VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

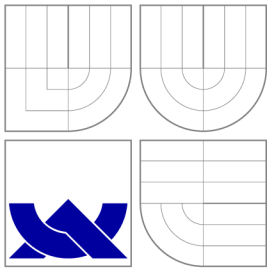
INTERAKTIVNÍ PROSTŘEDÍ PRO VÝVOJ ADVENTURE HER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

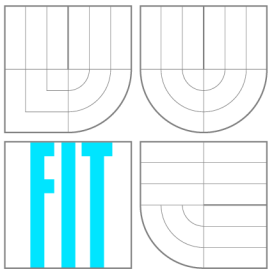
AUTOR PRÁCE
AUTHOR

Bc. MICHAL KONEČNÝ

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTERAKTIVNÍ PROSTŘEDÍ PRO VÝVOJ ADVENTURE HER

INTEGRATED DEVELOPMENT ENVIRONMENT FOR ADVENTURE GAMES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KONEČNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2013

Abstrakt

Cílem této práce je vytvořit na platformě PC interaktivní vývojové prostředí, které by umožňovalo výslednou adventure hru interpretovat na platformě Android. Práce popisuje teorii adventure her a jejich historii. Zabývá se návrhem a implementací knihovny, která je použita v ostatních aplikacích vytvořených v této práci, vývojového prostředí a interpretu pro platformu Android. Dále popisuje ukázkovou hru vytvořenou ve vývojovém prostředí. Nakonec je zde uvedeno testování vývojového prostředí a ukázkové hry na interpretu pro platformu Android.

Abstract

The aim of this thesis is to develop PC-based integrated development environment that would allow to interpret the resulting adventure game on the Android platform. This thesis describes the theory of adventure games and their history. It deals with the design and implementation of a library, that is used in other applications developed in this thesis, a development environment and platform interpreter for Android. It also describes the sample game created in the development environment. At end it contains testing of development environment and sample game on interpreter for the Android platform.

Klíčová slova

Adventure hra, interaktivní vývojové prostředí, Java, Android

Keywords

Adventure game, integrated development enviroment, Java, Android

Citace

Michal Konečný: Interaktivní prostředí pro vývoj adventure her, diplomová práce, Brno, FIT VUT v Brně, 2013

Interaktivní prostředí pro vývoj adventure her

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D.

.....
Michal Konečný
18. května 2013

Poděkování

Chtěl bych poděkovat všem, kdo mě při práci podporovali a umožnili mi na ní pracovat. Především bych chtěl poděkovat svému vedoucímu za kvalitní vedení a podnětné připomínky.

Obsah

Kapitola 1

Úvod

Pojem adventure označuje v anglickém jazyce dobrodružství či zážitek. V českém jazyce se pak pro hry, označené v anglickém jazyce jako adventure games, ujalo označení "adventúry" či adventure hry. Tento pojem je běžně používaný v české herní terminologii, jak je možné si ověřit z následujících zdrojů [3], [2] a [1]. Jedná se o magazíny zabývající se událostmi v herní komunitě. Rozhodl jsem se tedy v rámci této práce používat pojem adventure hra.

Žánr klasických tzv. "point & click" adventure her v dnešní době pomalu upadá. Spousta uživatelů dá raději přednost jiným herním žánrům. Dnešní adventure hry jsou spíše akčního charakteru se spoustou akčních scén a nutností rychlých reakcí uživatele. Přesto je však základní prvek adventure her, silný příběh a řešení různorodých hádanek, obsažen téměř v každé moderní hře. Tato práce se věnuje spíše tomu klasickému pojetí adventure her, které si i v dnešní době plné akčních a rychlých her najdou své příznivce.

Podstatou adventure her je hlavně silný příběh, jedná se tedy spíše o nějakou interaktivní knihu. Podobně, jako téměř každý člověk dokáže psát knihy, měla by i aplikace vytvořena v této práci umožnit tvorbu těchto klasických "point & click" adventure her téměř každému člověku bez nějakých hlubších programovacích znalostí. Tato aplikace by měla umožnit uživateli pohodlný a jednoduchý vývoj her tohoto žánru. Výslednou aplikaci jsem pojmenoval *AGCreator* (**A**dventure **G**ames Creator). Hry vytvořené touto aplikací je však nutné nějak interpretovat. Z tohoto důvodu je obsahem této práce ještě další aplikace, která se stará o interpretaci adventure her vytvořených v *AGCreatoru*. Tato druhá aplikace dostala jméno *AGI* (**A**dventure **G**ames **I**nterpreter). Cílovou platformou pro tuto aplikaci byla zvolena mobilní platforma *Android*, protože se v nynější době jedná o velmi rozšířenou platformu.

Základním stavebním kamenem této práce je především knihovna, v které jsou definovány jednotlivé prvky adventure her, jejich atributy a způsoby komunikace s okolním světem. Tuto knihovnu bylo potřeba navrhnout a implementovat. Knihovna by měla umožňovat uživateli, který s ní bude pracovat, pohodlnou práci s jednotlivými prvky adventure hry a také možnost komunikovat s těmito prvky pomocí komunikačního rozhraní. Toto rozhraní by mělo ulehčit práci při tvorbě interpretu pro tuto knihovnu a práci s modulem, kde budou reprezentována data, která budou popisovat danou adventure hru.

Účelem této práce je tedy vytvořit interaktivní vývojové prostředí pro tvorbu adventure her. Toto prostředí by mělo umožňovat vývojáři měnit obsah hry během hraní. Mělo by sloužit jako jednoduché uživatelské rozhraní pro tvorbu modulu definovaného knihovnou.

Z důvodu nutnosti interpretace adventure her vytvořených v tomto prostředí vznikne další aplikace, která by měla umožňovat interpretaci dat vytvořených ve vývojovém prostředí *AGCreator*. Mělo by se jednat o uživatelsky příjemné prostředí pro hraní těchto her.

Kapitola 2

Současný stav žánru adventure her

V této kapitole je popsán současný stav tohoto žánru. Nejprve je zde probrána historie těchto her. Poté jsou zde probrány principy adventure her. Nakonec se tato kapitola věnuje existujícím vývojovým nástrojům.

2.1 Historie adventure her

Tato kapitola je věnována vývoji žánru adventure her. Prochází celou historií od textových adventure her až po moderní trendy. Čtenář by po přečtení této kapitoly měl znát základní milníky v historii adventure her. Při tvorbě této kapitoly bylo čerpáno z [17] a [12].

2.1.1 Textové adventure hry

První textová adventure hra (tyto hry byly také nazývány "interactive fiction") vznikla v roce 1975. Autorem byl Willie Crowther, jež tímto spojil své zájmy mapování jeskyní a hraní role-playing games [1]. Hra se jmenovala *Colossal Cave Adventure*. Obsahovala několik spojených místností, mezi kterými se uživatel mohl pohybovat. Uživatel také potřeboval několik dalších objektů, s jejichž pomocí se pak dostával do dalších místností. Komunikace s uživatelem byla řešena pomocí jednoduchého parsování textových vstupů od uživatele (např. "get key" nebo "open door"). V roce 1976 Don Woods tuto hru dále vyvíjel a výsledný produkt nazval jednoduše *Adventure*.

První adventure hry se od ostatních her té doby lišily především přístupem k uživateli. Do té doby byla komunikace s uživatelem spíše strojová. Když se třeba objevil příkaz "Zadejte počet míčků:" a uživatel vložil nějaký neplatný vstup, hra mu odpověděla něco ve smyslu "Neplatná hodnota, zadejte znovu (5-100)". U adventure hry byl přístup více osobnější, uživatel zde již nebyl považován za stroj. Pokud uživatel zadal neplatnou hodnotu, hra mu odpověděla třeba něco ve smyslu "Tak tohle nepůjde". Takže uživatel byl mnohem lépe vtažen do hry.

2.1.2 První grafické adventure hry

První grafickou adventure hru vytvořil Warren Robinnet, který byl zaměstnán jako programátor u firmy *Atari*. Jeho hra se jmenovala *Adventure* a vyšla v roce 1979. Skládala se asi z 30 propojených obrazovek, mezi kterými procházel uživatel. Uživatel také neměl od

[1] Jedná se o hraní na hrdiny. Nejznámějším představitelem těchto her je *Dungeons & Dragons*

začátku přístup do všech obrazovek (potřeboval třeba klíč na otevření hradu). Také zde existovaly akce, které se děly mimo současnou obrazovku (netopýr mohl shodit nějaký objekt, který poté uživatel mohl sebrat).

V roce 1980 vyvinula Roberta Williams, zakladatelka společnosti *Sierra*, adventure hru *Mystery House*. Tato hra používala grafický výstup k dokreslení atmosféry a ke komunikaci s uživatelem stále používala textový výstup. Obrázky používané v této hře byly stylizované, jako by je uživatel viděl z vlastních očí.

Grafický výstup těchto adventure her se zlepšoval v závislosti na nových technologiích. Do roku 1984 existoval standard CGA (Color Graphics Adaptor), který nabízel rozlišení 320 na 200 pixelů ve 4 barvách. V roce 1984 vyšel standard EGA (Enhanced Graphic Adaptor), který dosahoval rozlišení 640 na 350 pixelů v 16-ti barvách.

2.1.3 "Point & click" adventure hry

V roce 1984, s příchodem *Apple Macintosh* na trh, se objevil zcela nový typ adventure her. Počítače od firmy *Apple* umožňovaly tzv. "point & click" ovládání, tedy ovládání pomocí myši. S tímto novým způsobem ovládání vznikl i nový způsob ovládání adventure her. Začaly se objevovat tzv. "point & click" adventure hry. Uživatel již nemusel psát příkazy a mohl využít myši či joysticku k ovládání hry.

První adventure hrou, která těžila z tohoto způsobu ovládání je *Deja Vu* od společnosti *ICOM Solutions*. Tato hra však používala "point & click" způsob ovládaní jen na výběr objektů z přenosné množiny prvků (známé také pod názvem "inventář") a pro výběr položky v menu. Nebyl však použit pro navigaci po herním světě.

Více uživatelsky přitažlivější a graficky orientovaný způsob ovládání se objevil v roce 1987 v herním engine *SCUMM*(**S**cript **C**reation **U**tility for **M**aniac **M**ansion). Tento herní engine byl poprvé použit ve hře Maniac Mansion od společnosti *LucasFilm Games* (dnes známé jako *LucasArts*). Uživatel zde mohl ovládat herního avatara nepřímo pomocí kliknutí na herní obrazovku. *SCUMM* také přidal možnost renderovaných animací, které v předchozích adventure hrách neexistovaly.

Postupně se v adventure hrách zlepšuje grafické zpracování, vylepšuje se zvuková stránka hry a občas se objeví i nějaký úplně nový prvek těchto her. V současné době však jsou "point & click" adventure hry založeny stále na stejných principech.

2.1.4 Moderní adventure hry

Moderní adventure hry využívají především výhod stále se zlepšujících grafických technologií. Existují "2.5D" adventure hry, kde se objevuje 3D postava na 2D pozadí. Objevují se i kompletně 3D adventure hry, kde má uživatel volnost pohybu a hru většinou vidí z vlastních očí. I v těchto moderních adventure hrách jsou však stále zachovány základní prvky původních adventure her.

Jedním z moderních žánru, který vznikl z adventure her, jsou akční adventure hry. Kde se objevují prvky adventure her, tedy kombinování předmětů a řešení hádanek, společně s prvky akčních her. V těchto hrách ale většinou převládají akční prvky, a proto se o nich nedá hovořit jako o adventure hrách.

2.2 Principy adventure her

V této kapitole jsou popsány principy adventure her. Jsou zde popsány všechny základní znaky, které jsou pro všechny adventure hry společné. Čtenář by po přečtení této kapitoly měl být seznámen se základními charakteristikami adventure her a měl by porozumět tomu, jak tyto adventure hry fungují. Při tvorbě této kapitoly bylo čerpáno z [12].

2.2.1 Příběh a jeho zasazení

Na rozdíl od ostatních žánrů je u adventure her hodně podstatný příběh a zasazení (zde je myšleno, kde se hra odehrává) celé hry. U ostatních žánrů se většinou příběh ztrácí v množství akce, která se odehrává mezi jednotlivými částmi samotného příběhu. Samotné zasazení příběhu je u ostatních žánrů naprosto nepodstatné pro většinu uživatelů. Adventure hry jsou v tomto směru odlišné. Jejich tempo je pomalejší než u ostatních žánrů, a proto má uživatel čas věnovat se příběhu. O adventure hrách se dá mluvit jako o "interaktivní fikci", z toho důvodu je vlastně příběh a zasazení tohoto příběhu hlavním stavebním kamenem adventure her. Pokud by adventure hra neměla dobrý příběh a zajímavé zasazení, nejednalo by se o dobrou adventure hru. Je také důležité, aby příběh měl nějakou emocionální hloubku a zapojil tak uživatele ještě více do hry.

2.2.2 Řešení hádanek

Dalším důležitým aspektem adventure her je řešení různorodých hádanek a problémů. Aby se uživatel mohl posunout v příběhu dále, je většinou nutné několik těchto hádanek vyřešit. Může se jednat o nějaký hlavolam či použití správného objektu (viz kapitola 2.2.3). Tento aspekt přebrala i spousta dalších herních žánrů, ale jedná se o naprosto jednoduché problémy, které se většinou dají vyřešit během pár minut (například najít správnou kombinaci tří tlačítek).

2.2.3 Používání a kombinování objektů

Jedním ze základních principů adventure her je používání a kombinování různých objektů, které se vyskytují ve světě adventure hry. Pokud uživatel použije nějaký objekt, spustí se předem připravená událost, která většinou umožní uživateli pokročit dále v příběhu.

Aby mohl uživatel objekty používat, musí mít nějakou možnost je přenášet. Většina adventure her nabízí nějakou přenositelnou množinu elementů (dále zkráceně PME) tzv. "inventář". V této množině jsou objekty, které uživatel posbíral během svého cestování po herním světě.

2.2.4 Dialogy s NPC

Ve spoustě adventure her není uživatel sám, v herním světě se často vyskytují tzv. NPC (Non-player characters). S těmito NPC může uživatel vést dialogy. Dialogy jsou tvořeny stromovou strukturou, kdy NPC něco odpoví a podle odpovědi uživatele se vybere větev, kterou se dialog bude ubírat dále.

2.2.5 Rozdělení na scény

U "point & click" adventure her je svět rozdělen na tzv. scény. Tato scéna většinou odpovídá jedné obrazovce na monitoru. Tyto scény mají mezi sebou pevně dané vazby, které určují, jak

se mezi nimi může uživatel pohybovat. I když jsou tyto vazby pevně dané, nemůže uživatel vstoupit do všech oblastí hned na začátku, ale musí většinou vyřešit nějakou hádanku (viz 2.2.2).

Každá scéna je reprezentována konečným automatem, který se přesunuje z jednoho stavu do dalšího podle akcí uživatele. Přičemž je žádoucí, aby při opuštění scény a návratu zpátky uživatel našel scénu ve stejném stavu, jako ji opustil.

2.3 Existující vývojové nástroje

Obsahem této kapitoly je popsat několik již existujících vývojových nástrojů pro tvorbu adventure her. Jsou zde zhodnoceny jejich výhody i nevýhody.

2.3.1 Adventure Game Studio

Jedná se o nástroj pro vývoj adventure her. Vznikl v roce 1997 pod názvem *AdventureCreator* a umožňoval tvorbu adventure her pod systémem *MS-DOS*. Obsahuje vlastní vývojové prostředí a grafický editor pro tvorbu scén typu *WYSIWYG* [2]. Dokáže pracovat s 2D grafikou, zvuky a přehrávat videa ve formátu *AVI*, *OGG* a *WMV*. Herní logika se zde vytváří pomocí skriptovacího jazyka se syntaxí podobnou jazyku Java. Nástroj podporuje i tvorbu NPC (Non-player Characters). Práce s tímto nástrojem je jednoduchá a snadná. Vytvořené hry používají *Direct3D* knihovny a z toho důvodu je lze spustit jen na platformě *Windows* [8].
Výhody:

- Je zdarma.
- Obsahuje vlastní skriptovací jazyk a vlastní grafický editor.
- Přívětivé a lehce pochopitelné vývojové prostředí.

Nevýhody:

- Funguje pouze pod Windows, což platí i pro adventure hry v něm vytvořené.
- Je nutné se učit skriptovací jazyk tohoto nástroje.

[2] What you see is what you got

2.3.2 SCI - Sierra Creative Interpreter

SCI je herní engine používaný firmou *Sierra* pro tvorbu adventure her, který nahradil dříve používaný *AGI* (**A**dventure **G**ame **I**nterpreter). Na oficiální stránce tohoto enginu je možné nalézt mnoho vývojových nástrojů, jak pro *AGI*, tak pro *SCI*. Hry vyvíjené těmito nástroji jsou bohužel vytvořeny pouze pro systém *MS-DOS* a na novějších systémech nejdou bez emulátoru spustit. Tyto nástroje už nejsou v dnešní době dále vyvíjeny [13].
Výhody:

- Je zdarma.
- Obsahuje vlastní skriptovací jazyk a vlastní grafický editor.

Nevýhody:

- Funguje pouze pod systémem MS-DOS.
- Grafické zpracování je v dnešní době zastaralé, obrázky je nutné konvertovat do 256 barev.
- Zastaven vývoj.
- Je nutné se učit skriptovací jazyk tohoto nástroje.

2.3.3 Quest

Quest je vývojový nástroj pro tvorbu textových adventure her. Umožňuje vytvářet textové adventure hry, do kterých je možné vložit i obrázek či video. Hry je možné hrát přímo ve webovém prohlížeči [16].
Výhody:

- Je zdarma.
- Přívětivé a lehce pochopitelné vývojové prostředí.
- Výslednou aplikaci vytvořenou tímto nástrojem je možné hrát přímo v prohlížeči.

Nevýhody:

- Umožňuje vytvářet jen textové adventure hry.

2.3.4 WME - Wintermute Engine

WME je engine pro tvorbu 2D a 2.5D (3D postava na 2D pozadí) adventure her. Pro tvorbu v tomto engine existuje několik grafických nástrojů. Tyto nástroje poskytují komplexní řešení od přehrávání zvuků až po tvorbu vlastního interface. Engine podporuje externí knihovny a obsahuje vlastní skriptovací jazyk se syntaxí podobnou jazyku *C*. Dále umožňuje usnadnění pro zrakově postižené (např. hlasový syntetizér pro čtení textu). Vytvořené aplikace jsou spustitelné na platformě *Windows*. V současné době se vyvíjí nástroj pro tvorbu aplikací pod *iOS* [4].
Výhody:

- Je zdarma pro nekomerční užití, v případě komerčního užití je potřeba zaplatit licenci za knihovnu *BASS*, která se používá pro zpracování zvukových souborů.
- Jsou k dispozici komplexní nástroje.
- Přívětivé vývojové prostředí.
- Dobře dokumentované s velkou komunitou.
- Usnadnění pro zrakově postižené.

Nevýhody:

- Vyžaduje alespoň základní znalosti o programování.
- Funguje pouze pod Windows, což platí i pro adventure hry v něm vytvořené.

2.3.5 Ogre 3D

Ogre (**O**bject oriented **G**raphics **R**endering **E**ngine) je objektově orientovaný 3D grafický engine vytvořený v jazyce *C++*. Pracuje nad knihovnami *OpenGL* a *Direct3D*. Engine je multiplatformový, je možné jej rozjet pod *Windows*, *Linux* a *iOS* [10].
Výhody:

- Je zdarma.
- Komplexní objektově orientovaný 3D grafický engine.
- Knihovna je multiplatformní.

Nevýhody:

- Vyžaduje pokročilé znalosti o programování.

2.3.6 SDL

SDL (**S**imple **D**irectmedia **L**ayer) je knihovna pro práci s audiovizuálními prvky. Poskytuje nástroje pro přístup k myši, klávesnici, zvuku a grafice (*OpenGL*). Knihovna je vytvořena v jazyce *C*, ale je možné ji používat v mnoha dalších jazycích (např.: *Java*, *Lisp*, *Haskell*, *C++*, *C#*, *Ada*). Funguje na spoustě platforem, mezi nimi např.: *Linux*, *Windows*, *FreeBSD*, *NetBSD*, *Solaris* a další [9].
Výhody:

- Je zdarma.
- Obsahuje nástroje pro práci s audiovizuálními prvky.
- Multiplatformní knihovna.
- Podporuje spoustu programovacích jazyků.

Nevýhody:

- Vyžaduje pokročilé znalosti o programování.

2.3.7 Allegro

Allegro je knihovna výhradně zaměřená na tvorbu her. Je napsána pro jazyk *C* a *C++*. Pro uživatele zpracovává nízkoúrovňové úkony (vytvoření okna, zpracování vnějších vstupů, vykreslování obrázku) a vytváří tak abstraktní vrstvu pro uživatele. Allegro podporuje následující platformy *Linux*, *Windows*, *MacOS X*, *iPhone* a *Android* [6].
Výhody:

- Je zdarma.
- Poskytuje abstraktní vrstvu pro práci s nízkoúrovňovými operacemi.
- Knihovna je multiplatformní.

Nevýhody:

- Vyžaduje pokročilé znalosti o programování.

2.3.8 Shrnutí

V dnešní době tedy existuje spousta vývojových nástrojů zaměřených na tvorbu adventure her.

Některé z nich tvoří komplexní řešení pro tvorbu různých žánrů her, viz kapitoly2.3.5, 2.3.6 a 2.3.7. Nejedná se tedy o nástroje vhodné přímo pro tvorbu adventure her, spíše o knihovny vhodné pro tvorbu audiovizuálního software. Uživatel také musí mít dostatečné znalosti programovacích jazyků, v kterých jsou tyto knihovny vytvořeny, aby s nimi dokázal pracovat.

Další z těchto nástrojů jsou omezeny jen na jednu platformu, viz nástroje *AGStudio*, *WME* a *SCI*. Toto omezení je v dnešní době spousty existujících platforem nežádoucí.

Poslední kategorií nástrojů jsou nástroje pro tvorbu pouze textových her. Z výše uvedených patří do této kategorie jen nástroj *Quest*. Vytvářet tyto hry je poměrně jednoduché, ale vizuálně nezajímavé. Proto se tvorbou těchto her v této práci nebudeme zabývat.

Každý z výše uvedených vývojových nástrojů pro adventure hry má své výhody i nevýhody. V této práci bylo snahou využít výhod těchto vývojových nástrojů a zároveň se vyvarovat jejich chybám.

První nevýhodou u výše popsaných nástrojů je nutnost znalosti nějakého programovacího jazyka. To je překážkou především lidem, kteří nemají žádné znalosti programovacích jazyků. Vývojový nástroj vytvořený v této práci by měl umožňovat práci v něm bez znalosti jakéhokoliv jazyka.

Další zjevnou nevýhodou objevující se u většiny výše uvedených nástrojů je omezení pouze na jednu platformu, to většinou platí i pro hry v tomto nástroji vytvořené. Nástroj vytvořený v rámci této práce by měl umožňovat vývoj na všech platformách, které jsou podporovány programovacím jazykem *Java*.

Hlavní výhodou, především u vývojových nástrojů *WME* a *AGStudio*, je jednoduché a přehledné vývojové prostředí. S těmito nástroji bylo možno po krátkém tutoriálu okamžitě pracovat. Tato výhoda by měla být zachována i u vývojového prostředí vytvořeného v rámci této diplomové práce.

Z výše uvedených nástrojů považuji za nejlepší *AGStudio*. Tento vývojový nástroj obsahuje všechno potřebné pro vývoj jednoduchých 2D a 2,5D adventure her. Tato funkcionalita je navíc zaobalena přívětivým a přehledným vývojovým prostředím.

Kapitola 3

Technický úvod

V této kapitole jsou popsány platformy a technologie použité při implementaci. V první části je popsána platforma *Android* a způsob práce v této platformě. Ve druhé části je pak popsána softwarová platforma *Java*. Při tvorbě této kapitoly byly použity zdroje [14], [5] a [11].

3.1 Platforma *Android*

Android je operační systém založený na open source[1] platformě postavený na jazyce *Java*. Za vznikem této platformy stojí společnost *Google*. Je určen primárně pro mobilní zařízení. Operační systém je postaven na linuxovém jádru verze 2.6, které zajišťuje zabezpečení systému, správu procesů a přístup k hardware zařízení. Aplikace vyvíjené na této platformě používají pro přístup k funkcím zařízení *Android API*[2]. Stejně jako softwarová platforma *Java* i *Android* platforma používá vlastní virtuální stroj. Tento virtuální stroj je navržen tak, aby optimalizoval práci s pamětí a hardwarovými prostředky v prostředí mobilních aplikací.

3.1.1 Historie platformy *Android*

Společnost stojící za vznikem platformy *Android* byla založena v roce 2003 pod názvem *Android, Inc.*. V roce 2005 se tato společnost stala dceřinou společností společnosti *Google*. V této době vznikla první verze operačního systému *Android*.

Oficiálně byla platforma představena 5.listopadu 2007. V tento den také vznikla asociace "Open Headset Alliance" zkráceně *OHA*. Členy *OHA* byly velké mobilní společnosti, mezi které patří např. *LG*, *T-Mobile* a *Motorola*. Při vzniku aliance také byl představen první mobilní telefon s operačním systémem *Android*.

Od vzniku platformy *Android* vyšlo již mnoho verzí této platformy. Nejstarší verze, tedy *Android* 1.0, byla poprvé představena v roce 2008. Tato verze obsahovala mimo jednoduchého webového prohlížeče taky několik aplikací od společnosti *Google*, jednalo se například o aplikace *Gmail*, *Google contacts*, *Google calendar* a *Google maps*. Zároveň s touto verzí platformy *Android* byla představena *Android API* level 1, která umožnila vývoj na této platformě. Nejnovější verzí *Android* platformy je v době psaní této práce verze 4.2. *Android API* související s touto verzí je *Android API* level 17.

[1] Jedná se o software s otevřenými zdrojovými kódy

[2] API (Application programming interface) jedná se o rozhraní pro komunikaci s aplikací

3.1.2 Základní stavební kameny *Android* aplikace

V operačním systému *Android* rozlišujeme tři základní druhy aplikací. Komplexní aplikace je však těžké zařadit mezi tyto základní tři druhy, většinou obsahují prvky ze všech tří druhů.

Aplikace na popředí jsou takové aplikace, které jsou užitečné pouze v případě, když jsou v popředí. Tyto aplikace většinou vyžadují komunikaci s uživatelem. Pokud je tato aplikace přesunuta do pozadí, pozastaví se.

Aplikace na pozadí jsou takové aplikace, které vykonávají svou činnost mimo aktuální běh zařízení. Není nutné, aby tyto aplikace byly vyvolány na popředí, aby vykonávaly nějakou činnost.

Posledním druhem aplikací jsou **aplikace s přerušovanou činností**. Tyto aplikace běží většinu času na pozadí, ale občas je nutné, aby o své činnosti informovaly uživatele, popř. od něj vyžadují nějaké potvrzení.

Za základní stavební kameny *Android* aplikací je možno považovat **aktivity**, **služby**, **poskytovatele obsahu** a **záměry**. Z těchto kamenů je postavena každá aplikace vytvořená na platformě *Android*.

- **Aktivita (Activity)**

  Jedná se o prezentační vrstvu aplikace. Každá aktivita typicky reprezentuje jednu obrazovku aplikace. Většina aplikací na platformě *Android* obsahuje více aktivit, které jsou navzájem provázány. Každá aktivita tedy může spouštět další aktivity s cílem provádět jiné funkce. Jednotlivé spuštěné aktivity aplikace jsou uloženy v zásobníku typu **LIFO**. Pokud tedy uživatel stiskne tlačítko pro návrat, ukončí se současná aktivita a do popředí se přesune aktivita předchozí. Životní cyklus aktivity bude popsán v kapitole 3.1.3.

- **Služba (Service)**

  Služba je komponenta aplikace běžící na pozadí. Neposkytuje žádné grafické rozhraní. Služba běží na pozadí, i když uživatel přejde do jiné aplikace. Komponenty je možné vázat ke službě a poté s touto službou komunikovat. Životní cyklus služby bude popsán v kapitole 3.1.4.

- **Poskytovatelé obsahu (Content providers)**

  Poskytovatelé obsahu pracují s daty a zpřístupňují tato data všem aplikacím běžícím na daném mobilním zařízení. Představují jediný způsob, jak sdílet data v rámci aplikace. Neexistuje totiž žádné běžné úložiště, do kterého by měly možnost přistupovat všechny balíčky systému *Android*.

- **Záměry (Intents)**

  Záměr je abstrakce operace, kterou chceme vykonat. Celý aplikační prostor můžeme rozdělit mezi komponenty, což jsou právě aktivity, a zprávy mezi těmito komponentami tedy záměry. Záměr se skládá z akce, která se má vykonat, komponenty, nad kterou má být tato akce vykonána, a aplikace, která má tuto akci vykonat. Záměry můžeme rozdělit na **explicitní** a **implicitní**.

  Explicitní záměr obsahuje pouze akci a komponentu, není zde nutné uvádět aplikaci, která má akci vykonat.

  Implicitní záměr potřebuje, aby byla definována mimo akce a komponenty i aplikace, která má akci provést.

3.1.3 Životní cyklus aktivity

Za životní cyklus aktivity můžeme považovat stavy, kterými aktivita prochází od svého vytvoření až po své zničení, viz obrázek 3.1. Aby bylo možno reagovat na změny těchto stavů, každá aktivita obsahuje tzv. "callback" metody. Tyto metody jsou zavolány, pokud aktivita přechází z jednoho stavu do jiného. Těchto "callback" metod je dohromady šest, jedná se o metody:

- **onCreate** metoda je volána při vytvoření aktivity.
- **onStart** metoda je volána ihned po vytvoření aktivity, popř. pokud byla aktivita zastavena a je znovu spuštěna.
- **onResume** metoda je volána pokud je aktivita přesunuta do popředí.
- **onPause** metoda je volána pokud je aktivita stále viditelná, ale není na popředí.
- **onStop** metoda je volána pokud dojde, ke ztrátě viditelnosti aktivity.
- **onDestroy** metoda je volána pokud dojde k ukončení aktivity, může se jednat o ukončení programové v rámci aplikace či ukončení operačním systémem.

V rámci životního cyklu aktivity můžeme rozpoznat 3 opakující se smyčky:

- **Úplný životní cyklus aktivity**

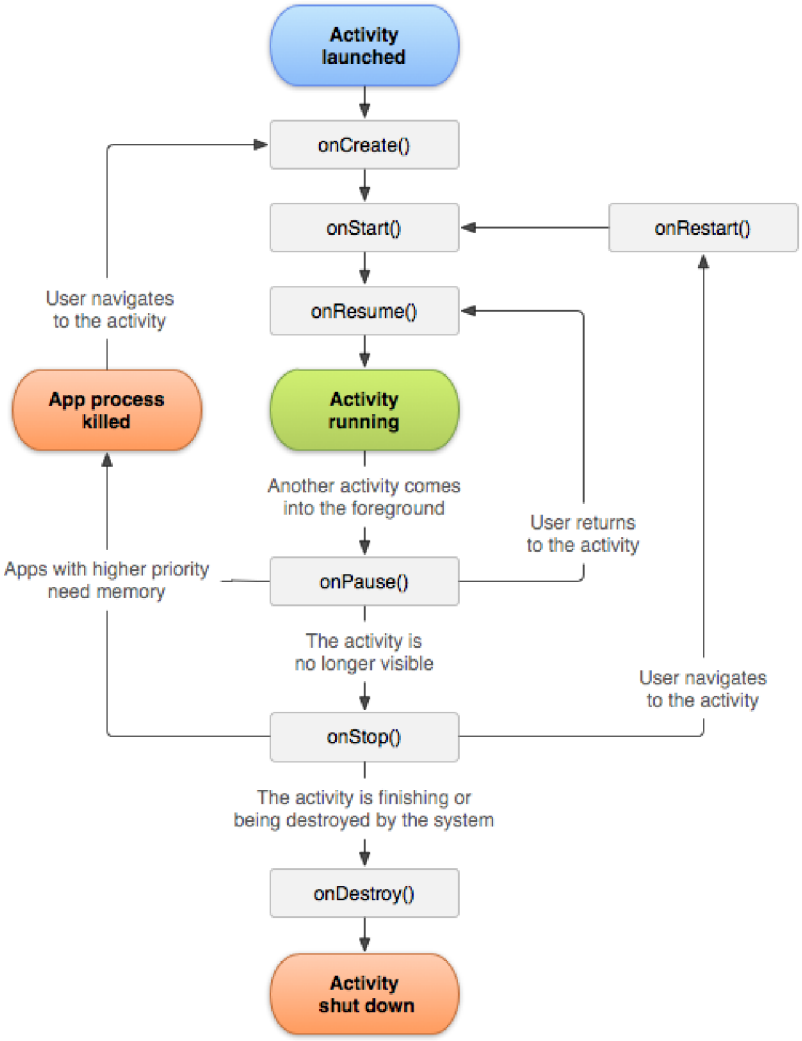
  Jedná se o cyklus aktivity od volání metody **onCreate** až po zavolání metody **onDestroy**. Aktivita při svém vytvoření nastaví svůj "globální" stav (jedná se například o inicializaci grafického rozhraní) a při svém zničení uvolní všechny prostředky, s kterými pracovala.

- **Viditelný životní cyklus aktivity**

  Tento cyklus probíhá mezi voláním metod **onStart** a **onStop**. Jedná se tedy o životní cyklus, kdy je aktivita viditelná a uživatel s ní může komunikovat.

- **Životní cyklus aktivity v popředí**

  V tomto cyklu se aktivita nachází od volání metody **onResume** až po volání metody **onPause**. Po tuto dobu je aktivita v popředí a překrývá všechny ostatní aktivity.



Obrázek 3.1: Životní cyklus aktivity [5]

3.1.4 Životní cyklus služby

Stejně jako aktivity mají i služby svůj životní cyklus, tento cyklus je však jednodušší, než u výše zmíněných aktivit (viz obrázek 3.2). Služby můžeme rozdělit na dva typy:

- **Started**

  Služba tohoto typu je volána některou komponentou aplikace. Služba může běžet na pozadí neomezeně dlouhou dobu a to i v případě zničení komponenty. Obvykle tento typ služby provádí jedinou operaci a nevrací žádný výsledek. Ukončení služby je pak provedeno buď některou z komponent aplikace, nebo se služba může ukončit sama.

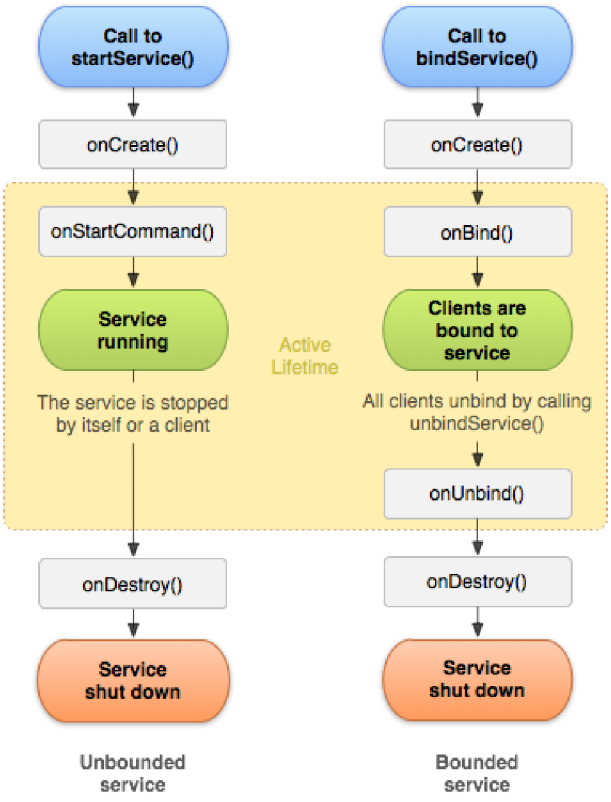
- **Bound**

  Spouštění tohoto typu služby je prováděno stejně jako v předchozím případě. Tento typ služby je pak vázán k jednomu či více klientům (např. komponenty aplikace) a s těmito klienty je navázáno persistentní spojení, které umožňuje zasílat požadavky

službě a přijímat její výsledky. Služba zůstává aktivní, dokud je k ní připojen alespoň jeden klient, poté je zničena.

Podobně jako u aktivit se i zde setkáváme s "callback" metodami. Tyto metody jsou volány pokud je změněn stav služby. Na rozdíl od aktivity je těchto metod jenom pět:

- **onCreate** metoda je volána při vytvoření služby.
- **onBind** metoda je volána, pokud se ke službě chtějí připojit některé komponenty a komunikovat s ní.
- **onStartCommand** metoda je volána, pokud některá komponenta požaduje spuštění služby. Služba potom běží na pozadí.
- **onUnbind** metoda je volána při odpojení komponenty.
- **onDestroy** metoda je volána, pokud dojde k ukončení služby. Ukončení služby se provede v případě, že se služba již nevyužívá, tedy není k ní připojen žádný klient.



Obrázek 3.2: Životní cyklus služby [5]

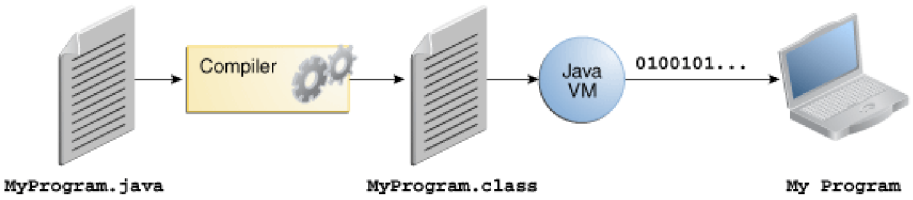
3.2 Softwarová platforma *Java*

Technologie *Java* není jen programovací jazyk, jedná se o samostatnou platformu. Aplikace napsané v jazyce *Java* dokáží běžet na jakékoli platformě, která obsahuje *JVM*[3]. *JVM* běží nad danou platformou společně s *Java API* (Application Programming Interface) a jedná se tedy o samostatnou softwarovou platformu. V současné době *JVM* obsahují platformy *Solaris OS*, *Linux*, *Windows*, *MAC OS* a *Android*.

Programovací jazyk *Java* je objektově orientovaný vysokoúrovňový programovací jazyk. V rámci této práce je jako programovací jazyk použita *Java SE* (Standard Edition). Mezi jeho hlavní výhody patří přenositelnost a bezpečnost, které jsou zaručeny pomocí *JVM*. V této práci byl zvolen jazyk *Java* hlavně kvůli své přenositelnosti a také, že se jedná o jazyk, na kterém je postavena platforma *Android*.

Proces vývoje na softwarové platformě *Java* tedy probíhá následovně, viz obrázek 3.3:

- Vytvoření zdrojových souborů
- Překlad zdrojových souborů do *bytecodes*
- Spuštění v *JVM* cílové platformy



Obrázek 3.3: Proces vývoje v jazyce *Java* [11]

[3]Java Virtual Machine - virtuální stroj, ve kterém běží aplikace

Kapitola 4

Návrh

Tato kapitola je rozdělena na čtyři části. V první části je popsán vztah mezi jednotlivými aplikacemi vytvořenými v této práci a knihovnou.

Ve druhé části je uveden návrh samotné knihovny *AGLib*. Jsou zde popsány jednotlivé prvky knihovny *AGLib* a programovací jazyk používaný touto knihovnou.

Ve třetí části je popsán návrh vývojového prostředí *AGCreator*. Je zde popsán grafický návrh aplikace, formát dat použitých pro ukládání modulu vytvořeného v *AGCreatoru* a návrh jednotlivých částí aplikace. Modulem je nazývána složka, vytvořená při ukládání ve formátu popsaném v této části návrhu.

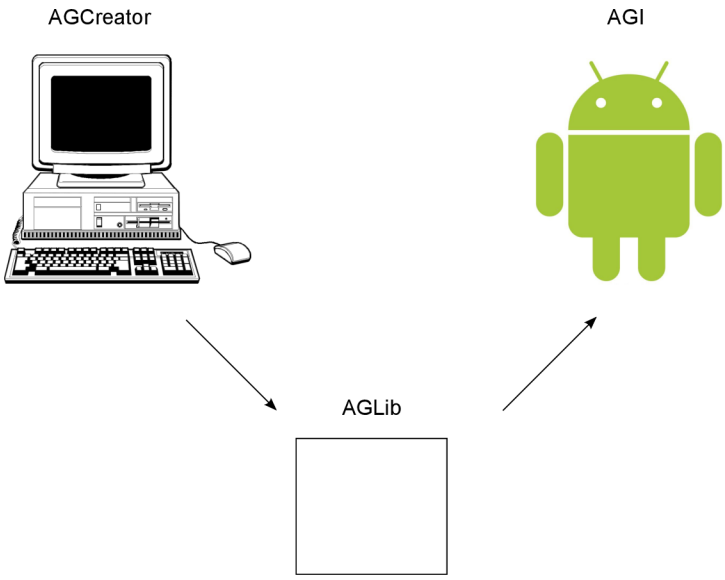
V poslední části je uveden návrh interpretu pro platformu *Android*, který byl pojmenován *AGI*. Je zde popsán cyklus scény tohoto interpretu a ukázán návrh grafického rozhraní.

4.1 Vztah knihovny *AGLib* a aplikací *AGI* a *AGCreator*

Tato práce obsahuje knihovnu a dvě různé aplikace, proto je nutné, aby byl vysvětlen vztah mezi těmito částmi práce. Tento vztah je popsán obrázkem 4.1.

Základem je knihovna *AGLib*, v které je uložena reprezentace dat adventure hry. Knihovna také obsahuje metody pro načítání a ukládání těchto dat do/z formátu modulu (viz kapitola 4.3.3). Knihovnu *AGLib* využívají obě aplikace.

Nejprve je vytvořen projekt ve vývojovém prostředí *AGCreator*. Data tohoto projektu jsou uložena v knihovně. Tato data je možné vyexportovat do formátu modulu. Modul je poté přenesen na zařízení s platformou *Android*, kde je možné pomocí aplikace *AGI* tento modul načíst a interpretovat.



Obrázek 4.1: Vztah knihovny *AGLib* a aplikací *AGI* a *AGCreator*

4.2 Návrh knihovny *AGLib*

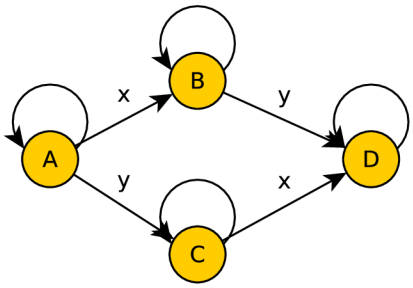
V této části bude popsán návrh jednotlivých součástí knihovny *AGLib*, která tvoří jádro této práce. Budou zde rozepsány principy jednotlivých aspektů knihovny a způsob, jakým by se mělo s jednotlivými aspekty pracovat.

4.2.1 Definice scény

Nejdůležitějším aspektem knihovny je definice scény. Scéna (viz kapitola 2.2.5) popisuje jeden oddělený celek v rámci celé adventure hry. Všechny scény nejsou uživateli přístupné hned, z tohoto důvodu byl navržen tzv. "zámek". Ten slouží k zjištění, zda je scéna pro uživatele přístupná, či nikoliv. V návrhu knihovny byly vytvořeny tři různé typy scén. Jedná se o dialogovou scénu, standardní scénu a příběhovou scénu.

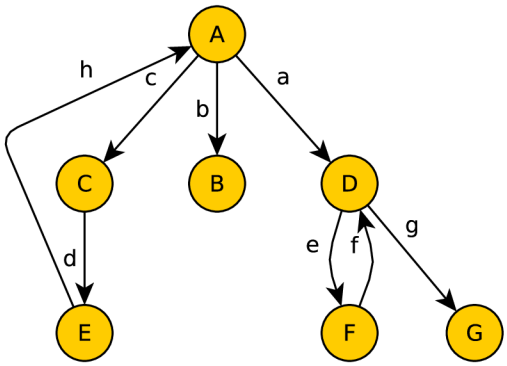
Standardní scéna definuje scénu, kde uživatel může interagovat s objekty, používat PME (přenosnou množinu elementů tzv. "inventář") a přecházet na jiné scény. Standardní scénu je možné popsat konečným automatem (ukázka takového automatu je uvedena na obrázku 4.2), který přechází do dalšího stavu podle akcí uživatele.

Na tomto automatu jsou uvedeny 4 stavy. Jedná se o jednoduchou scénu, která obsahuje minimálně dvě interakce. Začíná se ve stavu A a podle provedené interakce (x nebo y) uživatelem se přechází buď do stavu B nebo C. Z těchto stavů je pak možno dosáhnout koncového stavu D provedením zbývající interakce. V každém stavu je pak možné cyklit, pokud uživatel provádí jiné interakce než x nebo y. Jedná se o interakce, které nevedou k řešení dané scény.



Obrázek 4.2: Ukázka konečného automatu jednoduché standardní scény

Dialogová scéna definuje scénu, ve které je prováděn dialog mezi NPC a uživatelem. V této scéně neexistují žádné objekty, s kterými by uživatel mohl interagovat. Během dialogové scény je scéna řízena dialogovým stromem. Jedná se o n-nární strom, kde rozhovor začíná v kořenu tohoto stromu. Každý uzel tohoto stromu je jeden NPC text. Hrany zase určují jednotlivé reakce uživatele na tento text a podle reakcí uživatele se pak určuje další vývoj dialogu. U dialogového stromu je možné se vrátit do některé z předchozích úrovní stromu. Ukázka dialogového stromu je uvedena na obrázku 4.3.



Obrázek 4.3: Ukázka dialogového stromu

Na ukázce dialogového stromu je vidět, jakým způsobem uživatel interaguje během dialogu. Na začátku dialogu se zobrazí text náležící ke stavu A a uživatel na tento text může reagovat jednou z uvedených možností a,b nebo c. Po výběru jedné z možností se zobrazí text uvedený v následujícím stavu s novou množinou možností reakce uživatele. Dialog je ukončen, pokud uživatel vybere reakci, která je východem do další scény (tato reakce může být umístěna ve kterémkoli stavu v dialogovém stromě). Reakce, které jsou východem ze scény, nejsou v dialogovém stromě znázorněny, protože vedou mimo dialogový strom. Pokud uživatel vybere tuto reakci, znamená to, že dialog je ukončen a dojde k přechodu na jinou scénu.

Příběhová scéna definuje scénu, ve které je vyprávěn příběh adventure hry. Tento typ scény popisuje hlavně změny v příběhu a seznamuje s příběhem uživatele. Na této scéně není žádná možnost interakce a scéna má pevně definovanou cílovou scénu. V knihovně

v podstatě existují dva druhy tohoto typu scén.

Jedná se o scénu **finální**, tedy scénu, která nemá určenou cílovou scénu. Tento druh příběhové scény označuje konec adventure hry. Pokud se uživatel dostane do této scény, je hra považována za dohranou.

Dalším druhem příběhové scény je scéna **vyprávěcí**. Tento druh slouží pouze k vyprávění příběhu a pokud uživatel přejde na tuto scénu, může provést přechod pouze na další pevně danou scénu.

4.2.2 Definice manažera scén

Aby bylo možné přesouvat předměty mezi jednotlivými scénami a udržovat si informace o současné scéně, bylo potřeba navrhnout scénového manažera. Tento manažer v podstatě spravuje obsah PME (přenosnou množinu elementů tzv. "inventář") a obsahuje informaci o současné scéně, na které se uživatel právě vyskytuje. Obsah PME je možné během průchodu hrou měnit, tedy odebírat a přidávat objekty.

4.2.3 Definice objektu na scéně

Kdykoli je možné ve standardní scéně (viz kapitola 4.2.1) s čímkoli interagovat, jedná se o objekt. Objekt se může vyskytovat buď přímo ve scéně nebo v PME. Objektů může na scéně existovat několik typů:

- **Standardní objekt**

  Jedná se o objekt běžně se vyskytující na standardní scéně. Tento objekt obsahuje reakce na jednotlivé interakce od uživatele. Objekt musí také mít několik dalších atributů, jako viditelnost, zda je možné jej přesunout do inventáře nebo zda je možné jej použít.

- **Objekt vedoucí do jiné scény**

  Tento objekt slouží k přechodu do jiné scény, jak standardní, tak dialogové. Tento objekt má všechny vlastnosti standardního objektu, navíc však obsahuje odkaz na další scénu.

Každý objekt je schopen reagovat na určité akce od uživatele. Jedná se o standardní akce vyskytující se v adventure hrách [12]. Akce, na které je objekt schopen reagovat, jsou:

- **Prozkoumat**

  Tato akce umožňuje uživateli prozkoumat daný objekt a získat o objektu více informací.

- **Použít**

  Pokud je možné objekt použít, dojde k vykonání předem vytvořených akcí a konečný automat scény je převeden do dalšího stavu.

- **Sebrat**

  Tuto akci lze vykonat pouze pokud lze objekt přesunout do PME, pokud je tato podmínka splněna, je objekt přesunut do PME.

- **Kombinovat**

  Při kombinaci dochází ke spojení dvou objektů, reakce na tuto kombinaci musí být popsána v podmíněných výrazech jednoho z kombinovaných předmětů. Alespoň jeden z objektů však musí být v PME.

- **Odejít**, **Mluvit**

  Objekt musí být východem do další scény, jinak není možné tuto akci vykonat. V případě akce **Odejít** musí být cílová scéna standardní, v případě akce **Mluvit** dialogová. Při vykonání této akce je proveden přechod na novou scénu, definovanou jako cílová scéna objektu.

4.2.4 Podmíněné výrazy

Aby bylo možné implementovat složitější konstrukce v adventure hře (např. při interakci s jedním objektem se změní viditelnost druhého), bylo nutné navrhnout v knihovně podmíněné výrazy. Podmíněné výrazy jsou vyhodnocovány jako standardní logické výrazy. Při vytváření složitějších podmíněných výrazů je možné použít logické spojky *AND* a *OR*. Každý jednotlivý podmíněný výraz může být převeden do negace pomocí logického operátoru *NOT*. Na rozdíl od logických výrazů neumožňují tyto podmíněné výrazy použití závorek. Jsou vyhodnocovány zleva doprava. Každý podmíněný výraz musí náležet k nějakému objektu (viz kapitola 4.2.3) či reakci v dialogové scéně (viz kapitola 4.2.1). V rámci knihovny *AGLib* existují tři typy podmíněných výrazů, které je možné vzájemně spojovat pomocí logických operátorů:

- **Podmíněný výraz obsahující dynamické atributy** (viz kapitola 4.2.5)

  Tento výraz porovnává dynamický atribut nějakého objektu či scény v adventure hře na rovnost s hodnotou. Jsou porovnávány pouze dynamické atributy, protože statické atributy jsou v průběhu adventure hry neměnné.

- **Proměnný podmíněný výraz**

  Jedná se o podmíněný výraz obsahující proměnnou, tato proměnná je porovnána s hodnotou. V rámci návrhu této knihovny je možné použít následující operátory pro porovnání: $<$, $>$, $==$, $<=$, $>=$.

- **Podmíněný výraz reagující na akci od uživatele**

  Podmíněný výraz je splněn, pokud je s daným objektem provedena specifikovaná akce. Tyto akce odpovídají akcím, které jsou uvedeny v kapitole 4.2.3. Navíc se zde vyskytuje akce **reakce**, která je vyvolána při zvolení reakce v dialogovém stromě.

Podmíněné výrazy v knihovně *AGLib* také obsahují tělo, které se vykoná, je-li podmíněný výraz splněn. Tělo podmíněného výrazu se skládá z příkazů, kterých může být libovolné množství. Příkazy mohou být dvojího typu:

- **Proměnné příkazy**

  Jedná se o příkazy, které mění obsah proměnné. V návrhu knihovny *AGLib* je možné s proměnnými pracovat pomocí následujících matematických operací: $+$, $-$, $\div$, $\times$ a $=$. Poslední uvedený operátor pouze nastaví proměnou na pevně danou hodnotu. Každý příkaz může provést pouze jednu matematickou operaci.

- **Příkazy pro změnu dynamických atributů**

  Tyto příkazy nastaví dynamický atribut specifického objektu či scény na předem definovanou hodnotu.

4.2.5 Statické a dynamické atributy objektů a scén

Každý objekt (viz kapitola 4.2.3) i scéna (viz kapitola 4.2.1) mají množství atributů. Aby bylo možné bez ztráty ukládat kontext probíhající adventure hry, je nutné tyto atributy rozdělit na statické a dynamické.

Statické atributy jsou takové atributy, které jsou v průběhu adventure hry neměnné, jedná se např. o množinu objektů na scéně či o obrázek náležící k danému objektu. Tyto atributy je možné měnit při vývoji, ale už není možné je měnit v průběhu adventure hry.

Dynamické atributy jsou takové atributy, které je možné v průběhu adventure hry měnit. Mezi tyto atributy patří např. viditelnost objektu či "zámek" scény. Při uložení kontextu probíhající adventure hry je nutné ukládat pouze tyto atributy. Při načtení kontextu pak stačí pouze tyto dynamické atributy znovu nastavit a uživatel se objeví ve hře ve stejném stavu, v jakém hru opustil.

4.2.6 Množina reakcí na nekorektní vstup od uživatele

Nekorektním vstupem od uživatele je zde myšlen takový vstup, který je korektní v rámci knihovny, ale není korektní pro daný objekt. V takovém případě se použije reakce z této množiny. Tento přístup zde existuje proto, aby uživateli poskytoval zpětnou vazbu, pokud se snaží provést akci, která není korektní pro daný objekt. Množina reakcí je navržena tak, aby poskytovala náhodnou reakci. Tato náhodná reakce je vybrána z předem dané množiny těchto reakcí.

4.2.7 Programovací jazyk používaný knihovnou

Programovací jazyk používaný v knihovně *AGLib* je založen na standardu XML [15]. Každý zdrojový soubor popisuje jeden prvek v adventure hře. V této práci se jedná o scénu (viz kapitola 4.2.1) či objekt (viz kapitola 4.2.3). Zdrojový soubor se skládá ze dvou částí. Jedná se o část popisující atributy daného prvku a o část popisující podmíněné výrazy (viz kapitola 4.2.4) vztahující se k danému prvku. Vlastní zdrojový soubor má také manažer scén (viz kapitola 4.2.2), množina reakcí na nekorektní vstup od uživatele (viz kapitola 4.2.6) a jazykový soubor.

Pro formální popis syntaxe jazyka vytvořeného v této práci bylo použito *EBNF*[1], které je také použito pro formální popis syntaxe jazyka *XML* [15]. Syntaxe jednotlivých zdrojových souborů se liší podle typu prvku, který je zdrojovým souborem popsán. Pro lepší přehlednost budou jednotlivé *nonterminály* napsány kurzívou a **terminály** tučným písmem.

V rámci zdrojových souborů je možné používat komentáře *XML* jazyka. Syntaxe těchto komentářů je popsána ve zdroji [15].

Význam terminálů použitých v popisu syntaxe zdrojových souborů:

- **doctype**

  Tento terminál definuje typ dokumentu podle syntaxe XML, tato syntaxe je uvedena ve zdroji [15].

[1] Extended Backus-Naur Form - jedná se o matematický popis syntaxe programovacích jazyků

- **intNumber**

  Terminál definuje celé číslo, které může být kladné či záporné. Je definován následujícím regulárním výrazem [0..9]+.

- **boolean**

  Pomocí tohoto terminálu je definována logická hodnota. Terminál může nabývat hodnot *true* a *false*.

- **string**

  Tento terminál definuje řetězec znaků. Řetězec se musí skládat z *UTF-8* znaků. Pokud by byl jeden znak *UTF-8* označen jako *U*, je poté možné definovat terminál *string* regulárním výrazem *U*+.

- **floatNumber**

  Terminál definuje číslo s plovoucí desetinou čárkou. Může být popsán regulárním výrazem [0..9]+.[0..9]+

- **attribute**

  Každý element jazyka, který obsahuje pouze hodnotu popsanou terminály **floatNumber**, **string**, **boolean** a **intNumber** a je ohraničen *XML* tagy, bude označen jako terminál. Název terminálu bude odpovídat *XML* tagu. Například terminál **name** popisuje *XML* element '<name>'**string**'</name>' nebo prázdný *XML* element '<name />'.

Syntaxe zdrojových souborů objektu na scéně

Počátečním nonterminálem v každém zdrojovém souboru je nonterminál *document*. Tento nonterminál derivuje terminál popisující typ dokumentu **doctype**. Nonterminál *typeContents* definuje jednoznačný typ zdrojového souboru a identifikátor tohoto objektu. Důležitou součástí zdrojového souboru jsou dále atributy a podmíněné výrazy. V nonterminálu *attributes* jsou uvedeny ohraničující *XML* tagy a jednotlivé atributy, které musí zdrojový soubor obsahovat. Každý z těchto atributů musí být uveden pouze jednou, ale nezáleží na jejich pořadí. Jednotlivé terminály popisující atributy obsahují hodnotu tohoto atributu uvozenou v odpovídajícím *XML* tagu. Syntaxe podmíněných výrazů je uvedena dále v této kapitole. Jednotlivé zdrojové soubory mají podobnou strukturu, a proto budou u popisu syntaxe dalších zdrojových souborů popisovány pouze odlišnosti.

document	::=	**doctype** *typeContents*
typeContents	::=	'<item' *objectId* '>' *contents* '</item>'
objectId	::=	'id', '=', '"' *id* '"'
id	::=	**intNumber**
contents	::=	*attributes statements*
attributes	::=	'<attributes>' (**owner usable pickable combineWith image visible dimensionWidth dimensionHeight inInventory xPosition yPosition zPosition**) '</attributes>'

Syntaxe zdrojových souborů standardní scény

Ve zdrojovém souboru standardní scény stojí za povšimnutí nonterminál *exit* popisující východy ze scény, skládá se z identifikátoru objektu, který je východem, a identifikátoru scény, která je cílem. Tento nonterminál je derivován z nonterminálu *exits*, z kterého je možné derivovat buď prázdný *XML* tag nebo libovolný počet *exit* nonterminálů. Podobně je možné derivovat nonterminál *items*.

typeContents	::=	'<standardScene' *objectId* '>' *contents* '</standardScene>'
attributes	::=	'<attributes>' (**image music locked** *exits* *items*) '</attributes>'
exits	::=	'<exits>' *exit*+ '<exits>' | '<exits />'
exit	::=	'<exit>' '<itemId>' **intNumber** '</itemId>' '<sceneId>' **intNumber** '</sceneId>' '</exit>'
items	::=	'<items>' **item**+ '<items>' | '<items />'

Syntaxe zdrojových souborů dialogové scény

Zdrojový soubor dialogové scény v sobě obsahuje i samotné dialogy popsané nonterminálem *dialogues*. Z tohoto nonterminálu je možné generovat jednotlivé dialogy pomocí nonterminálu *dialogue*. Syntaxe každého dialogu je podobná syntaxi zdrojového souboru objektu na scéně. Odlišuje se však v syntaxi jednotlivých atributů.

Dialog může obsahovat libovolné množství reakcí, které jsou označeny nonterminálem *response*. Stejně jako dialog má i reakce podobnou syntaxi jako zdrojový soubor objektů na scéně. Odlišuje se však v syntaxi jednotlivých atributů.

typeContents	::=	'<dialogueScene' *objectId* '>' *contents* '</dialogueScene>'
attributes	::=	'<attributes>' (**image music locked startDialogue currentDialogue** *dialogues*) '</attributes>'
dialogues	::=	'<dialogues>' *dialogue*+ '</dialogues>' | '<dialogues />'
dialogue	::=	'<dialogue' objectId '>' *dialogueAttributes* *statements* '</dialogue>'
dialogueAttributes	::=	'<attributes>' (**owner** *responses*) '</attributes>'
responses	::=	'<responses>' *response*+ '</responses>' | '<responses />'
response	::=	'<response' objectId '>' *responseAttributes* *statements* '</response>'
responseAttributes	::=	'<attributes>' (*owner* **targetDialogue exit exitTarget**) '</attributes>'

Syntaxe zdrojových souborů příběhové scény

Zdrojové soubory příběhové scény se odlišují pouze v nonterminálech *attributes* a *typeContents*.

typeContents	::=	'<cinematicScene' *objectId* '>' *contents* '</cinematicScene>'
attributes	::=	'<attributes>' (**image music locked exitTarget**) '</attributes>'

Syntaxe zdrojového souboru manažera scén

Nonterminál *items* popisuje počáteční PME (přenositelnou množinu elementů, tzv. "inventář") v adventure hře.

typeContents	::=	'<sceneManager' *objectId* '>' *contents* '</sceneManager>'
attributes	::=	'<attributes>' (**currentScene** *items*) '</attributes>'

Syntaxe zdrojového souboru množiny reakcí na nekorektní vstup od uživatele

Jak je vidět z popisu syntaxe neobsahuje zdrojový soubor množiny reakcí na nekorektní vstup od uživatele žádná data. Tento zdrojový soubor funguje pouze jako odkaz do zdrojového souboru jazykových dat.

typeContents	::=	'<badList' *objectId* '>' *contents* '</badList>'
attributes	::=	'<attributes>' *staticMessages* '</attributes>'
staticMessages	::=	'<staticMessages>' **staticMessage**+ '</staticMessages>'
		| '<staticMessages />'

Popis zdrojového souboru modulu

Zdrojový soubor modulu slouží k popisu modulu (viz kapitola 4.3.3), obsahuje pouze atributy a neobsahuje podmíněné výrazy.

typeContents	::=	'<module>' *attributes* '</module>'
attributes	::=	'<attributes>' **version** '</attributes>'

Syntaxe zdrojového souboru jazykových dat

Ve zdrojovém souboru jazykových dat jsou uvedena jazyková data jednotlivých prvků uložených v modulu (viz kapitola 4.3.3). Zdrojový soubor obsahuje referenci na každý prvek v modulu. Každý z těchto prvků má syntaxi odpovídající danému zdrojovému souboru. V atributech těchto prvků jsou uvedeny pouze jazykově relevantní data. Jedná se převážně o názvy objektů či texty zobrazované uživateli. K jazykovému zdrojovému souboru se při načítání přistupuje nepřímo pomocí zdrojových souborů jednotlivých prvků.

typeContents	::=	'<language>' (*module badList sceneManager langContents**) '</language>' | '<language />'
langContents	::=	*item** | *standardScene** | *dialogueScene** | *cinematicScene**
module	::=	'<module>' *moduleAttributes* '</module>'
moduleAttributes	::=	'<attributes>' (**name description**) '</attributes>'
badList	::=	'<badList' *objectId* '>' *badListAttributes statements* '</badList>'
badListAttributes	::=	'<attributes>' *staticMessages* '</attributes>'
staticMessage	::=	'<staticMessage>' (**text soundPath**) '</staticMessage>' | '<staticMessage />'
sceneManager	::=	'<sceneManager' *objectId* '>' *sceneManagerContents* '</sceneManager>'
sceneManagerContents	::=	'<attributes />' *statements*
item	::=	'<item' *objectId* '>' *itemAttributes statements* '</item>'
itemAttributes	::=	'<attributes>' (**name pickUpSound useSound examineSound combineSound useText pickUpText examineText combineText**) '</attributes>'
standardScene	::=	'<standardScene' *objectId* '>' *standardSceneAttributes statements* '</standardScene>'
standardSceneAttributes	::=	'<attributes>' **name** '</attributes>'
cinematicScene	::=	'<cinematicScene' *objectId* '>' *cinematicSceneAttributes statements* '</cinematicScene>'
cinematicSceneAttributes	::=	'<attributes>' (**name text**) '</attributes>'
dialogueScene	::=	'<dialogueScene' *objectId* '>' *dialogueSceneAttributes statements* '</dialogueScene>'
dialogueSceneAttributes	::=	'<attributes>' (**name** *dialogues*) '</attributes>'
dialogue	::=	'<dialogue' *objectId* '>' *dialogueAttributes statements* '</dialogue>'
dialogueAttributes	::=	'<attributes>' (**text** *responses*) '</attributes>'
response	::=	'<response' *objectId* '>' *responseAttributes statements* '</response>'
responseAttributes	::=	'<attributes>' **text** '</attributes>'

Syntaxe podmíněných výrazů

Podmíněné výrazy se vyskytují v téměř každém zdrojovém souboru. Počátečním nonterminálem v podmíněných výrazech je *statements*. Nonterminál *statement* dokáže generovat *conditions* a *commnads*. Z jednotlivých nonterminálů *condition* a *command* se generují další nonterminály podle typu podmíněného výrazu (viz kapitola 4.2.4).

Popis speciálních terminálů vyskytujících se v podmíněných výrazech:

- **action**

  Může nabývat hodnot 'PICKUP' | 'EXAMINE' | 'COMBINE' | 'USE' | 'RESPONSE' | 'GOTO' | 'TALK' ohraničených *XML* tagem odpovídajícím názvu terminálu.

- **booleanOperator**

  Může nabývat hodnot 'AND' | 'OR' ohraničených *XML* tagem '<operator>'.

- **dynAttribType**

  Může nabývat hodnot 'locked' | 'visible' | 'usable' | 'pickable' | 'inInventory' | 'currentScene' ohraničených *XML* tagem odpovídajícím názvu terminálu.

- **equationOperator**

  Může nabývat hodnot 'EQUAL' | 'GREATER' | 'LESSER' | 'GREATEREQUAL' | 'LESSEREQUAL' ohraničených *XML* tagem '<operator>'.

- **mathOperator**

  Může nabývat hodnot 'ADD' | 'MUL' | 'SUB' | 'DIV' | 'SET' ohraničených *XML* tagem '<operator>'.

statements	::=	'<statements>' *statement*+ '</statements>'
		| '<statement />'
statement	::=	'<statement>' *conditions commands* '</statement>'
conditions	::=	'<conditions>' *condition* {**booleanOperator** *condition*}*
		'</conditions>' | '<conditions />'
condition	::=	'<condition>' (*actionCondition* | *dynamicCondition*
		| *variableCondition*) '</condition>' | '<condition />'
actionCondition	::=	'<type>ACTION</type>' **negated action combineWith**
dynamicCondition	::=	'<type>DYNAMIC</type>' **negated id**
		dynAttribType value
variableCondition	::=	'<type>VARIABLE</type>' **negated** '<variable'
		objectId '>' **name** '</variable>' **equationOperator value**
commnads	::=	'<commands>' *command*+ '</commands>'
		| '<commands />'
command	::=	'<command>' (*dynamicCommand* | *variableCommand*)
		'</command>' | '<command />'
dynamicCommand	::=	'<type>DYNAMIC</type>' **id dynAttribType value**
variableCommand	;;=	'<type>VARIABLE</type>' '<variable' *objectId* '>' **name**
		'</variable>' **mathOperator value**

4.3 Návrh vývojového prostředí *AGCreator*

V této kapitole je rozebrán návrh vývojového prostředí. Jsou zde popsány jednotlivé vlastnosti vývojového prostředí, způsob ukládání dat a návrh grafického rozhraní. Vývojové prostředí jsem nazval *AGCreator*.

4.3.1 Popis vývojového prostředí

Vývojové prostředí v sobě kombinuje dvě různá prostředí. První z nich je editační nástroj, umožňující upravovat jednotlivé aspekty adventure hry, a tím druhým je interpret programovacího jazyka. Tyto dvě části spolu dohromady utvářejí interaktivní vývojové prostředí. Interaktivní v pojetí tohoto vývojového prostředí znamená, že je možné procházet příběhem adventure hry pomocí interpretační části a v případě, že je to potřeba, se kdykoliv přepnout do editační části a zpět.

Interpretační prostředí umožňuje adventure hrou procházet a využívat všechny možnosti, které hra nabízí. Při setrvání v této části vývojového prostředí funguje vývojové prostředí jako interpret programovacího jazyka. Uživatel může interagovat s objekty (viz kapitola 4.2.3), přecházet mezi jednotlivými scénami (viz kapitola 4.2.1) atd.

Editační prostředí na druhou stranu umožňuje uživateli měnit jednotlivé aspekty adventure hry. V tomto prostředí je možné vytvořit jakýkoliv objekt či scénu a přiřadit jim potřebné atributy. Při editaci dynamických atributů (viz kapitola 4.2.5) je nutné uchovávat i počáteční hodnoty těchto dynamických atributů. Proto v editačním prostředí existuje každý dynamický atribut dvakrát, jeho počáteční hodnota a současná hodnota.

4.3.2 Návrh grafického rozhraní vývojového prostředí *AGCreator*

Při tvorbě vývojového prostředí bylo nutné navrhnout i grafické rozhraní vývojového prostředí. Toto grafické rozhraní bylo navrženo s ohledem na přehlednost a intuitivnost ovládání. Nákres návrhu grafického rozhraní je uveden na obrázku 4.4. Toto grafické rozhraní se skládá z několika různých komponent.

Popis jednotlivých komponent uvedených na obrázku 4.4:

- **Lišta s menu**

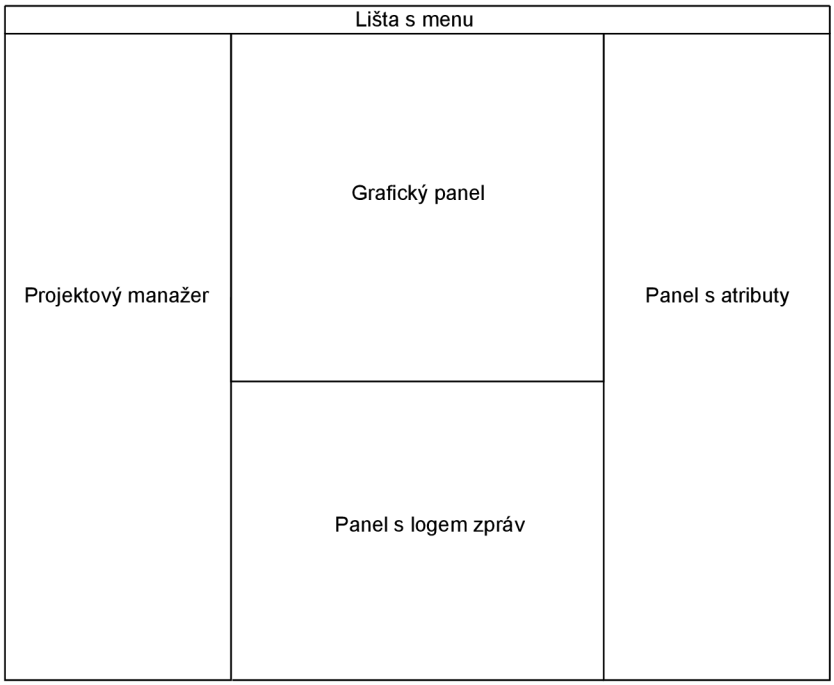
  Tato lišta obsahuje nabídku jednotlivých menu s důležitými funkcemi, mezi tyto funkce patří např. načtení a uložení projektu.

- **Projektový manažer**

  Projektový manažer slouží pro správu projektu. Umožňuje přidávat, odebírat a upravovat scény, objekty a další součásti projektu. Pomocí projektového manažera je také možné přepínat aktivní scénu a spravovat PME (přenosná množina elementů obsahuje objekty, které má uživatel během adventure hry u sebe v tzv. “inventáři”).

- **Grafický panel**

  Grafický panel slouží k zobrazení grafického obsahu současné scény. Zobrazovaný obsah se mění podle zvoleného typu scény (viz kapitola 4.2.1). Ve standardní scéně jsou zobrazeny viditelné objekty na dané scéně a PME. V dialogové a příběhové scéně je zobrazen pouze obrázek dané scény. Možnost interakce v grafickém panelu je dále závislá na současném režimu vývojového prostředí (viz kapitola 4.3.1). V interpretačním režimu je možné interagovat s objekty na scéně, popř. v PME. V editačním



Obrázek 4.4: Návrh grafického rozhraní vývojového prostředí *AGCreator*

režimu je zase možné přidávat nové objekty nebo měnit grafické zobrazení těch současných.

- **Panel s logem zpráv**

  V tomto panelu je umístěn log zpráv. Tento log slouží k zobrazení důležitých informací týkajících se současného projektu načteného ve vývojovém prostředí. Panel slouží především k informování uživatele, že někde v projektu nastala chyba.

- **Panel s atributy**

  V tomto panelu je možné prohlížet a editovat atributy označeného objektu. U dynamických atributů jsou zde zobrazeny dvě různé položky, počáteční verze atributu a současná verze atributu.

4.3.3 Formát ukládaných dat

Projekt vytvořený ve vývojovém prostředí *AGCreator* je uložen ve složce určené při vytváření projektu. Tato složka obsahuje další složku v níž je uložen modul a také binární soubor, kde jsou uloženy současné hodnoty dynamických atributů. Složka modulu obsahující data adventure hry končí znaky `.module`. Složka modulu obsahuje následující soubory a složky:

- *images*

  Ve složce *images* jsou umístěny obrázky, na které se odkazují jednotlivé scény či objekty.

- *src*

  V této složce jsou umístěny všechny zdrojové soubory. Zdrojové soubory mají příponu `.xml`.

- *lang*

  Ve složce *lang* jsou uloženy jazykové soubory. Jedná se o jednotlivé překlady hry.

- *module.xml*

  Jedná se o soubor popisující daný modul. Jeho verzi, název a popis.

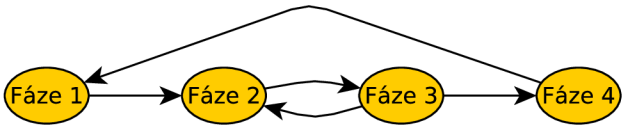
Při uložení kontextu probíhající adventure hry je použit binární formát s příponou `.sav`, který obsahuje serializované hodnoty všech dynamických atributů (viz kapitola 4.2.5). Binární formát je v tomto případě použit, aby nebylo jednoduché měnit hodnoty v uložené pozici.

4.4 Návrh interpretu *AGI*

Cílovou platformou pro tuto práci byla zvolena platforma *Android*. Pro tuto platformu byl navržen interpret *AGI* pracující s knihovnou *AGLib*. Tento interpret slouží jako interpretační vrstva pro data modulu uložená ve formátu uvedeném v kapitole 4.3.3.

4.4.1 Cyklus scény v *AGI*

Jedním cyklem scény jsou v této práci myšleny jednotlivé kroky, které interpret vykoná od zavedení jedné scény (viz kapitola 4.2.1) až po přechod na scénu následující. Během tohoto cyklu je možné interpret kdykoliv ukončit. Cyklus interpretu se skládá z následujících kroků uvedených na obrázku 4.5.



Obrázek 4.5: Cyklus scény v *AGI*

1. **Zavedení scény**

   Při zavedení scény se uživateli zobrazí grafický obsah scény a jednotlivé možnosti interakce. U standardní scény může uživatel interagovat s objekty na scéně nebo v PME (přenositelné množině elementů, tedy objektů, které uživatel může během průběhu hry přenášet u sebe). U dialogové scény jsou pak zobrazeny možnosti reakce na text NPC. U příběhové scény je zobrazeno tlačítko pro pokračování. Po provedení fáze zavádění scény se pokračuje fází 2.

2. **Čekání na vstup od uživatele**

   V této fázi interpret čeká na vstup od uživatele libovolně dlouhou dobu. Vstupem od uživatele může být jakákoliv akce provedená s objektem (viz kapitola 4.2.3). Pokud je obdržen vstup od uživatele, interpret se přesune do fáze 3.

3. **Reakce na vstup od uživatele**

   V případě, že uživatel zadal vstup, je v této fázi zjištěna a vykonána odpovídající reakce. O vykonání reakce se stará samotná knihovna *AGLib*. Pokud je reakce na vstup uživatele přechod na jinou scénu, přejde se do fáze 4. V opačném případě je vykonána odpovídající reakce a návrat do fáze 2.

4. **Přechod na další scénu**

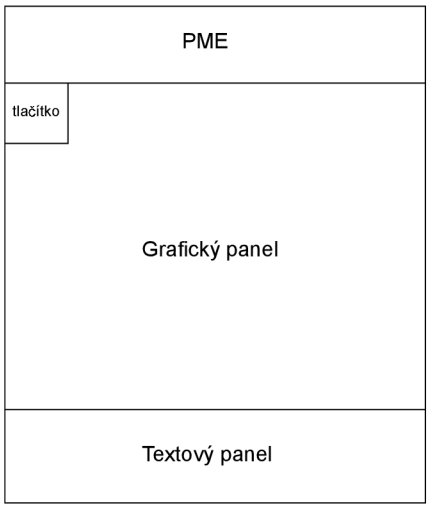
   Pokud byla reakce v předchozí fázi přechod na další scénu, je nalezena požadováná scéna a provede se přechod do fáze 1.

4.4.2 Návrh grafického rozhraní interpretu *AGI*

Při návrhu grafického rozhraní pro platformu *Android* je pro každou aktivitu vytvořeno samostatné grafické rozhraní. V případě této práce je *Android* aplikace rozdělena na následující aktivity:

- **aktivita standardní scény**

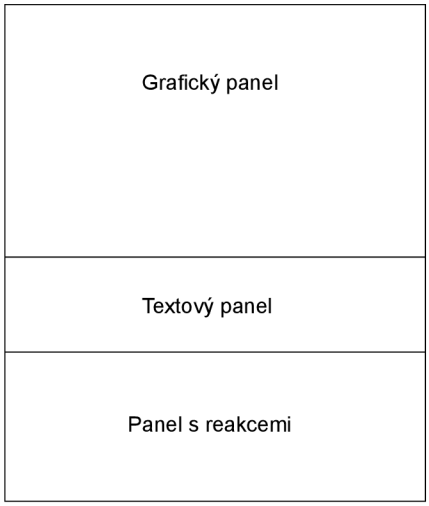
  Na obrázku 4.6 je uveden návrh grafického rozhraní standardní scény. Ve vrchní části je umístěna PME (přenositelná množina elementů, tedy objekty, které uživatel může během průběhu hry přenášet u sebe), která je viditelná po kliknutí na tlačítko uvedené pod ní. Ve střední části je uvedeno grafické znázornění scény, zde je možné interagovat s objekty na scéně. Ve spodní části je uveden textový panel, kde se zobrazují informace pro uživatele.



Obrázek 4.6: Návrh grafického rozhraní standardní scény

- **aktivita dialogové scény**

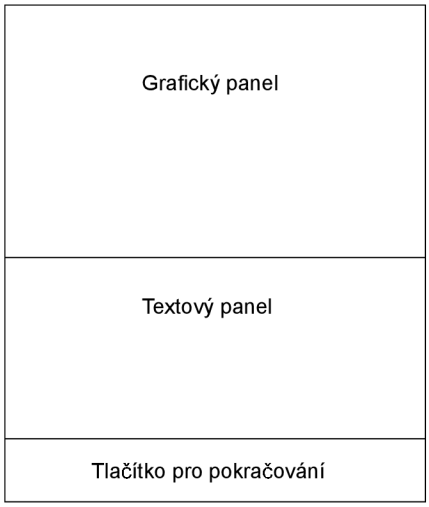
  Na obrázku 4.7 je uveden návrh grafického rozhraní dialogové scény. Ve vrchní části grafického rozhraní je umístěno grafické znázornění scény. Ve střední části je pak umístěn text současného dialogu a ve spodní části je umístěn seznam reakcí na daný dialog.



Obrázek 4.7: Návrh grafického rozhraní dialogové scény

- **aktivita příběhové scény**

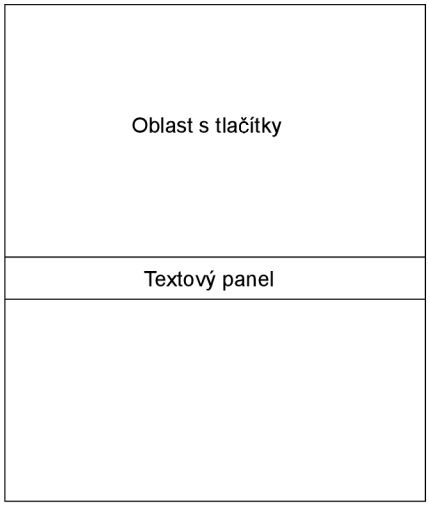
  Na obrázku 4.8 je uveden návrh grafického rozhraní příběhové scény. Ve vrchní části grafického rozhraní je umístěno grafické znázornění scény. Zbylá část grafického rozhraní je pak zabrána textem dané scény. Ve spodní části je pak umístěno tlačítko pro pokračování dále.



Obrázek 4.8: Návrh grafického rozhraní příběhové scény

- **aktivita hlavního menu**

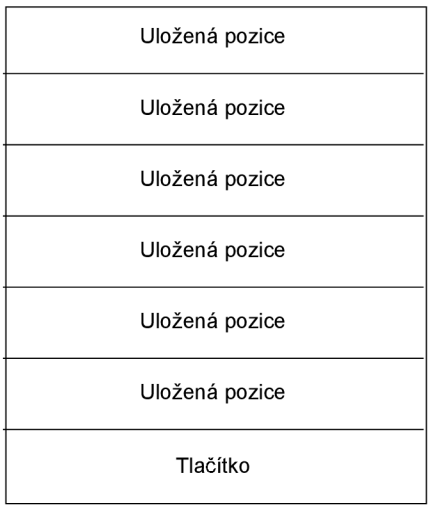
  Na obrázku 4.9 je uveden návrh grafického rozhraní hlavního menu. Toto grafické rozhraní se skládá z panelu s tlačítky, která umožňují uživateli vybrat si z různých možností (např. načtení modulu, nová hra). Pod tímto panelem je umístěno textové pole, v kterém je uvedeno jméno načteného modulu.



Obrázek 4.9: Návrh grafického rozhraní hlavního menu

- **aktivita menu pro ukládání hry**

  Na obrázku 4.10 je uveden návrh grafického rozhraní menu pro ukládání hry. Toto grafické rozhraní je z větší části vyplněno uloženými pozicemi, které může uživatel přepisovat. Ve spodní části je pak umístěno tlačítko pro vytvoření nové pozice. V případě, že by se uložené pozice nevešly na displej *Android* zařízení, je možné uložené pozice procházet pomocí posuvné lišty.



Obrázek 4.10: Návrh grafického rozhraní menu pro ukládání hry

- **aktivita menu pro načítání hry**

  Grafické rozhraní menu pro načítání hry je velmi podobné grafickému rozhraní menu pro ukládání hry (viz obrázek 4.10). Na rozdíl od grafického rozhraní pro ukládání hry není v grafickém rozhraní pro načítání hry umístěno tlačítko pro vytvoření nové pozice. V tomto grafickém rozhraní je možné pouze vybírat z uložených pozic. Stejně

jako u grafického rozhraní pro ukládání hry je možné uložené pozice procházet pomocí posuvné lišty.

- **aktivita menu pro načtení modulu**

  Grafické rozhraní menu pro načtení modulu je velmi podobné grafickému rozhraní menu pro ukládání hry (viz obrázek 4.10). Místo uložených pozic jsou zde zobrazeny jednotlivé moduly, které je možné načíst. V tomto grafickém rozhraní není, stejně jako v grafickém rozhraní pro načítání hry, umístěno tlačítko pro vytvoření nového modulu. Stejně jako u grafického rozhraní pro ukládání hry je možné uložené pozice procházet pomocí posuvné lišty.

Kapitola 5

Implementace

V první části této kapitoly je popsána implementace knihovny *AGLib*. Jsou zde popsány algoritmy, které jsou v této knihovně použity, a také struktura načtených dat. Nakonec je zde popsáno komunikační rozhraní.

V druhé části je uvedena implementace vývojového prostředí *AGCreator*. Tato část obsahuje popis grafického rozhraní a jeho funkcí. Dále jsou zde popsány zajímavé algoritmy implementované v tomto vývojovém prostředí.

V poslední části je popsána implementace grafického rozhraní interpretu *AGI*.

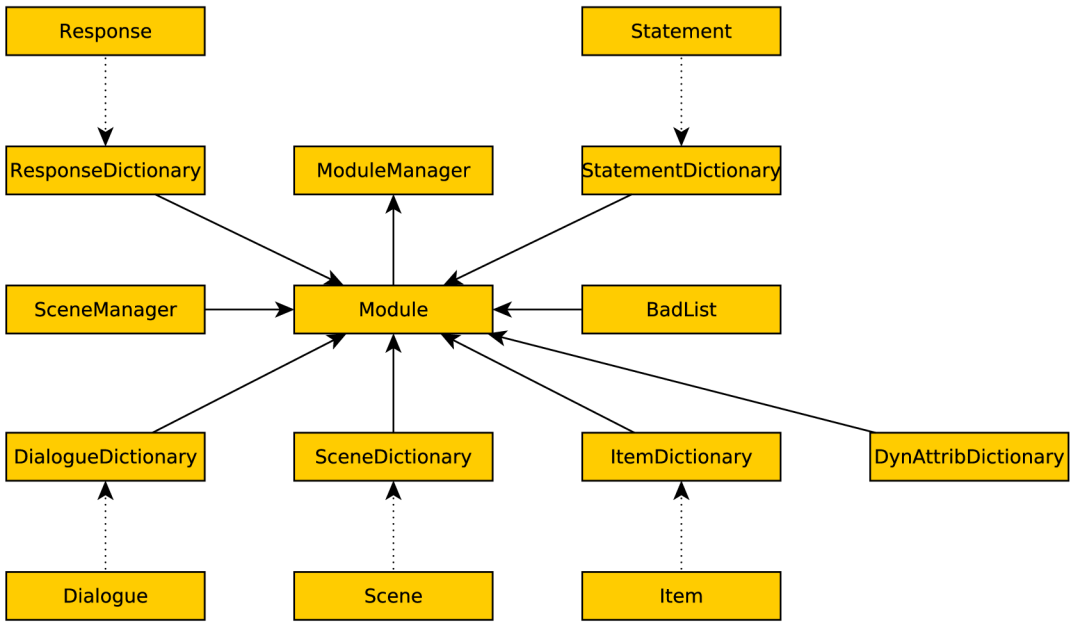
5.1 Implementace knihovny *AGLib*

Tato kapitola popisuje implementační detaily tvorby knihovny *AGLib*. Jsou zde popsány algoritmy používané v knihovně a je zde popsána struktura dat modulu. Nakonec je zde uvedeno komunikační rozhraní pro komunikaci s touto knihovnou.

5.1.1 Struktura dat

Data hry jsou uložena v modulu (viz kapitola 4.3.3). Tento modul je po načtení knihovnou *AGLib* uložen ve struktuře zobrazené na obrázku 5.1. O správu modulů se stará *ModuleManager*, který umožňuje práci s více moduly současně, přičemž pouze jeden modul může být aktivní.

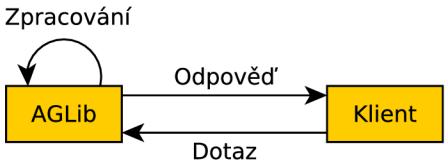
Každý modul (na obrázku označený *Module*) obsahuje několik *dictionary*, *BadList* a *SceneManager* (vztah je znázorněn plnou šipkou). Každý dictionary obsahuje jednotlivé prvky knihovny *AGLib* (viz kapitola 4.2), které jsou mapovány pomocí unikátních identifikátorů (vztah je znázorněn tečkovanou šipkou). Tento identifikátor je reprezentován celým číslem. Každý z prvků se potom odkazuje pouze identifikátorem do daného *dictionary* a zde je nalezen odpovídající prvek. Vyjímkou je *DynAttribDictionary*, který obsahuje hodnoty dynamických atributů, tyto atributy jsou mapovány pomocí identifikátoru objektu a poté ještě názvem atributu. *SceneManager* reprezentuje v modulu manažer scén (viz kapitola 4.2.2) a *BadList* reprezentuje množinu reakcí na nekorektní vstup od uživatele (viz kapitola 4.2.6).



Obrázek 5.1: Struktura dat v knihovně *AGLib*

5.1.2 Komunikační rozhraní

Aby bylo možno při interpretaci s knihovnou komunikovat, je potřeba implementovat komunikační rozhraní. Toto komunikační rozhraní funguje na principu dotaz-odpověď. Knihovna *AGLib* reprezentuje v komunikaci server a přijímá dotazy od klienta. Diagram komunikace je uveden na obrázku 5.2. Následující kroky popisují komunikační protokol použitý v knihovně:



Obrázek 5.2: Komunikační rozhraní knihovny *AGLib*

1. **Dotaz od interpretu**

   V prvním kroku přijmeme dotaz od interpretu. Dotaz obsahuje akci, která byla vykonána a identifikátor či identifikátory, s kterými byla akce provedena.

2. **Vyhodnocení dotazu**

   V druhém kroku je dotaz vyhodnocen a vytvořena odpověď. Tento algoritmus je popsán pseudokódem 5.1. Před samotným vyhodnocením akce je nejprve uložena současná scéna. Poté je pomocí konstrukce *switch* zjištěna provedená akce. Získaný identifikátor je poté otestován, zda je možné na něj akci použít. Pokud ano, uloží se do *response* text obsahující reakci na danou akci. V opačném případě je text vybrán z množiny reakcí na nekorektní vstup od uživatele. Trochu odlišná je situace při vyhodnocování akce *RESPONSE*. Tato akce je vyvolána pouze na dialogové scéně, a proto je nutné kontrolovat, zda je reakce popsaná identifikátorem východ do další scény či odkaz na další dialog. Po vyhodnocení akce jsou zkontrolovány podmíněné výrazy. Algoritmus popisující kontrolu podmíněných výrazů je uveden v kapitole 5.1.3.

3. **Odeslání odpovědi**

   V posledním kroku je odeslána odpověď vytvořená v předchozím kroku. Tato odpověď obsahuje text, který se má zobrazit, informaci o změně současné scény a informaci o změně současného dialogu, pokud je současnou scénou dialogová scéna.

```
Response checkAction(action, id1, id2){
  response = new Response();
  currentScene = SceneManager.getCurrentScene();

  switch(action){
    case USE:
      if(check(id1, USE))
        response.setText(getItem(id1).getText(USE));
      else
        response.setText(BadList.random());
      checkStatements(USE, id1, null);
      break;
    ...
    case RESPONSE:
      if(!exit(id1))
        response.setDialogue(getResponse(id1).getDialogue());
      else
        response.setSceneChanged(true);
      checkStatements(RESPONSE, id1, null);
      break;
  }

  if(currentScene != SceneManager.getCurrentScene())
    response.setSceneChanged(true);

  return response;
}
```

Pseudokód 5.1: Algoritmus pro vyhodnocení dotazu

5.1.3 Vyhodnocování podmíněných výrazů

Pomocí podmíněných výrazů je možné ovlivnit dynamické atributy za běhu hry. Vyhodnocení podmíněných výrazů se provádí při každé interakci s objektem. Algoritmus pro vyhodnocování podmíněných výrazů je popsán pseudokódem 5.2. Nejprve je vytvořen seznam identifikátorů objektů. Pro každý z těchto identifikátorů jsou poté získány podmíněné výrazy. Tyto podmíněné výrazy jsou vyhodnoceny pomocí metody *checkConditions*, pokud jsou vyhodnoceny *true* jsou vykonány příkazy (*commands*) spojené s těmito podmíněnými výrazy.

```
void checkStatements(action, id1, id2){
  ids[] = {id1, id2}
  for(id : ids){
    statements[] = getStatements(id);
    for(statement : statements){
      conditions[] = statement.getConditions();
      if(checkConditions(conditions)){
        commands[] = statement.getCommands();
        executeCommands(commands);
      }
    }
  }
}
```

Pseudokód 5.2: Algoritmus pro vyhodnocení podmíněných výrazů

5.1.4 Ukládání a načítání modulu

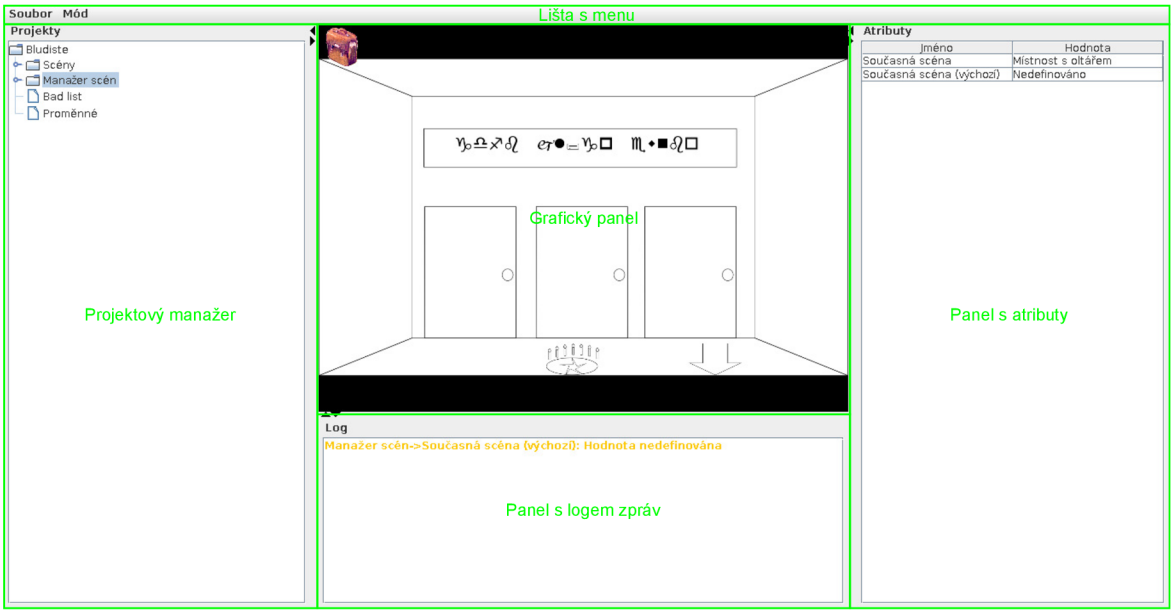
Modul (viz kapitola 4.3.3) obsahuje všechny potřebné soubory nutné k dané adventure hře. Zdrojové soubory jsou rozděleny podle typu prvku knihovny (viz kapitola 4.2.7). Pro parsování *XML* souborů byla použita knihovna *JDOM* [7].

Při načítání modulu se postupně načítají jednotlivé zdrojové soubory. Během načítání zdrojového souboru jsou zároveň načítána jeho jazyková data. **Lexikální analýza** je prováděna pomocí knihovny *JDOM*. Při **lexikální analýze** je zároveň prováděna **syntaktická analýza**. Data jsou při načítání ukládána do odpovídajících objektů. Pokud dojde při **syntaktické analýze** k nějaké chybě, je přerušeno načítání modulu a vyhozena vyjímka. Po načtení modulu je provedena kontrola identifikátorů. Pokud je v některém atributu uveden odkaz na neexistující identifikátor, je tento identifikátor nastaven na výchozí hodnotu.

Při ukládání se vytváří struktura *XML* souboru pomocí knihovny *JDOM*. Nejprve je vytvořena struktura složek popsaná v kapitole 4.3.3. Poté jsou ukládány jednotlivé prvky knihovny a vytvořen jazykový soubor.

5.2 Implementace vývojového prostředí *AGCreator*

V této kapitole budou popsány jednotlivé komponenty grafického rozhraní vývojového prostředí *AGCreator*. Při popisu jednotlivých komponent budou také popisovány zajímavé algoritmy s těmito komponentami spojené. Grafické rozhraní aplikace je zobrazeno na obrázku 5.3.



Obrázek 5.3: Grafické rozhraní aplikace *AGCreator*

5.2.1 Projektový manažer

Projektový manažer je v grafickém rozhraní (viz kapitola 4.3.2) reprezentován projektovým stromem. Kořenem projektového stromu je samotný projekt. Projekt reprezentuje ve vývojovém prostředí modul, který je v tomto vývojovém prostředí vyvíjen. Projekt je definován vlastním projektovým stromem, počátečními dynamickými atributy vyvíjeného modulu, modulem a složkou, kde je projekt vytvořen. V jeho větvích jsou pak uvedeny jednotlivé scény a objekty na těchto scénách, manažer scén a objekty umístěné v PME (viz kapitola 2.2.3), množina reakcí na nekorektní vstup od uživatele a seznam proměnných vytvořených v rámci tohoto projektu.

Projektový strom umožňuje určitou míru interakce. Při označení některého z objektů jsou jeho atributy načteny v panelu s atributy a tento objekt je ve stromu zvýrazněn. Při podržení pravého tlačítka myši je zobrazeno “pop up” menu, které nabízí možnosti editace stromu:

- **Smazat**

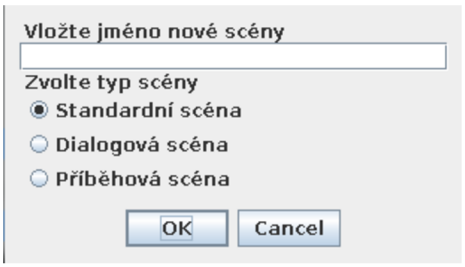
  Označený uzel je ze stromu smazán. Pokud uzel není koncovým uzlem, jsou smazány rekurzivně všechny objekty s ním související. Tato operace je přímo svázána s modulem načteným v knihovně a změna se tedy projeví přímo v načteném modulu, nejen v projektovém stromě.

- **Přejmenovat**

  Tato akce umožňuje přejmenovat označený uzel stromu. Jedná se vlastně o změnu atributu *name* u objektu reprezentující tento prvek. Tato změna se provádí pomocí dialogového okna.

- **Přidat scénu**

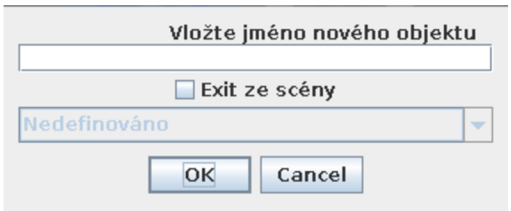
  Přidá novou scénu do *SceneDictionary*. Základní informace o této scéně jsou nastaveny pomocí dialogu pro vytvoření scény uvedeného na obrázku 5.4, kde je možné změnit typ scény a její název. Tato možnost je dostupná pouze z "pop up" menu vyvolaného nad seznamem scén.



Obrázek 5.4: Dialog pro přidání scény

- **Přidat objekt**

  Pomocí této akce je možné vytvořit nový objekt. Při zvolení této možnosti nad standardní scénou je zobrazeno dialogové okno uvedené na obrázku 5.5, kde je možné zvolit jméno objektu a také, zda se jedná o východ ze scény. V případě, že se jedná o východ ze scény, je nutné zvolit cílovou scénu. Možnost **Přidat objekt** je možné vybrat také při vyvolání "pop up" menu u manažera scén. V tomto případě je možné nastavit pouze jméno objektu. Objekt je přidán do *ItemDictionary* a do atributu *items* dané standardní scény či scénového manažera.



Obrázek 5.5: Dialog pro přidání objektu

- **Přidat proměnnou**

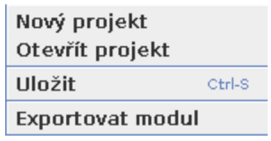
  Tato možnost je k dispozici pouze u seznamu proměnných. Po výběru této možnosti je zobrazeno dialogové menu, kde je možné zvolit název proměnné.

- **Nastavit jako současnou scénu**

  Při výběru této možnosti je nastavena označená scéna jako atribut *currentScene* ve scénovém manažeru a je vykreslena v grafickém okně. Tímto způsobem je možné se přepnout kdykoli na kteroukoli scénu.

5.2.2 Lišta s menu

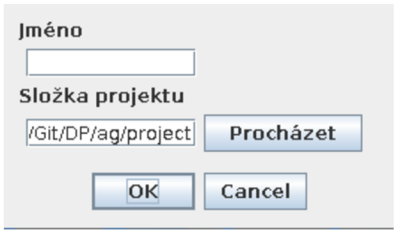
V této liště jsou umístěny hlavní nabídky celé aplikace. Celkově se jedná o dvě menu. První menu je uvedeno na obrázku 5.6. Druhé menu obsahuje pouze možnost přepnout mezi módy aplikace. Po výběru módu je nastavena proměnná *AGCMode* a podle této proměnné se poté vybírají interakce v grafickém panelu. V prvním menu jsou uvedeny následující nabídky:



Obrázek 5.6: Menu soubor

- **Nový projekt**

  Při vytvoření nového projektu se nejprve objeví dialog uvedený na obrázku 5.7. Zde je nutné vybrat složku v níž bude složka projektu vytvořena a název projektu. Poté je ve vývojovém prostředí vytvořen nový objekt projektu, nastaveny počáteční dynamické atributy a v projektovém manažeru je zobrazen strom tohoto projektu.



Obrázek 5.7: Dialog pro vytvoření projektu

- **Otevřít projekt**

  Při výběru této nabídky je otevřen standardní dialog pro otevření souboru, zde je zapotřebí vybrat složku obsahující projekt. Načítání je prováděno pomocí knihovny *AGLib*. Nejprve jsou načteny data modulu a poté jsou načteny současné hodnoty dynamických atributů a hodnoty v modulu jsou uloženy jako výchozí hodnoty.

- **Uložit**

  Při ukládání projektu jsou nejprve uloženy současné dynamické atributy jako současný průběh hry (viz kapitola 4.3.3). Poté jsou uložena data hry ve formátu modulu. Všechny metody využívané pro ukládání obsahuje knihovna *AGLib*.

- **Exportovat modul**

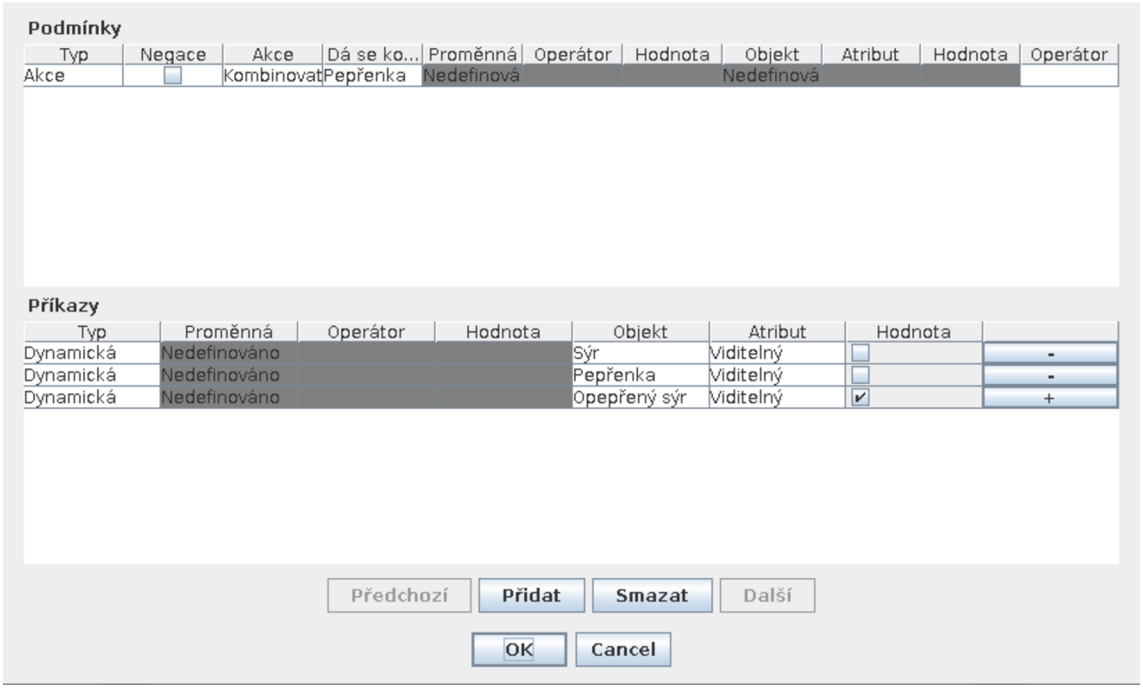
  Při exportování modulu je otevřen nejprve dialog pro výběr složky, kde má být modul uložen. Tento modul je poté pomocí metod v knihovně *AGLib* uložen do vybrané složky.

5.2.3 Panel s logem zpráv

V tomto panelu se uživateli zobrazují informace o nedefinovaných atributech jednotlivých prvků v projektu. Je zde uvedena cesta v projektovém stromě, kterou je možné se dostat k nedefinovanému atributu. Po označení řádku v logu, je označen odpovídající prvek v projektovém stromě. Log je automaticky obnovován při každé změně provedené v projektu.

5.2.4 Panel s atributy

Tento panel slouží k editaci atributů označeného uzlu projektového stromu. Použitý editor se mění podle datového typu daného prvku. Jedná-li se o datový typ *String* je k editaci použit objekt *TextArea*, v případě *boolean* hodnoty se jedná o *CheckBox* a pokud se jedná o *float* či *int* je použit *TextField* s validátorem, který neumožní vložit nekorektní hodnotu. Pokud je hodnota vybírána ze seznamu hodnot, je jako editor použit *ComboBox*. Speciální editor je použit pro editaci podmíněných výrazů a dialogů.

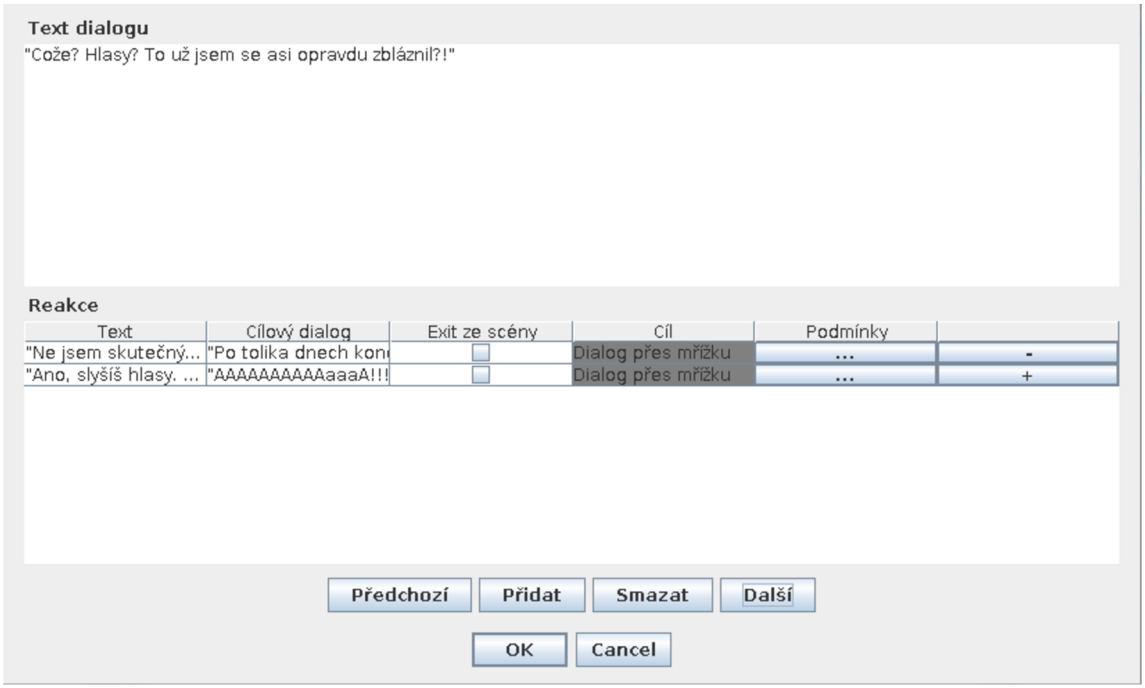


Obrázek 5.8: Dialog pro editaci podmíněných výrazů

Při editaci podmíněných výrazů se objeví dialog uvedený na obrázku 5.8. Ve vrchní části dialogu je možné vytvořit podmíněné výrazy, které je možné spojovat pomocí logických spojek *AND* a *OR*. Pokud je vybrána spojka přidá se do tabulky automaticky nový řádek. Ve střední části dialogu je možné vytvořit příkazy, které budou vykonány pokud budou podmíněné výrazy splněny (viz kapitola 5.1.3). Počet vytvořených příkazů je možné libovolně

měnit pomocí tlačítka v posledním sloupci. Dialog pro editaci podmíněných výrazů dále umožňuje vytvářet více podmíněných výrazů s vlastními příkazy. K tomu slouží tlačítka ve spodní části dialogu, kde je možné vytvářet, mazat podmíněné výrazy a také jimi procházet.

Pro editaci dialogů v dialogových scénách je použit dialog uvedený na obrázku 5.9. Ve vrchní části dialogu je možné nastavit text samotného dialogu. Ve střední části dialogu je pak možno nastavit jednotlivé reakce na daný dialog. U každé reakce je možné také nastavit podmíněné výrazy. Počet reakcí je možné libovolně měnit. Ve spodní části dialogu jsou vytvořena tlačítka, kterými je možné přidávat mazat a procházet jednotlivé dialogy.



Obrázek 5.9: Dialog pro editaci dialogů v dialogových scénách

5.2.5 Grafický panel

V grafickém panelu je vykresleno grafické ztvárnění současné scény. Ke grafickému panelu je připojen *MouseListener* jehož funkce se mění podle nastavení proměnné *AGCMode*, kde je uložen režim vývojového prostředí *AGCreator*. Obsah grafického panelu se mění podle typu současné scény. Jedná-li se o standardní scénu, je ve spodní části umístěn text pro uživatele, kde je zobrazeno jméno předmětu nebo text zpětné vazby na provedenou akci. V pozadí je umístěno grafické ztvárnění scény, kde je zobrazen obrázek samotné scény a obrázky objektů na této scéně umístěných. Ve vrchní části grafického panelu je umístěna PME (přenositelná množina elementů, objekty, které má uživatel u sebe v tzv. “inventáři”). U dialogové scény je v pozadí zobrazen obrázek scény a ve spodní části panelu je umístěn text dialogu a reakce na tento text. U příběhové scény je rozmístění podobné jako u dialogové scény, jen místo dialogu a reakcí je zde umístěn text příběhové scény.

Během vývojového režimu je možné v grafickém panelu označit existující objekty a pomocí “pop up” menu vytvářet nové objekty nebo přiřadit nový obrázek již existujícím objektům. Tyto obrázky jsou uloženy do herní cache, která k jednotlivým obrázkům vytvoří

zvýrazňovací mapu. Tato mapa je vytvářena pomocí algoritmu popsaného pseudokódem 5.3 a je zobrazena při najetí myši na daný objekt. Algoritmus nejprve prochází pixely po řádcích a pokud vedle sebe nalezne průhledný a neprůhledný pixel je současný pixel zvýrazněn. Tento postup se provede ještě jednou, ale pixely jsou procházeny po sloupcích.

```
Bitmap createHighlightMap(bitmap){
  highlight = new Bitmap();
  for(x in bitmap.getWidth()){
    pixelTransparent = false;
    previousTransparent = true;
    for(y in bitmap.getHeight()){
      if(pixel.isTransparent())
        pixelTransparent = true;
      else
        pixelTransparent = false;
      if(pixelTransparent != previousTransparent)
        highlight.highlightPixel(x,y);
      previousTransparent = pixelTransparent;
    }
  }

  for(y in bitmap.getheight()){
    pixelTransparent = false;
    previousTransparent = true;
    for(x in bitmap.getWidth()){
      if(pixel.isTransparent())
        pixelTransparent = true;
      else
        pixelTransparent = false;
      if(pixelTransparent != previousTransparent)
        highlight.highlightPixel(x,y);
      previousTransparent = pixelTransparent;
    }
  }

  return highlight;
}
```

Pseudokód 5.3: Algoritmus pro vytváření zvýrazňovací mapy

V interpretačním režimu aplikace je možné v grafickém panelu provádět akce s jednotlivými objekty na scéně. Tyto akce jsou prováděny pomocí levého tlačítka myši. Pomocí pravého tlačítka myši je možné mezi akcemi přepínat. Při provedení akce na objekt, je akce a identifikátor objektu odeslán knihovně *AGLib*. Po obdržení odpovědi je provedena změna scény nebo zobrazen text odpovědi. Poté je nutné zkontrolovat objekty na současné scéně, zda se nezměnili jejich atributy.

Herní render

Herní render obsahuje současný stav grafického zobrazení scény a také obsahuje několik funkcí, které umožňují s tímto obsahem lepé pracovat. Obrázky obsažené v herním renderu jsou rozděleny do vrstev, při vykreslování scény se pak vykreslují po vrstvách od nejnižší vrstvy. Vykreslování scény probíhá 25-krát za sekundu. Veškeré změny provedené v herním renderu se tedy okamžitě projeví. Každý obrázek má také uloženu informaci o své pozici na scéně a velikosti. Herní render se stará o přizpůsobení této velikosti velikosti displeje. Algoritmus použitý pro změnu velikosti je popsán pseudokódem 5.4 a funguje následovně:

1. **Změna velikosti obrázku scény**

   V tomto kroku algoritmus přizpůsobí velikost scény velikosti displeje, na který se má scéna přizpůsobit. Rozměry scény zachovávají poměr stran, aby nedošlo k deformaci obrázku. Nové rozměry obrázku se uloží do proměnných.

2. **Změna velikosti obrázků objektů na scéně**

   V druhém kroku je změněna velikost obrázků objektů. Tato velikost je přizpůsobena současné velikosti obrázku scény, protože velikost obrázku objektu je uložena jako procentuální velikost ku obrázku scény.

3. **Změna pozice objektů na scéně**

   V posledním kroku se přepočítává umístění obrázků objektů na scéně. K tomuto výpočtu je opět použita velikost obrázku scény, protože pozice objektů na scéně je uložena jako poměr ku velikosti obrázku scény.

```
void resizeScene(width, height)
{
  sceneImage.setSize(width, height);
  sceneWidth = sceneImage.getWidth();
  sceneHeight = sceneImage.getHeight();
  for(objectImage : renderImages){
    item = ItemDictionary.getItem(objectImage.getOwner());
    objectImage.setWidth(sceneWidth * item.getWidth());
    objectImage.setHeight(sceneHeight * item.getHeight());
    objectImage.setPosition(sceneWidth * item.getXPosition(),
       sceneHeight * item.getYPosition);
  }
}
```

Pseudokód 5.4: Algoritmus pro změnu velikosti scény

Dalším důležitým algoritmem je vyhledávání objektu podle souřadnic. Tento algoritmus postupně prochází všechny obrázky od nejvyšší vrstvy a hledá v některém z těchto obrázků neprůhledný pixel na daných souřadnicích. Pokud je pixel průhledný, nepočítá se objekt jako nalezený na daných souřadnicích. Vyhledávání pixelu v jednom obrázku je popsáno pseudokódem 5.5 a probíhá v následujících krocích:

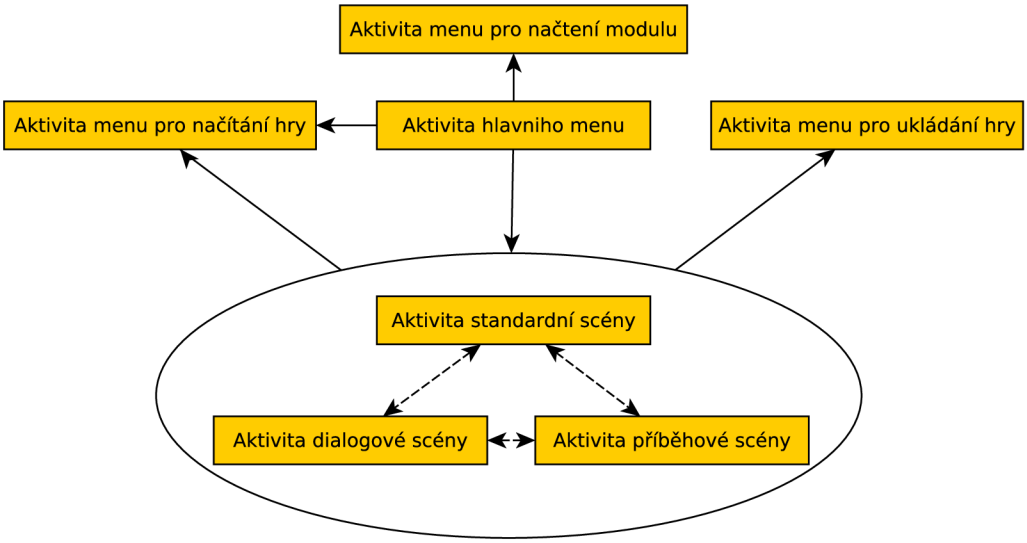
1. Odečtení pozice obrázku od souřadnic
2. Kontrola, zda souřadnice nejsou mimo plochu obrázku
3. Získání pixelu na daných souřadnicích
4. Kontrola průhlednosti daného pixelu

```
boolean isPointInimage(x, y, image)
{
  imageX = image.getX() - x;
  imageY = image.getY() - y;

  if(imageX < 0 || imageX > image.getWidth())
    return false;

  if(imageY < 0 || imageY > image.getHeight())
    return false;

  pixel = image.getPixel(imageX, imageY);
  if(!pixel.isTransparent())
    return true;
  return false;
}
```

Pseudokód 5.5: Algoritmus pro vyhledávání pixelu v obrázku

5.3 Implementace interpretu *AGI*

Interpret *AGI* slouží k interpretaci dat uložených ve formě modulu (viz kapitola 4.3.3) na platformě *Android*. Interpret *AGI* je složen z několika komponent. Jedná se o jednotlivé aktivity, kde každá z těchto aktivit má vlastní grafické rozhraní. Propojení jednotlivých aktivit je uvedeno na obrázku 5.10. Vztah znázorněný plnou šipkou znamená, že rodičovská aktivita není po spuštění aktivity ukončena a je možné se k ní navrátit pomocí tlačítka zpět. Vztah znázorněný přerušovanou šipkou označuje, že rodičovská aktivita je ukončena a není možné se k ní vrátit. Aktivity umístěné v oválu mají stejné vztahy znázorněné šipkami vedoucími do/od oválu.



Obrázek 5.10: Mapa aktivit aplikace *AGI*

5.3.1 Aktivita hlavního menu

Tato aktivita se uživateli objeví po zapnutí aplikace. Umožňuje uživateli používat základní funkce *AGI*. Uživatel v této aktivitě má možnost interakce pomocí tlačítek. Je zde také vytvořeno textové pole, kde je uveden název momentálně načteného modulu. Grafické rozhraní aktivity hlavního menu je zobrazeno na obrázku 5.11.

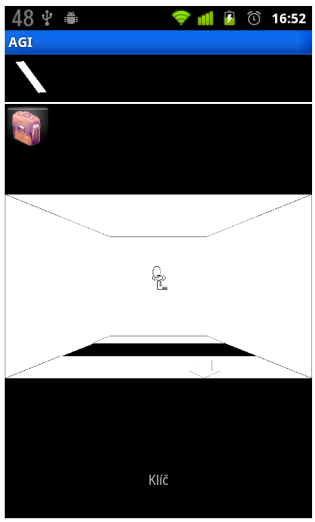
Obrázek 5.11: Grafické rozhraní aktivity hlavního menu

5.3.2 Aktivita standardní scény

Umožňuje uživateli interagovat s objekty na této scéně a v PME (viz kapitola 2.2.3). Uživateli poskytuje zpětnou vazbu na jeho akce pomocí textového pole umístěného ve spodní části displeje. V horním rohu displeje je umístěno tlačítko pro zobrazení PME. Po kliknutí na toto tlačítko je zobrazena lišta s obrázky objektů umístěných v PME. Grafické rozhraní této aktivity je uvedeno na obrázku 5.12.

K zobrazení grafického ztvárnění scény je použit stejný herní render jako v případě vývojového prostředí *AGCreator* (viz kapitola 5.2.5).

Objekty zobrazené v této scéně je možné zvýraznit kliknutím na objekt. Zvýraznění je provedeno pomocí zvýraznění hran objektu žlutou barvou a zobrazením jména označeného objektu v textovém poli. Pokud je některý objekt označen, je možné podržením vyvolat kontextové menu, které nabízí akce, které je možné s tímto objektem provést. Výběr akcí se liší podle umístění objektu na scéně či v PME.



Obrázek 5.12: Grafické rozhraní aktivity standardní scény

5.3.3 Aktivita dialogové scény

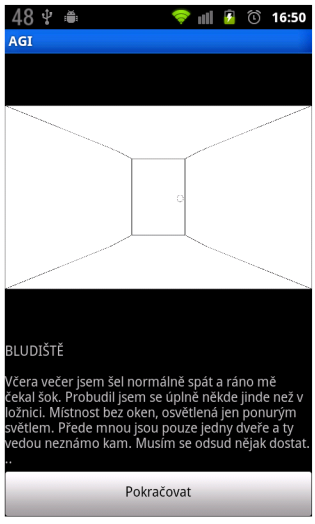
Umožňuje uživateli procházet dialogovým stromem a zároveň mu pomocí zobrazení nového dialogu poskytuje zpětnou vazbu. Grafické rozhraní této aktivity je zobrazeno na obrázku 5.13.

Při spuštění aktivity je zobrazen obrázek scény a počáteční dialog. Dialog se skládá z textu a z možných reakcí na tento text. Při výběru reakce uživatelem je tato reakce vyhodnocena pomocí knihovny *AGLib* (viz kapitola 5.1.2). Odpovědí na tuto reakci může být změna dialogu či změna scény. Při změně dialogu je nový dialog zobrazen na grafickém rozhraní aktivity. Pokud dojde ke změně scény, je současná aktivita ukončena a načtena nová odpovídající nové scéně.

Obrázek 5.13: Grafické rozhraní aktivity dialogové scény

5.3.4 Aktivita příběhové scény

Uživatel zde vidí text týkající se příběhu a tlačítko *Pokračovat*. Pokud je u příběhové scény nastaven atribut *exitTarget* dojde po kliknutí na tlačítko *Pokračovat* k ukončení současné aktivity a startu aktivity odpovídající typu cílové scény. V opačném případě dojde opět k ukončení aktivity a k návratu do aktivity hlavního menu. Grafické rozhraní aktivity příběhové scény je zobrazeno na obrázku 5.14.



Obrázek 5.14: Grafické rozhraní aktivity příběhové scény

Kapitola 6

Ukázková hra

V rámci této práce byla ve vývojovém prostředí *AGCreator* vytvořena ukázková hra, která dostala název *Bludiště*. Uživatel se v této hře dostane do role člověka uvězněného v podzemním bludišti a jeho cílem je nalézt východ z tohoto bludiště. Hra je koncipována tak, aby bylo těžké najít cestu dál, spousta lokací je proto podobná a uživatel tím získává pocit, jako by byl opravdu v bludišti. Ve hře je také zakomponováno několik logických hádanek, které musí uživatel vyřešit, aby ve hře pokročil dále. Na této hře jsou ukázány všechny typy scén uvedené v kapitole 4.2.1.

6.1 Popis scén v adventure hře *Bludiště*

Ve hře se celkově vyskytuje 32 scén, z nichž většina slouží jako křižovatky, aby hra co nejvíce připomínala bludiště. Na obrázku 6.1 je uvedena mapa scén, která popisuje propojení jednotlivých scén mezi sebou. V rámci této hry existuje spousta zajímavých scén, dále budou uvedeny jen některé z nich:

- **Scéna s klíčem** (5)

  Tato scéna obsahuje objekt *Klíč*, který je potřeba, aby uživatel pokročil ve hře dále. Na tento objekt však není možné použít akci zvednout, dokud není provedena kombinace objektu *Díra* s objektem *Deska*. Při pokusu použít akci zvednout na objekt *Klíč* bez vykonání této kombinace je uživatel přesunut na scénu *Smrt* (32), která je považována za špatný konec hry.

- **Dialogová scéna přes mřížku** (7)

  Tato scéna je jediná dialogová scéna ve hře *Bludiště* a umožňuje uživateli získat dodatečné informace k postupu ve hře. Uživatel si zde může promluvit s neznámým NPC (Non-playable character), které je uzavřeno v podobném bludišti na druhé straně mřížky. NPC od uživatele požaduje objekt *Klíč*, aby mohlo otevřít celu, v níž je zavřeno.

- **Scéna se zavřenými dveřmi** (15)

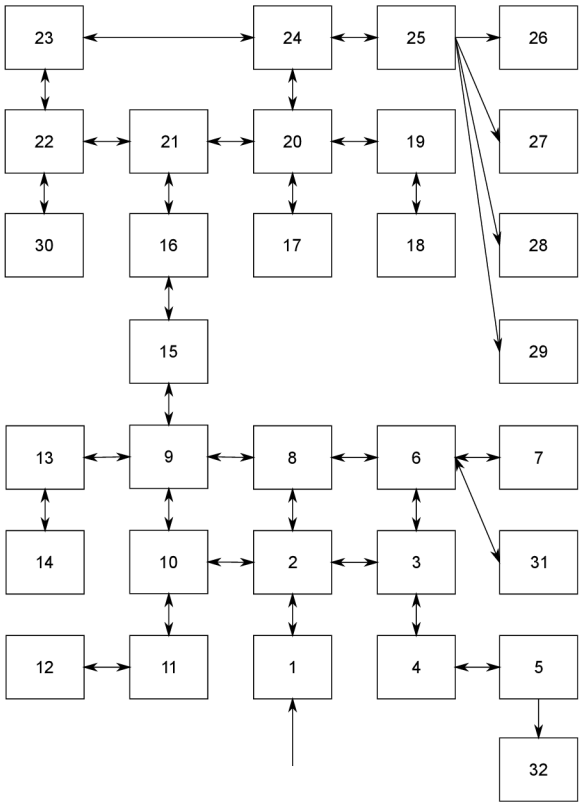
  Na této scéně je umístěn objekt *Zamčené dveře*. S tímto objektem není možné provádět žádnou interakci, která by vedla k postupu ve hře. Uživatel musí tedy vymyslet způsob, jak tyto dveře otevřít. Jedná se o jednu z logických hádanek, které musí uživatel vyřešit.

- **Scéna s myší dírou** (22)

  Úkolem uživatele na této scéně je získat objekt *Mrtvá myš*, tento objekt však není na scéně viditelný. Aby byl tento objekt zviditelněn je nutné, aby uživatel nejprve sestrojil objekt *Past na myš*. Tento objekt se získá kombinací několika objektů, které jsou umístěny v různých scénách.

- **Scéna s oltářem** (25)

  Na této scéně je umístěno několik objektů, jedná se o *Oltář*, *První dveře*, *Druhé dveře* a *Třetí dveře*. Při použití akce *Jít* na některé z dveří je hráč přesunut na scénu, která označuje špatný konec hry. Uživatel tedy musí přijít na způsob, jak se dostat přes tuto místnost, nápovědu mu poskytuje objekt *Nápis nad dveřmi* umístěný na této scéně a objekt *Papír s textem*, který je možné zvednout na jiné scéně.



Obrázek 6.1: Mapa scén hry *Bludiště*

6.2 Logické hádanky v adventure hře *Bludiště*

Adventure hra Bludiště obsahuje množství logických hádanek, které může uživatel vyřešit. V této kapitole budou uvedeny některé z těchto hádanek a jejich řešení. Logické hádanky budou popsány ve formě tabulek případu užití, ty budou obsahovat následující řádky:

- **Název** udává název dané logické hádanky.
- **Problém** popisuje problém, který je nutné v rámci hádanky vyřešit.
- **Cíl** udává, čeho by mělo být dosaženo.
- **Předpoklady**, které musí být splněny před řešením problému.
- **Postup řešení** popisuje, jakým způsobem je možné hádanku vyřešit.

Název	Odemčení dveří
Problém	Přejít na scénu za objektem *Zamčené dveře*
Cíl	Změnit viditelnost objektu *Odemčené dveře*
Předpoklady	Žádné
Postup řešení	1. Přejít na scénu 12. 2. Použit akci *Zvednout* na objekt *Deska*. 3. Přejít na scénu 5. 4. Použít akci *Kombinovat* na objekty *Deska* a *Díra*. 5. Použít akci *Zvednout* na objekt *Klíč*. 6. Přejít na scénu 6. 7. Použít akci *Kombinovat* na objekty *Klíč* a *Mřížka*.

Tabulka 6.1: Případ užití pro logickou hádanku *Odemčení dveří*

Název	Zapálení pochodně
Problém	Nelze přejít na následující scény bez zapálení objektu *Pochodeň*
Cíl	Změnit viditelnost objektu *Hořící pochodeň*
Předpoklady	Vyřešení hádanky *Odemčení dveří*
Postup řešení	1. Přejít na scénu 14. 2. Použít akci *Použít* na objekt *Skříň*. 3. Přejít na scénu 21. 4. Použít akci *Kombinovat* na objekty *Pochodeň* a *Sirky*.

Tabulka 6.2: Případ užití pro logickou hádanku *Zapálení pochodně*

Název	Získat mrtvou myš
Problém	Na objektu *Oltář* je nutné něco obětovat
Cíl	Změnit viditelnost objektu *Mrtvá myš*
Předpoklady	Vyřešení hádanek *Odemčení dveří* a *Zapálení pochodně*
Postup řešení	1. Přejít na scénu 19. 2. Použít akci *Zvednout* na objekt *Kámen*. 3. Přejít na scénu 17. 4. Použít akci *Použít* na objekt *Police*. 5. Použít akci *Kombinovat* na objekty *Sýr* a *Pepřenka*. 4. Přejít na scénu 22. 4. Použít akci *Kombinovat* na objekty *Myší díra* a *Kámen*. 5. Použít akci *Kombinovat* na objekty *Opepřený sýr* a *Kámen*.

Tabulka 6.3: Případ užití pro logickou hádanku *Získat mrtvou myš*

Kapitola 7

Testování

Tato kapitola je věnována testování aplikace *AGCreator* a ukázkového projektu vytvořeného v tomto prostředí. Celá kapitola testování je rozdělena na dvě části. V první části bude popsáno testování vývojového prostředí *AGCreator*. V druhé části pak bude popsáno testování ukázkového projektu vytvořeného ve vývojovém prostředí *AGCreator*. Toto testování je prováděno na *Android* aplikaci *AGI*.

7.1 Testování vývojového prostředí *AGCreator*

Testování vývojového prostředí bylo prováděno na skupince tří uživatelů uvedených v tabulce 7.1. Tito uživatelé byli nejprve seznámeni s ovládáním *AGCreatoru* a poté dostali zadání úkolu. Uživatelé dostali k dispozici obrázky, které byly použity při tvorbě ukázkového projektu. Všechny testy byly prováděny na operačním systému *Linux*.

Zadáním úkolu bylo vytvořit tři scény:

1. Standardní scéna, která obsahuje minimálně dva objekty. Jeden objekt, který je možné kombinovat s objektem v PME (přenosná množina elementů, tedy objekty, které může uživatel během hry přenášet u sebe) a druhý objekt, který funguje jako východ do další scény.

2. Dialogovou scénu obsahující nejméně dva dialogy.

3. Příběhovou scénu.

V rámci úkolu je nutné všechny tyto scény propojit, aby mezi nimi bylo možné přecházet. Další součástí zadání bylo vytvořit jeden objekt v PME, který bude možné kombinovat s objektem ve standardní scéně.

Uživatel	**Věk**	**Programátorské dovednosti**	**Počítačové znalosti**
Uživatel A	25	Žádné	Základní
Uživatel B	24	Pokročilé C++	Pokročilé
Uživatel C	40	Základní C	Mírně pokročilé

Tabulka 7.1: Seznam uživatelů

7.1.1 Test č.1

Testování prováděl: Uživatel A
Čas testování: 16:15 6.5.2013
Průběh testování:

Uživatel nejprve vytvořil příběhovou scénu. Při vytváření této scény nenarazil uživatel na žádné problémy. Jako druhou věc se uživatel pokusil vytvořit objekt v PME. Při vytváření tohoto objektu narazil nejprve na problém, jak vytvořit objekt přímo v PME. Po pár minutách se však uživateli podařilo tento objekt vytvořit. Při vytváření standardní scény pracoval uživatel rychle bez jediného problému. Nakonec uživatel vytvářel dialogovou scénu, kde měl problém pochopit, jak má jednotlivé dialogy vytvořit a propojit. Vytvoření dialogu mu tedy zabralo asi 3 minuty. Jako poslední krok uživatel propojil jednotlivé scény.
Doba testování: 30 min
Závěr:

Uživatel A pracoval díky vývojovému prostředí efektivně a během 30 minut měl zadání splněné. Při plnění zadání došlo k zdržení, které bylo zapříčiněno neznalostí vývojového prostředí *AGCreator*.

7.1.2 Test č.2

Testování prováděl: Uživatel B
Čas testování: 12:57 9.5.2013
Průběh testování:

Uživatel měl nejprve několik otázek k některým prvkům používaným v knihovně *AGLib*. Poté se mu povedlo vytvořit standardní scénu a dva objekty na této scéně bez problémů. Stejně bez problémů probíhalo vytvoření objektu v PME. První problém vznikl při vytváření dialogové scény, kde uživateli nějakou dobu trvalo vytvořit dialogy. U vytváření příběhové scény nedošlo k žádné komplikaci. Nakonec uživatel tyto scény propojil, aby se mezi nimi dalo procházet.
Doba testování: 15 min
Závěr:

Uživateli B umožnilo vývojové prostředí *AGCreator* rychlé a efektivní vytváření jednoduché struktury adventure hry. Uživatel B neměl s používáním aplikace žádné problémy.

7.1.3 Test č.3

Testování prováděl: Uživatel C
Čas testování: 18:15 7.5.2013
Průběh testování:

U uživatele C probíhalo testování pomaleji než v ostatních testech. Toto bylo zapříčiněno neznalostí adventure her, takže uživatel měl každou chvíli otázku, co znamenají jednotlivé atributy objektů, a jak se projeví ve výsledné hře. Těmito otázkami se uživatelova práce zpomalila, ale po obdržení odpovědi již uživatel pracoval rychle. Zadání nakonec splněno nebylo, protože uživatel byl nucen odejít. Ze zadání nebylo splněno vytvoření příběhové scény.
Doba testování: 40 min
Závěr:

Práce uživatele C nebyla efektivní ani rychlá, toto bylo způsobeno především neznalostí problematiky adventure her. Kdyby měl uživatel větší znalosti této problematiky, byla by

jeho práce rychlejší a efektivnější. Vzhledem k neznalosti problematiky uživatele C se však jedná o úspěch, že byl schopen ve vývojovém prostředí *AGCreator* vytvořit aspoň část zadání.

7.1.4 Dotazník

Po skončení testu byli uživatelé vyzváni k vyplnění dotazníku, kde bylo hodnoceno na stupnici od 1 (nejlepší) do 5 (nejhorší) kvalita jednotlivých aspektů aplikace. Výsledky dotazníku jsou uvedeny v tabulce 7.2. Aplikace byla hodnocena v následujících aspektech:

- Složitost - jak složité bylo pracovat s aplikací
- Intuitivnost - jak složité bylo nalézt požadované ovládací prvky
- Pohodlnost - jak pohodlně se s aplikací pracovalo
- Přehlednost - zda uživatel měl všechny informace po ruce
- Interaktivita - jak interaktivní je aplikace

Aspekt	**Hodnocení**
Složitost	2,3
Intuitivnost	2,5
Pohodlnost	2,7
Přehlednost	1,8
Interaktivita	1,2

Tabulka 7.2: Výsledky dotazníku

7.2 Testování ukázkové hry v aplikaci *AGI*

Testování ukázkové hry vytvořené ve vývojovém prostředí *AGCreator* bylo testováno na skupince tří uživatelů. Ani jeden z těchto uživatelů se nepodílel na vývoji aplikace. Tito uživatelé dostali k dispozici zařízení s aplikací *AGI* a modul obsahující data se hrou. Průběh ukázkové hry byl zaznamenáván pomocí logů, z kterých byli poté získány data. Každý z uživatelů měl za úkol dostat se ke konci hry nebo hrát, dokud je hra bavila. V rámci jednotlivých testování bude popisována cesta daného uživatele, přičemž budou uvedeny jen významné scény. U každého testování je také obrázkem znázorněna mapa jednotlivých průchodů hrou, při zakreslení těchto průchodů bylo vynecháno zbytečné bloudění, aby mapa zůstala přehledná. Každý průchod hrou je barevně odlišen, první průchod je červenou barvou, druhý průchod je modrou barvou a třetí průchod zelenou. V každém testování je také uvedena tabulka, kde jsou uvedena statistická data získaná z logů daného testování.

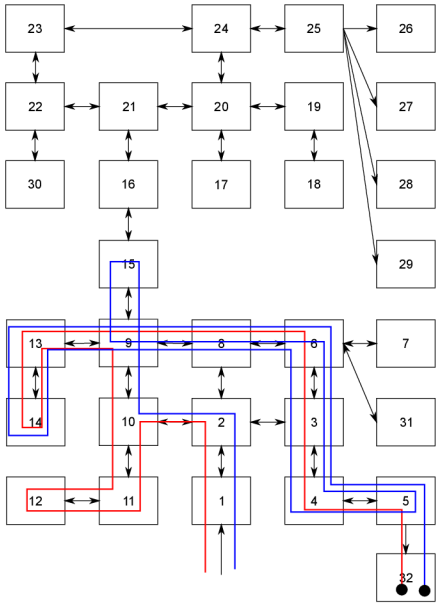
7.2.1 Testování uživatelem č.1

První uživatel nemá příliš zkušeností s hraním adventure her a jeho průchod hrou byl tedy pomalejší. Problémy mu činilo také dotykové ovládání.

Jeho cesta nejprve vedla na scénu 12, kde ale nezjistil, že objekt *Deska* je možné vzít. Dále pokračoval na scénu 14, kde získal objekty *Nůž* a *Sirky*. Nakonec se dostal na scénu 5, kde ale udělal chybu při pokusu získat objekt *Klíč*. Dostal se tak, ke špatnému konci, tedy scéně 32.

Při druhém průchodu uživatel mezi scénami zabloudil a po delší době se dostal na scénu 15, kde našel *Zamčené dveře*. Poté se vydal ke scéně 5, kde již neopakoval svoji chybu a raději odešel na scénu 14, kde opět získal *Nůž* a *Sirky*. Pak se vrátil opět na scénu 5 a pokusil se o jinou interakci, ta ale opět vedla ke scéně 32 a tedy špatnému konci.

Průchody hrou v tomto testování jsou znázorněny na obrázku 7.1. Statistická data získána z testování uživatelem č.1 jsou uvedena v tabulce 7.3.



Obrázek 7.1: Průchod hrou uživatelem č.1

Jméno	**Hodnota**
Doba hraní	39:58
Scéna, na které uživatel strávil nejvíce času	15
Čas	1:48
Nejčastěji používaná akce	Jít
Počet použití	141
Objekt, s kterým byla nejčastěji provedena akce	Prázdná skříň
Scéna, na které se objekt vyskytuje	14
Počet provedených akcí	26
Počet navštívených scén	128

Tabulka 7.3: Statistická data uživatele č.1

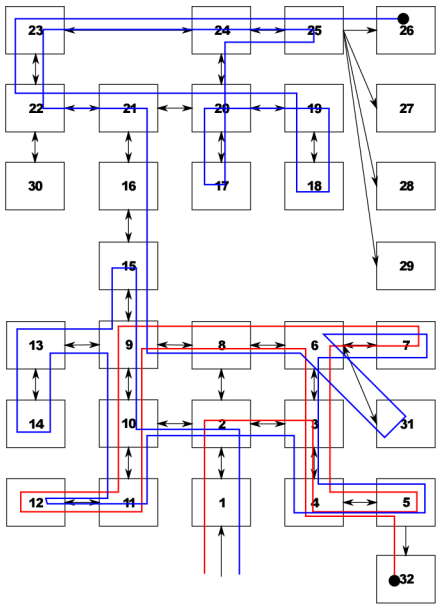
7.2.2 Testování uživatelem č.2

Druhý uživatel je zvyklý na každodenní používání platformy *Android*, takže neměl žádné problémy s ovládáním hry. Také neměl žádné problémy s řešením logických problémů, které hra nabízela a dohrál hru docela rychle.

Při prvním průchodu uživatel nejdříve zamířil na scénu 5, kde zjistil, že nemůže získat objekt *Klíč*. Po tomto zjištění se přesunul na scénu 6, kde se pomocí interakce s objektem *Mřížka* dostal na dialogovou scénu 7. Z dialogové scény se přesunul na scénu 12, kde získal objekt *Deska*. S tímto novým objektem zamířil zpátky na scénu 5, kde ale provedl špatnou interakci a dostal se tak ke špatnému konci, tedy na scénu 32.

Při druhém průchodu zamířil nejprve na scénu 15, kde našel objekt *Zamčené dveře*. Zde zjistil, že nemá zatím žádnou možnost pokračování a přesunul se tedy na scénu 14, kde získal objekty *Nůž* a *Sirky*. Z této scény se přesunul na scénu 12, kde získal objekt *Deska*. S tímto objektem v PME se přesunul na scénu 5, kde díky kombinaci objektu *Deska* s objektem *Díra* získal *Klíč*. Uživatel se poté přesunul na scénu 6, kde kombinací objektů *Klíč* a *Mřížka* došlo ke změně viditelnosti objektu *Odemčené dveře* na scéně 15. Uživatel dále pokračoval na scénu 22, kde nalezl objekt *Myší díra*, s kterým zatím neměl možnost interakce. Z této scény se přesunul na scénu 17, kde získal objekty *Sýr* a *Pepřenka* a kombinací těchto objektů získal nový objekt *Opepřený sýr*. Poté se vydal na scénu 19, kde získal *Kámen*. S tímto objektem se poté přesunul na scénu 22, kde kombinací objektů *Kámen* a *Opepřený sýr* získal objekt *Mrtvá myš*. Nakonec se přesunul na scénu 25, kde pomocí kombinace objektů *Oltář* a *Mrtvá myš* získal objekt *Myš na oltáři*, který poté kombinoval s objektem *Nůž*. Po provedení těchto kombinací se uživatel dostal na scénu 26, která je považována za dobrý konec hry.

Průchody hrou v tomto testování jsou znázorněny na obrázku 7.2. Statistická data získána z testování uživatelem č.2 jsou uvedena v tabulce 7.4.



Obrázek 7.2: Průchod hrou uživatelem č.2

Jméno	Hodnota
Doba hraní	22:48
Scéna, na které uživatel strávil nejvíce času	22
Čas	2:34
Nejčastěji používaná akce	Jít
Počet použití	93
Objekt, s kterým byla nejčastěji provedena akce	Myší díra
Scéna, na které se objekt vyskytuje	22
Počet provedených akcí	19
Počet navštívených scén	103

Tabulka 7.4: Statistická data uživatele č.2

7.2.3 Testování uživatelem č.3

Poslední uživatel měl trochu problémy s ovládáním hry, protože není zvyklý na dotykové ovládání. Měl velké problémy se orientovat mezi scénami, a proto se také často vracel stále na stejné scény.

Při prvním průchodu se uživatel vydal přímo na scénu 5, kde ale provedl špatnou interakci a dostal se na scénu 32, která značí špatný konec hry.

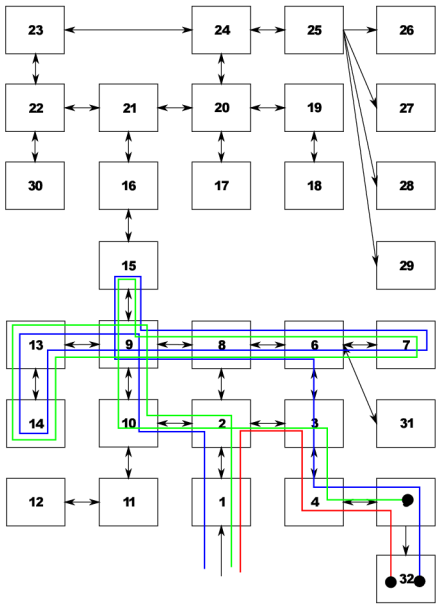
Při druhém průchodu uživatel zamířil opačným směrem a vydal se na scénu 14, kde získal objekt *Sirky*. Poté se přesunul na scénu 6, kde se pomocí akce mluvit přesunul na dialogovou scénu 7. Z této scény zamířil ke scéně 15, kde nalezl objekt *Zamčené dveře*, zde nepřišel na způsob, jak pokračovat dále. Z tohoto důvodu se přesunul na scénu 5, kde se pokusil získat objekt *Klíč*. Pří pokusu však použil špatnou akci a přesunul se tak opět na scénu 32.

V posledním průchodu uživatel opět navštívil scénu 14, kde získal opět objekt *Sirky*. Z této scény se opět vydal na scénu 6, odkud se přesunul na dialogovou scénu 7. Poté uživatel navštívil scénu 15, kde opět zjistil, že nemá možnost pokračovat a přesunul se tedy na scénu 5, kde byl nucen ukončit hru z časových důvodů.

Průchody hrou v tomto testování jsou znázorněny na obrázku 7.3. Statistická data získána z testování uživatelem č.3 jsou uvedena v tabulce 7.5.

Jméno	Hodnota
Doba hraní	1:14:56
Scéna, na které uživatel strávil nejvíce času	14
Čas	3:06
Nejčastěji používaná akce	Jít
Počet použití	228
Objekt, s kterým byla nejčastěji provedena akce	K přední křižovatce
Scéna, na které se objekt vyskytuje	2
Počet provedených akcí	20
Počet navštívených scén	228

Tabulka 7.5: Statistická data uživatele č.3



Obrázek 7.3: Průchod hrou uživatelem č.3

Kapitola 8

Závěr

Tato práce se dá rozdělit na tři části, které dohromady tvoří společné řešení. Základem je knihovna *AGLib*, kterou používají aplikace *AGI* a *AGCreator*. Ve vývojovém prostředí *AGCreatoru* je tedy možné vytvořit adventure hru a tu poté uložit jako modul pomocí knihovny *AGLib*. Takto vytvořený modul je možné načíst pomocí knihovny *AGLib* do aplikace *AGI* a zahrát si danou hru na platformě *Android*.

V rámci této práce jsem se seznámil s historií a problematikou adventure her. Věnoval jsem se podrobněji jejich principům a základním prvkům. Na základě těchto poznatků byla navržena a implementována knihovna *AGLib*. Tato knihovna v sobě obsahuje strukturu dat pro adventure hru a také se stará o průběh adventure hry. Knihovna byla napsána v jazyce *Java*, s kterým se mi pracuje velmi dobře. Při práci s tímto jazykem jsem však narazil na několik problémů, které plynuly hlavně z nemožnosti přistupovat přímo k paměti virtuálního stroje. Hlavním problémem bylo přepisování dat při předávání objektů odkazem (v jazyce *Java* není možné předávat objekty jinak než odkazem). Bylo tedy nutné vytvořit kopírovací konstruktory, které tuto hloubkovou kopii vytvořili. Při práci s *XML* jazykem jsem použil knihovnu *JDOM*, kterou jsem byl příjemně překvapen a ušetřil jsem si spoustu práce. Přesto by však tato knihovna mohla býti přehlednější. Knihovna *AGLib* je navržena tak, že by neměl být problém vytvořit pro ni libovolné grafické rozhraní, které by s knihovnou komunikovalo přes její interface. Knihovna je také připravena pro práci s vícejazyčnými hrami, kde je možné přepínat mezi jazykovými soubory. Do budoucna je možné knihovnu rozšířit, aby umožňovala použití animací a zvuků.

Jednou z aplikací, které v rámci této práce vznikly, je vývojové prostředí *AGCreator*. Tohle vývojové prostředí v sobě kombinuje vývojový nástroj společně s interpretem adventure her. *AGCreator* je postaven na knihovně *AGLib*, z níž používá reprezentaci dat a metody pro řízení průběhu adventure hry. Aplikace by měla umožňovat jednoduchý vývoj adventure her s možností libovolného přepnutí mezi interpretací a vývojem. Vývojové prostředí je vytvořeno pomocí grafické knihovny *Swing*, která je jednou z mála knihoven pro tvorbu grafických rozhraní v jazyce *Java*. Tato knihovna je již velmi stará a pracuje se s ní docela obtížně. Tvorba tabulek pro práci s atributy mi s touto knihovnou zabrala mnohem více času, než jsem očekával. Na druhou stranu se jedná o nejlépe dokumentovanou grafickou knihovnu, s kterou jsem měl možnost se setkat. Jako další rozšíření této aplikace by mohla být práce se zvukem a možnosti automatických oprav chyb v rozpracovaném projektu.

Poslední částí práce bylo vytvořit interpret pro platformu *Android*. Tento interpret jsem nazval *AGI*. *AGI* umožňuje hraní hry vytvořené ve vývojovém prostředí *AGCreator* a je také postaven na knihovně *AGLib*. Při tvorbě *AGI* jsem se poprvé setkal s vývojem na

platformě *Android*. S touto platformou se mi pracovalo velmi dobře a pohodlně. Stále se však mám co učit. Stejně jako v případě *AGCreatoru*, je i u *AGI* možnost rozšíření přidáním zvuků. Další rozšíření by mohla nabídnout online databáze her vytvořených v *AGCreatoru*, kde by se bylo možné připojit aplikací *AGI* a kteroukoliv z her si stáhnout.

Literatura

[1] Mladá fronta a. s. : Doupě.cz. 2013, [Online; cit. 18-04-2013].
URL http://doupe.zive.cz/

[2] Burda Praha s.r.o.: Hrej.cz. 2013, [Online; cit. 18-04-2013].
URL http://www.hrej.cz/

[3] Burda Praha s.r.o.: LEVEL. 2013, [Online; cit. 18-04-2013].
URL http://www.level.cz/

[4] Dead:Code Software: Wintermute Engine. 2010, [Online; cit. 15-12-2012].
URL http://dead-code.org/home/

[5] Google: Android. 2012, [Online; cit. 19-12-2012].
URL http://www.android.com/

[6] Hargreaves, S.: Allegro. 2012, [Online; cit. 15-12-2012].
URL http://alleg.sourceforge.net

[7] Jason Hunter : JDOM. 2012, [Online; cit. 26-04-2013].
URL http://www.jdom.org/

[8] Jones, C.: Adventure Game Studio. 2012, [Online; cit. 15-12-2012].
URL http://www.adventuregamestudio.co.uk

[9] Lantinga, S.: SDL. 2012, [Online; cit. 15-12-2012].
URL http://www.libsdl.org

[10] Ogre Team: Ogre 3D. 2009, [Online; cit. 15-12-2012].
URL http://www.ogre3d.org

[11] Oracle: Java SE Technical Documentation. 2011, [Online; cit. 19-12-2012].
URL http://docs.oracle.com/javase/

[12] Rollings, A.; Adams, E.: *On Game Design*. New Riders Games, New Riders Publ., 2003, ISBN 9781592730018.
URL http://books.google.cz/books?id=Qc19ChiOUI4C

[13] Sierra: Sierra Creative Interpreter. 2012, [Online; cit. 15-12-2012].
URL http://sciprogramming.com

[14] Ujbányai, M.: *Programujeme pro Android*. Průvodce (Grada), Grada, 2012, ISBN 9788024739953.
URL http://books.google.cz/books?id=o6kHGXua8oQC

[15] W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008, [Online; cit. 18-12-2012].
URL http://www.w3.org/TR/xml/

[16] Warren, A.: Quest. 2012, [Online; cit. 15-12-2012].
URL http://www.textadventures.co.uk/quest/

[17] Wolf, M.: *Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming*. ABC-CLIO, 2012, ISBN 9780313379369.
URL http://books.google.cz/books?id=deBFx7QAwsQC