PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

ROČNÍKOVÝ PROJEKT

Projektový seminář

Latrunculi

Červen 2016

Ondřej Grätz
Aplikovaná informatika, III. ročník

Abstrakt

Implementace deskové hry Latrunculi s použitím *.NET Framework* pro operační systém *Windows*.

Obsah

Seznam obrázků

1. Zadání projektu

Cílem projektu je vytvoření hry pro desktopový operační systém s grafickým uživatelským rozhraním (GUI) dle standardů.

Požadavek na přenositelnost spustitelných souborů nebyl stanoven. Uživatelské rozhraní se předpokládá objektově orientované s použitím oken a standardních prvků (hlavní nabídka, nástrojová lišta, tlačítka).

2. Volba technologie

2.1. Výběr operačního systému

Já jsem pro vývoj i běh hry (aplikace) zvolil operační systém *Windows*, jelikož je mi dobře známý a také proto, že je to s podílem 52 % (viz [4]) mezi uživateli i programátory nejpoužívanější operační systém pro osobní počítače.

2.2. Výběr vývojového prostředí

Pro vývoj byl zvolen nástroj *Visual Studio* od firmy *Microsoft*, jazyky *C#* a *F#* a pro vývoj uživatelského rozhraní *Windows Presentation Framework* (WPF).

Pro zvolený operační systém (*Windows*) by s ohledem na zadání bylo výhodné použít jeden z následujících typů aplikací.

- Win32 (Visual C++ s použitím Windows API nebo MFC)
- .NET Framework + WinForms
- .NET Framework + WPF
- Windows Runtime (C++/CX)

Win32 a *Windows Runtime* poskytují nejmenší úroveň abstrakce. Vývoj tohoto typu aplikací vyžaduje větší znalosti a zkušenosti programátora. *Windows Runtime* aplikace je navíc možné spustit pouze pod operačním systémem *Windows 8* (nebo novějším). Výhodou je ale standardní vzhled výsledných aplikací a nejlepší výpočetní výkon.

Použití *WinForms* by bylo výhodné, jelikož vývoj je jednoduchý (událostmi řízený, objektově orientovaný). Výsledná aplikace má vzhled v souladu se standardy operačního systému a očekáváním uživatele. Nevýhodou je strohý design, který se příliš nepřizpůsobuje rozlišení obrazovky a jehož vzhled se může jevit zastaralý. Design je navíc velmi svázán s kódem, který má na starosti výpočty a samotnou logiku aplikace.

Naproti tomu u *WPF* je již při vývoji myšleno na responzivní design. Prvky uživatelského rozhraní mají velmi bohaté a pro programátora snadno použitelné vlastnosti, které umožňují přízpůsobit aplikaci různým velikostem obrazovky a různým způsobům vstupu od uživatele (dotykové obrazovky, přizpůsobení pro zrakově nebo tělesně hendikepované uživatele). Návrh uživatelského rozhraní je navíc velmi dobře oddělen od samotného kódu a umožňuje s použitím nástroje *Microsoft Blend* výrazně zasáhnout do designu aplikace i grafikovi (bez nutnosti znalosti programování a bez zásahu do modelu aplikace).

3. Organizace kódu

3.1. Zvyklosti pro .NET aplikace

Při vývoji pro *.NET Framework* je nutno veškerý kód umisťovat do metod tříd. Třídy musejí být povinně organizovány do jmenných prostorů. Jmenné prostory je vhodné stromově strukturovat dle navržené architektury aplikace.

Pro *.NET* aplikace (s výjimkou aplikací v jazyku F#) je typické rozdělení zdrojového kódu do několika projektů, které jsou přidány do skupiny projektů (Solution). Rozdělení se provádí na základě jmenných prostorů (`namespaces`) tak, aby třídy patřící do stejného prostoru byly ve stejném projektu.

Spustitelný (EXE) soubor obsahje jádro programu a definice hlavních částí uživatelského rozhraní, ale komponenty uživatelského rozhraní i třídy obsahující doménový nebo datový model, rozhodovací logiku a třídy pro práci se soubory, databázemi či síťovou komunikaci jsou umístěny v tzv. knihovnách tříd (DLL soubory). Toto dělení poté představuje výhodu při požadavku na změnu typu výsledné aplikace. Lze například textové rozhraní aplikace nahradit GUI. Nebo nahradit samostatně spustitelnou konzolovou aplikaci službou systému Windows. Společné součásti tak mohou být opětovně využívány (sdíleny).

DLL i EXE soubory poté obsahují překladačem vytvořený *bytecode*, který je poté při spuštění aplikace částečně kompilován do strojového kódu (just-in-time kompilace) a částečně také interpretován běhovým prostředím *.NET*.

3.2. Prevence cyklických závislostí

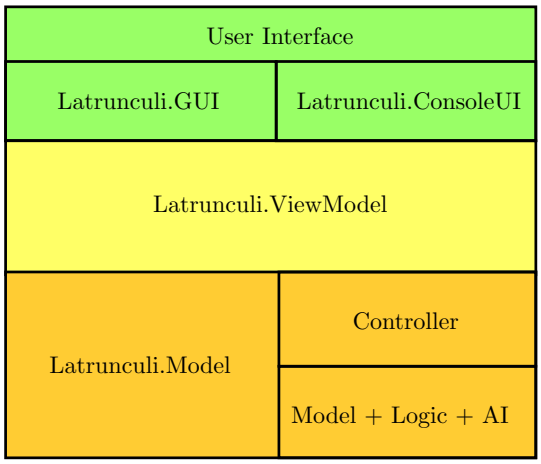
Abstraktní rozdělení kódu na vrstvy pomáhá při vývoji předcházet cyklickým (rekurzivním) závislostem mezi třídami (respektive mezi jednotlivými knihovnami tříd). Nevhodné závislosti mezi třídami by totiž mohly při dokončování aplikace způsobit komplikace, které by v krajním případě bránily překladu aplikace a vyžadovaly by refaktorování celého kódu. Jako prevence se proto činnosti jednotlivých součástí seskupují tak, aby vyšší vrstvy byly závislé na nižších vrstvách. Nevhodné je naopak odkazovat se z nižších vrstev na vyšší, ačkoliv i tato možnost může být potřebná. V případě potřeby se ale namísto reference používají události tak, že nižší vrstva generuje událost a vyšší vrstva událost zachytává. Bližší informace viz [6].

3.3. Soubory projektu

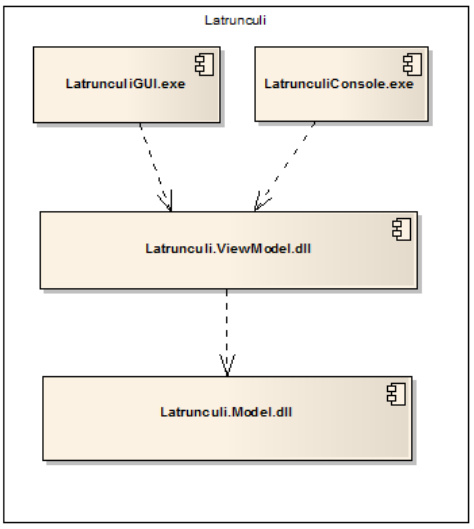
Třídy projektu jsou organizovány v podprostorech jmenného prostoru `Latrunculi`.

Rozvrstvení aplikace je zobrazeno na obr.1.. Tomu odpovídající výsledné přeložené soubory (assemblies) a jejich závislosti jsou zobrazeny na obr.2.. Názvy souborů jsou shodné s názvy jmenných prostorů, které jsou v jednotlivých souborech obsaženy. Výjimku tvoří spustitelné soubory, u kterých je záměrně oproti názvu jmenného prostoru, který obsahují, vynechána tečka, aby bylo případné spouštění z příkazového řádku pohodlnější.

4. Rozvrstvení aplikace



Obrázek 1. Vrstvy aplikace



Obrázek 2. Závislosti souborů

4.1. Vrstva uživatelského rozhraní

Nejvyšší vrstvou v našem diagramu z obr.1. je *User Interface*, tj. uživatelské rozhraní. Vrstva je horizontálně rozdělena, jelikož naše aplikace má dva typy uživatelských rozhraní. Konzolové rozhraní (`Latrunculi.ConsoleUI`) a grafické rozhraní (`Latrunculi.GUI`).

Reference

[1] Mgr. Jan Outrata, Ph.D.: *Projekt - implementace*, http://outrata.inf.upol.cz/courses/ps/navrh.pdf, listopad 2008.

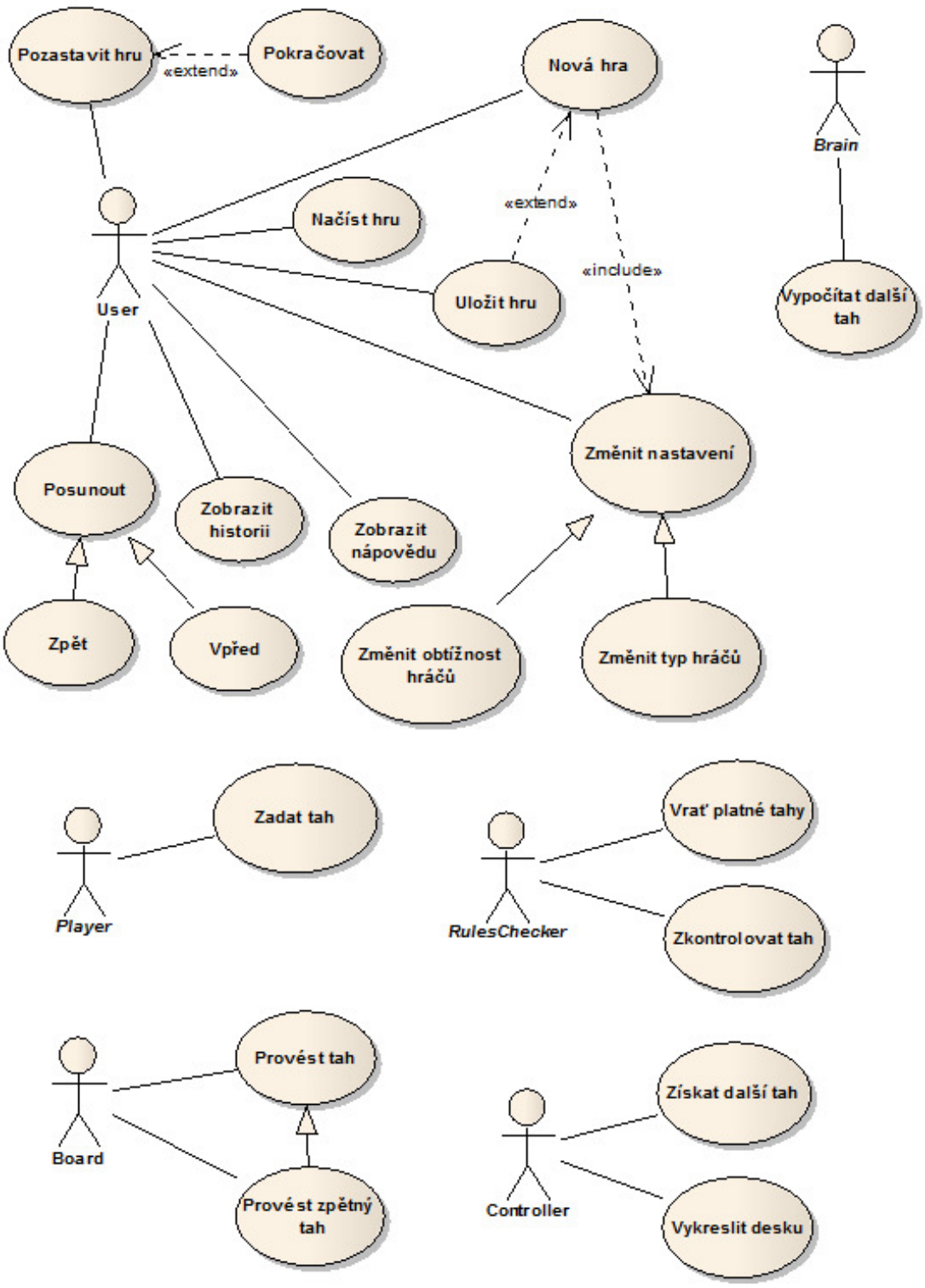
[2] Mgr. Jan Outrata, Ph.D.: *Projekt - analýza a návrh*, http://outrata.inf.upol.cz/courses/ps/implementace.txt, listopad 2010.

[3] Mgr. Tomáš Kühr, Ph.D.: *Algoritmy realizující počítačového hráče*, http://www.inf.upol.cz/downloads/studium/PS/algoritmy.pdf, říjen 2011.

[4] Wikipedia: *Usage share of operating systems*, https://en.wikipedia.org/wiki/Usage_share_of_operating_systems#Desktop_and_laptop_computers, květen 2016.

[5] Scott Wlaschin: *F# For Profit And Fun*, https://fsharpforfunandprofit.com/, květen 2016.

[6] Scott Wlaschin: *Dependency Cycles*, https://fsharpforfunandprofit.com/series/dependency-cycles.html, květen 2016.



Obrázek 3. Případy užití