ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ v PRAZE

Fakulta elektrotechnická

Katedra kybernetiky

diplomová práce

PROSTŘEDÍ PRO SNADNÉ VYTVÁŘENÍ JEDNODUCHÝCH (LOGICKÝCH) HER

Tibor Strašrybka

Vedoucí diplomové práce: ING. PETR NOVÁK

Studijní program: *Elektrotechnika a informatika*

Obor: *Biomedicínské inženýrství*

květen 2010

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Tibor S t r a š r y b k a

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný

Obor: Biomedicínské inženýrství

Název tématu: Prostředí pro snadné vytváření jednoduchých (logických) her

Pokyny pro vypracování:

1. Prostudujte možnosti snadného vytváření jednoduchých (logických) her (např. pexeso, patnáctka,...) pouze pomocí několika konfiguračních souborů (např. souboru obsahujícího grafickou reprezentaci dané hry a souboru obsahujícího (logická) pravidla neboli postup hry).
2. Navrhněte a implementujte potřebné prostředí (podporu) pro vytváření her tímto způsobem. Do navrženého prostředí budou vstupovat již zmíněné konfigurační soubory a tyto budou definovat jak vzhled tak i postup hry.
3. Jako ukázku vhodnosti a použitelnosti navrženého prostředí vytvořte (nejméně) tři jednoduché (logické) hry zaměřené na snadnou ovladatelnost pro uživatele s omezenou schopností pohybu.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Petr Novák

Platnost zadání: do konce zimního semestru 2009/2010

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 3. 9. 2008

Anotace

Hlavním cílem práce je vytvoření (naprogramování) běhového herního prostředí se zjednodušeným ovládáním přístupným i handicapovaným uživatelům. Podpůrná aplikace je proto ovladatelná i pouze jediným vstupem (tlačítkem), ale zároveň umožňuje i komfortnější ovládání celou klávesnicí a/nebo myší bez nutnosti přepínání. Celá aplikace je optimalizována na co nejmenší nutný počet úkonů.

Do prostředí vstupují jednoduché hry a hlavolamy popsané jednotnou sadou XML souborů a souborů grafické reprezentace. Nejdůležitějším souborem je sada pravidel, jimiž je řízen průběh hry. Navržený systém pravidel zajišťuje, že v každém kole hry jsou nabízeny pouze platné tahy, díky čemuž je usnadněn jejich výběr.

Pravidla jsou vyhodnocována po každém platném tahu zvláštním modulem za pomoci sady regulárních výrazů. Syntax pravidel je původní a vychází z jazyka C#. S daným systémem pravidel je kompatibilních více než 70 her a hlavolamů, z nichž 12 bylo pro ukázku realizováno.

Klíčová slova

Běhové prostředí – hra – jednoduchá hra – logická hra – desková hra – maticová hra – hlavolam – jednoduché ovládání – ovládání jedním vstupem – rozhraní – pravidlo – regulární výraz – parsování – XML – C# – .NET

Hry a hlavolamy: Bludiště, Jezdcova procházka, Loydova patnáctka, Návrat z Klondiku, Netwalk, Nonogramy/Kódované obrázky/Malované křížovky, Pexeso, Sokoban, Sudoku, Temné rovnice, Viktoriánský hlavolam, Zhasni.

Anotation

The main task is to create (program) a game runtime environment with simplified handicapped users accessible controlling. The supporting application is therefore controllable even with a single input (switch), but also allows a more comfortable keyboard or mouse control without switching. The entire application is optimized to the minimum number of necessary operations.

Simple games and puzzles described by a unified set of XML files and graphical representations enter the environment. The main file is a set of rules which operates the play process. The proposed system of rules ensures that in each round are offered only valid moves, making their selection easier.

The rules are evaluated after each move with a special module with a help of a set of regular expressions. Syntax of the rules is original and is based on C# language. Scheme of the rules is compatible with more than 70 games and puzzles, of which 12 were implemented.

Keywords

Framework – runtime engine – game – simple game – logic game – board game – matrix game – puzzle – rule – simple control – single-switch control – interface – parsing – XML – C# – .NET

Games and puzzles: Maze, Knight's tour, Sam Loyd's Fifteen, Back from the Klondike, Netwalk, Nonograms/Griddlers/Paint by Numbers, Concentration/Memory/Matches, Sokoban, Sudoku, BlackOut, Makeover, Lights Out.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 12.5.2016 .....

.....

podpis

Poděkování

Rád bych zde poděkoval panu Ing. Petru Novákovi za trpělivé vedení diplomové práce, za jeho cenné rady i podnětné otázky.

Děkuji též svým rodičům a nejbližším, kteří mě podporovali nejen během psaní této práce, ale i po celou dobu mého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

Seznam schémat

Seznam diagramů

Seznam zkratek

Zkratka	Zkracuje	*Překlad/Význam*
A	Action	*poloha (souřadnice) dalšího tahu*
GDL	Game Description Language	*formální jazyk popisující diskrétní hry s úplnou informací*
GGP	General Game Playing	*systém hraní her zadaných formálním popisem pravidel*
GUI	Graphical User Interface	*grafické uživatelské rozhraní*
P	Position	*stávající poloha (souřadnice) na herní ploše*
RV	regulární výraz	
XML	Extensible Markup Language	*rozšiřitelný značkovací jazyk*
XNA	XNA's Not Acronymed	*podpůrné prostředí fy Microsoft® na vývoj her*

Typografická konvence

V textu budou některé jeho části uvedeny jiným stylem písma pro lepší orientaci. Zde je uveden přehled základních použitých stylů s vysvětlením jejich významu.

Odsazený bezpatkový font	základní text
Tučný řez	významný pojem
(Kurzíva v závorkách)	většinou anglický pojem
Kurzíva	jiné zvýraznění
Kurzíva šedé barvy	poznámka
Tučný řez písma modré barvy menší velikosti	návěstí titulku obrázku, tabulky, grafu, schématu nebo diagramu
Tučný řez písma kurzívou modré barvy menší velikosti	jméno obrázku, tabulky, grafu, schématu nebo diagramu
Tučný řez písma kurzívou modré barvy	odkaz na jméno nebo číslo kapitoly
TUČNÉ KAPITÁLKY	nadpis logického celku
`Neproporcionální písmo`	jména souborů a adresářů
`Neprop. písmo modré barvy`	název datového typu
`Neprop. písmo s růžovým podbarvením`	příklad nesprávně zapsaného pravidla nebo „breakující“ pravidlo (v Příloze A)
`Neprop. písmo se zeleným podbarvením`	příklad správně zapsaného zdrojového kódu (pravidla a XML)
`Neprop. písmo se žlutým podbarvením`	regulární výraz
`Neprop. písmo s šedým podbarvením`	zdrojový kód (C#)
`Neprop. písmo světle modré barvy`	číslo řádku zdrojového kódu (XML, C# a regulární výrazy)
Písmo s růžovým podbarvením	vysvětlující text ke zdrojovému kódu (XML a C#)
Patkové písmo kurzívou	pojem ze zdrojového kódu (název třídy, funkce, proměnné)
Tučný řez písma s šedým podbarvením	řetězcová konstanta
Ohraničený text s šedým podbarvením	tlačítko aplikace nebo klávesa
Podtržený text modré barvy	hypertextový odkaz

Poznámka k jazyku

V dalším textu se často operuje s anglickými výrazy, názvy XML tagů a zdrojový kód jsou v angličtině kompletně. Angličtina byla zvolena pro případ budoucí prezentace práce v cizím jazyce, především ale pro mnohdy větší výstižnost pojmů. Věřím, že pro čtenáře to nebude překážkou a věci bude jen ku prospěchu. Běžného uživatele se toto však netýká; aplikace, soubor nápovědy, popis her i slovní popis pravidel atd. jsou v češtině.

„Hra je radost. Učení při hře jest radostné učení.“
Jan Amos Komenský

1 Úvod

Má-li zdravý člověk dlouhou chvíli, má kolem sebe většinou nepřeberně možností, jak si ji zkrátit. Ať už je to povyražení s rodinou či přáteli, krátká procházka, návštěva koncertu či jiného představení, čtení knih a časopisů, poslech hudby, zhlédnutí filmu v kině, hraní deskových a karetních her, skládání puzzle, … Kvůli většině těchto činností navíc v dnešní době nemusí ani „vytáhnou paty z domova“, především díky **internetu**; přátele potkáváme na *Facebooku*, záznamy koncertů jsou k vidění na *YouTube*, noviny a časopisy mají svoje internetová vydání, televize i rádia vysílají online, vybranou hudbu a filmy si můžeme (konečně i legálně) stáhnout do svého počítače, o množství videoher ani nemluvě.

Lidé handicapovaní mají možností o poznání méně, i když samozřejmě přibývá míst, kde se již v návrhu počítá s bezbariérovým přístupem. Mají-li navíc „štěstí v neštěstí“ a mohou bez omezení používat počítač, otevírá se jim v podstatě stejná škála virtuálních požitků, jako pro zdravé jedince. Nedovoluje-li stupeň postižení používání počítače klasickou cestou, jsou zde ještě alternativní metody. Ty mají svá pro i proti, je ale dobře, že vůbec nějaké již existují a další vznikají.

Hovořit o příjemném trávení volného času u nejtěžších forem postižení (kvadruplegie, locked-in syndrom) se již téměř nedá. Tito lidé vyžadují stálou odbornou péči a jsou na ni již doživotně odkázáni. Jakékoli vytržení z každodenní rutiny je tak pro ně mimořádně přínosné. Ocení každou pomůcku či zařízení, které je činí méně závislými na cizí péči a dávající jim moc nad svým bezprostředním okolím. V prvé řadě jde především o komunikaci s blízkým okolím a elektronický kontakt na dálku, dále o volný pohyb pomocí elektrického vozíku, ovládání domácích spotřebičů a až po zajištění těchto bazálních potřeb je možné uvažovat i o způsobu trávení volného času.

Jelikož zde nepřipadají v úvahu jakékoli pohybové aktivity, jedinou možností zůstávají elektronické volnočasové aplikace s přizpůsobeným zjednodušeným ovládáním. Může jimi být přehrávač hudby a videa, prohlížeč fotografií a webových stránek, e-mailový klient, malování a další základní aplikace. Bonusem pak může být sada her přizpůsobených omezeným možnostem ovládání. Navrhnout a implementovat takovou sadu her a podpůrné prostředí bude hlavním cílem této práce.

Prostředí bude úzce svázané s širším projektem nazvaným **Inteligentní elektrický vozík** probíhajícího na *Katedře kybernetiky FEL ČVUT v Praze* ve spolupráci skupin *Nature Inspired Technologies Group* a *Inteligentní a mobilní robotiky*. Jeho cílem je postupně implementovat výše zmíněné potřeby a aplikace a sdružit je do souhrnného systému ovládajícího nejen samotný vozík, ale i některé domácí spotřebiče atp. To za pomoci komplexní softwarové výbavy, jejíž součástí bude i navržená a dále popisovaná herní aplikace ze zamýšleného souboru aplikací pro trávení volného času.

2 Současný stav

Zde se seznámíme s problematikou a podíváme se, s jakými řešeními se lze dnes v jejím rámci setkat. Naznačíme také jejich vhodnost pro náš konkrétní problém.

2.1 Elektronické hry pro handicapované

Pokud hovoříme o handicapovaných osobách, je třeba rozlišit, o jaký druh a stupeň postižení se u nich jedná. Závažnost postižení totiž určuje možné způsoby ovládání herního zařízení, potažmo hry.

- Sluchové postižení
- Zrakové postižení
- Inteligenční deficit
- Postižení pohybového aparátu[1]
    - Paraplegie [léze v segmentu Th2 – S5]
    - Kvadruplegie / Tetraplegie [léze v segmentu C4 – Th1]
    - Pentaplegie [Kvadruplegie a nefunkční brániční dýchání]
    - Locked-in syndrom [Kvadruplegie s neschopností mluvit; sluch, zrak i vnímání zachováno]

Jelikož procento výskytu handicapovaných ve společnosti není vysoké, není zde ani dostatečná poptávka, kterou by se zabývaly společnosti vydávající herní tituly. Přesto však vznikají převážně nadšenecké servery a komunity, které sdružují hráče i vývojáře takových her a paleta nabízených her se tak pomalu rozšiřuje. Dalším okruhem lidí, kteří se seriózně problematikou zabývají, jsou různé univerzitní skupiny.

Je nasnadě, že co do množství titulů hry pro handicapované nejspíš nikdy nedoženou komerční sféru, rozhodující je zde však kvalita nad kvantitou.

Při ztrátě pohyblivosti pouze dolních končetin (**paraplegie**) nevyplývají pro hráče žádná vážná omezení. Počítač i jiná herní zařízení mohou ovládat klasickou cestou bez větších obtíží.

Dalšími z handicapovaných, kteří se obejdou při ovládání počítače bez zvláštních omezení, jsou **neslyšící**. Ztráta sluchu není při převažující vizuální stránce dnešních moderních videoher většinou překážkou. Mohou tedy hrát obdobné hry jako zdraví hráči např. se zapnutými titulky, které dnes již různé výpravné tituly obsahují.

Další skupiny handicapů již vyžadují zvláštní přístup. Hry pro **zrakově postižené** jsou specifické pochopitelně absencí grafického rozhraní. Jedná se většinou o hry na specializovaných zařízeních s hmatovým rozhraním. Cílem her je pak např.

přeuspořádat hrací pole podle pravidel nebo rychle reagovat na jeho změny. Anebo jde o hry, které mohou mít i vizuální reprezentaci, která je hlasovým výstupem hráči předávána. To pak umožňuje hraní i složitějších her, jako např. šachy. Aby hráč nemusel v paměti držet kompletní informaci o stavu hry a rozmístění kamenů, může se na ně kdykoli zeptat.

Podle stupně **mentální retardace** můžeme hovořit buď o téměř neznatelných rozdílech, nebo naopak o úplném vyloučení možnosti elektronického hraní. U těchto typů postižení připadají v úvahu spíše fyzické hry a cvičení, která jsou pro rozvoj jedince příhodnější.

Pokud stupeň **pohybového postižení** nedovoluje používání PC skrze standardní periferie (myš, klávesnice, joystick apod.), přichází na řadu nejprve nějak modifikovaná zařízení nebo pak už přímo specializovaná vstupní rozhraní mezi uživatelem a PC.

Kromě počítače jako herního zařízení jsou zde také opět speciální dedikované herní systémy pro pohybově postižené. Tyto nástroje poskytují přirozeně větší uživatelský komfort a lepší funkcionalitu, ovšem většinou za mnohem vyšší pořizovací cenu. V našich krajinách jsou navíc téměř nedostupné. Využití klasické osobního počítače s běžnými periferiemi a vstupy (monitor, reproduktory, klávesnice, myš) a vhodně tyto vstupy modifikovat je tedy pro našince zatím nutností.

U nejtěžších forem (kvadruplegie, locked-in syndrom), kdy je od hráče možné získat jen velmi omezenou sadu signálů až pouze jediný, je již nutné zásadně omezit sadu možných her a přizpůsobit jejich ovládání. Právě tato skupina handicapovaných bude pro další práci cílovou.

Do budoucna jsou velká očekávání od zařízení, která by uměla „číst myšlenky“ na základě např. analýzy EEG. Až se jednou plně naplní vize známé zatím jen ze sci-fi, rozdíl mezi takto pohybově handicapovanými a zdravými hráči se rázem vytratí.

Ani tyto pokročilé techniky však zatím neumožňují obousměrnou interakci, kdy je mozek nejen snímán, ale zároveň i ovlivňován. Takováto přímá audiovizuální stimulace mozku by i v případě locked-in syndromu dovolovala hraní nijak neomezené. Prioritním účelem takového zařízení by pochopitelně nebylo hraní, ale nejprve základní zlepšení kvality života.

2.1.1 Single-switch hry

Tento pojem lze do češtiny přeložit jako *jednovstupé hry* nebo *hry ovladatelné jedním tlačítkem*. Právě tato podskupina přístupných her je cílovou pro tuto práci.

Cíl je v těchto hrách většinou velmi prostý[2]. Např. tisknutím klávesy střílet na cizí vesmírné lodě, zatímco hráčova sama náhodně pluje prostorem.

Pokud nás zajímají i hry s nějakými složitějšími pravidly a hlubším významem, je nabízené množství takových her mizivé. Přesto se dá najít několik zdařilých.

Ukázkovým příkladem takové single-switch hry a inspirací pro návrh aplikace je hra v šachy od skupiny *Human-Computer Interaction Laboratory[3]*. Možnosti jejího ovládání jsou velmi široké, od pouhého jediného vstupu až po klasické myší. Podobná funkcionalita bude u navrhované aplikace žádoucí.

2.2 Herní systémy

Nelze se zde nezmínit o dvou pojmech: *Game Description Language* (GDL) a *General Game Playing* (GGP) [4].

GDL je projektem Standfordovy univerzity a jde o mocný nástroj, kterým se dá popsat téměř jakákoli myslitelná hra, přesněji hry „konečné, diskrétní, deterministické hry pro více hráčů s kompletní informací. GDL je jazyk pro popis her, které jsou určeny pro hraní programy umělé inteligence. GDL je rozšířenou variantou Datalogu. Popisuje stav hry jako množinu pravdivých fakt·, která jsou uložena v interní databázi. Přechody mezi stavy jsou realizovány logickými pravidly, která definují, co bude platit v příštím kole v závislosti na tom, co platí v tomto kole a tazích hráčů·. Rovněž obsahuje logická pravidla pro rozlišení počátečního stavu hry, konečných stav· a stav·, kdy daný hráč je vítěz nebo poražený.“[5]

GGP jsou pak systémy na hraní her zapsaných v GDL. O nejlepší se každoročně dokonce vypisuje soutěž.

2.3 Nástroje

Příkladem konkrétního nástroje na vytváření her je např. podpůrné prostředí XNA[6] společnosti Microsoft®. Nejde přímo o vizuální editor, ale o doplněk, který se doinstaluje do vývojového prostředí Microsoft® Visual Studio®[b] a které pak poskytuje množství užitečných tříd.

3 Vymezení cílů

Herní běhové prostředí, též jen *běhové prostředí* či *prostředí* je abstraktní pojem, jehož reálným odrazem je implementovaná **herní aplikace** či též jen *aplikace* nebo *program*. Tato aplikace obsahuje navíc i podpůrné prostředky pro správu uživatelů, souborů apod. Pojmy prostředí a aplikace jsou tedy provázané a někdy i zaměnitelné.

Typickým cílovým uživatelem aplikace bude handicapovaný hráč, ovšem nikoli bezpodmínečně. Z jeho pohledu pak plynou požadavky na prostředí i typ her.

3.1 Prostředí

Jasným požadavkem je co nejjednodušší ovladatelnost her, a to například i jediným vstupním signálem (tlačítkem). Tímto jediným vstupem může být např. stisk definované klávesy na běžné klávesnici. Prostředí by ale zároveň nemělo být vázáno na nějaký specifický hardware, jakým klávesnice je. Mělo by umět pracovat i s libovolným externím zařízením poskytujícím jediný signál generovaný např. záklonem hlavy, mrknutím (systém I4Control®), pohybem prstu ruky (projekt **Senzorické rukavice**) a dalšími alternativami vyvíjenými nejen v *Gerstnerově laboratoři* na Katedře kybernetiky.

Zároveň je potřeba se všemi těmito rozdílnými vstupy v prostředí zacházet rovnocenně. Proto bude výstup každého zařízení včetně kláves klávesnice přemapován na stisk virtuální interní klávesy, se kterou již bude aplikace dále nakládat. Toto řešení je univerzálnější a navíc umožňuje ovládání her nezávisle na právě připojeném zařízení. Hraní je tak možné skrze klávesnici, myš, speciální hardware nebo i jejich kombinace. To vše bez nutnosti změny nastavení v aplikaci.

Jediný výstup (signál) od uživatele je samozřejmě značně limitující, proto je aplikací nebo řídicím systémem generován speciální přidaný výstup (signál), tzv. **Next**, neboli **Další**. Jeho generování probíhá s nastavitelným intervalem neustále a nezávisle na aktuálním stavu aplikace. Tento vstup pak pomáhá uživateli snadněji vybrat vždy další volbu podle kontextu, v našem případě například další možný tah.

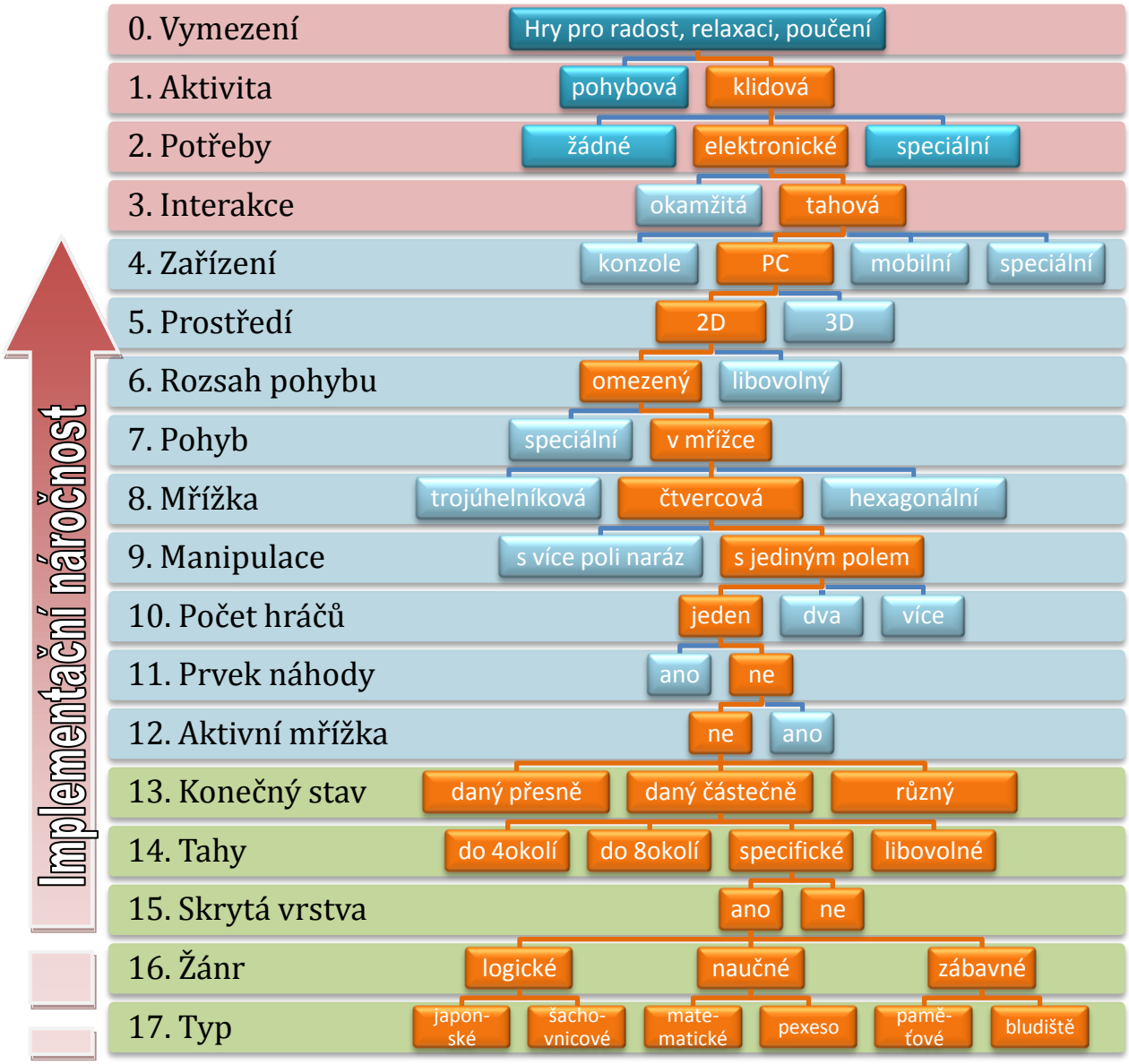
Kromě Nextu, který vybírá po jedné položce, se v návrhu projektu do budoucna počítá i s používáním dalších dvou speciálních signálů **NextRow** (další řádek) a **NextColumn** (další sloupec). Právě v případě maticových her by tyto dva vstupy mohly usnadnit výběr tahu, neboť u rozlehlejších her už může být výběr po jedné položce zdlouhavý. Zatím se však v návrhu aplikace s touto možností nepočítá.

3.2 Hry

Pod pojmem *hra* si lze představit přešírokou škálu činností od hokeje až po piškvorky. I pojem z titulku práce *jednoduché (logické) hry* je natolik obecný, že jej pro naše potřeby více vymezíme.

Níže uvedené Schéma 1 si neklade za cíl absolutní úplnost či přesnost a i pořadí dělení her nevychází z žádných konkrétních ludologických[1] poznatků. Hierarchie odráží potřebu klasifikovat hry z pohledu jejich implementace pro handicapovaného hráče. Pořadí je voleno podle subjektivního pocitu snižující se implementační náročnosti, kdy každá vyšší vrstva vyžaduje větší programátorské úsilí, znalosti a prostředky.

Schéma 1: ***Hierarchie klasifikace her***



U každé vrstvy jsou oranžově označeny kategorie her, do kterých spadají realizovatelné hry.

[1] Ludologie je mladý vědní obor (Gonzalo Frasca, 1999) zabývající se hrami a jejich rolemi ve společnosti.

Ve schématu jsou dále vrstvy podbarvením rozděleny do tří skupin:

- U vrstev v zelené spodní skupině byly implementovány všechny jejich kategorie a oranžová propojení mezi nimi již nejsou směrodatná.
- Červeně jsou kategorie, které již nelze zahrnout, pokud chceme zachovat hratelnost i handicapovanými osobami. Nejvyšší tři pak vycházejí ze samé podstaty věci, kdy hledáme elektronickou formu zábavy.
- Mezi těmito dvěma vrstvami je široký prostor (modře) pro rozšíření aplikace o další typy her.

JEDNOTLIVÉ VRSTVY A KATEGORIE

0. **Vymezení.** Hra se dá chápat velice obecně, je ale zřejmé, že v tuto chvíli nás zajímají hry za účelem zpříjemnění dlouhé chvíle, rozptýlení, obveselení či rozvíjení psychické kondice apod. a nikoli matematická teorie her nebo společenské hry, kdy každý jedinec hraje ve společnosti různé sociální role.
1. **Aktivita.** U některých her je třeba zapojit i fyzickou činnost, jiné můžeme hrát takříkajíc od stolu. Mezi první spadají např. různé sportovní hry, videohry pak patří k druhým, ačkoli dnes díky 3D simulacím a konzoli Wii™ se i první pomalu přesouvají z venkovního prostředí do obývacích pokojů.
2. **Potřeby.** Hry, ke kterým není zapotřebí více než hráč samotný, je mnoho. Jsou to ku příkladu všechny slovní hry. K jiným potřebujeme tužku a papír, karty, hrací kostky, hrací desky a plány atd. To jsou hry s potřebami zde označenými jako *speciální*. Velkou rodinou jsou pak *elektronické* videohry, i když většina her z prvních dvou zmiňovaných skupin má již svoji elektronickou verzi, takže dnes není problém si mariáš zahrát s přáteli online z domova nebo sehrát partii šachu proti počítači.
3. **Interakce.** Videohry pro naše potřeby rozdělíme pro začátek na *tahové* (statické) a ty, které vyžadují *okamžitou* interakci (dynamické). Mezi ty druhé patří různé akční tituly, potažmo všechny, u nichž je rozhodující rychlost odezvy hráče na aktuální herní scénu. Neboli hry, u kterých dochází s časem samočinně ke změně stavu hry (obsahují dynamické prvky). Tyto všechny jsou bohužel pro handicapované při omezené rychlosti fyzického vstupu většinou nedosažitelné.

   Vyloučením těchto „akčních“ her je zajištěno, že každý tah lze libovolně dlouho promýšlet bez rizika, že se během nečinnosti změní stav hry v náš neprospěch. V případě handicapovaných hráčů nemusí jít pouze o rozvahu nad tahem, ale také o možné obtíže spojené s uskutečněním samotného tahu. Proto je tato vlastnost klíčovou a zároveň bohužel limitující. Překonání této hranice snad v budoucnu přinese ovládání myšlenkou například na

základě analýzy EEG. V dnešní době jsou však podobná zařízení[7] ještě stále příliš nepřesná, drahá a mají nízkou rychlost odezvy i procento úspěšnosti. Celkový užitek a funkcionalita jednoduše ještě nejsou srovnatelné s klasickými periferiemi.

4. **Zařízení.** Hraní videoher bylo kdysi doménou speciálních *konzolí* připojených k televizi. Později přišli ke slovu osobní *počítače*. Dnes jsou oba přístupy prakticky vyrovnané a přibyla k nim platforma her pro *mobilní* telefony a zařízení. Pro ni můžeme předchůdce hledat v různých „digihrách“ a později kapesní konzoli GameBoy™. *Speciálními* jsou pak zde myšlena zařízení dedikovaná hraní většinou jen jedné konkrétní hry, jako např. výherní automaty nebo ony „digihry“. Mohou jimi být ale i specializované herní systémy pro handicapované.

   Zde se zaměříme na aplikaci pouze pro platformu PC s operačním systémem Microsoft Windows, jenž je dnes nejrozšířenější.
5. **Prostředí.** *Trojrozměrné* prostředí nabízí živější zážitek ze hry, někdy ovšem na úkor přehlednosti. Aplikace proto pracuje pouze s *2D* reprezentací her.
6. **Rozsah pohybu.** Zde ve smyslu dalšího tahu/kroku. Jelikož pracujeme s digitální technikou, i tzv. neomezený pohyb bude vždy diskrétní s konečným počtem možností. Jde tedy o poměr počtu možných tahů/kroků těchto dvou skupin her. Příklad hry s *neomezeným* rozsahem je prosté malování na plátno, hrou s *omezeným* rozsahem jsou například šachy. Jedna figura je pochopitelně reprezentovaná také v tomto smyslu neomezenou množinou pixelů, výběr každého z nich (např. myší) však má jednotný efekt.
7. **Pohyb.** Známe mnoho především deskových her, které využívají *speciální* hrací plochy, po nichž je pohyb hráče a jeho kamenů vymezen. Např. Člověče, nezlob se!, Monopoly®, Žížalky. Na druhé straně máme hry, u nichž je pohyb možný po celé, často nekonečné, ploše. Typickým příkladem jsou piškvorky a další hry se čtverečkovaným papírem. Nutno poznamenat, že i speciální hrací desky lze „napasovat“ do určité *sítě* s tím, že přebytečná pole jsou pak neaktivní a graficky odlišená.
8. **Mřížka.** Tento rastr může být tří základních pravidelných typů. *Trojúhelníkový* (např. GIPF[8]), *hexagonální* - šesterečný (např. Hexxagon) a nejjednodušší *čtvercový*. Kromě těchto se lze setkat i s dalšími exotickými rastry, některé z nich uvádí[9]. V diagramu již ale uvedené nejsou.
9. **Manipulace.** Typickým příkladem manipulace *s více kostičkami* v jednom tahu *naráz* je hra Tetris nebo Blokus. *S jedním polem* zachází např. Pexeso a Dáma, byť pak může ovlivnit i jiná pole (přeskočit a odebrat kámen).

10. **Počet hráčů.** Herní interakce zdravého a handicapovaného hráče vyžaduje od prvního z nich jistou dávku trpělivosti. Skutečný protihráč však může být nahrazen počítačem s umělou inteligencí. Hry a hlavolamy pro *jednoho* hráče jsou přirozeně méně atraktivní, ale i mezi nimi se najde mnoho zajímavých titulů. I při nich je hráč poháněn k dosažení vyššího skóre, lepšího konečného času, menšího počtu tahů atd.
11. **Prvek náhody.** Nahodilost může být potřebná buď pouze před začátkem hry, kdy generuje novou úroveň (Pexeso, Sudoku, Loydova patnáctka), a/nebo i během hry. Typicky např. hod kostkou před dalším tahem (Člověče, nezlob se!), losování písmene z pytlíku (Scrabble), další karta z balíku atp. Pokud se náhoda vyloučí, je zaručena existence shodného optimálního řešení při každém opakování hry. Což po několika odehráních vyústí v zapamatování vítězné kombinace tahů a tedy i pokles zábavnosti hry. Naopak tato vlastnost umožňuje snadné vyhodnocení úspěšnosti uživatele s počtem odehraných opakování úrovně.
12. **Aktivní mřížka.** Vlastnosti stěny/hranice/přechodu mezi poli mohou u některých her mít svoji roli. Příkladem je hra Theseus[10] či Židi[11]. U realizovaných her tato možnost není nebo je emulována speciální kostičkou, např. u Bludiště.
13. **Konečný stav.** Hra může mít buď přesně definovaný konečný stav známý ještě před započetím hry, může jich mít více opět přesně známých anebo je konečnost daná neúplně, kdy se pouze některé pozice musí shodovat s konečným stavem.
14. **Tahy.** První tři uvedené kategorie se vážou na hry, u nichž má smysl definovat aktuální polohu/pozici. Pak můžeme rozlišit tahy z této pozice do *4-okolí* (Bludiště), do *8-okolí* (Návrat z Klondiku) nebo *specifické* (tahy jezdcem na šachovnici). V ostatních případech (*libovolné*) je k dispozici tah na všechna platná pole (Pexeso, Sudoku).
15. **Skrytá vrstva.** Přirozeným cílem hraní všech her je vyhrát. K vítězství vede pouze omezená kombinace kroků a úkolem hráče je ji odhalit. U většiny her je tato kombinace při důkladné analýze zjistitelná v libovolném okamžiku hry (např. Loydova patnáctka, Bludiště). Některé hry však obsahují *skrytou* informaci (vrstvu), kterou je třeba během hry odhalit a získat tak vítěznou kombinaci tahů (Pexeso).
16. **Žánr** a 17. **Typ.** Tato dvě dělení jsou již značně subjektivní, jedna hra může být na pomezí žánrů a být zároveň vícetypová. Přesto své místo zde tato dělení mají, neboť u různých typů her bývají pravidla značně odlišná.

Jistě by se dala najít mnohá další hlediska, podle kterých se hry dají dále dělit, jako jsou potřebný herní čas, autor, obtížnost,… Pro implementaci již sice nemají příliš význam, mohou ale být důležitá pro hráče při rozhodování, kterou hru ze sady zvolí. Proto budou některá tato kriteria obsažená v souboru popisu hry. Pro zájemce o problematiku lze doporučit[12].

Mohlo by se zdát, že her, které splňují takto striktní požadavky, nebude mnoho. Přesto opak je pravdou. Celkem jim vyhovuje bezmála 80 her, z nichž několik bude pro ukázku implementováno. Jejich kompletní přehled uvádí Tabulka 7 v Příloze B.

3.3 Herní systém

GDL sice obsahuje pravidla pro určení, zda je daný tah legální či nikoliv, ovšem až po jejich provedení. Především z tohoto důvodu, ale i proto, že GDL je čistě deklarativní jazyk bez vazby na pevnou datovou strukturu, bude definován systém původní, který však z těchto obecných myšlenek bude vycházet.

3.4 Nástroje

XNA sice poskytuje pokročilé nástroje usnadňující návrh her, ale vždy právě jedné konkrétní. Není určené na vytváření prostředí, v kterém by šly hrát různé hry s definicí oddělenou od zdrojového kódu. Více her v jedné aplikaci je sice podporováno, ale opět by každá hra musela být naprogramována zvlášť, takže XNA se pro účely práce příliš nehodí.

Aplikace tedy bude naprogramovaná v jazyce C#[13] na platformě .NET[14] s využitím vývojového prostředí Microsoft® Visual Studio® 2008. Jde o moderní programovací jazyk (aktuálně již ve své 4. verzi) s širokou technickou podporou i vývojářskou komunitou. V tomto jazyce jsou psány i zbylé komponenty celé platformy projektu Inteligentního vozíku, a tak i z důvodu kompatibility a snadné orientace v kódu ostatními vývojáři v budoucnu byl zvolen právě jazyk C#.

Aplikace bude pracovat s mnoha datovými soubory. Aby mohla s daty později pracovat i jiná aplikace nebo být čtena a editována přímo člověkem, byl pro všechny datové soubory zvolen otevřený samopopisný formát XML. Tedy s jedinou pochopitelnou výjimkou, kterou jsou grafická data (obrázky). XML zápis je sice objemnější, ale kromě čitelnosti člověkem má z hlediska implementace další nespornou výhodu snadného ukládání a načítání celých tříd přes tzv. serializátor podporovaný v .NET prostředí.

4 Návrh herního systému

4.1 Definice pojmů

Přirozeným **cílem** každé hry je **vyhrát** a musí mít tedy **konec**. Ten může u většiny her nastat až po konkrétním **minimálním počtu** tahů provedených z **výchozího / počátečního** stavu. **Stavem** je myšleno rozložení **kamenů / kostiček / polí(ček)** na **hrací ploše**. Na druhou stranu u všech her při nenáhodném opakování nesprávných tahů nevede k cíli ani nekonečný počet tahů, např. u hry Zhasni, kdy opakovaně volíme stejné pole.

Samotným pojem **tah** je definován jako zvolení jednoho pole, které stvrdíme akčním příkazem (viz dále). Všechny tahy (volba každého pole) jsou vždy **proveditelné**, ale jen některé z nich jsou v daný okamžik **platné / validní / legitimní**. Příklad **neplatného** tahu je např. tažení jezdcem z jeho **pozice** pouze o jedno pole doprava. Sama platnost ještě nezaručuje správnost tahu. **Správné / dobré** tahy vedou k cíly nejrychleji a je úkolem hráče je odhalit. **Nesprávné / špatné** tahy sice hráč může zvolit, ale zhorší jimi svoji situaci - oddálí se řešení.

Výběr tahu a *volba tahu* jsou zde rozlišeny jako dva různé pojmy. Tah se nejprve musí **vybrat** z možností/množiny platných tahů a následně **zvolit**. Právě možné způsoby výběru tahu jsou pro handicapovaného hráče klíčové. Aby se tento výběr hráči usnadnil, musí mu být nabízena pouze **množina platných tahů**, které tedy musí být pro dané kolo **známé dopředu**. Nabízet hráči pouze platné tahy je úkolem hry resp. jejích **(herních) pravidel**.

Provedení tahu je potvrzení volby a její předání ke zpracování těmto herním pravidlům. Provedením tahu se ukončí jedno **kolo** hry.

U každé hry musí být její **cíl / úkol** jasně formulován **slovně popsanými pravidly** spolu s postupem, jak ho hráč má dosáhnout. Skutečný **konečný stav** hry (rozložení kamenů) nemusí být však nutně pouze jeden a nemusí být ani zadaný zcela přesně. V ***kapitole 3.2 Hry*** jsme se ale omezily na hry, u kterých jsou tyto konečné stavy známy už před začátkem hry. Toho využijeme ke zjednodušenému vyhodnocení konce hry, kdy aktuální stav po konci kola porovnáme s konečným stavem nebo stavy.

4.2 Návrh struktur

Máme tedy tři základní stavební prvky, které definují každou hru. Počáteční stav, cílový stav a množinu platných tahů pro dané kolo. Všechny tři budeme reprezentovat maticemi shodných rozměrů. Konkrétní velikost se hry od hry může lišit. Počáteční stav je zadaný pevně, stejně jako prvotní množina platných tahů. Jak hra postupuje, mění se stav hry i možné platné tahy, obojí podle pravidel hry. Konečný stav je u všech

realizovaných her neměnný, ale pro případ, že by jej nějaká hra modifikovala pravidly, je tato možnost ponechána.

Kromě těchto matic je u některých her definovatelná aktuální pozice (stávající poloha) a spolu s ní **následující poloha**, kterou volí hráč. Pak i jejich souřadnice musí být součástí prvotního rozložení hry. Pokud pro hru smysl nemají, definujeme u nich nulovou hodnotu.

4.3 Datová reprezentace

Stavová matice (dále označovaná jako **STATE**) a matice cílového stavu (dále jako **FINAL**) v sobě obsahují řetězce (typ `string`) odkazující na konkrétný kámen ze sady všech kamenů vyskytující se ve hře. Podle obsahu STATE se vykresluje a mění herní plocha. Matice platných tahů (dále jako **VALID**) obsahuje pravdivostní informaci, zda je tah na dané pozici v tomto kole platným (typ `bool`).

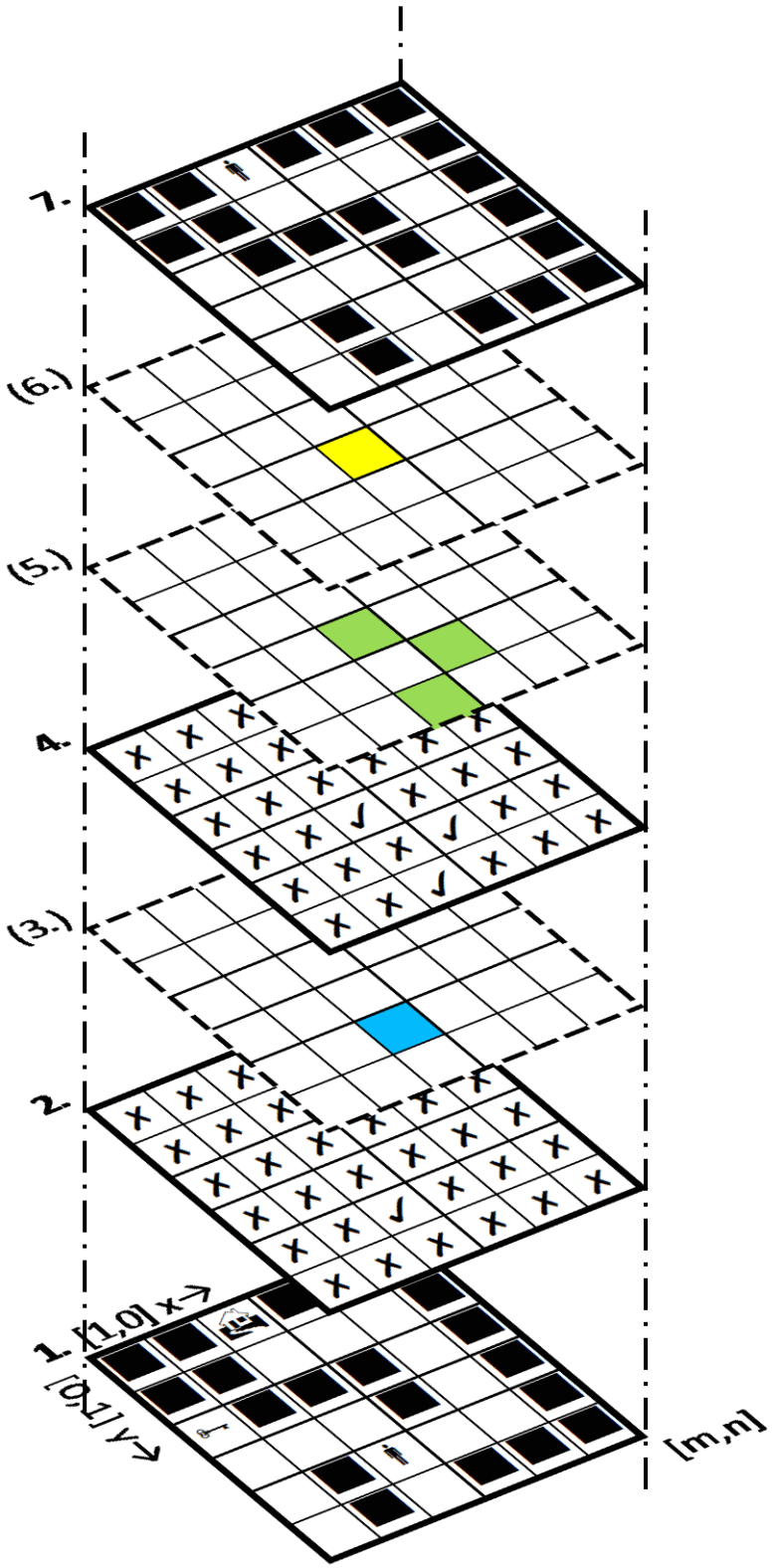
Stávající pozice (dále jako **P**) a právě zvolený tah/následující poloha (dále jako **A** z anglického *Action*) jsou souřadnice odkazující do matic. Mají svoji x-ovou a y-ovou složku. Pro snazší orientaci a zpracování jsou zavedeny pro složky A i P dvě samostatné proměnné **Ax**, **Ay** a **Px**, **Py**. Ty jsou pak skutečně zapsané v souboru s počátečním rozložením.

Jelikož některé hry vyžadují navíc další vlastní pomocné položky, které nemusí být jen číselného typu (`int`), jsou všechny položky včetně Ax, Ay, Px a Py obecného datového typu `object` a jejich skutečný typ je u každé explicitně uvedený v zápisu. Všechny tyto datové struktury ukazuje následující souhrnný Obrázek 1.

1. vrstva na něm odpovídá matici STATE, 4. matici VALID a 7. matici FINAL. Kromě těchto tří základních matic je zde ještě 2. vrstva s označenou aktuální polohou P. Ta, jak již jsme řekli, není ve skutečnosti reprezentovaná maticí, nýbrž dvěma položkami Px a Py. Zde jsou však tyto převedeny na konkrétní odkaz kvůli korespondenci se třemi hlavními maticemi.

Vrstvy 3, 5 a 6 jsou pak uživatelem volitelné vizualizace. Modře (3. vrstva) může být na hrací ploše zvýrazněna **aktuální poloha**, zeleně (5. vrstva) všechny aktuálně **platné tahy** a žlutě (6. vrstva) je zamýšlený tah neboli **fokus**. Ten se přesouvá spolu s kurzorem myši či při stisku směrové klávesy (viz ***kapitola 6.4.2***). Nejde o položku spjatou s pravidly hry, dává pouze hráči najevo, jaký tah bude zvolen, pokud v daný okamžik stiskne akční klávesu (Enter) nebo stiskne tlačítko myši apod. Není proto třeba ji uchovávat nebo na začátku hry definovat spolu s ostatními položkami.

Obrázek 1: *Reprezentace stavu hry*



4.4 Průběh hry

Hra začíná načtením základního rozložení, tedy matic a položek. Pokud jde o zcela novou hru (první hraní konkrétní úrovně), vezme se toto rozložení

z předdefinovaného souboru. Později, např. při přerušení hry a jejím obnovení, se tato rozložení berou z uložených souborů vygenerovaných aplikací.

Po každém platném tahu[2] se vykonají všechny akce dle pravidel. Pak je matice STATE porovnána se všemi maticemi FINAL a pokud se alespoň jedna shoduje, znamená to úspěšný konec hry. Pakliže nenastal konec, zjistí se, zda ještě existují nějaké platné tahy pro další kolo, neboť hráč se mohl dostat do situace, kdy dle pravidel již žádný neexistuje. V tom případě je vynucen restart hry (úrovně). Při něm se opět načte původní výchozí rozložení úrovně.

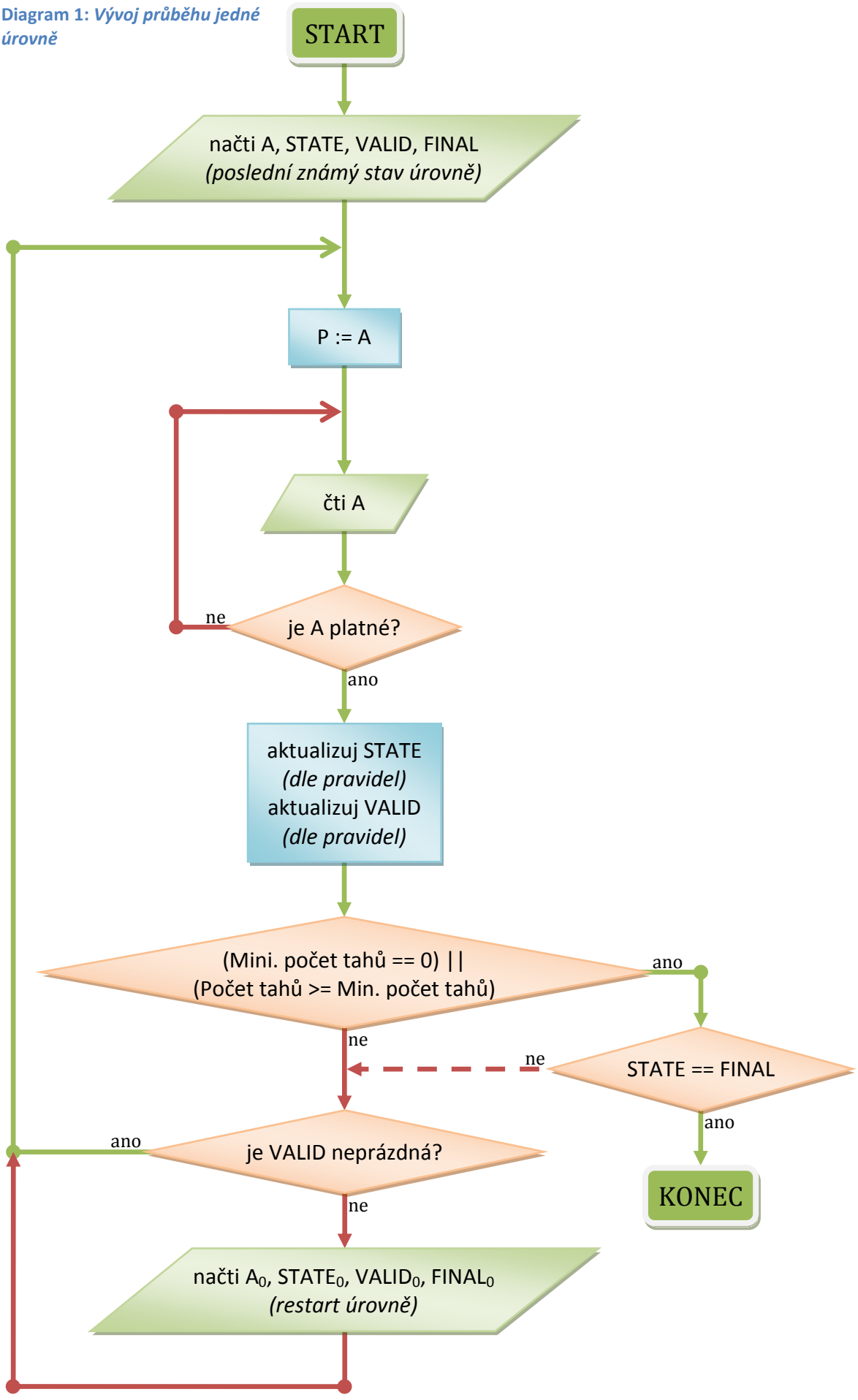
Pokud je u hry definovaný minimální počet tahů vedoucích k řešení, je vyhodnocení konečnosti (porovnání s maticemi FINAL) odloženo až do překročení tohoto počtu z důvodu výpočetní úspory. Tato úspora se pochopitelně projeví jen při optimálním postupu řešení. Při mnohonásobném překročení tohoto minimálního počtu se dokonce může změnit v nevýhodu, neboť čas potřebný pro samu operaci porovnání počtů tahů, která je vlastně navíc, z dlouhodobého hlediska převáží počáteční úsporu. Pokud v definici úrovně informace o minimálním počtu chybí (je nulová), provede se test konečnosti vždy.

Průběh odehrání jedné úrovně i úkony vykonávané po jednotlivých kolech názorně vyjadřuje vývojový diagram na níže (viz Diagram 1).

Z důvodu úspory místa při zachování čitelnosti a přehlednosti byl v diagramu použit nekorektní konstrukt skoku, kdy se po vyhodnocení podmínky STATE==FINAL *skočí do* else *části jiné podmínky (naznačeno přerušovanou čarou). Správně má při nesplnění této podmínky následovat další samostatný blok s vyhodnocením podmínky* je VALID neprázdná? *a následně se obě větve opět spojí. Ve zdrojovém kódu je zapsána tato korektní varianta.*

[2] Řekli jsme sice, že hra musí poskytovat sadu platných tahů, ale uživatel přesto může mít zapnutou volbu vybírání i neplatných tahů (viz ***kapitola 6.4.1*** dále).

Diagram 1: *Vývoj průběhu jedné úrovně*



5 Pravidla

Pravidla patří také k Hernímu systému z předchozí kapitoly, jsou však natolik podstatných rysem práce, že je jim věnována zvláštní kapitola.

Stěžejní pro řízení průběhu hry jsou její správně zapsaná a úplná **pravidla**. Pod tímto pojmem dále nebude myšlen jejich slovní zápis ve větách, který je zobrazen hráči, neboť z něj nikdy nevyplývají všechny události a akce, které po každém tahu nastávají a které musí *herní prostředí* vykonat. Např. triviální zápis pravidla (úkolu) „posuň panáčkem krabici na hvězdičku“ v sobě obsahuje řadu dílčích podmínek a akcí, které jsou hráči skryty. Díky nim však hráč může tento úkol uskutečnit a jsou to právě tyto skryté podmínky a akce, které budou dále souhrnně označované jako **pravidlo**. Při jejich zápisu je potřeba se na hru podívat z té nejnižší úrovně, kdy se manipuluje s jednotlivými kameny, a popsat jimi všechny možné myslitelné situace od začátku do konce hry. Jedním takovým pravidlem může být:

- jediný příkaz (přiřazení)
- podmíněný příkaz, který může zároveň být

	BUĎ:		*NEBO:*
○	s jedinou podmínkou	X	s více podmínkami
○	pouze s *if* částí	X	i s *else* částí
○	s jediným přiřazením	X	s blokem přiřazení
○	***breakující***	X	*nebreakující*

(ukončující aplikování zbylých pravidel při splnění podmínky)

5.1 Záznam

Všechna pravidla jsou uložená v souboru `rules.xml` v adresáři s příslušnou hrou. Kompletní pravidlo (<GameRule>) je rozděleno na tělo (v elementu <Contents>), což je vlastní zápis ve formě řetězce (`string`), a na pravdivostní hodnotu (`bool`) určující, zda je pravidlo tzv. *breakující* (v elementu <Break>). Příklad zápisu pravidla:

```
<GameRule>
  <Contents>
    if (STATE[A.x, A.y] == 01Back)
    {
        Before = 01Back;
        STATE[A.x, A.y] = 01Front;
    }
  </Contents>
  <Break>false</Break>
</GameRule>
```

V dalších částech bude popisován již jen samotný řetězec pravidla. Ten může v zápise obsahovat libovolné množství netisknutelných (bílých) znaků, jako jsou mezery, tabulátory a nové řádky, aby jeho čitelnost i zapsání byly snazší a přehlednější. Při vlastním vyhodnocení jsou tyto znaky ignorovány. Taktéž jsou při něm všechna

písmena převedena na malá, ale v zápise je lze opět dle libosti použít pro lepší orientaci. Např. matice psát velkými písmeny, hodnoty malými.

Ještě jeden znak je z řetězce přej jeho zpracováním vyloučen. Tím znakem je tečka, která v záznamu umožňuje elegantnější rozepsání souřadnic bodů. Je-li v pravidle zapsáno „ A.x = 1 “, bude tento řetězec upraven na „Ax=1“. Z x-ové složky bodu A se tak stane nová samostatná proměnná Ax, se kterou se dále zachází odděleně od obdobné y-ové složky Ay. V definici počátečního rozložení jsou tedy všechny body definované podobnými dvěma proměnnými (položkami).

5.2 Datové typy

Všechna pravidla mohou operovat s proměnnými a hodnotami několika základních typů. Při manipulaci s nimi je třeba dbát na typovou shodu, jinak dojde k výjimce a pravidlo nebude aplikováno. Definice těchto základních typů je shodná jako ve specifikaci jazyka C#, neboť prostředí aplikace s nimi později manipulující je napsáno právě v tomto jazyce. Přehled uvádí následující tabulka:

Identifikátor	České pojmenování	Obor hodnot
`bool`	pravdivostní hodnota	true/false (pravda/nepravda)
`int`	celé číslo	v rozsahu <-2 147 483 648, +2 147 483 647>
`string`	řetězec	řetězec libovolných znaků ze základní sady ANSI téměř neomezené délky[3] *Použití pětice symbolů* **& < > " '** *je z definice XML zakázáno. Pokud se nelze vyhnout použití těchto znaků a dalších speciálních národních znaků, je třeba definovat zvláštní zástupné entity. Více o problematice v [15].*
`object`	objekt	obecný datový typ položek, jejichž skutečný typ se určí až v souboru definice úrovně

Tabulka 1: ***Přípustné datové typy***

Všechny proměnné i hodnoty jsou samozřejmě ze souboru pravidel načteny nejprve jako řetězce. Běhové prostředí však při zpracování pravidel v těchto řetězcích rozpozná identifikátory matic, položek i jmen kostiček atd. a automaticky provede jejich konverzi na příslušný datový typ. Ten bude v dalším textu uveden v závorce za každou položkou, hodnotou, maticí apod.

[3] Záleží na architektuře operačního systému (32b/64b), souborovém systému počítače, dostupné operační paměti a její fragmentaci. Řádově však stamiliony až miliardy znaků.

5.3 Syntax

I vlastní syntax vychází z jazyka C# s mírnými odchylkami popsanými dále. V pravidlech lze manipulovat se všemi maticemi, vždy však právě s jednou položkou na konkrétních souřadnicích. Přiřazení jedné hodnoty celé matici nebo celému jednomu řádku či sloupci není možné, stejně jako není povoleno porovnávání celých matic nebo řádků. Dále může pravidlo obdobně manipulovat se všemi položkami. Manipulací může být přiřazení nebo podmíněné přiřazení. Cykly ani další operace podporovány nejsou.

V zápisu syntaxe budou používány symboly závorek ve zvláštním významu:

<aaa>	identifikátor entity
[aaa|bbb]	výběr právě jedné z hodnot oddělených symbolem ‚|‘
(aaa)	jedno či více opakování části (nutný alespoň jeden výskyt)
{aaa}	nula či více opakování části (zcela volitelná část)

Aby nedošlo k nechtěné záměně, jsou zbylé konstanty (řetězce a symboly) tučně a podbarvením zvýrazněny.

5.3.1 Odkaz do matice

Před dalším popisem je třeba definovat tento pojem, se kterým přiřazení i podmíněné příkazy pracují.

OBECNÁ SYNTAX

<jméno matice>**[**<Index_X> {[**+**|**-**] <Index_X >}**,** <Index_Y> {[**+**|**-**] <Index_Y >}**]**

<jménem matice> může být:

- jedna ze tří povinných matic (STATE | VALID | FINAL)
- jméno dalších matic vlastní konkrétní hře (`string`)

Všechny *<Indexy>* bez rozlišení musí být z těchto možností:

- položka (`int`)
- celé číslo (`int`)

Všechny dále uváděné příklady syntaxe struktur jsou pouze modelové a většinou nejsou ve hrách skutečně použity. Množství reálných příkladů pravidel konkrétních her uvádí Příloha A.

PŘÍKLADY

Odkaz na položku matice STATE na souřadnicích x = 3 a y = 10. V příkladu si všimněte, že oba indexy jsou o jedna menší. To je dáno konvencí jazyka C#, kdy se položky v maticích indexují od nuly. První položka matice má index [0,0] místo nám přirozenějšího [1,1], na což je třeba pamatovat.

```
STATE[2, 9]
```

Odkaz na položku z matice VALID na aktuální pozici:

```
VALID[P.x, P.y]
```

Odkaz do matice STATE na položku, která se nachází za aktuálním tahem v prodlouženém směru, ze kterého táhneme:

```
STATE[A.x + A.x - P.x, A.y + A.y - P.y]
```

Další platný zápis:

```
VALID[A.x - 1 + 8 + 10 - P.y, 6]
```

Neplatný zápis (chybějící index pro Y):

```
STATE[A.x + 5]
```

Neplatný zápis (nesprávné hranaté závorky):

```
STATE<A.x, A.y>
```

Neplatný zápis (za znaménkem nenásleduje hodnota):

```
STATE[A.x - , A.y + ]
```

Neplatný zápis (před první hodnotou je znaménko):

```
STATE[-A.x + 9, +5 + A.y]
```

Neplatný zápis (index odkazem do matice):

```
STATE[STATE[1, 5], P.y]
```

5.3.2 Přiřazení

Obecná syntax jednoduchého přiřazení

<proměnná> = <hodnota>;

Jako *<proměnná>* může figurovat:

- položka (`object`)
- odkaz do matice (`string` | `bool` | `int`)

Jako *<hodnota>* mohou vystupovat:

- položka (`object`)
- odkaz do matice (`string` | `bool` | `int`)
- název kostičky (`string`)
- pravdivostní hodnota (`bool`)
- celé číslo (`int`)
- obecný řetězec (`string`)

Příklady

Nastavení x-ové souřadnice A na hodnotu 7:

```
A.x = 7;
```

Nastavení y-ové souřadnice P na hodnotu z vlastní matice VECTORS s položkami typu `int` na pozici A:

```
P.x = VECTORS[A.x, A.y];
```

Uložení nepravdy do matice VALID o jedna vpravo od aktuální pozice:

```
VALID[P.x + 1, P.y] = false;
```

Záměna kostičky na pozici A v matici STATE za kostičku s identifikátorem Apple:

```
STATE[A.x, A.y] = Apple;
```

Přiřazení kostičky uložené v pomocné položce Temp do matice STATE na pozici o jedna níže, než je současná poloha:

```
STATE[P.x, P.y - 1] = Temp;
```

Mezi tímto a předchozím příkladem na první pohled není rozdíl. Zatímco v prvním je *Apple* názvem kostičky definovaným v souboru `pieces.xml`, v druhém příkladu je *Temp* identifikátorem položky definované v souboru `startingLayout.xml` úrovně.

Záměna kostičky v matice STATE na současné pozici za kostičku, na kterou táhneme:

```
STATE[P.x, P.y] = STATE[A.x, A.y];
```

Uložení řetězce „Peter“ do vlastní položky Name:

```
Name = Peter;
```

OBECNÁ SYNTAX SLOŽENÉHO PŘIŘAZENÍ

<proměnná> **=** <hodnota> ([**+**|**-**] <hodnota>)**;**

<proměnná> zde musí být:

- položka (`int`)
- odkaz do matice (`int`)

A všechny *<hodnoty>* musí být z těchto možností:

- položka (`int`)
- odkaz do matice (`int`)
- celé číslo (`int`)

PŘÍKLADY

Přičtení deseti bodů k položce zaznamenávající skóre:

```
Score = Score + 10;
```

Zmenšení síly zdi o jedna na pozici A, kdy WALLFORCE je vlastní matice hry.

```
WALLFORCE[A.x, A.y] = WALLFORCE[A.x, A.y] - 1;
```

Další platný zápis:

```
Score = 20 - Penalty - Score + 10 + coefficient;
```

5.3.3 Podmíněné výrazy

OBECNÁ SYNTAX JEDNODUCHÉHO PODMÍNĚNÉHO VÝRAZU

(<hodnota> [**==**|**!=**] <hodnota>**)**

Pro *<hodnoty>* platí stejné vymezení jako u *<hodnot>* v jednoduchém přiřazení.

PŘÍKLADY

Vrátí vždy nepravdu (kontradikce):

```
(true != true)
```

Vrátí vždy pravdu (tautologie):

```
(true == true)
```

Použitím tautologie u podmíněného pravidla lze do bloku zapsat více příkazů (viz dále Podmíněný výraz), které by jinak stály ve strukturách samostatně, což zvyšuje čitelnost. Dále také umožňuje bezpodmínečné přerušení vyhodnocení všech zbývajících pravidel v kombinaci s nastavením příznaku break, i když toto použití již nedává příliš smysl.

Vrátí pravdu, pokud je v matici na pozici A uložena kostička s pojmenováním Back:

```
(STATE[A.x, A.y] == Back)
```

Vrátí pravdu, pokud se kostičky vlevo a vpravo od P rovnají:

```
(STATE[P.x - 1, P.y] == STATE[P.x + 1, P.y])
```

Vrátí pravdu, pokud položka Animal neobsahuje kostičku Dog:

```
(Animal != Dog)
```

Další platný zápis:

```
(NUMBERS[A.x + P.x - 5, A.y - CoorY + 5] == 1000])
```

Neplatný zápis (nepovolený operátor):

```
(Score >= 10)
```

Neplatný zápis (neuzavřená závorka):

```
(STATE[A.x, A.y] == Path
```

Neplatný zápis (porovnání nestejných typů):

```
(STATE[P.x, P.y] == VALID[A.x, A.y])
```

OBECNÁ SYNTAX SLOŽENÉHO PODMÍNĚNÉHO VÝRAZU

(<jednoduchý podmíněný výraz>([**AND**|**OR**] <jednoduchý podmíněný výraz>)**)**

- Proti jazyku C#, kde je operátor logického součinu (A) zapisován symboly „&&“ a operátor logického součtu (NEBO) symboly „||“, zde byla syntaxe upravena a tyto operátory musí být vyjádřeny slovně anglickými pojmy AND a OR. Je to z toho důvodu, že XML zakazuje u hodnot elementů užití některých speciálních znaků, mezi nimi i symbol ‘&’. Symbol ‘|’ je sice platný, ale aby nebyla syntaxe roztříštěná, není zápis „||“ pro operátor NEBO využit. Nakonec i s ohledem na skutečnost, že výrazy AND a OR jsou většině lidí srozumitelnější.
- Ve složeném pravidle není dovolena kombinace operátorů. Celý výraz musí být zřetězen buď jedním, nebo druhým operátorem.
- Nejsou také dovoleny vnořené pod-podmínky.

PŘÍKLADY

Vrátí pravdu, pokud jsme v tomto kole vybrali znovu ten samý tah (sami sebe):

```
((A.x == P.x) AND (A.y == P.y))
```

Vrátí pravdu, pokud je v položce Car uložen řetězec Opel a hodnota Score je nenulová:

```
((Car == Opel) AND (Score != 0))
```

Vrátí pravdu, pokud je alespoň v jednom poli ve 4-okolí P uložena kostička se jménem Empty:

```
((STATE[P.x + 1, P.y] == Empty) OR (STATE[P.x - 1, P.y]
== Empty) OR (STATE[P.x, P.y + 1] == Empty) OR
(STATE[P.x, P.y - 1] == Empty))
```

Neplatný zápis (chybějící obalující závorky):

```
(STATE[1, 1] == STATE[2, 2]) AND (STATE[2, 1] !=
STATE[1, 2])
```

Neplatný zápis (kombinace operátorů):

```
((STATE[A.x, A.y] == Number1) AND (STATE[A.x - 1, A.y]
== Number2) OR (STATE[A.x, A.y + 5] == Number3))
```

Neplatný zápis (vnořené pod-podmínky):

```
(((A == 1) AND (B != 1)) AND (Word == guitar))
```

Neplatný zápis (kombinace operátorů vnořené pod-podmínky):

```
((STATE[A.x, A.y] != 1) AND ((STATE[A.x, A.y] == 2) OR
(STATE[A.x, A.y] == 3)))
```

5.3.4 Podmíněné příkazy

OBECNÁ SYNTAX PODMÍNĚNÉHO PŘÍKAZU POUZE S HLAVNÍ (IF) VÝKONOVOU ČÁSTÍ

if[<jednoduchý podmíněný výraz>|<složený podmíněný výraz>]
{
([<jednoduché přiřazení>|<složené přiřazení>])
}

- Provedení výkonové části může být podmíněno jednoduchým nebo složeným výrazem. Pokud je celý výraz pravdivý, zkusí se provést všechna přiřazení. Pokud je navíc u tohoto pravidla definován break, přeskočí se všechna další pravidla. Pokud podmínka splněná není, neprovede se nic.
- Proti jazyku C# je zde opět drobný rozdíl, kdy i jediný příkaz (přiřazení) musí být uzavřen do složených závorek.
- Jako příkaz nemůže vystupovat další podmíněný příkaz.

PŘÍKLADY

Pokud je na tahu kostička s číslem 1, zvýší se na 2 (přiřadí kostičku s číslem 2):

```
if (STATE[A.x, A.y] == Number1)
{
    STATE[A.x, A.y] = Number2;
}
```

Byla-li zvolena kostička Bonus, přičte se ke skóre 100 bodů, uloží nový stav do pomocné proměnné a zruší platné tahy v původním 4-okolí.

```
if (STATE[A.x, A.y] == Bonus)
{
    Score = Score + 100;
    Temp = STATE[A.x, A.y];
    VALID[P.x + 1, P.y] = false;
    VALID[P.x - 1, P.y] = false;
    VALID[P.x, P.y + 1] = false;
    VALID[P.x, P.y - 1] = false;
}
```

Neplatný zápis (chybějící složené závorky):

```
if (STATE[P.x, P.y] == Toy)
    Play = true;
```

Neplatný zápis (chybějící středník za přiřazením):

```
if (STATE[10, 20] != STATE[2, 22])
{
    VALID[10, 20] = true
}
```

Neplatný zápis (vnořené podmíněné příkazy):

```
if ((STATE[P.x + 1, P.y] == Path) OR (STATE[P.x - 1,
P.y] == Path))
{
     VALID[P.x, P.y] = false;
     if (STATE[P.x + 1, P.y] == Path)
     {
         STATE[P.x + 1] = true;
     }
     if (STATE[P.x -1 1, P.y] == Path)
     {
         STATE[P.x - 1] = true;
     }
}
```

OBECNÁ SYNTAX PODMÍNĚNÉHO PŘÍKAZU I S VEDLEJŠÍ (ELSE) VÝKONOVOU ČÁSTÍ

if [<jednoduchý podmíněný výraz>|<složený podmíněný výraz>]
{
([<jednoduché přiřazení>|<složené přiřazení>])
}
else
{
([<jednoduché přiřazení>|<složené přiřazení>])
}

- První část je syntakticky shodná s předchozí definicí. Oproti ní je zde však navíc alternativní výkonová část (blok za else), která se provede, jen pokud podmínka není splněná - celý výraz vrací hodnotu false. I zde platí nutnost uzavření do složených závorek.
- Ani zde nemohou být jiné podmíněné příkazy.

PŘÍKLADY

Pokud je v matici STATE na tahu kostička Mouse, ulož místo ní CatFull, jinak na toto místo dej kostičku CatEmpty:

```
if (STATE[A.x, A.y) == Mouse)
{
     STATE[A.x, A.y] = CatFull;
}
else
{
     STATE[A.x, A.y] = CatEmpty;
}
```

Neplatný zápis (chybějící příkazy v else části):

```
if (A.x == 20)
{
     P.x = P.x + 20;
}
else
{
}
```

Neplatný zápis (obsahuje vnořený podmíněný příkaz):

```
if (STATE[A.x, A.y] == Red)
{
     STATE[A.x, A.y] = Yellow;
}
else
{
     if (STATE[A.x, A.y] == Yellow)
     {
         STATE[A.x, A.y] = Off;
     }
     else
     {
         STATE[A.x, A.y] = Red;
     }
}
```

6 Implementace

Z titulu práce i vymezení v ***kapitole 3.2*** vyplývá zaměření na hry „jednoduché“, potažmo „logické“. Jednoduché ve smyslu nikoli snadné k dohrání, ale s jednoduchým ovládáním a pravidly. Slovo „logické“ pak napovídá, že důraz je kladen spíše na hráčův intelekt, než hbitost a postřeh. Z těch požadavků budeme při implementaci vycházet.

Kromě následujících podkapitol implementovanou aplikaci z uživatelského hlediska popisuje vyhotovená uživatelská příručka, která je součástí příloh (***Příloha B***).

6.1 Architektura

Hlavní částí aplikace je běhové prostředí, do kterého vstupuje hra v definovaném formátu, jejíž instance, přesněji instance jednotlivých úrovní, jsou konfrontovány s hráčovými tahy, dokud nenastane konec. Prostředí má jediný výstup, kterým je nové uspořádání herní plochy po každém tahu. To je jednak interpretováno graficky změnou obrazového výstupu (na monitoru), tak i fyzicky zaznamenáno (ukládáno) do souborů pod názvy `<00000001-99999999>.xml` v hráčově podadresáři (`SAVES`).

Hra je definovaná svým slovním popisem (`description.xml`), nastaveními ovládání a vzhledu (`settings.xml`), pravidly (`rules.xml`), kostičkami, které se ve hře objevují (`pieces.xml`) a sadou jejich skutečné grafické podoby (obrázky v podadresáři `PIECES`) a volitelně miniaturou (`thumbnail.*`). Aby bylo co hrát, musí ještě obsahovat sadu úrovní a podadresáři `LEVELS`.

Úroveň je daná svým slovním popisem (`description.xml`) a počátečním rozložením (`startingLayout.xml`). O možném obsahu všech těchto souborů a adresářů pojednává příští kapitola ***Vytvoření nové hry***.

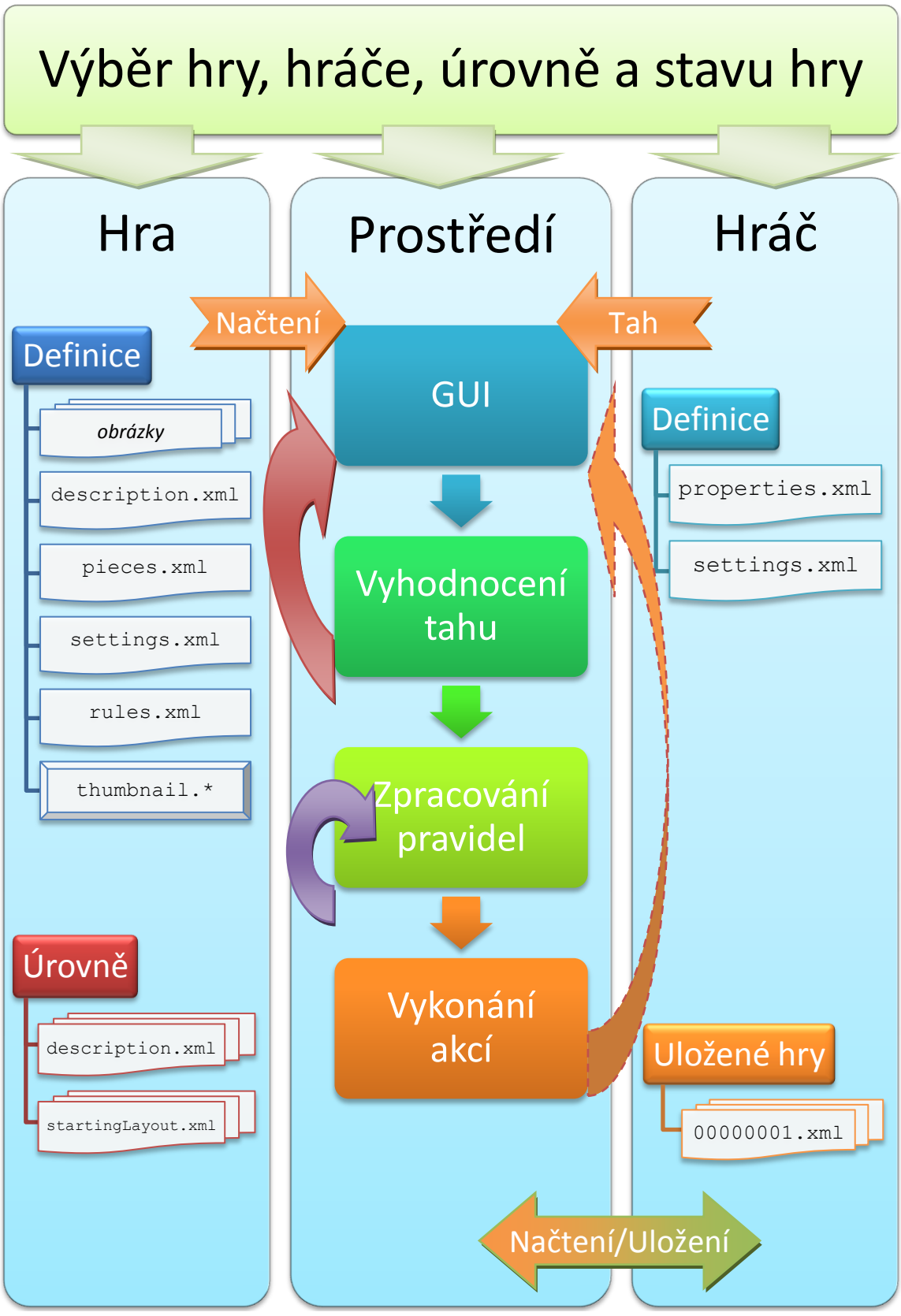
Uživatel je popsán pouze dvojicí souborů. V prvním (`properties.xml`) jsou uloženy jeho nacionále a v souboru `settings.xml` pak jeho vlastní nastavení her.

Před začátkem vlastního hraní se přes speciální dialog nejprve zvolí (přihlásí) uživatel (hráč), který si následně vybere hru, kterou chce hrát. Aplikace již sama rozpozná, kterou úrovní hráč naposledy skončil a v jakém stavu se tato úroveň nacházela. Tyto informace aplikace před začátkem hry získá právě z ukládaných souborů a hráč se tímto nemusí zabývat. Pokud jde o novou (první) hru, nastaví se herní plocha dle výchozího (počátečního) rozložení úrovně.

Po všech načteních je konečně zobrazena hra a hráč přes grafické rozhraní *(GUI)* vybírá a volí tahy. Tahy se vyhodnotí z hlediska platnosti. Při neplatném tahu se čeká na další tah. Pokud je platný, jsou v cyklu zpracována všechna pravidla a podle nich vykonány akce. Pokud akce ovlivní rozložení kamenů na hrací ploše, je tato informace přes GUI předána hráči. V každém případě se uloží nový stav hry a čeká se na další tah.

Celou tuto kapitolu výstižně shrnuje Schéma 2 na následující stránce.

Schéma 2: *Blokové schéma aplikace*



6.2 Třídy

Jednotlivé moduly a komponenty aplikace jsou definované jako třídy. Jejich metody a funkce pak realizují různé úkony. Tyto struktury a akce nejlépe vystihují třídové diagramy *(class diagram)*. Kompletní abecedně řazený přehled všech tříd bez dalších detailů a vazeb mezi nimi (dědičnosti apod.) uvádí následující diagram:

Diagram 2: *Všechny třídy aplikace*

Dále budou uvedeny již pouze nejvýznamnější třídy s jejich stručným popisem. U každé třídy budou zobrazeny hlavní datové i statické položky včetně jejich typů a vybrané metody.

Všechny diagramy byly vygenerovány vestavěným nástrojem Visual Studia® 2008.

6.2.1 Položky

Každá položka má svůj identifikátor *(name)* a informaci, zda se má vypisovat uživateli *(show)*. To může být zajímavé u položek jako je počet bodů, stav života apod. Položky nemají žádné vlastní metody a slouží hlavně pro uchování hodnoty *(value)* v podstatě libovolného typu.

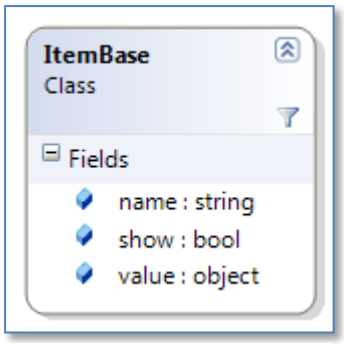


Diagram 3: *Třída položek*

6.2.2 Matice

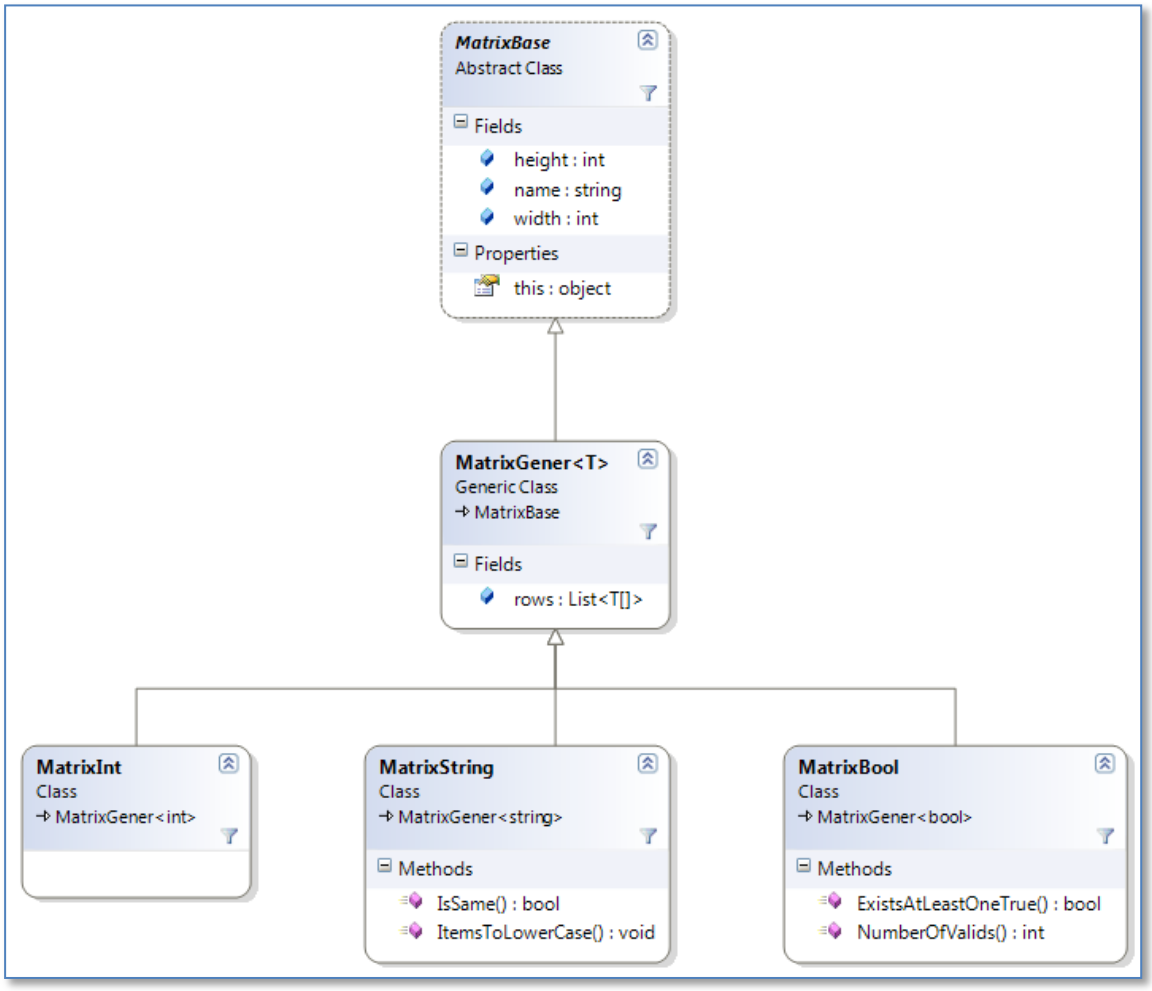


Diagram 4: *Hierarchie tříd matic*

Všechny tři matice STATE, VALID a FINAL i případné další jsou instancí jedné ze tří spodních matic na diagramu. Ty se liší typem obsažených položek i svými metodami. Všechny však mají některé shodné vlastnosti, které definuje jejich bázová třída *MatrixBase*. Těmi jsou jméno *(name)*, šířka *(width)* a výška *(height)*. Od této je poděděna ještě generická třída *MatrixGener*, která již definuje položky matic jako

seznam *(List)* polí typu *T*. Matice řetězců *MatrixString* implementuje funkci porovnání s jinou maticí *(IsSame)* kvůli vyhodnocení konečnosti hry a převedení všech jejích položek na malá písmena *(ItemsToLowerCase)* (viz ***kapitola 5.1***).

6.2.3 Běhové prostředí

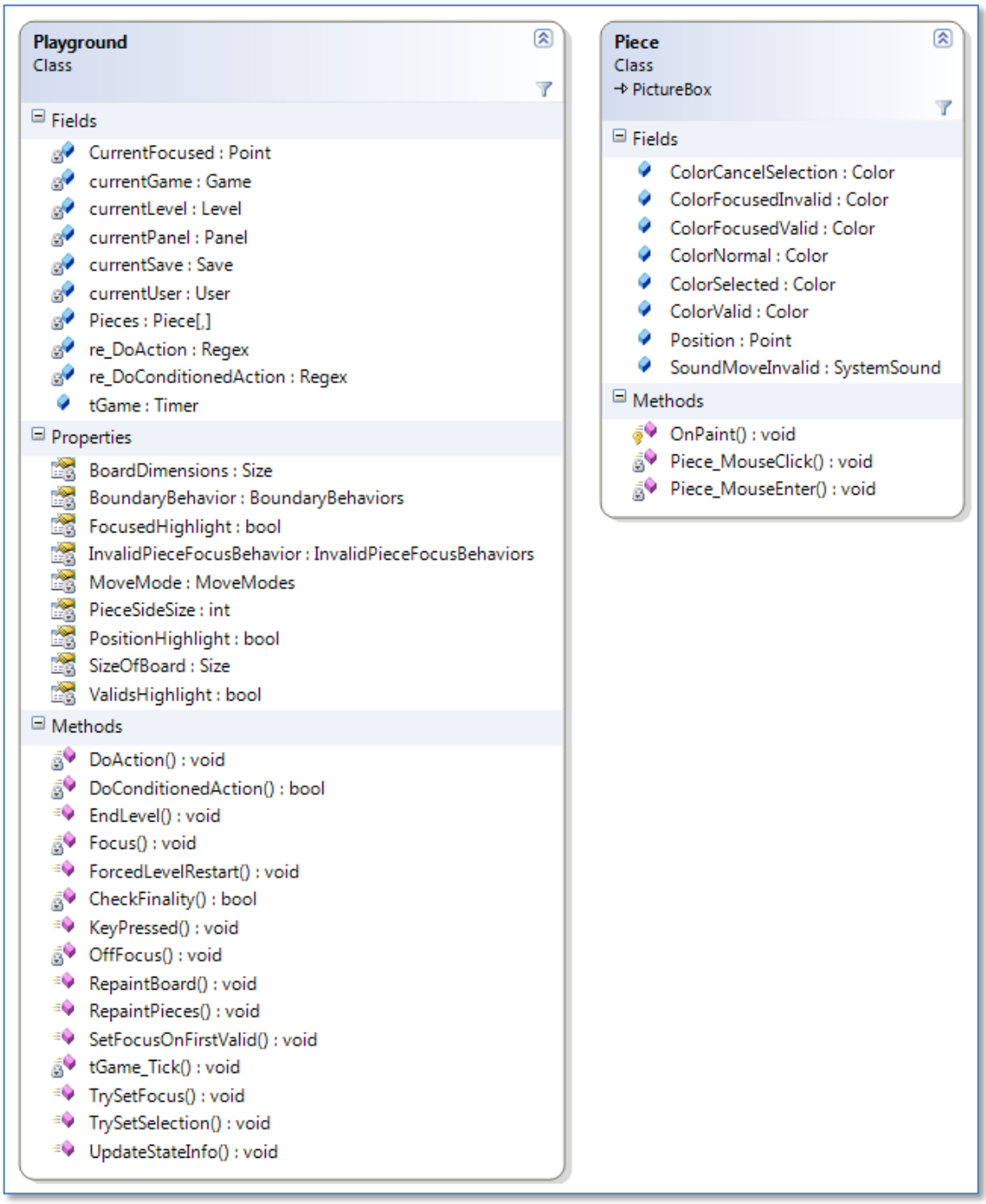


Diagram 5: *Třída běhového prostředí a třída kostičky*

Ústřední třídou je běhové prostředí (tzv. hřiště, *Playground*), kde se odehrávají všechny klíčové aktivity. Do prostředí vstupují hráč *(currentUser)*, hra *(currentGame)*, úroveň *(currentLevel)* a poslední známý (nebo nový) stav hry *(currentSave)*. Prostředí

si udržuje vizuální podobu hrací plochy ve dvojrozměrném poli kostiček *Pieces[,]*. Třída kostičky je na diagramu zobrazena vpravo. Tato třída je poděděná od standardního *PictureBox*u, neboť potřebuje implementovat vlastní konstruktor a pamatovat si svoji pozici *(Position)* v poli *Pieces*. Kromě toho v sobě také obsahuje statické definice barev kostiček v různých stavech *(Color*)* a zvuk při neplatném tahu *(SoundMoveInvalid)*.

Nad polem kostiček se pohybuje vizuální fokus, jehož poloha je držena v proměnné *CurrentFocused*. Fokus a zvýrazňování kostiček řeší metody *TrySetFocus*, *Focus*, *OffFocus* a *SetFocusOnFirstValid* na základě konfrontace nastavení hry a uživatele. Ta se děje v accessoru *(get)* všech **Highlight* vlastností, stejně tak vlastnosti *InvalidPieceFocusBehavior* a *BoundaryBehavior*.

MoveMode udává možný způsob výběru dalšího tahu (viz další **kapitola 6.4.1**).

Při stisku interní aplikační klávesy se v metodě *KeyPress* nejprve rozliší, o jakou klávesu se jedná, a na základě zjištění proběhne odezva.

V běhovém prostředí se také vykonávají všechna pravidla, ať už prostá přiřazení *(DoAction)* nebo podmíněné akce *(DoConditionedAction)*. Realizace těchto metod vyžaduje použití regulárních výrazů, které jsou zde také definované *(re_DoAction)* a *(re_DoConditionedAction)*. O nich podrobněji v následující **podkapitole 6.3.2**.

Po každé akci se zjišťuje konečnost stavu *(CheckFinality)*, která pokud nastane, je hra ukončena a přeskočí se na další úroveň *(EndLevel)*. Pokud již hráč nemá k dispozici žádné platné tahy, je vynucen restart úrovně *(ForcedLevelRestart)*.

Vlastní vykreslování plochy obstarává metoda *RepaintBoard*, která k tomu potřebuje mimo jiné znát rozměry využitelné plochy (z předaného *currentPanelu*) a rozměry matice STATE *(BoardDimensions)*. Z nich spočítá velikost strany *(PieceSideSize)* jednoho čtverečku (kostičky) a pak je všechny vykreslí *(RepaintPieces)*.

S přibývajícími tahy metoda *UpdateStateInfo* aktualizuje výpis počtu tahů a případně nesystémových položek. Ubíhající čas hry *(tGame)* se vypisuje nezávisle na dění v při události *tick*.

6.3 Vykonávání pravidel

Pravidla jsou vyhodnocována a vykonávána běhovým herním prostředím v pořadí, v jakém jsou zapsaná v souboru pravidel.

6.3.1 Hlavní cyklus

Po každém kole se vyhodnotí vždy všechna pravidla. Vyhodnocování v daném kole je přerušeno, pokud prostředí narazí na podmíněný příkaz, jehož podmínka je splněna a pravidlo je označeno v zápise jako breakující. Žádné z pravidel zapsaných v pořadí dále již není zkoumáno a čeká se na další tah hráče.

Může nastat situace, kdy chce pravidlo manipulovat s neexistující položkou či přiřadit neplatnou hodnotu. To se může stát např. překročením indexu matice nebo i

špatně zapsaným pravidlem. Aby aplikace neskončila výjimkou, je vykonání každého příkazu zahrnuto v bloku *try catch*. Týká se to jak jednoduchého přiřazení, tak složeného přiřazení, tak i vyhodnocování podmínek a provádění akcí ve výkonových blocích podmíněných pravidel. Má to své opodstatnění nejen kvůli ošetření možného chybného zápisu nebo opomenutí nějaké situace, ale především pro zjednodušení pravidel, kdy se v jejich zápisu nemusí ošetřovat např. chování na okrajích hracího pole apod. Neplatná pravidla a příkazy se jednoduše přeskočí.

Text této podkapitoly lze shrnout do následujícího zdrojového kódu:

```
foreach (GameRule rule in currentGame.rules)
{
    try
    {
        if (IsConditionedAction(rule.Contents))
        {
            if (DoConditionedAction(rule.Contents) && rule.Break)
                break;
        }
        else
            DoAction(rule.Contents);
    }
    catch { }
}
```

6.3.2 Regulární výrazy

Klíčovou roli hrají regulární výrazy (RV), které umožňují snadné parsování (převedení řetězce pravidla na jednotlivé entity), vyhodnocování a vykonání pravidel.

Vykonání každého pravidla zajišťuje postupně až 10 RV, jejichž výstupy (shody *matches*, skupiny *groups* a záchyty *captures*) slouží jako vstupy (parametry) přidružených funkcí a procedur, které jsou níže očíslovány. V těch se hodnoty vstupních parametrů předají konkrétnímu regulárnímu výrazu nebo výrazům (uvedených zde pod hlavičkou funkce) a výstup těchto RV pak řídí další kroky.

Postupně si všechny funkce a procedury stručně popíšeme. RV všechny kromě prvního hledají shodu řetězce pravidla s dílčí částí definované syntaxe v ***kapitole 5.3***. Význam symbolů užitých ve vlastním řetězci RV je daný definicí regulárních výrazů[16] v prostředí .NET a předpokládá se jejich znalost, nejsou dále vysvětlovány. Níže je pouze uveden jeden příklad jeho zápisu v C#. Dále už nás bude zajímat jen vlastní řetězec regulárního výrazu.

PLNÝ PŘÍKLAD ZÁPISU S VYSVĚTLENÍM

```
1 Regex re_IsAllTrue_conditions = new Regex(
2 @"(?<condition>\(.+?\))",
3 RegexOptions.ExplicitCapture | RegexOptions.Compiled);
```

1. Deklarace a definice nového regulárního výrazu s pojmenováním *re_IsAllTrue_conditions*
2. Vlastní řetězec výrazu uzavřený v uvozovkách. Jelikož zápis může obsahovat i symbol '\', který je normálně v jazyce C# u řetězců vyhodnocen jako tzv. ecape sekvence, je před řetězcem symbol '@'. Ten říká, že v následujím řetězci mají být escape sekvence ignorovány a má na ně být pohlíženo jako na součást řetězce.
3. Aby vyhodnocení probíhalo rychleji, má každý výraz nastaveny parametry *RegexOptions.ExplicitCapture* a *RegexOptions.Compiled*. První zajišťuje, že se zaznamenávají záchyty *(captures)* pouze explicitně pojmenované. Vždy nás totiž zajímá pouze několik vybraných (pojmenovaných) částí z řetězce pravidla. Druhý parametr by pak měl zajistit plynulejší zpracování při velkém množství vyhodnocení. Jeho vliv je ovšem sotva postřehnutelný, takže jeho použití nemá až takové opodstatnění.

1. VSTUPNÍ ÚPRAVA ŘETĚZCE

```
rule.Contents = Regex.Replace(rule.Contents, @"(\s|\.)",
string.Empty);
rule.Contents = rule.Contents.ToLowerInvariant();
```

Tímto příkazem s RV jsou z řetězce vymazány všechny netisknutelné znaky a tečky (kvůli rozepsaným složkám indexů A a P). Řetězec je poté převeden na malá písmena. Tato procedura se s každým pravidlem vykoná pouze jednou před začátkem celé hry.

2. `bool IsConditionedAction(string rule)`

```
(?<isConditioned>^if)
```

Pokud je pravidlo na vstupu (parametr) podmínkou, vrátí pravdu.

3. `bool DoConditionedAction(string rule)`

```
^if\((?<conditions>.+)\)\{(?<thenActions>[^\}]+)\}
(else\{(?<elseActions>[^\}]+)\})?$
```

Vykoná podmíněný příkaz na vstupu. Může jím být pouze s hlavním nebo i s vedlejším výkonovým blokem. Vrátí pravdu, pokud je podmínka příkazu splněna.

4. `bool IsAllTrue(string conditions)`

```
\)(?<logicalOperator>(or|and))\(

(?<condition>\(.+?\))
```

Tato funkce pracuje se dvěma RV. Vrátí pravdu, pokud je celý vstup pravdivý. Vstupem může být jak jednoduchý podmíněný výraz, tak i složený, kdy se v cyklu zjistí pomocí 5. funkce pravdivostní hodnota všech dílčích jednoduchých podmíněných výrazů.
V závislosti na hodnotě operátoru může být tento cyklus přerušen, ještě než se ohodnotí všechny výrazy.

5. `bool IsTrue(string condition)`

```
^\((?<leftSide>.+)(?<operator>(==|!=))(?<rightSide>.+)\)
```

Vrátí pravdu, pokud je jednoduchý podmíněný výraz na vstupu pravdivý.

6. `void DoActions(string actions)`

```
(?<action>[^;]+;)+
```

Provede v cyklu všechna přiřazení (jednoduchá i složená) na vstupu s pomocí 7. funkce. Podobně jako u hlavního cyklu, i zde jsou všechny akce přiřazení obaleny konstrukcí *try catch*.

7. `void DoAction(string action)`

```
^(?<leftSide>.+)=(?<rightSide>.+);
```

Vykoná přiřazení dané na vstupu.

8. `object EvaluateExpresion(string expresionShell)`

```
^(?<initValue>(\w+\[.+,.+\]|\w+))((?<operation>(\+|-){1})
(?<nextValue>(\w)+))*
```

Vrátí konečnou hodnotu po vykonání všech pod-operací složeného přiřazení.

9. `object GetValue(string valueShell)`

```
(?<matrix>\w+)\[(?<indexX>(\w)+((\+|\-){1}(\w)+)*),
 (?<indexY>(\w)+((\+|\-){1}(\w)+)*)]$
```

Důležitá funkce, která zjistí, co za entitu má na vstupu a vrátí hodnotu této entity přetypovanou na datový typ entity. Hodnotu získá buď jejím vytažením z matic, položek, nebo přetypováním řetězce, který je již samotnou hodnotou. Níže je uveden kompletní zdrojový kód této funkce i s vysvětlujícími komentáři.

```
Match matchResults = re_ProcessMatrixAndIndexes.Match(valueShell);
// pokud jde o odkaz do matice
if (matchResults.Success)
{
    // ze zadaneho retezce s matici a jejimi indexy vrati hodnotu ulozenou v
    // teto matici na dane pozici
    string matrix = matchResults.Groups["matrix"].Value;
    int x = (int)EvaluateExpresion(matchResults.Groups["indexX"].Value);
    int y = (int)EvaluateExpresion(matchResults.Groups["indexY"].Value);
    // vratime hodnotu vytazenou z matice
    return currentSave.Layout.matrixes.Find(mb => mb.name.ToLowerInvariant()
                                                  == matrix)[x, y];
}
else
    // nebo muze jit o polozku
    try
    {
        // pak polozku vytahneme z ulozeni
        return currentSave.Layout.items.Find(i => i.name.ToLowerInvariant()
                                                  == valueShell).value;
    }
    catch
    {
        // nebo jde o konkretni cele cislo (-5, 0, 1, 7, 100, ...);
        try
        {
            return Int32.Parse(valueShell);
        }
        catch
        {
            // muze jeste jit o hodnotu true/false
            if ((valueShell == "true") || (valueShell == "false"))
                return Convert.ToBoolean(valueShell);
            // jinak jde o vlastni prosty retezec bez vyznamu a vratime jej
            // tak, jak stoji
            else
                return valueShell;
        }
    }
```

6.4 Ovládání

Zde budou popsány všechny možnosti a kombinace, které aplikace dovoluje a implementuje. Konkrétní výchozí kombinace ze všech dále popisovaných *variant, možností, způsobů, druhů, voleb, zvýrazňování* a *chování* je pro každou hru specifická a je uložena v definičním souboru hry `settings.xml`. Některé z vlastností mohou být nastaveny ve hře napevno, hráč si je nemůže přizpůsobit. Nejčastěji jde o zakázání přímých výběrů u nepozičních her, kdy by přímá volba znemožnila samotné hraní (viz dále). Všechna ostatní povolená nastavení může hráč kdykoli během hry měnit.

6.4.1 Výběr a volba tahu

Pod pojmem **ovládání hry** budeme dále rozumět **způsob volby dalšího tahu (kostičky)**. V aplikaci byla implementována podpora obou základních možností, kterými jsou:

- *Přímé*
- *Nepřímé* (též *S možnostmi*)

U obou dále rozlišujeme, skrze jaké vstupním zařízením byl tah uskutečněn. U přímého:

- *Přímé myší*
- *Přímé klávesnicí*

U nepřímého figuruje navíc speciální signál Next popsaný v ***kapitole 3.1***:

- *Nepřímé myší* (*S možnostmi myší*)
- *Nepřímé klávesnicí* (*S možnostmi klávesnicí*)
- *Nepřímé generovanými Nexty* (též jen *Generováním Nextů*)

Nyní popíšeme, co se skrývá pod jednotlivými pojmy. Pokud je hra toho typu, kdy u ní lze definovat polohu hráče, má smysl hovořit o 4 okolí této polohy. Dalším platným tahem většinou bývá jedno z polí v tomto bezprostředním okolí. Hráč na ně tedy může vstoupit (zvolit je) **přímo** stisknutím směrové klávesy (šipek). Toto je definice **přímé** volby **klávesnicí**. Novou pozici (tah) však můžeme vybrat i přesunutím kurzoru myši nad žádané pole bez nutnosti potvrzení našeho výběru stiskem tlačítka myši. Pak mluvíme o **přímém** ovládání **myší**. Oba způsoby lze kombinovat, po každém ukončeném tahu může být další vybrán jiným způsobem.

Při přímých volbách se tah provede okamžitě bez přesunu fokusu (žlutého zvýraznění), resp. další tah i fokus se změní okamžitě na stejné pole.

U **nepřímých** voleb je nejprve přesunut fokus nad nějaké potenciální cílové pole a až po stisku akční klávesy Enter nebo tlačítka myši se právě fokusované pole zvolí jako další tah. Fokus může být přesouván směrovými klávesami (**nepřímé klávesnicí**), pohybem kurzoru myši (**nepřímé myší**) nebo samočinně ve stanoveném intervalu

(**generováním Nextů**). I zde platí, že metody (zařízení) lze libovolně kombinovat, na rozdíl od přímé volby navíc i při výběru jednoho tahu.

U nepřímých možností **myší** a **klávesnicí** je navíc situace ještě o něco komplikovanější, neboť dle preferencí hráče a/nebo hry může být zapnuta volba **výběru neplatných tahů**. Pokud skutečně je zapnutá, objevuje se fokus i nad políčky, která nejsou platnými tahy, pokud se nad ně přesune právě myší nebo šipkami. Fokus má v tom případě červenou (varovnou) barvu. Nejenže jsou fokusována, ale mohou být i zvolena (jako tah). Ten však bude vyhodnocen jako neplatný, o čemž je hráč informován varovným zvukem. Samozřejmě se též nevykonají žádná pravidla a akce, jen se čeká dál na platný tah.

Pokud je aktivní možnost **nevybírat neplatné tahy** a kurzor myši se přesto ocitne nad nějakým takovým polem, zůstane žlutý fokus na posledním platném poli, z kterého se kurzor následně přesunul na neplatné. Že nejde o platnou polohu kurzoru myši je signalizováno jeho změnou na varovný. Chování stisku směrových kláves pak při této volbě záleží ještě na jednom aspektu, který je popsán hned dále.

U **nepřímého** ovládání **klávesnicí** podstatně ovlivňuje výběr dalšího tahu tzv. **chování fokusu na hranici**. Neboli co se stane s polohou fokusu, pokud se ocitne na kraji hrací plochy a další směrová klávesa směřuje za hranici plochy. Jsou možné tři varianty a všechny jsou v implementaci podporované. Buď se nestane nic, fokus zůstane **stát** na místě (zarazí se o okraj hracího pole). Nebo se fokus přesune v řádku/sloupci na jeho začátek/konec, bude **rotovat**. Pokud je např. fokus na konci třetího řádku, stiskem šipky doprava se ocitne opět na začátku třetího řádku a obráceně. Obdobně je-li fokus na spodku pátého sloupce, šipkou dolů se ocitne nahoře tohoto sloupce a naopak. Poslední možností je **přeskočení** na předchozí/nový řádek/sloupec. Např. fokus na konci sedmého řádku při šipce doprava skočí na začátek osmého řádku a fokus v levém horním rohu se při šipce doleva i nahoru ocitne v pravém dolním rohu. Toto chování se nemusí rozlišovat vyloženě až na hranicích herní desky, ale obecně při absenci dalších platných tahů na všech polích nacházejících se mezi aktuální pozicí fokusu a hranicí hrací plochy ve směru určeném stisknutou šipkou.

Ze dvou výše uvedených pravidel (výběr neplatných tahů a chování na hranici) existuje jedna výjimka z aktuálního nastavení a tou je ovládání při generování **Nextů**. Aby byla vždy zaručena možnost postupného výběru všech platných tahů až k zamýšlenému, je při příchodu vstupu Next užito na hranici **přeskočení** bez ohledu na současné nastavení. **Neplatné** tahy jsou také vždy přeskakovány, neboť postupný výběr

všech polí, včetně neplatných, po každém tahu může u rozměrných hracích desek být neúměrně zdlouhavé (viz měření v ***kapitole 8***). Jinak je **chování** signálu Next stejné, **jako** by byla stisknuta klávesa **šipka doprava**.

U všech **přímých** i **nepřímých** variant se po každém platném tahu **fokus** automaticky umístí **na první platný tah**, neboť provedením tahu minulý fokus pozbude platnosti. Prvním platným tahem je buď sama nová pozice, pak zůstane fokus na ní, nebo se hledá směrem doprava a dolů (jako čtení). Pokud se při hledání platného tahu úspěchu dojde až na konec (pravý dolní roh matice), přeskočí se na začátek (levý horní roh) a hledá se dle stejného klíče dál.

6.4.2 Zvýrazňování

Jak už bylo naznačeno i v ***kapitole 4.3***, některá pole mohou být na herní ploše zvýrazněna různými barvami. Třemi základními jsou:

- *Zvýraznění fokusu*
- *Zvýraznění platných tahů*
- *Zvýraznění aktuální polohy*

Pokud je povoleno i vybírání neplatných tahů, tak ještě:

- *Zvýraznění neplatného tahu*

V případě, že platným tahem může být i aktuální poloha, jde pak vlastně o její odznačení (např. zrušení tahu danou figurou). Pak ještě navíc:

- *Zvýraznění zrušení výběru*

Ve všech ostatních případech má pole svoji vlastní barvu a je

- *Bez zvýraznění*

Všechny možné barvy kostičky a jejich pojmenování užívané ve zdrojovém kódu uvádí tato tabulka (0 = nepravda, 1 = pravda):

Má fokus?	Jde o aktuální pozici?	Je to platný tah?		Barva	Název barvy
1	1	1	→		*ColorCancelSelection*
1	1	0	→		*ColorFocusedInvalid*
1	0	1	→		*ColorFocusedValid*
1	0	0	→		*ColorFocusedInvalid*
0	1	1	→		*ColorValid*
0	1	0	→		*ColorSelected*
0	0	1	→		*ColorValid*
0	0	0	→		*ColorNormal*

Tabulka 2: ***Možná zvýraznění kostičky***

Tato plná paleta je však využita jen v případě, že kromě nastavení *vybírání neplatných tahů* jsou nastaveny i tyto tři příznaky pravdivostní hodnoty:

- *Zvýrazňování zamýšleného tahu (fokusu)*
- *Zvýrazňování platných tahů*
- *Zvýrazňování aktuální pozice*

Opět jde o nastavení vlastní každé hře a stejně jako možnosti ovládání, i tato nastavení mohou být uzamčena pro uživatelskou změnu. Co se stane, pokud jsou nastaveny na nepravdu je jasné z jejich pojmenování: fokus (žlutý ani červený) nebude nikdy zobrazen, žádné platné tahy nebudou nijak zvýrazněny a aktuální poloha nebude modře zvýrazněna. Nutno dodat, že i když fokus není zobrazován, je šipkami přesouván, jako by vidět byl.

7 Vytvoření nové hry

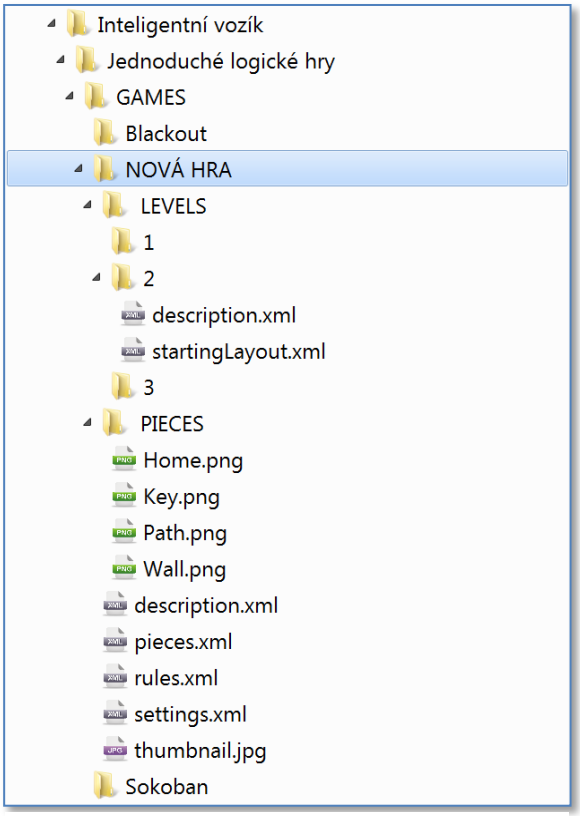
Níže je uveden postup na přidání nové hry. Jako ukázkový příklad je vzata hra Pexeso. Výpisy jednotlivých konfiguračních souborů jsou zkráceny, jejich plné znění naleznete po nainstalování v příslušném adresáři hry nebo na přiloženém CD v adresáři `GAMES\Pexeso`. Zeleně podbarvené odstavce jsou výpisem konkrétních XML souborů s očíslovanými řádky a tučně psanými příklady hodnot tagů, růžové (světle červené) podbarvení má pak vysvětlující text, kde čísla řádků odkazují na zazeleněný text.

7.1 Rozhodnutí

Předně je třeba si uvědomit, zda lze zamýšlenou hru popsat pouze pravidly, na která se vztahují omezení uvedená v ***5. kapitole Pravidla***. Pokud ano, pokračujte dalším krokem. Pokud ne, je třeba hru buď modifikovat/zjednodušit, nebo zvolit jinou.

7.2 Soubory a složky

Všechny dále popisované soubory mají své pevné místo a na jejich správném umístění a korektním obsahu záleží úspěšné načtení hry a její hraní. V adresáři `GAMES` nejprve vytvořte nový podadresář s libovolným pojmenováním. Vlastní a zobrazovaný název hry aplikace určí z popisujícího souboru (`desciption.xml`). Takto se zamezí možným problémům při používání diakritiky v názvech souborů a adresářů. Vpravo je uvedena jejich struktura s několika příklady obrázků a úrovní, kterou je třeba respektovat. Abyste nemuseli všechny soubory ručně vytvářet, můžete využít prázdnou šablonu pro novou hru, kterou najdete v podadresáři aplikace `HELP\#NEW_GAME#`. Tento adresář zkopírujte mezi stávající hry do adresáře GAMES, přejmenujte a soubory v něm pouze editujte. U všech souborů jsou v komentářích zapsány vysvětlivky k jednotlivým tagům (stejné jako vysvětlující text s růžovým



Obrázek 2: *Struktura adresářů a souborů nové hry*

podbarvením). Při používání diakritiky v XML souborech dbejte na uložení souboru v kódování UFT-8 a nikoli ANSI, to může způsobit jejich nenačtení.

U všech souborů pracujících se jmény kostiček, proměnných (matic a položek) a jejich hodnot nezáleží na velikosti písmen, kterými je zapíšete (tzv. *case-insensitive*). Je to jednoduchá prevence proti překlepům, kdy se všechny tyto řetězce převedou automaticky na malá písmena. Platí to jak pro zápis pravidel a zpracování pravidel, tak pro popis kostiček, rozvržení úrovně, záznam odehraných kol atd. V ukázkách je však pro lepší čitelnost a přehlednost užito i velkých písmen.

7.3 Definování hry

Hru definuje čtveřice XML souborů, sada obrázků kostiček a volitelně miniatura.

7.3.1 description.xml

Detailní slovní popis hry. Všechny položky kromě textových pravidel jsou načteny do úvodního seznamu her, aby uživatel měl základní představu o tom, jakou hru se chystá hrát a kolik času mu asi zabere.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <GameDescription
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <Name>Pexeso</Name>
4   <RulesTxt>Odhalte všechny shodné dvojice obrázků.
      Otočte nejprve jeden obrázek a pak druhý. Za každou
      správnou dvojici získáte deset bodů. Za každou
      špatně otočenou dvojici se Vám jeden bod odečte.
      Takže... PEKelně SE SOustřeď!</RulesTxt>
5   <LevelTakeTime>10 minut</LevelTakeTime>
6   <Difficulty>střední</Difficulty>
7   <Category>paměťová</Category>
8 </GameDescription>
```

1 hlavička souboru (neměnná)
2 otevření popisu hry (neměnné)

Tyto první dva řádky obsahují povinné systémové řetězce s odkazem na verzi, instanci a definici XML schématu a kódování. Musí být obsaženy v každém XML souboru, takže dále budou automaticky předpokládány, ale nebudou již uváděny.

3 název hry (řetězec)
4 slovní popis pravidel hry, instrukce k ovládání apod. (řetězec)
5 odhad herního času na jednu úroveň (řetězec)
6 slovní vyjádření obtížnosti celé hry (řetězec)
7 typ/zařazení hry (řetězec)
8 zavření popisu hry (neměnné)

7.3.2 pieces.xml

Popis libovolného množství kostiček, které se ve hře mohou vyskytnout. Mohlo by se zdát, že jde o nadbytečnou část, jelikož by se za jméno kostičky mohlo jednoduše dosadit jméno souboru. Pak by ale pro kostičky, které mají mít stejný obrázek (např.

karetní rub), bylo potřeba mnoho stejných obrázků pouze jiného jména. Tomu se zamezí právě naplněním tohoto souboru.

```
  2 <ArrayOfGamePiece>
  3   <GamePiece>
  4     <Name>01Front</Name>
  5     <PictureFileName>01Front.gif</PictureFileName>
  6     <ShowBorder>false</ShowBorder>
  7   </GamePiece>
  8   <GamePiece>
  9     <Name>02Front</Name>
  …
159     <Name>32Front</Name>
  …
164     <Name>01Back</Name>
165     <PictureFileName>Back.gif</PictureFileName>
  …
  …
169     <Name>02Back</Name>
170     <PictureFileName>Back.gif</PictureFileName>
  …
319     <Name>32Back</Name>
320     <PictureFileName>Back.gif</PictureFileName>
  …
323   <GamePiece>
324     <Name>Empty</Name>
  …
327   </GamePiece>
328 </ArrayOfGamePiece>
```

2	otevření seznamu kostiček (neměnné)
3	otevření kostičky (neměnné)
4	název kostičky. Tento identifikátor zastupuje kostičku v zápisu pravidel a v maticích STATE a FINAL, viz dále. (neměnné)
5	jméno souboru obrázku, který patří kostičce. Více kostiček může sdílet jeden soubor, což právě např. u pexesa je výhodné, neboť všechny otočené kostičky mají jeden a ten samý rub, kdežto líc má každá rozdílný. Jméno souboru se uvádí i s jeho příponou. Existuje jedna výjimka, a tou je použití uživatelova avataru (panáčka), což může být zajímavé např. u her typu bludiště. V tomto případě se zapíše zástupné slovo „Avatar“ bez žádné přípony a aplikace dosadí obrázek z uživatelova profilu. (řetězec)
6	určení, zda se kolem dané kostičky má vykreslit rámeček (cca 1pixelový černý obrys). U každé kostičky může být toto chování být nastaveno rozdílně. (true/false)
7	uzavření kostičky (neměnné)
8-327	otevření, popis a zavření dalších kostiček (neměnné)
328	uzavření seznamu kostiček (neměnné)

7.3.3 rules.xml

Univerzální sada pravidel ovlivňující chování všech elementů hry a řešící každou možnou situaci od začátku až do konce. Pravidla jsou vyhodnocována a akce v nich obsažené vykonávány po každém platném tahu v pořadí jejich záznamu v souboru.

Pokud jde o pravidlo s podmínkou, může u něj být nastaven příznak break, který vyhodnocení zbývajících pravidel ukončí. To může být pro některé hry nezbytné.

Tělo pravidla je zapsáno v syntaxi vycházející z jazyka C# s drobnými odchylkami. Detailní popis, co mohou pravidla obsahovat, s čím a jak mohou manipulovat a co ji i není platným zápisem uvádí samostatná ***5. kapitola Pravidla***. Jak jsou pravidla vykonávána a vyhodnocována je popsáno v ***kapitole 6.3***. Množství konkrétních příkladů i s vysvětlujícím popisem pak uvádí ***Příloha A***.

```
  2 <ArrayOfGameRule>
  3   <GameRule>
  4     <Contents>
  5         if (STATE[A.x, A.y] == 01Back)
  6         {
  7             Before = 01Back;
  8             STATE[A.x, A.y] = 01Front;
  9         }
 10     </Contents>
 11     <Break>false</Break>
 12   </GameRule>
 13   <GameRule>
  …
401         SecondValue = nothing;
  …
404   </GameRule>
405 </ArrayOfGameRule>
```

2	otevření seznamu pravidel (neměnné)
3	otevření pravidla (neměnné)
4	otevření obsahu pravidla (neměnné)
5-9	tělo pravidla (řetězec)
10	zavření obsahu pravidla (neměnné)
11	určení, zda se má v případě platnosti podmínky (pokud jde o podmíněné pravidlo) ukončit vyhodnocování zbylých pravidel od toho dále (pravdivostní hodnota pravda/nepravda, resp. jejich anglický ekvivalent true/false)
12	zavření pravidla (neměnné)
13-404	otevření, obsah a zavření dalších pravidel (neměnné)
405	zavření seznamu pravidel (neměnné)

7.3.4 settings.xml

Přizpůsobení výchozích nastavení aplikace. Jde především o způsob ovládání a vzhled. Správná kombinace a uzamčení různých nastavení je klíčová pro bezproblémové hraní. Uzamčením je myšleno nastavení zvláštní párové vlastnosti typu `bool`, kterou má většina nastavení. Pokud je nastavení odemčené, může se jeho hodnota měnit dle preferencí uživatele.

```
  2 <GameSettings>
  3   <MoveMode>WithOptions</MoveMode>
  4   <MoveModeChange>false</MoveModeChange>
  5
<InvalidPieceFocusBehavior>Skip</InvalidPieceFocusBehavior>
```

```
    6
<InvalidPieceFocusBehaviorChange>true</InvalidPieceFocusBehav
iorChange>
    7    <BoundaryBehavior>Jump</BoundaryBehavior>
    8    <BoundaryBehaviorChange>true</BoundaryBehaviorChange>
    9    <PositionHighlight>false</PositionHighlight>
   10
<PositionHighlightChange>false</PositionHighlightChange>
   11    <ValidsHighlight>false</ValidsHighlight>
   12    <ValidsHighlightChange>true</ValidsHighlightChange>
   13    <FocusedHighlight>true</FocusedHighlight>
   14    <FocusedHighlightChange>true</FocusedHighlightChange>
   15    <MarginWidth>25</MarginWidth>
   16    <BorderWidth>0</BorderWidth>
   17    <PaddingWidth>5</PaddingWidth>
   18    <BorderColorChange>false</BorderColorChange>
   19    <BoardColorChange>false</BoardColorChange>
   20    <BorderColor>Black</BorderColor>
   21    <BoardColor>Green</BoardColor>
   22 </GameSettings>
```

2 otevření herních nastavení (neměnné)

3 výchozí působ výběru dalšího tahu. (výčet S možnostmi/Přímé pouze klávesami/Přímé jak klávesnicí, tak myší; resp. jejich anglické ekvivalenty WithOptions/DirectJustKeyboard/DirectBothKeyboardAndMouse)

4 určení, zda si způsob výběru dalšího tahu může uživatel přizpůsobit. Pokud je výchozím způsobem jakékoli Přímé…, musí být nastevena na *true*, jinak hra nebude přístupná i handicapovaným. Naopak pokud je výchozím způsobem S možnostmi, je třeba rozmyslet, zda je nějaké Přímé… ovládání přípustné. U her typu pexeso, sudoku apod. přímé ovládání nemá smysl, při stisknutí směrové klávesy by se okamžitě vybrala kostička vedle posledně táhnuté. (true/false)

5 výchozí způsob chování fokusu, pokud narazí na neplatný tah. Buď jej může přeskočit/nevybrat, nebo se zvýrazní červeným rámečkem. (výčet, Přeskakovat/Vybírat; resp. jejich anglické ekvivalenty Skip/Select)

6 zamčení předchozí vlastnosti. Doporučuje se ponechat na *true*, jelikož ne každému uživateli může vyhovovat přeskakování např. již otočených kartiček u pexesa, při kterém může ztratit přehled. (true/false)

7 chování fokusu, pokud dorazí na hranici hracího pole nebo konec oblasti platných tahů. Při nastavení Stát se fokus zarazí ve své aktuální poloze. Při Rotovat přeskočí na začátek/konec stávajícího řádku nebo sloupce. Při Přeskočit fokus skočí na začátek dalšího sloupce nebo řádku/na konec předchozího řádku nebo sloupce. (výčet Přeskočit/Rotovat/Stát; resp. jejich anglické ekvivalenty Jump/Rotate/Stay)

8 zamčení předchozí vlastnosti. Je třeba mít na paměti, že pokud je pohyb možný pouze do 4 okolí bez středového pole, nemá pak uživatel pomocí šipek šanci vybrat všechna pole, pokud je chování nastaveno jiné, než *Jump*. Samozřejmě i nadále bude fungovat výběr přes Next (true/false)

9 určení, zda se má aktuální pozice zvýrazňovat. U některých her jako právě pexeso nemá pojem pozice/poloha smysl, pak nastavte na *false*. (true/false)

10 zamčení předchozí vlastnosti (true/false)

11 určení, zda se mají zvýrazňovat všechny platné tahy v daný okamžik. Někdy může být toto nadbytečné a rušivé, především u her, kdy se

matice VALID nemění a je možné táhnout vždy všemi/na všechny kostičky. (true/false)
12 zamčení předchozí vlastnosti (true/false)
13 určení, zda se má zvýrazňovat zamýšlený tah (tzv. fokus). (true/false)
14 zamčení předchozí vlastnosti. Pokud je zvýraznění fokusu nastaveno na *false* a tato vlastnost je zamčena, znemožní to pak hru ovládat pomocí výběru dalšího tahu, což je ale nutností pro handicapované hráče. (true/false)
15 odsazení (mezera) hrací desky od okraje okna v pixelech (celé číslo)
16 tloušťka okraje (rámečku) kolem všech kostiček v pixelech (celé číslo)
17 mezera mezi kostičkami v pixelech (celé číslo)
18 určení, zda si uživatel může přizpůsobit barvu rámečku podle svých preferencí. (true/false)
19 určení, zda si uživatel může přizpůsobit barvu pozadí hrací plochy podle svých preferencí. (true/false)
20 jméno barvy rámečku. Pokud je tento okraj nulový, je uvedená barva ignorována, ovšem nějaká uvedená být vždy musí. (výčet, standardní anglická pojmenování webových barev)
21 jméno barvy pozadí hrací desky, v podstatě barva mezery mezi kostičkami. Pokud je mezera nulová, je uvedená barva ignorována, ovšem nějaká uvedená být vždy musí. (výčet, standardní anglická pojmenování webových barev)
22 zavření herních nastavení (neměnné)

7.3.5 `thumbnail.*`

Volitelně můžete ke hře do jejího adresáře přidat miniaturní obrázek (tzv. thumbnail) vystihující hru. Obrázek umístěte do adresáře hry pod jménem `thumbnail.*`, kde místo * může být kterýkoli z formátů bmp/gif/jpeg/jpg/png/wmf. Rozměry miniatury mohou být libovolné, ale doporučená velikost je cca 128 × 128 px, neboť na tuto bude obrázek zmenšen resp. roztažen. Pokud není žádný obrázek nalezen, použije aplikace svůj výchozí (s hrací kostkou).

7.3.6 Obrázky

Vytvořte nebo vezměte hotové obrázky, které definují všechny objekty a jejich různé stavy na hrací ploše, se kterými se lze během celé hry setkat. Každý takový objekt je popsán právě jedním obrázkem. Jelikož se pracuje se čtvercovou maticí, měly by i obrázky být čtvercové, tedy mít oba rozměry shodné. Není to ale podmínkou, obrázky se vždy přizpůsobují do čtvercového tvaru. Výhodou je, pokud mají obrázky transparentní podklad, při fokusu (výběru) jsou pak výrazněji zviditelněny. Opět ale nejde o podmínku, výběr je vždy zvýrazněn i barevným rámečkem o šířce 10% velikosti strany kostičky. Obrázky mohou být v libovolném z formátů bmp/gif/jpeg/jpg/png/wmf, včetně animovaných, což může při některých hrát zpestřit jinak statickou hrací plochu.

Několik vhodných jednotných obrázků, z nichž můžete pro začátek vybírat, se po nainstalování objeví v podadresáři aplikace `HELP\PICTURES`. Nemusíte se jimi samozřejmě řídit, tato sada má pouze ulehčit hledání vhodných obrázků.

7.4 Úrovně

Vytvořte různé úrovně, má-li to u hry smysl. Jde-li např. pouze o samostatný hlavolam, pak stačí i jedna. Každá úroveň má vlastní podadresář. Na jeho pojmenování záleží sled načítání úrovní do hry, které probíhá podle abecedního pořadí jmen adresářů. Nejvhodnější a nejsnazší je pojmenovat složky čísly s doplněním nul na začátku tak, aby všechny úrovně měly stejný počet znaků. Tedy např. 01, 02, …, 09, 10, 11, 12. Pak je zaručeno pořadí načítání od 01 po 12. Těžší úrovně umístěte do adresářů s vyšším číslem, aby se do hry dostaly později.

Klidně ale můžete zvolit i slovní pojmenování adresářů, chování bude obdobné. Pojmenování ale na nic dalšího vliv nemá, aplikace si úrovně čísluje interně sama právě podle abecedního pořadí. Pokud chcete úrovni dodat nějaký popis, zaměřte se na první soubor:

7.4.1 description.xml

Slovní popis úrovně. Všechny položky jsou volitelné, ale soubor byť bez vyplněných položek musí být přítomen.

```
2 <LevelDescription>
3   <Name>Něco na zub</Name>
4   <Difficulty>střední</Difficulty>
5   <MinimumNumberOfMoves>64</MinimumNumberOfMoves>
6 </LevelDescription>
```

2 otevření popisu úrovně (neměnné)
3 pojmenování úrovně (řetězec)
4 slovní vyjádření obtížnosti dané úrovně (řetězec)
5 nejmenší možný počet tahů k úspěšnému dokončení úrovně. Pokud je nastavený, vyhodnocuje se konec hry až po překročení tohoto počtu tahů. Proto musí být zadáný bezchybně. Nejste-li si jisti, raději jej neuvádějte. (celé číslo)
6 zavření popisu úrovně (neměnné)

7.4.2 startingLayout.xml

Úroveň je definovaná trojí matic STATE, VALID a FINAL a volitelně výchozí pozicí P, které se postupně, jak hra ubíhá, mění. V tomto souboru musí být uloženo počáteční rozestavení kamenů (matice STATE), platných tahů (matice VALID), jeden či více konečných stavů (matice FINAL) a výchozí pozice (položky Ax, Ay, Px a Py), pokud to má pro hru smysl. I když však nemá, musí být tyto čtyři koordináty obsaženy a mít například nulovou hodnotu. Volitelně pak zde může být libovolné množství dalších

matic a položek potřebných ke hře. Jejich pojmenování pak musí korespondovat s názvy zvolenými v pravidlech hry.

```
<LevelLayout>
  <Items>
    <Item>
      <Name>Ax</Name>
      <Value xsi:type="xsd:int">0</Value>
      <Show>false</Show>
    </Item>
    <Item>
      <Name>Ay</Name>
…
      <Name>Px</Name>
…
      <Name>Py</Name>
…
      <Name>FirstValue</Name>
      <Value xsi:type="xsd:string">nothing</Value>
…
      <Name>FirstBefore</Name>
      <Value xsi:type="xsd:string">nothing</Value>
…
      <Name>FirstX</Name>
      <Value xsi:type="xsd:int">0</Value>
…
      <Name>FirstY</Name>
…
      <Name>SecondValue</Name>
…
      <Name>SecondBefore</Name>
…
      <Name>SecondX</Name>
…
      <Name>SecondY</Name>
…
      <Name>Before</Name>
      <Value xsi:type="xsd:string">nothing</Value>
…
      <Name>Bodů</Name>
      <Value xsi:type="xsd:int">0</Value>
      <Show>true</Show>
…
  </Items>
  <Matrixes>
    <MatrixString>
      <name>STATE</name>
      <width>8</width>
      <height>8</height>
      <Rows>
        <Row>
          <string>03Back</string>
          <string>20Back</string>
…
        </Row>
        <Row>
          <string>21Back</string>
          <string>14Back</string>
…
        </Row>
…
```

```
  161       </Rows>
  162     </MatrixString>
  163     <MatrixBool>
  164       <name>VALID</name>
    …
  168         <Row>
  169           <boolean>true</boolean>
  170           <boolean>true</boolean>
    …
  249     </MatrixBool>
  250     <MatrixString>
  251       <name>FINAL</name>
    …
  256           <string>Empty</string>
  257           <string>Empty</string>
    …
  269           <string>01Front</string>
  270           <string>Empty</string>
    …
  277           <string>01Front</string>
  278           <string>Empty</string>
    …
  336     </MatrixString>
  337     <MatrixString>
  338       <name>FINAL</name>
    …
 2948       <name>FINAL</name>
    …
 2973           <string>32Front</string>
    …
 3033     </MatrixString>
 3034   </Matrixes>
 3035 </LevelLayout>
```

2	otevření počátečního rozvržení úrovně (neměnné)
3	otevření položek (neměnné)
4	otevření položky (neměnné)
5	pojmenování položky. Zde Ax (zkrácené anglické Action X), neboli x-ová souřadnice právě odehraného tahu (řetězec)
6	hodnota položky. Jelikož se před každým vyhodnocením pravidel hodnota z Ax a Ay přiřadí do Px a Py, je zde v počátečním rozložení výchozí pozice uložena právě do akčních položek (Ax a Ay). (obecně libovolného datového typu, ten musí být uveden za částí `xsd`. V úvahu připadají tyto tři hlavní typy: `xsd:boolean`, `xsd:string`, `xsd:int`. Zde konkrétně posledně jmenovaný)
7	určení, zda se má hodnota dané položky vypisovat po každém odehraném kole do statistiky na panelu vpravo. Může být užitečné např. pro výpis skóre, života, energie, peněz apod. (true/false)
8	zavření položky (neměnné)
9–72	otevření a pojmenování dalších položek. Všimněte si různých datových typů a výpisu uživateli u položky *Bodů*
74	zavření položek (neměnné)
75	otevření matic (neměnné)
76	otevření matice, v tomto případě matice STATE (podle typu položek v ní obsažených buď `<MatrixString>`, `<MatrixInt>` nebo `<MatrixBool>`, neměnné)
77	pojmenování matice (řetězec)

```
   78  šířka matice - počet kostiček na šířku (celé číslo)
   79  výška matice - počet kostiček na výšku (celé číslo)
   80  otevření řádků (neměnné)
   81  otevření řádku (neměnné)
   82  hodnota buňky uzavřená do tagu určujícího její typ. V případě matic
       STATE a FINAL je hodnota odkazem na název kostičky ze souboru
       pieces.xml (různého typu podle matice, viz výše)
 83-100  další hodnoty buněk, otevření a zavření dalších řádků (různého typu)
  161  zavření řádků (neměnné)
  162  zavření matice (stejně jako u otevření záleží na aktuálním typu matice, viz
       výše)
163-249  otevření, pojmenování a naplnění matice VALID
250-336  otevření, pojmenování a naplnění první matice FINAL
337-3033  otevření, pojmenování a naplnění dalších 31 matic FINAL
 3034  zavření matic (neměnné)
 3035  zavření počátečního rozvržení úrovně (neměnné)
```

7.4.3 Šablona pro generování matic

Psát všechny položky každé matice ručně, navíc rozepisovat každou buňku pod sebe, místo přirozenějšího řazení vedle sebe, by bylo únavné a jistě by vedlo k množství chyb. Aplikace sice zatím nedisponuje grafickým rozhraním pro tvorbu úrovní, za tímto účelem však byla vyvinuta podpůrná šablona pro Microsoft Excel, kterou naleznete po nainstalování v podadresáři aplikace `HELP\#NEW_GAME#\LEVELS` nebo na přiloženém CD v adresáři `GAMES\#NEW_GAME#\LEVELS`. Tato šablona umožňuje zjednodušený návrh úrovní až do rozměru 50 × 50 kostiček. Nejprve zvolíte, o jakou ze tří matic se jedná, pak ji naplníte daty, z prostředního listu zkopírujete celý vygenerovaný obsah do posledního listu (Kopírovat → Vložit jinak… → Hodnoty) a spustíte makro, které vymaže prázdné řádky. Výsledek pak můžete zkopírovat tak jak je do souboru `startingLayout.xml` na příslušné místo matice. Postup opakujte pro každou ze tří či více matic.

7.5 Spuštění hry

Při příštím dialogu s výběrem hry se nově vytvořená hra načte mezi ostatní a můžete začít hrát. Pokud dojde k nějaké chybě, překontrolujte dodržení souborové a adresářové struktury, přítomnost všech souborů, validitu XML kódu (uzavřené tagy apod.), správné pojmenování obrázků včetně jejich odpovídacích přípon, shodnost rozměrů všech tří matic, jejich přítomnost a přítomnost 4 systémových položek (Ax, Ay, Px, Py) včetně hodnot. Pokud se hra načte správně, ale její průběh není dle Vašich představ, zaměřte se na soubor s pravidly. Zamyslete se nad pořadím jejich vyhodnocování, a zda některá nemají být breakující. Pokud ani poté neuspějete, zkuste se obrátit na autora, viz kontakt v okně *O aplikaci* (viz Obrázek 35).

8 Výsledky

Funkčnost aplikace byla ověřena několika testovacími partiemi. Předmětem těchto her bylo zjištění efektivity navržených způsobů ovládání, což je klíčové z hlediska smysluplnosti celého projektu. Ze sady realiovaných her byly vybrány čtyři, u kterých je jasně daný optimální postup řešení a tento byl dodržen u všech způsobů ovládání.

8.1 Sběr dat

Aplikace při hraní postupně generuje soubory obsahující vždy kompletní informaci o stavu hry (obsah všech matic a položek). Tyto soubory jsou generovány po každém platném tahu. Na základě těchto souborů je možná úplná zpětná rekonstrukce odehrané partie (úrovně). Kromě vlastních datových položek se v každém souboru ukládají navíc další informace, které mohou posloužit při další analýze. Jsou jimi:

- Absolutní čas (kompletní datum a čas tahu)
- Čas od začátku hry
- Čas od posledního tahu
- Číslo tahu
- Nastalá událost. Tou může být buď vlastní platný tah *(Action)*, nebo začátek hry *(StartOfGame)*, přerušení hry *(GameInterrupted)* nebo úspěšný konec hry *(GameFinished)*.

Všechny tyto údaje o odehraných hrách spolu s hodnotami všech položek je v aplikaci možné zobrazit do přehledové tabulky pod tlačítkem Vývoj hry, v které jeden řádek odpovídá jednomu stavu hry (souboru). Ukázku jedné takové tabulky uvádí Obrázek 3 níže. Plný obsah matic v ní pro přehlednost zobrazován není, vypsány jsou pouze informace o počtu matic FINAL v dané úrovni a počtu platných tahů v daném kole. Tato informace může být zajímavá např. v souvislosti s časem potřebným na jednotlivé tahy. S klesajícím počtem platných možností (např. u hry Pexeso a v závěru hry Jezdcova procházka) by se výběr tahů měl zrychlovat.

Úroveň	Číslo uložení	Datum	Čas od začátku	Čas od posledního tahu	Počet tahů	Počet platných možností	Počet konečných stavů	Událost	Ax	Ay	Px	Py
01	00000000	7.5.2010 12:02:46	00:00:00	00:00:00	0	8	8	StartOfGame	3	4	0	0
01	00000001	7.5.2010 12:02:50	00:00:03	00:00:03	1	7	8	Action	5	3	3	4
01	00000002	7.5.2010 12:02:55	00:00:08	00:00:05	2	7	8	Action	3	2	5	3
01	00000003	7.5.2010 12:02:58	00:00:11	00:00:03	3	3	8	Action	4	0	3	2
01	00000004	7.5.2010 12:03:00	00:00:13	00:00:02	4	5	8	Action	2	1	4	0
01	00000005	7.5.2010 12:03:01	00:00:14	00:00:01	5	1	8	Action	0	0	2	1
01	00000006	7.5.2010 12:03:02	00:00:15	00:00:01	6	5	8	Action	1	2	0	0
01	00000007	7.5.2010 12:03:06	00:00:19	00:00:04	7	3	8	Action	0	4	1	2
01	00000008	7.5.2010 12:03:11	00:00:24	00:00:05	8	3	8	Action	1	6	0	4
01	00000009	7.5.2010 12:03:14	00:00:27	00:00:03	8	3	8	GameInterrupted	1	6	0	4
01	00000010	7.5.2010 12:11:37	00:00:33	00:00:09	9	3	8	Action	3	7	1	6
01	00000011	7.5.2010 12:11:38	00:00:34	00:00:01	10	5	8	Action	5	6	3	7
01	00000012	7.5.2010 12:11:39	00:00:35	00:00:01	11	1	8	Action	7	7	5	6
01	00000013	7.5.2010 12:11:41	00:00:37	00:00:02	12	4	8	Action	6	5	7	7
01	00000014	7.5.2010 12:11:46	00:00:42	00:00:05	13	3	8	Action	7	3	6	5
01	00000015	7.5.2010 12:11:51	00:00:47	00:00:05	13	3	8	GameInterrupted	7	3	6	5

Obrázek 3: ***Ukázka přehledové tabulky vývoje hry***

8.2 Ověření ovladatelnosti

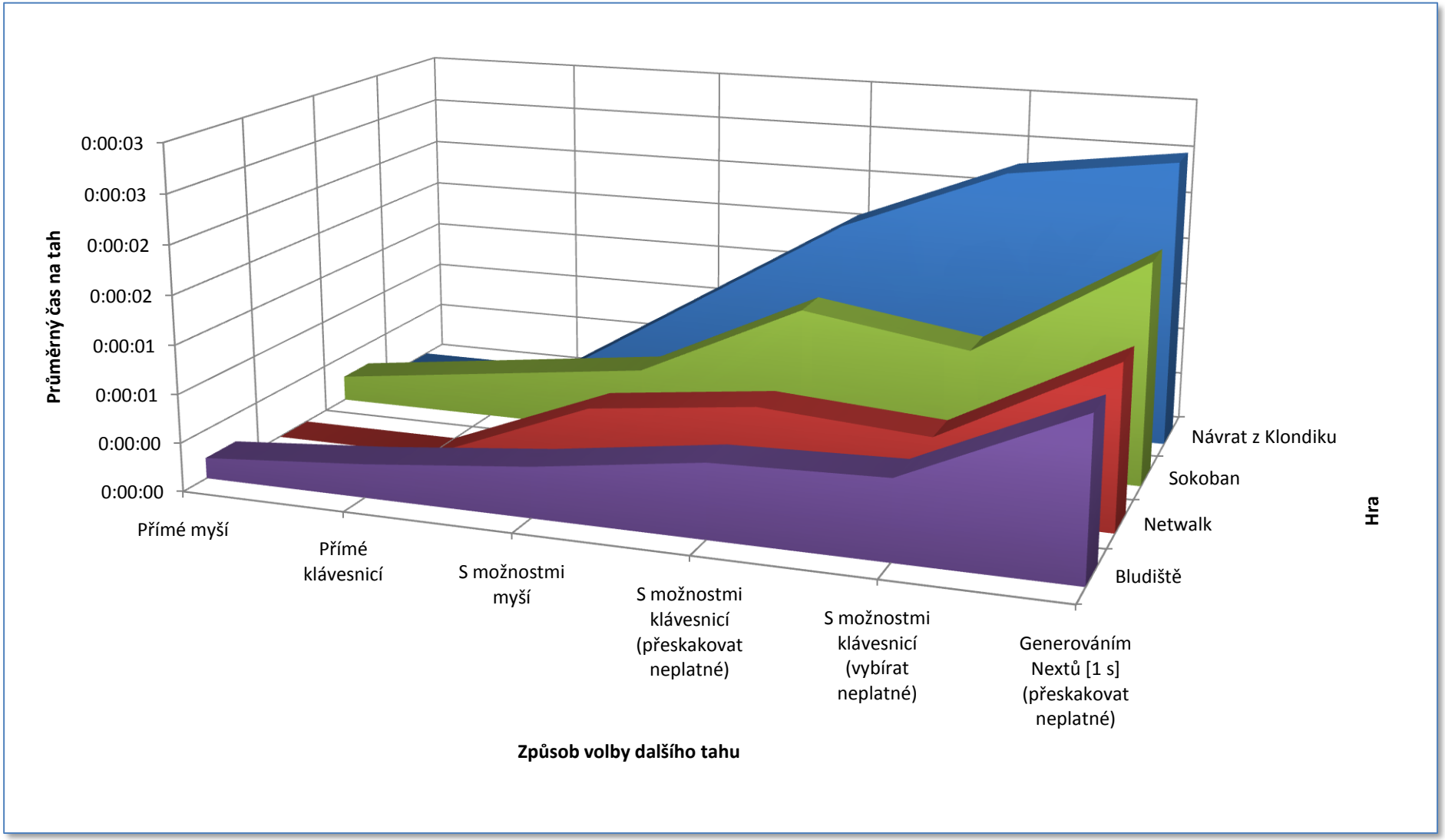
Z přehledových tabulek odehraných testovacích partií analyzovány dva údaje: *Čas od posledního tahu* a *Čas od začátku* hry. Kritériem pro ověření ovladatelnosti jsou průměrná dobu mezi tahy a celkový čas potřebný k dohrání hry při použití různých způsobů ovládání. Průměrná doba je prostým aritmetickým průměrem časů mezi tahy a čas dohrání je získán z celkového času na posledním řádku, kde už je hra dohraná. Takto získaná data z jednotlivých měření (hraní) shrnuje Tabulka 3 a Tabulka 4. Přehledné srovnání těchto časů pak ukazují Graf 1 a Graf 2.

Tabulka 3: *Průměrné časy potřebné na tah při různých způsobech jeho volby*

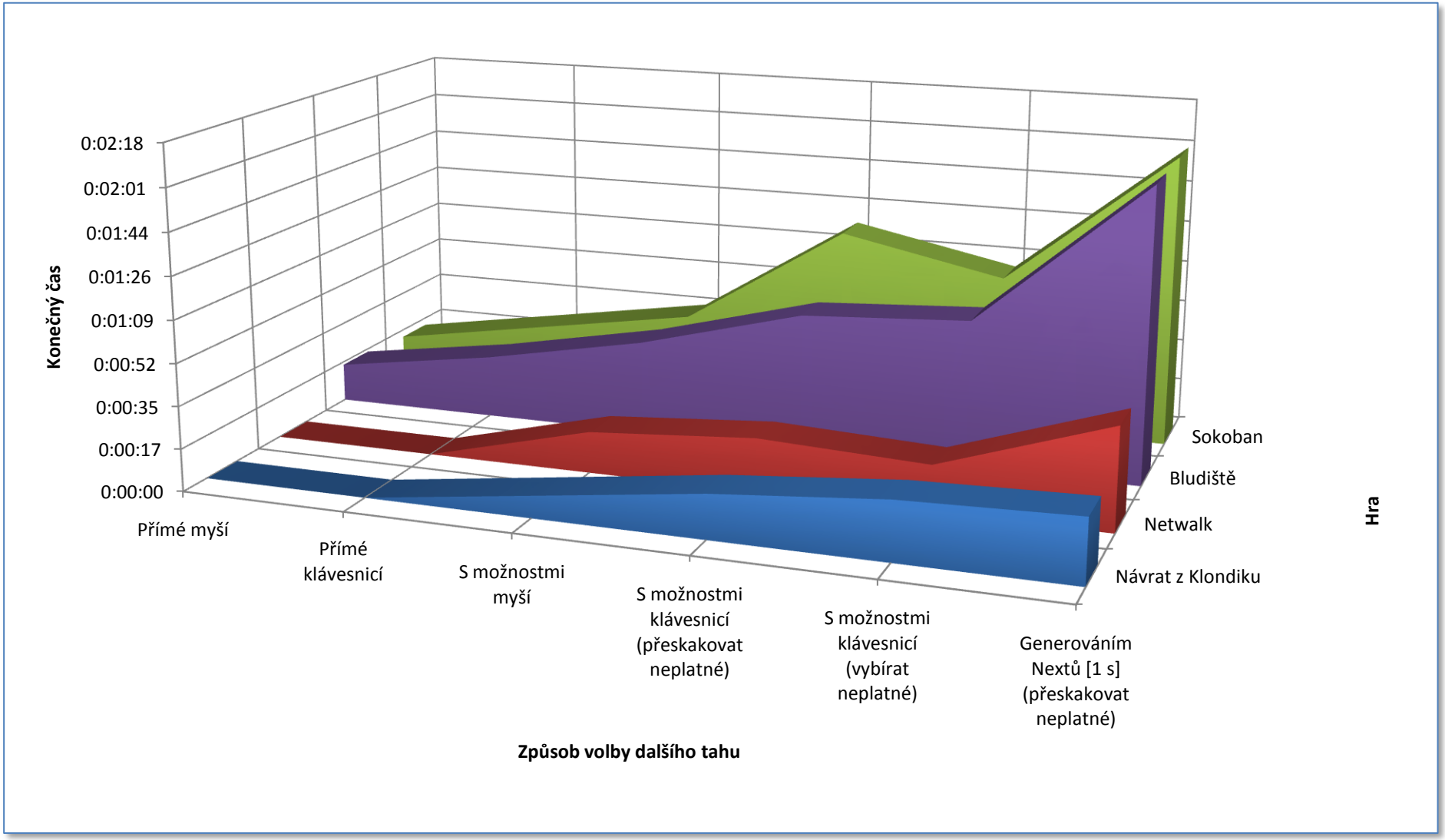
Způsob volby tahu	Návrat z Klondiku	Netwalk	Sokoban	Bludiště
Přímé myší	x	x	0:00:00	0:00:00
Přímé klávesnicí	x	x	0:00:00	0:00:00
S možnostmi myší	0:00:01	0:00:01	0:00:01	0:00:00
S možnostmi klávesnicí *(přeskakovat neplatné)*	0:00:02	0:00:01	0:00:01	0:00:01
S možnostmi klávesnicí *(vybírat neplatné)*	0:00:02	0:00:01	0:00:01	0:00:01
Generováním Nextů [1 s] *(přeskakovat neplatné)*	0:00:03	0:00:01	0:00:02	0:00:01
Generováním Nextů [1 s] *(vybírat neplatné)*	0:02:40	0:00:02	0:00:24	0:02:49

Tabulka 4: *Časy potřebné k dohrání hry při různých způsobech volby tahu*

Způsob volby tahu	Návrat z Klondiku	Netwalk	Sokoban	Bludiště
Přímé myší	x	x	0:00:14	0:00:16
Přímé klávesnicí	x	x	0:00:25	0:00:26
S možnostmi myší	0:00:09	0:00:17	0:00:36	0:00:39
S možnostmi klávesnicí (přeskakovat neplatné)	0:00:18	0:00:22	0:01:19	0:00:57
S možnostmi klávesnicí *(vybírat neplatné)*	0:00:24	0:00:19	0:01:05	0:01:01
Generováním Nextů [1 s] *(přeskakovat neplatné)*	0:00:26	0:00:42	0:02:01	0:02:01
Generováním Nextů [1 s] *(vybírat neplatné)*	0:26:38	0:01:05	0:24:53	4:05:24



Graf 1: ***Průměrné časy potřebné na tah při různých způsobech jeho volby***



Graf 2: *Časy potřebné k dohrání hry při různých způsobech volby tahu*

Nejrychlejší ze sedmi možností ovládání je dle očekávání **Přímé myší** následované **Přímým klávesnicí**, tedy u her, které je připouštějí.

O něco pomalejším způsobem je ovládání **S možnostmi** volenými **myší**. Rozlišovat zde chování fokusu na neplatných polích nemá význam, neboť vlastní poloha kurzoru myši se při vybírání ani přeskakování programově neovlivňuje a není tedy ovlivněna ani rychlost volby tahu.

U způsobu **Možnosti klávesnicí** je situace hry od hry různá, někdy je výhodnější **neplatné tahy** nechat **přeskakovat** (Bludiště a Návrat z Klondiku), jindy naopak **vybírat** (Sokoban a Netwalk).

Co nás ovšem zajímá nejvíce je srovnání předchozích možností s volbou pomocí **Generovaných Nextů**. Záleží samozřejmě na intervalu generování, zde byla zvolena jedna sekunda. V praxi však může být časový interval mnohem delší, záleží na možnostech uživatele. Opět bylo měřeno se zapnutou i vypnutou volbou **Vybírat neplatné**. Už z tabulek výše je evidentní, že pokud se mají **vybírat** i neplatné tahy, stává se hra při rozlehlých hracích plochách (Návrat z Klondiku a Bludiště) enormně dlouhou, neboť je většinou nutné projít je pokaždé celé znovu. To tuto volbu činí prakticky nepoužitelnou a byla proto ze zdrojového kódu aplikace odstraněna (podmíněným překladem). Volba není ani vizualizovaná v grafech, neboť by natolik rozšířila škálu času, že by zbývající plochy ztratily vypovídající hodnotu.

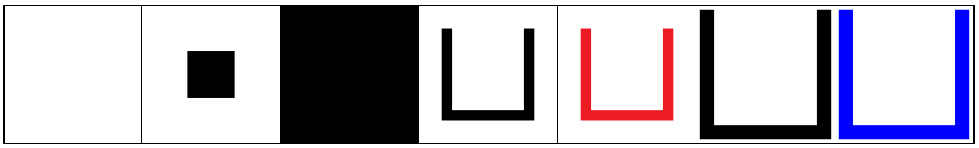
Pozitivní je však zjištění, že při Generování Nextů a **Přeskakování** neplatných tahů lze hru dohrát v čase srovnatelném s konvenčními způsoby ovládání.

8.3 Návrhy na vylepšení

Stávající architektura aplikace a pojetí všech jejích součástí jistě není optimální. V této fázi však šlo především o prokázání správnosti navržené koncepce *(proof of concept)* a ověření její funkčnosti na několik konkrétních hrách. Dále budou uvedeny některé známé nedostatky.

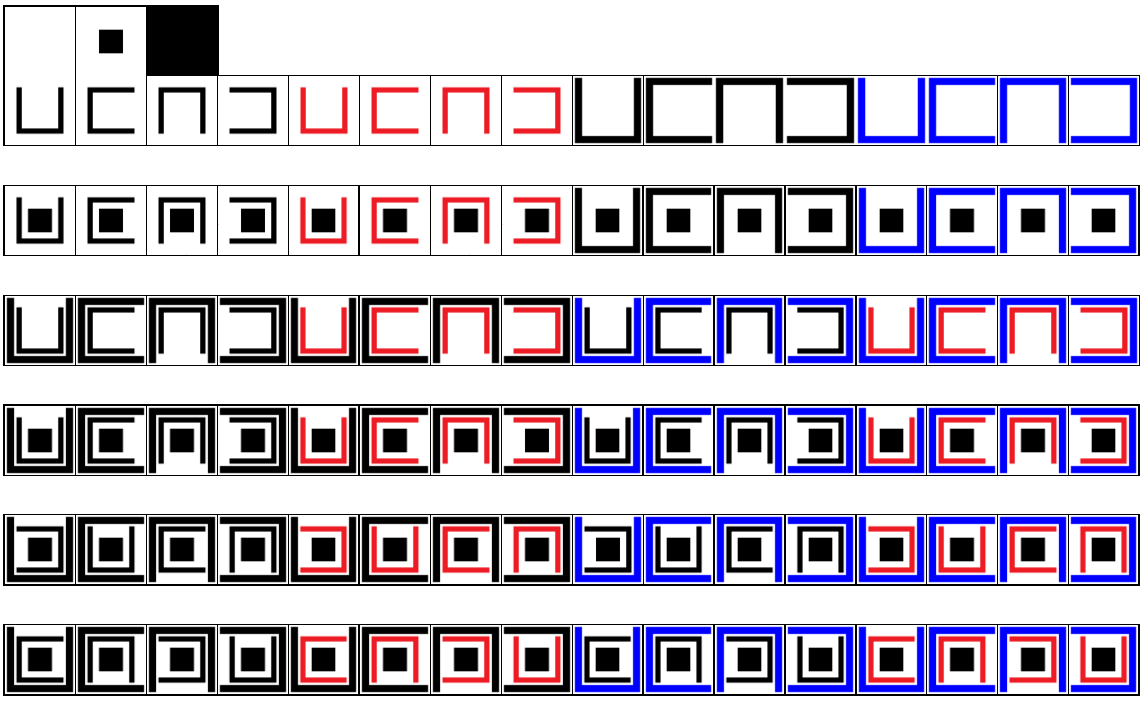
8.3.1 Pravidla

Dané pojetí pravidel s sebou nese jistá omezení. Pravidla např. neumožňují využít rotace kostiček a je proto třeba mít 4 různé obrázky. Dalším zásadním omezením je pouze jediný obrázek pro každou pozici na hrací desce a nemožnost tedy obrázky skládat do vrstev, což by na druhou stranu vyžadovalo jejich transparentnost. Důsledky těchto dvou omezení lze jasně vidět na příkladu hry Zabal to! (viz Tabulka 7). Pokud by šla využít rotace a vrstvení obrázků, stačilo by na popis všech možných stavů 7 základních obrázků:



Tabulka 5: *Kostičky potřebné při podpoře rotace a vrstvení*

Nyní by však bylo zapotřebí bez jednoho 100 různých obrázků popisující kombinující stavy:



Tabulka 6: *Kostičky potřebné bez podpory rotace a vrstvení*

Nejen, že je to nepřehledné, ale především zapsání interakcí všech těchto kostiček se stává nadlidským úkolem, byť je však možné.

Jak již bylo naznačeno v ***kapitole 5.3 Syntax***, v zápise pravidel nelze použít zanořených podmínek a příkazů, což by umožňovalo implementaci složitějších her, např. Zhasni se třemi barvami. Také není možné zapisovat podmínky se současnou kombinací operátorů AND a OR a různě vnořené pod-podmínky uzavřené v závorkách. Obojí by šlo řešit vhodně nastavenou rekurzí.

8.3.2 Soubory

Ukládání každého tahu s kompletní informací o hře (všechny matice a položky) může být časově i prostorově náročné, pokud je hrací plocha rozsáhlá a/nebo existuje větší množství matic FINAL. U všech realizovaných her se matice FINAL během hry nemění a je tedy nadbytečné ji ukládat spolu s maticemi STATE a VALID. Dokonce i matice VALID se u mnoha her od začátku do konce nemění, takže by bylo vhodné do

definice matice ve třídě *MatrixBase* přidat vlastnost typu `bool`, zda se matice během hry mění a neměnící se pak neukládat. Na druhou stranu by toto diferencovalo do teď jednotnou strukturu ukládaných souborů, což by zkomplikovalo jejich případné hromadné vyhodnocování.

V aplikaci se většinou předpokládá přítomnost a korektnost všech definujících souborů. Pokud některý chybí nebo má neplatný obsah, nastane výjimka. Základní výjimky a ty, které mohou nastat při vytváření uživatele nebo prvním spuštění hry, jsou samozřejmě ošetřeny. Ošetření všech myslitelných by však již bylo v tomto prvotním návrhu na úkor čitelnosti zdrojového kódu.

8.3.3 Grafika

Vykreslování obrázků kostiček do *PictureBoxů* je zdá se při jejich větším množství poněkud problém. Prvotní vykreslení rozložení může trvat až nepříjemně dlouho, což samozřejmě kazí dojem ze hry. Možným řešením by bylo vykreslovat všechny obrázky na plochu okna pomocí funkce *DrawImage* s použitím grafického kontextu hlavního okna. Tato funkce však na druhou stranu neposkytuje takový komfort jako *PictureBox*, neboť jí nakreslený objekt nemá k dispozici metody *OnPaint*, *OnMouseEnter* a *OnMouseClick*, které jsou z hlediska implementace interakce s hráčem velmi vhodné.

Také přechody mezi stavy, kdy se vrací tah, jsou zatím řešeny překreslením celé scény, což není efektivní. Možným řešením je podržet vždy poslední tah v paměti a při žádosti na vrácení tahu změnit zaměnit pouze posledně manipulovanými kameny.

8.3.4 Generátor úrovní

Aplikace v současném stádiu vývoje neobsahuje žádný nástroj, jak pro danou hru zcela automaticky vygenerovat nové úrovně. Ty je zatím třeba editovat ručně podobně, jako popis hry. To je samozřejmě nepohodlné a neefektivní, navíc malé množství úrovní samozřejmě hráče časem omrzí. Přestože v zadání práce tento požadavek nebyl dán, je implementace tohoto nástroje v další fázi aplikace klíčovou.

Dobrou výchozí pozicí je modifikace výstupů již existujících ad-hoc algoritmů pro generování úrovní konkrétních her. Stejně jako jsou univerzální pravidla popisující hry, měl by i generátor být univerzální a na vstupu mít sadu těchto pravidel. To ovšem nestačí. Pravidla her sama o sobě nezjišťují konec hry, ten je předem daný maticí FINAL. Proto bude třeba pro generátor tato pravidla přenést na vyšší úroveň a zajistit, aby sada pravidel sama o sobě byla schopná na základě aktuální situace na desce vyhodnotit fázi hry. Zpětná náhodná aplikace těchto pravidel na nějaký konečný stav („zamíchání hry“) by pak mohla být základní myšlenkou tohoto generátoru.

Samozřejmostí generovaných úrovní je tedy jejich dohratelnost, volitelnou funkcionalitou je předání minimálního počtu tahů k úspěšnému dokončení hry. Prostředí jej umožňuje zobrazit a motivovat tak hráče při opakování hry k nalezení nejlepšího řešení a má význam i pro optimalizaci výpočetní náročnosti (viz ***kapitola 4.4 Průběh hry***). Toto číslo však není vždy jednoduché získat. Optimální řešení mnoha her je NP-úplným problémem[16], takže generátor sice může zajistit řešitelnost, ale už těžko zjistí, jaké řešení je optimální, potažmo počet kroků k němu potřebných. Ani pokud by generátor pracoval na principu zpětného přeuspořádání finální pozice, nemusí být tato úloha primitivní. Například pokud zamícháme Loydovu patnáctku z jejího konečného rozložení 40 tahy, je možné, že existuje i jiné lepší řešení na méně tahů. Dokonce pokud bychom ji zamíchali např. 200 tahy, lepší řešení bude existovat vždy, neboť je dokázáno, že každé rozložení je řešitelné maximálně 80 tahy[25].

Ne vždy má samozřejmě toto číslo vypovídající hodnotu. Např. u pexesa je optimální počet kroků vždy konstantní, navíc i při sebelepší paměti kvůli počáteční skryté informaci nedosažitelný. Stejně tak u Jezdcovy procházky je tento počet nejen optimální, ale dokonce jediný možný. Faktorem úspěšnosti by v tomto případě byl spíše počet restartů úrovně potřebných k bezchybnému projití šachovnice, než samotný počet kroků.

9 Závěr

SHRNUTÍ

V úvodních kapitolách byl nastíněn současný stav na poli her pro tělesně postižené s důrazem na nejtěžší formy handicapu, kdy je jedinec schopen ovládat pouze jediný vstup (tlačítko). Že dnes takových nabízených her není mnoho, je i vzhledem k šíři této specifické hráčské komunity pochopitelné. Dále byly uvedeny základní předpoklady pro hraní elektronických her takto tělesně postiženými osobami. Předně je to nutnost znát dopředu množinu platných tahů a pak jejich snadný výběr. Navržený systém pravidel her naplňuje první požadavek, druhý pak běhové herní prostředí, které je součástí herní aplikace implementované v jazyce C#.

Další kapitoly byly již věnované samotnému návrhu herního systému, reprezentaci stavu hry a zápisu pravidel. Návrh sice vychází ze základních myšlenek GDL, nicméně jej neimplementuje. Jednak kvůli jeho až *nadbytečné* funkcionalitě při vymezeném okruhu her, na druhé straně paradoxně *nedostatečné* z pohledu handicapovaného hráče. Byl proto specifikován vlastní, sice jednodušší, systém, který ovšem respektuje požadavky těchto hráčů a pro dané typy her je dostačující.

V souladu s touto specifikací je na 70 her *(přehled v Příloze B)*, z nichž 12 bylo pro ukázku skutečně realizováno. Jak sadu stávajících her rozšířit o novou, říká zvláštní kapitola. Její součástí jsou i okomentované ukázky obsahu XML souborů, které jednotně definují každou hru a které vstupují do herní aplikace.

Implementaci této aplikace rozebírá samostatná kapitola, a to včetně podkapitoly o parsování, vyhodnocování a vykonávání pravidel za pomoci sady regulárních výrazů. Tato část je pro řízení průběhu hry klíčovou. Kapitola se také zabývá širokou škálou možných způsobů ovládání hry.

Rozdíly mezi těmito různými styly hraní byly demonstrovány na sadě naměřených údajů z několika testovacích partií. Z analýzy dat vyplývá, že i ovládání jediným vstupem je možné a srovnatelné s konvenčními metodami, a tedy že navržená koncepce pravidel, her, prostředí potažmo celé aplikace je správná.

V závěru byla naznačena případná vylepšení stávajícího návrhu. Na toto navazuje i následující oddíl.

DALŠÍ VÝVOJ

Otázka dalších vylepšení připadá v úvahu samozřejmě hlavně v případě, že se daná koncepce platformy *Inteligentního invalidního vozíku* a s ním spojených projektů ujme a aplikace se stane jeho žádoucí součástí. Směrů, kterými se lze vydat, existuje

více. Nejpodstatnějším vylepšením do budoucna by měl být generátor úrovní, o němž pojednává ***kapitola 8.3.4***.

Aplikace by též mohla pracovat i s hrami, které nebyly pro její účel přímo zapsány, ale které by respektovali navržené rozhraní. Základní podmínkou je, aby hra poskytovala běhovému prostředí přehled platných tahů, nebo alespoň dokázala odpovědět otázku platnosti ještě před provedením tahu. V tom případě by se aplikace postupně zeptala na všechna pole a z odpovědí by vygenerovala aktuální množinu platných tahů.

Hry lze sice velice jednoduše vytvářet, ovšem za pomoci textového či XML editoru. Aplikace v některé pokročilé fázi by mohla navíc obsahovat vizuální editor jak scény hry (hrací desky), tak i jejích pravidel.

A pak také rozšíření systému pravidel o obecnější typy her, než které byly vymezeny v ***kapitole 3.2***.

HLAVNÍ PŘÍNOSY PRÁCE

Do implementované aplikace lze velmi snadno přidávat množství nových her na základě pouhého definování jejich pravidel a vzhledu bez nutnosti každou hru realizovat samostatným programem. Navržený herní systém umožňuje velice snadné ovládání i handicapovanými uživateli. Pro ně je velkým přínosem i jednotné ovládání každé hry. Zaručený uniformní formát dat ukládaných při hraní umožňuje snadnou analýzu odehraných kol napříč všemi hrami.

Všechny body zadání byly tímto splněny. Nepsaným cílem této diplomové práce bylo přiblížit handicapovaným svět elektronických her, což věřím, že se mi alespoň z části podařilo. Je jen škoda, jak málo si zdravý člověk uvědomuje, že možnost trávit volný čas mimo obrazovku je něco, za co by jiní dali cokoli.

Literatura a reference

[1] PFEIFFER, Jan. Neurologie v rehabilitaci: pro studium a praxi. 1. Praha : Grada Publishing, 2007. 352 s. ISBN 978-80-247-1135-5.

[2] ELLIS, Barrie. *One Switch* [online]. 2010 [cit. 2010-04-16]. Switch Software Downloads. Dostupné z WWW: <http://www.oneswitch.org.uk/2/switch-downloads.htm>.

[3] Human-Computer Interaction Laboratory. *UA-Chess* [online]. April 23, 2007 [cit. 2010-04-16]. Play UA-Chess. Dostupné z WWW: <http://www.ics.forth.gr/hci/ua-games/ua-chess/play.html>.

[4] LOVE, Nathaniel; HINRICHS, Timothy; HALEY, David; SCHUKUFZA, Eric; GENESERETH Michael. General Game Playing : Game Description Language Specification [online]. California : Stanford University, March 4, 2008 [cit. 2010-04-07]. Dostupné z WWW: <http://games.stanford.edu/language/spec/gdl_spec_2008_03.pdf>.

[5] HRÁZKÝ, Lukáš. *Stolní multimediální hra Cetus navigátor*. Praha, 2008. 120 s. Diplomová práce. FEL ČVUT.

[6] Microsoft Corporation. *XNA* [online]. 2007 [cit. 2010-05-08]. Dostupné z WWW: <http://www.xna.com>.

[7] Emotiv Systems, Inc. *Emotiv* [online]. 2009 [cit. 2010-04-22]. EPOC Headset. Dostupné z WWW: <http://www.emotiv.com/apps/epoc/299>.

[8] GIPF. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 22. 9. 2008, last modified on 20. 10. 2009 [cit. 2010-04-18]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/GIPF>.

[9] LEE, Xah. *Xah Lee Web 李杀网* [online]. 2003-12 [cit. 2010-04-18]. Go Variations on Tilings. Dostupné z WWW: <http://xahlee.org/Periodic_dosage_dir/20031227_goboard.html>.

[10] NELSON, Toby; ABBOTT, Robert. *Logic Mazes* [online]. 2010 [cit. 2010-04-18]. Theseus and That Pesky Minotaur. Dostupné z WWW: <http://www.logicmazes.com/theseus.html>.

[11] Židi. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1. 4. 2006, last modified on 3. 3. 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/%C5%BDidi>.

[12] TĚŠÍNSKÝ, Jakub A. *Deskovehry.cz* [online]. 29. 6. 2007 [cit. 2010-04-25]. Návrhy metod deskripce a klasifikace her. Dostupné z WWW: <http://deskovehry.cz/index.php/N%C3%A1vrhy_metod_deskripce_a_klasifikace_her>.

[13] Microsoft Corporation. MSDN [online]. 2010 [cit. 2010-05-08]. C# Reference. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/618ayhy6>.

[14] Microsoft Corporation. .NET Framework Developer Center [online]. 2010 [cit. 2010-05-08]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/netframework/default.aspx>.

[15] List of XML and HTML character entity references. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5 July 2005 , last modified on 29 March 2010 [cit. 2010-05-24]. Dostupné z WWW: <http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references>.

[16] Microsoft Corporation. *MSDN* [online]. 2010 [cit. 2010-05-08]. Regular Expression Language Elements. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/az24scfc.aspx>.

[17] Category:NP-complete problems In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29 November 2004, 19 October 2009 [cit. 2010-05-01]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Category:NP-complete_problems>.

9.1 Použitý software

[a] Microsoft® Windows 7™ Professional (x64) CZ *(MSDN Academic Alliance)*

[b] Microsoft® Visual Studio® 2008 Professional Edition (x86 and x64 WoW) *(MSDN Academic Alliance)*

[c] RegexBuddy ver. 3.4.2

[d] SharpDevelop XmlSerialization Util

[e] CopySourceAsHtml ver. 3.0.0

[f] Microsoft® Office Word® 2007

[g] Microsoft® Office Excel® 2007

[h] Microsoft® Windows® SDK for Windows 7™ and .NET Framework® 3.5 SP1 *(MSDN Academic Alliance)*

[i] Imagicon ver. 3.4

[j] Nero® Burning ROM™ ver. 9.4.26.0

[k] ACDSee™ Pro ver. 3.0.386

[l] Microsoft® Malování ver. 6.1.7600

PŘÍLOHY

Příloha A

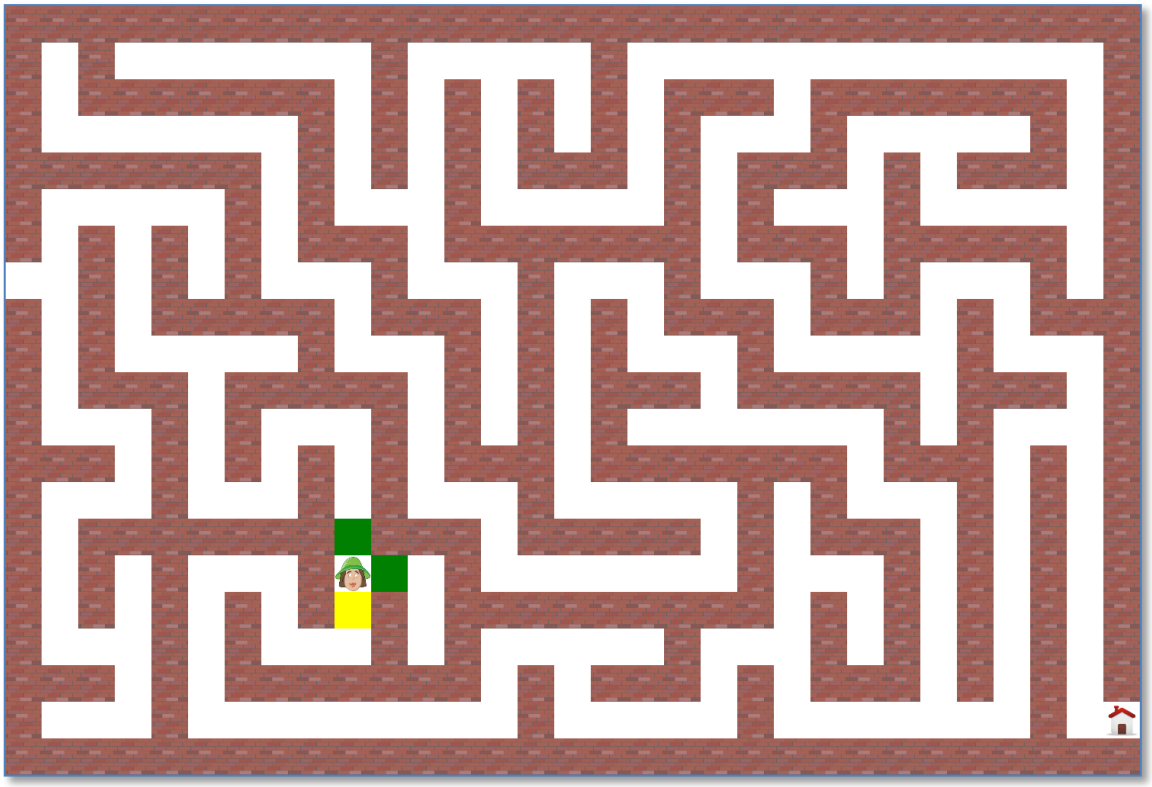
Běžně se v přílohách uvádí pouze pár příkladů konkrétní realizace. Jelikož ale zápis pravidel dle navrženého systému nemusí být vždy intuitivní, jsou zde pro lepší pochopení všech skýtaných možností uvedeny hry všechny.

Všechny realizovné hry

Následující přehled uvádí detailní popis všech 12 (13) realizovaných her, včetně zdrojů, ze kterých bylo čerpáno při jejich sestavování. Pokud není zdroj uveden, jedná se o vlastní tvorbu. U pravidel je pro úspornější vyjádření červené podbarvení u tzv. breakujících pravidel, u ostatních zelené. Výpis je také zkrácen, pokud se pravidlo pouze opakuje pro jiné souřadnice (pro 4- a 8-okolí) nebo podobné kostičky. Pravidla jsou pro snazší pochopení popsána neformálně v první osobě.

Bludiště

Ukázka



Obrázek 4: ***Ukázka ze hry Bludiště***

Úkol (cíl hry)

Nalezněte co nejkratší cestu domů.

Pravidla

Zaměň cílové pole za kostičku na stávající pozici, tedy za mě samého (Avatara).

```
STATE[A.x, A.y] = STATE[P.x, P.y];
```

Tam, kde jsem původně stál, vlož cestu (prázdné pole).

```
STATE[P.x, P.y] = Path;
```

4-okolí mé původní pozice označ jako neplatný další tah.

```
VALID[P.x + 1, P.y] = false;
…
```

Pokud je 4-okolí nové pozice cestou nebo domečkem (je různé od zdi), označ ho jako platný další tah.

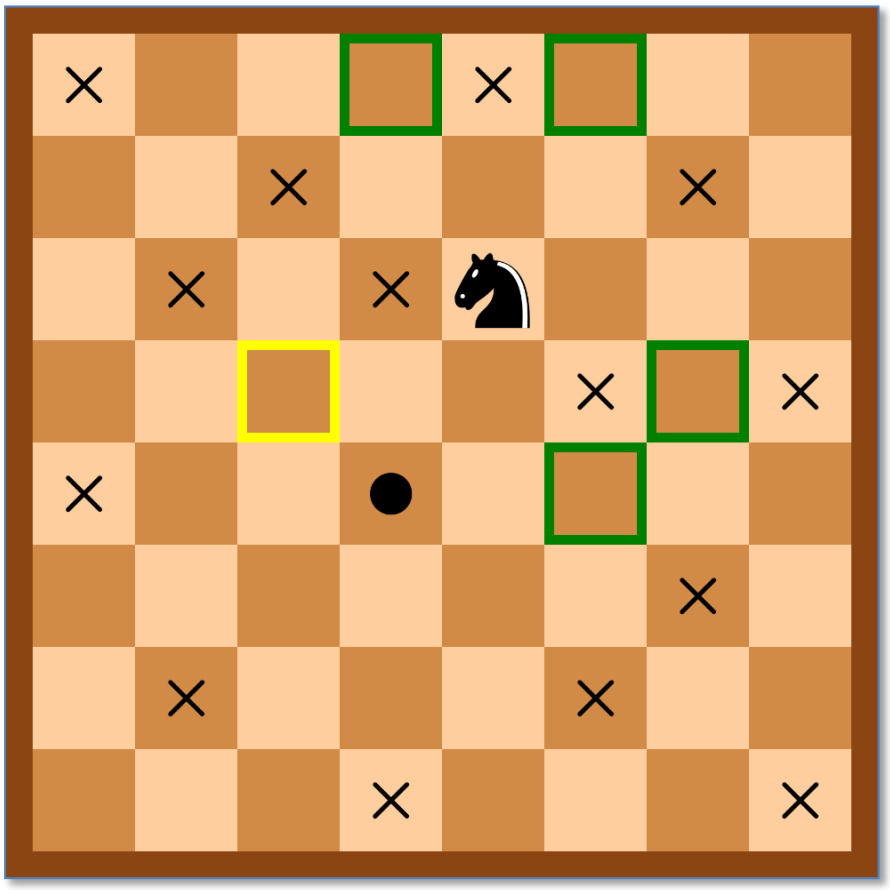
```
if (STATE[A.x + 1, A.y] != Wall)
{
     VALID[A.x + 1, A.y] = true;
}
…
```

Zdroje

- Idea a počáteční rozvržení úrovně[19]
- Grafické soubory
    a. miniatura[20]
    b. kostičky[18], [21], [63]

Jezdcova procházka

Ukázka



Obrázek 5: *Ukázka ze hry Jezdcova procházka*

Úkol (cíl hry)

Vyjeďte jezdcem ze startovního pole a postupně s ním projděte všechna prázdná pole šachovnice tak, abyste skončili opět na výchozím poli. Každé pole musí být navštíveno právě jednou a jezdec smí táhnout jen v souladu s šachovými pravidly.

Pravidla

Nejprve odznačení všech platných tahů ze stávající pozice, v dalším tahu budou jiná.

Postupně tedy zkus odznačit všech až 8 původních možných tahů jezdce, neboli o +/-2 a +/-1 (viz šachová pravidla pro táhnutí jezdcem).

```
     VALID[P.x + 1, P.y - 2] = false;
```

Další pozice k odznačení.

```
     VALID[P.x + 2, P.y - 1] = false;
     …
```

Naopak označ jako platné tahy ty, které od nové pozice splňují stejná pravidla pro táhnutí koněm. Může jít libovolně o černé nebo bílé pole.

```
     if ((STATE[A.x + 1, A.y - 2] == Black) OR (STATE[A.x +
1, A.y - 2] == White))
     {
          VALID[A.x + 1, A.y - 2] = true;
     }
```

Další pozice k označení.

```
     if ((STATE[A.x + 2, A.y - 1] == Black) OR (STATE[A.x +
2, A.y - 1] == White))
     {
          VALID[A.x + 2, A.y - 1] = true;
     }
     …
```

Zvláštní úlohu má výchozí bod celé hry, který když opouštím, označím jej tečkou. Pokud táhnu na černé pole, pravidla nepřipouštějí jinou variantu, než že jsem původně stál na bílém poli, tudíž do A uložím sebe na černém poli a do výchozího bodu tečku na bílém.

```
     if ((STATE[A.x, A.y] == Black) AND (STATE[P.x, P.y] ==
KnightOnWhiteStart))
     {
          STATE[P.x, P.y] = WhiteStart;
          STATE[A.x, A.y] = KnightOnBlack;
     }
```

Pokud netáhnu ze startovního bodu, ale z běžné bílé, uložím původní pozice křížek na bílém poli a sebe (jezdce) na černém poli do A.

```
     if ((STATE[A.x, A.y] == Black) AND (STATE[P.x, P.y] !=
KnightOnWhiteStart))
     {
          STATE[P.x, P.y] = VisitedWhite;
          STATE[A.x, A.y] = KnightOnBlack;
     }
```

Variace pro bílou barvu obou výše uvedených pravidel.

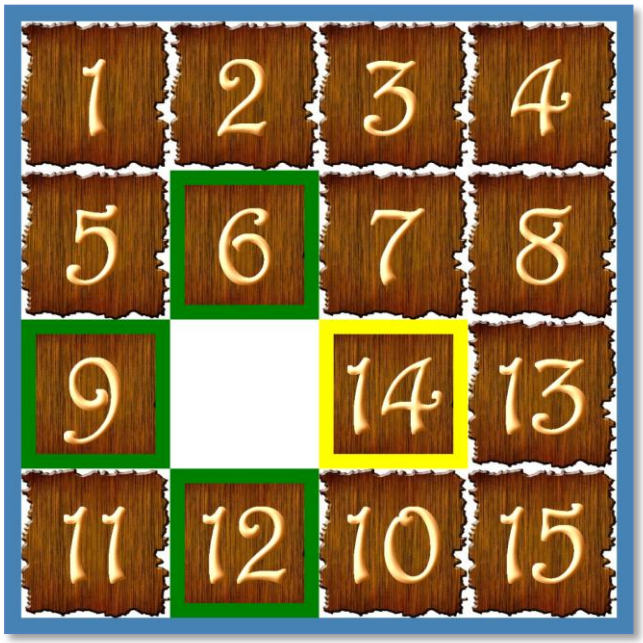
```
     if ((STATE[A.x, A.y] == White) AND (STATE[P.x, P.y] ==
KnightOnBlackStart))
     {
          STATE[P.x, P.y] = BlackStart;
          STATE[A.x, A.y] = KnightOnWhite;
     }
     …
```

Zdroje

- Idea a počáteční rozvržení úrovně[22]
- Grafické soubory
    a. miniatura[23]
    b. kostičky[24]

Loydova patnáctka

Ukázka



Obrázek 6: *Ukázka ze hry Loydova patnáctka*

Úkol (cíl hry)

Seřaďte všechna čísla vzestupně od 1 po 15 tak, aby poslední pole zůstalo volné. Čísla lze přesouvat pouze přes prázdné pole a v ortogonálním (kolmém) směru.

Pravidla

Záměna kostiček na stávající a nové pozici.

Do stávající pozice (bílé prázdné pole) ulož kostičku, na kterou jsem táhl (může být libovolné číslo).

```
STATE[P.x, P.y] = STATE[A.x, A.y];
```

Na místo, kam táhnu ulož prázdno.

```
STATE[A.x, A.y] = Empty;
```

4-okolí původní pozice označ jako neplatný další tah.

```
VALID[P.x + 1, P.y] = false;
…
```

Naopak 4-okolí nové pozice označ jako platné.

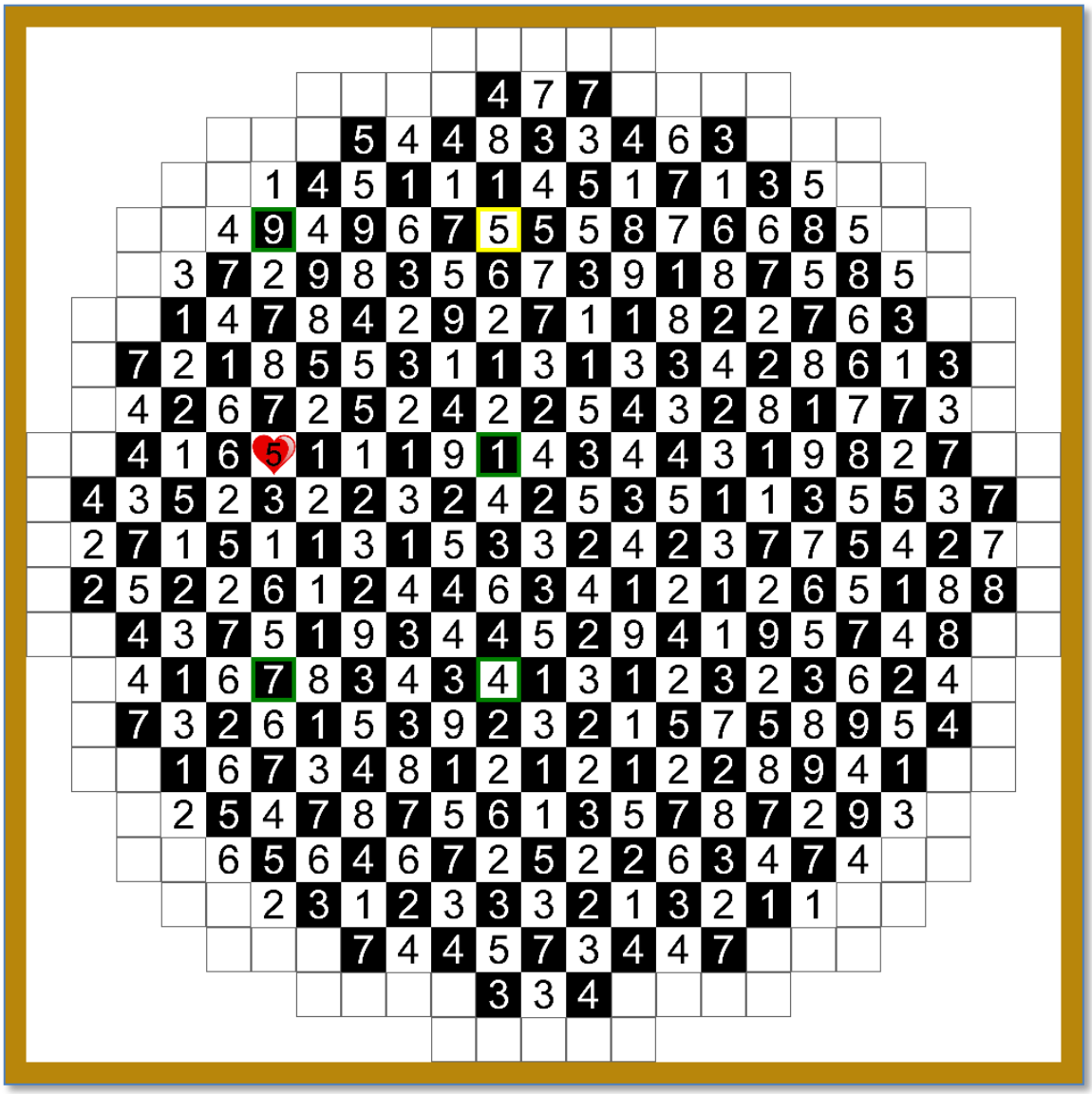
```
VALID[A.x + 1, A.y] = true;
…
```

Zdroje

- Idea a počáteční rozvržení úrovně[25]
- Grafické soubory
    a. miniatura[26]
    b. kostičky[18], [27]

Návrat z Klondiku

Ukázka



Obrázek 7: *Ukázka ze hry Návrat z Klondiku*

Úkol (cíl hry)

Nacházíte se uprostřed Klondiku (označení srdcem). Vaším úkolem je dostat se těsně za jeho hranice. V každém kole můžete táhnout v libovolném z osmi směrů (S, J, V, Z, SV, SZ, JV, JZ) o takový počet kroků, jako má hodnotu číslo, na němž právě stojíte. Abyste Klondike opustili, musíte skončit přesně jeden krok za hranicemi.

Pravidla

Nejprve musím zjistit, jaké číslo opouštím, abych na své původní místo mohl uložit variantu tohoto čísla bez symbolu srdce.

Pokud jsem byl na černém čísle 1 (navštívené číslo 1 se srdcem), ulož do pozice nenavštívené číslo (bez srdce).

```
     if (STATE[P.x, P.y] == BlackNumber1Visited)
     {
          STATE[P.x, P.y] = BlackNumber1;
     }
```

Stejně pokračuj i pro zbývající čísla do 9.

```
     if (STATE[P.x, P.y] == BlackNumber2Visited)
     {
          STATE[P.x, P.y] = BlackNumber2;
     }
     …
```

A také pro bílé varianty čísel od 1 po 9.

```
     if (STATE[P.x, P.y] == WhiteNumber1Visited)
     {
          STATE[P.x, P.y] = WhiteNumber1;
     }
     …
```

Dále musím zrušit 8-okolí původní pozice pro platnost dalšího tahu.

Pokud jsem tedy stál na 1, ať už bílé nebo černé, okolí +/-1 označím jako neplatné.

```
     if ((STATE[P.x, P.y] == BlackNumber1) OR (STATE[P.x,
P.y] == WhiteNumber1))
     {
          VALID[P.x + 1, P.y] = false;
          VALID[P.x - 1, P.y] = false;
          VALID[P.x, P.y + 1] = false;
          VALID[P.x, P.y - 1] = false;
          VALID[P.x + 1, P.y + 1] = false;
          VALID[P.x + 1, P.y - 1] = false;
          VALID[P.x - 1, P.y + 1] = false;
          VALID[P.x - 1, P.y - 1] = false;
     }
```

Stejně tak pokud jsem byl na 2… až opět po 9.

```
     if ((STATE[P.x, P.y] == BlackNumber2) OR (STATE[P.x,
P.y] == WhiteNumber2))
     {
          VALID[P.x + 2, P.y] = false;
          VALID[P.x - 2, P.y] = false;
          VALID[P.x, P.y + 2] = false;
          VALID[P.x, P.y - 2] = false;
          VALID[P.x + 2, P.y + 2] = false;
          VALID[P.x + 2, P.y - 2] = false;
          VALID[P.x - 2, P.y + 2] = false;
          VALID[P.x - 2, P.y - 2] = false;
     }
     …
```

Nastavení nových platných tahů. Platný je tah pouze na číslo nebo těsně za hranici Klondiku. Každá ze souřadnic je rozepsaná zvlášť pro případ, že by došlo k výjimce (souřadnice odkazuje mimo pole). Pak by totiž zkolabovalo celé pravidlo a zbývající souřadnice by nebyly správně označeny jako platné.

Pokud místo, kam táhnu, je 1, označ 8-okolí různé od prázdna jako platné.

```
     if ((STATE[A.x, A.y] == WhiteNumber1) AND (STATE[A.x +
1, A.y] != Empty))
     {
          VALID[A.x + 1, A.y] = true;
     }
```

Stejně tak pokud táhnu na 2. Při číslech vyšších než 1 je však situace komplikovanější, neboť kvůli zubaté hranici by se mohlo stát, že číslo sice odkazuje na destinaci, ale mezi ní a číslem je ještě jedna nebo více destinací, což je v rozporu s pravidly. Musím totiž bezpodmínečně opustit pole tahem právě těsně za hranici. Proto u čísel vyšších než 1 je přidána ještě jedna podmínka, která testuje, zda pozice o 1 blíž není destinace. Pokud ano, je to právě situace výše popsaná a tento tah se neoznačí jako platný.

```
     if ((STATE[A.x, A.y] == WhiteNumber2) AND (STATE[A.x +
2, A.y] != Empty) AND (STATE[A.x + 1, A.y] != Destination))
     {
          VALID[A.x + 2, A.y] = true;
     }
     …
```

Černá čísla jsou rozepsána zvlášť, nebo nelze kombinovat operátory AND a OR.

```
     …
     if ((STATE[A.x, A.y] == BlackNumber9) AND (STATE[A.x -
9, A.y - 9] != Empty) AND (STATE[A.x - 8, A.y - 8] !=
Destination))
     {
          VALID[A.x - 9, A.y - 9] = true;
     }
```

Nakonec zaměním cílové pole za jeho navštívenou variantu (se srdcem).

Pokud táhnu na černou 1, zaměním za navštívenou černou 1.

```
     if (STATE[A.x, A.y] == BlackNumber1)
     {
          STATE[A.x, A.y] = BlackNumber1Visited;
     }
     …
```

Stejně i pro bílá čísla. Breaknutí zde nemá příliš význam, spíš je to snaha o drobné urychlení.

```
     …
     if (STATE[A.x, A.y] == WhiteNumber9)
     {
          STATE[A.x, A.y] = WhiteNumber9Visited;
     }
```

Nakonec pokud táhnu na cílové pole (destinaci), zaměň jej za mě (srdce).

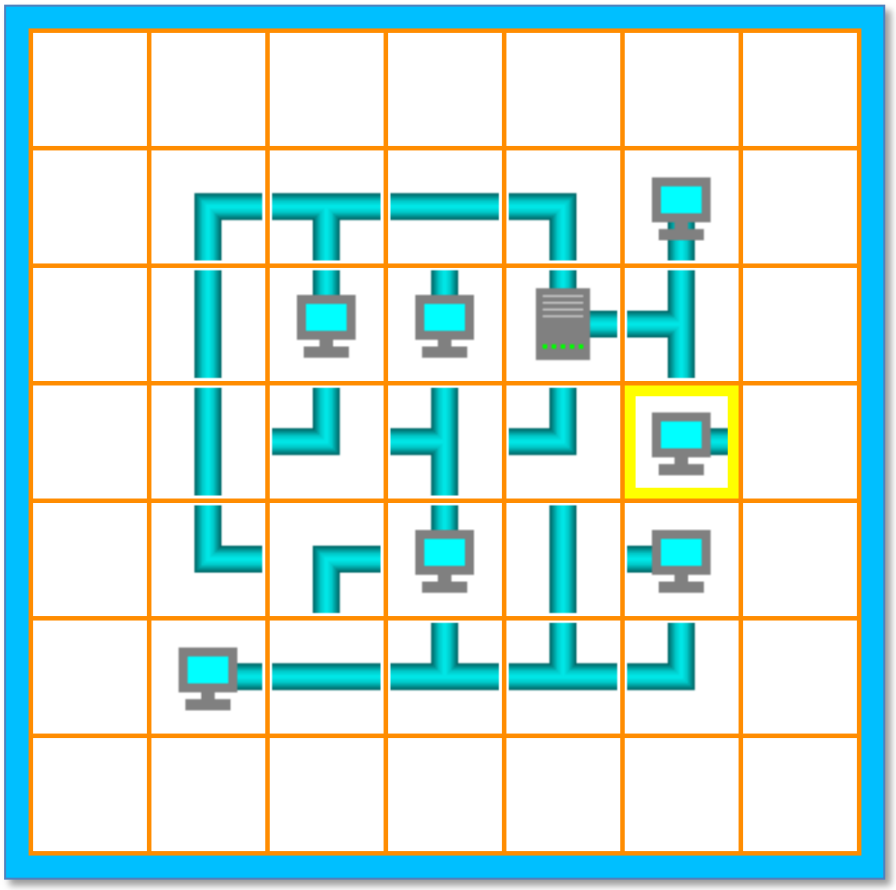
```
     if (STATE[A.x, A.y] == Destination)
     {
          STATE[A.x, A.y] = DestinationVisited;
     }
```

Zdroje

- Idea a počáteční rozvržení úrovně[28]
- Grafické soubory
    a. miniatura[29]
    b. kostičky[30]

Netwalk

Ukázka



Obrázek 8: *Ukázka ze hry Netwalk*

Úkol (cíl hry)

Propojte všechny počítače se serverem. Žádná z "datových rour" nesmí ústit do prázdna. Po vybrání jednotlivé části rotují proti směru hodinových ručiček, tím měníte cestu "datového toku". Pro nejtěžší úrovně mohou trubky přecházet přes hranici pole a navázat spojení na opačné straně.

Pravidla

Při výběru libovolné kostičky ji otoč o 90°, neboli zaměň obrázek v pořadí. Všechna pravidla jsou breakovací, neboť vždy mohu vybrat pouze jednu kostičku a pokud ji najdu (je splněna podmínka), o jinou již jít nemůže a stejně jako u Sudoku nechci, aby kvůli okamžitému vykonávání akcí kostička „probublala“ k jedinému stavu.

Pokud jsem vybral počítač otočený rourou na sever, otoč ji na západ.

```
if (STATE[A.x, A.y] == ComputerA1)
{
     STATE[A.x, A.y] = ComputerA2;
}
```

Ze západu na jih…

```
if (STATE[A.x, A.y] == ComputerA2)
{
     STATE[A.x, A.y] = ComputerA3;
}
```

Z jihu na východ…

```
if (STATE[A.x, A.y] == ComputerA3)
{
     STATE[A.x, A.y] = ComputerA4;
}
```

A z východu zpátky na sever.

```
if (STATE[A.x, A.y] == ComputerA4)
{
     STATE[A.x, A.y] = ComputerA1;
}
```

Stejné pravidlo rotace uplatni i pro všechny druhy rour.

```
if (STATE[A.x, A.y] == PipeB1)
{
     STATE[A.x, A.y] = PipeB2;
}
…
if (STATE[A.x, A.y] == PipeC1)
{
     STATE[A.x, A.y] = PipeC2;
}
…
if (STATE[A.x, A.y] == PipeD1)
{
     STATE[A.x, A.y] = PipeD2;
}
```

I všechny typy serverů.

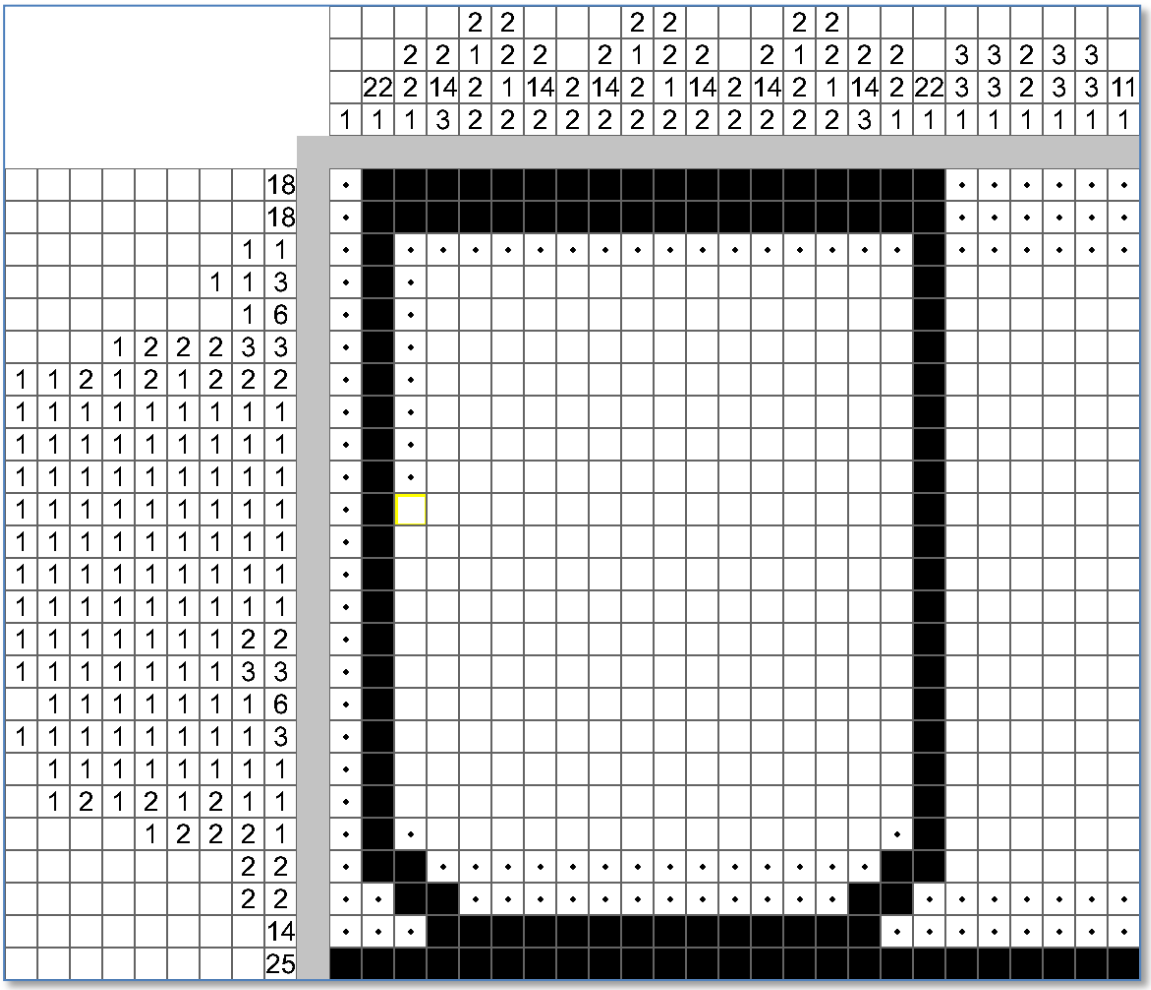
```
if (STATE[A.x, A.y] == ServerA1)
{
     STATE[A.x, A.y] = ServerA2;
}
…
if (STATE[A.x, A.y] == SeverB1)
{
     STATE[A.x, A.y] = SeverB2;
}
…
if (STATE[A.x, A.y] == ServerC1)
{
     STATE[A.x, A.y] = ServerC2;
}
…
if (STATE[A.x, A.y] == ServerD1)
{
     STATE[A.x, A.y] = ServerD2;
}
…
```

Zdroje

- Idea a počáteční rozvržení úrovně[31]
- Grafické soubory
    a. miniatura[32]
    b. kostičky[31]

Nonogramy

Ukázka



Obrázek 9: *Ukázka ze hry Nonogramy*

Úkol (cíl hry)

Odhalte obrázek, který je zakódovaný do čísel na začátku řádků a v záhlaví sloupců. Každé číslo udává počet za sebou jdoucích vyplněných čtverečků. Pokud je v jednom řádku nebo sloupci více čísel, je mezi těmito úseky alespoň jedno prázdné políčko. Mezi okrajem a prvním, či posledním úsekem mezery být mohou, ale nemusí.
Po každém vybrání čtverečku se změní jeho obsah. Od prázdného přes označené tečkou až po vyplněné. Bílá barva značí ještě neurčené pole, tečkou si označte ta pole, kde má být mezera a černá pak značí vyplněné pole. Abyste obrázek úspěšně dokončili, nesmí na hrací ploše zbýt žádné neurčené pole, musíte tedy tečkou označit všechny mezery.

Pravidla

Prostá záměna prázdna za tečku…

```
if (STATE[A.x, A.y] == Empty)
{
     STATE[A.x, A.y] = Point;
}
```

…tečky za výplň (černá barva)…

```
if (STATE[A.x, A.y] == Point)
{
     STATE[A.x, A.y] = Filled;
}
```

…a výplně opět za prázdno. Aby tah neprošel až k poslednímu, je každé breakovací.

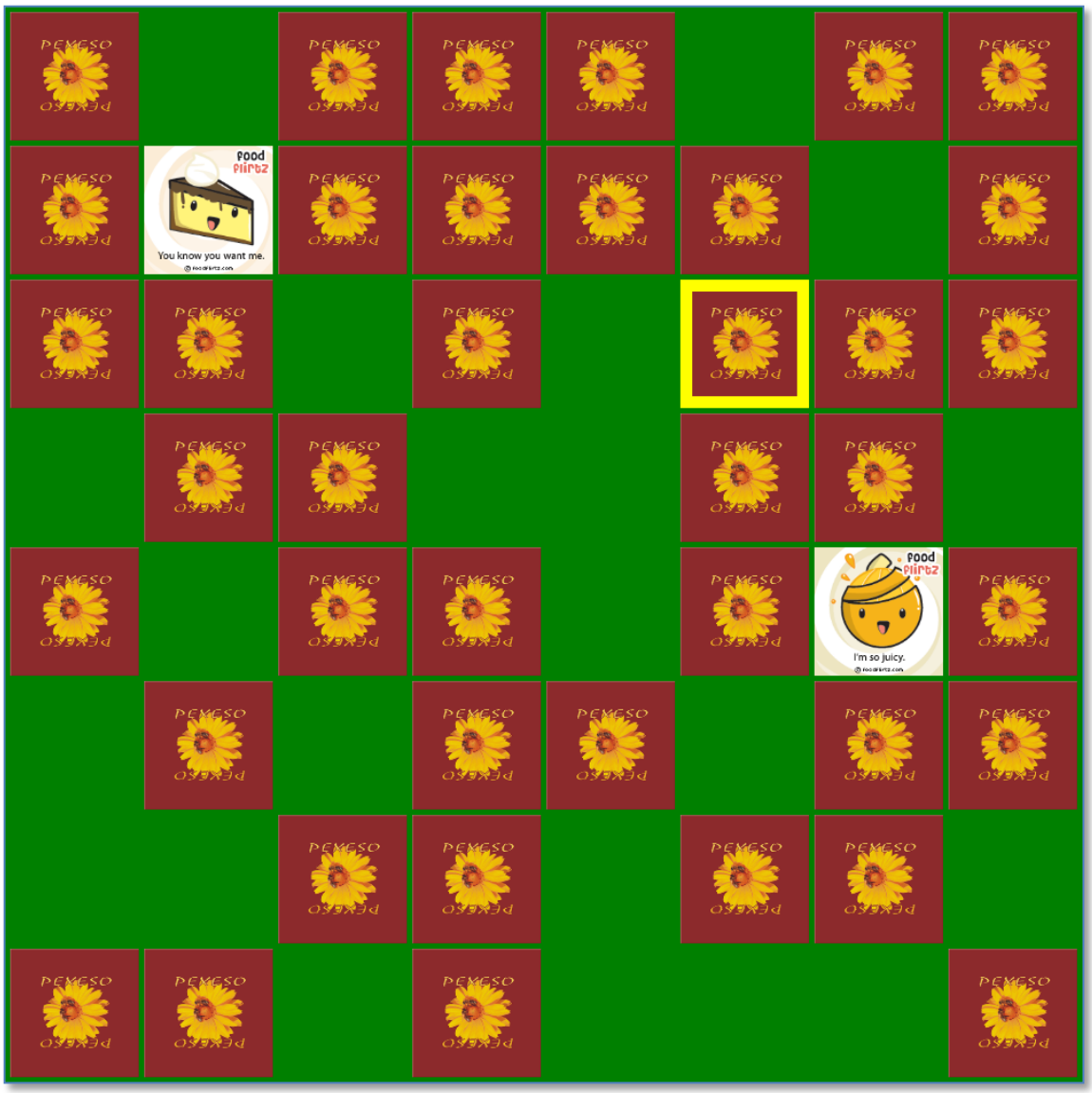
```
if (STATE[A.x, A.y] == Filled)
{
     STATE[A.x, A.y] = Empty;
}
```

Zdroje

- Idea a počáteční rozvržení úrovně[33]
- Grafické soubory
    a. miniatura[34]

Pexeso

Ukázka



Obrázek 10: *Ukázka ze hry Pexeso*

Úkol (cíl hry)

Odhalte všechny shodné dvojice obrázků. Otočte nejprve jeden obrázek a pak druhý. Za každou správnou dvojici získáte deset bodů. Za každou špatně otočenou dvojici se Vám jeden bod odečte. Takže... PEKelně SE SOustřeď!

Pravidla

Zatím jediná realizovaná hra, která využívá i jiné než systémové položky. Ač na první pohled velice jednoduchá hra, bylo nezbytné je dodat.

Nejprve zjistíme, kterou kostičku jsem vybral a její původní hodnotu (rub) si uložím do pomocné proměnné Before, neboli co zde bylo před mým tahem. Poté aktuálně táhnutou kartičku otočím, neboli uložím do pole A líc kartičky. Toto musím postupně otestovat na všech 32 možných karet.

```
if (STATE[A.x, A.y] == 01Back)
{
     Before = 01Back;
     STATE[A.x, A.y] = 01Front;
}
…
if (STATE[A.x, A.y] == 32Back)
{
     Before = 32Back;
     STATE[A.x, A.y] = 32Front;
}
```

Pokud v první pomocné proměnné FirstValue nic není, znamená to, že momentálně není otočená žádná kartička. Uložím si proto do této proměnné otočenou kartičku a zároveň si do další pomocné proměnné FirstBefore uložím, co bylo na místě této kartičky předtím - proměnnou Before. To pro případ, že se hráč nestrefí do dvojice a já budu muset kartičky opět otočit rubem navrch, takže potřebuji tuto informaci uchovat. Navíc si potřebuji zapamatovat, kde k tahu došlo, koordináty A.x a A.y si proto uložím do dalších pomocných proměnných. Nakonec označím aktuální tah jako neplatný, abych právě otočenou kartičku nemohl znovu vybrat. Mám protzatím první z dvojice kartiček a vyhodnocení pravidel ukončím.

```
if (FirstValue == nothing)
{
     FirstBefore = Before;
     FirstValue = STATE[A.x, A.y];
     FirstX = A.x;
     FirstY = A.Y;
     VALID[A.x, A.y] = false;
}
```

Pokud jsem se dostal sem, znamená to, že mám již otočenou jednu kartičku a nyní jsem otočil druhou. Vykonám stejné akce jako výše s tím rozdílem, že tentokrát si informace uchovávám do proměnných pro druhou kartičku. Opět zde ukončím.

```
if (SecondValue == nothing)
{
     SecondBefore = Before;
     SecondValue = STATE[A.x, A.y];
     SecondX = A.x;
     SecondY = A.Y;
     VALID[A.x, A.y] = false;
}
```

Dostal-li jsem se s vyhodnocováním až sem, znamená to, že mám otočeny dvě kartičky a potřebuji zjistit, zda jsou shodné. Pokud ano, přičtu 10 bodů za uhodnutou dvojici a na její místo uložím prázdná pole, která už až do konce hry zůstanou neplatnými tahy.

Pokud jsem odhalil špatný pár, odečtu jeden trestný bod a otočím je opět lícem dolů. K tomu mi poslouží právě pomocné proměnné First a Second, v kterých mám uchovány původní hodnoty polí i jejich souřadnice. Obě pole nakonec opět označím jako platné tahy.

```
if (FirstValue == SecondValue)
{
     Bodů = Bodů + 10;
     STATE[FirstX, FirstY] = Empty;
     STATE[SecondX, SecondY] = Empty;
}
else
{
     Bodů = Bodů - 1;
     STATE[FirstX, FirstY] = FirstBefore;
     STATE[SecondX, SecondY] = SecondBefore;
     VALID[FirstX, FirstY] = true;
     VALID[SecondX, SecondY] = true;
}
```

Předchozí pravidlo není breakovací, neboť potřebuji ještě zpracovat nově příchozí tah. Ten je nyní zákonitě opět první otočenou kartičkou, takže provedu stejné akce, jako u pravidla uvedeného výše, tentokrát ale již bezpodmínečně.

```
FirstBefore = Before;
…
FirstValue = STATE[A.x, A.y];
…
FirstX = A.x;
…
FirstY = A.Y;
…
VALID[A.x, A.y] = false;
```

Na závěr si vynuluji hodnotu v druhé pomocné proměnné, v které mám zatím hodnotu z předešlé dvojice. To aby se „chytlo“ druhé podmíněné pravidlo.

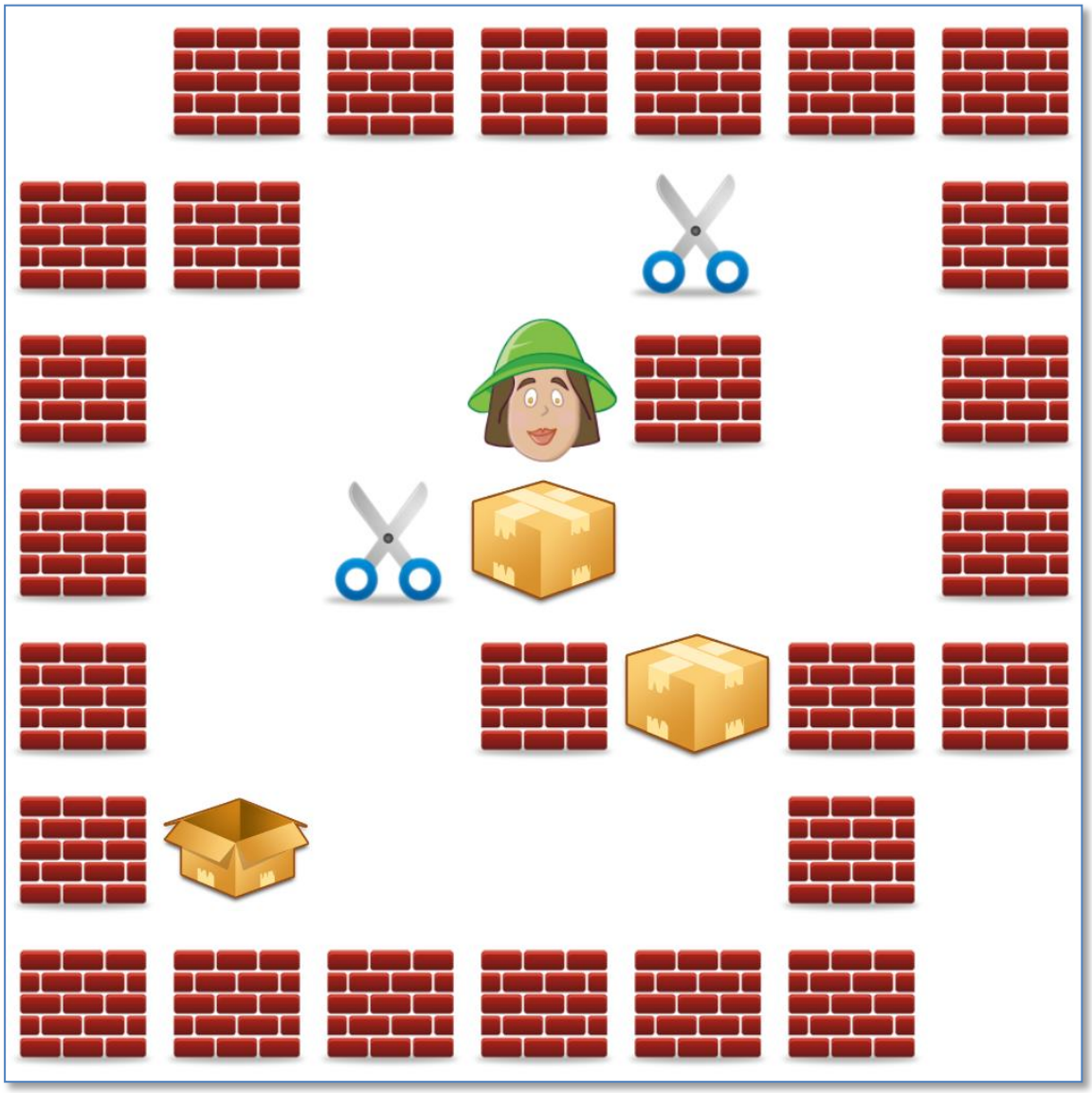
```
SecondValue = nothing;
```

Zdroje

- Idea[35]
- Grafické soubory
    a. miniatura[36]
    b. kostičky[37], [38]

Sokoban

Ukázka



Obrázek 11: *Ukázka ze hry Sokoban*

Úkol (cíl hry)

Dopravte všechny krabice na nůžky a tím je otevřete. Přes nůžky můžete volně přecházet. Stejně tak můžete již umístěnou krabici z nůžek odsunout, pak se ale opět zavře. Nelze přesouvat dvě krabice najednou, stejně tak není možné krabici zatlačenou do rohu nebo vysvobodit a je pak nutné začít hru od začátku.

Pravidla

4-okolí původní pozice označ jako neplatný další tah, byť směr, ze kterého přicházím, bude opět označen za platný. Zjištění tohoto směru by však bylo výpočetně náročnější, než když udělám tento jeden přebytečný krok.

```
VALID[P.x + 1, P.y] = false;
…
```

Pokud přicházím z prázdného pole (pouze Avatar) a táhnu také na prázdné pole, do cíle ulož mě (avatar) a do původní pozice prázdno.

```
     if ((STATE[A.x, A.y] == Empty) AND (STATE[P.x, P.y] ==
Avatar))
     {
          STATE[A.x, A.y] = Avatar;
          STATE[P.x, P.y] = Empty;
     }
```

Pokud přicházím z prázdného pole (pouze Avatar) a táhnu na nůžky, do cíle ulož mě na nůžkách (kostička avatar+nůžky) a do původní pozice prázdno.

```
     if ((STATE[A.x, A.y] == Scissors) AND (STATE[P.x, P.y]
== Avatar))
     {
          STATE[A.x, A.y] = AvatarOnScissors;
          STATE[P.x, P.y] = Empty;
     }
```

Další možnou kombinací je tah na prázdné pole z nůžek…

```
     if ((STATE[A.x, A.y] == Empty) AND (STATE[P.x, P.y] ==
AvatarOnScissors))
     {
          STATE[A.x, A.y] = Avatar;
          STATE[P.x, P.y] = Scissors;
     }
```

…nebo tah z nůžek na nůžky.

```
     if ((STATE[A.x, A.y] == Scissors) AND (STATE[P.x, P.y]
== AvatarOnScissors))
     {
          STATE[A.x, A.y] = AvatarOnScissors;
          STATE[P.x, P.y] = Scissors;
     }
```

Následující část popisuje každou z osmi kombinací, které mohou nastat, pokud posouvám krabici. Osm kombinací je dáno dvěma možnými stavy tří za sebou jdoucích kostiček (avatar/krabice/cílové pole). Můžu buď stát na prázdném poli, nebo na nůžkách, tlačená krabice mohla stát také buď na prázdném poli, nebo na nůžkách a konečně krabici můžu tlačit buď na prázdné pole, nebo na pole s nůžkami.

První z možností, kdy všechny tři kostičky jsou bez nůžek. Do původní pozice ulož prázdno, do A ulož mě a do pole o jedno dál ulož krabici. Aby se obešlo zdlouhavé zjišťování cíle 4-okolí, je zde použito chytrého určení cílového pole pro krabici přičtením rozdílu A - P ke stávající pozici A v obou složkách x a y. To zaručí, že se vždy správně ve výsledku přičte nebo odečte jednička.

```
     if ((STATE[P.x, P.y] == Avatar) AND (STATE[A.x, A.y] ==
BoxClosed) AND (STATE[A.x + A.x - P.x, A.y + A.y - P.y] ==
Empty))
     {
          STATE[P.x, P.y] = Empty;
          STATE[A.x, A.y] = Avatar;
          STATE[A.x + A.x - P.x, A.y + A.y - P.y] =
BoxClosed;
     }
     …
```

Poslední z osmi možností, kdy všechna tři pole obsahují nůžky. Do původního pole ulož nůžky, do A mě na nůžkách a do třetího krabici na nůžkách, neboli otevřenou krabici.

```
     if ((STATE[P.x, P.y] == AvatarOnScissors) AND
(STATE[A.x, A.y] == BoxOpened) AND (STATE[A.x + A.x - P.x,
A.y + A.y - P.y] == Scissors))
     {
          STATE[P.x, P.y] = Scissors;
          STATE[A.x, A.y] = AvatarOnScissors;
          STATE[A.x + A.x - P.x, A.y + A.y - P.y] =
BoxOpened;
     }
```

Nakonec je třeba určit nové platné tahy.

Pokud jsou 4-okolí nové pozice buď nůžky anebo prázdno, smím tam táhnout.

```
     if ((STATE[A.x + 1, A.y] == Empty) OR (STATE[A.x + 1,
A.y] == Scissors))
     {
          VALID[A.x + 1, A.y] = true;
     }
     …
```

Pokud je ve 4-okolí nové pozice zavřená krabice a o jedno políčko za ní není zeď nebo jiná zavřená nebo otevřená krabice, znamená to, že je v příštím tahu mohu na toto místo přetlačit.

```
     if ((STATE[A.x + 1, A.y] == BoxClosed) AND (STATE[A.x +
2, A.y] != Wall) AND (STATE[A.x + 2, A.y] != BoxClosed) AND
(STATE[A.x + 2, A.y] != BoxOpened))
     {
          VALID[A.x + 1, A.y] = true;
     }
     …
```

Totéž platí, pokud je ve 4-okolí již otevřená krabice, na možnou manipulaci s ní to nemá žádný vliv.

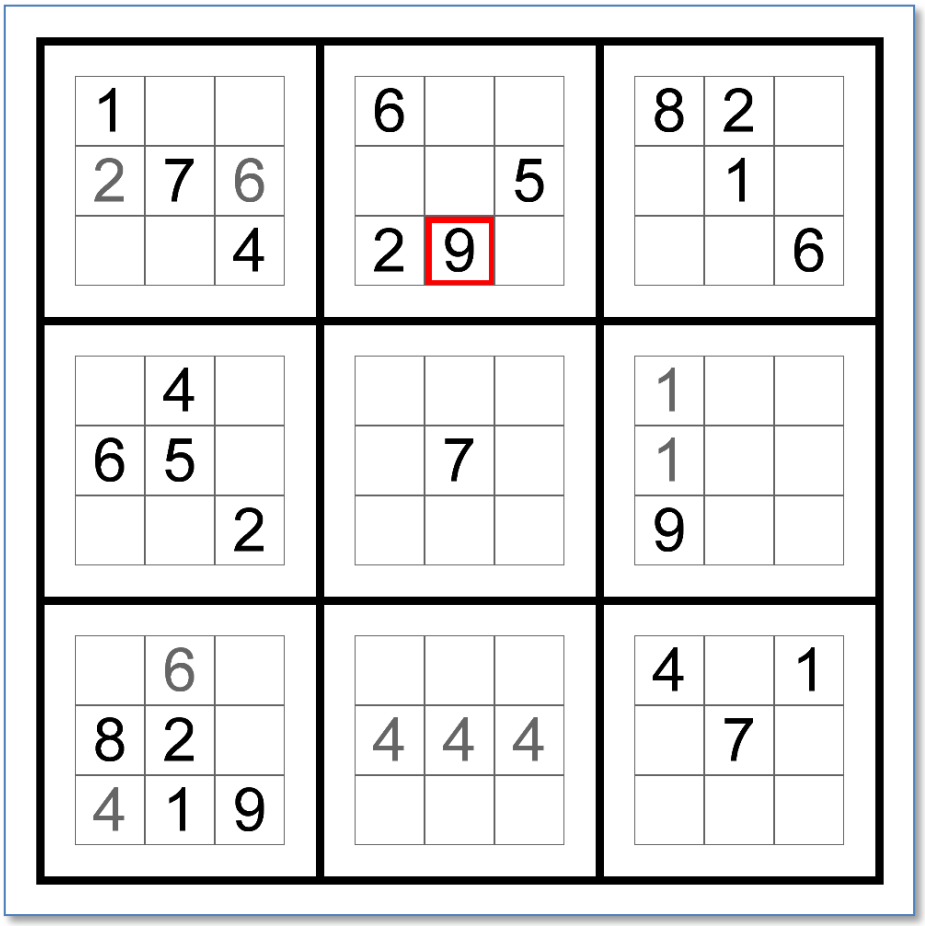
```
     if ((STATE[A.x + 1, A.y] == BoxOpened) AND (STATE[A.x +
2, A.y] != Wall) AND (STATE[A.x + 2, A.y] != BoxClosed) AND
(STATE[A.x + 2, A.y] != BoxOpened))
     {
          VALID[A.x + 1, A.y] = true;
     }
     …
```

Zdroje

- Idea a počáteční rozvržení úrovně[39]
- Grafické soubory
    a. miniatura[40]
    b. kostičky[18], [41], [63]

Sudoku

Ukázka



Obrázek 12: *Ukázka ze hry Sukoku*

Úkol (cíl hry)

Do prázdných polí herní plochy umístěte čísla v rozsahu 1 až 9. V žádném z 9 menších čtverců, řádku ani sloupci se nesmí opakovat žádné z čísel. Jinak řečeno v každém z řádků, sloupců a menších čtverců musí být všechna čísla od 1 po 9.

Pravidla

Postupně rotuj čísla od 1 po 9 až opět k prázdnému poli.

Pokud zvolím prázdné pole, ulož do něj číslo 1 (přičte se jednička). Jelikož se záměna polí děje okamžitě, je důležité, aby po každé splněné podmínce nastal break, jinak bych totiž vždy skončil až u posledního pravidla a do políčka by se uložilo prázdno.

```
if (STATE[A.x, A.y] == Empty)
{
     STATE[A.x, A.y] = OwnNumber1;
}
```

Obdobně přičti 1 ke každému dalšímu číslu.

```
if (STATE[A.x, A.y] == OwnNumber1)
{
     STATE[A.x, A.y] = OwnNumber2;
}
```

Nakonec pokud jsem zvolil číslo 9, ulož do něj prázdno (výchozí u všech čísel).

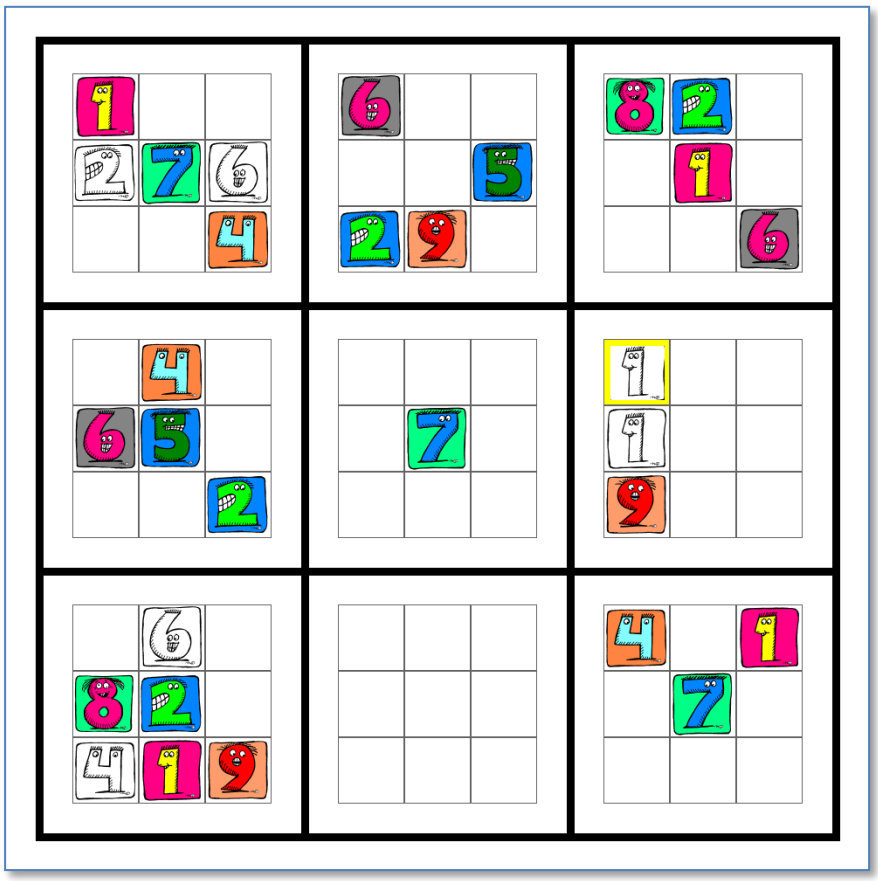
```
if (STATE[A.x, A.y] == OwnNumber9)
{
     STATE[A.x, A.y] = Empty;
}
```

Zdroje

- Idea a počáteční rozvržení úrovně[42]
- Grafické soubory
    a. miniatura[43]

Sudoku pro děti

Ukázka



Obrázek 13: ***Ukázka ze hry Sudoku pro děti***

Úkol (cíl hry) a Pravidla

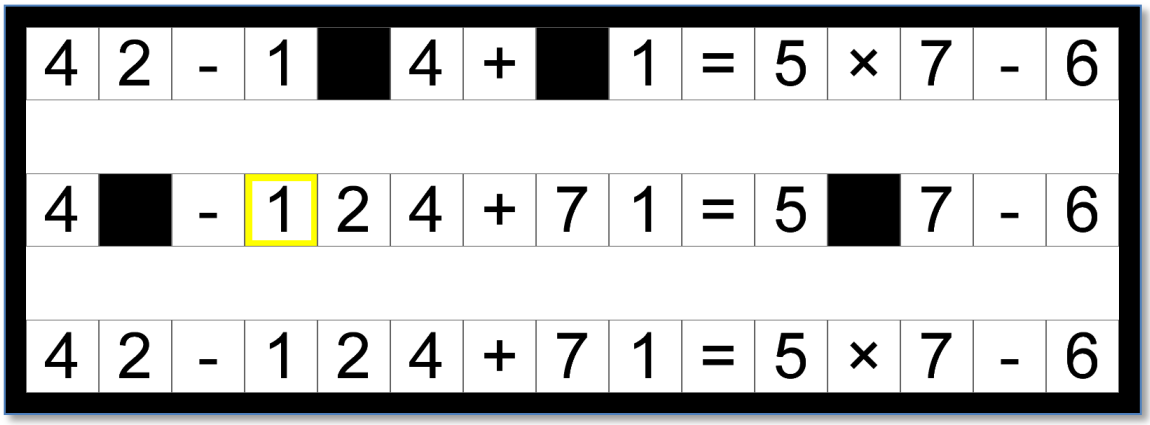
Obojí a část Zdrojů je totožná s klasickou verzí Sudoku uvedenou výše. Tato hra má pouze demonstrovat snadnou modifikovatelnost pomocí záměny obrázků kostiček.

Zdroje

- Grafické soubory
    a. miniatura[44]
    b. kostičky[45]

Temné rovnice

Ukázka



Obrázek 14: *Temné rovnice*

Úkol (cíl hry)

Ze zadané nerovnice utvořte tři různé rovnice zakrytím vybraných operací nebo čísel. V prvním řádku zakryjte dvě políčka, v prostředním tři a v posledním čtyři.
Pořadí operací je standardní: násobení a dělení má přednost před sčítáním a násobením. Počítá se pouze s celými čísly, výsledek může být i záporný.

Pravidla

Prosté zaměňování čísel a operací za jejich zatmavenou verzi a naopak.

Pokud jsem zvolil číslo 0, zatemni jej. Pokud je to pravda, přeruš vyhodnocení pravidel, žádná jiná možnost to již být nemůže.

```
if (STATE[A.x, A.y] == Number0)
{
    STATE[A.x, A.y] = Number0Filled;
}
…
```

Naopak pokud zvolím „rub“ čísla 0, zviditelni jej zpátky. Podobně pro všechna ostatní čísla až po 9. Důvod, proč tato část není spojená s předchozím do bloku else je ten, že pravidlo by nebylo vyhodnoceno jako pravdivé a tudíž by neproběhl break a testovala by se i zbylá pravidla, i když je jasné, že nic jiného to již být nemůže.

```
if (STATE[A.x, A.y] == Number0Filled)
{
    STATE[A.x, A.y] = Number0;
}
…
```

Stejná akce platí i pro matematické operace.

```
if (STATE[A.x, A.y] == Plus)
{
    STATE[A.x, A.y] = PlusFilled;
}
```

Včetně jejich opětovného zviditelnění.

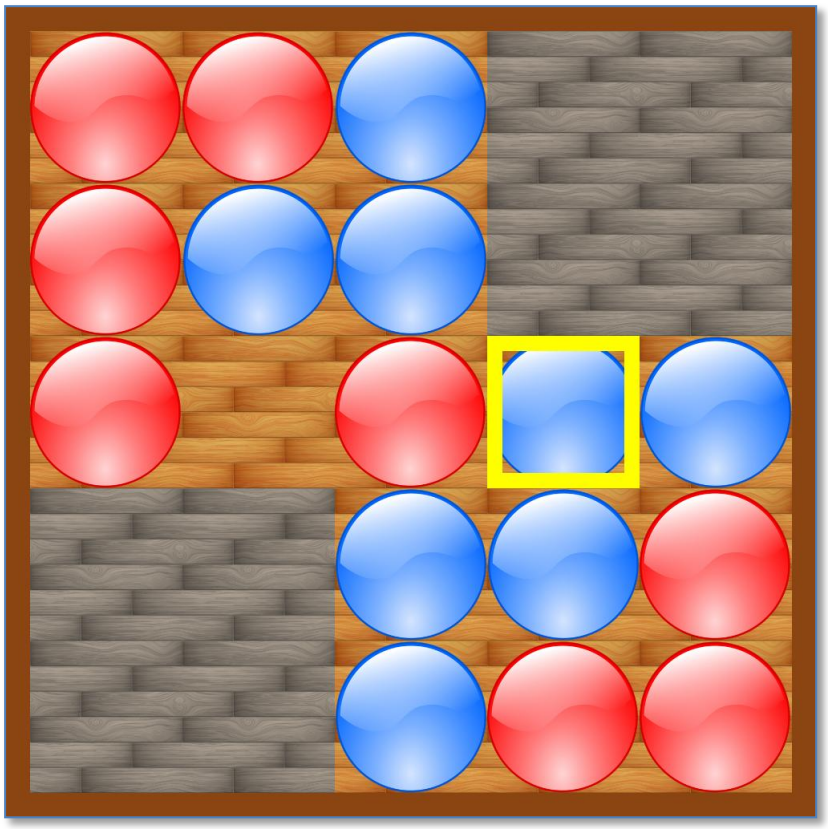
```
if (STATE[A.x, A.y] == PlusFilled)
{
    STATE[A.x, A.y] = Plus;
}
…
```

Zdroje

- Idea a počáteční rozvržení úrovně[46]
- Grafické soubory
    a. miniatura[47]

Viktoriánský hlavolam

Ukázka



Obrázek 15: *Ukázka ze hry Viktoriánský hlavolam*

Úkol (cíl hry)

Vyměňte červené a modré kameny. Kameny můžete posunovat pouze na sousední volné pole, a to i za pomoci přeskoku přes jiný kámen (jakékoli barvy), za nímž je další pole volné. Povoleny jsou pouze tahy v ortogonálním směru (přímo doleva, doprava, nahoru a dolů od volného pole).

Pravidla

Pravidla jsou podobná jako ve hře Loydova patnáctka.

Do stávající pozice (prázdné pole) ulož kostičku, na kterou jsem táhl (může být buď modrý nebo červený kámen).

```
STATE[P.x, P.y] = STATE[A.x, A.y];
```

Na místo, kam táhnu, ulož prázdno.

```
STATE[A.x, A.y] = Empty;
```

4-okolí původní pozice označ jako neplatný další tah.

```
VALID[P.x + 1, P.y] = false;
…
```

4-okolí původní pozice rozšířené o +1 označ jako neplatný další tah.

```
    VALID[P.x + 2, P.y] = false;
    …
```

Naopak 4-okolí nové pozice označ jako platné, pokud jde o červený nebo modrý kámen.

```
    if ((STATE[A.x + 1, A.y] == Blue) OR (STATE[A.x + 1,
A.y] == Red))
    {
        VALID[A.x + 1, A.y] = true;
    }
    …
```

Stejně tak i 4-okolí rozšířené o +1 označ za platné, pokud jde o kameny a ne prázdno.

```
    if ((STATE[A.x + 2, A.y] == Blue) OR (STATE[A.x + 2,
A.y] == Red))
    {
        VALID[A.x + 2, A.y] = true;
    }
    …
```

Zdroje

- Idea a počáteční rozvržení úrovně[48]
- Grafické soubory
    a. miniatura[48]
    b. kostičky[49], [50]

Zhasni

Ukázka

Obrázek 16: ***Ukázka ze hry Zhasni***

Úkol (cíl hry)

Zhasněte všechna světla. Po zvolení žárovky se přepne její stav a stav čtyř žárovek v jejím nejbližším okolí (vlevo, vpravo, nahoře a dole), pokud to hranice pole dovoluje.

Pravidla

Pokud jsem zvolil vypnuté světlo, zapnu jej. Pokud ne, je jasné, že jsem zvolil zapnuté světlo a pak jej vypnu.

```
if (STATE[A.x, A.y] == Off)
{
     STATE[A.x, A.y] = On;
}
else
{
     STATE[A.x, A.y] = Off;
}
```

Stejný postup aplikuji i na 4-okolí tahu.

```
if (STATE[A.x, A.y + 1] == On)
{
     STATE[A.x, A.y + 1] = Off;
}
else
{
     STATE[A.x, A.y + 1] = On;
…
```

Zdroje

- Idea a počáteční rozvržení úrovně[51]
- Grafické soubory
    a. miniatura[52]
    b. kostičky[53][54]

Zdroje výše uvedených her

[18] File:1x1.png In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-05-01]. Dostupné z WWW: <http://commons.wikimedia.org/wiki/File:1x1.png>.

[19] KENDRICK, Bill. *BillsGames.com* [online]. 1995-2009 [cit. 2010-05-05]. The Maze Generator. Dostupné z WWW: <http://www.billsgames.com/mazegenerator>.

[20] KENDRICK, Bill. *Online Generators* [online]. April 24, 2006 [cit. 2010-05-01]. Maze Generator. Dostupné z WWW: <http://online-generator.blogspot.com/2006/04/maze-generator.html>.

[21] Brain Waves LLC. *Clker.com* [online]. 2010 [cit. 2010-05-05]. Isometric Brick Tile clip art. Dostupné z WWW: <http://www.clker.com/clipart-isometric-brick-tile.html>.

[22] Jezdcova procházka. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14. 3. 2008, last modified on 13. 11. 2009 [cit. 2010-05-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Jezdcova_proch%C3%A1zka>.

[23] File:Chess ndt45.svg In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-05-05]. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:Chess_ndt45.svg>.

[24] Template:Chess diagram In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5 August 2005, 17 January 2010 [cit. 2010-05-05]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Template:Chess_diagram>.

[25] KÖLLER, Jürgen. *Mathematische Basteleien* [online]. 2000 [cit. 2010-05-05]. The 15 Puzzle. Dostupné z WWW: <http://www.mathematische-basteleien.de/15puzzle.htm>.

[26] File:15-puzzle.svg In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-05-01]. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:15-puzzle.svg>.

[27] Fotolia. *Fotolia* [online]. 2004-2010 [cit. 2010-05-01]. Wooden board. Dostupné z WWW: <http://en.fotolia.com/id/6437708>.

[28] LOYD, Sam. *Cyclopedia of Puzzles* [online]. New York : The Lamb Publishing Company, 1914 [cit. 2010-05-01]. Back From the Klondike, s. 384. Dostupné z WWW: <http://mathpuzzle.com/loyd/cop106-107.html>.

[29] Náš Tramp. *Picasa* [online]. 24.11.2009 [cit. 2010-04-25]. Klondike x logo. Dostupné z WWW: <http://picasaweb.google.cz/nastramp/K#5407895611982693826>.

[30] CKSinfo.com. *Free Clipart Junction* [online]. 2010 [cit. 2010-05-01]. Free Hearts Clipart Images, Graphics, Animated Gifs & Animations. Dostupné z WWW: <http://www.cksinfo.com/holidays/valentinesday/hearts/page4.html>.

[31] METZLER, Josh. *Logic Games Online* [online]. 2005-2010 [cit. 2010-05-01]. Netwalk. Dostupné z WWW: <http://www.logicgamesonline.com/netwalk>.

[32] IconPixel. *Graphics Design for Software Companies* [online]. 1999 - 2006 [cit. 2010-05-01]. Winning Software Graphic Design. Dostupné z WWW: <http://www.iconpixel.com>.

[33] PETŘÍČEK, Martin. *Kódované obrázky* [online]. 2006 [cit. 2010-04-25]. Kódované obrázky. Dostupné z WWW: <http://kod.petricek.net>.

[34] Infiworks. *ITunes* [online]. 2010 [cit. 2010-05-01]. A Griddler. Dostupné z WWW: <http://itunes.apple.com/us/app/a-griddler/id353887596?mt=8>.

[35] Pexeso. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9. 3. 2006, last modified on 2. 4. 2010 [cit. 2010-05-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Pexeso>.

[36] t-hry.cz. *T-hry* [online]. 2007 - 2010 [cit. 2010-05-05]. Pexeso Duel. Dostupné z WWW: <http://www.t-hry.cz/game.php?game=pexeso>.

[37] topbun.com. *TopBun* [online]. 2008 [cit. 2010-03-25]. Food Flirtz Myspace Comments. Dostupné z WWW: <http://www.topbun.com/myspace-graphics/47/page/1>.

[38] HALFROVÁ, Zuzana; MATELA, Lukáš. *Pexeso* [online]. 2004-2005 [cit. 2010-05-05]. Pexeso ke stažení. Dostupné z WWW: <http://pexeso.webzdarma.cz>.

[39] milaadesign.com. *Milaadesign.com* [online]. 2010 [cit. 2010-05-01]. Ledix. Dostupné z WWW: <http://www.milaadesign.com/Ledix.html>.

[40] STEIN, Patrick; LÄUFER, Florian. *Smart Sokoban* [online]. 21 Jul 2009 [cit. 2010-04-20]. Smart Sokoban Pro 1.9. Dostupné z WWW: <http://m.macupdate.com/info.php/id/28162/smart-sokoban-pro>.

[41] TPDK. *Tpdkdesign.net* [online]. November 23rd, 2005 [cit. 2010-04-20]. Refresh CL. Dostupné z WWW: <http://blog.tpdkdesign.net/2005/11/23/refresh-cl>.

[42] LabPixies. *LabPixies* [online]. 2009 [cit. 2010-05-01]. Sudoku Puzzles. Dostupné z WWW: <http://www.labpixies.com/campaigns/sudoku/sudoku.html>.

[43] Electronic Arts. *ISmashPhone* [online]. 2008/07 [cit. 2010-05-05]. EA's Sudoku iPhone App Will Blow Your Mind Away. Dostupné z WWW: <http://www.ismashphone.com/2008/07/eas-sudoku-ipho.html>.

[44] Popgadget Media. *PopGadget* [online]. 11. 09. 2007 [cit. 2010-05-01]. Handful of Sudoku fun: No pen or pencils needed. Dostupné z WWW: <http://www.popgadget.net/2007/11/handful_of_sudo.php>.

[45] HICKS, Mark A. *Discovery Education* [online]. 2010 [cit. 2010-05-01]. Letters & Numbers. Dostupné z WWW: <http://school.discoveryeducation.com/clipart/category/letr.html>.

[46] FRIEDMAN, Erich. *Erich's Place* [online]. 2009 [cit. 2010-05-05]. Blackout Puzzles. Dostupné z WWW: <http://www2.stetson.edu/~efriedma/blackout>.

[47] WINER, Martin C. *Martin C. Winer* [online]. March 9th 2010 [cit. 2010-04-25]. Helpful Math Typesetting Tool – Prime Twin Counting Function Example. Dostupné z WWW: <http://www.martincwiner.com/helpful-math-typesetting-tool-prime-twin-counting-function-example>.

[48] LADA a MAREK. *Klub přátel deskových her* [online]. 10/19/2009 [cit. 2010-05-01]. Viktoriánský hlavolam. Dostupné z WWW: <http://www.deskovehry.info/pravidla/viktoria.htm>.

[49] Dots In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-05-01]. Dostupné z WWW: <http://commons.wikimedia.org/wiki/Dots>.

[50] LoveToKnow. *Home Improvement* [online]. 11 August 2007 [cit. 2010-05-01]. Image:Wood Plank Look Tile.JPG. Dostupné z WWW: <http://homeimprovement.lovetoknow.com/Image:Wood_Plank_Look_Tile.JPG>.

[51] KATO, N. *N. Kato's Homepage* [online]. 2010 [cit. 2010-05-01]. A Lights Out Puzzle with Solver (JavaScript). Dostupné z WWW: <http://www.ueda.info.waseda.ac.jp/~n-kato/lightsout/index.html>.

[52] Image Envision LLC. *ClipartOf.com* [online]. 2007-2010 [cit. 2010-05-01]. Clipart Picture of a Light Bulb Mascot Cartoon Character With Welcoming Open Arms. Dostupné z WWW: <http://www.clipartof.com/details/clipart/9686.html>.

[53] File:Dialog-information.svg In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 1 April 2006 [cit. 2010-04-25]. Dostupné z WWW: <http://commons.wikimedia.org/wiki/File:Dialog-information.svg>.

[54] File:Dialog-information on.svg In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 15 September 2008, 15 September 2008 [cit. 2010-05-01]. Dostupné z WWW: <http://commons.wikimedia.org/wiki/File:Dialog-information_on.svg>.

Příloha B

Další možné hry

Tabulka níže uvádí pro zájemce přehled her, které jsou i se stávající verzí pravidel implementovatelné. Při důkladnější rešerši by se jistě našly i další vhodné.

Český název hry	Přesně definovaný konečný stav	Web(y) [aktuální k 10. 5. 2010]	Možná implementace
Akari EX	ano	http://mellowmelon.wordpress.com/akari/akari-ex	I se stávajícími pravidly
Alenka v říši divů	ano	http://www.mathpuzzle.com/alicesolvers.htm http://www.milaadesign.com/Alice.html http://farfarfar.com/games/alice_mazes	I se stávajícími pravidly
Barevné bludiště	ano	http://www2.stetson.edu/~efriedma/color	I se stávajícími pravidly
Bludiště (bez možnosti jít doleva)	ano	http://www.logicmazes.com/easy/maze1.html http://www.clickmazes.com/noleft/ixnoleft.htm	I se stávajícími pravidly
Bludiště (průchody barvou)	ano	http://www2.stetson.edu/~efriedma/birds	I se stávajícími pravidly
Bludiště (přikázané směry)	ano	http://www2.stetson.edu/~efriedma/leftright	I se stávajícími pravidly
Bludiště (různě dlouhé tahy)	ano	http://www2.stetson.edu/~efriedma/unequal	I se stávajícími pravidly
Castle Wall	ano	http://mellowmelon.wordpress.com/castle-wall	I se stávajícími pravidly
Conwayova hra života	částečně	http://www.clickmazes.com/life/ixlife.htm	I se stávajícími pravidly
Červená-zelená-žlutá	ano	http://www.logicmazes.com/super.html	I se stávajícími pravidly
Číselné skoky	ano	http://www.logicmazes.com/n1mz.html	I se stávajícími pravidly
Double Back	ano	http://mellowmelon.wordpress.com/double-back	I se stávajícími pravidly
Dva Robinsoni	ano	http://www.deskovehry.info/pravidla/robinson.htm	I se stávajícími pravidly

Hidato	ano	http://en.wikipedia.org/wiki/Hidato http://www.hidato.com	I se stávajícími pravidly
Hitori	ano	http://en.wikipedia.org/wiki/Hitori http://www.nikoli.co.jp/en/puzzles/hitori	I se stávajícími pravidly
Hlavolamy na šachovnici	ne	http://www.deskovehry.info/pravidla/hlavolamy.htm	I se stávajícími pravidly
Kakuro	ano	http://cs.wikipedia.org/wiki/Kakuro http://www.nikoli.co.jp/en/puzzles/kakuro	I se stávajícími pravidly
Klikatice	ano	http://www2.stetson.edu/~efriedma/corner	I se stávajícími pravidly
Kurodoko	ano	http://www.nikoli.co.jp/en/puzzles/kurodoko	I se stávajícími pravidly
Kuromasu	ano	http://en.wikipedia.org/wiki/Kuromasu http://www.nikoli.co.jp/en/puzzles/where_is_black_cells	I se stávajícími pravidly
Latinský čtverec	ano	http://www2.stetson.edu/~efriedma/latin	I se stávajícími pravidly
Loydova 15 (obrázková)	ano	http://cs.wikipedia.org/wiki/Patn%C3%A1ctka	I se stávajícími pravidly
Loydova 15 (s bludištěm)	ano	http://cs.wikipedia.org/wiki/Patn%C3%A1ctka	I se stávajícími pravidly
Mochikoro	ano	http://en.wikipedia.org/wiki/Mochikoro http://puzzlinks.com/tag/mochikoro	I se stávajícími pravidly
Numberlink	ano	http://www.nikoli.co.jp/en/puzzles/numberlink	I se stávajícími pravidly
Nurikabe	ano	http://en.wikipedia.org/wiki/Nurikabe http://www.nikoli.co.jp/en/puzzles/nurikabe	I se stávajícími pravidly
Omezené skoky	ano	http://www.clickmazes.com/rules/ixrules.htm	I se stávajícími pravidly
Out of Sight	ano	http://mellowmelon.wordpress.com/out-of-sight	I se stávajícími pravidly
Peg Solitaire (anglická deska)	částečně	http://en.wikipedia.org/wiki/Peg_solitaire http://gwydir.demon.co.uk/jo/games/solitaire/index.htm http://www.deskovehry.info/pravidla/soliter.htm	I se stávajícími pravidly
Peg Solitaire (francouzská deska)	částečně	http://en.wikipedia.org/wiki/Peg_solitaire http://gwydir.demon.co.uk/jo/games/solitaire/index.htm http://www.deskovehry.info/pravidla/soliter.htm	I se stávajícími pravidly

Prospektoři	částečně	http://clanky.rvp.cz/clanek/r/ZF/1956/PROSPEKTORI.html	I se stávajícími pravidly
Přímé cesty	ano	http://www2.stetson.edu/~efriedma/straight	I se stávajícími pravidly
Right Face	ano	http://mellowmelon.wordpress.com/right-face	I se stávajícími pravidly
Rozsviť (Akari)	ano	http://en.wikipedia.org/wiki/Light_Up http://www.puzzle-light-up.com http://www.nikoli.co.jp/en/puzzles/akari	I se stávajícími pravidly
Různé vzdálenosti	ano	http://www2.stetson.edu/~efriedma/redpoint	I se stávajícími pravidly
Shinro	ano	http://en.wikipedia.org/wiki/Shinro http://shinropuzzles.web.officelive.com/aboutus.aspx	I se stávajícími pravidly
Skid	ano	http://www.milaadesign.com/Skid.html http://www2.stetson.edu/~efriedma/skid	I se stávajícími pravidly
Sokoban s barvami	částečně	http://www.milaadesign.com/Colorsok.html	I se stávajícími pravidly
Solitér dáma	ne	http://www.deskovehry.info/pravidla/dama-soliter.htm	I se stávajícími pravidly
Sorvo	ano	http://en.wikipedia.org/wiki/Survo_Puzzle http://www.survo.fi/puzzles	I se stávajícími pravidly
Sousedící žárovky	ano	http://www2.stetson.edu/~efriedma/bulb	I se stávajícími pravidly
Str8ts	ano	http://www.str8ts.com	I se stávajícími pravidly
Suraromu	ano	http://www.nikoli.co.jp/en/puzzles/suraromu	I se stávajícími pravidly
Šipky	ano	http://www2.stetson.edu/~efriedma/arrow	I se stávajícími pravidly
Špión	ano	http://www.milaadesign.com/Spy.html	I se stávajícími pravidly
Tamibari	ano	http://en.wikipedia.org/wiki/Tatamibari http://indi.s58.xrea.com/tatamibari	I se stávajícími pravidly
Tatebo-Yokobo	ano	http://en.wikipedia.org/wiki/Tatebo-Yokobo	I se stávajícími pravidly
Yajilin	ano	http://en.wikipedia.org/wiki/Yajilin http://www.nikoli.co.jp/en/puzzles/yajilin	I se stávajícími pravidly
Yajisan Kazusan	ano	http://en.wikipedia.org/wiki/Yajisan-Kazusan http://mellowmelon.wordpress.com/yajisan-kazusan	I se stávajícími pravidly

Z místa na místo	ano	http://www2.stetson.edu/~efriedma/rookend	I se stávajícími pravidly
Zabal to!	ano	http://www.milaadesign.com/Boxup.html http://www.clickmazes.com/boxup/ixboxup.htm	I se stávajícími pravidly
Zaurus	ne	http://www.milaadesign.com/Znumbers.html	I se stávajícími pravidly
Živá hradba	ano	http://www.clickmazes.com/ovd.act/ixamaze.htm	I se stávajícími pravidly
Cubilus	ano	http://www.milaadesign.com/Cubilus.html	Vyžaduje úpravu
Hledání min	ano	http://en.wikipedia.org/wiki/Minesweeper_%28video_game%29 http://www.milaadesign.com/Minesweeper.html	Vyžaduje úpravu
Housenky	ano	http://www.clickmazes.com/wrigc/ixwrigc.htm http://www.clickmazes.com/tjwrig/ixjwrig.htm http://www.clickmazes.com/wriggle/ixwriggle.htm	Vyžaduje úpravu
Odpich	ano	http://www.clickmazes.com/punt/ixpunt.htm	Vyžaduje úpravu
Q Tilt	ano	http://www.milaadesign.com/Q.html	Vyžaduje úpravu
Řetězová reakce	ano	http://www.milaadesign.com/Chainreaction.html	Vyžaduje úpravu
Stejné ke stejnému	ano	http://www.logicmazes.com/eyeball.html	Vyžaduje úpravu
Tilt	ano	http://www.milaadesign.com/Tiltmaze.html http://www.clickmazes.com/newtilt/s7b.htm	Vyžaduje úpravu
Trojka	ano	http://www.milaadesign.com/Threesome.html	Vyžaduje úpravu
Volný pád	ano	http://www.milaadesign.com/Freefall.html	Vyžaduje úpravu
Vyplňovačka	ano	http://www.milaadesign.com/Fullboard.html http://www2.stetson.edu/~efriedma/full	Vyžaduje úpravu
CELKEM	**64 dalších her**		

Tabulka 7: ***Přehled dalších implementovatelných her***

Příloha C

Uživatelská příručka aplikace

K aplikaci byla sepsána stručná příručka popisující základní možnosti ovládání a význam jednotlivých ovládacích prvků. Tento dokument je v aplikaci dostupný pod tlačítkem Nápověda. Vlastní soubor `documentation.rtf` se nachází podadresáři aplikace `HELP`. Jelikož jde o *Rich Text Format*, kde jsou všechny obrázky převedeny na bitmapy, je tento soubor poměrně velký – přes 70 MB. Tento formát však byl zvolen pro možnost programového načtení do ovládacího prvku *RichTextBox* a jeho snadné zobrazení.

Obsah

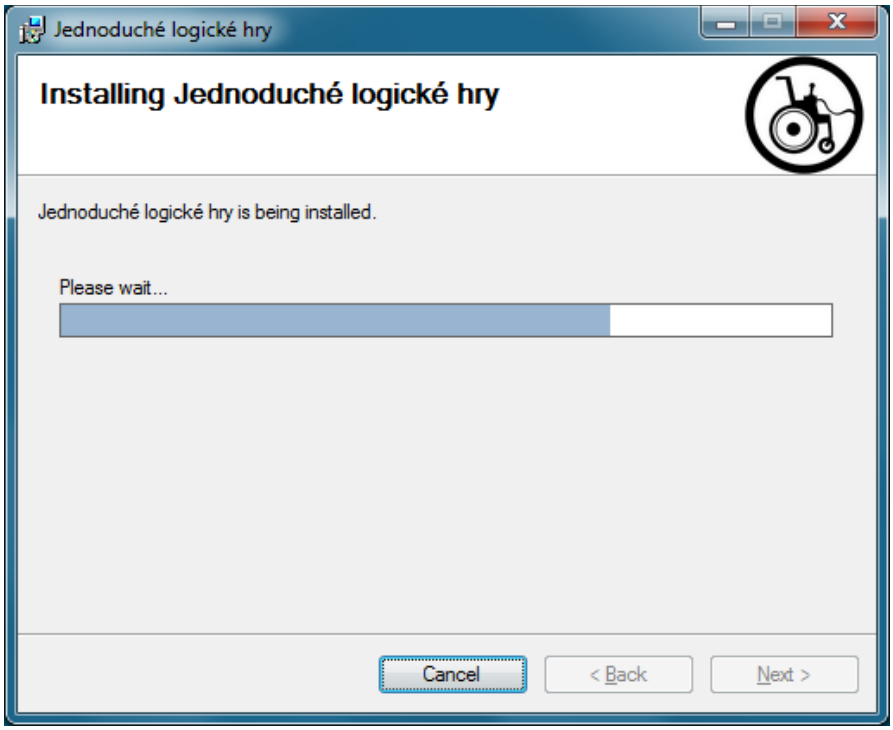
1. Úvod

Tato aplikace je součástí vypracování diplomové práce na téma *„Prostředí pro snadné vytváření jednoduchých (logických) her"*. Práce byla zadána a vedena Ing. Petrem Novákem ze skupiny *Nature Inspired Technologies Group* na *Katedře kybernetiky FEL ČVUT v Praze*. Program bude v budoucnu součástí komplexní softwarové výbavy platformy tzv. **Inteligentního invalidního vozíku**, což je aktuálně vyvíjený projekt, na němž se podílí i skupina *Inteligentní a mobilní robotiky*. Aplikaci však lze používat i samostatně bez závislosti na této vyvíjené platformě.

Aplikace kromě svého primárního účelu ukrátit volný čas také sbírá údaje o odehraných kolech. Tato data mohou později posloužit např. k vyhodnocení zlepšování hráče, k porovnání efektivity různých způsobů ovládání apod.

2. Instalace programu

Otevřete soubor `Jednoduche_logicke_hry_ver_1.0_setup.msi`. Spustí se standardní instalátor, který Vás provede všemi kroky instalace (viz Obrázek 17). V systémech Windows Vista a Windows 7 při zapnuté systémové službě *„Řízení uživatelských účtů (UAC)"* bude třeba vlastní instalací potvrdit administrátorem počítače.



Obrázek 17: *Průběh instalace*

2.1. Instalované součásti

Program po úspěšné instalaci vytvoří v počítači následující soubory a složky zabírající na disku zhruba 80 MB:

- na ploše:
  - zástupce spuštění programu
- v nabídce Start – Programy:
  - adresář `Inteligentní vozík\Jednoduché logické hry\` a v něm:

        - zástupce spuštění programu
        - zástupce odkazující na Nápovědu (tento text)
- v adresáři `C:\Program Files\`[4]:
    - adresář `Inteligentní vozík\Jednoduché logické hry\` obsahující:
        - adresář `HELP` obsahující:
            - adresář `#NEW_GAME#` obsahující:
                - šablonu k vytvoření nové hry neboli sadu všech potřebných prázdných souborů doplněných komentáři a Excelový nástroj k vytváření úrovní
            - adresář `PICTURES` obsahující:
                - sadu jednotných obrázků do případných nových her
            - soubor `documentation.rtf` (tento soubor)
        - soubor `BannerBitmap.bmp` (ikona instalace)
        - soubor `Simple Logic Games Framework.exe` (vlastní program)
- v adresáři `C:\Documents and Settings\<jméno uživatele>\Data aplikací\Inteligentní vozík\Jednoduché logické hry\`[5]:
    - adresář `GAMES\` obsahující:
        - *<název hry>* což jsou složky jednotlivých her obsahující všechny potřebné XML a grafické soubory ke všem úrovním

Během hraní se vytváří v dokumentovém adresáři uživatele další adresáře a soubory, kam se ukládají rozehrané hry. Struktura vypadá takto:

- adresář `C:\Documents and Settings\<jméno uživatele>\Dokumenty\Inteligentní vozík\Jednoduché logické hry\USERS`[6]
    - adresáře *<jméno uživatele>*, kde každý uživatel má svůj vlastní prostor a odehrané hry i všechna nastavení jsou ukládána zvlášť
        - soubor `properties.xml` s personáliemi uživatele
        - soubor `settings.xml` s nastaveními uživatele
        - adresář `SAVES` s uloženými hrami
            - adresáře *<název hry>* v nichž jsou uloženy soubory pro každou hru zvlášť
                - adresáře *<název úrovně>* v nichž jsou uloženy soubory zvlášť pro každou úroveň
                    - soubory *<00000000-99999999>*`.xml` do nichž se ukládá stav hry po každém platném tahu
                - soubor `last.xml` kde se uchovává posledně hraná úroveň
            - soubor `last.xml` kde se uchovává posledně hraná hra
    - soubor `last.xml` kde se uchovává posledně zvolený uživatel a nastavení výběrového okna

[4] V 64-bitových systémech Windows Vista a Windows 7 v adresáři `C:\Program Files (x86)\`

[5] V systémech Windows Vista a Windows 7 v adresáři `C:\Users\<jméno uživatele>\AppData\Roaming\Inteligentní vozík\Jednoduché logické hry\`

[6] V systémech Windows Vista a Windows 7 se soubory ukládají do složky `C:\Users\<jméno uživatele>\Documents\Inteligentní vozík\Jednoduché logické hry\USERS`

2.2. Odinstalování programu

Odinstalování provedete pomocí nástroje „Přidat nebo odebrat programy“ ve Windows XP a nižších (Start – Nastavení – Ovládací panely – Přidat nebo odebrat programy), nebo „Programy a funkce“ ve Windows Vista a Windows 7 (Start – Ovládací panely – Programy a funkce). V seznamu programů vyberte „Jednoduché logické hry“ a zvolte Odebrat či Odinstalovat.

Soubory a adresáře, které aplikace vytvoří během své činnosti, v počítači zůstanou. Můžete je ručně smazat z dokumentového adresáře.

3. Spuštění aplikace

Aplikace může být spuštěna ve dvou základních režimech, lišících se pouze způsobem výběru uživatele. Pokud je spuštěna z příkazového řádku s jedním parametrem (argumentem) následujícím za jménem *exe* souboru, vezme se tento jako jméno uživatele. Pokud není uživatel daného jména nalezen v adresáři schraňujícím jejich data, je založen nový uživatel zadaného jména s výchozími *(defaultními)* hodnotami nacionálů, avatara i uživatelských nastavení (viz dále). Možnost tohoto módu byla přidána pro případ spouštění her z řídící aplikace, kdy tato zavolá herní aplikaci s již známým jménem uživatele.

Při klasickém spuštění např. přes ikonu programu na ploše je nejprve zobrazen dialog výběru uživatele (viz dále).

4. Ovládání aplikace

Požadavkem bylo učinit ovládání co nejpřístupnější různým metodám při zachování jednoduchosti a intuitivnosti. Aplikaci tak lze rovnocenně ovládat myší, klávesnicí, jiným externím zařízením nebo i jejich kombinacemi.

Kromě nich může být pro handicapovaného hráče generován tzv. signál Next, neboli Další. Při příchodu tohoto signálu je dle kontextu vybrána další volba, např. další tah. Všechny hry jsou ovladatelné právě i jen tímto vstupem a jedním potvrzovacím (akčním) vstupem od uživatele.

Byla snaha ovládání i neherních součástí aplikace přizpůsobit tomuto vstupu. Většina hlavních částí tomu skutečně vyhovuje, některé je však i nadále možné ovládat jen pouze klávesnicí nebo myší. Aby bylo alespoň usnadněno používání myši, je aplikace vybavena vlastností *mouse-follow-focusu*, neboli česky *fokus sledující myš*. Přepínání aktivního okna se při něm děje pouhým najetím kurzoru na plochu okna.

Pozn.: Toto chování lze v novějších verzích Microsoft® Windows Vista a Windows 7 aktivovat i pro jakékoli jiné aplikace. Stačí zvolit příslušné nastavení v: Centrum usnadnění přístupu – Usnadnit používání myši – Usnadnění správy oken – Aktivovat okno ukázáním pomocí myši. U staršího systému Windows XP lze toto chování vynutit přes speciální nástroj Tweak UI ze sady Microsoft PowerToys nebo zásahem do systémového registru (jen pro zkušené uživatele!). V operačním systému Mac OS X by měla být k dispozici obdobná možnost (neověřeno).

Právě vybrané okno nebo ovládací prvek je navíc pro lepší přehled zvýrazněno světle zeleným pozadím. Ostatní zobrazené prvky jsou zašedlé.

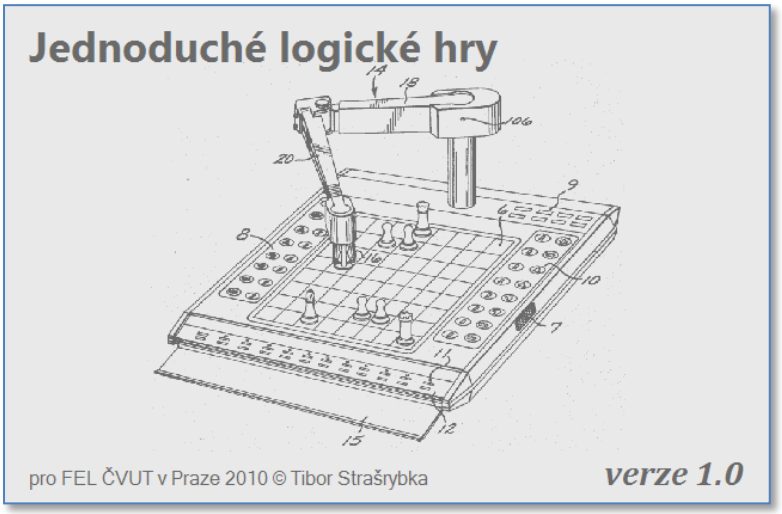
Požadavku jednoduchosti ovládání je dostáno minimalizací počtu nutných kroků ze strany uživatele. Výběry hráče, hry, úrovně i uložení jsou automatizovány. Všechna

tlačítka mají také nastavenou zrychlenou volbu přes klávesovou zkratku Alt + *písmeno*. Toto písmeno je u každého tlačítka jiné a je zvýrazněno podtržením v popisku tlačítka po stisku klávesy Alt. Dalším podobným usnadněním je definování takzvaných horkých kláves (hotkeys), jejichž přehled uvádí kapitola 6.

5. Okna

5.1. Úvodní

Po spuštění aplikace je zobrazena na 3 sekundy uvítací obrazovka *(splash screen)* (Obrázek 18) aplikace, která poté sama zmizí nebo i dříve, pokud je na ni kliknuto myší.



Obrázek 18: *Uvítací obrazovka*

5.2. Uživatel

5.2.1. Výběr

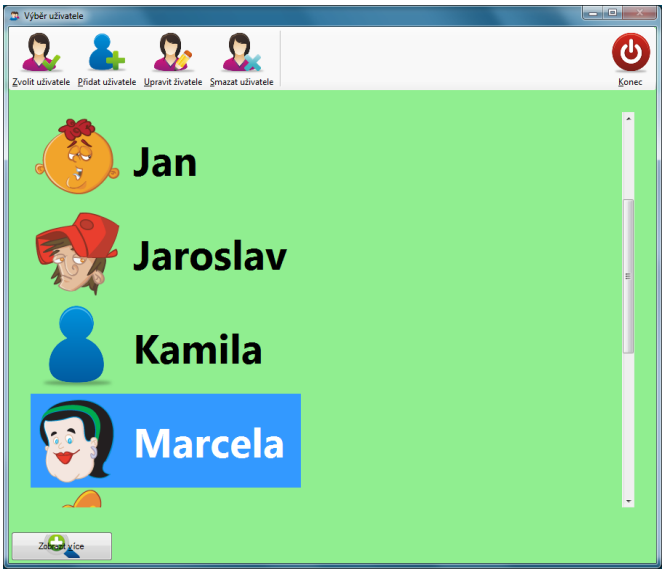
Po spuštění aplikace se objeví přihlašovací dialogové okno s výběrem uživatele. (Obrázek 19, rodná čísla jsou smyšlená). Uživatelé jsou zobrazeni v tabulce pod sebou, v které jsou zobrazeny všechny základní informace o uživatelích. Toto zobrazení lze z plného přepnout do úsporného, kdy jsou zobrazena pouze jména a profilové obrázky (Obrázek 20). Přepínání těchto dvou možných zobrazení se děje výběrem tlačítka Zobrazit méně / Zobrazit více.

Pro usnadnění přihlášení je při každém zobrazení tohoto okna spuštěn odpočet automatického výběru uživatele. Pokud se během 15 sekund nestane žádná událost (stisk klávesy nebo tlačítka myši), je po uplynutí této doby automaticky vybrán uživatel, který byl přihlášen naposledy. Při události se odpočet automatického výběru přeruší a uživatele je třeba vybrat ručně.

Při stisknutí šipky dolů nebo příchodu Nextu se zvýrazní další řádek pod právě zvýrazněným. Když výběr (fokus) řádku dojde až na konec seznamu, přeskočí při dalším Nextu nebo šipky dolů na první řádek. Šipka nahoru zase pro opačný přechod v tabulce.

Kromě automatické volby může být manuální výběr uživatele potvrzen třemi způsoby: buď dvojitým poklepáním myši na řádku s údaji, stiskem klávesy Enter nebo stiskem tlačítka Zvolit uživatele v liště okna.

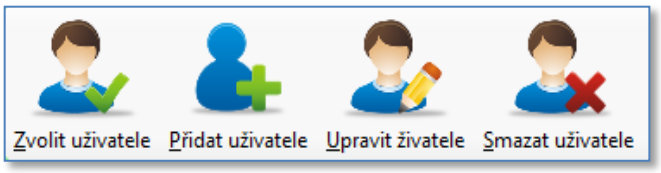
Obrázek 19: *Výběr uživatele (plné zobrazení)*



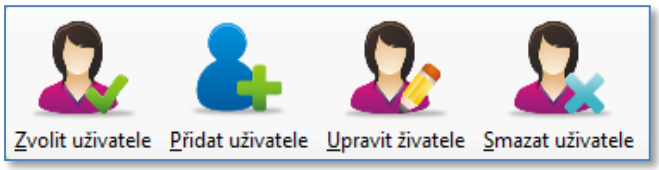
Obrázek 20: *Výběr uživatele (úsporné zobrazení)*

V liště obou oken je sada funkčních ikon. První je již výše zmíněný výběr uživatele, poslední je Konec, po jejíž volbě dojde k okamžitému ukončení aplikace. Dále jsou zde tlačítka Přidat uživatele, Upravit uživatele a Smazat uživatele, která popisují samostatné podkapitoly dále.

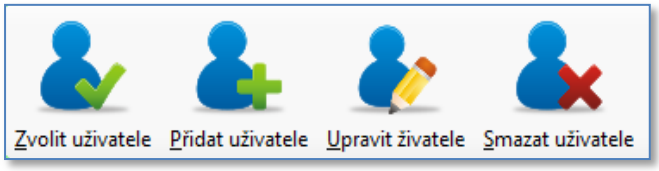
Tyto ikony jsou navíc různé pohlaví uživatele (Obrázek 21, Obrázek 22, Obrázek 23).



Obrázek 21: *Ikony uživatele mužského pohlaví*



Obrázek 22: *Ikony uživatele ženského pohlaví*



Obrázek 23: *Ikony uživatele neurčeného pohlaví*

5.2.2. Přidání uživatele

V tomto dialogovém okně (Obrázek 24) je potřeba vyplnit všechny tři údaje: jméno, příjmení a rodné číslo ve správném formátu. Z něho se automaticky určí pohlaví a věk uživatele. Pokud je přidáván první uživatel, je u položky jméno nabídnuto *systémové jméno* přihlášeného *uživatele* počítače. V pravé části si uživatel zvolí ze sady obrázků, který bude použit jako jeho profilový, tzv. Avatar. Ten pak v některých hrách může být zobrazen jako aktuální pozice.

Ve spodní části je speciální táhlo, kterým lze volit přibližný stupeň postižení. Toto nastavení má zásadní význam, neboť se jím hromadně přizpůsobí všechna nastavení ovládání a zobrazovaní do optimální kombinace vzhledem k danému stupni postižení. Všechna nastavení lze sice měnit i později, ovšem již skrze dialogové okno nebo klávesové zkratky. Změna nastavení pouze pomocí Nextů není možná.

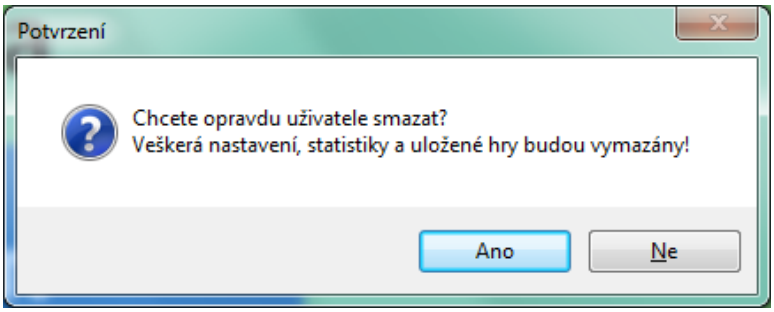
Obrázek 24: *Založení a úprava uživatele*

5.2.3. Úprava uživatele

Stiskem tohoto tlačítka se otevře stejný dialog jako při přidání, jen jsou do něj již načtena stávající hodnoty. Při potvrzení změn platí stejná omezení, jako u přidání nového uživatele. Editovat lze i uživatele, kteří byli založeni předáním řetězce jako argumentu z příkazové řádky a předefinovat tak u nich defaultní hodnoty.

5.2.4. Smazat uživatele

Smaže uživatelův profil včetně všech uložených souborů her. Pro případ, že by došlo k jejímu nechtěnému stisku, je tuto volbu nutno stvrdit v dialogu (Obrázek 25). Smazání lze vyvolat i stiskem klávesy Del.



Obrázek 25: *Potvrzovací dialog smazání uživatele*

5.3. Výběr hry

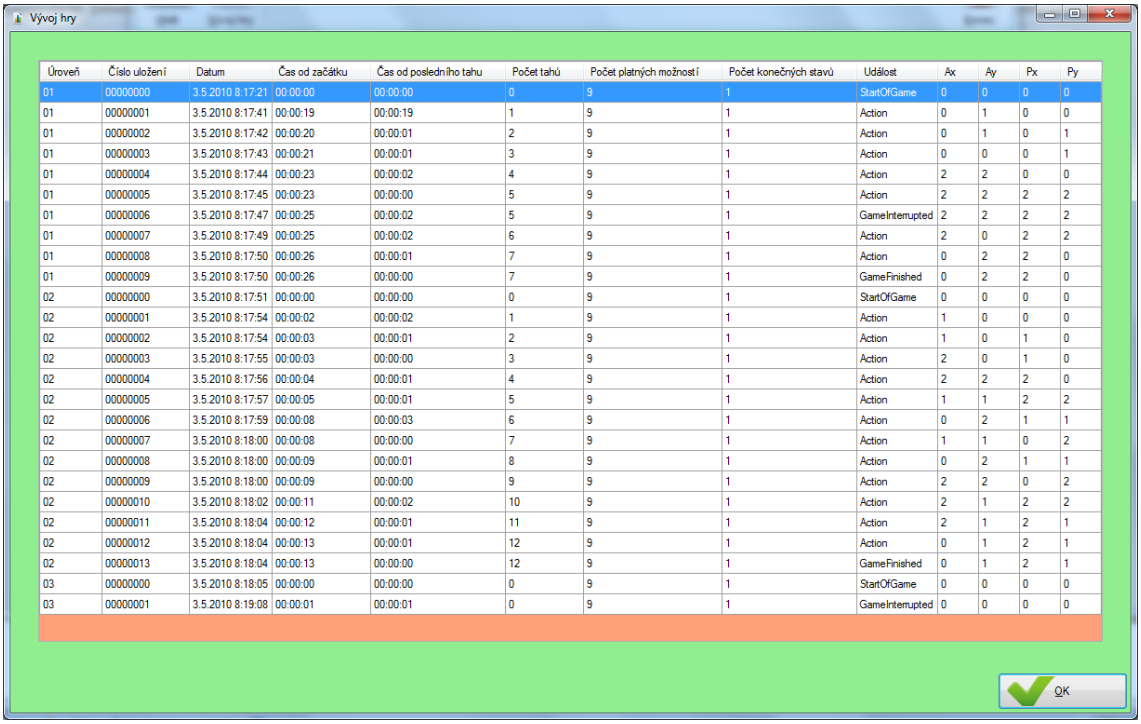
Po přihlášení uživatele si tento vybere hru z dialogu, u kterého platí stejné možnosti jako u výběru uživatele. Pouze úsporné zobrazení zde chybí.

Obrázek 26: *Výběr hry*

Výběr hry je opět možné stvrdit třemi způsoby + automaticky. V liště je také výběrové tlačítko (Hrát). O druhém z nich (Vývoj hry) je další podkapitola.

5.3.1. Vývoj hry

Aplikace po každém kole uloží stav celé hry do příslušného uživatelova podadresáře. Z těchto souborů jsou generovány jisté informace, které spolu s dalšími lze zobrazit do přehledové tabulky (Obrázek 27). Pro vlastní hraní není tato tabulka důležitá, slouží jako podklad k analýze odehraných her. Při každém restartu hry jsou všechny soubory kromě prvního smazány, takže i tato tabulka se vyprázdní.



Úroveň	Číslo uložení	Datum	Čas od začátku	Čas od posledního tahu	Počet tahů	Počet platných možností	Počet konečných stavů	Událost	Ax	Ay	Px	Py
01	00000000	3.5.2010 8:17:21	00:00:00	00:00:00	0	9	1	StartOfGame	0	0	0	0
01	00000001	3.5.2010 8:17:41	00:00:19	00:00:19	1	9	1	Action	0	1	0	0
01	00000002	3.5.2010 8:17:42	00:00:20	00:00:01	2	9	1	Action	0	1	0	1
01	00000003	3.5.2010 8:17:43	00:00:21	00:00:01	3	9	1	Action	0	0	0	1
01	00000004	3.5.2010 8:17:44	00:00:23	00:00:02	4	9	1	Action	2	2	0	0
01	00000005	3.5.2010 8:17:45	00:00:23	00:00:00	5	9	1	Action	2	2	2	2
01	00000006	3.5.2010 8:17:47	00:00:25	00:00:02	5	9	1	GameInterrupted	2	2	2	2
01	00000007	3.5.2010 8:17:49	00:00:25	00:00:02	6	9	1	Action	2	0	2	2
01	00000008	3.5.2010 8:17:50	00:00:26	00:00:01	7	9	1	Action	0	2	2	0
01	00000009	3.5.2010 8:17:50	00:00:26	00:00:00	7	9	1	GameFinished	0	2	2	0
02	00000000	3.5.2010 8:17:51	00:00:00	00:00:00	0	9	1	StartOfGame	0	0	0	0
02	00000001	3.5.2010 8:17:54	00:00:02	00:00:02	1	9	1	Action	1	0	0	0
02	00000002	3.5.2010 8:17:54	00:00:03	00:00:01	2	9	1	Action	1	0	1	0
02	00000003	3.5.2010 8:17:55	00:00:03	00:00:00	3	9	1	Action	2	0	1	0
02	00000004	3.5.2010 8:17:56	00:00:04	00:00:01	4	9	1	Action	2	2	2	0
02	00000005	3.5.2010 8:17:57	00:00:05	00:00:01	5	9	1	Action	1	1	2	2
02	00000006	3.5.2010 8:17:59	00:00:08	00:00:03	6	9	1	Action	0	2	1	1
02	00000007	3.5.2010 8:18:00	00:00:08	00:00:00	7	9	1	Action	1	1	0	2
02	00000008	3.5.2010 8:18:00	00:00:09	00:00:01	8	9	1	Action	0	2	1	1
02	00000009	3.5.2010 8:18:00	00:00:09	00:00:00	9	9	1	Action	2	2	0	2
02	00000010	3.5.2010 8:18:02	00:00:11	00:00:02	10	9	1	Action	2	1	2	2
02	00000011	3.5.2010 8:18:04	00:00:12	00:00:01	11	9	1	Action	2	1	2	1
02	00000012	3.5.2010 8:18:04	00:00:13	00:00:01	12	9	1	Action	0	1	2	1
02	00000013	3.5.2010 8:18:04	00:00:13	00:00:00	12	9	1	GameFinished	0	1	2	1
03	00000000	3.5.2010 8:18:05	00:00:00	00:00:00	0	9	1	StartOfGame	0	0	0	0
03	00000001	3.5.2010 8:19:08	00:00:01	00:00:01	0	9	1	GameInterrupted	0	0	0	0

Obrázek 27: *Okno s vývojem hry*

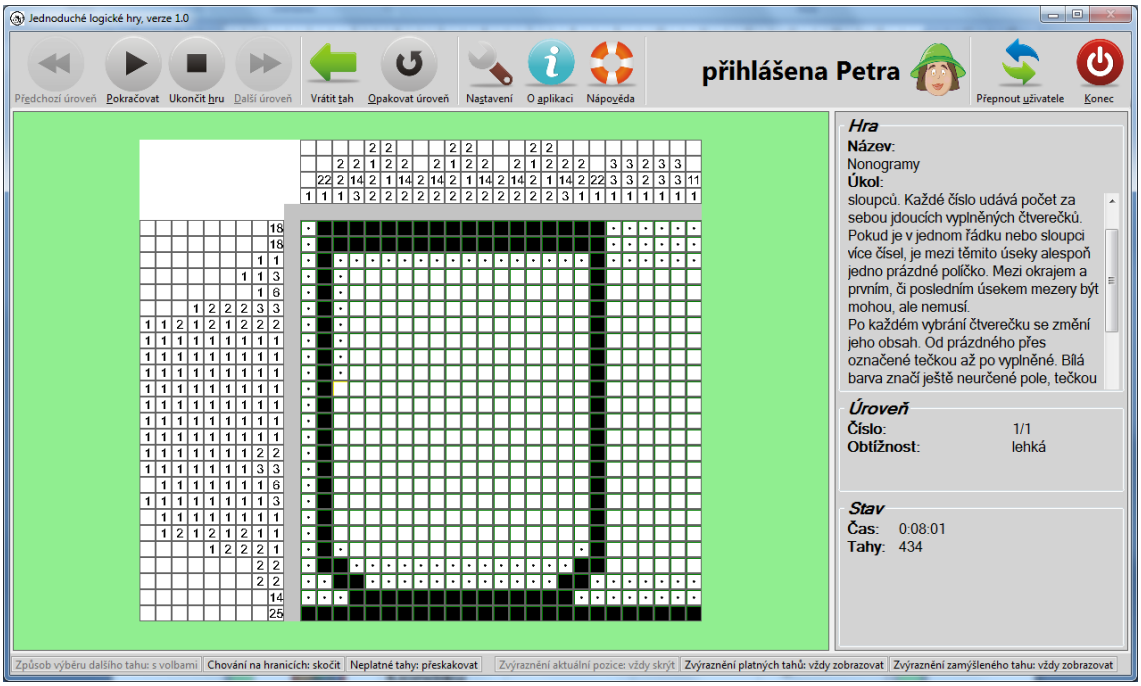
5.4. Hlavní

Po výběru hráče a hry se zobrazí hlavní okno (Obrázek 28) aplikace s posledně známou hranou úrovní s posledně známým stavem hry. Okno je rozděleno do tří hlavních částí:

1) Vlastní plocha (panel) hry v levých ¾ okna
2) Informační panel v pravé čtvrtině okna s řádkovým popisem přes dolní část okna
3) Řádek menu s ovládacími prvky (tlačítky) přes horní část okna

Dvě poslední lze nechat volitelně skrýt (viz kapitola Klávesové zkratky), aby byla herní plocha a na ní zobrazované prvky větší a přehlednější. Velikost všech kostiček se totiž přizpůsobuje aktuální velikosti panelu a snaží se vždy být maximální. Okno lze roztahovat a přizpůsobovat tak velikost hrací plochy. Přes klávesovou zkratku nebo volbu v nastaveních lze okno nechat zobrazit i přes celou obrazovku *(fullscreen)*.

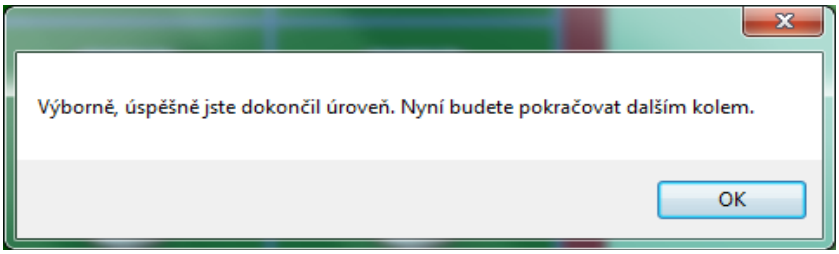
Jednou nastavený fullscreen se pro příští spuštění zapamatuje, informační panel a menu se zobrazí při nové hře vždy.



Obrázek 28: *Hlavní okno*

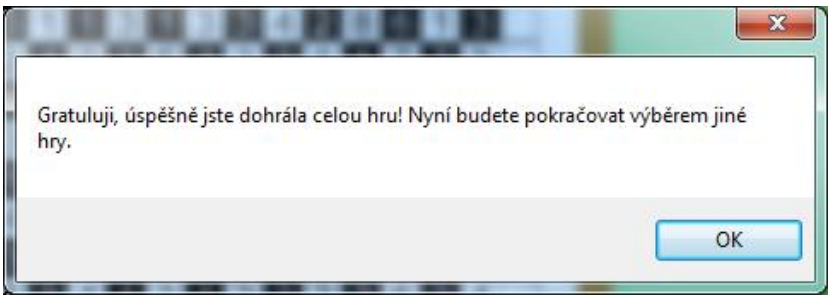
1) Na ploše se odehrává vlastní hra. Popis ovládání jednotlivých her uváděn nebude, měl by být specifikován ve slovním popisu pravidel (Úkolu) hry. Obecné herní možnosti jsou u všech her stejné a měly by být intuitivní. Jejich detailní popis obsahuje vlastní text diplomové práce.
2) V informačním panelu se zobrazují informace o hře (pravidla a název), o úrovni (číslo, název, obtížnost a na kolik tahů je hra minimálně řešitelná) a informace aktuálním stavu hry (čas od začátku, počet odehraných tahů a případně další u každé hry různé).
   Na proužku dole jsou pak zobrazeny volby základních nastavení ovládání hry.
3) Řádek menu obsahuje celkem 11 tlačítek, jejichž význam je zřejmý z jejich popisků. Kromě nich se zde zobrazuje avatar a jméno přihlášeného uživatele.
   Některé položky menu mohou být během hry neaktivní, pokud např. ještě není ze hry odehrán ani jeden tah, jsou tlačítka Vrátit tah a Opakovat úroveň do tohoto prvního tahu neaktivní.

O úspěšném dohrání úrovně je hráč informován (Obrázek 29) a pokud existuje další úroveň hry, je na ni automaticky přeskočeno, aby se ušetřily nutné kroky.



Obrázek 29: *Zjednodušený výběr další úrovně*

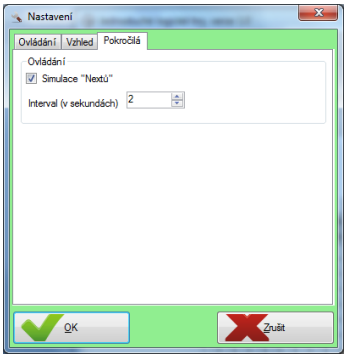
Podobně při dohrání všech úrovní celé hry, kdy je tedy dohraná celá, je hráč o této skutečnosti informován (Obrázek 30) a je mu automaticky nabídnut znovu výběr her.



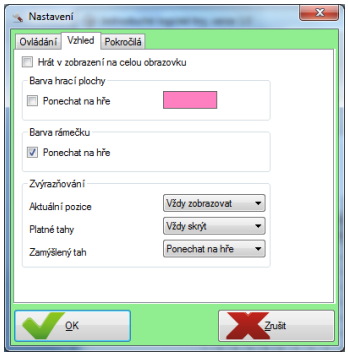
Obrázek 30: *Zjednodušený výběr další hry*

5.5. Nastavení

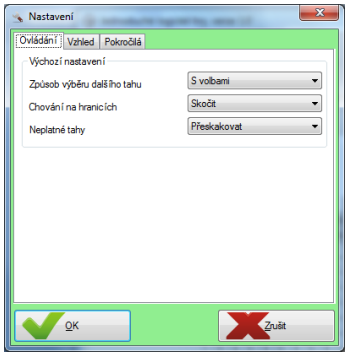
Většinu nastavení lze ovlivnit a měnit speciálními klávesovými zkratkami a povely, ovšem zároveň také rovnocenně přes toto dialogové okno. V něm jsou tři záložky podle skupin nastavení. Na první (Obrázek 31) se přizpůsobuje ovládání aplikace, na druhém (Obrázek 32) vzhled aplikace a konečně na třetím (Obrázek 33) je pouze jediná volba, kterou je vypnutí/zapnutí programového generování Nextu a nastavení intervalu tohoto generování v sekundách.



Obrázek 33: *Nastavení 3*



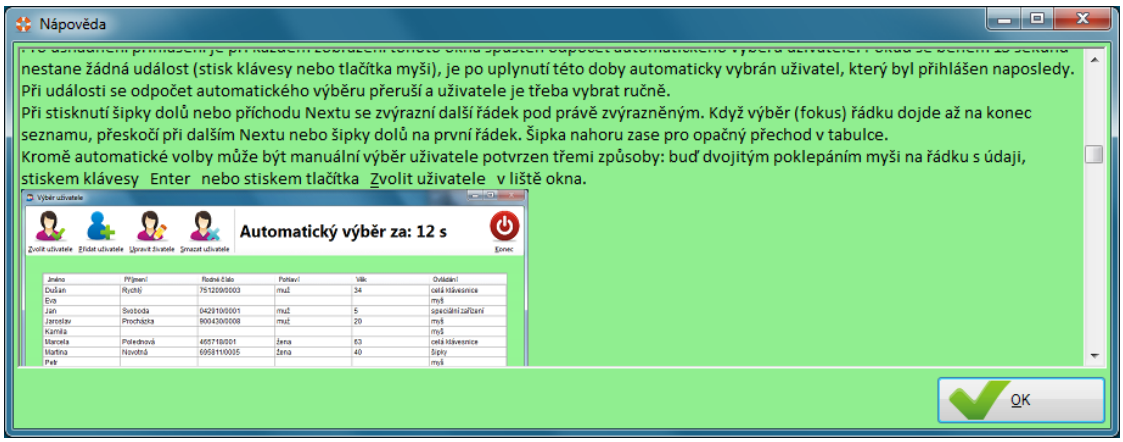
Obrázek 32: *Nastavení 2*



Obrázek 31: *Nastavení 1*

5.6. Nápověda

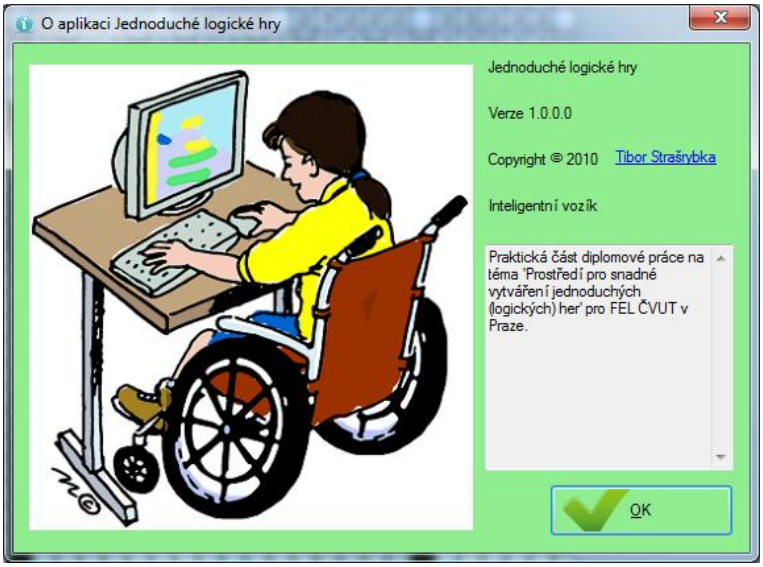
V prostoru toho okna (Obrázek 34: *Nápověda*) je zobrazován text, který právě čtete. Aby byla zajištěna jeho čitelnost i bez nutnosti navigace myší, stejně jako u dlouhých popisů pravidel her se text samovolně odvíjí nahoru a dolů.



Obrázek 34: *Nápověda*

5.7. O aplikaci

V tomto okně (Obrázek 35) jsou zobrazeny základní informace o aplikaci včetně e-mailového kontaktu na autora.



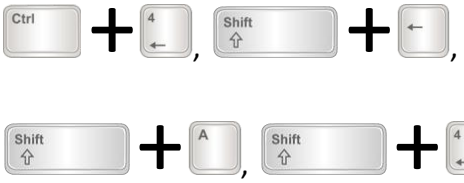
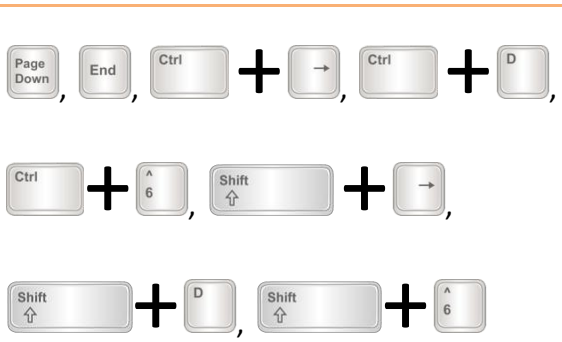
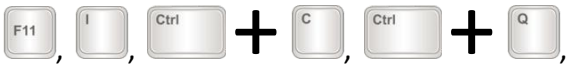
Obrázek 35: *O aplikaci*

6. Horké klávesy a klávesové zkratky

Ovládání her i vlastní aplikace pouze myší by mohlo být zdlouhavé, nebo dokonce u některých uživatelů přímo nemožné. Všechny změny v aplikaci jsou proto řízeny sadou interních příkazů, na které se povely myši, klávesnice i jiných zařízení převádějí.

Při ovládání klávesnicí je většina těchto příkazů přiřazena zároveň více různým klávesám. Je to z důvodu individuality každého uživatele, kdy si každý může vybrat, zda mu např. vyhovují spíše klasické směrové šipky nebo čtveřice kláves W A S D. Akce uvedené v hranatých závorkách s opakovaným stiskem rotují.

Klávesy nebo jejich kombinace	**Akce *(interní příkaz)***
↑, W, 8 ↑	pohyb nahoru
→, D, ^ 6	pohyb doprava
↓, S, 2 ↓, X, Y, Z	pohyb dolů
←, A, 4 ←	pohyb doleva
Enter	potvrzení volby, akční klávesa
N	Next (další volba)
Backspace	vrátit poslední tah
, R	restart úrovně
P, Pause Break	[pauza | opětovné spuštění]
Page Up, Home, Ctrl + ←, Ctrl + A,	skok na předchozí úroveň

	
	skok na další úroveň
	[zobrazit | nezobrazit] na celou obrazovku
	[skrytí | zobrazení] navigačního menu
	zobrazení okna Nápovědy
	zobrazení okna Nastavení
	[skrýt | zobrazit] informační panel
	změna módu Výběru dalšího tahu: [ponechat na hře | s možnostmi | přímý (pouze klávesnicí) | přímý (klávesnicí i myší)]
	změna módu Chování na hranicích: [ponechat na hře | zastavit | rotovat | skočit]
	změna módu Vybírání neplatných tahů: [ponechat na hře | vybírat | přeskakovat]
	změna Zvýraznění aktuální polohy: [ponechat na hře | vždy zobrazovat | vždy skrýt]
	změna Zvýraznění platných tahů: [ponechat na hře | vždy zobrazovat | vždy skrýt]
	změna Zvýraznění zamýšleného tahu *(fokusu)*: [ponechat na hře | vždy zobrazovat | vždy skrýt]
	okamžité ukončení aplikace
	přerušení stávající hry a výběr jiné

Tabulka 8: *Klávesy a klávesové zkratky v aplikaci*

7. Zdroje

Zde je seznam online zdrojů, ze kterých byly čerpány použité obrázky a zvuky v aplikaci mimo samotných her.

IKONY A OBRÁZKY

[55] Stichting Bartiméus Accessibility. *Game Accessibility* [online]. 2006 [cit. 2010-05-01]. Introduction. Dostupné z WWW: <http://www.game-accessibility.com/index.php?pagefile=introduction>.

[56] VEIGA, Dirceu. *Fasticon.com* [online]. 2003 - 2010 [cit. 2010-05-01]. Avatar Girls Icons. Dostupné z WWW: <http://www.fasticon.com/stockicons/index.php/avatar-girls>.

[57] VEIGA, Dirceu. *Fasticon.com* [online]. 2003 - 2010 [cit. 2010-05-01]. Avatar Boys Icons. Dostupné z WWW: <http://www.fasticon.com/stockicons/index.php/avatar-boy>.

[58] KeyboardIcons. *KeyboardIcons* [online]. 2008 [cit. 2010-05-01]. 104 Standard PC Keyboard Key Icons Download. Dostupné z WWW: <http://keyboardicons.com/free-keyboard-icons/104-standard-pc-keyboard-key-icons-download.html>.

[59] devianART. *IconArchive* [online]. 2010 [cit. 2010-05-01]. Tulliana 2 Icons. Dostupné z WWW: <http://www.iconarchive.com/category/system/tulliana-2-icons-by-umut-pulat.2.html>.

[60] WEEKS, Mark. *Chess for All Ages* [online]. 2009 [cit. 2010-05-05]. Robot Computer Chess Game. Dostupné z WWW: <http://www.mark-weeks.com/aboutcom/imgallry/aa07l22>.

[61] HICKS, Mark A. *Discovery Education* [online]. 2010 [cit. 2010-05-05]. Computer girl. Dostupné z WWW: <http://school.discoveryeducation.com/clipart/clip/wheelchair-access-color.html>.

[62] ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. *ČVUT* [online]. 2005 [cit. 2010-04-16]. Logo ČVUT. Dostupné z WWW: <http://www.cvut.cz/informace-pro-media/logo-cvut>.

[63] DryIcons.com. *DryIcons.com* [online]. 2007-2009 [cit. 2010-04-25]. Coquette Icons. Dostupné z WWW: <http://dryicons.com/free-icons/preview/coquette-icons-set>.

ZVUKY

[64] Amazing Sounds. *Amazing Sounds!* [online]. 2001 [cit. 2010-05-01]. Human Sounds. Dostupné z WWW: <http://amazingsounds.iespana.es/awwww.wav>.

[65] A1FSE. *Free Radio Sounds* [online]. 1998-2010 [cit. 2010-05-01]. Other Free Sound Downloads. Dostupné z WWW: <http://www.a1freesoundeffects.com/freesounds5/fanfare.wav>.

[66] WavWeb. *The Website For Sounds* [online]. 2010 [cit. 2010-05-01]. WAV SOUND FILES (Complete Archive). Dostupné z WWW: <http://www.members.tripod.com/~buggerluggs/wavs/fanfare.wav>.

Příloha D

Obsah přiloženého CD

Adresář	Popis obsahu
DOC	Text diplomové práce ve formátech *MS Word 2007*, *MS Word 2003* a *PDF*
GAMES	Kolekce realizovaných her spolu se šablonou na jejich vytváření
INSTALL	Instalační soubor aplikace *Jednoduché logické hry, verze 1.0*
OUTPUTDATA	Výstupní data z několika ukázkových her při různém způsobu jejich ovládání
SCANN	Naskenované originální zadání a prohlášení o vypracování.
SUPPORT	Podpůrné systémové instalační soubory *(.NET Framework a Windows Installer)*

Po vložení CD do mechaniky se automaticky spustí instalátor herní aplikace.