

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Malý

Osadníci z Katanu

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Filip Zavoral, Ph.D.

Studijní program: Informatika, Programování

2007

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů.
Souhlasím se zapůjčováním, práce a jejím zveřejňováním.

Jakub Malý

V Jičíně dne: 31. 7. 2007

Obsah

1	Úvod	7
2	Analýza logiky hry	8
2.1	Součásti hry z pohledu sdílení	8
2.2	Tahy a střídání hráčů	9
2.3	Veřejné/soukromé informace hráčů	10
2.4	Podíl náhody ve hře	11
3	Základní struktura programu	12
3.1	Platforma, operační systém	12
3.2	Výhody a nevýhody .NET Frameworku	12
3.2.1	Implementace hráčské části pomocí WinForms	12
3.2.2	Práce se sítí	12
3.2.3	Nevýhody .NET Frameworku	13
3.3	Internetová varianta	14
3.3.1	Dynamicky generované stránky	14
3.3.2	Applety	14
3.3.3	Standalone hráčská část aplikace, internetový server	15
3.4	Architektura klient/server	15
3.4.1	Úkoly serveru z pohledu herní logiky	16
3.4.2	Úkoly klienta z pohledu herní logiky	16
3.4.3	Implementace serveru z pohledu herní logiky	16
3.4.4	Implementace klienta z pohledu herní logiky	16
4	Rozdělení do modulů	17
4.1	Přehled modulů	18
4.1.1	Modul SupportingClasses	18
4.1.2	Modul ClientServerInterfaces	18
4.1.3	Modul NetClasses	18
4.1.4	Modul Client	19
5	Komunikace	20
5.1	Logická komunikace	20
5.1.1	Komunikace od klienta k serveru	20
5.1.2	Komunikace serveru s klienty	21

5.1.3	Přímá komunikace mezi klienty.....	22
5.2	Vrstvy komunikace	23
5.2.1	ClientConnector a ServerConnector.....	24
5.2.2	LocalPlayer a OneGameServer	25
5.2.3	TradingPlayer a Trader	26
5.3	Zpracování příchozí zprávy.....	27
5.3.1	Přenos zprávy	27
5.3.2	Zpracování zprávy.....	28
6	Hlavní třídy herní logiky	31
6.1	Reprezentace hracího plánu	31
6.1.1	Struktura hexů.....	32
6.1.2	Reprezentace hracího plánu v programu	33
6.2	Reprezentace hráčů	35
6.3	Pomocné třídy herní logiky	36
6.4	Obchodování	36
7	Grafické rozhraní klientské části	38
7.1	Zpracování herního plánu	38
7.2	Obchodní dialog	39
8	Možná rozšíření.....	40
8.1	Desková rozšíření původních Osadníků.....	40
8.2	Počítačem hraní protihráči	40
8.3	Odlišné hrací plány	41
8.4	„Spravedlivější“ kostka	41
8.5	Změna architektury a změna komunikační metody	42
9	Závěr.....	43
10	Literatura	44
11	Dodatky	45
	Dodatek 1: Konfigurace serveru	45
	Dodatek 2: Konfigurace klienta.....	45
	Dodatek 3: Deployment.....	45
	Dodatek 4: Konfigurace hracího plánu	46

Princip konfigurace plánu.....	46
Popis dokumentu pro generování plánu	46

Název práce: Osadníci z Katanu

Autor: Jakub Malý

Katedra (ústav): Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Filip Zavoral, Ph.D.

e-mail vedoucího: Filip.Zavoral@mff.cuni.cz

Abstrakt: Cílem předložené práce je převést známou deskovou hru Osadníci z Katanu do podoby počítačové hry pro dva až čtyři hráče pro hraní přes počítačovou síť. Nejprve jsou podrobně rozebrány aspekty deskové hry, které musí programátor brát v úvahu při návrhu podoby budoucího programu a rozboru několika možností, jak lze takovou to hru implementovat, zbytek práce je již věnován konkrétní implementaci jedné specifické hry – Osadníků z Katanu, popisu reprezentace komunikace, hráčů, hracího plánu a všech herních mechanismů a pravidel této hry.

Klíčová slova: Osadníci z Katanu, desková hra, počítačová hra, multiplayer

Title: Settlers of Catan

Author: Jakub Malý

Department:

Supervisor: RNDr. Filip Zavoral, Ph.D.

Supervisor's e-mail address: Filip.Zavoral@mff.cuni.cz

Abstract: The aim of the present work is to convert a well-known board game Settlers of Catan into a computer game for two to four players playing over a computer network. Common aspects and attributes of board games that must be taken into consideration during the design of the program including several possibilities of implementation of such a game are covered in the first parts of this work. The rest of the work is devoted to the implementation of one particular game – Settlers of Catan, description of the communication, representation of players, game plan and all the game mechanisms and rules of the game.

Keywords: Settlers of Catan, board game, PC game, multiplayer

1 ÚVOD

Desková hra Osadníci z Katanu (původně Die Siedler von Catan), jejímž autorem je Klaus Teuber (návrhář deskových her z povolání), již dnes patří mezi skutečnou klasiku. Bývá také označována za průlomovou hru, která odstartovala „boom“ moderních deskových her (nebo k němu alespoň přispěla). Poprvé byla vydána v Německu roku 1995 a od té doby byla přeložena do mnoha jazyků, včetně češtiny, prodaly se miliony kopií a posbírala řadu ocenění.

Jedná se o tahovou strategickou/ekonomickou hru pro tři až čtyři hráče (je ale hratelná i ve dvou hráčích) s částečně variabilním herním plánem, rozvinutou komunikací mezi hráči a spíše nízkým podílem náhody (házení kostkami).

Fanoušci deskových her obvykle zaujímají i pozitivní přístup k hraní na PC a online hraní, proto jsem se rozhodl implementovat PC verzi této deskové hry.

2 ANALÝZA LOGIKY HRY

2.1 SOUČÁSTI HRY Z POHLEDU SDÍLENÍ

Asi základním znakem deskových her je to, že všichni klienti sdílí jeden společný plán - tím se velice liší od ostatních multiplayerových počítačových her. Hráči tam sice musí sdílet na nějaké úrovni společný svět, ale mohou se po něm například nezávisle pohybovat (u 3D akcí), u strategií zase hráči „existují“ v nějaké části mapy. To, že se hráči pohybují/existují nezávisle, bývá kritickou částí hry.

Naproti tomu u (typické) deskové hry mají hráči společný hrací plán, buď

- vždy stejný, (např. u her s nějakým historickým pozadím – třeba Imperial odehrávající se na mapě Evropy)
- definovaný před začátkem vlastní hry, prakticky vždy stejný, liší se jen parametry
- definovaný/odkrývaný během hry případně i během hry modifikovaný (např. hra Tikal)

Ve všech těchto případech ale vždy *všichni hráči vidí to samé*.

Osadníci z Katanu patří do druhé skupiny her – jejich hrací plán je tvořen políčky tvaru pravidelného šestiúhelníku sestavenými do většího šestiúhelníku (se stranou délky tři + jeden pás políček moře kolem celého ostrova). Tento základní tvar je vždy stejný¹. Jednotlivá políčka označují různé krajiny ostrova (pět krajín dávajících suroviny – hory, lesy, louky, pole, pláně; poušť a moře (s případným přístavem)). Počet zástupců každé krajiny je také předem dán a pro každou hru stejný. Jednotlivé hry se liší pouze rozmístěním krajín na ostrově a čísla na jednotlivých krajínách – tyto čísla určují, jak často která krajina dává hráči suroviny. Množina těchto čísel je také vždy stejná (čísla od tří do jedenácti vždy po dvou, jednou čísla jedna a dvanáct) a různé je tak jen jejich rozmístění po jednotlivých krajínách.

Hrací plán je od začátku hry úplně odkrytý a během hry se nijak nemění.

⇒ Program musí před zahájením hry nejprve vygenerovat hrací plán – určit rozmístění jednotlivých políček s krajínami, políček s přístavy a čísel na jednotlivých krajínách. Tento vygenerovaný plán potom musí redistribuovat mezi všechny hráče. Po zbytek hry už hráčům tento plán zůstává.

V průběhu hry jednotliví hráči umísťují na plán hrací kameny označující jejich vesnice, města a cesty. Potéco se hráč rozhodne umístit nějaký kámen, dozvídají se o tom vždy všichni hráči.

Kromě hracího plánu mají hráči společný také lízací a odkládací balíček akčních karet.

¹ Toto platí pro základní verzi hry popsanou v pravidlech. Existují různá rozšíření původních Osadníků, která mj. upravují i základní hrací plán. O možnostech rozšíření programu o různé hrací plány se zmíním v části 8 - Možná rozšíření

2.2 TAHY A STŘÍDÁNÍ HRÁČŮ

Osadníci z Katanu jsou hra, ve které se hráči pravidelně střídají v tazích. Pouze hráč, který je na tahu, může provádět akce (až na několik výjimek). Hráč se sám rozhodne, kdy svůj tah skončí a hraje hráč po něm. Akce, které může hráč provádět, jsou tyto:

- na začátku tahu hráč hodí kostkami, podle hodu se určí, která pole vynáší suroviny
- stavění vesnic/měst/silnic
- nakupování *akčních karet*
- používání již koupených *akčních karet*
- obchodování

⇒ Program musí zajistit střídání hráčů na tahu a umožnit hráčům ukončit svůj tah – předat tah dál.

Ve hře nastává ale také několik situací, které vyžadují akce i od těch hráčů, kteří nejsou na tahu. Jsou to:

- odevzdávání karet *zloději* – pokud některý z hráčů hodí součet sedm, musí všichni hráči, kteří mají víc jak 7 karet surovin, polovinu z těchto karet odevzdat zpět do banky. Které konkrétní suroviny to budou, vybírá každý hráč podle sebe.
- obchodování mezi hráči – obchodovat (měnit karty surovin) může hráč, který je právě na tahu, s libovolným jiným hráčem.

⇒ Program musí umožňovat zasáhnout do hry i těm hráčům, kteří nejsou aktuálně na tahu. Zatímco u první situace (*zloděj*) je vše jednoduché (jeden dotaz – které karty hráč chce odevzdat, jedna odpověď – množina vybraných karet), otázka obchodování je daleko komplexnější. Při hraní skutečných deskových Osadníků je to totiž možná nejdůležitější a také nejzábnější část hry.

- Program musí zajistit přehledné a intuitivní obchodování se surovinami. Suroviny se mohou měnit v libovolném množství (tedy ne nutně karta za kartu, ale několik karet za několik karet), vždy mezi dvěma hráči, z nichž jeden musí být právě na tahu.
- Mírným zjednodušením je, že obchodovat se dá skutečně výhradně se surovinovými kartami.
- Vhodným doplněním bude také kanál pro textovou komunikaci mezi hráči (jakýsi chat).

2.3 VEŘEJNÉ/SOUKROMÉ INFORMACE HRÁČŮ

Hra Osadníci z Katanu je z větší části otevřená, rozhodně nepatří mezi hry, které jsou založeny na tom, že hráči před sebou některé věci skrývají.

Během hry hráči umísťují na hrací plán hrací kameny vesnic, měst a silnic – otevřeně.

⇒ Hrací kameny se umísťují na plán během hry a zůstávají na něm stejně jako v deskové hře, stávají se součástí plánu.

Během hry hráči získávají surovinové karty – buď je těží z políček na plánu, získávají je obchodem nebo použitím *akčních karet* nebo při zahrání *zloděje*. Pohyby surovinových karet jsou vždy otevřené s jedinou výjimkou – když hráč umístí *zloděje* na pole vedle protihráčovy vesnice nebo města. Potom si hráč deskových Osadníků vybírá jednu surovinovou kartu z ruky tohoto protihráče (protihráč mu ale karty neukazuje). Tuto kartu si hráč nechá a ostatním spoluhráčům neprozrazuje, o jakou surovinu se jednalo.

To, jaké karty hráč v dané chvíli vlastní, ale otevřené není. Je na ostatních hráčích, zda si budou pamatovat, jaké karty kdo během hry dostal a které z nich už použil.

⇒ Program musí informovat o pohybech surovinových karet. Informace o tom, kolik má ale který hráč karet a jaké karty to jsou, zobrazena není

Hráči během hry nakupují takzvané *akční karty*. Informace, že hráč akční kartu koupil, veřejná je. To, o jakou konkrétní kartu se ale jedná, zůstává skryto do té doby, než hráč akční kartu zahraje.

⇒ Program musí respektovat tato pravidla.

Významnou součástí hry je komunikace mezi hráči. Tato komunikace je vždy otevřená (to plyne z „deskové“ podstaty hry), společná – žádná forma „šeptání“ ve hře není. Jinak není komunikace nijak omezená. Běžnou náhradou mluvené komunikace je v multiplayerových počítačových hrách zasílání textových zpráv – chat. Rozvinutější možností je digitalizace hlasu popř. obrazu.

⇒ Program umožní textovou komunikaci. Komunikace nemá přímý vliv na vlastní hru, takže pokud hráči budou chtít přidat také hlasovou (video) komunikaci, mohou k tomu použít nějaký specializovaný externí program.

2.4 PODÍL NÁHODY VE HŘE

Osadníci z Katanu patří mezi hry s nízkým podílem náhody, přesto ale ve hře několik náhodných prvků figuruje:

- Rozmístění krajin a přístavů na herním plánu a čísel na krajinách je částečně náhodné – mechanismus tvorby plánu bude detailně popsán v Dodatku 4: Konfigurace hracího plánu.
 - Každý hráč začíná svůj tah hodem dvěma kostkami, součet čísel na kostkách pak určí krajinu, které v daném kole vynáší suroviny.
 - Když hráč koupí akční kartu, bere si svrchní kartu z lízacího balíčku akčních karet, který se zamíchá na začátku hry; pokud akční karty dojdou, znovu se zamíchají již zahrané akční karty a tak se vytvoří nový lízací balíček.
 - Pokud hráč přemístí *zloděje* na políčko sousedící s vesnicí/městem nějakého protihráče, tahá si jednu surovinovou kartu z ruky spoluhráče
- ⇒ Všechny tyto náhodné prvky musí program dobře odsimulovat – a všechny stačí zobecnit na výběr náhodného prvku z množiny.
- ⇒ Při případné implementaci logování herních událostí je potřeba pamatovat i na správné zalogování náhodných prvků a při obnově z logu negenerovat náhodné prvky znovu, ale použít původní vygenerované hodnoty.

3 ZÁKLADNÍ STRUKTURA PROGRAMU

3.1 PLATFORMA, OPERAČNÍ SYSTÉM

- Platforma: Microsoft .NET Framework 2.0
- Jazyk: C#
- Operační systém: operační systémy podporující Microsoft .NET Framework 2.0, oficiálně:
 - Windows XP
 - Windows 2000
 - Windows Vista

3.2 VÝHODY A NEVÝHODY .NET FRAMEWORKU

Platformu .NET 2.0 jsem vybral proto, že poskytuje obecnou a dobře použitelnou abstrakci především nad grafickým rozhraním a prací se sítí a také opravdu širokou paletu tříd, ze které lze snadno vybrat základní stavební prvky budoucího programu bez nutnosti hledání dalších knihoven (a případných problémů s nekompatibilitou).

3.2.1 IMPLEMENTACE HRÁČSKÉ ČÁSTI POMOCÍ WINFORMS

Prostředí WinForms umožňuje pracovat s grafickým uživatelským rozhraním pomocí hierarchického objektového systému ovládacích prvků. Kromě základní nabídky ovládacích prvků běžných v operačním systému lze do palety ovládacích prvků přidávat také vlastní ovládací prvky (CustomControls) a již existující prvky shlukovat do dalších-uživatelských ovládacích prvků (UserControls).

Díky možnému rozšiřování nabídky ovládacích prvků lze snadno využít principu zapouzdření, kdy jednotlivé uživatelské ovládací prvky zajišťují zpracování určitého aspektu vlastní hry a navenek zveřejňují jen to rozhraní, které je potřeba k interakci s ostatními prvky. Takto jsou například implementovány jednotlivé prvky hracího plánu – každý druh prvku má svůj odpovídající ovládací prvek, který ho v programu reprezentuje. Jednotlivé prvky jsou pak seskupovány do větších celků mechanismem UserControls.

3.2.2 PRÁCE SE SÍTÍ

.NET Framework obsahuje několik technologií pro práci se sítí, např. Remoting, TCP a HTTP kanály, WebServices nebo Sockety. Kolem každé z nich existuje balík příslušných tříd, které umožňují základní práci s danou technologií.

Z těchto několika možností práce se sítí jsem vybral .NET Remoting, framework pro použití vzdálených objektů. Při použití Remotingu lze totiž přímo propojit objekty na serveru i na klientovi tak, že mohou navzájem volat své metody a přitom vše probíhá transparentně (za použití proxy). Přímou práci se vzdáleným objektem nijak neliší od práce s běžnými objekty.

Pracovat přímo na úrovni TCP by znamenalo připravit se o transparentnost, objektovou orientaci a typovou bezpečnost, kterou poskytuje .NET Remoting. Komunikace ve skutečnosti v programu bude probíhat po TCP kanálu, ale bude abstrahovaná do mechanismu Remotingu.

Proti použití WebServices mluví především nutnost pracovat pod webovým serverem² (v případě .NETu tedy IIS), tato nutnost u .NET Remotingu odpadá (i když webový server lze také použít – tato metoda je popsána v [2] v kapitole Using IIS as an Activation Agent – není to nutné a je možné ho nahradit vlastním aplikačním serverem). WebServices jsou ale i přes tento problém velice zajímavou alternativou, především v tom momentě, kdy by měla aplikace běžet v prostředí Internetu. O tom více v části 3.3.

3.2.3 NEVÝHODY .NET FRAMEWORKU

Velkou nevýhodou .NET Frameworku je přeci jen jeho přílišná svázanost s technologiemi a především operačními systémy firmy Microsoft. Ačkoliv již probíhá proces portování .NET Frameworku na operační systém Linux, Microsoftem zatím tento OS nijak podporován není. Aplikace verze 2.0 a vyšší zatím nelze na Linuxu ani jiných operačních systémech mimo Windows spolehlivě provozovat.

Ani na operačních systémech Windows není provoz .NET aplikací úplně bezproblémový. V současné době ještě .NET Framework není ve verzi 2.0 nainstalován na mnoha běžných uživatelských PC (i když počet instalací roste a .NET se pravděpodobně již brzy stane standardní součástí systému). Z tohoto důvodu je k programu vytvořen instalační balíček, který ověří přítomnost požadované verze frameworku na uživatelské počítači a pokud ji nenalezne, doinstaluje ji³.

Použití .NET Frameworku také znamená nezanedbatelné zpomalení aplikace – just-in-time překládaný kód⁴ je pomalejší, než nativně přeložený program. Navíc při prvním spuštění dochází k spuštění .NET Runtime a to prodlužuje dobu nabíhání aplikace. Navíc k tomu i nic nedělající aplikace používající WinForms zabírá 10 MB v operační paměti.

GDI+ – část .NET Frameworku obsahující třídy a objekty pro kreslení – obsahuje sice všechny základní kreslicí funkce (známé z Win32 GDI), není ale nijak výkonná.

Hlavní nevýhody .NET Frameworku jsou tedy režie při spuštění a provádění kódu a nutnost dodávat .NET Runtime spolu s aplikací pro ty uživatele, kteří si .NET Runtime zatím ještě neinstalovali.

² jedním z cílů programu je, aby si herní server mohl uživatel pustit na svém počítači. Alternativou by byl centrální server (např. v Internetu), který by hostoval hry, a jednotliví hráči by se připojovali k tomuto serveru. Tato alternativa je díky modularitě programu možná a bude rozpracována v části 8 - Možná rozšíření.

³ Více podrobností v Dodatku 3 - Deployment

⁴ kód .NET aplikací je při kompilaci přeložen pouze do mezikódu, dodatečný překlad zajišťuje až .NET Runtime při spuštění aplikace na uživatelském systému

3.3 INTERNETOVÁ VARIANTA

To, že by mohla aplikace běžet v prostředí Internetu, byla možnost, kterou jsem dlouho zvažoval. Hraní po Internetu je stále populárnější a Osadníci z Katanu by jistě našli své příznivce (co se týká českého prostředí, nevím o žádném serveru, který by tuto hru provozoval). Hraní po Internetu by mohlo mít několik následujících podob.

3.3.1 DYNAMICKY GENEROVANÉ STRÁNKY

Jednou z možností by bylo implementovat jak herní server, tak velkou část hráčské aplikace na serveru a s klientem komunikovat pouze pomocí protokolu HTTP. Klient by pak hrál přes webový prohlížeč. Na straně serveru by byla použita některá z technologií pro dynamické generování stránek (např. ASP.NET nebo PHP), na straně klienta by přicházel v úvahu pouze prohlížeč (tedy DHTML, CSS, JavaScript).

Takto bývá implementováno mnoho her na běžících na webu, obvykle to ale bývají hry typu piškvorky nebo dáma – tedy hry mnohem jednodušší, než Osadníci z Katanu.

Problémem by především byla komunikace iniciovaná serverem. Protokol HTTP je stavěn pro komunikaci dotaz/odpověď, přitom ve hře musí server nějak šířit informace na popud jednoho hráče mezi ostatní hráče (pokud možno s co nejmenším zpožděním, zvláště u obchodování by dlouhé prodlevy silně narušovaly zážitek ze hry). Toto by šlo obejít použitím nějakých AJAXových⁵ technik.

Dalším problémem by bylo zobrazování průběhu hry klientovi – herní plán nelze nijak snadno rozložit na obdélníkové tvary, na které jsou prohlížeče zvyklé, buďto by bylo potřeba jednotlivé hexy „rozřezat“ nebo plán chápat (a zobrazovat) jako jeden celek nebo každý hex „obalit“ opsaným obdélníkem a ty pak překrývat s použitím průhlednosti.

Ačkoliv se AJAXové aplikace i frameworky začínají rozšiřovat, mají velké problémy s kompatibilitou prohlížečů a vývoj za použití těchto frameworků se nedá srovnat s komfortem jazyků jako je Java nebo C#. Složitější klientské skriptování má také větší nároky na hardware.

Pokud by hra fungovala na principech DHTML + dynamicky generovaných stránek, uživatel by sice nemusel nic instalovat (což je nesporná výhoda, protože je známo, že uživatelé mívají nechuť cokoli instalovat), k připojení k serveru, spuštění a hraní by mu stačil webový prohlížeč, ale celá implementace by byla mnohem náročnější a výsledek by pravděpodobně nedosahoval komfortu klasické aplikace.

3.3.2 APPLETY

Kompromisem mezi klasickou aplikací a webovou aplikací by bylo použití appletu, např. v jazyce Java. Applety dávají programátorovi téměř stejné prostředky, jako klasické aplikace, lze je ale vložit jako celé objekty do těla webové stránky. Uživateli by tak stačil k hraní webový prohlížeč, ale na rozdíl od varianty s DHTML by

⁵ AJAX – zkratka nejčastěji vykládaná jako *Asynchronous Javascript and XML*, v poslední době čím dál oblíbenější metoda, jak obejít omezení (a někdy i základní principy) HTTP protokolu pro tvorbu webových aplikací

kromě něj potřeboval také nainstalované běhové prostředí. A to už vyjde skoro nastejno, jako instalace .NET Frameworku.

3.3.3 STANDALONE HRÁČSKÁ ČÁST APLIKACE, INTERNETOVÝ SERVER

Nejlepší variantou pro hraní v prostředí Internetu se mi tak zdá oddělit klientskou část a implementovat ji jako samostatnou klasickou aplikaci, se kterou by bylo možné připojit se na internetový server. Internetový server by mohl obstarávat veškerou přidruženou administrativu (registrace uživatelů-hráčů, statistiky apod.), dále pak nabízet ke stažení klientskou aplikaci a přijímat připojení klientů.

Můj program sice nepoužívá internetový server, ale je psán tak, aby bylo možné na tento model přejít. Klientská část je rozdělena na část obsahující vlastní herní logiku a na část zajišťující komunikaci (a tedy závislou na způsobu připojování). Pro přechod na výše popsanou architekturu by tak stačilo napsat pouze objekty pro komunikaci splňující stejné rozhraní.

Podobným způsobem jako klientská část je rozdělena i část serverová a tak by bylo možné i v tomto modelu použít tu část serveru, která obsahuje herní logiku.

Vhodnou metodou komunikace mezi klientskou aplikací a internetovým serverem by byly webové služby.

Uživatel by tak byl nucen stáhnout si a nainstalovat na svůj počítač klientskou část (+ případně .NET Framework, pokud ho ještě nemá), s tímto klientem by se mohl připojit k internetovému serveru. Nevýhodou tohoto modelu by byla absence možnosti spustit si svůj vlastní server (proto jsem zvolil raději aplikační server místo internetového).

Rozhodnout se mezi variantou s applety a variantou popsanou v této části je asi spíše záležitostí vkusu. Obecně jsou uživatelé ochotní instalovat samostatnou aplikaci pro nějakou větší hru, naopak u menších a jednodušších her by dali přednost appletové verzi. Hra Osadníci z Katanu je podle mne někde na pomezí, obě řešení by tak asi byla přijatelná. To, jestli si musí uživatel instalovat .NET Runtime nebo běhové prostředí pro spuštění appletu, je jen drobný rozdíl.

3.4 ARCHITEKTURA KLIENT/SERVER

Z analýzy v druhé kapitole je dobře vidět, že pro program se nejlépe hodí klient/server architektura. Zatímco server bude udržovat společné prvky hry a koordinovat hráče, klientská část bude sloužit k vizualizaci hry a k zadávání herních příkazů – tahů.

Díky použití .NET Remotingu lze server implementovat jako vzdálený objekt. Všichni klienti pak získají referenci k tomuto vzdálenému objektu a voláním metod tohoto vzdáleného objektu budou provádět své herní akce.

3.4.1 ÚKOLY SERVERU Z POHLEDU HERNÍ LOGIKY

Server má následující úkoly (čistě z pohledu herní logiky)

- Vygenerování herního plánu před začátkem hry, distribuování plánu klientům.
- Házení kostek (tímto bude nahrazen nabízející se model, který by přesně kopíroval hru – totiž aby klient „hodil kostkami“ a hozená čísla odeslal na server, který je rozešle ostatním klientům). Protože házení kostek by mělo být naprosto náhodné⁶ a hráč by tak neměl mít žádnou možnost ho ovlivnit, bude výsledek stejný, jako když se házení odehraje na serveru.
- Udržování lízacího a odkládacího balíčku akčních karet.
- „Líznutí“ akční karty – v deskové hře se bere vždy svrchní karta z lízacího balíčku. V programu se bere vždy náhodná karta z množiny volných karet. Tato vybraná karta se pak přidělí hráči.
- Výběr „ukradené karty“, kterou hráč v deskové hře vybírá z ruky spoluhráče při přemístění zloděje. Tento výběr by měl být opět náhodný, proto ho může stejně dobře simulovat samotný server bez vlivu klienta (podobně jako při házení kostek).

3.4.2 ÚKOLY KLIENTA Z POHLEDU HERNÍ LOGIKY

Klient má následující úkoly (čistě z pohledu herní logiky)

- Zobrazování vygenerovaného plánu hráči.
- Zobrazování surovinových karet a akčních karet, které hráč vlastní.
- Umožnit klientovi nakupovat a umísťovat vesnice, města, silnice a akční karty, hrát akční karty.
- Umožnit obchodování.

3.4.3 IMPLEMENTACE SERVERU Z POHLEDU HERNÍ LOGIKY

Herní server je implementován jako objekt (OneGameServer), na kterém jednotliví hráči volají metody, které odpovídají logice hry (např. „koupit vesnici“, „skončit tah“ apod.). Herní server udržuje všechny informace o hře.

3.4.4 IMPLEMENTACE KLIENTA Z POHLEDU HERNÍ LOGIKY

Objekt hráče (LocalPlayer) zveřejňuje všechny metody potřebné k tomu, aby uživatel mohl provádět herní akce. Tyto metody se vhodným způsobem provází s uživatelským rozhraním klienta.

⁶ I když se míra náhody v Osadnících považuje za spíše nízkou, dávají přesto někteří hráči přednost hraní hry s ještě omezenějším vlivem náhody. Podrobnosti v části 8.4 – Možná rozšíření – „Spravedlivější“ kostka.

4 ROZDĚLENÍ DO MODULŮ

Program obsahuje pět hlavních projektů:

- SupportingClasses
- ClientServerInterfaces
- NetClasses
- Server (MultiServer)
- Client

Vzájemnou závislost hlavních modulů ilustruje diagram 1.

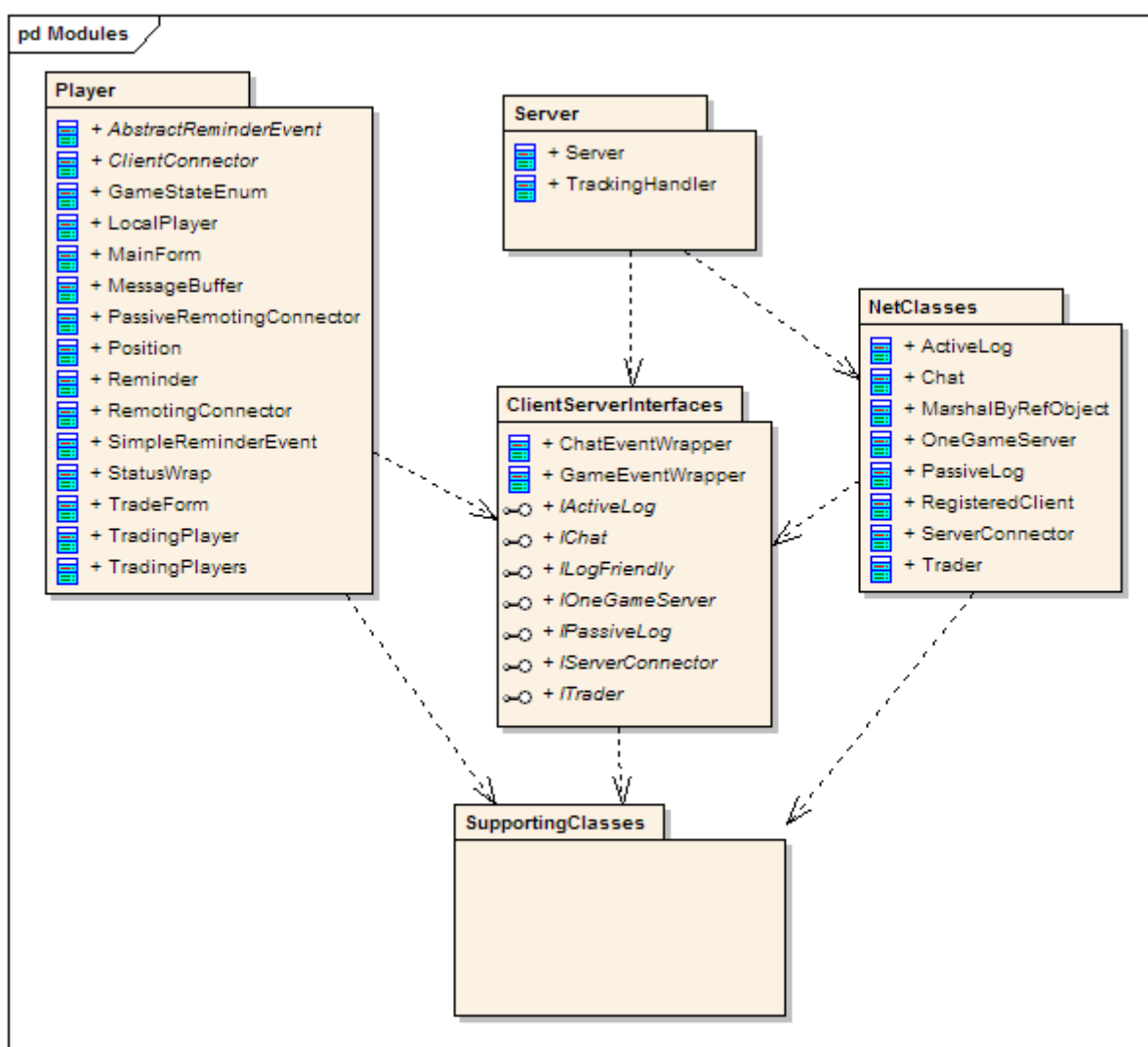


Diagram 1: Závislost hlavních modulů

4.1 PŘEHLED MODULŮ

Ke spuštění serveru jsou třeba knihovny:

- SupportingClasses
- ClientServerInterfaces
- NetClasses

Ke spuštění klienta jsou třeba knihovny:

- SupportingClasses
- ClientServerInterfaces

4.1.1 MODUL SUPPORTINGCLASSES

Modul SupportingClasses obsahuje třídy použité v serverové i klientské části:

- BoardClasses – třídy v jazyce logiky hry Osadníci z Katanu, reprezentace hry v programu (reprezentace hracího pole, hráčů, karet atd.)
- SupportingClasses – pomocné třídy

4.1.2 MODUL CLIENTSERVERINTERFACES

Do modulu ClientServerInterfaces patří rozhraní pro komunikaci serveru a klienta. Především rozhraní objektů dědicích od MarshalByRef. Obsahuje také implementaci logování. Klienti pracují výhradně s rozhraními a serverové objekty tyto rozhraní implementují. Při použití Remotingu jako komunikační vrstvy klienti pracují přes rozhraní s transparentní proxy serverových objektů. Pokud by byla použita jiná technologie pro komunikační vrstvu, stačí dát klientovy objekt, který bude splňovat dané rozhraní a nějak komunikovat se serverovým protějškem (Remoting umožňuje automatické vytvoření proxy, jiná technologie by vyžadovala proxy k serverovému objektu implementovat zvlášť).

Klíčová rozhraní:

- IOneGameServer – rozhraní serveru a hry
- IServerConnector – připojování klientů

4.1.3 MODUL NETCLASSES

Objekty modulu NetClasses implementují rozhraní z ClientServerInterfaces na straně serveru (serverové objekty), reprezentace serveru a hry jako celku.

Klíčové třídy:

- OneGameServer – implementace IOneGameServer – reprezentace hry
- ServerConnector – implementace IServerConnector – spojení s klienty

4.1.4 MODUL CLIENT

Modul Client obsahuje GUI hry pro jednoho klienta, připojování klienta k serveru

Klíčové třídy:

- ClientConnector – abstraktní třída pro připojení k serveru, jeho hlavním úkolem je přijímání zpráv GameMessage (viz oddíl 5.1.2 – Komunikace serveru s klienty)
- RemotingConnector, PassiveRemotingConnector – potomci ClientConnector, připojují k serveru, který komunikuje s klienty pomocí Remotingu
- LocalPlayer – reprezentace klienta jako hráče
- TradingPlayers, TradingPlayer – ta část implementace hráče, která se týká obchodování
- MainForm, TradeForm – GUI – formuláře hry

5 KOMUNIKACE

Účastníky komunikace jsou jednotliví klienti (hráči) a server. Komunikace se dá rozdělit na několik částí.

5.1 LOGICKÁ KOMUNIKACE

Základní schéma herní komunikace je následující

1. Hráč chce provést herní akci (např. koupit cestu), musí proto o akci dát vědět sereru – zavolá metodu serverového objektu (část 5.1.1)
2. Serverový objekt akci prověří, zpracuje a o výsledku informuje klienty (část 5.1.2)

5.1.1 KOMUNIKACE OD KLIENTA K SERVERU

Pokud hráč chce provádět nějakou herní akci, musí se zavolat metoda na serverovém objektu (OneGameServer). Proto musí existovat odpovídající komunikační kanál.

Klienti (objekt LocalPlayer) pracují s rozhraním IOneGameServer, toto rozhraní předepisuje všechny metody, které klienti potřebují pro vlastní hraní. Protože jsem implementoval OneGameServer jako potomka MarshalByRefObject, mohou klienti získat transparentní proxy tohoto objektu, přetypovat ji na IOneGameServer a přes tuto proxy vzdáleně volat metody.

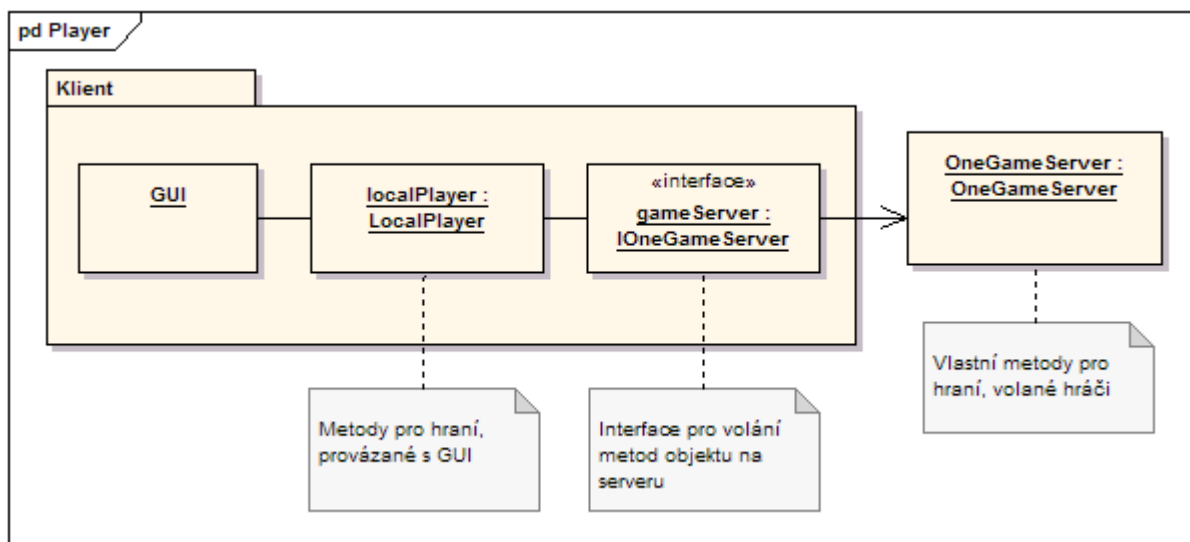


Diagram 2: Schéma přístupu k serverovým metodám

Komunikace ze strany klienta je transparentní, klientská část obstarávající herní logiku (objekt LocalPlayer) jen čeká na to, až získá objekt implementující rozhraní IOneGameServer, jaký objekt to bude a jak bude ve skutečnosti komunikovat se serverem tato část klienta neřeší. Díky této modularitě je možné přejít na jiný systém komunikace – např. na WebServices.

Příklad jednoduché herní akce:

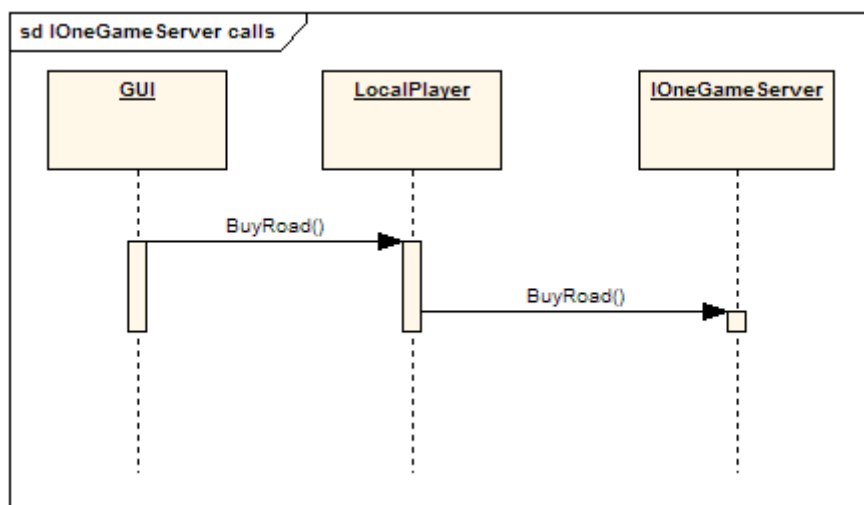


Diagram 3: Příklad průběh herní akce (nákup cesty)

1. Hráč chce koupit novou cestu, kliká na odpovídající tlačítko na formuláři
2. Obsluha události tlačítka zavolá metodu BuyRoad objektu LocalPlayer
3. V té je pak volána metoda BuyRoad rozhraní IOneGameServer. Klient tak pracuje pouze s rozhraním (proxy) IOneGameServer a o princip přenosu volání na server se nestará.

5.1.2 KOMUNIKACE SERVERU S KLIENTY

Nejjednodušší způsob předání informací ze serveru na klienty je vrácení návratové hodnoty metody vzdáleně volané serverem. Tento jednoduchý způsob se hodí ale pouze k potvrzování mezikroků, protože nevyhovuje výše zmíněnému „globálnímu“ principu komunikace – tedy pokud se něco stane, musí se o tom dozvědět všichni – přitom návratovou hodnotu metody se dozví pouze volající klient.

Další možností by bylo vytvořit stejný systém komunikace, jako u komunikace opačným směrem – totiž nechat server, aby si vytvořil transparentní proxy ke klientským objektům LocalPlayer a při herní události by pak server vždy zavolal stejnou metodu na všech těchto proxy. Místo toho jsem ale zvolil jiný způsob – server se neodkazuje přímo na objekt LocalPlayer, ale místo toho sám posílá zprávy těm klientům, kteří se k serveru připojili (a přihlásili k odebrání). Zprávy, které server rozesílá, jsou zapouzdřeny v objektu GameMessage.



Diagram 4: Struktura objektu GameMessage

Atributy objektu GameMessage:

- player: hráč, kterého se zpráva týká (ale není to žádný „adresát“ zprávy, zprávu dostanou vždy všichni)
- type: typ zprávy – označuje, ke které herní události se zpráva váže (např. nákup akční karty)
- description – textový popis zprávy
- messageArgs – parametry zprávy
- messageNumber – číslo zprávy

Atribut messageNumber je pořadí dané zprávy od zahájení hry. Klienti díky němu mohou kontrolovat, zda jim zprávy přicházejí ve správném pořadí. Pokud ne, mohou si vyžádat od serveru chybějící zprávu (z historie).

Hlavní výhodou modelu používajícího zprávy je lepší modularita programu – server nemusí s klienty komunikovat přímo, pouze během hry odesílá zprávy všem přihlášeným klientům a ti už si je sami zpracovávají. Díky tomu, že není přímá vazba mezi serverem a klientem (taková, kde by server volal metody klienta), lze snadno zvolit jiný způsob komunikace mezi serverem a klienty – např. kdyby server měl fungovat jako webová služba, mohou se klienti v pravidelných intervalech dotazovat serveru, zda pro ně nemá nové zprávy (novější, než poslední zpráva, kterou dostali – pozná se podle parametru messageNumber) a případné nové zprávy zpracovat (tímto se de facto úplně odstraní komunikace iniciovaná serverem).

Další výhodou tohoto způsobu je logování zpráv hry – pokud se k serveru připojí log, který bude zapisovat každou odeslanou zprávu GameMessage, může se z tohoto logu rekonstruovat průběh hry (protože klienti sami vždy svou akci jen zavolají nějakou metodu serveru, případnou reakci na toto volání se dozví až z příchozí zprávy GameMessage).

Nevýhodou modelu zpráv je nutnost analyzovat zvláště každou příchozí zprávu a až podle jejího messageType teprve určit, jakou akci má zpráva vyvolat.

5.1.3 PŘÍMÁ KOMUNIKACE MEZI KLIENTY

Jak vyplývá z analýzy, téměř veškeré události ve hře jsou otevřené (dozívají se o nich všichni hráči) – tahy hráče musí být oznámeny všem ostatním hráčům. I pokud nastane událost, která není úplně odkrytá (např. hráč koupí akční kartu), musí být o této události alespoň částečně informováni i ostatní hráči (dozví se, že hráč koupil kartu, ale nedozví se jakou).

Protože je komunikace takto „globální“, je zbytečné vytvářet komunikační cesty mezi jednotlivými hráči (dokonce ani pro textové zprávy, ty jsou také vždy směřované všem hráčům, „šeptání“ povoleno není). Proto v programu není žádná možnost pro klienta komunikovat s ostatními klienty – komunikovat lze vždy pouze centrálně, přes server (voláním jeho metod).

5.2 VRSTVY KOMUNIKACE

Komunikace od klienta k serveru probíhá ve třech hlavních vrstvách, vždy platí stejný princip – klientský objekt komunikuje s proxy přes rozhraní, které implementuje serverový objekt (tuto proxy vytváří .NET Framework automaticky).

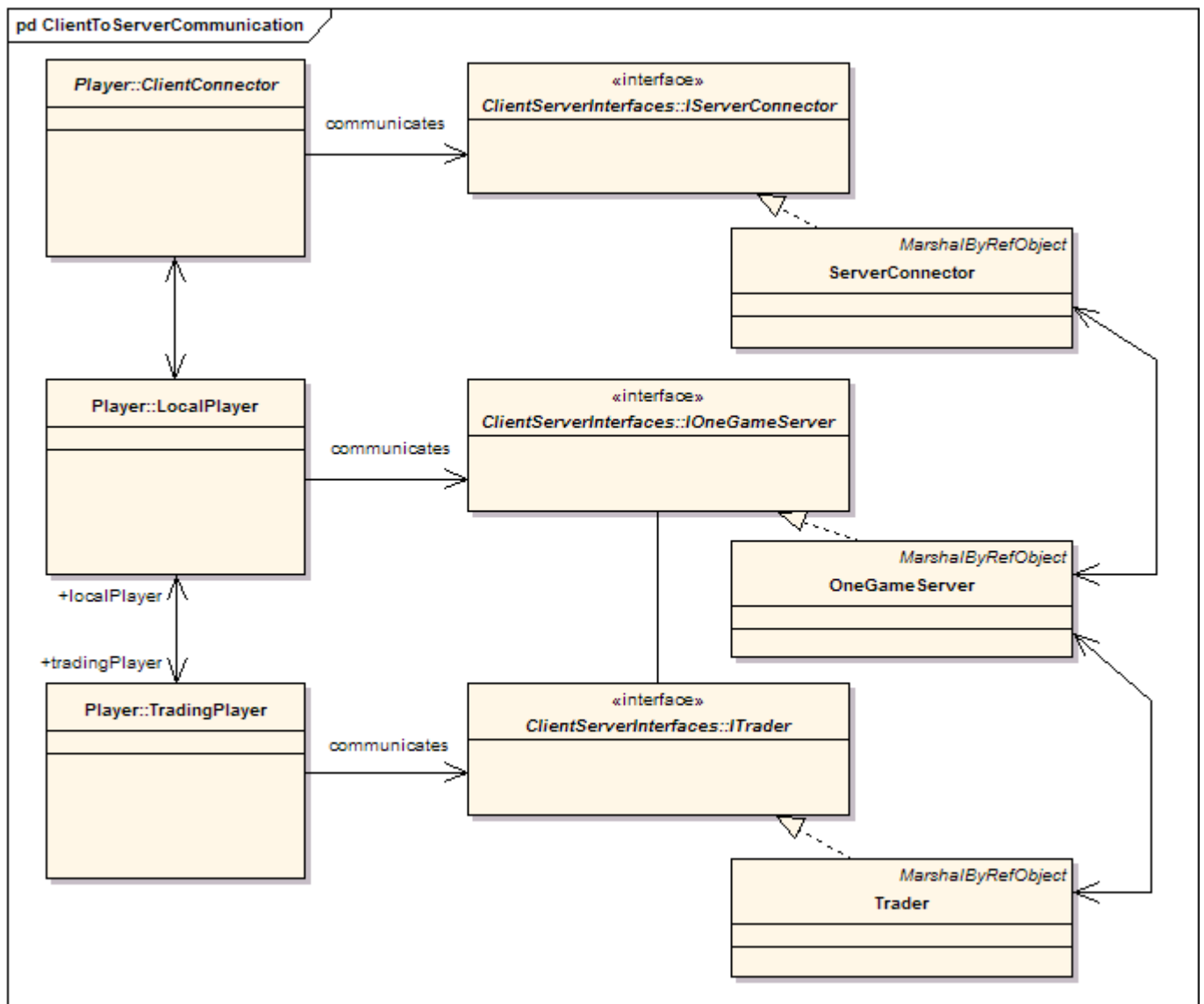


Diagram 5: Vrstvy komunikace

5.2.1 CLIENTCONNECTOR A SERVERCONNECTOR

Objekty ClientConnector a ServerConnector zajišťují přímo propojení serverové a klientské části. ServerConnector spravuje hráče každé hry, umožňuje jejich připojování, odpojování (metody ConnectPlayer resp. DisconnectPlayer). Dále obstarává registraci klientů pro odebrání zpráv GameMessage a rozesílání zpráv registrovaným klientům⁷ (metody RegisterForEvents resp. UnregisterForEvents).

Vyšší vrstva může pomocí metody GetServer získat proxy objektu OneGameServer, se kterým pak pracují již připojení klienti.

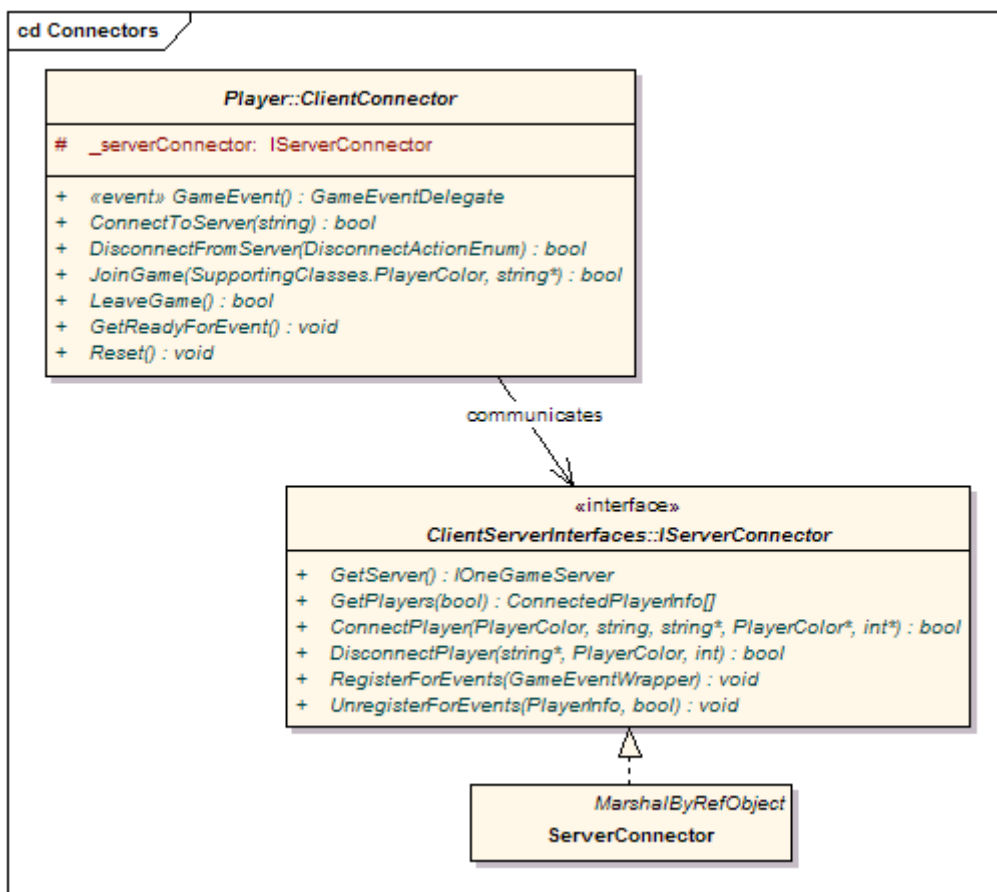


Diagram 6: Spodní vrstva komunikace, správa klientů

Objekt ServerConnector je zaregistrován (v případě standalone serveru, který hostuje pouze jednu hru) jako WellKnownServiceType a je hlavním přístupovým bodem k serveru. Klienti komunikují se serverem přes proxy objektu ServerConnector.

Abstraktní třída ClientConnector je protějšek ServerConnectoru u klienta. Umožňuje klientu připojit se k ServerConnectoru (voláním Activator.GetObject získá klient proxy k ServerConnectoru).

Vyšším vrstvám předává proxy objektu OneGameServer a ty potom s touto proxy pracují. Jeho událost GameEvent potom signalizuje vyšším vrstvám novou příchozí zprávu GameMessage.

⁷ Princip rozesílání a zpracování zpráv GameMessage je popsán v části 5.3

5.2.2 LOCALPLAYER A ONEGAMESERVER

Objekty LocalPlayer a OneGameServer fungují na úrovni logiky hry,

Objekt LocalPlayer

- zveřejňuje metody pro provádění herních akcí (ty jsou navázané na uživatelské rozhraní, především na formulář MainForm). Tyto metody volají (přes proxy) metody serverového objektu OneGameServer
- jako odpověď na herní akce dostává od serveru zprávy GameMessage od nižší vrstvy (ClientConnector), tyto zprávy analyzuje a zpracovává. Zprávy GameMessage zpracovává metoda LocalPlayer.GameEventInnerReaction, která analyzuje typ zprávy a podle něj rozhoduje, jak bude zpráva dále zpracovávána.

Propojení vrstvy logické komunikace:

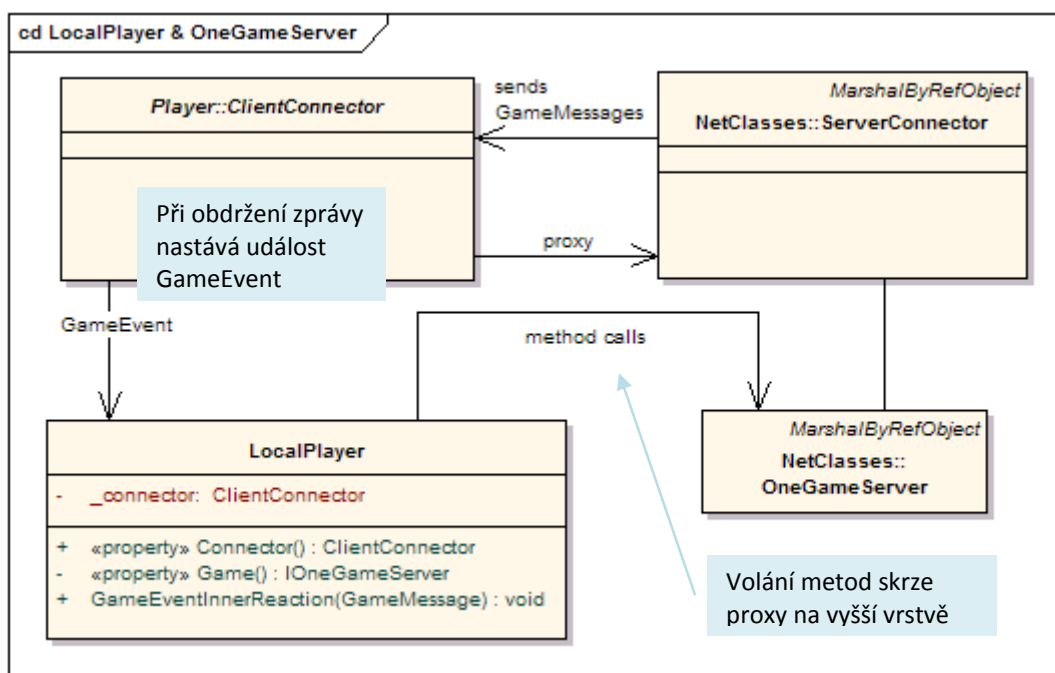


Diagram 7: Propojení vrstev komunikace

Z diagramu je patrné, jak probíhá komunikace mezi oběma vrstvami

- Zprávy GameMessage dostává ClientConnector od ServerConnectoru. Při nové příchozí zprávě nastavá na ClientConnectoru událost GameEvent, na kterou je navázána metoda GameEventInnerReaction objektu LocalPlayer, který zprávy zpracuje.
- Volání serverových metod provádí objekt LocalPlayer přes proxy k objektu OneGameServer. Tuto proxy mu poskytuje objekt ServerConnector svému protějšku ClientConnector.

5.2.3 TRADINGPLAYER A TRADER

Objekty TradingPlayer a Trader zpracovávají čistě logiku obchodování ve hře, jsou podobnými objekty objektů LocalPlayer a OneGameServer. Obchodování bylo odděleno od této dvojice objektů do samostatné dvojice čistě kvůli zjednodušení objektů LocalPlayer a OneGameServer.

Princip komunikace mezi TradingPlayer a Trader je stejný, jako mezi LocalPlayer a OneGameServer, zprávy TradeGameMessage přicházejí stejně jako zprávy GameMessage přes objekt ClientConnector a LocalPlayer je při zpracování předává objektu Trader. Volání metod vzdáleného objektu Trader probíhá opět přes proxy.

Na úrovni GUI je na této vrstvě formulář TradeForm.

Princip obchodování a úloha jednotlivých objektů v jeho implementaci z pohledu herní logiky je vysvětlen v části 6.4, grafické zpracování potom v části 7.2.

5.3 ZPRACOVÁNÍ PŘÍCHOZÍ ZPRÁVY

5.3.1 PŘENOS ZPRÁVY

Zatím jsem nic neříkal o tom, jak jsou zprávy `GameMessage` přeneseny ze serveru na klienty. Tento přenos závisí na použité technologii – na klientu stačí jen implementovat potomka abstraktní třídy `ClientConnector`, ta je ale velmi obecná a způsob propojení je čistě na potomcích.

V programu je implementováno spojení pomocí remotingu (potomek `RemotingClientConnector` třídy `ClientConnector`). Předávání funguje tak, že třída `RemotingClientConnector` vytváří speciální objekt `GameEventWrapper`, který je potomkem `MarshalByRefObject`. Tento objekt je předán serveru (v metodě `IServerConnector.RegisterForEvents(GameEventWrapper gameWrapper)`) a tak získá server proxy na tento objekt u klienta. V momentě, kdy má server rozeslat zprávu `GameMessage`, iteruje přes své zaregistrované proxy k objektům `GameEventWrapper` a volá na nich metodu `GameEventWrapper.NewGameMessageEventHandler(GameMessage gm)`, která už se vykonává u klienta).

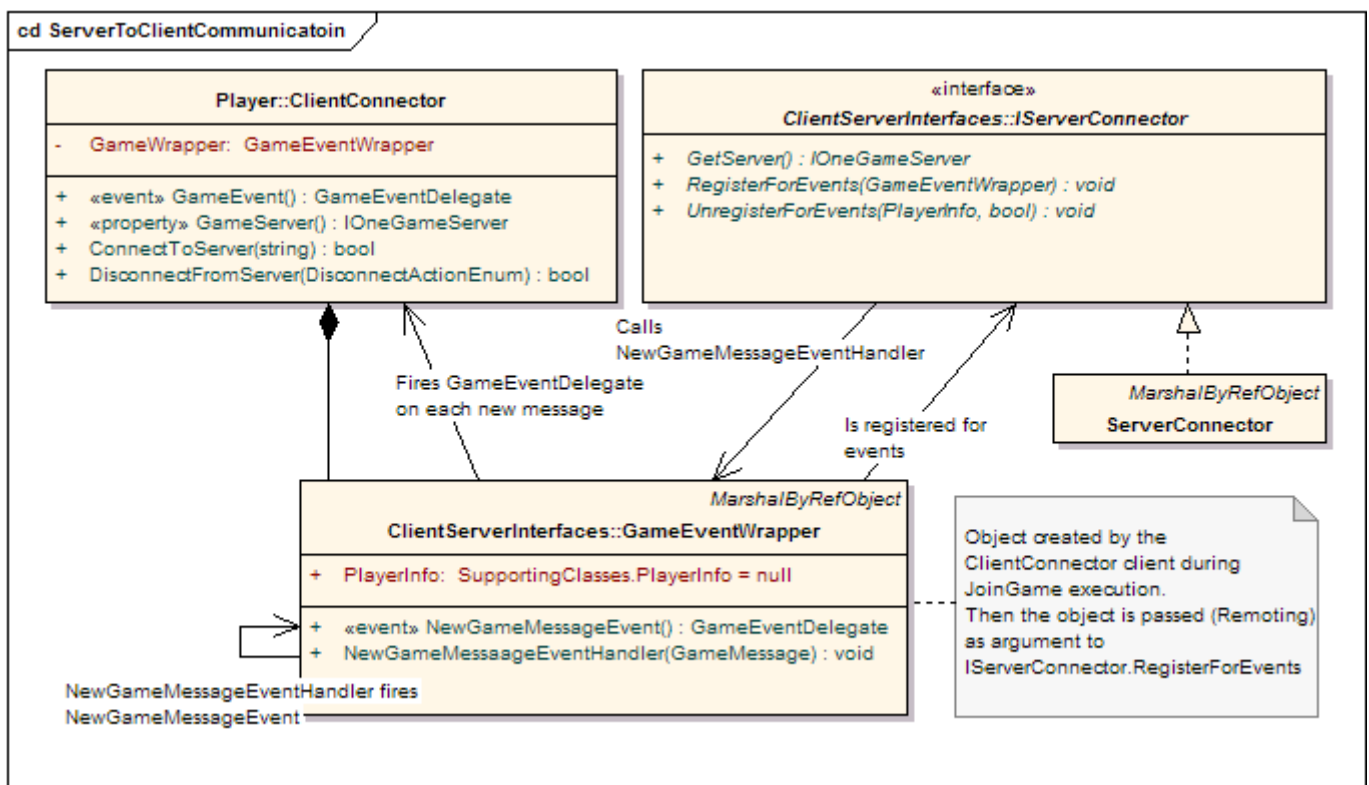


Diagram 8: Zprávy ze serveru zasílané přes `GameEventWrapper`

Jiný způsob přenosu zpráv `GameMessage` by vyžadoval jinou implementaci `ClientConnector` a `ServerConnector`.

Přímočarý způsob – navazovat metodu lokálního objektu na událost vzdáleného objektu (čistě pomocí operátoru `+=`), který lze nalézt v [2], již není doporučován pro .NET verze 2.0.

Postup registrace pro odebrání zpráv:

1. ClientConnector získá objekt s rozhráním IServerConnector.
2. ClientConnector vytvoří objekt GameEventWrapper a ten předá metodě RegisterForEvents.
3. Pro případné ukončení odebrání ClientConnector zavolá metodu UnregisterForEvents.

Předávání zprávy:

1. Serverový objekt ServerConnector iteruje přes registrované objekty GameEventWrapper.
2. Na každém objektu GameEventWrapper volá metodu NewGameMessageEventHandler, které předá danou zprávu (objekty GameEventWrapper jsou u jednotlivých klientů, ServerConnector k nim má jen reference).
3. Metoda NewGameMessageEventHandler vyvolá událost NewGameMessageEvent, na kterou je navázána (s několika mezikroky – popsáno v další části) událost GameEvent.

Událost GameEvent již slouží k odchytnutí na vyšší vrstvě – objektem LocalPlayer. Ať tedy libovolný potomek abstraktní třídy ClientConnector zajišťuje přijímání zpráv ze serveru, touto událostí posílá přijatou zprávu k dalšímu zpracování vyšším vrstvám.

5.3.2 ZPRACOVÁNÍ ZPRÁVY

S předáváním a zpracováním zpráv souvisí ještě několik dalších problémů

- Klient předpokládá, že zprávy mohou přicházet v jiném pořadí, než byly rozeslány (kontroluje se podle atributu messageNumber každé GameMessage).
- Zprávy mohou přicházet rychleji, než je klient stihne zpracovávat.
- Je potřeba oddělit zprávy týkající se obchodování (ty zpracuje objekt TradingPlayer, ostatní zpracuje LocalPlayer).
- Na zprávy reaguje GUI (formuláře), GUI musí být obnoveno po zpracování zprávy, pracovat s prvkem GUI ale může jen to vlákno, které prvek vytvořilo.
- Zprávy mohou přicházet rychle za sebou, což by mohlo být v některých případech matoucí pro uživatele (nestíhal by sledovat), proto je potřeba některé zprávy uměle „zdržovat“.

Protože klient předpokládá, že zprávy mohou přicházet ve špatném pořadí a že mohou přicházet tak rychle, že je nestihne zpracovávat, jsou všechny příchozí zprávy nejprve ukládány do bufferu.

Tento buffer (třída MessageBuffer)

- ukládá všechny příchozí zprávy
- propouští vždy jen tu zprávu, která je na řadě
- v případě, že dostane jinou zprávu, než očekává (zprávu s vyšším číslem messageNumber, než které je na řadě), požádá server o znovuzaslání chybějících zpráv

- vždy propustí jen jednu zprávu, která je na řadě, a dál čeká, dokud není zavolána metoda `GetReadyForEvent`

Buffer je privátní součástí třídy `RemotingClientConnector` (potomek abstraktní třídy `ClientConnector`) a tak další třída (`LocalPlayer`), která je na řadě ve zpracování zprávy, má již zajištěno správné pořadí zpráv. Po každém zpracování musí `LocalPlayer` zavolat `GetReadyForEvent`, aby buffer propustil další zprávu.

`LocalPlayer` zpracovává zprávu v metodě `GameEventInnerReaction`, po jejím proběhnutí ještě kontroluje, zda aktuální zpráva není typu `TradeGameMessage` (potomek `GameMessage`, zprávy týkající se obchodování). Pokud ano, předává ji k dalšímu zpracování třídě `TradingPlayer`. Ta ji zpracovává podobným způsobem, jako `LocalPlayer`. Pokud ne, předává ji k dalšímu zpracování třídě `MainForm`.

`MainForm` je hlavní formulář hry, který především zobrazuje plán hry. Zpráva `GameMessage` je předána metodě `MainForm.NewGameEvent(GameMessage gm)`, která rozhoduje, zda má být zpráva zdržena (aby před uživatelem neproběhlo příliš rychlé zpracování několika zpráv za sebou). Například zprávy `Rolled` a `RollProfit` (hod hráče na kostkách a kolik který hráč získal surovin po hodu na kostkách) jsou odesílány serverem okamžitě za sebou, proto objekt formuláře „uměle“ nastavuje jejich zdržení. Zprávám, které mají být propuštěny okamžitě, nastavuje zdržení na nulu. Zprávy pak metoda předá objektu `Reminder`, která nové zprávy zařazuje do fronty podle doby zdržení a z této fronty je také vyzvedává a předává k dalšímu zpracování.

Toto další zpracování provádí metoda `ProcessDelayedEvent`, která zaprvé vypíše zprávu hlášení do okna zpráv pro uživatele a zavolá metodu `ReflectGameEventsInUI` pomocí `BeginInvoke` tak, aby změny ovládacích prvků provádělo vždy to správné vlákno, které daný ovládací prvek vytvořilo.

Podobným způsobem, jak `MainForm`, zpracovává zprávy `TradeGameMessage` formulář `TradeForm`, jen s tím rozdílem, že žádné obchodní zprávy není potřeba uměle zdržovat, takže odpadá potřeba `Reminderu` a metody `ProcessDelayedEvent`. Jinak je zpracování analogické.

Celý proces zpracování příchozí zprávy je znázorněn na digramu 6 na následující straně.

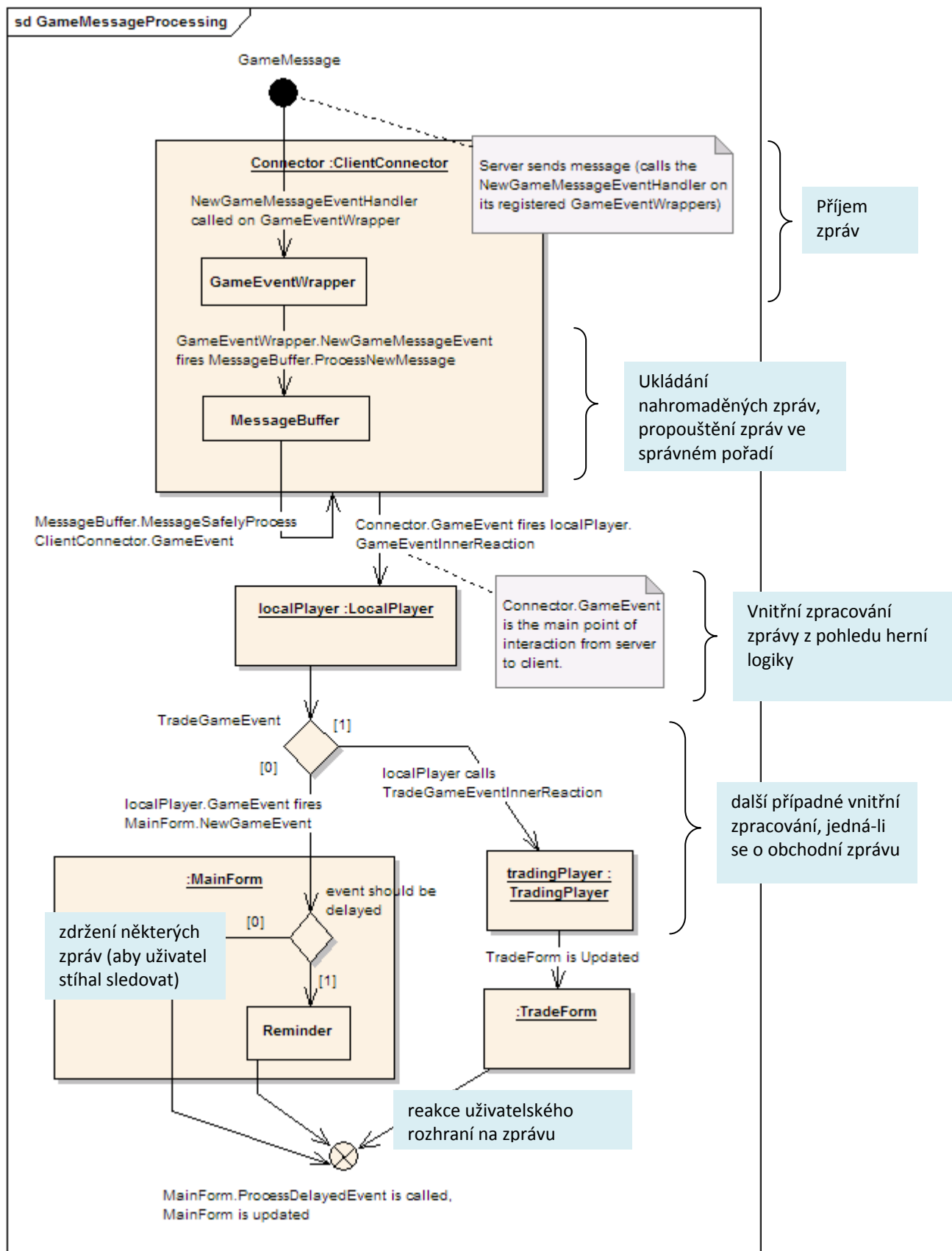


Diagram 9: Zpracování přichodí zprávy na klientovi

6 HLAVNÍ TŘÍDY HERNÍ LOGIKY

6.1 REPREZENTACE HRACÍHO PLÁNU

Program musí obsahovat přehlednou reprezentaci hracího plánu, se kterou se bude v programu snadno pracovat. Hrací plán se skládá ze tří základních druhů prvků:

- šestiúhelníková políčka s krajinami – *hex*,
- „křižovatky“ v rozích hexů, na kterých hráči staví vesnice – *site*,
- hrany mezi hexy, kde hráči staví své vesnice – *communication spot*.

Pro ilustraci obrázek části hracího plánu:



Obrázek 1: Část plánu s jednou postavenou vesnicí a silnicí

Reprezentace hracího plánu podporovat především snadné:

- dohledání prvku na hracím plánu a následné přístupu k jeho atributům,
- dohledání sousedících prvků nějakého prvku.

Nabízí se vytvořit objekt pro každý z prvků hracího plánu, z těchto prvků vytvořit tři kolekce a nakonec se hodí (z důvodu zapouzdření) i tyto tři kolekce vložit jako podobjekty do jednoho objektu.

Je třeba také pamatovat na to, že i když u jednotlivých krajin na orientaci nezáleží, u moří s přístavy orientace hexu hraje roli a tak hexy musí mít atribut určující jednu z šesti možných orientací.

Atributy hexů tak budou poloha, krajina, číslo (pro hod kostkami) a orientace

Společnými atributy sítí a communication spotů budou poloha a vlastník objektu (vesnice, město, cesty), u sítí bude atribut typ budovy – *BuildingType* (nic - *Nothing*⁸, prázdný – *Empty*, zakázaný - *Forbidden*⁹, vesnice – *Village*, město – *Town*) a obdobně u communication spotů typ cesty – *CommunicationType* (*Nothing*, *Empty*, cesta - *Road*)

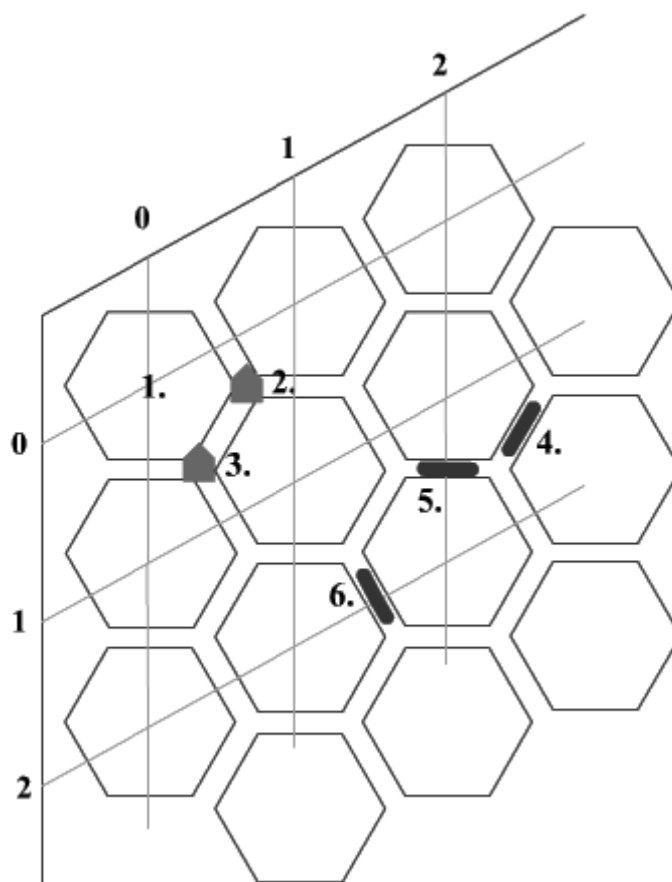
⁸ *BuildingType.Nothing* a *CommunicationType.Nothing* znamená, že tento site/communication spot, i když je fyzicky reprezentovaný, tak logicky do plánu nepatří, reprezentování i nehrajících sítí totiž usnadní indexaci.

⁹ *BuildingType.Forbidden* je příznak toho, že na jednom ze sousedních sítí už je postavena vesnice nebo město. Pravidla zakazují stavět vesnice na sousedících sitech, proto se při postavení nové vesnice všem sousedním sítím nastaví tento příznak.

6.1.1 STRUKTURA HEXŮ

programové uložení pole vytvořeného ze sousedících pravidelných šestiúhelníků by se na první pohled mohlo zdát složitější než ve skutečnosti je. Když si totiž člověk odmyslí šest možných směrů přechodu z jednoho hexu na jiný, lze se na hrací plán dívat jako na dvourozměrnou strukturu. Když postavíme všechny hexy tak, aby byly dvě jejich hrany vodorovně, potom:

- osa y bude svislá a kolmá na dvě vodorovné hrany hexů, začíná nulou a kladný směr vede dolů,
- osa x bude svírat úhel 120° s osou Y, bude také kolmá na dvě hrany hexů, začíná nulou a kladný směr vede vpravo nahoru.



Obrázek 2: Indexování na hracím plánu

Jednotky na osách jsou vyznačeny na obrázku, proto Hex označený 1. bude mít souřadnice $[0, 0]$, hex s číslem 4. bude mít souřadnice $[2, 3]$. Hex $[0, 0]$ je počátkem souřadné soustavy a bude vždy prvním reprezentovaným hexem. Hexy lze takto snadno ukládat v dvourozměrném poli.

Získanou dvourozměrnou souřadnou soustavu využijeme i pro ostatní prvky pole. U sítí i u communication spotů je ale potřeba sítí i hexy vhodně rozdělit mezi hexy (protože jeden site sdílí tři hexy a jeden communication spot dva hexy), například tak, že sítí v západním, severozápadním vrcholu hexu přísluší danému hexu. K tomu zavedeme „orientaci“ sítí (SiteOr), která bude právě určovat, ve kterém z těchto tří vrcholů hexu site leží. Na obrázku pak bude mít site označený 2. souřadnice $[1, 1, \text{SiteOr.northeast}]$ a site

označený 3. bude mít souřadnice [1, 1, SiteOr.west]. Pro uložení sitů tak budou potřeba dvě dvourozměrná pole, která budou odpovídat rozměry rozměrům pole hexů; v jednom poli budou uloženy všechny „západní“ a v druhém poli všechny „severozápadní“ sity.

U communication spotů je situace analogická, každému hexu přiřadíme ty, které leží na jeho jihozápadní, severozápadní a severní hraně, zavedeme „orientaci“ communication spotu (CommOr) a communication spoty budeme reprezentovat ve třech dvourozměrných polích, do kterých je rozdělíme podle orientace. Na obrázku pak bude mít spot s číslem 4. souřadnice [3, 2, CommOr.northwest], s číslem 5. souřadnice [2, 2, ComOr.north] a s číslem 6 souřadnice [2, 2, CommOr.southwest]

6.1.2 REPREZENTACE HRACÍHO PLÁNU V PROGRAMU

Čistě pro určování polohy (souřadnic) každého prvku jsou v programu použity třídy HexLoc, SiteLoc, CommLoc. Tyto třídy slouží čistě pouze k uchování souřadnic.

Třídy Hex, Site a CommunicationSpot¹⁰ už reprezentují konkrétní prvek na hracím plánu se všemi jeho atributy.

Kolekce Hexes, Sites a Communications sdružují odpovídající prvky a především publikují *indexem* (obdoba přetěžování operátoru []), dobrý popis lze nalézt v [1]), které umožňují pohodlně získat objekt Hex k odpovídajícím souřadnicím HexLoc (a obdobně u ostatních prvků). Tyto tři kolekce jsou zastřešeny třídou LogicalBoard, která tak reprezentuje celý hrací plán (a zpřístupňuje některé související metody). Třída LogicalBoard je použita jak v klientské tak serverové části.

Zbývá vyřešit dohledávání sousedících prvků ke každému typu prvku. Souřadnice prvků jsou odvozeny od souřadné soustavy určené hexy. Bylo by možné vytvořit u každého prvku v okamžiku jeho vsazení do plánu sadu referencí na sousedící prvky. Tento způsob by ale zbytečně zesložil inicializaci plánu. Díky zvolenému souřadnému systému lze totiž vytvořit statickou sadu pravidel, která vypočítá z daného *Loc objektu sousední objekty daného typu. Vznikne tak sada devíti statických metod třídy LogicalBoard, které vrátí pole odpovídajících Loc objektů. Tyto objekty jsou čistě objekty souřadnic sousedních objektů, může se stát, že vrácené souřadnice se ocitnou mimo nějaký konkrétní hrací plán (s tím musí volající počítat). Takto např. vypadá vrácení sousedních SiteLocs k nějakému HexLoc.

```
public static SiteLoc[] HexSiteNeighbour(int X, int Y)
{
    SiteLoc[] result = new SiteLoc[6];
    result[0] = new SiteLoc(X, Y, SiteOr.northwest);
    result[1] = new SiteLoc(X + 1, Y, SiteOr.west);
    result[2] = new SiteLoc(X + 1, Y + 1, SiteOr.northwest);
    result[3] = new SiteLoc(X + 1, Y + 1, SiteOr.west);
    result[4] = new SiteLoc(X, Y + 1, SiteOr.northwest);
    result[5] = new SiteLoc(X, Y, SiteOr.west);
    return result;
}
```

Struktura třídy LogicalBoard je také znázorněna na diagramu 10.

¹⁰ Prvky Site a CommunicationSpot jsou vytvořeny již v době inicializace plánu, tedy ne až když např. hráč umístí vesnici nebo cestu na dané místo.

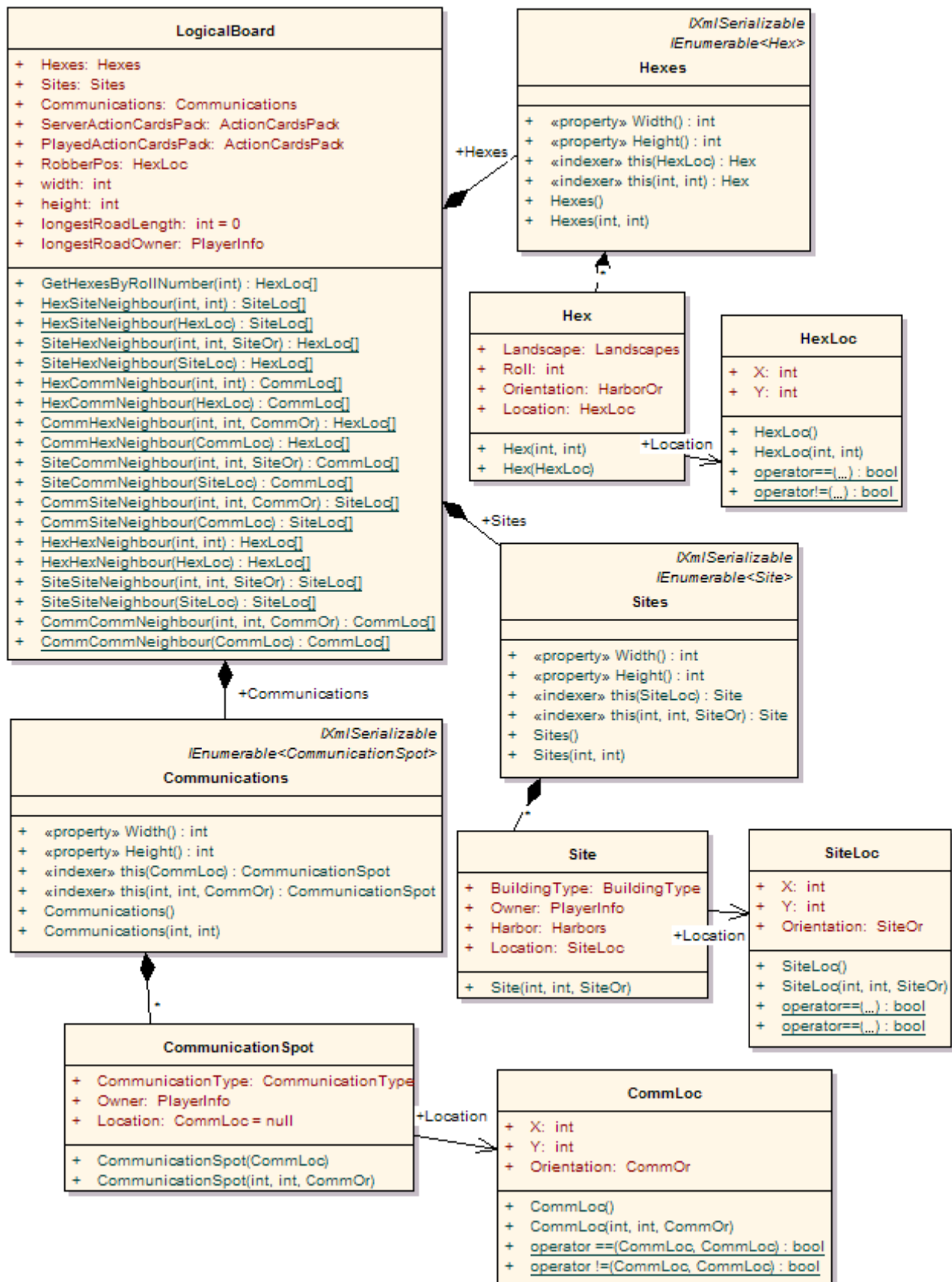


Diagram 10: Repräsentace hracího plánu

6.2 REPRESENTACE HRÁČŮ

Na hráče se lze podívat dvěma pohledy (které se samozřejmě z části prolínají) – jednak je to klient připojený k serveru a za druhé je to hráč hry. Nejjednodušší identifikací hráče je třída `PlayerInfo`, kterou mírně rozšiřuje třída `ConnectedPlayerInfo`.

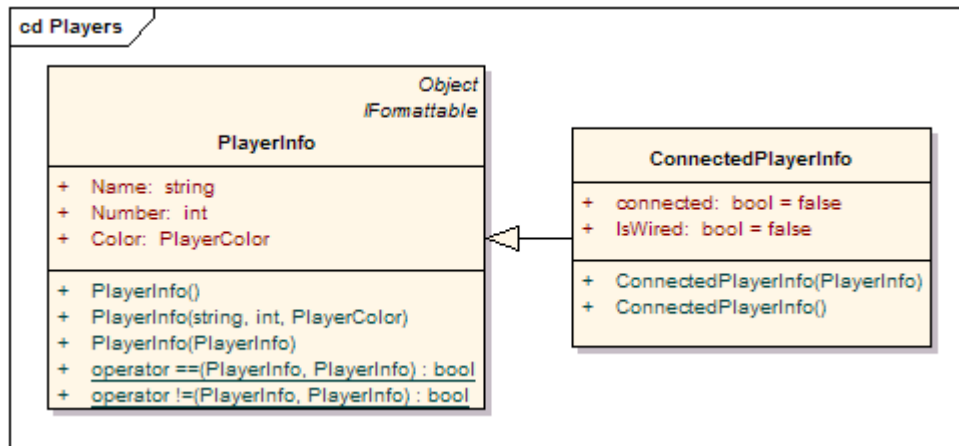


Diagram 11: Hráč

Třída `PlayerInfo` má široké použití v celém programu, především jako identifikátor hráče. Proto je parametrem většiny serverových metod a atributem zpráv `GameMessage` (jen u některých zpráv `GameMessage` je tento atribut ignorován). Typické použití `PlayerInfo` je např. hráče, který si postavil vesnici, při volání `IOneGameServer.PlaceVillage`. Identifikaci `PlayerInfo` umí vracet všechny třídy, které nějak reprezentují hráče. Třída `ConnectedPlayerInfo` navíc ještě obsahuje informaci, zda je hráč připojen k serveru (`connected`) ke hře a odebírá zprávy (`wired`).

Širší třídou, která reprezentuje hráče na serveru, je třída `DistantPlayer`. Ta již obsahuje všechny atributy hráče (kromě atributů z `PlayerInfo` také vlastněné akční karty, suroviny, počet vítězných bodů, speciální karty a dostupné přístavy), má metody pro připojování a odpojování hráče, ne ale metody pro vlastní hru (protože tyto metody nepotřebuje). Vesnice, města a silnice jsou součástí reprezentace hracího plánu a u třídy `DistantPlayer` nejsou.

Kolekce `DistantPlayers` tříd `DistantPlayer` především sdružuje všechny hráče (podle pravidel až čtyři) a kromě připojování a odpojování především publikuje indexer pro dohledání třídy `DistantPlayer` na základě identifikace `PlayerInfo`.

Metody pro provádění herních akcí neobsahuje třída `DistantPlayer`, ale třída `OneGameServer` a třída `Trader`.

U klienta je hráč reprezentován třídou `LocalPlayer` (a `TradingPlayer`). Ta je hlavní klientskou třídou, obsahuje stejné atributy, jako třída `DistantPlayer` a k tomu navíc metody, pro provádění herních akcí (které jsou provázané s uživatelským rozhraním). Třída `LocalPlayer` také zpracovává zprávy `GameMessage` zaslané serverem.

6.3 POMOCNÉ TŘÍDY HERNÍ LOGIKY

Pomocnými třídami jsou třídy:

- `MaterialsPack` – reprezentuje libovolný balíček surovinových karet. Slouží k držení surovin vlastněných hráčem a také k předání všech plateb.
- `Harbors` – reprezentuje množinu přístavů, přístavů je celkem šest druhů, pět pro vývoz konkrétní suroviny, jeden pro vývoz libovolné suroviny.
- `ActionCardsPack` – reprezentuje balíček akčních karet, podobná struktura jako `MaterialsPack`.

6.4 OBCHODOVÁNÍ

Obchodování je jednou z hlavních součástí celé hry. Jsou rozlišeny dva případy

- Obchodování se zámořím – hráč si mění svoje suroviny za suroviny z „banky“ (podrobnější popis v pravidlech hry). Tato herní akce není nijak odlišná od ostatních herních akcí (tedy je to akce, kterou provádí jen hráč na tahu a bez ostatních hráčů) a spočívá pouze v zavolání serverové metody a přijetí odpovědi v podobě `GameMessage`.
- Obchodování mezi hráči.

Obchodování mezi hráči podle pravidel probíhá vždy mezi právě dvěma hráči, kteří si po vzájemné dohodě vymění surovinové karty (vždy musí měnit alespoň jednu kartu). Obchodovat může vždy pouze hráč, který je na tahu, s libovolným jiným hráčem. Průběh domlouvání obchodu pravidla nijak nepředepisují. V programu je obchodování vyřešeno následujícím způsobem:

Obchodování u klienta zajišťuje třída `TradingPlayer` (svázaná s `LocalPlayer`). Protože si hráči vždy vyměňují balíček surovin za balíček surovin (tedy v programu `MaterialsPack` za `MaterialsPack`), má každý `TradingPlayer` svoji aktuální *nabídku* a *poptávku* (`TradingPlayer.Offer`, `TradingPlayer.Demand`). Obchod mohou dva hráči uzavřít, když se spárují jejich nabídky a poptávky.

Protože všichni hráči musí vidět nabídku i poptávku všech hráčů, existuje na každém klientovi instance třídy `TradingPlayer` pro každého hráče. Tyto (až čtyři) objekty sdružuje kolekce `TradingPlayers`. Protějškem tříd `TradingPlayer` a `TradingPlayers` je serverový objekt s rozhraním `ITrader` (ten je centrální, svázaný s `OneGameServer`)

Každý objekt `TradingPlayer` udržuje aktuální nabídku a poptávku přiřazeného hráče. Každý hráč může kdykoliv upravovat svoji nabídku i poptávku (voláním metod vzdáleného objektu přes proxy `ITrader`). Hráč může pomocí sady čtyř metod `ITrader` přidávat a odebírat jednotlivé suroviny z nabídky a poptávky.

Když po postupném upravování nabídek a poptávek dojde ke spárování nabídek a poptávek dvou hráčů, mohou tito hráči uzavřít obchod. To provedou voláním potvrzovacích metod, nejprve metody *Consider* (která uzamkne nabídku a poptávku) a když oba hráči zavolají metodu *Consider*, mohou zavolat metodu *Confirm*. Když

to oba udělají, dojde k výměně surovin. Pokud je obchod ve stavu *considered* nebo *confirm* a někdo ze zúčastněných hráčů změní nabídku nebo poptávku, přejdou oba zpět do počátečního stavu (*negotiation*). Stejně tak je stav zresetován i po tom, kdy se změní suroviny některého ze zúčastněných hráčů. Stav obchodu hráčů je jednosměrný (zatímco jeden hráč označil stav jako *considered*, druhý může být ve stavu *negotiation*) – K obchodu dojde, až když *oba* hráči označí obchod stavem *confirmed*.

Hráč na tahu může vyjednávat s několika hráči najednou, proto je potřeba udržovat stav vyjednávání s každým hráčem (hráč může být s některým hráčem ve stavu *negotiation*, s dalším ve stavu *considered* apod.). Udržování stavu obchodu s ostatními hráči zajišťuje třída *StatusWrap*. Třída *StatusWrap* informuje o změně stavu svojí událostí *StatusChanged*.

Zprávy týkající se obchodování mají svůj vlastní typ – *TradeGameMessage* – a rozšiřují typ *GameMessage* o některé atributy:

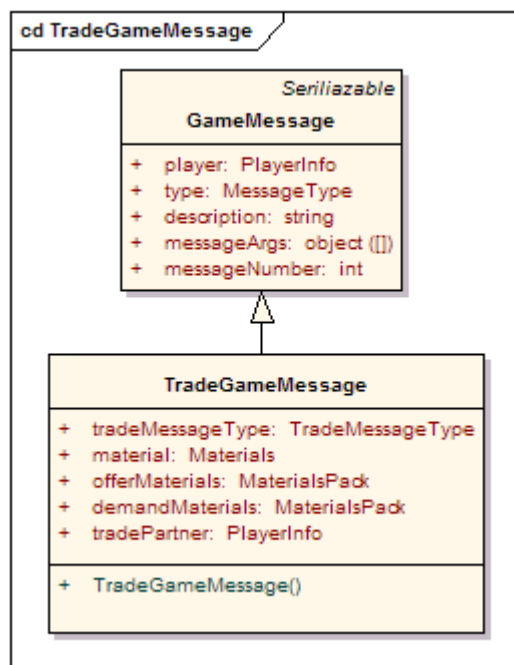


Diagram 12: Zpráva *TradeGameMessage*

Význam jednotlivých položek se mírně mění podle typu zprávy, v zásadě ale *tradePartner* určuje druhého hráče, ke kterému se zpráva vztahuje (prvního určuje *player* zděděný z *GameMessage*), atribut *material* označuje typ suroviny při přidávání/odebírání z nabídky, *offerMaterials* a *demandMaterials* nabídku a poptávku.

Příchozí zprávy *TradeGameMessage* předává třída *LocalPlayer* ke zpracování třídě *TradingPlayer(s)* – viz oddíl 5.3 Zpracování příchozí zprávy.

7 GRAFICKÉ ROZHRANÍ KLIENTSKÉ ČÁSTI

7.1 ZPRACOVÁNÍ HERNÍHO PLÁNU

Serverová část nemá žádné grafické rozhraní, jejím úkolem je pouze registrovat příslušné objekty, jejichž metody jsou dále volány vzdáleně. Server běží jako konzolová aplikace¹¹.

Grafické zpracování klientské části využívá části .NET Frameworku pro tvorbu formulářů – Windows Forms. Klientská část aplikace se tak stává z formulářů a dialogových oken. Program, kromě vestavěných standardních ovládacích prvků .NET Frameworku, využívá také několik specializovaných odvozených ovládacích prvků vytvořených pro účely hry – User Controls a Custom Controls. Všechny prvky používají k vykreslování .NET GDI+.

Základem rozhraní jsou formuláře MainForm a TradeForm, které jsou svázané s třídami LocalPlayer resp. TradingPlayer(s). Třídy LocalPlayer a TradingPlayer zajišťují herní logiku, formuláře pak poskytují rozhraní k jejich metodám a zobrazují samotnou hru.

Pro zobrazení hracího plánu je použita sada vlastních ovládacích prvků. Ty jsou všechny potomky standardní třídy Control (která implementuje základní funkce ovládacích prvků – jejich umístění do formuláře a reakce na uživatelský vstup). Všechny tyto prvky jsou zařazeny do namespace CustomControls, zde je jejich základní přehled:

Ovládací prvek z uživatelského rozhraní	Odpovídající prvek herní logiky
HexButton (HexButtons)	Hex (Hexes)
SiteButton (SiteButtons)	Site (Sites)
CommunicationButton (CommunicationButtons)	CommunicationSpot (Communications)
GraphicalBoard	LogicalBoard

Tabulka 1: Prvky uživatelského rozhraní ve vztahu k prvkům herní logiky

Je vidět, že prvky grafického rozhraní věrně kopírují strukturu třídy LogicalBoard. S touto třídou je pak třída GraphicalBoard provázána a reaguje tak na změny na herním plánu (přidání vesnice/města/silnice).

GraphicalBoard

- zobrazuje jednotlivé prvky
- při umísťování prvků na plán zobrazuje uživateli povolené lokace
- reaguje na klikání myší

Třída GraphicalBoard je konfigurovatelná tak, že lze měnit velikost jednotlivých prvků a například tak reagovat na velikost okna (v programu ale této možnosti využito není).

¹¹ Možné rozšíření serverové části bude probráno v části 8.

7.2 OBCHODNÍ DIALOG

Vnitřní průběh obchodování byl popsán v oddíle 6.4 Obchodování, zbývá ještě popsat zobrazení průběhu obchodování na klientovi. K tomuto slouží formulář TradeForm a sada uživatelských ovládacích prvků.

K zobrazení stavu jednotlivých objektů TradingPlayer slouží ovládací prvek TradeControl. Ten kromě jména a barvy hráče zobrazuje

- suroviny v nabídce a poptávce daného hráče
- stav obchodování (negotiation/match/considered/confirmed) k ostatním hráčům.

Prvek TradeControl sám o sobě použit není, ale až jeho dva oddělené prvky

- PassiveTradeControl použitý pro zobrazení stavu *ostatních* hráčů, tedy těch, kterým nepatří daný formulář. PassiveTradeControl přidává tlačítko Přijmout, které posouvá obchod do další fáze.
- ActiveTradeControl použitý pro zobrazení stavu hráče, kterému patří formulář. Tento prvek přidává tlačítka pro upravování hráčovy nabídky a poptávky.

Prvky TradeControl jsou navázané na události objektů TradingPlayer a tak obnovují svůj stav při přecházejících zprávách TradeGameMessage.

Zobrazování stavu obchodu obstarává prvek TradeIndicator (na formuláři jako barevná šipka), který je na každém TradeControlu třikrát (jednou pro každého z ostatních hráčů).

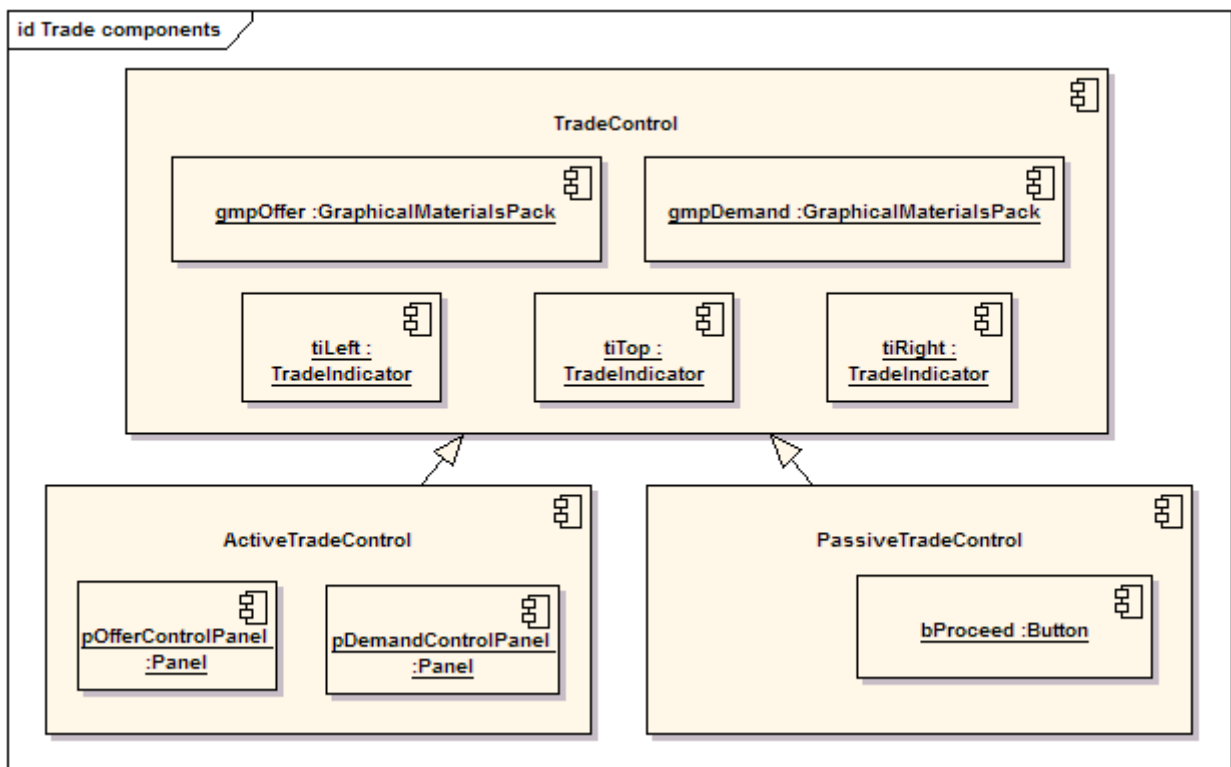


Diagram 13: Struktura ovládacích prvků obchodování

8 MOŽNÁ ROZŠÍŘENÍ

Program plně implementuje všechna pravidla hry Osadníci z Katanu, věrně kopíruje deskovou hru a umožňuje síťové hraní ve více hráčích. V budoucnu by bylo možné program rozšiřovat v několika různých směrech.

8.1 DESKOVÁ ROZŠÍŘENÍ PŮVODNÍCH OSADNÍKŮ

Fenomenální úspěch deskové hry motivoval autora hry k vytvoření několika dalších rozšíření této hry, her, které v názvu obsahují slova osadníci nebo Katan už dnes existují desítky. Některé z nich mají sice s původní hrou společné jen málo, existuje ale několik přímých rozšíření původní hry (dokonce bez původní hry nejsou ani hratelné).

Rozšíření hry pro pět až šest hráčů pouze zvětšuje hrací plán (a samozřejmě povoluje hrát až v šesti lidech), nepřidává žádné nové herní mechanismy, a tak by bylo implementování do stávajícího programu poměrně jednoduché.

Rozšíření Námořníci z Katanu už přidává několik nových herních prvků – především možnost stavby lodí (které se ale chovají stejně jako silnice, jen jsou položeny na hranách hexů s mořem) a osídlování dalších ostrovů. Kromě přidání nového prvku – lodě, se další pravidla v základní variantě tohoto rozšíření nemění a tak by implementace i tohoto rozšíření byla bezproblémová. Součástí tohoto rozšíření je ale také několik herních scénářů, které mírně upravují některá pravidla hry. Tyto změny nejsou aditivní, ale skutečně mění stávající principy hry a tak by bylo nutné pro každý scénář upravovat třídy LocalPlayer a OneGameServer a dále vše, co se od těchto tříd odvíjí.

Rozšíření Města a rytíři už naprosto zásadně mění herní systém, například úplně odstraňuje původní akční karty a rytíře, přidává nové druhy surovin a částečně mění způsob jejich těžby. K tomu přidává několik desítek druhů akčních karet, z nichž každá má jinou funkci (pro srovnání, v původních Osadnících je jen pět druhů akčních karet). Pravidla tohoto rozšíření jsou mnohem komplikovanější, než u původních Osadníků a tak při případné implementaci by bylo nutné dělat velké změny v třídách herní logiky. Především implementace funkce každé z akčních karet by byla časově náročná.

Další rozšíření Osadníků jsem nehrál.

8.2 POČÍTAČEM HRANÍ PROTIHRÁČI

Program neobsahuje možnost nechat uživatele hrát proti počítačem ovládaným protihráčům. Implementace umělé inteligence by byla jistě zajímavá, protože na Osadníky nelze použít jednoduché algoritmy z piškvorek nebo dámy. Za prvé, Osadníci obsahují některé skryté informace a prvek náhody. Největším oříškem by ale jistě bylo „naučit“ počítač, které obchody s ostatními hráči jsou výhodné a které ne.

8.3 ODLIŠNÉ HRACÍ PLÁNY

Zatímco desková hra samotná je v otázce rozšiřování plánu dost omezená – hráč má k dispozici jen určitý počet hexů každého druhu, které sice může permutovat, ale větší plán nikdy vytvořit nemůže – počítačová simulace deskové hry takto omezující být nemusí. Program umožňuje serveru předložit buď přímo předepsaný plán, jaký má pro hru použít (přesné rozestavení políček-hexů a čísel na nich), nebo může být předložen pouze základ plánu, kterého se má server držet a tu část plánu, která není pevně definovaná, dogenerovat – takto funguje například standardní hrací plán, ve kterém mohou být vnitrozemská políčka libovolně permutována, potom jsou na ně umístěna čísla.

Vhodnou reprezentací pro hrací plán (jak pro možnost úplného tak částečného definování) je XML dokument. Ten by v možném rozšíření mohl být doplněn také samostatným WYSIWYG editorem plánů.

Popis toho, jak vypadá dokument hracího plánu lze najít v dodatku 4 – Konfigurace hracího plánu.

8.4 „SPRAVEDLIVĚJŠÍ“ KOSTKA

Podíl náhody je jeden z hlavních atributů jakékoliv deskové hry. Osadníci z Katanu jsou charakterizováni jako hra spíše s malým podílem náhody. Přesto se ale některým hráčům může zdát, že náhoda má na hru až příliš velký vliv (někomu naopak náhodný prvek může vyhovovat).

Tah každého hráče začíná hodem dvěma kostkami, součet bodů na kostkách určí pole, která v tomto kole budou vynášet suroviny. Možných kombinací hodů je celkem 36, možných součtů je ale méně – jsou to čísla od 2-12. Z těchto čísel jsou nejpravděpodobnější „střední“ součty – součet 8 nastává v pěti případech z šestatřiceti možných (2+6, 3+5, 4+4, 5+3, 6+2), podobně součty 6 (5 z 36) a 7 (6 z 36). Naopak součty 2 resp. 12 nastávají pouze v jednom případě ze všech 36 možností (1+1 resp. 6+6). Hráči se při umísťování svých vesnic řídí právě těmito poměry a snaží se obsadit především políčka s čísly 6, 8, 5 a 9 (číslo 7 nemá žádné políčko).

Problém je, že se kostky často nechovají tak, „jak by měly“ – podle poměrů možných součtů – a tak i hráč, který šikovně obsadil políčka se slibnými součty, může prohrát jen kvůli tomu, že jeho čísla prostě nepadala. Někteří hráči proto upravili herní systém tím, že „usměrnili“ náhodu tak, aby se více držela očekávaného chování.

Nejběžnějším způsobem asi je napsat na kartičky všech 36 kombinací hodů na dvou kostkách, z těchto kartiček vždy na začátku tahu jednu vybrat (tím se nahradí hod kostkami) a dát ji mimo balíček. Až když kartičky dojdou (po 36 hodech), začne se losovat opět z celého balíčku.

Hra umožňuje nahradit objekt standardní kostky libovolným jiným objektem. Třída `OneGameServer` obsahuje jako atribut referenci na objekt typu `Dices`, což je abstraktní třída publikující jedinou metodu `Roll`, která vrací dvě čísla – hody na dvou kostkách. Program poskytuje standardní implementaci `SimpleDices`, která vrací vždy dvě náhodná čísla získaná z generátoru `System.Random`. Tuto standardní implementaci je možné

nahradit libovolným jiným potomkem abstraktní třídy Dices (např. takovým, který simuluje model z předchozího odstavce).

8.5 ZMĚNA ARCHITEKTURY A ZMĚNA KOMUNIKAČNÍ METODY

Program funguje jako klient – server aplikace, kde se více hráčů připojuje k jednomu serveru, na kterém vždy může běžet pouze jedna hra. Připojení k serveru funguje přes volání metod serverového objektu OneGameServer přes proxy k tomuto objektu u klienta.

Díky této architektuře může server běžet jako samostatná aplikace.

Serverové objekty OneGameServer mohou dobře existovat vedle sebe a tak by nebyl větší problém vytvořit serverovou aplikaci, na které by mohlo běžet víc her (jen pro ukázkou, že objekty OneGameServer mohou existovat vedle sebe jsem vytvořil jednoduchou implementaci serveru více her – MultiServer).

Pokud by nějaký počítač trvale hostoval takový server více her, bylo by možné kolem něj vystavět větší herní portál, který by mohl obsahovat přihlašování hráčů, ukládání her, statistiky, možnost přisednout k hrám, které hraje někdo jiný, a další podobné funkce běžné na herních serverech.

Pokud by měl takový server běžet v prostředí Internetu, mohlo by být vhodné nahradit Remoting jinou komunikační technologií, více svázanou s Internetem – webovými službami. Místo volání vzdálených metod přes proxy by klienti volali zveřejněné WebMethods, které by pak volali metody objektu OneGameServer na serveru. Tato změna komunikace by si vyžádala nahradit třídy RemotingClientConnector (tedy bylo by nutné vytvořit odpovídající implementaci abstraktní třídy ClientConnector) a ServerConnector.

9 ZÁVĚR

Cílem práce bylo převést populární deskovou hru Osadníci z Katanu na PC. Výsledný program plně implementuje deskovou hru se všemi jejími pravidly a umožňuje síťovou hru dvou až čtyřech hráčů (tolik maximálně povolují pravidla deskové hry).

Výsledný program je součástí práce a lze ho nalézt na přiloženém CD. Program lze nainstalovat a spustit na libovolný počítač s operačním systémem podporujícím platformu Microsoft .NET 2.0, žádné další požadavky program neklade – i serverovou část může na svém počítači spustit libovolný uživatel bez instalace žádných dalších programů.

Program lze rozšiřovat a upravovat změnami tříd a modulů na straně serveru i klienta tak, aby nové třídy mohly spolupracovat se zbytkem programu. Program odděluje logiku hry od logiky síťové komunikace a síťovou komunikaci dále dělí do několika vrstev. Jednotlivé principy v implementované v programu by tak šly dobře použít i v jiných programech – například nižší vrstva síťové části může být použita jako základ nějakého síťového programu s komunikací přes centrální server, spolu s vyšší vrstvou (ServerConnector, ClientConnector) by mohly tvořit komunikační část např. pro jinou deskovou hru podobného typu (tahová hra, u které lze komunikaci převést na komunikaci přes centrální server).

Dalším mým záměrem bylo popsat problematiku převodu deskových her na PC. Analýza logiky hry (kapitola 2) demonstruje, jaké body musí programátor libovolné deskové hry vzít v úvahu před tím, než začne s implementací dané hry. Především identifikování sdílených částí, hlavního stylu komunikace, stupně variability hry a míra náhody mají velký vliv na budoucí podobu programu a proto je třeba všechny tyto aspekty zvážit před vlastním započítáním implementace hry.

Zatímco klasické deskové hry – šachy, dáma atd. – mají velmi jednoduchá pravidla a málo herních prvků a jsou hodně abstraktní, moderní deskové hry se naopak snaží být co nejvíce originální, mají velký počet herních prvků a rozsáhlejší pravidla – proto jsou tyto hry zábavnější a přístupnější (ale také dražší) a také náročnější pro převod do podoby počítačové hry.

Použité postupy a základní strukturu a rozdělení částí programu pak lze aplikovat pro implementaci i jakékoliv jiné deskové hry.

10 LITERATURA

[1] Liberty J.: *Programming C#, 2nd Edition*, O'Reilly 2002

[2] Prorise J.: *Programming Microsoft .NET*, Microsoft Press 2002

11 DODATKY

DODATEK 1: KONFIGURACE SERVERU

Serverem teď myslím tu část programu, která registruje třídu `OneGameServer` a umožňuje klientům získávat reference na tuto třídu. Protože registrovaný typ (`OneGameServer`) je neměnný, lze nakonfigurovat pouze port, na kterém bude aplikace poslouchat. Port lze nastavit v konfiguračním souboru nebo parametrem příkazové řádky (`-port 2343`). Pokud uživatel port nenastaví, bude použit port 5678. Dalším možným parametrem je cesta k souboru definujícímu hrací plán (`-board plan.xml`). Více o možnostech tvorby plánu v části Dodatku 4 - Konfigurace hracího plánu.

DODATEK 2: KONFIGURACE KLIENTA

Konfigurace klienta je oproti tomu bohatší. V konfiguračním souboru jsou uloženy servery, ke kterým se klient může připojovat. Tento soubor lze upravovat jak pomocí textového editoru (ale nepředpokládá se, že by uživatelé používali tento způsob), tak především v klientské aplikaci v dialogu *Výběr serveru*. Tento dialog umožňuje klientovi zadávat IP adresy serverů a porty, na kterých poslouchají a pojmenovávat jednotlivé servery.

Při otevření dialogu program testuje dostupnost všech serverů, které si přečte z konfigurace (testuje přítomnost zaregistrovaného objektu). Dostupné servery jsou uživateli zvýrazněny. Vlastní testování serverů probíhá na pozadí (tak, aby uživatel nemusel čekat na dobíhání testů – které může trvat déle, pokud jsou některé servery nedostupné a program čeká na vypršení timeoutu), test každého serveru běží v samostatném vlákně. Opětovné otestování serveru může uživatel také vyvolat sám.

Všechny změny v seznam serverů v dialogovém okně se po uzavření dialogu opět promítnou do konfiguračního souboru tak, že při příštím spuštění se uživateli zobrazí stejný seznam serverů.

DODATEK 3: DEPLOYMENT

K programu byl vytvořen také standardní instalátor (ve formě `.msi` balíčku), který uživatele v několika krocích provede instalací aplikace. Při tvorbě instalátoru jsem využil *Setup Project* Visual Studia 2005.

Instalátor mj. také kontroluje, zda je již na uživatelově počítači nainstalovaný `.NET Framework` ve verzi 2.0 a pokud ne, doinstaluje ho.

DODATEK 4: KONFIGURACE HRACÍHO PLÁNU

PRINCIP KONFIGURACE PLÁNU

Serveru lze předat XML dokument, podle kterého má vygenerovat hrací plán. Dokument se skládá ze dvou částí. V první části jsou uvedeny parametry budoucího plánu – výška a šířka plánu, počet políček od každé krajiny a moře a počet čísel na políčkách (čísla, podle kterých se určuje, která pole vynáší). Druhá část dokumentu pak určuje vlastní rozložení políček.

Plán může být úplně fixní (s pevně daným rozložením políček i čísel), nebo mohou být některá čísla na políčkách v dokumentu neurčená – v takovém případě server během generování tato políčka očíslovuje (drží se počtů čísel uvedených v první části dokumentu).

Při návrhu plánu lze také některá políčka označit jako neurčená – nastavením typu krajinky na AnyInland zajistí, že server při generování dosadí na toto políčko některý z vnitrozemských typů krajiny. Podobně typ AnyHarbor dosadí jeden z přístavů (tady je nutné u hexu uvést taky orientaci) a nakonec typ Any povolí na onom políčku libovolný typ krajiny.

Tímto částečně náhodným způsobem je definován i základní plán hry v originálních pravidlech – počty čísel jsou pevné, ale jejich rozmístění ne, jsou pevně dané druhy políček vnitrozemí/přístav/moře a počty jednotlivých krajin. Vnitrozemská políčka a přístavy mohou být před každou hrou libovolně permutována.

Vlastní generování plánu zajišťuje třída BoardFactory, nejprve je potřeba načíst XML dokument s popisem plánu, vlastní generování pak obstarává metoda CreateBoard. Který dokument má server použít při tvorbě plánu se nastaví parametrem `-board_jmenosouboru.xml` při spuštění serveru.

POPIS DOKUMENTU PRO GENEROVÁNÍ PLÁNU

Součástí programu je xml schéma dokumentu pro generování plánu. Soubor Board.xsd lze nalézt na příloženém CD, zde je význam jednotlivých elementů:

- Board – kořenový element, atributy:
 - width, height – rozměry plánu
 - name – jméno plánu
 - robber – počáteční umístění zloděje, možné jsou hodnoty placeOnDesert (zloděj je na začátku hry umístěn na poušti) a placeOnMarkedHex – zloděj je na políčku označeném v dokumentu
 - numberingStrategy – způsob očíslování políček, je možno použít StandardNumbering (dle pravidel, pouze pro základní plán) a RandomNumbering (náhodné rozmístění čísel)
- GenerationData – první podelement elementu Board, obsahuje informace pro tu část plánu, která není dokumentem pevně určená a musí být částečně degenerována třídou BoardFactory
 - LooseInland – element obsahující počty jednotlivých krajin, které nejsou pevně umístěné

- LooseHarbors – element obsahující počty jednotlivých přístavů, které nejsou pevně umístěné
- LooseSeas - element obsahující počet políček moře, které nejsou pevně umístěné
- RollNumbers – element obsahující čísla určující výnos, která budou rozmístěna na jednotlivá políčka a počet každého čísla. Tyto čísla budou umístěna na ta políčka, která nemají v dokumentu čísla pevně určená
- Fields – druhý podelement elementu Board, obsahuje jednotlivá políčka plánu. Políčka, která nebudou vyjmenována v této části, ale přesto do pole patří, budou mít nastaven typ krajiny na empty – hexy se pak nezobrazují. Políčka jsou rozdělena do řádků (elementy Row s atributem number – číslo řádku). Každý element Row pak má podelementy Hex
 - Hex – atributy:
 - x, y – souřadnice hexu
 - orientation – orientace hexu (hraje roli u přístavů)
 - landscape – krajina, může být buď pevně daná, nebo jen částečně – hodnotou AnyInland, AnyHarbor a Any – při použití těchto hodnot BoardFactory při generování plánu vybere náhodná vyhovující políčka z volných políček z části /Board/GenerationData/LooseInland resp. /Board/GenerationData/LooseHarbors, /Board/GenerationData/LooseSeas
 - robber – pokud je nastaven na hodnotu „true“, bude se na tomto políčku na začátku hry nacházet zloděj