VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

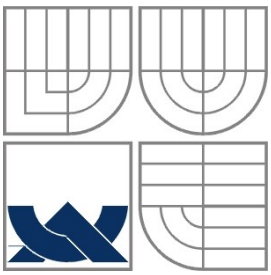
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GOOGLE ANDRIOD: EDITOR A PŘEHRÁVAČ HER

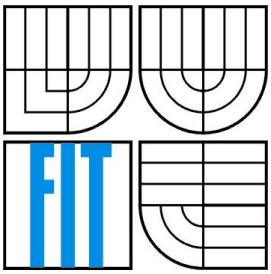
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ADAM SLOVÁČEK
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GOOGLE ANDRIOD: EDITOR A PŘEHRÁVAČ HER

GOOGLE ANDROID: GAME EDITOR AND GAME PLAYER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ADAM SLOVÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. IGOR SZÖKE

BRNO 2009

Abstrakt

Bakalářská práce pojednává o návrhu a implementaci univerzálního systému pro tvorbu aplikací, které lze zjednodušit na množinu elementárních aktivit a vztahů mezi nimi. Konkrétně je pak implementován editor her, ve kterém je možné takové aplikace skládat a přehrávače na platformě Google Android, který je interpretuje. Nabízí také výhled na uplatnění takového systému v praxi.

Abstract

The goal of the bachelor thesis is to design and implement a universal framework for developing applications that can be simplified to a structure of atomic activities and relations between them. An editor for developing applications based on the framework is provided as well as a Google Android compatible application interpreter. A study of possible usages of the framework is included.

Klíčová slova

herní systém, tvorba her, logické sítě, android, editor her, přehrávač her, herní platforma

Keywords

game system, game developing, logical nets, android, game editor, game player, game platform

Citace

Adam Slováček: Google Android: Editor a přehrávač her, bakalářská práce, Brno, FIT VUT v Brně, 2009

Google Android: Editor a přehrávač her

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Adam Slováček
17.5.2009

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Igoru Szökemu, za jeho ochotu čas a pomoc, při psaní mé bakalářské práce.

Obsah

1 Úvod

Programování se s postupem času stává přístupné stále širšímu okruhu lidí. Jak se vyvíjí programovací jazyky a vývojová prostředí, přinášejí stále větší komfort ve psaní rozsáhlých a komplexních aplikací.

Už objektové programování s sebou nese možnost využívat již hotové kusy kódu či knihovny velmi komfortním způsobem. Pokud by měl programátor skládal aplikaci již jen z hotových knihoven a vlastní vyvíjet nepotřeboval, to za něj dělal jiný programátor, komfort vývoje aplikací by dále vzrostl.

Nelze mít ale tolik, a tak obecných knihoven na to, aby stačilo pro napsání libovolně složité aplikace skrze jejich využití. Lze však utvořit takové, které svou funkcionalitou a přiměřenou obecností pokryjí nějaký obor.

Systém je postaven na myšlence jednoduchého tvoření aplikací poskládaných z nějaké skupiny implementovaných prvků (nazvěme je **místa**).

Tyto místa symbolizují akce, ke kterým může v aplikaci dojít. Pokud by systém poskytoval množinu takových míst pro klasické adventurní hry, obsahovala by například prvek pro dialog, animaci, inventář, nalezení předmětu, příchozí hovor apod. Je nemožné najít univerzální skupinu míst tak, aby pokryla veškeré možné typy adventurních her a jejich nuance. Vytvořit lze takovou, která poskytne většinu a poskytnout možnost dovolí tvořit nové typy míst dle potřeb dané aplikace.

Možnosti využití definuje právě množina typů implementovaných míst. Práce nabízí takové, které demonstrují jeho funkčnost na jednoduchých hrách.

Platforma Android byla vybrána záměrně právě proto, že je relativně nová a poskytuje široké pole pro uplatnění softwaru. Herní systém může využívat některé zařízení, které na běžném stolním počítači nalézt nelze. Ať už jde o telefon, GPS modul, teploměr apod.

Původní koncept počítal právě se zmíněným GPS modulem, jako nástrojem pro tvoření adventurních her v prostředí reálného světa. Až zobecněním vznikl nový, který nabízí přístup, jak bylo zmíněno výše.

Práce se zaobírá všemi stupni vývoje herního systému (složeného z editoru a přehrávače), které bylo nutno zvládnout. V kapitole „Utváření konceptu“ zmiňuje existující řešení, ze kterých se při návrhu konceptu systému vycházelo. Ten je rozebrán v kapitole „Herní systém“, která obsahuje i jeho formální popis. Následující kapitoly se věnují již implementaci jednotlivých částí systému. Pokračuje kapitolou demonstrující princip stavění sítí dle scénáře (kapitola „Ukázková hra“). Protože vývoj systému nekončí s touto prací, popisuje v kapitole „Další vývoj“ vylepšení a rozšíření, která jsou plánována do budoucna, stejně jako jiná eventuální využití konceptu systému.

2 Utváření konceptu

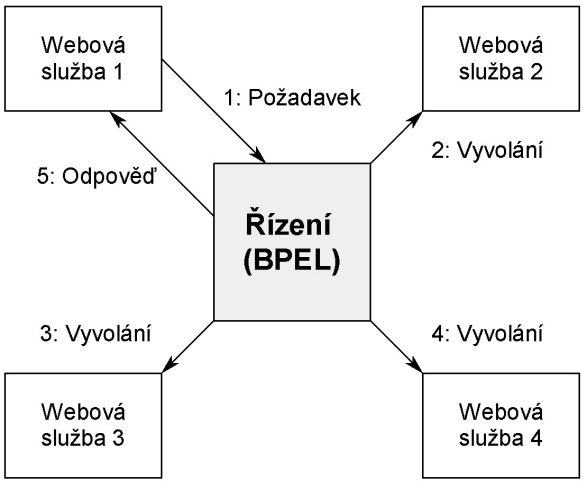
Z myšlenky, kde figurovala jen hra a její editor, bylo třeba vystavět koncept systému, který by co nejšířeji pokryl možné očekávané chování. Skloubit univerzálnost a nabídnout systém tak, aby v něm mohl pracovat „běžný uživatel“.

Existuje celá řada jazyků (a editačních programů včetně interpretů) pro popis chování složitých činností [1] (tzv. „Workflow languages“, dále jen WL), která by se dala využít pro popis chování systému. Bylo třeba brát zřetel na platformu přehrávače (žádný z existujících nenabízí oddělený interpret pro platformu Android nemluvě o optimalizaci na hardware). Kapitola pokračuje popisem jazyků, ze kterých byl vyextrahován koncept pro implementovaný systém a výčtem těch, které nabízejí obdobné možnosti deklarace toku úloh.

2.1 Vybrané existující WL jazyky

WS-BPEL (Web Services Business Process Execution Language) – je jazyk na bázi XML, který slouží pro zápis business procesů na vykonatelné úrovni. Ty mohou být buď přímo spustitelné, nebo abstrahovat nějakou komplexnější funkcionalitu [2]. Proces implementovaný pomocí BPEL je nasazen do běhového prostředí, které je schopno jej interpretovat. Tímto prostředím může být samostatný procesní server, modul v aplikačním serveru apod. Hlavní funkcí BPELu je řízení webových služeb, tedy řízení spolupráce funkcionalit, kterou systém nabízí (obrázek 2.1.1).

Tato funkcionalita je složena z operací, které je možné volat přes webovou službu. Na druhé straně BPEL sám stojí za webovou službou, která definuje jeho rozhraní. Pro každý vstup do procesu je ve webové službě, která popisuje rozhraní BPELu, jedna operace.



Obrázek 2.1.1: Funkce BPEL

Dá se tedy říci, že BPEL implementuje webovou službu. Přitom aplikace, která webovou službu používá, neví, jak vnitřně funguje. BPEL je platformě nezávislý [3].

Jazyk je pro systém nevhodný. Hlavně kvůli úzké specializaci a závislosti na webových službách.

Scufl (Simple Conceptual Unified Flow Language) – Využit v projektu „myGrid“ v aplikaci „Taverna Workbench“[1]. Na rozdíl od WS-BPEL dovoluje, aby prováděné služby (procesy) a data byla umístěna buď na lokálním počítači nebo kdekoli v internetu (Zajišťuje také jejich distribuci a zpřístupňuje v editoru). Své uplatnění našel ve vědecké praxi. Služby lze tedy již stáhnout nebo tvořit vlastní – ty lze pak nabízet dalším uživatelům. Přitom i první zmíněné lze pomocí skriptovacího jazyka upravovat [4]. Editor slouží zároveň k validaci a spuštění vymodelovaných procesů.

YAWL (Yet Another Workflow Language) – další z WL jazyků, vycházející z Petriho sítí. K modelování slouží přiložený grafický editor. Celý popis se skládá z jednoho počátečního a jednoho koncového místa. Dále z množiny úkolů (task) a podmínek. Úkoly mohou být atomické nebo složené. Spojnicemi se definují vztahy mezi nimi – jak po sobě úkoly následují.

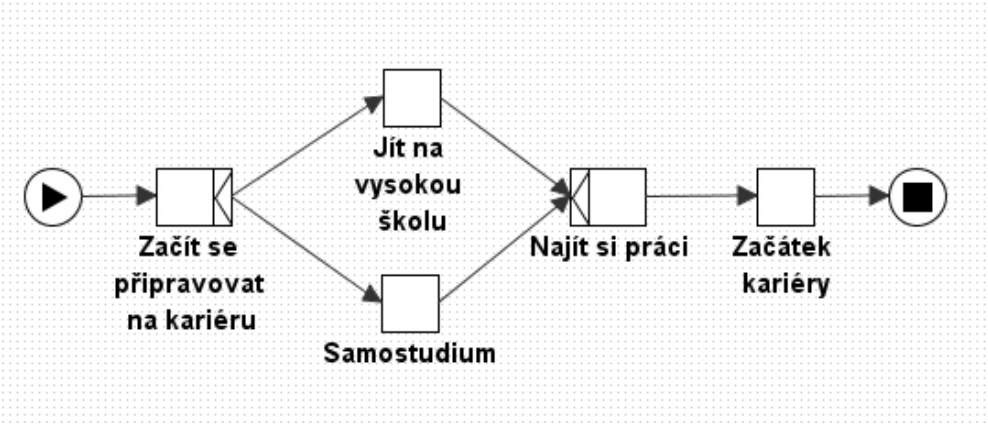
Každý úkol může mít zadán vstupní a výstupní port („Join“ a „Split“). Výstupní port říká, které další úkoly mohou být prováděny. Vstupní pak, jaké úkoly musí být splněny, aby se mohlo s vykonáváním úkolu započít. Jsou čtyři možnosti, jak port definovat:

- **NONE** – Další úkol se může začít provádět, resp. předchozí úkol musí být dokončen.
- **AND** – Začnou se vykonávat všechny další úkoly, resp. všechny předchozí musí být dokončeny.
- **XOR** – Může se započat provádět lichý počet úkolů resp. lichý počet úkolů musí být dokončen.
- **OR** – Začne se vykonávat alespoň jeden další úkol, resp. alespoň jeden předchozí úkol musí být splněn.

Popis obrázku 2.1.1 (oba porty jsou typu „XOR“):

„Až se začnu připravovat na kariéru, musím se rozhodnout, zda půjdu na vysokou školu nebo budu studovat sám. Najdu si práci až tehdy, pokud právě jedním z těchto způsobů dokončím svá studia. Až si najdu práci, začne mi kariéra.“

1 Aplikaci lze nalézt na adrese http://www.mygrid.org.uk.



Obrázek 2.1.2: Reprezentace úkolů v editoru YAWL

Příklad byl převzat z [5]. Součástí editoru je sofistikovaný validátor. Dokáže například detekovat problémy typu uváznutí procesů (deadlock) již při návrhu. Své uplatnění našel i v praxi (i když v mnohem menší míře než jazyk Scufl). Jako příklad uveďme projekt „YAWL4Film“[2].

2.2 Derivace konceptu

Jazyk YAWL a jeho editor splnily z velké části představy o podobě konceptu systému. Nabízí mentálně srozumitelný přístup k tvorbě složitých úkolů a vztahů mezi nimi. Bylo ale potřeba udělat některé úpravy, aby byl editor skutečně použitelný i pro širokou skupinu vývojářů a interpret nepožadoval prakticky žádnou intelektuální invenci uživatele. Jde především o:

- jen jeden druh úloh, obecně jakkoli komplexních
- více koncových míst
- zavedení zachazeče, který může řídit aktivitu úloh (míst)
- kontrola podmínek předchází vykonání úkolu[3]
- fixní nastavení vstupních a výstupních portů[4]
- Interpret zcela oddělen od editoru

2 Lze nalézt na adrese http://www.yawl4film.com.

3 Změna během implementace – kontrola podmínek probíhá jako samostatný úkol (podkapitola 4.6.1)

4 Vstupní porty jsou typu „OR“ a výstupní typu „AND“

3 Herní systém

Prvotní myšlenkou bylo pouhé vytvoření jakési poziční hry. Postupně se ale rozvinul nápad na celý herní systém, ve kterém by bylo možno hry tvořit a přehrávat. V této kapitole postupně rozebereme, jak se systém koncipoval, co se od něj očekávalo, a jak má být stavěn.

3.1 Obecný pohled

Připomeňme, že smyslem práce je vytvořit herní systém, který by umožnil tvorbu a interpretaci jednoduchých, ale v principu i velmi složitých a komplexních aplikací. Přitom tvorbu her by prováděl **vývojář,** hrál ji **uživatel**. Jako **programátor** bude dále v textu označován ten, co programuje herní systém jako takový. Z toho plyne, že vývoj hry může být ne zcela triviální, kdežto interpretace uživatelsky velmi přirozená.

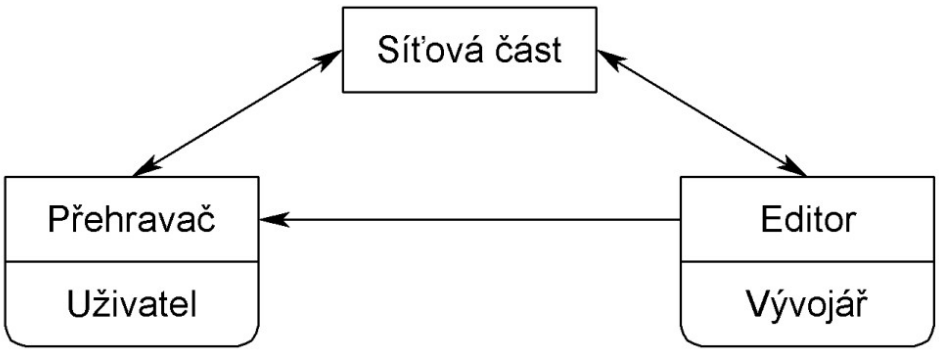
Editor může být složitý do té míry, aby se vývojář byl schopen naučit v něm tvořit.

Přehrávač musí být schopen interpretovat dostatečné množství funkcí, aby vývojáři ve své činnosti nebyli omezeni, ale zároveň aby se editor nestal mentálně nesrozumitelným.

Výstup editoru – jeho interpretace by měla vyžadovat minimální intelektuální invenci uživatele. A to právě podle toho, jak zamýšlí vývojář.

Samozřejmě zde musí být cesta, jak uživateli data poskytnout, a jak uživatel může data přijímat. Bude třeba implementovat přenos dat mezi oběma částmi.

Nejjednodušší pohled na celý herní systém nabízí Obrázek 3.1.1.



Obrázek 3.1.1: Nejjednodušší pohled na systém

3.2 Koncept

Není přirozeně možné postavit editor na libovolnou hru nebo aplikaci – proto se myšlenka přenáší na editor a přehrávač her. Na první pohled jde o velmi omezenou škálu programů.

Při zamyšlení nad tím, kolik aplikací využívá jen množiny míst, přechody mezi nimi a nějaký jednoduchý kontext, se obzory poněkud posunou. Několik způsobů využití zmiňuje kapitola Jiné možnosti využití konceptu. Místo reprezentuje vykonávání nějakého programu nebo jeho části. Aktivních může být více míst najednou. Konfiguraci právě aktivních míst nazvěme **stav systému**.

Zdánlivě si koncept protiřečí. Na jednu stranu má býti stavový, ale jeho součástí má být **kontext** (viz podkapitola 3.3). Jak bylo již uvedeno, místa jsou abstrakcí nějakých akcí programu, ke kterým vývojář přistupuje. Aby byla práce v editoru jasná a mentálně uchopitelná, je od nich kontext oddělen.

Editor je zamýšlen jako kreslící plocha, kde vývojář vkládá místa, vytváří relace mezi nimi a upravuje kontext – resp. zadává místům přístup ke kontextu. Relace udávají pozice předchozího resp. dalšího místa. Každé má pak dvě brány – vstupní a výstupní. Ty reprezentují postupně přechod z předchozího na následující stav. Podrobněji níže v podkapitole Formální popis systému.

Místa jsou již implementované stavební prvky, které jsou kritické a omezují množství vyvýjitelných aplikací. Po vložení místa se má jednoduše přidat jeho obsah (jeho akce). Pokud bychom potřebovali místo simulující dialog, jednalo by se o jakýsi strom frází od postav v kontextu hry. Jak za sebou budou jednotlivé fráze již neřeší celý systém nebo chceme-li množina míst, ale místo samo o sobě. Jak vnitřně implementuje takovou funkčnost je již zcela skryto a vývojáři je ponechán jen jednoduchý editor dialogu (místa).

Stejně tak by tomu bylo např. u místa simulující pexeso. Pexeso je již hotový stavební prvek a lze si představit, že zde vývojář specifikuje jen jak má vypadat, a jak se chovat (např. množina obrázků, rozložení hracího pole, časové omezení apod.). Toto chování neprogramuje, ale přistupuje k němu skrze editor tohoto místa.

Každé místo má mít tak svůj vlastní, vývojářsky příjemný editor.

Případné rozšíření síly a možností editoru a přehrávače je možno účelově rozšířit pouze vytvořením dalšího typu místa (jeho implementací).

3.3 Formální popis systému

Čtenář bude uveden do celého návrhu herního systému. Rozebrány budou jak jeho základní stavební prvky tak vztahy mezi nimi.

3.3.1 Místo

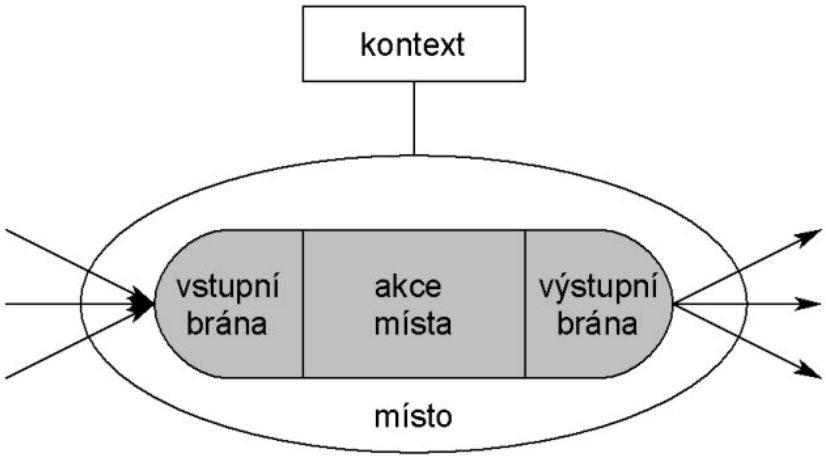
Místo je základní stavební prvek. Spojením dvou a více míst vznikne logická síť. Místo obsahuje vstupní a výstupní bránu. Jejich funkce jsou popsány v následujících kapitolách. Akce místa nazýváme množinu procesů, kterou implementuje daný typ místa. Tento typ může být např. dialog, pexeso, animace, kompas apod.

Samotná implementace místa neobsahuje logiku, která se stará o zpracování podmínek odpálení svých akcí či odpalování následujících míst.

Odpálením rozumíme zprávu danému prvku, že může započít ve vykonávání svých akcí. Výstupní brána spouští vstupní bránu (či brány) a vstupní brány spouští akce implementované pro dané místo. Po skončení akcí místa je spuštěna výstupní brána.

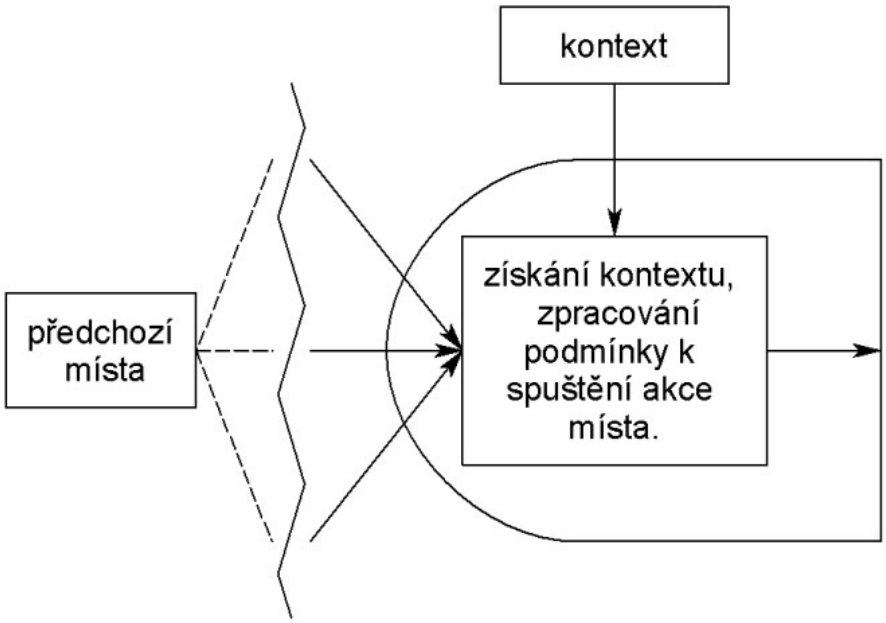
Struktura místa je vyvedena na Obrázku 3.3.1 . Ilustrace znázorňuje také kontext. Tomu se věnuje podkapitola 3.3.4. Logice jeho přidělování pak kapitola 3.3.5.

Nebudou zde dále rozebírány možností akcí místa. Těmi totiž může být prakticky cokoli co vývojáři potřebují ke stavění logických sítí a je možné je vývojářům poskytnout (implementací takového místa). Později bude uveden seznam typů míst implementovaných v rámci této práce.



Obrázek 3.3.1: Struktura místa obecného typu

3.3.2 Vstupní brána



Obrázek 3.3.2: Vstupní brána

Vstupní brána (Obrázek 3.3.2) je abstrakce procesu, který hlídá odpálení akce místa. Předchozí místa zasílají místu zprávy, že má započít v konání svých akcí. Protože je ale třeba ošetřit situace, kdy akce místa očekávají různé kombinace těchto zpráv nebo konfigurace kontextu, je třeba zařídit takovouto obsluhu. Specifikace podmínek je na vývojáři. Jedná se o relativně složitou operaci vyžadující přesnost v návrhu a psaní takových podmínek.

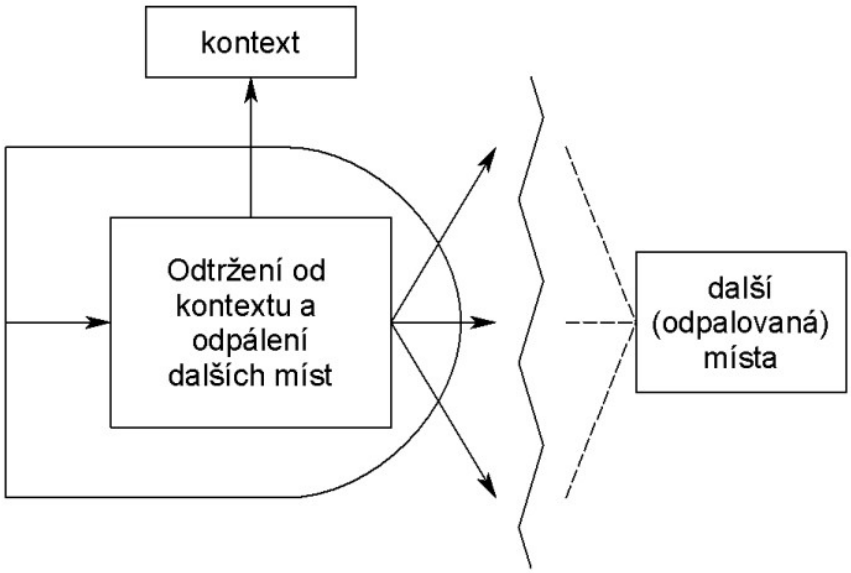
Již na tomto místě přiděluje **zachazeč kontextu** (podkapitola 3.3.5) místu kontext .

Jakmile jsou splněny všechny podmínky, kterými vývojář spuštění akcí místa podmínil, pošle vstupní brána zprávu a může začít provádění akcí místa. Pokud podmínka nebyla splněna, brána se vzdá kontextu a oznámí mu, že nemá spouštět akce místa. Je nutno podotknout, že místo se nestává aktivním příchodem zprávy na vstupní bránu, ale splněním podmínek na vstupní bráně.

3.3.3 Výstupní brána

Jakmile akce místa ukončí své operace, přechází se na výstupní bránu (Obrázek 3.3.3). Důvodem k zavedení je logické oddělení procesů, které se provádějí po ukončení akce. Místo se totiž musí vzdát kontextu a nechat se označit za pasivní.Dalším úkolem výstupní brány je odpálení vstupních bran následujících míst.

Jde o podstatně jednodušší prvek, než je vstupní brána. Žádné podmínky zde nejsou kontrolovány. Vzdání se kontextu požaduje pouze minimální režii ve smyslu uzavření operací nad daty v něm. Odpálení vstupních bran je taky jen jednoduché odeslání zprávy.



Obrázek 3.3.3: Výstupní brána

3.3.4 Kontext

Další součástí je herní kontext. Ten uchovává veškeré informace globálních proměnných – data pro všechny místa. Ve hře může jít například o inventář, telefon, poznámkový blok, fotky navštívených míst, počet bodů apod. Jedná se tedy o jakýsi datový sklad, ze kterého mohou aktivní místa vybírat data. Vybírání neřídí jen samy místa. Povolení k přístupu ošetřuje zachazeč kontextu.

Místa mohou žádat také o přístup k jinému místu. Je na zachazeči kontextu, jak s takovým požadavkem naloží.

Místo kontext získá, jakmile mu je poslána zpráva na ošetření podmínky na vstupní bráně. Dále zůstane přístupný jen pro ošetření vstupních podmínek. V okamžiku splnění podmínky na vstupní bráně je kontext odebrán a zpřístupněn akci místa. Vstupní brána je již zcela ignorována, a to do okamžiku, kdy je opět místo uvolněno odpálením následujících míst nebo oznámením o ukončení. Způsob reprezentace dat uvnitř kontextu jej již zcela na implementaci přehrávače a editoru.

3.3.5 Zachazeč kontextu

Aby bylo možné kontrolovat, jak místa s kontextem komunikují, je zde zachazeč kontextu. Jeho hlavním smyslem je zaručit, aby místa dostala přístup ke kontextu tak, jak požadují a mají dovoleno. Dále aby s ním mohla komunikovat daným způsobem, přidávat data, mazat a podobně.

Musí řešit i konflikty, které mohou nastat. Jde např. žádosti o přístup k zařízením s výhradním přístupem. Implementace zachazeče kontextu je kritická ve smyslu zaručení správné interpretace logické sítě.

Zaručuje přístup ke globálním funkcím. Těmi je myšlena jakási skupina utilit, které zachazeč místům nabízí a je již zcela na implementaci konkrétního zachazeče pro konkrétní množinu typů míst.

Dalo by se konstatovat, že kontext uchovává data a zachazeč kontextu zpřístupňuje operace nad těmito daty. Poslední důležitou úlohou je řízení spouštění dalších míst, odpálí-li místo svou výstupní bránu.

3.3.6 Speciální typy míst

Existují dva případy, kde místo není předcházeno nebo následováno nějakým dalším. Jde o **počáteční** a **koncová místa**. Přitom existuje právě jedno počáteční místo, které má jen výstupní bránu.

Koncových míst může být více. Má je smysl dávat tam, kde končí nějaká větev míst. Koncová místa nemají výstupní brány. Smysl těchto bran je oznámení zachazeči, že hra byla dokončena. Lze si představit, že vývojář bude chtít po dokončení interpretace nabídnout například uložení pozice, zaznamenání počtu bodů apod. Tyto funkce musí poskytovat a podporovat zachazeč kontextu a přehrávač. Nutno upozornit, že vstupní brána zde není upravitelná a je vždy proveditelná. Ošetření podmínky by mělo být funkcí obyčejného místa před tímto speciálním.

Koncové místo ale nemusí znamenat konec interpretace logické sítě. Jiné větve (místa) mohou být nadále aktivní a mohou být interpretovány. Je na vývojáři, aby tímto neuvedl uživatele ve zmatek.

Jiné speciální místo – nazvěme jej „**destruktivní skok**“. Ten nemá ani vstupní ani výstupní bránu. Jeho smyslem je možnost nechat vývojáře zahodit veškerý rozvoj logické sítě (aktivita všech míst najednou ustane), který při interpretaci vznikl a označit jedno místo, které bude odpáleno.

Je to prvek, který umožní vývojáře přinutit uživatele, aby se vrátil do jisté konfigurace sítě a nemusel přitom takový problém složitě modelovat v editoru.

Vývojář si musí uvědomit, že se nejedná o změnu kontextu v celém rozsahu. Pouze se změní seznam aktivních míst. Proměnné v kontextu se nijak neobnovují. Pro úplné ukončení hry je tedy třeba za sebe vložit destruktivní skok a koncové místo.

Posledním výjimečným místem je takové, které nemá žádné akce místa. Dalo by se říci, že obsahuje pouze vstupní a výstupní bránu. Toto místo není zcela nezbytné pro princip fungování (lze implementovat dodatečně bez vlivu na funkčnost systému), ale jeho praktická využitelnost jako kontroly podmínky je natolik užitečná, že je zavedena do základních prvků. V textu bude označováno jako „**prázdné místo**“.

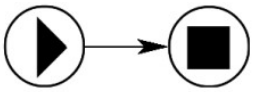
3.4 Několik základních programátorských struktur

Pro získání lepší představy o principu fungování sítí bude uvedeno několik příkladů částí sítí a vysvětlen jejich význam.

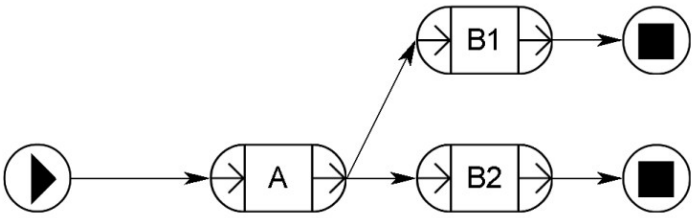
Popis logických sítí bude reprezentován graficky. Zavedeme značky pro stavební prvky. Na obrázku 3.4.1 je postupně zleva doprava vyvedeno počáteční místo, koncové místo, destruktivní skok, prázdné místo, obyčejné místo (nespecifikovaného typu) a **spojnice míst**.

Obrázek 3.4.1: Prvky sítě

Na obrázku 3.4.2 je uvedena nejjednodušší možná síť. Nemá žádné obyčejné místo. Interpretaci nic nebrání. V takové síti by se provedly jen akce, které vývojář zadal do koncového místa. Další ilustrace 3.4.3 zobrazuje jednoduché větvení podmínky. Místo A1 nějak nastaví kontext (např. změní hodnotu proměnné x na 1 s pravděpodobností 50%). Místa B1 a B2 si toto ověří. B1 se odpálí, pokud je x rovno jedné. Jinak se odpálí B2.

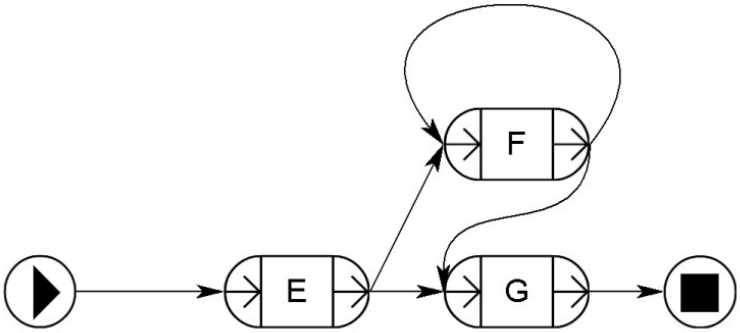


Obrázek 3.4.2:
Nejjednodušší možná síť

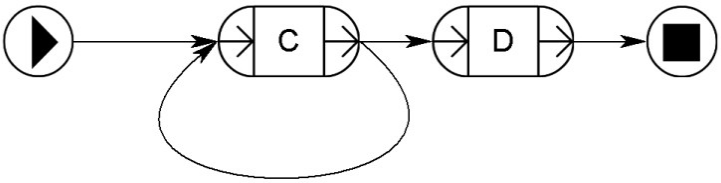


Obrázek 3.4.3: větvení podmínky

Dále jsou vyvedeny cykly (obrázky 3.4.4 a3.4.5) „while“ a „do while“. Soustřeďme se pouze na první z nich. Lze si představit, že místo E je zde pro nastavení určitých proměnných kontextu. Místo F je tělem cyklu a G místo odpálitelné právě pokud je místo F neodpálitelné. Podmínky se hlídají na vstupních branách obou těchto míst.

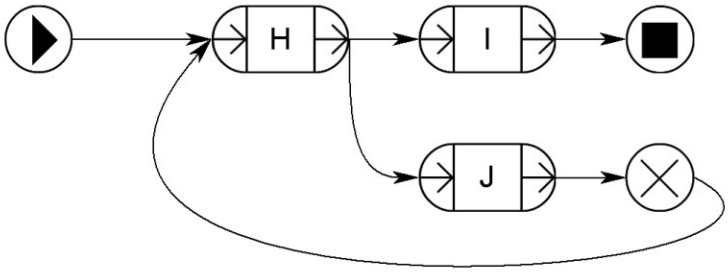


Obrázek 3.4.4: cyklus „while“



Obrázek 3.4.5: cyklus „do while“

Poslední ukázka (3.4.6) nastiňuje použití destruktivního skoku. Jak je vidět zde, koncový stav neznamená nutně konec hry (interpretace). Pokud bude místu J trvat zpracování akce místa déle, než místu I, proběhne hra do koncového místa dříve, než se může provést destruktivní skok. Jestliže místo J odpálí svou výstupní bránu, bude aktivita sítě zapomenuta a následně zaktivováno právě jedno místo (zde místo H). V případě nutnosti aktivace množiny míst je toto třeba ošetřit předsunutým místem, které danou množinu odpálí.



Obrázek 3.4.6: destruktivní skok

4 Implementace obecně

Kapitola se věnuje popisu implementace společných částí systému. Rozebírá postupně jednotlivé důležité části konceptu, obsahuje specifikace implementovaných typů míst a změny v implementaci oproti návrhu. Na závěr krátce pojednává o struktuře generovaných dat.

4.1 Výběr implementačních jazyků

Aby byl zaručen co nejširší okruh vývojářů, měl by editor být co nejvíce otevřen různým platformám. Jde o základní předpoklad k obsáhnutí široké komunity eventuálních uživatelů editoru (vývojářů). Pro dosažení takových vlastností je třeba vhodně vybrat implementační jazyk.

Jak bylo řečeno, jazykem pro platformu Android je jazyk Java. Pro přehrávač proto není volby. U editoru v zásadě nic nebrání implementaci v jiném jazyce. Rozhodující je pak pouze formát vstupních a výstupních dat.

Přestože by bylo možné použít prakticky libovolný jazyk schopný produkovat výstupní data pro interpreta (přehrávač) a umožnit jednotlivé funkce editoru, byl vybrán jazyk Java. Nese to s sebou řadu velmi podstatných výhod. Především se tedy jedná o pozdější úpravu obou součástí systému. Při případných změnách programu stačí programátorovi jeden jazyk. Další výhodou je pohodlná a rychlá práce a velmi kvalitní vývojářské nástroje. Síťová část bude velmi jednoduchá. Zcela postačí jednoduchý skriptovací jazyk PHP.

4.2 Kontext

Přehrávač a editor mají společnou podobu kontextu. Proměnné v něm z praktických důvodů byly rozděleny do tří skupin:

- Privátní proměnné místa
- Globální proměnné místa
- Univerzální proměnné

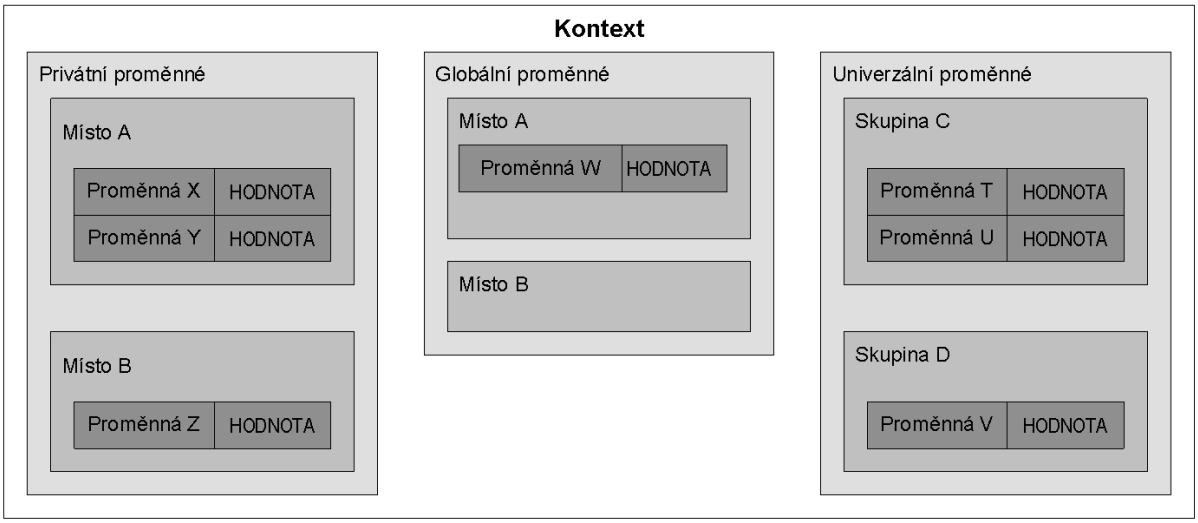
Každé místo má vymezeno prostor, kam může ukládat proměnné, které nemá smysl, aby o nich ostatní místa věděly – aby do nich jiná místa nemohly ani zapisovat ani je číst. Takové proměnné označme jako „privátní“. Zachazeč kontextu nesmí při interpretaci dopustit, aby je místo žádající o privátní proměnné jiného místa dostalo.

Naopak je tomu u globálních proměnných. Jedná se o prostor vymezený každému místu na to, aby zde zveřejnilo některou svou důležitou hodnotu a pro čtení a zápis. Každé místo má své jedinečné jméno. Tím se dá dotazovat na danou proměnnou.

Rozdíl v použití lze najít např. u níže popsaného typu místa „Timer“. U něj lze nastavit možnost, že zobrazí hodiny s odpočtem, nebo ne. Tato informace není poplatná pro jiné místa a jde pouze o nastavení vzhledu. Jinak je tomu u proměnné času, kterou mění. Lze si představit, že ji bude jiné místo chtít využít pro získání nějaké informace (např. počtu dosažených bodů). Taková proměnná pak musí být globální.

Zvláštní skupinou jsou pak proměnné univerzální. Ty nenáleží žádnému místu. Každé místo z nich ale může číst, nebo do nich zapisovat. Jde o jakési globální proměnné herního systému. Místo na jméno místa se zachazeč dotazuje na skupinu v těchto proměnných. V každé skupině je pak množina proměnných, ke kterým lze přistupovat. Místo dokonce může tyto skupiny procházet.

Univerzální proměnné mohou v průběhu přehrávání měnit své skupiny i proměnné v nich. Je také několik skupin, které jsou do kontextu zadávány automaticky. Pro účely přehrávače her jsou zavedeny skupiny „Score“, „Inventory“, „PropertyHeap“ a „Characters“ potřebné postupně pro ukládání bodů za hru, věcí inventáře, nespecifikovaných proměnných a postav. Skupiny lze jednoduše editovat v jednom z výčtů v API herního systému a přizpůsobit je tak požadovanému typu vyvíjené hry. Pro názornost je na obrázku 4.3.1 vyvedena struktura dat v kontextu.



Obrázek 4.3.1: Kontext

4.3 Proměnné (Properities)

Jak bylo řečeno výše, kontext obsahuje jakési skupiny proměnných. Protože proměnné je třeba uchovávat a hlavně přenášet mezi editorem a přehrávačem, jsou proměnné v kontextu instancemi nějaké třídy implementující rozhraní obecné proměnné. Proměnné, které programátor při vývoji míst bude chtít uchovávat a přenášet, mohou být libovolně obecného typu. Jediné, co musí podporovat je serializace a deserializace dat. Proměnná tak musí umět změnit nebo vrátit svou hodnotu jak v obecné, tak serializované formě.

Pokud při vývoji nového typu místa dojde k nutnosti vytvořit novou (komplexní) proměnnou, má programátor možnost implementovat dané rozhraní a proměnnou začít používat.

4.4 Místa

Sílu herního systému definuje z velké části právě kvalita a množství míst, které je možné upravovat a interpretovat. Zřetel byl proto brán také na to, aby se daly nové typy míst implementovat co nejjednodušeji. Nové místa by se měly dát tvořit bez nutnosti upravovat místa jiná, nebo ostatní libovolné prvky herního systému jako je třeba kontext či jeho zachazeč. Byly vytvořeny dvě rozhraní pro místa. Různé pro editor a přehrávač.

Pro vytvoření nového typu je třeba tyto interfejsy implementovat a zahrnout do takto vytvořených tříd požadované chování. Ohled je přitom také brán na to, že místo nevyžaduje žádný editor. Z návrhu herního systému se jedná například o destruktivní skok, který má pouze jednu funkci – ukončit aktivitu všech míst. Editor si také vede seznam typů míst, která má vývojář přístupné. Poslední krok je tedy zavést definici obou tříd do tohoto seznamu.

Popis obou rozhraní:

- Rozhraní pro třídu v editoru:
  - přiřazení přístupu ke kontextu
  - získání grafického uživatelského rozhraní editoru místa
  - verifikace zadaných údajů
  - příkaz k uložení dat do kontextu editoru
- Rozhraní pro třídu v přehrávači:
  - spusť aktivitu místa
  - ukonči aktivitu místa
  - získej grafický kontext[5]

4.5 Implementované typy míst

Místa v tomto projektu jsou účelově vybrána tak, aby demonstrovala funkčnost herního systému. Pokud by měl být systém opravdu pro tvorbu využíván a produkované hry by měly být nabízeny hráčům, bylo by třeba zapracovat jak na grafické, tak funkcionální stránce.

Seznam míst a jejich popis:

- **ActivityKiller** (Ukončovač aktivit) – V editoru vybere vývojář seznam míst, která mají být vyřazena ze seznamu aktivních a jejich činnost ukončena. Dotaz na ukončení aktivity jiných míst zpracovává zachazeč kontextu, který tento příkaz přímo podporuje.
- **SimpleCondition** (Jednoduchá podmínka) – Tomuto místu lze zadat porovnání nějaké proměnné z kontextu s jinou proměnnou a nebo její serializovanou podobou. Toto místo nahrazuje (Jak je zmíněno v podkapitole 4.6) vstupní bránu míst. Proměnné, které lze převést na číslo se mohou porovnávat dle velikost (větší, menší). Při porovnání serializovaných hodnot jde o test na shodnost řetězců. Pokud je podmínka splněna, místo odpaluje svou výstupní bránu a kontext zaktivuje následná místa. V opačném případě jen oznámí zachazeči, že akce místa byla ukončena.
- **ContextChanger** (Měnič kontextu) – Protože každé místo má z principu přístup ke globálním a univerzálním proměnným, lze je také měnit. Tento typ místa dovoluje vybrat jednu z těchto proměnných a přiřadit jí novou hodnotu. Přiřazovaná hodnota může být opět z kontextu nebo v serializované podobě (Jako u místa „SimpleCondition“). Navíc je možné zvolit možnost zavolat nějakou statickou funkci dané třídy a přiřadit proměnné hodnotu jejího výsledku. Seznam těchto funkcí je uveden ve výčtu, podle kterého editor funkce nabízí.

5 Více o grafickém kontextu přehrávače v podkapitole 6.4

Z implementačních důvodů mají funkce omezen počet parametrů na maximálně deset. Do hodnoty proměnné se přiřazuje opět serializovaná podoba výsledku.

- **Counter** (Počítadlo) – Umožňuje vést globální proměnnou, která se s každým odpálením změní o daný krok v rozmezí zadané minimální a maximální hodnoty. Hodnota kroku může být také záporná.
- **DestructiveJump** (Destruktivní skok) – Typ místa implementující jeho podobu v návrhu. Místo pošle zachazeči kontextu zprávu, aby ukončil veškerou aktivitu míst. Poté odpálí svou výstupní bránu a eventuálně zaktivuje jen vybraná místa. Místo svou funkcí odpovídá místu „ActivityKiller“, ve kterém jsou k ukončení vybrána všechna místa.
- **GPSPath** (Hlídač pozice) – Tomuto místu lze zadat polohu na planetě Zemi, na které se přístroj musí dostat, aby došlo k odpálení výstupní brány. Je třeba zadat také odchylku od tohoto místa. Nelze totiž očekávat, že by uživatel „trefil“ přesně zadanou polohu. Odchylka se zadává jako poloměr kružnice, která má střed v hlídaném místě. Taková kružnice pak vymezuje oblast, do které se uživatel musí dostat.
  Polohu lze zadat buď jako vzdálenost od současné polohy přístroje, anebo absolutně pomocí souřadnic. Tyto hodnoty lze navíc vybírat z kontextu (z univerzálních a globálních proměnných).
- **InventoryChanger** (Měnič inventáře) – Nabízí možnost upravovat položky inventáře. U každé položky lze definovat název, popis a počet kusů. Protože inventář je místo, které by mělo být přístupno během celé doby interpretace sítě, jsou jeho proměnné uchovávány v univerzálních proměnných. Inventář jako takový v editoru najít nelze, avšak přehrávač jej spouští automaticky se startem sítě (více v podkapitole 4.6.3).
  V tomto místě lze měnit počet položek inventáře nebo je z něj vyřazovat.
- **Monolog** (Textová zpráva) – Slouží k výpisu textu na obrazovku. Text lze rozdělit do více „obrazovek“. Najednou je vždy vypsán text pouze z jedné z nich. Místo pak poskytuje ovládací prvky dovolující přejít na výpis dalších textů. Lze zadat také zarovnání odstavců.
- **Start** (Počáteční místo) – Počáteční místo celé sítě jak jej definuje návrh. V celé síti musí být právě jednou. Pokud místo chybí nebo je více počátečních míst, přehrávač vypíše chybovou hlášku a skončí. Toto místo nelze nijak upravovat v editoru. Po spuštění vypíše název hry a zpřístupní tlačítko k pokračování (odpálení výstupní brány).
- **Stop** (Koncové místo) – oznamuje konec hry a pokud bylo, vypisuje počet bodů získaných ve hře. Neznamená nutně konec interpretace sítě. Pokud jsou jiná místa aktivní, přehrávač pokračuje v jejich interpretaci (viz podkapitola 3.3.6).
- **Timer** (Stopky) – umožňují odpočítávat čas ve zvoleném rozsahu a směru. Proměnná času v kontextu může být dle výběru privátní nebo globální. Také lze určit, jestli se mají zobrazit

hodiny v grafickém kontextu, či má běžet „na pozadí“. Po doběhnutí časového intervalu je odpálena výstupní brána.

4.6 Významné odchylky implementace od návrhu

Během implementace došlo ke změně některých koncipovaných pravidel chování systému z důvodu zvýšení uživatelské stravitelnosti či uvrhnutí konceptu do konkrétní podoby přehrávače a editoru her. Kapitola tyto změny popisuje.

4.6.1 Vstupní brána

Vstupní brána je v návrhu popsána jako součást daného místa. Stará se o podmínky a spouští samotné jeho akce. Lze ji vynechat, pokud nemá být žádná podmínka kontrolována a akce se mají spustit za každých okolností. Takových situací může být celá řada a zatěžovat vývojáře nutností vkládat před každé místo podmínku či ji tam jen zobrazovat, je zbytečné.

Z hlediska vývojáře je tak vstupní brána pouze jeden typ místa, které se chová a zachází se s ním stejně jako s každým jiným. Obecně má pak každé obyčejné místo vstupní bránu, která je průchozí vždy.

4.6.2 Ukončování aktivity

Zachazeč má možnost ukončit činnost jakéhokoli místa a také každé místo může požádat o ukončení aktivity – jak své tak jiného. Samozřejmě pokud implementace takového místa dané chování vyžaduje (viz podkapitola 4.5, definice místa „ActivityKiller“).

4.6.3 Počáteční místo

Koncept uvádí, že je pouze jedno počáteční místo, které může být odpáleno. Při implementaci vyvstala otázka, jak naložit s místem typu inventář – tedy místem, které má být přístupné po celou dobu interpretace hry.

Jedna možnost by byla donutit vývojáře zadat toto místo ihned po počátečním, již při práci v editoru. Obecně ale takových míst může být více. Byl zaveden výčet, který definuje skupinu míst, které mají být odpáleny ihned po odpálení počátečního místa. Tuto funkcionalitu zpřístupňuje zachazeč hry (ve vytvořené aplikaci zároveň zachazeč kontextu).

Obecně lze takovým způsobem odpálit jakýkoli typ místa. Lišit se budou od ostatních tím, že nebudou mít možnost přístupu ke svým proměnným v kontextu (jak privátním, tak globálním).

4.7 Popis struktury dat

Veškeré údaje potřebné k interpretaci sítě obsahuje herní kontext, specifikace míst a relace mezi nimi. K serializaci dat dochází v systému ve dvou různých případech. První je uložení sítě tak, jak je reprezentována v editoru. Druhý případ je serializace pro publikování hry – generování popisného souboru pro přehrávač.

Informace v obou souborech:

- **Popis hry** – pole pro název, popis, obtížnost, verzi apod. Pole, kromě názvu, nejsou nikde využity pro čtení. Počítá se s nimi v dalších verzích přehrávače. Uživatel bude pomocí nich smět filtrovat hry uložené na serveru.
- **seznam míst** – každý popis místa zahrnuje seznam následných míst, globální a privátní proměnné. Je zde také název třídy, která v editoru reprezentuje editor místa případně název třídy pro interpreta. Poslední položkou je pak unikátní název místa.
- **univerzální proměnné**

Ošetřit možné zásahy uživatele do XML souboru je prakticky nemožné. Soubory s uloženými sítěmi lze libovolně měnit také mimo editor. Tím může docházet k obcházení zamýšleného chování. Tento problém není zatím nijak řešen, ale koncepce již existuje. Věnuje se jí část podkapitoly 9.6.

5 Implementace editoru

Kapitola popisuje vybrané úseky implementačních problémů a jejich řešení v projektu. Krátce uvádí také některá omezení v aplikaci.

5.1 Přístup ke kontextu

Při vytváření instance editoru některého místa je mu přiřazen přístup ke kontextu. Programátorovi je na rozdíl od tvorby míst pro přehrávač nabídnut celý, tedy včetně privátních proměnných jiných míst. Má se za to, že programátor je s tímto faktem obeznámen a nepočítá s využitím těchto dat při interpretaci.

Důvodem k ponechání celého kontextu byla snaha umožnit programátorům vytvářet editory míst, které mohou mít i zcela jiné uplatnění než popis aktivity pro přehrávač.

Tímto způsobem by mohlo být implementováno místo, které by např. ověřovalo validitu sítě, nabízelo statistiky odhadu velikosti dat k publikaci apod. Jde de-facto o přidávání funkcí editoru skrze implementaci editorů míst. Taková místa by v přehrávači po aktivaci jednoduše jen odpálila svou výstupní bránu.

5.2 Grafické uživatelské rozhraní

Celý vzhled editoru (viz obrázek 8.2.1 v kapitole 8) se sestává ze tří oblastí. První je mřížka, do které se vkládají grafické reprezentace míst a spojnice. Na levé straně sloupec s místy, které je možné do této mřížky vkládat. Rozdělení na základní a pokročilá (Basic places a Advanced places) je z důvodu přehlednosti. Vespod v tomto sloupci lze nalézt tlačítka pro přepínání módu kurzoru (přepínání vrstev pro editaci míst nebo spojnic).

Mřížka svou fixní velikostí plochy také omezuje možnosti vývojářů a limituje počet míst, které lze do jedné sítě vložit. I když hlavní menu editoru nabízí možnost zadat tuto velikost, není implementována funkce, která by změnu vykonala.

5.3 Omezení funkcí

V hlavním menu editoru lze najít několik odkazů (funkcí editoru), které nefungují. Z důvodů časové tísně a nepodstatnosti nebyly dosud implementovány. Editor umí pouze ukládat a načítat uložené sítě, zadávat Popis hry (Game Options) a publikovat hry. Názvy míst musí být stejně jako zbylý text zadáván bez diakritiky.

6 Implementace přehrávače

Kapitola v úvodu krátce pojednává o platformě Google Android a důvodech jejího výběru. Následuje popis některých zajímavých a důležitých částí přehrávače.

6.1 Platforma Android

Android je softwarová platforma a operační systém pro mobilní zařízení, založená na linuxovém jádru. Z počátku byl vyvíjen firmou Google Inc., později uskupením pro vývoj otevřených standardů pro mobilní telefony (Open Handset Alliance) [6].

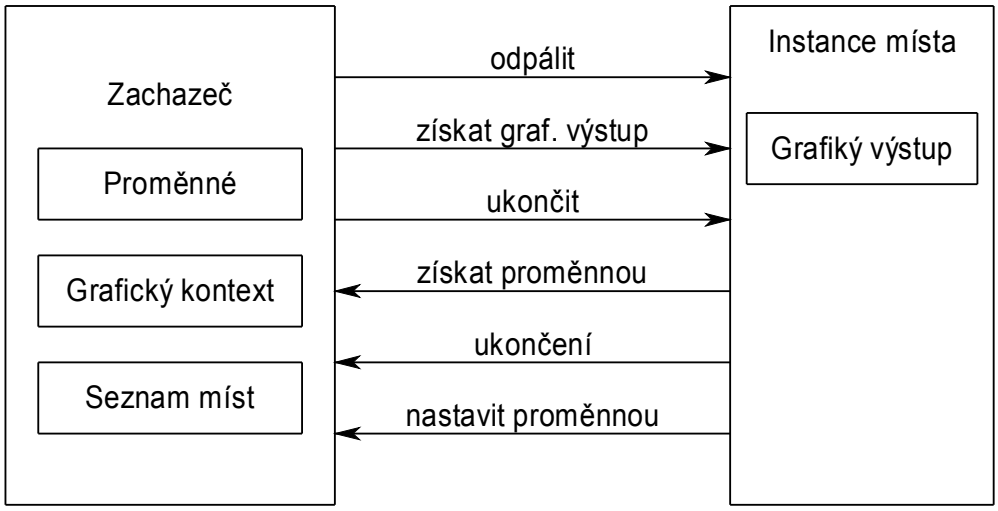
Systém byl koncipován obecně a platforma Android byla vybrána z ryze praktických důvodů. Již v úvodu byly některé z nich zmíněny. Z dalších výhod lze vybrat i programovací jazyk, který je podporován. Jde o jazyk Java. I přesto, že je operační systém postaven na linuxovém jádru, není jiný jazyk než právě Java oficiálně podporován. S výhodou tak lze psát aplikace v objektovém jazyce, což jistě přispěje k rychlosti implementace.

Taková volba s sebou nese ale i úskalí v tom, že platforma obsahuje chyby a emulátor, který je používán pro testování aplikací pro tuto platformy není zcela stabilní a stále se měnící SDK přináší riziko nefunkčnosti v nových verzích.

6.2 Zachazeč kontextu

V přehrávači jak byl implementován, se zachazeč stará vlastně o celý životní cyklus interpretované hry (sítě). Jakmile je spuštěn, vytvoří si instanci parseru dat zvolené hry. Z něj získá seznam míst a naplní kontext proměnnými. Pokud parser zahlásí nenalezení počátečního místa, končí s chybou.

Udržuje seznam míst a ošetřuje případné výjimky, které mohou nastat. Místa mají možnost dát zachazeči vědět, že během jejich vykonávání došlo k neočekávané události či chybě. Interpret zareaguje ukončením činností takového místa a odpálení jeho následovníků. Celý průběh komunikace zachazeče s jeho částmi je vyveden na obrázku 6.3.1.



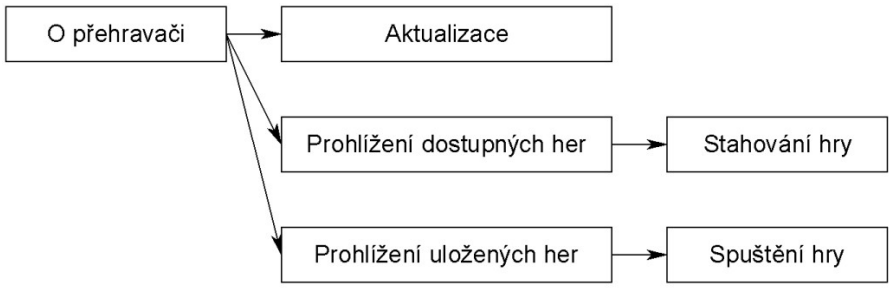
Obrázek 6.3.1: Komunikace zachazeče

6.3 Struktura aktivit (Activities)

Aplikace spouštěné na platformě Android se sestává z tzv. aktivit (Activity). Nejedná se o aktivitu místa v herním systému. Každá významná funkce aplikace by měla být implementována právě takto oddělenou aktivitou. Pro představu uveďme, že pokud by se jednalo o klienta elektronické pošty, byla by jedna aktivita pro prohlížení pošty, další pro vytváření nové zprávy a aktivita pro čtení vybrané položky [7].

I přehrávač her je takto koncipován. Je zde celkem 6 různých aktivit. Jediná z nich (aktualizace) je pro chybové chování emulátoru prozatím deaktivována. Libovolná aktivita může spustit nějakou jinou. Ta je pak vykreslena na stávající. Takto zásobníkově jsou aktivity skládány na sebe. Pokud se aktivita ukončí, přejde se k té, která je v zásobníku pod ní.

Struktura aktivit přehrávače je zobrazena na obrázku 6.4.1. Jak naznačuje obrázek, interpret přehrávače je samostatná aktivita. Ta sama o sobě již další negeneruje. Interpretace sítě tak probíhá „uvnitř“ jediné aktivity a je třeba s tím při implementaci nových typů míst počítat. V zásadě nic nebrání tomu, aby místo spustilo novou aktivitu. Z důvodu neparalelizovatelnosti to ale nelze doporučit při implementaci nových typů míst. Místům je namísto toho nabízen grafický kontext (viz níže), který zpřístupňuje přepínání míst (podkapitola 6.4).



Obrázek 6.4.1: Struktura aktivit přehrávače

6.4 Grafický kontext

Grafický kontext slouží k zobrazování grafiky (aktivních) míst. Pokud je místo interpretováno a má z principu nabízet grafický výstup je nutné, aby k němu měl uživatel přístup. Protože může být aktivních míst současně více, je třeba řešit problém, jak tento výstup zobrazovat ode všech. Problém je v přehrávači řešen následovně:

Zachazeč kontextu se zeptá právě odpalovaného místa, jestli nabízí grafický výstup. Pokud ano, zachazeč přidá do grafického kontextu další obrazovku – grafický výstup. Ten je pak přístupný skrze konkrétní tlačítko. Tato tlačítka spravuje grafický kontext. Pokud je více míst aktivních současně a zároveň podávají grafický výstup, lze mezi nimi přepínat a zároveň je nechat běžet na pozadí.

Uživatel si tedy vybírá, grafiku kterého místa chce vykreslovat. Android definuje vlastní způsob vytváření grafických elementů z tzv. Pohledů (View). Ty se mohou skládat do stromových struktur (jeden pohled se zanořuje do jiného). Grafický výstup všech implementovaných míst je právě tohoto typu. Grafický kontext jim pak přidělí předka a umístí je do něj. Přitom jim nepřekryje ovládací tlačítka pro přístup ke grafice jiných míst.

V dalším vývoji, který bude přehrávač podstupovat bude grafický kontext vrstvy, starající se o přístup k výhradním zdrojům. Nástin této vrstvy je uveden v kapitole 9.5.

7 Síťová část

Síťová část a rozhraní obou prvků systému je řešena pouze minimalisticky a ponechána na další vývoj. Další text mimo jiné shrnuje, jak probíhá komunikace editoru a přehrávače.

7.1 Internetové stránky

V rámci práce byly vytvořeny jednoduché internetové stránky[6]. Na úvodní stránce lze nalézt odkazy na stažení editoru a přehrávače. Každá aplikace pro platformu Android musí být podepsána klíčem vývojáře nebo klíčem vývojářským, a to proto, aby uživatel stahující si tuto aplikaci věděl od koho je a mohl si ji prověřit – nelze podvrhnout jinou aplikací.

Protože je přehrávač ve stádiu vývoje a nelze jej nabídnout široké veřejnosti, není soubor s aplikací podepsán (resp. je podepsán vývojářským klíčem). Aplikaci lze spustit pouze na emulátoru, který je součásti Android SDK verze 1.1. Stránky také obsahují odkaz na návod, jak zacházet s oběma aplikacemi.

7.2 Přístup z editoru a přehrávače

Pokud vývojář dokončil své dílo a chce jej uveřejnit, může tak učinit stiskem tlačítka „Publish“ z hlavního menu. Přenos dat z této strany probíhá skrze FTP protokol a na server jsou přímo zapisována data.

Editor při publikaci produkuje celkem dva soubory. První je soubor ve formátu XML, který obsahuje jen popis hry, tzn. její jméno a další popisné údaje. Druhý pak samotný serializovaný popis sítě jako ZIP archiv. Editor je připraven na to, že si editory míst budou moci ukládat svá data (jako audio soubory, video soubory a jiné) a přenášet si je jako součást specifikace sítě. Tento archiv by pak obsahoval i všechna tato data.

Soubory pro popis hry tak slouží k tomu, aby přehrávač při jejich procházení nemusel rozbalovat jednotlivé archívy. Informace o všech hrách pak vede server jako jeden soubor ve formátu XML, který si přehrávač přečte a vypíše seznam dostupných her. Popis hry mimo jiné uvádí odkaz na archiv s daty s ním spojené.

Na straně přehrávače pak probíhá rozbalení a uložení souboru s popisem sítě do lokálního souborového systému. Přenos již probíhá přes http protokol. Nutno zdůraznit že síťové rozhraní má velké rezervy v zpříjemnění práce ze strany přehrávače. Hrubý koncept dalšího vývoje zmiňuje kapitola 9.2.

6 Adresa internetových stránek: http://game-system.ic.cz

8 Ukázková hra

V rámci práce byla v editoru vytvořena jednoduchá síť na demonstraci funkcionality systému. Kapitola předvádí zpracování scénáře hry na logickou síť v editoru.

8.1 Scénář

Na začátku je hráči oznámeno, že se v jeho okolí rozsypala jablka. Je vymezen čas, ve kterém musí být každé jedno jablko sebráno. Pokud hráč stihne jablko sebrat, přibude mu do inventáře. Pokud ne, dojde k tomu, že dvě jablka ztratí. Pokud ztratí i poslední jablko, hra pro něj končí. Stejně tak hra končí, pokud hráč nasbírá určitý počet jablek.

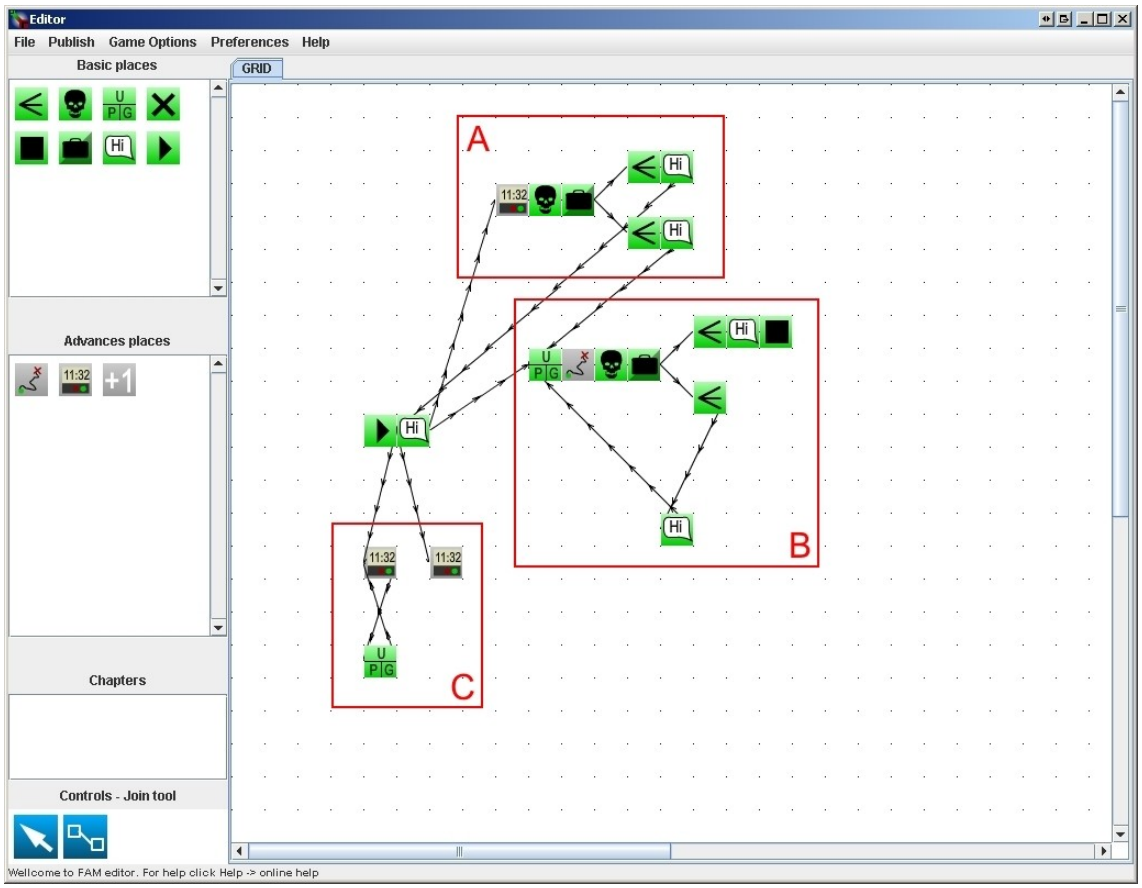
Pozice jablek je náhodně losována do určité vzdálenosti od současné pozice. Tedy od pozice, kde začal se hrou, sebral jablko nebo místo, kde mu vypršel čas na jeho sebrání.

8.2 Síť v editoru

K ilustraci popisu slouží obrázek 8.2.1[7]. Jsou na něm vyznačeny 3 regiony (A, B a C). Blíže popíšeme, co jednotlivé z nich znamenají:

- **Region A –** Místo typu „Timer“ zde hlídá čas, do kdy musí hráč sebrat jablko. Pokud tak neučiní, dojde k odebrání dvou jablek z inventáře. Následné podmínky kontrolují, zda je počet jablek větší než nula či ne. V každém případě se pak hráči tato informace zobrazí v místech typu „Monolog“. Ty pak dle výsledku buď hru započnou znovu nebo vyberou novou pozici na mapě.
- **Region B –** Nejprve se zde vylosuje v místě typu „ContextChanger“ nová pozice na mapě, která se začne hlídat. Jestliže hráč dosáhne v časovém intervalu dané lokace, ukončí se aktivita místa, které hlídá maximální časový interval pro sebrání jednoho jablka. Pokračuje se přidáním jablka do inventáře a prověřením, zda již bylo sebráno jejich požadované množství. Pokud ano, hra končí. Jinak se vybírá nová poloha a hráč hledá další jablko.
- **Region C** – Místa tohoto regionu jsou odpálena hned po počátečním a mají za úkol zapisovat body za hru. Jedno místo typy „Timer“ odpočítává čas po celou dobu trvání hry. Druhé z nich každý časový interval zapisuje hodnotu tohoto času do univerzální proměnné, reprezentující body za hru. Koncové místo pak zobrazuje tento údaj.

7 Jedná se o síť „BPPriklad1.xml“ v adresáři „implementovana-hra“ na DVD a o hru „BPPriklad1“ v přehrávači



Obrázek 8.2.1: Síť hry v editoru

8.3 Jednodušší ukázka

Protože implementovaná hra byla značně nestabilní a do termínu odevzdání práce se nepodařilo přijít na chybu, která toto chování způsobovala, je přiložen ještě jeden jednodušší příklad sítě. V něm je demonstrována funkčnost základních typů míst. Síť lze opět nalézt na přiloženém DVD, nebo si ji přímo stáhnout do přehrávače[8].

8 Síť „BPPriklad2.xml“ a hra „BPPriklad2“ stáhnutelná v přehrávači.

9 Další vývoj

Následující kapitola rozvíjí koncept editoru a přehrávače her tak, jak byly postaveny v rámci této práce. Z větší části polemizuje nad zlepšením použitelnosti v praxi, než nad vylepšením herního sytému jako takového.

9.1 Uživatelská přívětivost editoru

Při stavění sítí je vývojář často uvržen do složitých situací, kdy si změna z důvodů přehlednosti vyžádá přemístění ikon symbolizujících rozdílné typy míst. V takovém případě musí již vložená a specifikovaná místa (jejich grafickou reprezentaci) smazat, přidat na požadované pozice a nově zadat jak jejich parametry, tak relace k ostatním místům.

Pole s mřížkou, do kterého lze v editoru vkládat místa se skládá z několika vrstev. Nyní je přístupná vrstva pro ikony míst a pro reprezentanty relací (spojnice). Mezi těmito vrstvami se musí ručně přepínat, aby aplikace editoru nemusela rozpoznávat, jakou akci vývojář zamýšlí.

Rozvinutý koncept počítá s vylepšeními, které odstraní obě nevýhody. Přibude navíc možnost vkládání komentářů do mřížky – obdoba komentáře klasického programovacího jazyka. Nemají mít vliv na funkčnost.

9.2 Síťové rozhraní

Rozšíření funkcionality této části je zásadní pro pohodlný přístup uživatelů ke hrám. Přehrávač zatím nabízí jen pohled na všechny hry, které byly publikovány. Dalo by se říci, že z hlediska uživatelské přívětivosti je tato část z celého projektu nejslabší.

Všechny parametry popisu hry, které je možné v editoru specifikovat lze s výhodou využít i pro třídění her a jejich selekci dle požadavků, které uživatel zadá. Nový koncept počítá se službou na serveru, díky které by se na něj data nahrávala, a kterou by využíval i přehrávač při stahování. Současný způsob přenosu přes FTP protokol není udržitelný a limituje množství vývojářů her. Nevýhody s sebou nese i nezaručená bezpečnost a závislost na FTP serveru jako taková.

Internetové stránky mají později nabízet strukturovaný přehled her a veškeré hráčské i vývojářské zázemí.

9.3 Validace sítě

Nelze očekávat, že vývojář bude znát a přesně vědět, co musí v síti a editorech míst specifikovat, aby jeho model byl interpretovatelný (validní). Současná podoba obou aplikací takovou funkcionalitu nenabízí a na vývojáři je ponecháno, jak s tvorbou naloží a jak správně síť vystaví. Aby síť byla interpretovatelná, musí být splněny následující podmínky:

- Síť obsahuje právě jedno počáteční místo.
- Počáteční místo nemůže být odpalováno jiným.
- Síť neobsahuje taková místa, k nímž se přes relace míst nedá z počátečního místa dostat.
- Koncová místa nemají na výstupní bráně přiřazena jiné.
- Všechna místa jsou plně specifikovaná (jak požaduje editor místa).

Ve zdrojových souborech lze již najít konstrukce, které kontroly těchto podmínek mají zprostředkovávat. Interfejs editoru míst již požaduje implementaci funkce, kterou se může validátor dotazovat na platnost dat ve specifikaci aktivit místa. Validace dat načítaných souborů má býti řešena jinak (viz podkapitola 9.6).

9.4 Ukládání stavu sítě

V systému, jak byl vytvořen, není možné rozehranou hru uložit a nebo pozastavit její interpretaci. Přitom stav hry definuje jen seznam aktivních míst a proměnné kontextu.

Nabízí se možnost data serializovat stejně, jako je tomu při tvoření souboru pro publikaci na server. Existuje ale významný rozdíl oproti statickému modelu sítě v editoru. Aktivní místa se stávají počátečními místy. Zároveň není možné zahrnout do uložené konfigurace změny, které tato místa provedla na herním kontextu či změnou aktivity sítě. Představme si takový typ místa, které by ve každou časovou jednotku ukončilo aktivitu místa jiného. Tuto funkcionalitu zachazeč kontextu přímo nabízí. Aktivní místo tak způsobí změnu stavu celé sítě a je třeba si pamatovat, jaká byla síť, než se toto místo odpálilo.

Takové chování může sítí prostupovat dále a možností, které by bylo třeba si pamatovat, tak může velmi rychle přibývat přes obecně zvládnutelnou mez. Celý problém je ve stádiu rozvahy a jako vhodné řešení se nabízí si aktivitu všech míst uchovat bez ohledu na to, zda vznikla či zanikla v průběhu stále aktivního místa. Místa by pak měla implementovat metodu, která by uchovala v kontextu proměnné tak, aby při znovu odpálení mohla aktivita pokračovat tam, kde byla přerušena.

9.5 HAL (Hardware abstraction layer)

Při implementaci přehrávače vyvstala otázka, jak zacházet s požadavky míst na zařízení s výhradním přístupem. V rámci tohoto projektu to byla pouze obrazovka, která vykresluje grafiku místa.

Obecně ale při implementaci jiných typů může dojít k obdobné situaci u jiných zařízení jako je kamera, fotoaparát, mikrofon apod. Není zaručeno, jak operační systém s těmito požadavky naloží. Může dojít k případům, kdy daný hardware na zařízení není vůbec přítomen, nebo neposkytuje chtěné vlastnosti (např. rozlišení obrazovky).

Při implementaci nového typu místa nemůže programátor tušit, kolik jeho instancí je v daném okamžiku aktivních. Z vypsaných důvodů bude pro přehrávač v budoucnu postavena vrstva pro přístup k takovýmto zařízení, jejíž součástí se stane i již implementovaný grafický kontext.

9.6 Podepisování her a přenos dat

Při ukládání sítí v editoru a jejich distribuci skrze webové rozhraní nastává problém zabezpečení validity dat. XML dokumenty lze lehce změnit a podvrhnout tak přehrávači či editoru neplatná data. Řešení na úrovni ošetření takových chyb na úrovni parseru souborů nepřichází v úvahu hned ze dvou důvodů. Výrazně tím rostou požadavky na čas procesoru při načítání dat. Druhým souvisejícím úskalím je složitost implementace takového parseru. Dále nutnost jej udržovat a měnit pro různé změny principu fungování obou aplikací. Tato kontrola by se musela provádět nejméně ve dvou případech. A to při ukládání sítí v editoru a načítání her v přehrávači. Současná implementace také nenabízí možnost členění dle autorů. Každý může vydávat svou hru za jinou. Stačí zadat hře stejné jméno a hráč nemá možnost rozeznat, zda jde o hru od daného autora či ne.

Přímočařejší možnost nabízí podepisování a šifrování dat. Při ukládání by editor zašifroval tato data. Ta by pak při změně byla nedešifrovatelná. Jedná se o zabránění podvrhnutí dat editoru. Při publikaci se vypočte kontrolní součet a podepíše se klíčem vývojáře. Přibalí se k datům hry a vše se nyní zašifruje veřejným klíčem serveru. Může dojít k přenosu dat.

Server po příjetí data dešifruje, zkontroluje podpis autora a přečte informace o hře. Následně jsou data opět šifrována , tentokrát veřejným klíčem přehrávače. Ten takto přijatá data může svým privátním klíčem odemknout.

9.7 Jiné možnosti využití konceptu

Hry jsou jen vybraný obor softwaru, který byl zvolen, aby demonstroval fungování herního systému. Původně byl koncept modelován tak, aby dokázal pojmout daleko širší obor tvorby softwaru. Ne jen hry jistého typu.

Vyplatí se jej využít v případech, kdy dekompozice dějů v aplikaci vede na tak omezenou skupinu akcí , že je výhodný poměr vzhledem k množství jejich využití. Ponecháme na fantazii čtenáře, aby si na základě informací tohoto textu udělal představu, jak mohou být v systému implementovány následující aplikace:

- interpret Petriho sítí
- průvodci (přírodou, hrady apod.)
- tvorba dějových linií filmů
- simulace logických obvodů
- klasické adventurní klikací hry
- multimediální interaktivní knihy

10 Závěr

Cílem mé bakalářské práce bylo vymyslet, navrhnout a implementovat stavový herní systém, vytvořit editor a přehrávač, který jej bude využívat a nabízet prostředí pro vývoj her.

Systém jako takový byl navrhnut s ohledem na obecnost tak, aby našel uplatnění i v jiných podobách využití, než je editor a přehrávač her. Sám o sobě nabízí návod, jak stavět editory a interprety (přehrávače) pro různé druhy aplikací, které mohou s výhodou využít jeho principu. Implementace pak poskytuje vývojářskou bázi pro tvorbu nových typů míst a jejich začlenění k používání v editoru a přehrávači.

Aplikace, které jsou také výsledkem práce, slouží pro demonstraci funkčnosti a použitelnosti takového způsobu vyvíjení programů. I přes jeho značnou omezenost, zejména v množství implementovaných typů míst, zprostředkovávají pohled na to, jak lze sítě (hry) touto cestou stavět a využívat.

Při práci na obou aplikacích bylo třeba vypustit z časových důvodů několik funkcí, které by zpřístupnili výsledné produkty širší veřejnosti. Proto také nebyl přehrávač podepsán autorským klíčem a nelze jej spouštět na fyzických zařízeních. I z důvodu nedostupnosti těchto telefonů nemohl být zkoušen jinak, než skrze emulátor.

Výsledky testů provedených na jednoduchých sítích ale prokázaly, že je přehrávač dokáže interpretovat a fungovat tak, jak koncept předepsal – pokud jsou mu poskytnuta validní data.

Prozatím nebyly provedeny žádné uživatelské testy na aplikacích, které by jistě napověděly více o smyslu a správnosti takového způsobu vyvíjení her. Spíše než nemožnost jejich provedení byla důvodem nedokončenost některých (práci zpříjemňujících) prvků obou aplikací.

Software má být dále vyvíjen a nabídnout vývojářům sítí skutečnou bázi pro tvorbu širokého spektra aplikací (tomuto tématu je věnována celá kapitola „Další vývoj“). Dle uplatnění bude zvážena implementace přehrávače i pro další platformy (mobilní i nemobilní).

Literatura

[1] Wikipedia, The Free Encyclopedia: Workflow [online]. c2009 [citováno 16. 05. 2009]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Workflow>.

[2] ASIS, Web Services Business Process Execution Language Version 2.0 [online]. 2009. 2009 [cit. 2009-05-17]. Dostupný z WWW: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>.

[3] VAŠÍČEK, Petr . 5. část: Tvorba BPEL modulu [online]. 2008 [cit. 2009-05-16]. Dostupný z WWW: <http://bpm-sme.blogspot.com/2008/04/5-tvorba-bpel-modulu.html>.

[4] WILLIAMS, Alan R . Taverna 1 [online]. 2009 [cit. 2009-05-16]. Dostupný z WWW: <http://www.mygrid.org.uk/tools/taverna>.

[5] Dumas Marlon, Bradford Lindsay, Aldred Lachlan. YAWL Editor 1.5 User Manual. Queensland University of Technology

[6] Wikipedia, Android (operating system) [online]. 2009. 2009 [cit. 2009-05-13]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Android_(operating_system)>.

[7] Application Fundamentals [online]. 2009. 2009 [cit. 2009-05-17]. Dostupný z WWW: <http://developer.android.com/guide/topics/fundamentals.html>.

Seznam příloh

Příloha 1. DVD:

- kompletní zdrojové soubory ke všem modulům v jazyce Java
- obě části systému jako projekty pro IntelliJ Idea a NetBeans
- binární spustitelné verze a návody na jejich spuštění
- popis ovládání editoru
- soubory internetových stránek
- zdrojová podoba textu bakalářské práce