabízí dostatečné množství funkcí, aby nebylo nutné doprogramovat rozsáhlé knihovny pro práci s grafikou. Starší verze není vhodné používat z důvodů podpory hardwaru i podpory ze strany Microsoftu.

V rámci programování testovací aplikace bylo vidět, že XNA je framework, který se snadno používá.

2.2.4 Ogre

Ogre je jeden z rozšířenějších open sourcový grafických frameworků. Je na vyšší úrovni abstrakce než XNA a nabízí správu grafických objektů, která v tomto projektu není potřeba vzhledem k relativní jednoduchosti hry.

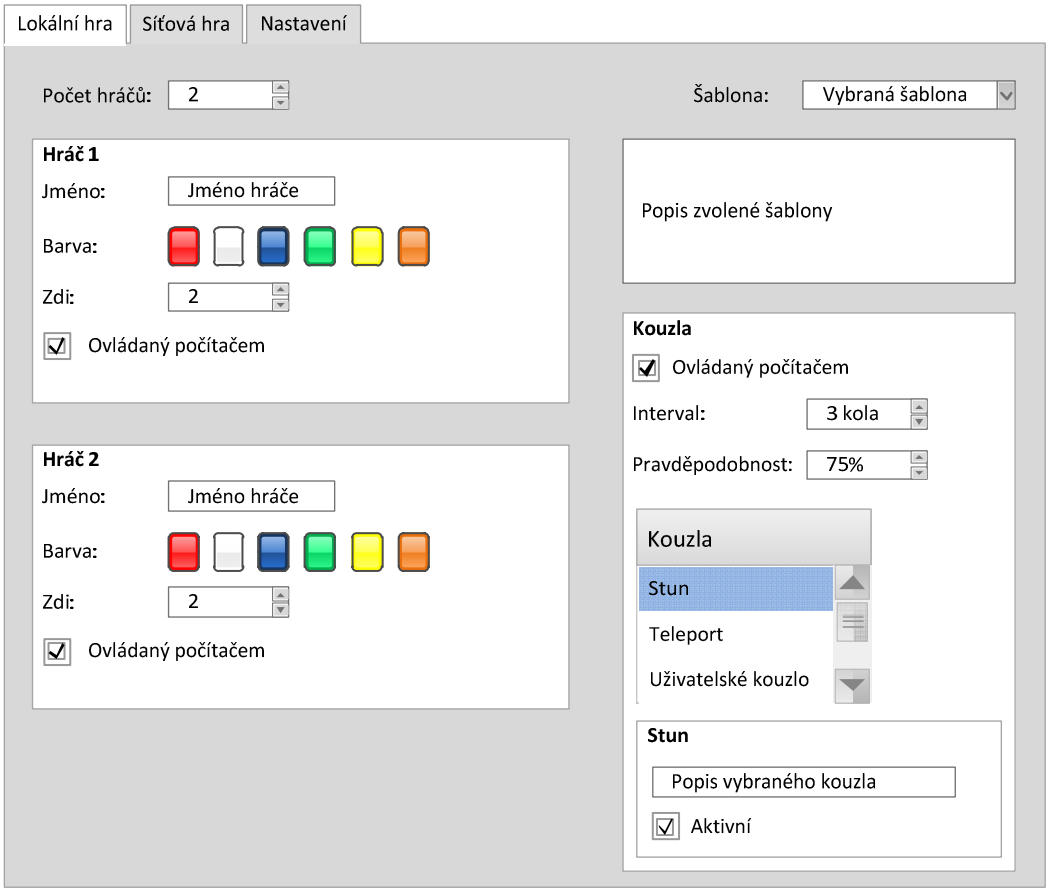
K tomuto frameworku jsou dostupné tutoriály a návody především na oficiálních webových stránkách (4). Jinde na internetu je návodů relativně málo.

Ogre3D je určen pro C++ .NET, ale existuje wrapper knihovna s názvem Mogre, která zpřístupňuje funkce původního frameworku z jazyka C# a ostatních řízených jazyků .NETu.

Ogre nebyl vybrán z důvodů zbytečně vysoké míry abstrakce, která by do projektu zavlekla složitější konstrukce bez nutnosti použití funkcí, které by tento framework přinesl navíc oproti ostatním alternativám. Stejně jako OpenGL nesplňuje snahu o zachování jedné platformy.

2.3 Grafické rozhraní

Vzhledem k tomu, že jako grafický framework bylo vybráno XNA, je ještě nutné vybrat vhodnou knihovnu obsahující komponenty grafického rozhraní, jako tlačítko a podobně. Taková knihovna v XNA není, což je v tomto případě jeho hlavní nevýhodou. Vybrané řešení by opět mělo být vzhledem k povaze projektu k dispozici zdarma a mělo by nabízet dostateční množství vhodných komponent pro vytvoření grafického rozhraní hry. Obrázek 3 nastiňuje jeho požadovaný vzhled.



Obrázek 3. Grafické rozhraní aplikace

Vzhledem k tomu, že hry mohou mít buď velmi složité, nebo také žádné grafické rozhraní mimo samotné hry, je pochopitelné, že nástroj pro tvorbu grafického rozhraní není součástí XNA. Také to ale v mnoha případech komplikuje nasazení. Často stačí použít XNA v rámci WPF (Windows Presentation Foundation) nebo v rámci WinForms, což ale může vést k nekonzistenci vzhledu aplikace. Toto řešení navíc znemožňuje použití ovládacích prvků přímo ve hře.

Za existence XNA vzniklo značné množství komerčních i nekomerčních řešení, která s tvorbou grafického rozhraní měla pomoci. Ty jsou však často špatně udržovány a v mnoha případech nevyšla nová verze po dobu několika let. Hodně těchto knihoven ani nepodporuje nejnovější XNA 4.0.

Z důvodů povahy projektu nebyla komerční prostředí testována. Existují například tato: OrbUI (5), DigitalRune (6), Squid (7).

Při výběru knihovny pro tvorbu grafického rozhraní v rámci XNA byla uvažována tato nekomerční řešení:

Nuclex

Nuclex (8) se aktuálně zdá být nejlépe udržovaným projektem, který podporuje nejen platformu PC ale i XBox 360. Navíc je možné jeho vzhled jednoduše upravovat pomocí skinů. Obsahuje ale malé množství ovládacích prvků a zejména pak ty, které by odpovídaly představě o vzhledu uživatelského rozhraní. Proto nebyl vybrán pro tvorbu grafického rozhraní ve hře Sector 66.

Simple Gui

Simple Gui (9) je další z nekomerčních knihoven pro tvorbu grafického rozhraní v rámci XNA. Jeho vývoj ale skončil pro XNA 3.1 a proto nebylo možné ani vytvořit testovací aplikaci. Stejně jako Nuclex obsahuje jen malé množství komponent.

Neoforce

Neoforce (10) je nekomerční knihovna pro tvorbu grafického rozhraní pro XNA. Přestože podporuje XNA verze 4.0, jeho vývoj je momentálně pozastaven a je pravděpodobné, že nebude pokračovat. Platforma XBox 360 není podporována bez změn ve zdrojových kódech knihovny, které jsou na internetových stránkách projektu k dispozici. U testovaných komponent tyto změny nebyly zásadní a týkaly se většinou způsobu přejímání vstupu od uživatele, například z virtuální klávesnice a podobně.

Neoforce nabízí možnost skinování a proto je možné upravit vzhled komponent tak, aby vypadal konzistentně se samotnou hrou. Obsahuje velké množství komponent od jednoduchých tlačítek až po záložky, které umožňují rozdělit okno aplikace. Komponenty jsou řízeny událostmi, což umožňuje intuitivní používání této knihovny. Knihovna Neoforce byla vybrána jako řešení pro tvorbu grafického rozhraní v aplikaci Sector 66, protože jako jediná z testovaných řešení nabízí komponenty odpovídající návrhu grafického rozhraní.

2.4 Lokalizace

Tato kapitola řeší otázku lokalizace, což je bod 3 kapitoly 1.3.

Jak bylo řečeno v zadání, aplikace by měla být lokalizovatelná a měla by nabízet utilitu pro tvorbu překladů aplikace. Volbu jazyka by mělo jít změnit v nastavení aplikace.

Formát lokalizace

Bylo by vhodné, aby se dal překlad vytvořit i bez utility. Vzhledem k nízkému očekávanému počtu spíše jednodušších frází, které pravděpodobně budou buď (téměř) jednoslovné, nebo naopak budou tvořit celé věty, bude důraz při tvorbě lokalizačního mechanismu pro Sector 66 kladen na co nejjednodušší možnost tvorby překladů a jejich jednoduché načítání do aplikace. Proto nebude použit běžný způsob, který využívá *Resources* aplikace, kde jsou uloženy dvojice klíčů spolu s frázemi, které mají být zobrazeny. Tento mechanismus má tu výhodu, že je přímou součástí aplikace, a pro každou jazykovou mutaci je při kompilaci vytvořena knihovna DLL, která obsahuje překlad. Nevýhoda spočívá v tom, že překlad je možné vytvářet jen před kompilací projektu.

V tomto projektu je s ohledem na výše uvedené nevýhody zvolen jiný způsob. Překlad bude uložen ve formě XML souboru, který umožňuje přehledné čtení a editování i bez specializované utility. Navíc je tento formát dobře podporovaný knihovnami .NETu a jednoduše se načítá. V případě potřeby se na něj dají aplikovat transformace a jinak ho upravovat. Tento soubor se za běhu aplikace nahraje pomocí XML serializace a poté se z takto nahraných frází podle klíče vybírá ta, která bude zobrazena na obrazovce.

Výchozí anglický překlad bude přímou součástí aplikace. Pro uchování souboru s výchozím překladem budou použity *Resources*, což zajistí, že výchozí překlad bude vždy k dispozici a to i tehdy, když dojde k náhodné ztrátě datových podkladů aplikace.

XML soubor s překladem bude uložen ve formátu UTF-8, což umožní používání libovolných národních a jiných speciálních znaků. Jeho formát je znázorněn v příloze 2.

Aby bylo možné překlady a jejich verze jednoduše rozlišit, budou v XML souboru uvedeny mimo přeložených frází informace o autorovi, verzi a jazyku.

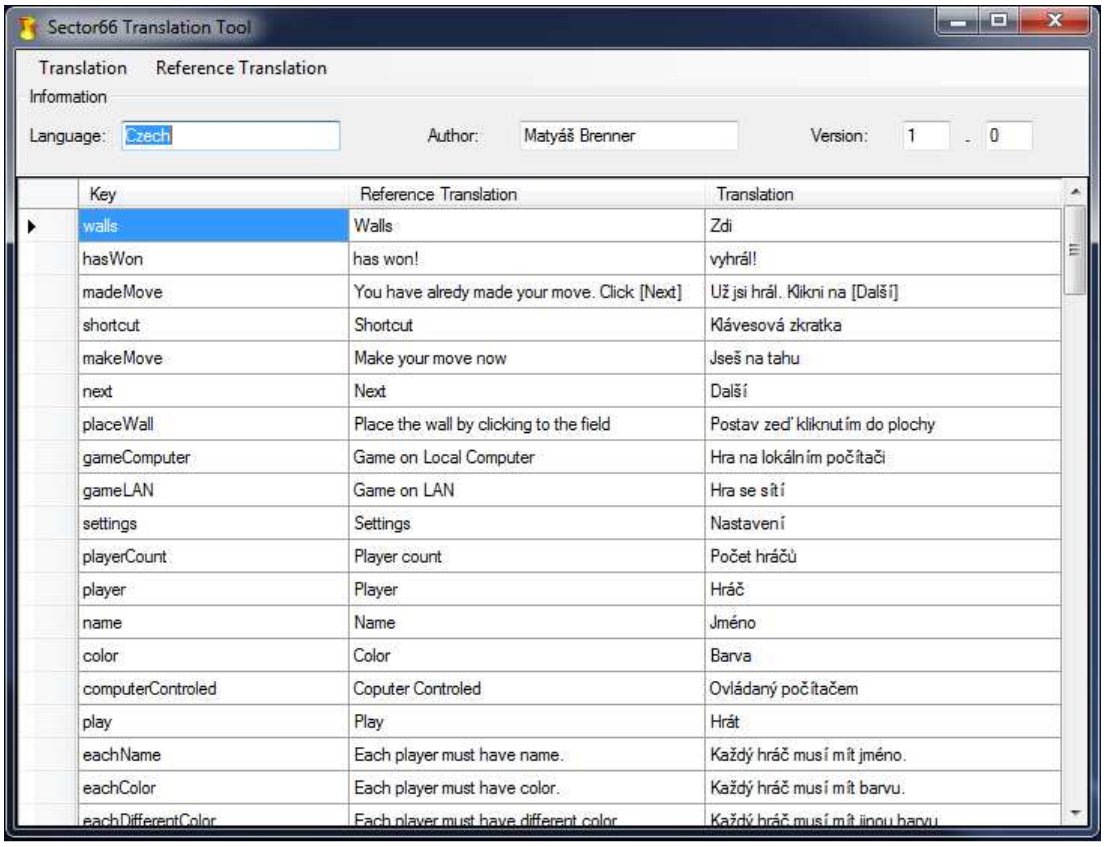
Aby nemohlo dojít k výpadku překladu při načtení jen částečné lokalizace, dojde při načtení překladu do aplikace k přepsání výchozích, anglických frází těmi, které budou obsaženy v XML. Tím se zajistí, že vždy bude k dispozici alespoň anglická lokalizace.

Český překlad bude také součástí řešení. Nebude ale přímou součástí aplikace a bude dodán ve formě XML souboru.

Utilita pro tvorbu překladu

Aby bylo zajištěno co nejpohodlnější překládání, měla by utilita pro tvorbu překladu zobrazovat dvojici referenční překlad – překládaná fráze. Referenční překlad by navíc mělo být možné načíst z jakéhokoliv existujícího překladu hry. Utilita také umožní uložení a načtení rozpracovaného překladu. Utilita bude sama lokalizovaná do angličtiny, což bude výchozí překlad, a do češtiny. V tomto případě bude využit standartní způsob využívající *Resources* aplikace, který je nabízen komponentami .NETu ze jmenného prostoru *System.Windows.Forms*. Lokalizace bude v tomto případě automaticky zvolena při startu aplikace podle jazykového nastavení uživatelského účtu, pod kterým bude program spuštěn.

Obrázek 4 ukazuje představu o vzhledu utility pro tvorbu překladů.



Obrázek 4. Představa o vzhledu utility pro tvorbu překladů

2.5 Zásuvné moduly

Tato kapitola je věnována výběru vhodného mechanismu pro připojení a tvorbu zásuvných modulů do hry Sector 66 a tím řeší bod 4 kapitoly 1.3.

V souladu se zadáním by aplikace měla být co nejjednodušším způsobem rozšiřitelná zásuvnými moduly. Ty by mělo být snadné nejen připojit do hlavní aplikace ale i vytvářet.

Nejjednodušší způsob pro uživatele jak připojit zásuvný modul do aplikace je nakopírovat soubor, který ho reprezentuje, do určeného adresáře. Alternativní a o něco složitější způsob by byla registrace modulů v nastavení aplikace. V tomto případě byl zvolen první uvedený způsob z důvodů své přímočarosti.

Pro jednoduché vytváření zásuvných modulů v rámci projektu vznikne sada rozhraní a tříd, které budou nabízet metody pro přetížení. Programátor pak při tvorbě zásuvného modulu oddědí svoji třídu od daného předka, nastaví malé množství vlastností a přetíží požadované metody. Rozhraní budou sloužit pro komunikaci s ostatními herními objekty.

Zásuvné moduly s sebou nesou bezpečnostní riziko v podobě načtení kódu programátorů třetích stran do aplikace, které by potenciálně mohlo vést ke zneužití přístupu k počítači. Speciálně by se to mohlo projevit v případě zvoleného připojování modulů, kdy si uživatel ani nemusí být vědom, že je nějaký modul připojen.

Pro řešení popsaného bezpečnostního problému se v .NETu nabízí použití aplikačních domén. Zásuvné moduly budou načítány do vedlejší aplikační domény, která bude mít razantně omezená práva pouze na čtení ze složky se zásuvnými moduly a pro jejich načtení, což zabrání zásuvným modulům v manipulaci s daty na pevném disku a v komunikaci s okolím.

Pro komunikaci mezi aplikačními doménami .NET využívá marshalling, což je mechanismus podobný serializaci, který umožňuje přenášet paměť reprezentující objekt mezi částmi aplikace nebo aplikacemi.

Ke zvolenému řešení není žádná jiná přímočará alternativa, která by zajišťovala stejně bezpečné a jednoduché použití.

2.6 Síťová komunikace a server hry

Cílem této kapitoly je nalézt vhodné řešení na bod 5 kapitoly 1.3 a tím nastínit síťovou komunikaci hry s dedikovaným serverem.

Požadavky

V souladu se zadáním by server měl držet informace o běžících hrách, jejichž počet bude omezen proměnnou (aktuálně nastaveno proměnnou na 100 současně hraných her), aby nedocházelo k zahlcení. K mazání her ze serveru bude docházet na základě doby platnosti, která bude u dohraných her kratší než u hraných. Pokud po danou dobu nebude ke hře přistoupeno některým z klientů, bude hra ze serveru odstraněna a tím se uvolní místo pro další.

Komunikace mezi klienty a serverem by měla probíhat tak, aby kladla co nejmenší požadavky na nastavení síťových zařízení s aktivním překladem adres. Speciálně to znamená, že server bude muset být od klientů v síti viditelný, ale klienti ze serveru ne. Stejně tak klienti nebudou muset mít v síti mezi sebou přímou viditelnost. Tento model je podobný tomu, který se používá například pro HTTP servery.

Serverová aplikace by také měla usnadnit co nejjednodušší nasazení, které by v případě potřeby zajistilo možnost hostovat ho v rámci webového serveru, aplikace nebo jako službu operačního systému s co nejmenším úsilím.

Pro zajištění bezpečnosti a zamezení připojení nežádoucích klientů ke hře, bude každá síťová hra opatřena heslem, specifikovaným při jejím založení. Po sítí bude přenášen pouze hash (aktuálně získaný algoritmem SHA1) tohoto hesla, který bude také uložen na serveru.

Řešení

.NET poskytuje řešení pro síťovou komunikaci v podobně WCF (Windows Communication Foundation), které umožňuje spravovat synchronní i asynchronní volání po síti bez nutnosti vývoje vlastního protokolu pro komunikaci. Zároveň vyhovuje výše uvedeným požadavkům.

WCF umožňuje použít pro přenos dat většinu používaných protokolů, jako HTTP, TCP/IP a další. Pro přenos dat se využívá serializace objektů. V případě WCF je tento stream přenesen zvoleným protokolem po síti a u protějšku komunikace je proveden opačný postup pro rekonstrukci původní struktury dat a dat samotných.

WCF umožňuje volání po síti tak, jako kdyby volaná metoda byla na stejném počítači a ve stejné aplikační doméně. Rozdíl je navenek jen v nutnosti serializace.

Výhoda WCF spočívá také v tom, že třída označená atributem *ServiceBehavior* je hlavní třídou serverové služby a WCF zajišťuje, že operace nad ní běží ve vláknech a je tedy možné vyřizovat více požadavků zároveň. Programátor tedy nemusí sám řešit problémy s vlákny, frontou požadavků a podobně.

Alternativy

Alternativou ke zvolenému jednosměrnému provozu ve směru od klienta k serveru by byla obousměrná komunikace, která by využívala callbacky, což WCF také umožňuje. Toto řešení by odstranilo potřebu periodických dotazů na server, které zjišťují aktuální stav. Callback by se vyvolal ve chvíli, kdy se stav na serveru změní. Obousměrný HTTP kanál je při použití WCF možné vytvořit jen mezi klientem a serverem, kteří na sebe mají v síti přímou viditelnost, což znemožňuje používání aktivních síťových prvků bez nastaveného přeposílání paketů a odporuje to výše popsaným požadavkům. Použití spojení přes protokol TCP, který je z principu obousměrný, není možné využít, protože by potom server nebylo možné hostovat výše nastíněným způsobem. Proto by obousměrné řešení bylo vhodnější v lokální síti, kde by šetřilo síťové zdroje, ale není ho možné použít v internetu.

Alternativou k WCF může být sestavování komunikace pomocí síťových streamů. To by ale vedlo k nutnosti zavedení vlastního protokolu pro identifikaci obsahu a složité parsování přijatých zpráv. V případě, že by se u tohoto způsobu komunikace využívala serializace a neposílala by se přímo textová nebo binární data, stejně by nebyl vyřešen problém s vytvořením složitého serveru, který by musel využívat více vláken a frontu požadavků. WCF poskytuje již hotové, odladěné řešení.

2.7 Umělá inteligence

V této kapitole je řešena umělá inteligence pro hru Sector 66, což odpovídá bodu 6 kapitoly 1.3.

Teorie

Analýza umělé inteligence vychází z práce P. J. C. Mertense, který uvádí, že vyřešit umělou inteligenci pro Quoridor je podobně těžký problém, jako vyřešit umělou inteligenci pro šachy (11).

Obecně je možné řešit umělou inteligenci více způsoby. Přímočarým způsobem je prohledávání stavového prostoru hry, kde každý stav je nějak ohodnocený. Umělá

inteligence vybírá takové tahy, které jsou v danou chvíli pro hráče nejvýhodnější. Tato metoda bývá na bázi minimaxu a jeho modifikací, jako minimax s α-β ořezáváním nebo negamax. Tyto metody jsou ale náročné v tom, že procházejí velkou část stavového prostoru hry. V konkrétním případě hry Sector 66 je tento prostor příliš veliký na to, aby se výsledek vešel do operační paměti počítače, natož aby se stihly všechny stavy vygenerovat. Pokud není možné uložit všechny stavy až do konce hry do paměti, není možné ohodnotit je přímočaře podle počtu tahů do výhry, ale musí se ohodnocovat pomocí nějaké funkce, která na základě daného stavu hry bez kontextu určí jeho hodnotu. Tato funkce navíc musí být poměrně rychlá, protože se volá pro velký počet stavů. Takovou funkci v případě Quoridoru a Sectoru 66 není jednoduché najít.

Jiným způsobem mohou být rozhodovací algoritmy fungující na základě neuronových sítí a evolučních algoritmů. Tento typ algoritmů ale přesahuje rámec práce a proto tyto algoritmy nebyly podrobněji zkoumány.

Dalším problémem je nemožnost rychle zjistit hodnotu daného stavu hry. Obecně se počet tahů na cílové pole dá zjistit prohledáváním do šířky nebo do hloubky s časovou složitostí $O(|V|+|E|)$, kde $|V|$ je počet stavů hry a $|E|$ je počet přechodů mezi nimi. Což vzhledem k velikosti stavového prostoru hry není únosné. Toto řešení navíc nezohledňuje strategičnost rozmístění zábran a v případě hry Sector 66 ani rozmístění kouzel po herním plánu. Proto připadá v úvahu pouze odhad hodnoty stavu hry, vypočítaný ze vzdálenosti hráče od cílového pole, který případně může brát v úvahu, zda mezi hráčem a cílovým polem stojí zábrana, nebo ne. Tento odhad ale může být velmi nepřesný a z toho důvodu umělá inteligence nebude mít dostatečně dobré výsledky. V některých případech dokonce nemusí najít žádnou cestu k cílovým polím.

Vzhledem k tomu, že Sector 66 umožňuje provádět více akcí než původní Quoridor, dá se předpokládat, že stavový prostor hry bude ještě několikanásobně větší. Z toho důvodu bude umělá inteligence vyřešena jen částečně a nebude podporovat sesílání kouzel.

Realizace

Vzhledem k výpočetní náročnosti generování stavů bude výpočet spojený s tímto problémem probíhat na pozadí v tolika vláknech, kolik je procesorových jader na

počítači, na kterém je hra spuštěna. Přesněji řečeno budou podporovány 1, 2 a 4 procesorová jádra (viz dále).

Výpočet u tříprocesorového počítače bude probíhat ve dvou vláknech a u více než čtyřprocesorového na čtyřech. Na základě strategie pro daný počet procesorů bude generování rozděleno tak, aby vznikalo co nejméně duplicit při co nejrovnoměrnějším rozdělení zátěže.

Rozdělení generování stavů na jiný počet dílčích úloh bude možné doplnit. Vytvoření dalších strategií ale může být složité, a proto budou vytvořeny pouze pro výše uvedený počet procesorových jader. Pro 1, 2 a 4 dílčí úlohy je rozdělení přímočaré, protože podle pravidel hry Sector 66 existují čtyři možné směry, podle kvadrantů, kam se může figurka hráče pohnout. U zábran je možné uvažovat každou n-tou pro n-tou dílčí úlohu a podobně.

Pro pohyb figurek se budou generovat všechny stavy, ale pro stavbu zábran jen ty, kde bude nová zábrana postavena dostatečně blízko od jiné již postavené zábrany nebo figurek hráčů (aktuálně proměnnou omezeno na vzdálenost 2 herní pole). Sesílání kouzel nebude do umělé inteligence zahrnuto z důvodů výpočetní náročnosti. Stavy se budou generovat do té doby, než bude naplněn paměťový limit. (Aktuálně nastaven proměnnou na 0,5 GB, aby bylo možné hru spustit tak, aby se vešla do operační paměti i starších počítačů za běhu operačního systému a případných dalších aplikací.) Poté bude výpočet zastaven do té doby, než nedojde k uvolnění části paměti.

Při dotazu na další stav budou pozastavena pracovní vlákna a proběhne sběr dat. Takto získané stavy budou ohodnoceny jednoduchou funkcí, která bude fungovat na základě počtu sloupců a řádek herního plánu mezi figurkou hráče a cílovým polem. Pro stavy, které mají ohodnoceného potomka, bude hodnota určena jako o jedna větší hodnota potomka.

Pokud stav, pro který se hledá následník, nebyl vygenerován, bude zvolen jeden z jeho bezprostředních potomků.

Při určení vhodného následníka dojde k pročištění stavů. Kvůli rychlosti tohoto čištění a vzhledem k tomu, že nové stavy se začnou generovat od stavu vybraného umělou inteligencí, budou vymazány všechny dosud vygenerované stavy.

Při generování stavů není možné se vyhnout duplicitám. Proto bude potřeba mechanismus, který rychle určí, zda nově vygenerovaný stav už existuje, nebo ne. Lineární prohledávání vygenerovaných stavů by bylo příliš pomalé. Proto bude použito binární vyhledávání, které operuje v čase $O(\log_2 n)$, kde n je počet existujících stavů. Stavy budou seřazeny podle hashovací funkce popsané níže.

Hashovací funkce pro rozlišení stavů hry

Aby bylo možné stavy co nejrychleji rozlišit, bude vytvořena hashovací funkce, která tento požadavek zajistí. Stavy hry budou v rámci řešení zařazeny do kolekcí ze jmenného prostoru *System.Collections.Generic*, které používají pro rozlišení objektů metodu *GetHashCode* a metodu *Equals*.

Metoda *GetHashCode* v jazyce C# vrací hodnotu typu integer, což je 32bitové číslo se znamínkem. Hashovací funkce bude využívat všech 32 bitů.

Vytvořená hashovací funkce bude fungovat na bázi příčné a podélné parity, která bude počítána z počtu zábran na řádku a sloupci. Posledních 8 bitů ve výsledném čísle zabere příčná parita (parita pro počet zábran v řádku). Dalších 8 bitů od konce bude rezervováno pro podélnou paritu (parita pro počet zábran ve sloupci). Dalších 14 bitů bude kódovat pozici prvních dvou hráčů a počet zábran, které mohou postavit. Poslední bit bude zaplněn zbytkem po dělení 2 indexu hráče, který je na tahu. Obrázek 5 naznačuje využití bitů hodnoty hashovací funkce.

Využití	Parita indexu hráče	nevyužitý bit	Parita počtu zábran hráče 2	Pozice hráče 2 - souřadnice Y	Pozice hráče 2 - souřadnice X	Parita počtu zábran hráče 1	Pozice hráče 1 - souřadnice Y	Pozice hráče 1 - souřadnice X	Parita 8. sloupce zábran	...	Parita 1. sloupce zábran	Parita 8. řádku zábran	...	Parita 1. řádku zábran
Pořadí bitů	0	1	2	3 - 5	6 - 8	9	10 - 12	13 - 15	16	17 - 22	23	24	25 - 30	31

Obrázek 5. Využití bitů hodnoty hashovací funkce

Ukazuje se, že tato funkce bude schopna rozlišit velké množství stavů hry. Kolidující stavy budou porovnány metodou *Equals* podle přesného rozmístění zábran, přesné pozice hráčů a všech dalších vlastností, který stav hry má.

2.8 Editor šablon

Tato kapitola rozšiřuje zadání v bodě 7 kapitoly 1.3. V souladu se zadáním by měl být vytvořen editor šablon, který bude umožňovat jejich pohodlné vytváření a upravování.

Vzhledově by měl maximálně připomínat samotnou hru, od které se bude lišit především tím, že nebude vynucovat pravidla pro tahy hráčů, ale bude kontrolovat smysluplnost rozmístění herních objektů. Tím by měl zajistit, že všichni hráči budou mít šanci vyhrát. Editor by měl například zabránit tomu, aby jeden z hráčů měl cestu k cílovým polím zastavěnou překážkami. Pro každého hráče by vždy měla existovat alespoň jedna cesta k cíli.

Samotné šablony by měly být uloženy v takovém formátu, aby bylo možné je jednoduše načítat v aplikaci. Pro uživatele nemusí být formát pochopitelný z důvodu existence editoru. Navíc je žádoucí, aby uživatelé nedělali náhodné změny v šabloně, čímž by mohli poškodit logiku hry. Přesto by načtená šablona měla být zkontrolována, zda splňuje invarianty dané pravidly.

Zdrojové kódy aplikace budou v maximální míře využívat jádro hry Sector 66, aby nebylo v případě změny nutné měnit pravidla jak v editoru, tak v samotné hře.

3 Analýza – Implementace

V této kapitole jsou popsány problémy a technologie, se kterými v rámci analýzy nebylo počítáno, a které bylo potřeba vyřešit při samotné implementaci, nebo svoji náplní přesahují předchozí kapitolu.

3.1 XNA

Pro práci s grafikou byl v rámci analýzy vybrán framework XNA 4.0 od firmy Microsoft.

XNA je možné nastavit, aby operovalo v jednom ze dvou módů.

HiDef je mód, který umožňuje používat všechny vlastnosti XNA bez jakéhokoliv omezení. Vyžaduje ale pokročilejší grafickou kartu operující nad DirectX 10. Toto omezení pak zároveň platí nejen pro kompilaci projektu, ale také pro spuštění výsledné aplikace. Při použití *HiDef* jsou zpřístupněny funkce, které nebude možné použít nad platformou Windows Phone.

Druhým módem je *Reach*. V něm jsou některé funkce omezeny. Například velikosti textur musí být zarovnány na mocninu 2. V tomto módu aplikace neklade téměř žádné nároky na grafickou kartu a operuje nad DirectX 9.

Jeden z požadavků analýzy je, aby hru bylo možné spustit i na starších, méně výkonných strojích. Proto projekt bude vyvinut a zkompilován v *Reach* módu, což umožní jeho jednodušší nasazení bez omezení kladeného na grafický výkon počítače. V projektu se nepředpokládá použití žádných složitých grafických efektů, takže zvolený mód nebude omezovat možnosti programu.

XNA využívá pro kódování barev strukturu *Microsoft.Xna.Framework.Color*. Obecně v .NETu se pro reprezentaci barvy nejčastěji používá struktura *System.Drawing.Color*, kde je barva uložená jako čtveřice čísel v rozmezí 0 až 255, které po řadě udávají červenou, zelenou a modrou složku. Poslední číslo je hodnota alfa, které udává průhlednost.

V XNA je kvůli rychlejší práci s grafikou barva uložena v přednásobeném formátu, který více vyhovuje prostředí DirectX. V *Microsoft.Xna.Framework.Color* se barva kóduje jako čtveřice čísel (R, G, B, alfa) s plovoucí desetinou čárkou mezi 0 a 1, kde 1 odpovídá 255. Častá konverze mezi těmito dvěma typy je relativně pomalá, a proto

bylo v projektu nutné využít cachování textur vypočítaných z obrázků dodaných v zásuvných modulech. Obrázky jsou v zásuvných modulech uloženy jako instance třídy *System.Drawing.Bitmap*, ze které se dají získat informace o barvě jednotlivých pixelů ve formátu barev takovém, jaký používá .NET obecně. V XNA je ovšem pro tvorbu textur nezbytné dodat barevné informace v přednásobeném formátu barev XNA.

Další nepříjemností bylo, že texturu, jako instanci třídy *Texture2D* z XNA není možné vytvořit bez instance třídy *GraphicsDevice*, která reprezentuje hardware pro práci s grafikou. Třídu *Texture2D* nejde serializovat, což vedlo k nutnosti uložit texturu jako pole barev, které odpovídají jednotlivým pixelům, aby bylo možné použít přenos mezi aplikačními doménami.

3.2 Zásuvné moduly

Protože sestavení XNA není označeno atributem *AllowPartiallyTrustedCallers*, neumožňuje volání metod z aplikačních domén, které mají omezené oprávnění. Při pokusu o takové volání vznikne výjimka s popisem „That assembly does not allow partially trusted callers“. Samotný jazyk C# ale takové omezení nemá.

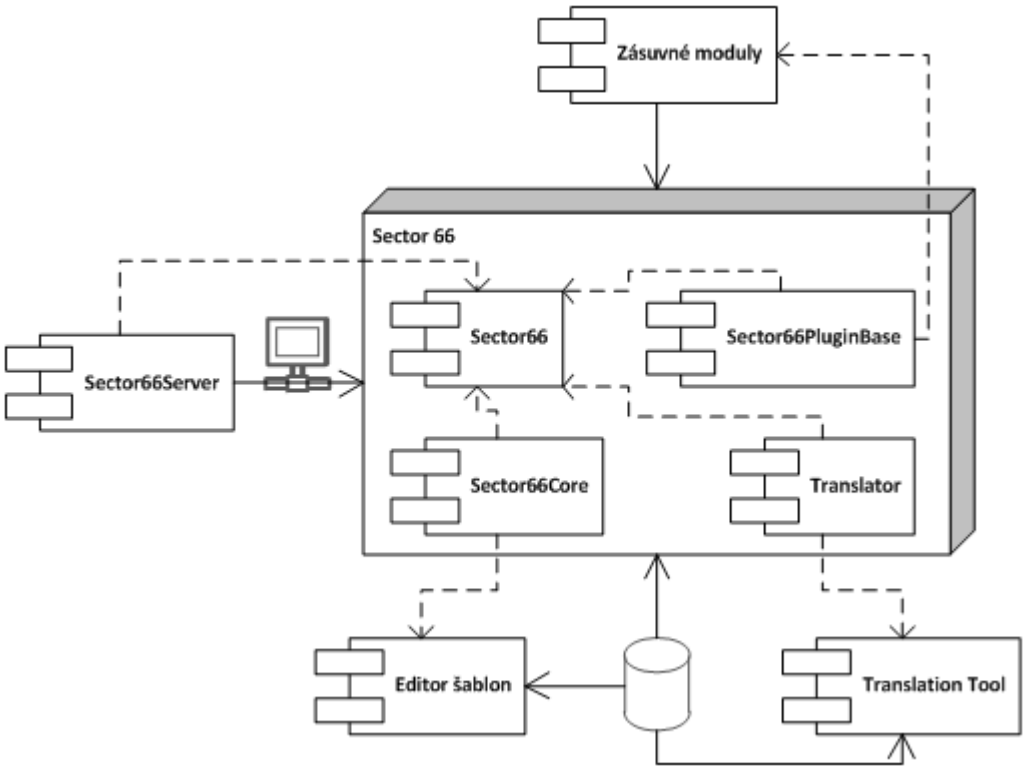
Posílání instance třídy *Bitmap* mezi aplikačními doménami se ukázalo jako velice pomále. Pravděpodobně je to způsobeno tím, že třída *Bitmap* je oděděna od třídy *MarshallByRefObject*, což má za následek, že při každém volání mezi doménami dojde k přenesení objektu mezi nimi. Získat kopii není přímočaře umožněno. Problém byl vyřešen tak, že data z třídy *Bitmap* byla uložena do streamu a ten potom jako pole bytů přenesen mezi aplikačními doménami, jak už bylo naznačeno výše. V cílové aplikační doméně je potom použit opačný proces na sestavení kopie původní bitmapy, která už je přístupná rychleji.

4 Vývojová dokumentace

V následující části jsou popsány moduly aplikace a je naznačeno, jakým způsobem spolu jednotlivé části komunikují. Bližší informace jsou popsány v příloze 3.

Zdrojové kódy byly vytvořeny v prostřední Microsoft Visual Studio 2010 v jazyce C# 4.0 pro .NET 4.0. Na přiloženém CD jsou ve složce *Zdrojové kódy*.

Obrázek 6 naznačuje, jakým způsobem spolu komunikují jednotlivá sestavení popsaná dále v této kapitole.



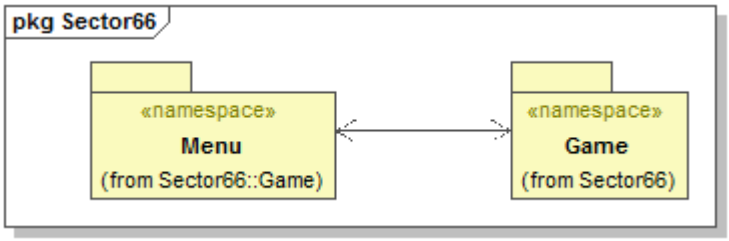
Obrázek 6. Komunikace mezi sestaveními Sectoru 66

Sestavení Sector 66 je rozděleno do projektů Sector66, Sector66Core a Sector66PluginBase. Tyto projekty jsou popsány po řadě v kapitolách 4.1, 4.2 a 4.3.

4.1 Sector 66

Sector 66 je hlavním projektem řešení. Je přeložen do spustitelné aplikace, která používá ostatní projekty. Je v něm část programu, která definuje vzhled okna aplikace a její ovládání.

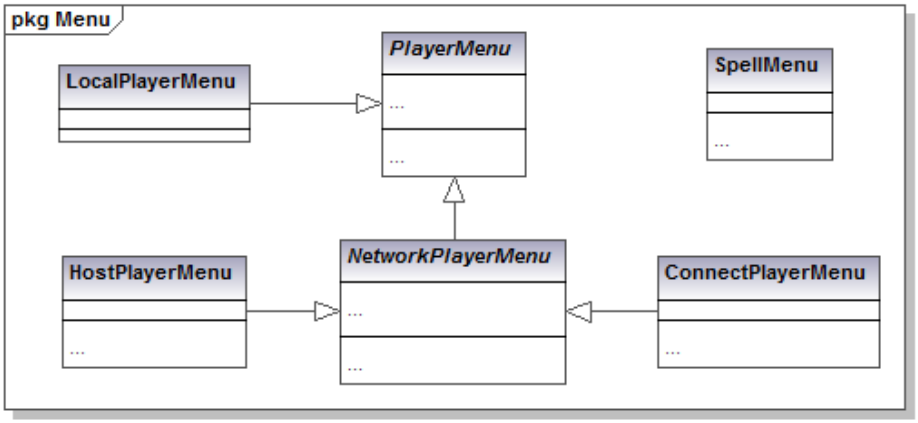
Projekt je rozdělen do dvou jmenných prostorů *Sector66.Game* a *Sector66.Game.Menu*. Obrázek 7 naznačuje, jak jsou na sobě závislé.



Obrázek 7. Závislost jmenných prostorů v projektu Sector 66

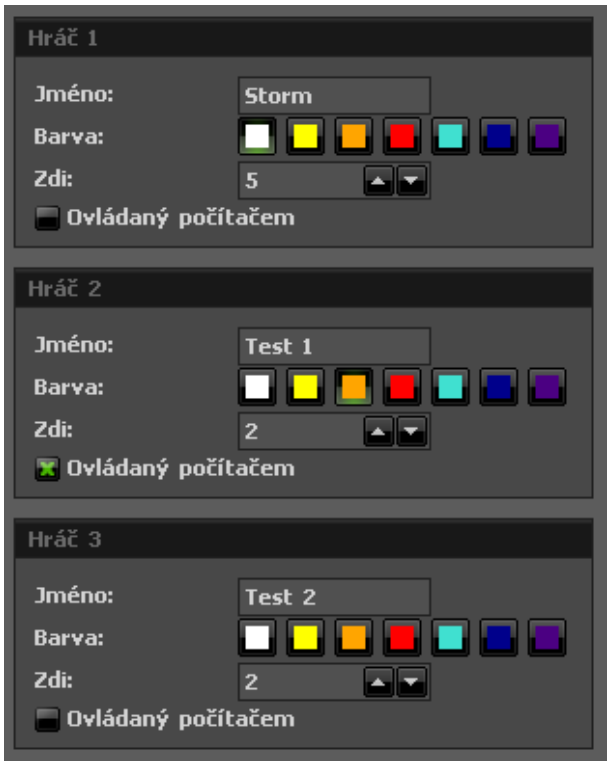
Jmenný prostor Menu

Obrázek 8 zachycuje strukturu tříd jmenného prostoru *Sector66.Game.Menu*.



Obrázek 8. Třídy jmenného prostoru Menu

Třída *PlayerMenu* definuje společné vlastnosti svých potomků, což je vzhled a umístění většiny ovládacích prvků, které v menu jsou. Je abstraktním předkem třídy *LocalPlayerMenu*, která slouží pro nastavení hráčů před startem hry. Obrázek 9 zachycuje její vzhled v aplikaci při nastavení pro tři hráče.

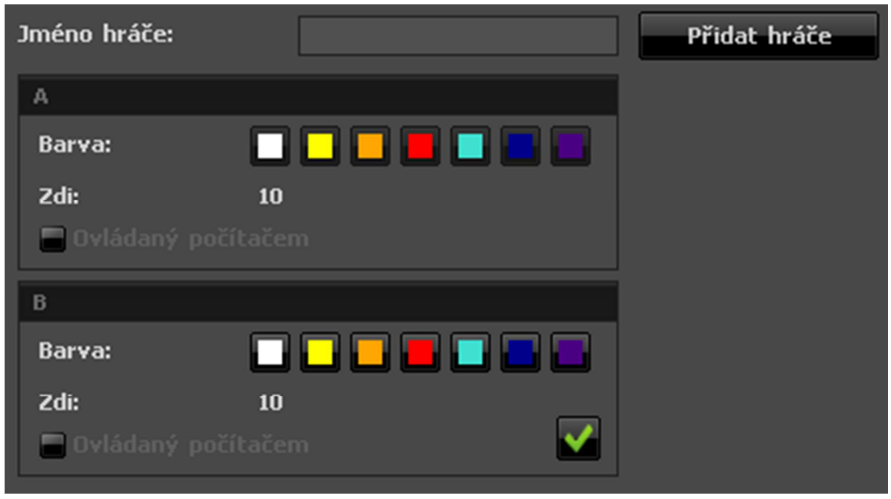


Obrázek 9. Vzhled LocalPlayerMenu

Abstraktní třída *NetworkPlayerMenu* navíc oproti svému předkovi přináší možnost komunikace se serverem hry, jehož stav se aktualizuje podle vstupu od uživatele. Navíc přidává ovládací prvky pro označení připravenosti hráče na hru a pro přidání dalšího hráče do hry.

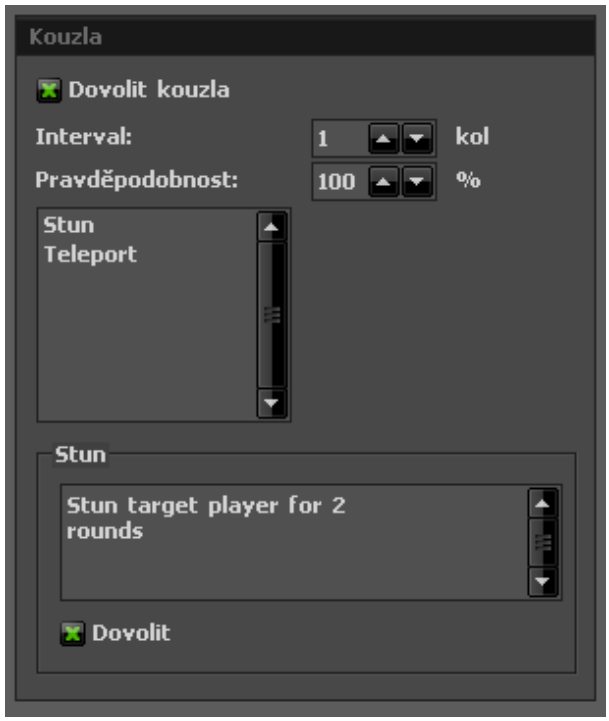
Od třídy *NetworkPlayerMenu* jsou odděděny třídy *HostPlayerMenu* a *ConnectPlayerMenu*, které po řadě slouží pro nastavení hry při zakládání síťové hry a pro nastavení při připojování k síťové hře. Liší se tím, že v případě *HostPlayerMenu* je možné hráčům nastavit počet zábran, které budou mít k dispozici, v případě *ConnectPlayerMenu* je tato informace pouze zobrazena. Podobně je tomu s nastavením hráčů ovládaných počítačem. Obrázek 10 a 11 zobrazuje vzhled těchto dvou tříd.

Obrázek 10. HostPlayerMenu s jedním lokálním hráčem



Obrázek 11. ConnetPlayerMenu s hráčem A vzdáleným a hráčem B lokálním

Třída *SpellMenu* slouží pro nastavení kouzel v rámci hry. Toto menu je použito při zakládání lokální i síťové hry. Obrázek 12 naznačuje jeho vzhled.



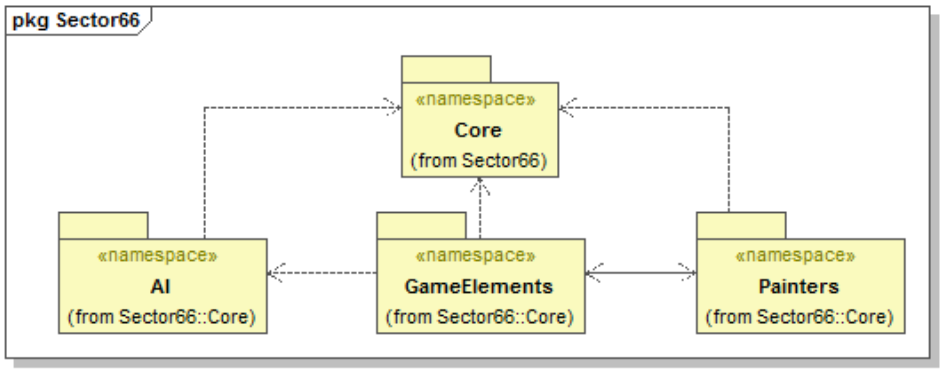
Obrázek 12. Vzhled SpellMenu

Jmenný prostor Game

Jmenný prostor *Sector66.Game* obsahuje třídu *Sector*, která definuje chování hry a třídu *Program*, která obsahuje metodu *Main*, jakožto hlavní vstupní bod aplikace, ve kterém je instance třídy *Sector* spuštěna.

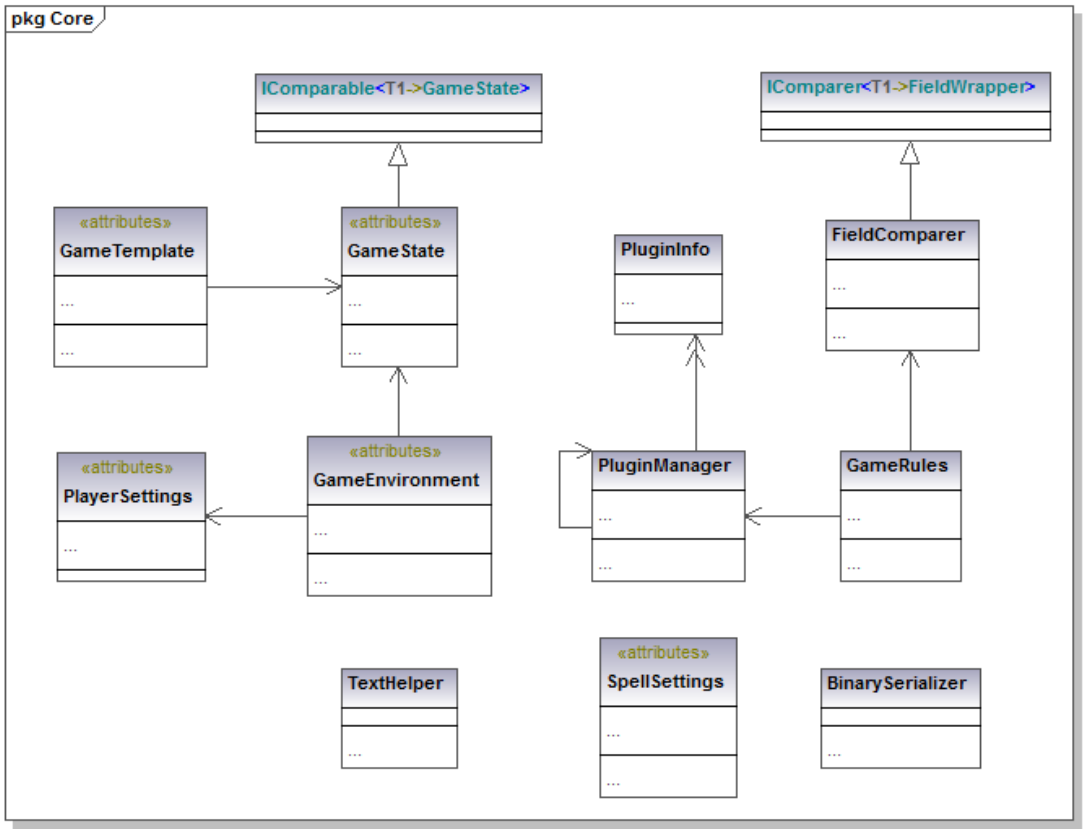
4.2 Sector66Core

Projekt Sector66Core se skládá ze jmenných prostorů *Sector66.Core*, *Sector66.Core.GameElements*, *Sector66.Core.Painters* a *Sector66.Core.AI*. Obrázek 13 znázorňuje jejich závislosti.



Obrázek 13. Závislosti jmenných prostorů v projektu Sector66Core

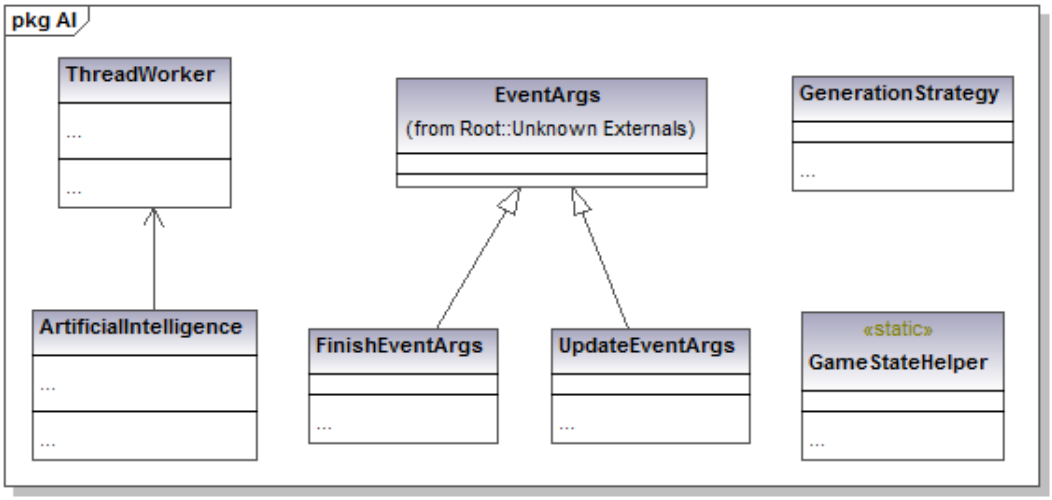
Jmenný prostor *Sector66.Core* obsahuje třídy reprezentující stav hry (*GameState*) a její prostředí (*GameEnvironment*). Také obsahuje třídu pro správu zásuvných modulů (*PluginManager*) a vedle podpůrných tříd a tříd pro správu nastavení (*PlayerSettings*, *SpellSettings*) obsahuje třídu reprezentující pravidla hry (*GameRules*). Také obsahuje třídu reprezentující šablonu hry (*GameTemplate*). Obrázek 14 ukazuje závislosti tříd tohoto jmenného prostoru.



Obrázek 14. Třídy jmenného prostoru Sector66.Core

Jmenný prostor *Sector66.Core.AI* obsahuje třídy umělé inteligence.

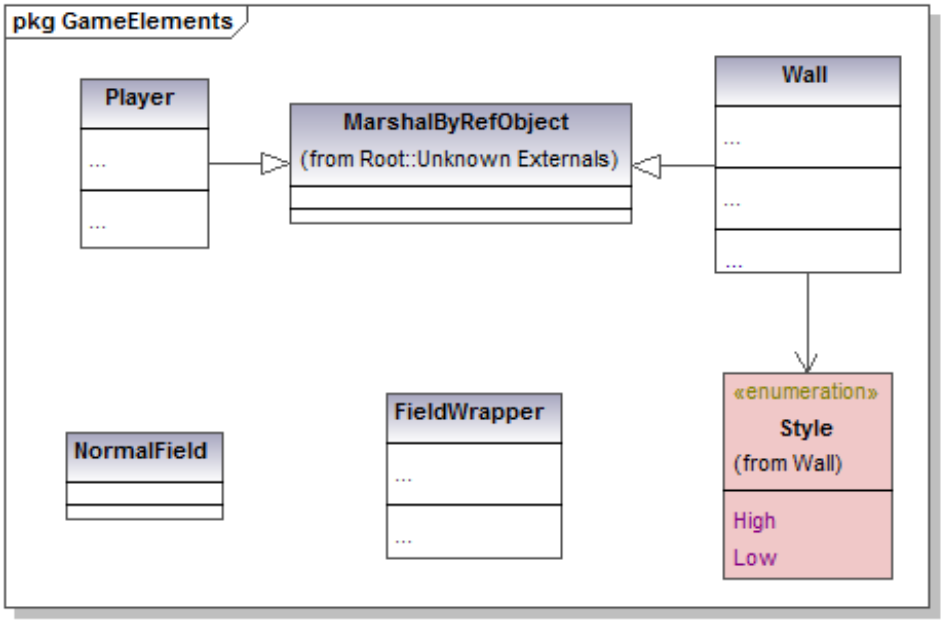
Třída *ArtificialIntelligence* tvoří hlavní část tohoto projektu. Třída *ThreadWorker* reprezentuje vlákno, které se stará o generování stavů a třída *GenerationStrategy* určuje, které stavy připadnou kterému vláknu. Třídy *FinishEventArgs* a *UpdateEventArgs* slouží jako argumenty událostí, které jsou vyvolány po skončení výpočtu umělé inteligence a po aktualizaci stavu výpočtu. Obrázek 15 zobrazuje jejich závislosti.



Obrázek 15. Třídy jmenného prostoru Sector66.Core.AI

Jmenný prostor *Sector66.Core.GameElements* obsahuje herní objekty, které nejsou určeny k rozšiřování pomocí zásuvných modulů, jako například třídu reprezentující hráče. Obrázek 16 znázorňuje závislosti v projektu.

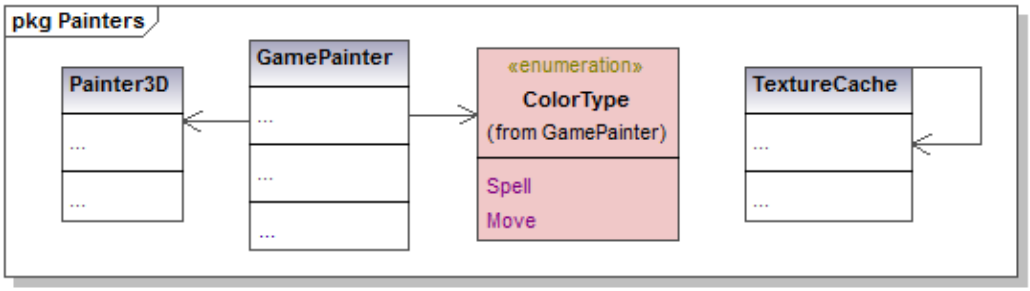
Třídy *Player*, která reprezentuje hráče, a *Wall*, která reprezentuje zábranu, implementují po řadě rozhraní *IPlayer* a *IWall* ze jmenného prostoru *Sector66.PluginBase* (viz kapitola 4.3). Ze třídy *Field* ze stejného jmenného prostoru je oděděna třída *NormalField*, která reprezentuje běžné políčko herního plánu. Výčet *Style* určuje, jak bude vypadat zábrana na herním plánu. Třída *FieldWrapper* obaluje instance tříd reprezentující pole herního plánu spolu s kouzly, které se na nich mohou vyskytovat a body určujících jejich tvar spolu s jejich texturami.



Obrázek 16. Objekty jmenného prostory Sector66.Core.GameElements

Důležitou součástí projektu je jmenný prostor *Sector66.Core.Painters*, který obsahuje objekty vykreslující 3D herní prvky. Obrázek 17 zachycuje jeho třídy.

Třída *GamePainter* je určena k vykreslování herních objektů, jako například hráčů, políček a zábran. Informace o vzhledu objektů jsou předány třídě *Painter3D*, která se stará o vykreslení bodů. Výčet *ColorType* určuje o jaký typ zvýraznění objektu se jedná a podle něho je vybrána správná barva. Třída *TextureCache* slouží jako cache pro uchování textur herních objektů.



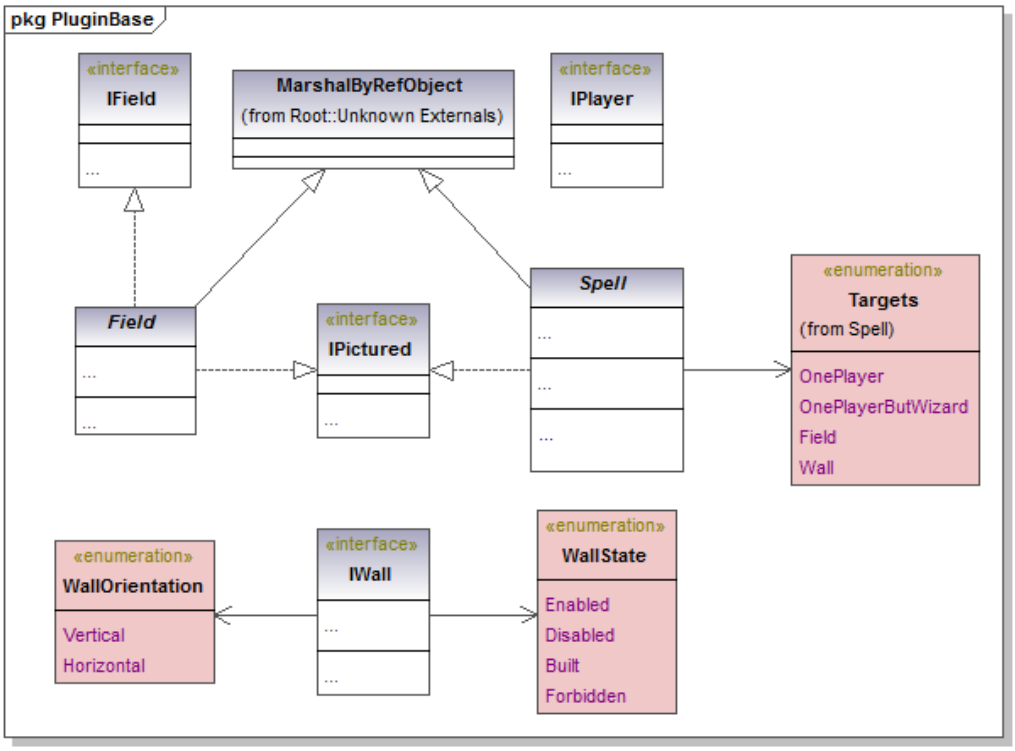
Obrázek 17. Objekty jmenného prostory Sector66.Core.Painters

4.3 Sector66PluginBase

Projekt Sector66PluginBase je navržen jako co nejmenší množina tříd a rozhraní určených pro tvorbu zásuvných modulů hry. Obrázek 18 zobrazuje strukturu tříd.

Rozhraní *IField* a *IPlayer* jsou určeny pro komunikaci zásuvných modulů s poli herního plánu a s figurkou hráče. Rozhraní *IPictured* je implementováno všemi herními objekty, kterým může programátor zásuvného modulu přiřadit texturu.

Rozhraní *IWall* je určeno pro komunikaci zásuvného modulu se zábranou. Výčty *WallOrientation* a *WallState* určují, jak je zeď orientována v herním plánu a jaký je její stav ve hře. Třídy *Field* a *Spell* jsou popsány v kapitolách 4.10.1 a 4.10.2 a slouží jako základ pro zásuvné moduly polí herního plánu a kouzel.

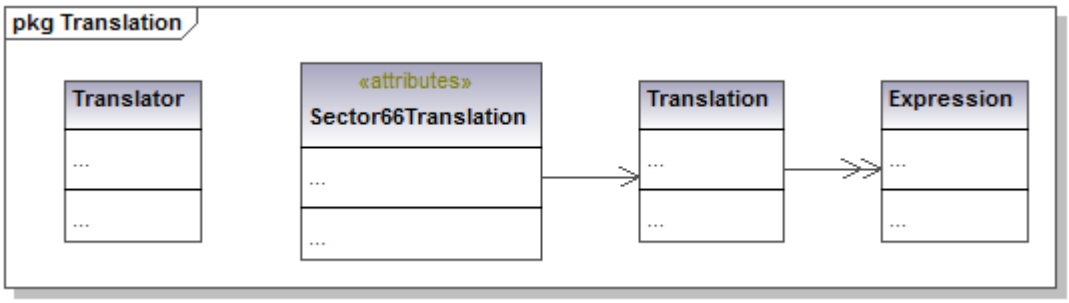


Obrázek 18. Struktura tříd projektu Sector66PluginBase

4.4 Translator

Projekt Translator má na starost načítání a ukládání překladů hry. Přímo v sobě nese výchozí anglickou lokalizaci hry. Obrázek 19 zachycuje závislosti jeho tříd. Tyto závislosti zároveň vytváří strukturu XML souboru s překladem, který je možné najít v příloze 2.

Hlavní součást projektu tvoří třída *Translator*, která se stará o načítání překladů a také o vyhledávání frází podle klíčů. Ostatní třídy tvoří strukturu souboru s překladem.



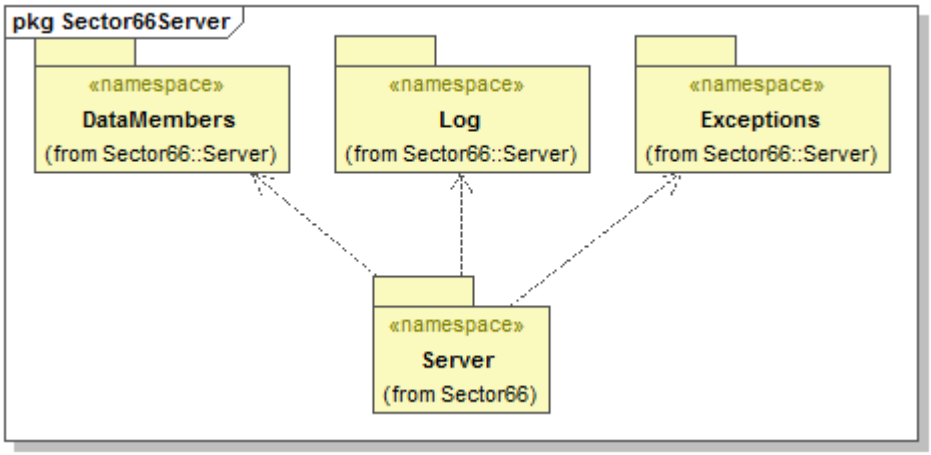
Obrázek 19. Třídy jmenného prostory Sector66.Translator

4.5 Sector66TemplateEditor

Většina zdrojového kódu obsaženého v projektu Sector66TemplateEditor definuje vzhled editoru šablon hry. Druhá část se stará o načítání a ukládaní šablon a o reakci na vstup od uživatele. Pro kontrolu rozmístění prvků hry a její nastavení je použit projekt Sector66Core. Obsahuje proto jenom dvě třídy. *Sector66TemplateEditor* a třídu *Program*, která obsahuje vstupní bod programu v podobně metody *Main*.

4.6 Sector66Server

Sector66Server je projekt, který v sobě sdružuje funkci herního serveru, který udržuje stav síťových her a slouží jako služba, vůči níž se aktualizují klienti připojení k síťovým hrám. Obsahuje čtyři jmenné prostory. Obrázek 20 naznačuje jejich závislost.



Obrázek 20. Závislost jmenných prostorů projektu Sector66Server

4.7 Sector66ServerLauncher

Sector66ServerLauncher je velmi jednoduchým projektem, který načte knihovnu vzniklou kompilací projektu Sector66Server a spustí tak herní server. Také obsahuje konfigurační soubor *Sector66ServerLauncher.exe.config* (viz kapitola 5.4), ve kterém je možné nastavit chování serveru.

4.8 Sector66ServiceLauncher

Podobně jako Sector66ServerLauncher je projekt Sector66ServiceLauncher určen pro hostování herního serveru. V případě tohoto projektu ve formě služby operačního systému Windows. Soubor *Sector66ServiceLauncher.exe.config* slouží pro

konfiguraci serveru. Postup hostování služby i konfigurační soubor jsou popsány v kapitole 5.4.

4.9 TranslationTool

TranslationTool je projekt jehož překladem vznikne jednoduchá okenní aplikace sloužící k vytváření překladů. Tento projekt využívá části definované v projektu *Translator*.

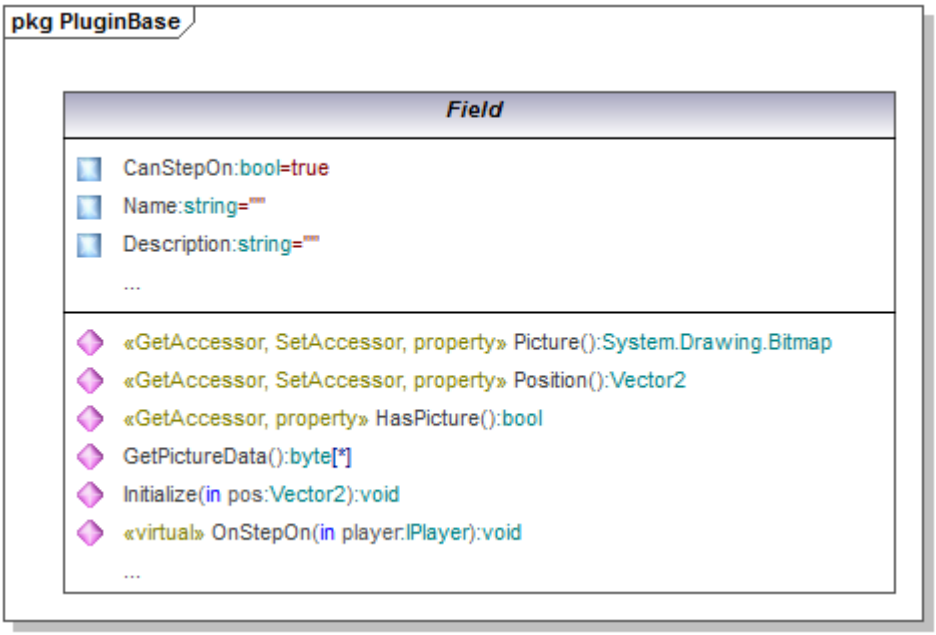
4.10 Tvorba zásuvných modulů

Pro vytvoření vlastního zásuvného modulu kouzla nebo pole herního plánu je nutné vytvořit projekt s referencí na knihovnu *Sector66PluginBase.dll* a tento projekt zkompilovat do podoby knihovny DLL. Projekt Sector66PluginBase je popsán v kapitole 4.3. V této knihovně jsou umístěny potřebné třídy a rozhraní pro komunikaci s hrou. Zásuvné moduly je poté nutné nakopírovat do složky *FieldPlugins* v případě, že se jedná o zásuvný modul pole herního plánu, nebo do složky *SpellPlugins*, pokud se jedná o zásuvný modul kouzla.

Nejjednodušší způsob, jak vytvořit zásuvný modul je použít jeden z ukázkových, které jsou součástí řešení a přepsat požadované vlastnosti. Tyto ukázky je možné nalézt v projektech ForbiddenField, StunSpell a TeleportSpell.

4.10.1 Zásuvný modul pole herního plánu

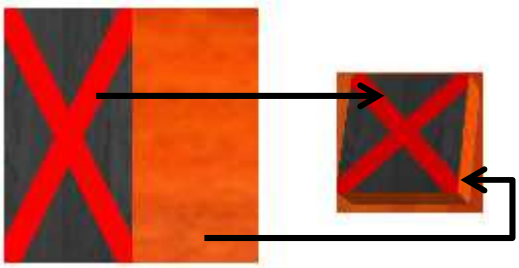
V případě vytváření vlastního pole herního plánu je nutné třídu, která ho reprezentuje oddědit od třídy *Field*. Obrázek 21 ukazuje její strukturu. Pro nastavení vlastností pole je určena metoda *OnStepOn*, která ve výchozím stavu nepřidává žádnou funkcionalitu. Tato metoda je virtuální a je ji tedy možné přepsat požadovanou implementací. Nastavením vlastnosti *CanStepOn* se dá povolit nebo zakázat vstup figurky na nově vytvořené pole.



Obrázek 21. Diagram třídy Field

Je důrazně doporučeno nastavit vlastnosti *Name* a *Description*, které určují jméno pole a jeho popis, který umožňuje uživatelům zjistit efekt zásuvného modulu.

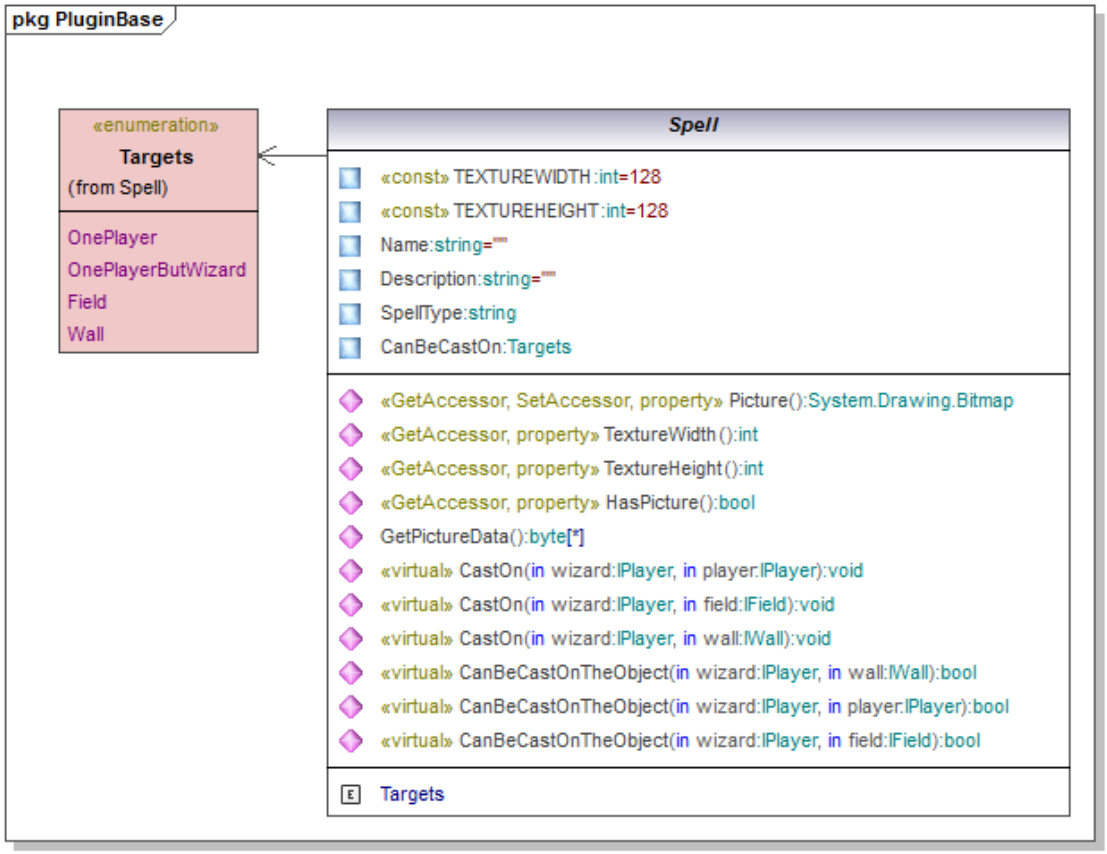
Aby bylo možné pole zobrazit ve hře, musí mít nastavenou texturu. K tomu slouží vlastnost *Picture*, do které je tvůrce modulu povinen přiřadit obrázek reprezentující texturu. Je velmi vhodné, aby tento soubor byl součástí knihovny, k čemuž se dají použít *Resources*. Textura by měla mít velikost 128×128 pixelů. Pokud tyto rozměry mít nebude, dojde ke změně velikosti při načítání zásuvného modulu do hry, což muže vést ke zpomalení aplikace. Obrázek 22 ukazuje formát textury. Levá polovina textury reprezentuje horní plochu pole a pravá polovina okraje. Pokud textura nebude nastavena, pole se nepodaří připojit do hry.



Obrázek 22. Formát textury pole herního plánu

4.10.2 Zásuvný modul kouzla

Třídu zásuvného modulu kouzla je nutné oddědit od třídy *Spell*. Obrázek 23 zobrazuje její strukturu.



Obrázek 23. Struktura třídy Spell

Ve výchozím stavu bez nastavení vlastností a přepsání metod je kouzlo nepoužitelné. Aby bylo možné zásuvný modul připojit do hry, opět musí být nastavena textura. K tomu slouží vlastnost *Picture*. Rozměry obrázku by opět měli být 128 × 128 pixelů. Podobně jako v předchozím případě by měly být nastaveny vlastnosti *Name* a *Description*.

Pro správnou funkci kouzla je také nezbytné nastavit vlastnost *CanBeCastOn*, která určuje, na jaký herní objekt bude možné kouzlo zacílit. Pokud je potřeba nastavit nějakou další podmínku na cíl, je možné přepsat metodu *CanBeCastOnTheObject*, která je přetížena pro všechny herní objekty. Ve výchozím stavu tato metoda nijak neomezuje cíl kouzla.

Po nastavení samotného efektu kouzla slouží metoda *CastOn*, která je opět přetížena pro všechny herní objekty. Stejně jako v případě metody *CanBeCastOnTheObject* stačí přepsat pouze to přetížení, které se týká požadovaného cílového objektu hry.

5 Uživatelská dokumentace

V následujících kapitolách je popsán způsob práce s hrou Sector 66 v české lokalizaci, editorem jejích šablon, serverem hry a nástrojem pro vytváření překladů.

Žádná část aplikace nevyžaduje instalaci. Programy je možné spustit z libovolného umístění, kam bude aplikacím umožněn zápis. Proto je pro spuštění nutné zkopírovat Sector 66 z přiloženého CD na pevný disk počítače nebo jiné vhodné umístění. Detaily o spuštění jednotlivých součástí je možné najít níže.

5.1 Sector 66

Hru Sector 66 je možné spustit voláním souboru *Sector66.exe*. Ke svému běhu vyžaduje nainstalovaný .NET 4.0 a XNA Redistributable 4.0. Obě tyto komponenty jsou k legálním Windows zdarma dostupné z internetových stránek Microsoftu.

V adresářové struktuře hry jsou tyto složky:

- Content – obsahuje grafiku a efekty nutné pro spuštění hry
- FieldPlugins – složka se zásuvnými moduly políček herního plánu
- SpellPlugins – složka se zásuvnými moduly kouzel
- Templates – složka s šablonami her
- Translations – složka s překlady hry

Ke spuštění hry je nezbytná pouze složka Content.

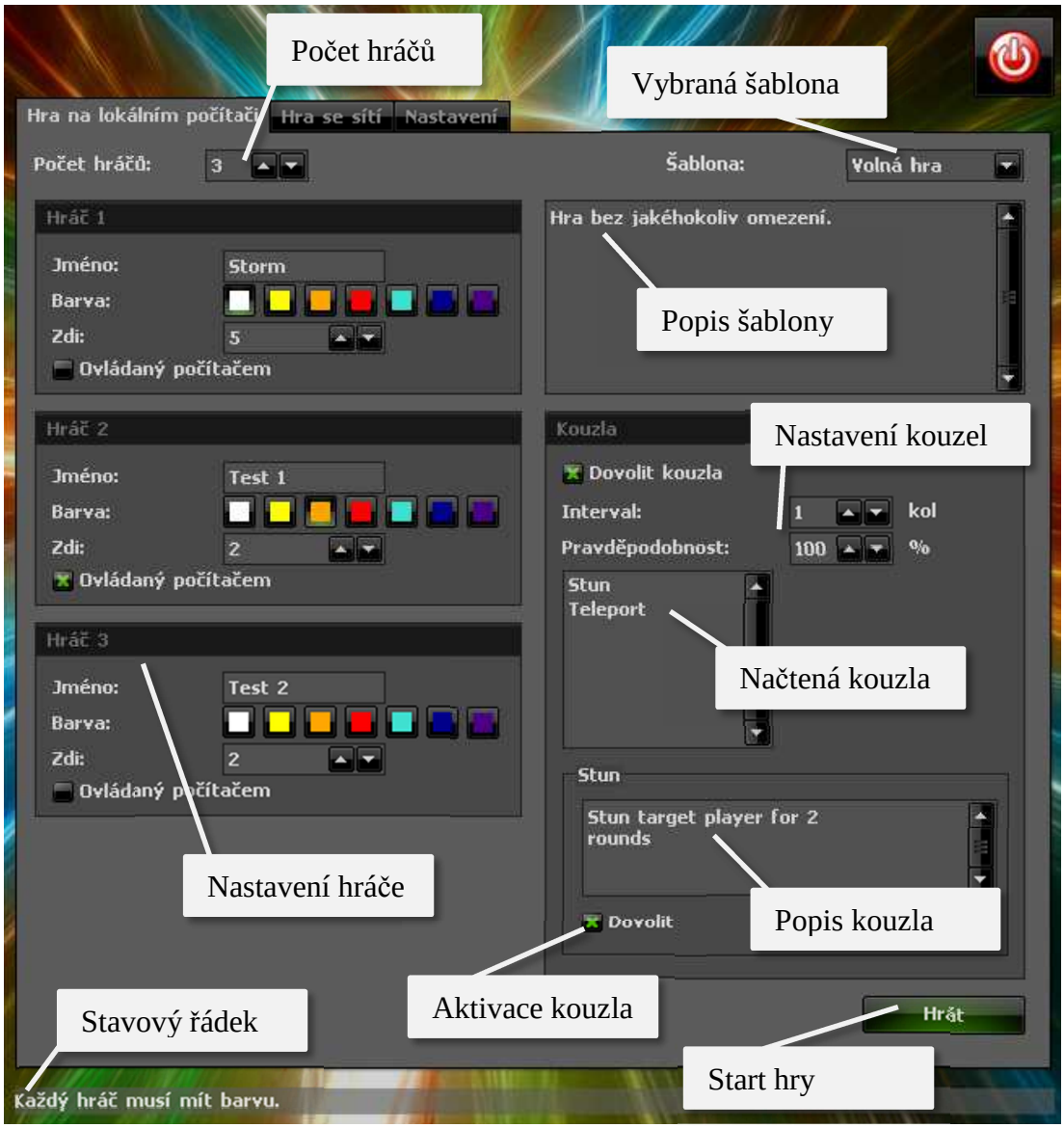
Okno aplikace je rozděleno záložkami do sekcí „Hra na lokálním počítači“, „Hra se sítí“ a „Nastavení“.

Nad hlavní částí okna je umístěno tlačítko pro ukončení aplikace a pod ní je umístěn řádek, kam se vypisuje stav aplikace nebo případné chybové výstupy.

5.1.1 Hra na lokálním počítači

V sekci „Hra na lokálním počítači“ je možné nastavit vlastnosti hry, která bude hrána ve více hráčích na jednom počítači. Za některé z nich může hrát počítač.

Obrázek 24 ukazuje okno aplikace sloužící pro nastavení spouštěné lokální hry.



Obrázek 24. Okno nastavení spouštěné hry

V horní levé části okna je možné nastavit počet hráčů v rozsahu od dvou do čtyř v závislosti na šabloně.

Vpravo nahoře si uživatel může vybrat jednu ze šablon uložených ve složce *Templates*, která určuje základní rysy hry. S některými šablonami nemusí být možné měnit některé vlastnosti hry. Pod výběrem šablony je místo pro popis, který zadal uživatel při vytváření zvolené šablony.

Podle nastavení počtu hráčů se v levé části okna objeví příslušný počet oblastí s možností jejich nastavení. Před startem hry musí mít každý hráč uvedeno unikátní jméno v rámci hry, které musí obsahovat alespoň jeden tisknutelný znak a také musí mít nastavenou unikátní barvu. Pro hráče je možné nastavit, kolik budou mít k dispozici zábran a také jestli za ně bude hrát počítač, nebo ne.

V pravé části okna se nastavují kouzla, která bude možno použít ve hře, stejně jako pravděpodobnost, že se objeví nové a interval, ve kterém se provede pokus o jejich vytvoření. Po výběru kouzla ze seznamu se ve spodní části příslušné oblasti okna objeví popis, který zadal tvůrce zásuvného modulu.

Hra je spuštěna po kliknutí na tlačítko „Hrát“, které se nachází ve spodní pravé části okna.

5.1.2 Hra se sítí

Nejprve je v horní části nutné zadat adresu a port serveru, ke kterému se uživatel chce připojit ve formátu adresa serveru:port, například 192.168.1.1:8066, nebo localhost:8066. Výchozí číslo portu pro herní server je 8066. I to je však nutné zadat. Po kliknutí na tlačítko „Připojit“ se provede pokus o nalezení a připojení ke specifikovanému serveru. Tlačítkem „Reset“ je možné toto připojení zrušit.

Po úspěšném kontaktování serveru si uživatel vybere, jestli chce vytvořit novou hru, nebo jestli se chce připojit k existující výběrem příslušné záložky.

Založení hry

Při zakládání nové hry uživatel specifikuje její jméno, heslo, popis, šablonu hry a maximální počet hráčů. Hra zvoleného jména nesmí na serveru v době vytváření existovat. Uživatel také uvede jméno hráče, pod kterým se k nově vytvořené hře připojí. Popis, který slouží pro informování ostatních hráčů o charakteru založené hry a heslo nemusí být vyplněny.

Obrázek 25 zobrazuje okno pro založení síťové hry.

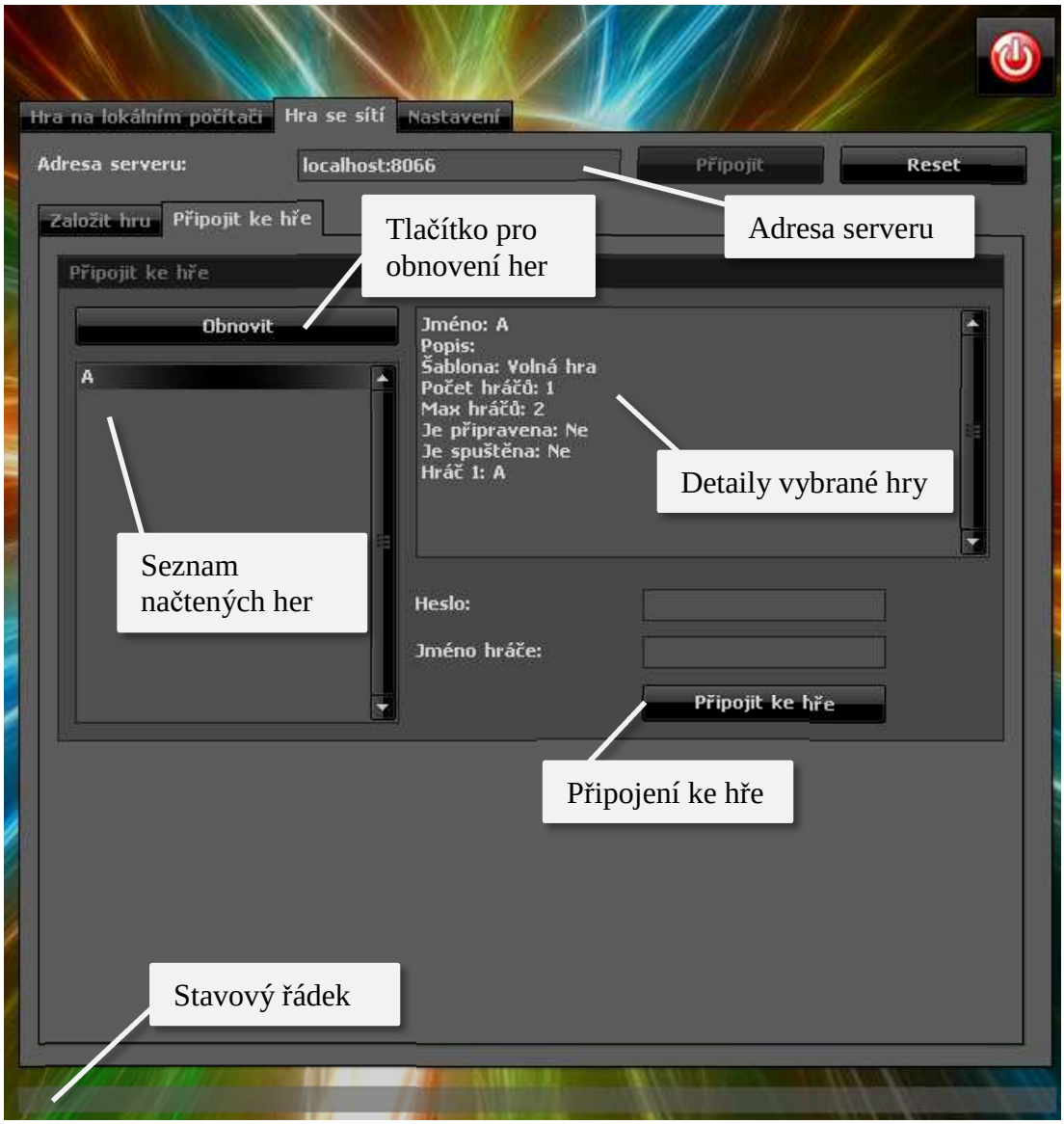
Obrázek 25. Okno pro založení síťové hry

Po kliknutí na tlačítko „Vytvořit hru“ se zkontrolují výše popsané invarianty, a pokud souhlasí, je hra vytvořena.

Po úspěšném vytvoření hry se objeví okno, které je podobné oknu při vytváření lokální hry. Jedinými rozdíly je, že v horní části přibyl řádek pro přidávání hráčů, kteří budou hrát ze stejného počítače, na kterém je prováděno nastavení. Druhý rozdíl spočívá v tom, že hru je možné spustit až tehdy, když jsou všichni hráči připraveni. To se pozná tak, že mají buď promáčknuté tlačítko se zeleným zátržítkem v případě lokálních hráčů, nebo zobrazen podobný symbol, pokud jsou připojeni vzdáleně. Hra se spouští podobně jako lokální kliknutím na tlačítko „Hrát“.

Připojení k existující hře

V levé části obrazovky je seznam her, které jsou na serveru. Nad tímto seznamem je tlačítko „Obnovit“, které vyžádá ze serveru aktuální seznam her. V pravé části okna se zobrazují informace o vybrané hře. Obrázek 26 zobrazuje okno pro připojení k existující hře.



Obrázek 26. Okno pro připojení k existující hře

Pro připojení ke zvolné hře uživatel musí zadat heslo, které bylo zadáno při jejím vytváření a jméno, pod kterým se ke hře připojí. To musí být v rámci hry unikátní. Pokud při vytváření hry bylo zadáno prázdné heslo, může zůstat nevyplněné i zde.

Po úspěšném přihlášení se zobrazí podobné okno, jako v případě vytváření hry. Není ale možné měnit nastavení kouzel, ani některé vlastnosti hráčů, jako například počet zábran, které budou mít k dispozici. Toto nastavení provádí uživatel, který hru

založil. Aby bylo možné hru spustit, musí všichni uživatelé postupně promáčknout tlačítko se zeleným zátržítkem, čímž dávají najevo, že jsou připraveni ke startu hry. Hru spouští ten, kdo ji založil.

Hraní hry

Po spuštění ať už síťové, nebo lokální hry se objeví okno zobrazující herní plán a figurky hráčů. Obrázek 27 ukazuje vzhled tohoto okna. V záhlaví je zobrazeno jméno hráče, který je na tahu. Okolo jeho figurky jsou zvýrazněna pole, na která je možné udělat tah. Barvou hráče jsou na herním plánu zvýrazněna cílová pole, na která se má dostat jeho figurka.

Obrázek 27. Okno hry

V dolní části je zobrazeno menu, které zleva obsahuje tlačítko pro stavbu zábran, počet zábran protihráčů, kouzla hráče a vpravo tlačítko pro ukončení kola a tlačítko pro vstup do menu.

Po kliknutí na tlačítko pro stavbu zdí se podle toho, kde se nachází myš, zvýrazňují zábrany, které je možné kliknutím postavit. Podobným způsobem je možné aktivovat kouzla. Kliknutím na příslušné tlačítko se kouzlo vybere a kliknutím na herní objekt ve hře se kouzlo sešle.

Po dokončení akce hráče je nutné kliknout na tlačítko pro ukončení kola, které se nachází v pravé dolní části okna. Tím se umožní dalšímu hráči v řadě, aby provedl svůj tah.

Kliknutím na tlačítko „Menu“, se zobrazí menu, které umožňuje návrat do hlavní nabídky aplikace, čímž se zruší hra. Na stejném místě jde také přepnout režim hraní na celé obrazovce, což jde udělat i stiskem klávesové zkratky <Alt> + <Enter>.

K většině akcí ve hře se zobrazují informace nad tlačítky pro ukončení kola a vstupu do menu.

Rozehranou hru není možné uložit, ač to samotný program umožňuje.

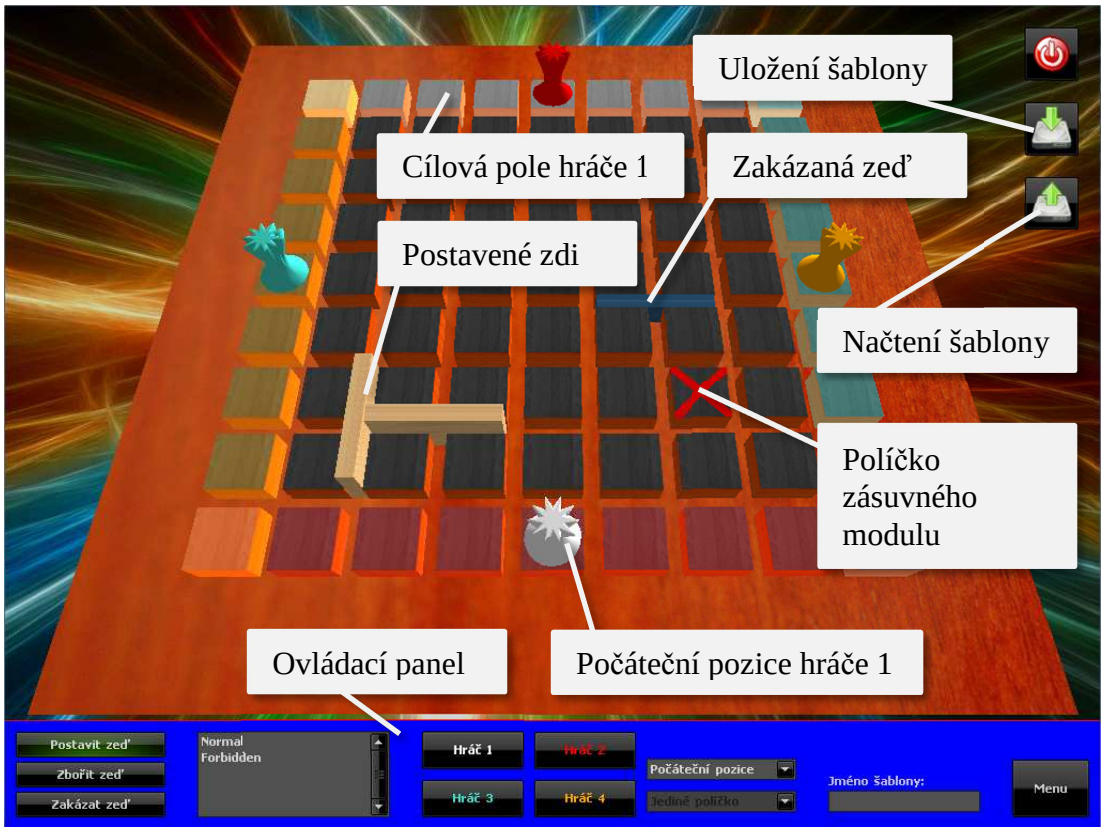
5.2 Sector 66 – Template Editor

Sector 66 – Template Editor je nástroj k vytváření a upravování šablon her. Tyto šablony umožňují do jisté míry modifikovat pravidla původního Sectoru 66, jako například nastavovat minimální a maximální počet hráčů, nebo třeba měnit pole herního plánu.

Aplikace se spouští voláním souboru *Sector66TemplateEditor.exe* a pro jeho spuštění musí být na počítači nainstalován .NET verze 4.0 a XNA Redistributable verze 4.0 podobně jako v případě samotné hry.

Editor šablon využívá stejnou strukturu složek jako samotná hra.

Po spuštění editoru se objeví okno zobrazující herní pole podobné jako hra Sector 66. Obrázek 28 toto okno zobrazuje. Ve spodní části okna jsou rozmístěny ovládací prvky. Zleva to jsou tlačítka pro stavbu, bourání a zakázání zábran, výběr polí herního plánu a nastavení počáteční a koncové pozice jednotlivých hráčů. Vpravo je řádek pro vyplnění jména šablony a tlačítko, které zobrazí menu s dalšími volbami. V pravé horní části je tlačítko pro ukončení aplikace, uložení a načtení šablony z disku.



Obrázek 28. Okno editoru šablon

Kliknutím na tlačítko akce ve spodní části okna, nebo na typ pole se aktivuje režim umisťující vybraný prvek do herního plánu. Umístění se provede kliknutím na zvolenou pozici.

Při nastavení pozic hráče je nutné vybrat, zda půjde o koncovou, nebo počáteční pozici a u koncové ještě, zda cílem bude sloupec, řádek, nebo jedno pole. Tlačítka „Hráč 1“ až „Hráč 4“ určují, pro koho se dané nastavení aplikuje.

Kliknutím na tlačítko „Menu“ se přeruší úprava herního plánu a otevře se okno, ve kterém jde nastavit minimální a maximální počet hráčů, počet zdí a popis šablony. Na další záložce je umístěno nastavení kouzel. Tady se dá specifikovat, zda budou ve hře kouzla povolena a případně jaká to budou. Také lze nastavit, zda se budou dát použít jiná, než vybraná kouzla. Stejně tak minimální a maximální interval v počtu kol, mezi generováním nového kouzla a pravděpodobnost, se kterou se kouzlo objeví.

Na poslední záložce je nastavení aplikace, kde se dá zvolit jazyk a mód plné obrazovky.

Šablonu je možné uložit, pokud má vyplněný název a popis. Dvě šablony stejného jména nemohou existovat současně. Šablona se uloží do složky *Templates* a tím je

hned zpřístupněna ve hře Sector 66. Šablona půjde použít pouze za předpokladu, že jsou ve hře přístupné všechny zásuvné moduly, které byly při vytváření šablony použity.

5.3 Translation Tool

Translation Tool je jednoduchá aplikace pro vytváření překladů hry Sector 66. Spouští se voláním souboru *TranslationTool.exe*. Aby bylo možné aplikaci spustit, je nutné mít v počítači nainstalovaný .NET 4.0. Tato aplikace je lokalizována do češtiny a angličtiny. Popisována bude v české lokalizaci.

V roletě „Překlad“ je možné načíst nebo uložit překlad, který se objeví v posledním sloupci hlavní části aplikace. Roleta „Referenční překlad“ ovládá druhý sloupec hlavní části okna, který slouží jako podklad pro vytváření překladu ve třetím sloupci. První sloupec zobrazuje klíče, které identifikují překládané fráze v aplikaci.

Před uložením překladu je nutné v záhlaví okna vyplnit jazyk překladu, jeho autora a verzi.

5.4 Sector 66 – Server

Server hry Sector 66 se skládá z následujících tří částí. Knihovny *Sector66Server.dll*, která obsahuje samotnou funkcionalitu serveru a spustitelné aplikace *Sector66ServerLauncher.exe* a *Sector66ServiceLauncher.exe*.

Sector66ServerLauncher.exe načte danou knihovnu a spustí tím samotný server. Jeho nastavení je možné ovlivnit editací souboru *Sector66ServerLauncher.exe.config*. Ve výchozím nastavení bude server spuštěn na portu 8066. Pokud se server spustí tímto způsobem, otevře se konzolová aplikace, která textově zobrazuje akce probíhající na serveru.

Sector66ServiceLauncher.exe je služba operačního systému Windows, která spouští server hry Sector 66. Její instalaci je možné provést z příkazové řádky Windows příkazem:

```
sc create [název služby] [binPath= ”Adresa souboru Sector66ServiceLauncher.exe”]
```

Odinstalování služby lze provést opět z příkazové řádky příkazem:

```
sc delete název služby
```

Po nainstalování je službu možné spustit z okna nastavení služeb operačního systému Windows.

Službu je možné konfigurovat podobně jako v předchozím případě editací konfiguračního souboru *Sector66ServiceLauncher.exe.config*.

Do konfiguračních souborů *Sector66ServerLauncher.exe.config* a *Sector66ServiceLauncher.exe.config*, které mají stejný formát, není doporučeno zasahovat. Pouze v případě nutnosti změny portu, na kterém server poběží, je možné změnit v sekci *baseAddress* koncovou adresu. Aby byla zachována funkčnost serveru, formát musí zůstat stejný, tedy http://adresa serveru:port/Sector66Server. V jiném případě nebude možné se k serveru připojit. Stejně tak jiné změny v konfiguračním souboru mohou vést ke ztrátě funkčnosti serveru hry.

6 Závěr

V této kapitole je možné najít zhodnocení práce a vymezení vůči bodům zadání z kapitoly 1.3.

6.1 Vymezení vůči cílům

Cílem práce bylo vytvořit jednoduše zásuvnými modulu rozšiřitelnou implementaci hry Sector 66, která vychází z hry Quoridor od firmy Gigamic. Pro hru vytvořit server umožňující hraní se vzdálenými protihráči, systém umělé inteligence a editor šablon hry.

Všechny tyto cíle byly splněny až na systém umělé inteligence, který byl vyřešen jen částečně. Důvody jsou popsány v kapitole 2.7 a shrnuty níže.

Zhodnocení dílčích cílů práce:

1. **Vybrat vhodný grafický framework pro účely tohoto projektu**

   Byl vybrán framework XNA 4.0 firmy Microsoft, který pro účely práce splnil všechna očekávání a ukázal se být vhodnou volbou.

2. **Vytvořit aplikaci reprezentující hru s aktivním vynucováním pravidel a zvýrazňováním možných tahů**

   Aplikace aktivně vynucuje pravidla tím, že nabízí možné tahy a žádné jiné vstupy uživatele v průběhu tahu neakceptuje. Možné tahy jsou zvýrazňovány.

3. **Vytvořit mechanismus lokalizace a utility umožňující tvorbu překladů**

   Sector 66 a jeho editor šablon je lokalizován pomocí dodaného XML souboru, což umožňuje volbu jazyka v menu aplikace i jednoduché vytváření překladů.

4. **Realizovat mechanismus pro načítání zásuvných modulů**

   Zásuvné moduly jsou dynamicky načítány z DLL knihoven uložených v určených složkách do vedlejší aplikační domény, což zajišťuje bezpečnost a jednoduché připojování. Pro vytváření zásuvných modulů je v projektu k dispozici projekt *Sector66PluginBase*.

5. **Vytvořit server hry a navrhnout síťovou komunikaci**

   Server hry byl vytvořen včetně aplikací, které ho jsou schopny hostovat jako službu operačního systému Windows a také jako aplikaci. Uchovává v sobě informace o běžících hrách. Klientská aplikace postrádá grafické rozhraní pro připojení k rozehrané hře. Aplikační rozhraní to však z principu umožňuje.

6. **Realizovat umělou inteligenci**

   Umělá inteligence pro hru Quoridor se ukázala být větším problémem, než bylo při zadávání práce předpokládáno, jak je popsáno v kapitole 2.7. Proto byla implementována v omezené míře. Umělá inteligence předpokládá pouze pohyb hráčů a stavění zábran. Funkce ohodnocující stavy jen velmi přibližně určuje hodnotu stavu hry a nebere v úvahu jeho strategickou hodnotu.

7. **Vytvořit editor šablon hry**

   Editor šablon umožňuje vytváření i editaci šablon a tím splňuje zadání. Nepodařilo se vyřešit zobrazení dotazu na uložení šablony v případě, že uživatel aplikaci zavře jiným způsobem než tlačítkem, které je proto určeno, například stiskem tlačítka pro zavření aplikace v rámečku okna.

8. **Naprogramovat ukázkové zásuvné moduly do hry**

   Součástí řešení jsou dva zásuvné moduly kouzel (*StunSpell* a *TeleportSpell*) a jeden zásuvný modul pole herního plánu (*ForbiddenField*).

6.2 Možnosti rozšíření

Hru Sector 66 by bylo možné rozšířit těmito způsoby:

- Vylepšení umělé inteligence hry.
- Přidání podpory pro připojení k rozehrané hře například po pádu spojení se serverem hry.
- Uložení a načtení rozehrané hry. Podpora v aplikačním rozhraní hry je připravena.
- Doplnění dalších zásuvných modulů kouzel a políček.
- Upravit a zkompilovat hru pro platformu XBox, Windows Phone a další.

7 Seznam použité literatury

1. **Gigamic.** Quoridor Classic. [Online] Gigamic. [Citace: 13. Květen 2012.] http://www.gigamic.com/index.php/en/quoridor-classic-c-31-p-40.html.

2. MonoGame. [Online] [Citace: 22. Červenec 2012.] http://monogame.codeplex.com/.

3. **Microsoft.** *XNA Game Studio 4.0 Refresh.* [Online] [Citace: 13. Květen 2012.] http://msdn.microsoft.com/en-us/library/bb200104.

4. Ogre. [Online] [Citace: 13. Květen 2012.] http://www.ogre3d.org/.

5. *OrbUI.* [Online] [Citace: 13. Květen 2012.] http://www.thespoofnet.com/the_spoof_net_orbui_details.html.

6. *DigitalRune.* [Online] [Citace: 13. Květen 2012.] http://www.digitalrune.com/.

7. *Squid.* [Online] [Citace: 13. Květen 2012.] http://www.ionstar.org/?page_id=4.

8. *Nuclex.* [Online] [Citace: 13. Květen 2012.] http://nuclexframework.codeplex.com/.

9. *Simple Gui.* [Online] [Citace: 13. Květen 2012.] http://simplegui.codeplex.com/.

10. **Shane, Tom.** *Neoforce.* [Online] 18. Září 2010. [Citace: 13. Květen 2012.] http://neoforce.codeplex.com/.

11. **Martens, P. J. C.** *A Quoridor-playing Agent.* [Online] 21. Červen 2006. [Citace: 13. Květen 2012.] http://www.unimaas.nl/games/files/bsc/Mertens_BSc-paper.pdf.

12. **Gigamic.** Pravidla hry Quoridor. 2007.

8 Přílohy

Všechny přílohy jsou uloženy ve složce *Přílohy* na CD.

Příloha 1

Příloha 1 obsahuje pravidla hry Sector 66 a je uložena v souboru *Pravidla.pdf*.

Příloha 2

Příloha 2 je ukázkou souboru s překladem aplikace. Tento soubor je pojmenován *Czech.xml* a obsahuje aktuální českou lokalizaci hry.

Příloha 3

Přílohu 3 tvoří programátorská dokumentace, obsahující převážně popis tříd a jejich metod, je k dispozici v anglickém jazyce ve složce *Documentation* a spouští se otevřením souboru *index.html*. K plné funkčnosti je nutné mít v prohlížeči zapnutou podporu javascriptu.