Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

DIPLOMOVÁ PRÁCE

Plzeň, 2012

Jaromír Janisch

Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Interaktivní grafická aplikace pro platformu Android

Plzeň, 2012 Jaromír Janisch

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jaromír JANISCH**
Osobní číslo:	**A09N0077P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Název tématu:	**Interaktivní grafická aplikace pro platformu Android**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování:

1. Prozkoumejte dostupné možnosti pro tvorbu her pro platformu Android.
2. Porovnejte vybrané grafické, fyzikální či jiné podpůrné knihovny pro implementaci her pro platformu Android.
3. Na základě získaných znalostí navrhněte interaktivní grafickou hru.
4. Navrženou hru implementujte a ověřte její funkcionalitu.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **min. 40 stran původního textu**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Ing. Ladislav Pešička**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **31. srpna 2011**

Termín odevzdání diplomové práce: **17. května 2012**

Doc. Ing. František Vávra, CSc.
děkan

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Prof. Ing. Jiří Šafařík, CSc.
vedoucí katedry

V Plzni dne 15. září 2011

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2012 Jaromír Janisch

Abstract

The Master Thesis explores methods and suitability of making games on the Android platform. It is divided into two sections. The first section investigates the potential of the platform from the view of companies, developers and users. It also describes several libraries and game engines usable for game development. Based on the knowledge gained in the first part, an interactive game is designed. The second part explains the overall composition of the application and also describes its modules in detail.

Obsah

Kapitola 1

Úvod

Diplomová práce se zaobírá problematikou vývoje her pro platformu Android. Ačkoliv je pojem hra velmi obsáhlý termín a existují dokonce výzkumy definující jeho hranice [14], práce se bude zabývat pouze interaktivními grafickými hrami. Grafická hra bude pro tyto účely definována jako netriviální grafická aplikace využívající alespoň engine na úrovni OpenGL.

V první části jsou prozkoumány možnosti systému Android z pohledu programování her. Je probrána jeho vhodnost z několika úhlů pohledu – firmy zajímající se o zisk, uživatele a jeho možnostech a nakonec programátora. Pro každé téma jsou uvedeny referenční hry, které se ho týkají. V technických záležitostech je stručně popsáno jejich použití a uvedeny odkazy na literaturu či jiné zdroje, kde může čtenář dále rozšířit své znalosti. Dále jsou probrány knihovny a enginy pro podporu vývoje her na Androidu.

Na základě získaných znalostí jsou vybrány vhodné externí knihovny a enginy a na jejich základě je vyvinuta ukázková aplikace, která je nedílnou součástí této práce. V druhé části textu je uveden stručný manuál a popsána její implementace. Součástí druhé sekce je také popis dosažených výsledků. Jsou provedeny testy na několika různých zařízeních a souhrnně prezentovány.

Kapitola 2

Teoretická část

2.1 Stručný přehled platformy

Android je softwarový framework postavený na operačním systému Linux pro mobilní zařízení od firmy Google Inc., ačkoliv některá literatura označuje Android samotný jako operační systém [1]. V dalším textu nebude na toto rozlišení brán zřetel a Android bude označován jako systém nebo platforma. Android je publikovaný pod licencí Apache 2.0[1], ačkoli některé jeho části jsou pod jinou licencí, např. jádro Linuxu s licencí GPLv2[2] [2]. V praxi to znamená, že systém Android není vázaný pouze na jednu hardwarovou platformu, ale je adaptován na spoustu rozmanitých zařízení. Je např. velmi rozšířen na trhu levných čínských mobilních zařízení [3]. Firma Amazon dokonce zašla tak daleko, že modifikovala Android pro své potřeby a vydala jeho upravenou verzi v podobě zařízení Kindle Fire [4]. Aplikace od Googlu jsou v zařízení nahrazeny aplikacemi od Amazonu a dokonce v něm není obsažen ani Google Market – namísto něj je v zařízení Amazon App Store.

Android je velmi perspektivní platforma. Dle statistik firmy Gartner se jen během roku 2010 zvětšil tržní podíl Androidu z 9.6% na 36% a překonal tak všechny ostatní platformy [8]. Kromě liberální licence za to může zejména velmi vyspělá vývojářská podpora v podobě spousty tutoriálů, video návodů a konferencí[3]. Vývojové prostředí Androidu je postaveno na velice populární platformě Eclipse a programy se píší v jazyce Java. Vývojáři se tak nemusí učit další programovací jazyk nebo odlišné prostředí. Důležitý je také nepřetržitý vývoj celé platformy. Za čtyři roky existence Android prošel čtyři velké verze, z nich každá znamenala podstatné vylepšení. Vývojový kit Android SDK[4] je navržen velmi profesionálně a zvládnutí práce s ním je poměrně jednoduché. Pro výkonově náročné části aplikace nebo pro snadný import aplikací z jiných platforem Android podporuje i programování v nativním C++ kódu díky Android NDK[5].

[1] `http://www.apache.org/licenses/LICENSE-2.0.html`
[2] `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt`
[3] Nejznámnější je každoroční Google I/O
[4] Software developement kit, `http://developer.android.com/sdk/index.html`
[5] Native developement kit, `http://developer.android.com/sdk/ndk/index.html`

O programování aplikací na systému Android a jeho struktuře bylo již napsáno bezpočet publikací a proto se tato práce nebude znovu zabývat vysvětlováním základních principů platformy. Čtenář by měl mít, pro pochopení obsahu této práce, základní poznatky o systému a programování na něm. V práci J. Rejthara [6] se čtenář může seznámit se základními součástmi systému a projít si ukázkové aplikace. V knize [7] je navržena metodika vývoje androidí aplikace, kterou je do jisté míry tato práce ovlivněna.

2.2 Zaměření mobilních her

Prvním faktorem, který určuje, jaké hry budou na mobilních zařízeních vyvíjeny, je výkon. Dnešní mobilní zařízení se systémem Android, tedy mobilní telefony a tablety, ukrývají až nečekané možnosti. Pokud srovnáme moderní zařízení, dojdeme k závěru, že většina již disponuje dvěmi jádry CPU (např. velmi rozšířený Samsung Galaxy S II [5]) a dedikovaným grafickým čipem [9]. Frekvence CPU se pohybují kolem 1 GHz a paměti RAM bývá k dispozici 512 MB. Grafické čipy jsou schopny akcelerovat grafiku za použití OpenGL ES 2.0, včetně vymožeností ještě před několika lety vyhrazenými pro výkonné desktopové počítače. Zařízení s čipy Nvidia Tegra 2, které v sobě integrují jak CPU, tak GPU, se na trhu začaly objevovat v roce 2010 a jejich výkon umožňuje hraní skutečně vizuálně stimulujicích her. Zařízení dále obsahují vnitřní úložnou paměť ve velikostech kolem 10 GB, často dále rozšiřitelnou externí SD kartou. Není tedy problém vybavit aplikaci prvky náročnými na místo v úložné paměti, jako jsou kvalitní textury, modely či hudba.

Limitujícím faktorem mobilních telefonů je malý displej, který je předurčuje spíše na občasné hraní. Výzkum Qiang Xu [10] ukázal, že lidé hrají hry hlavně když se pohybují mimo domov, tedy na cestách, v tramvaji apod. Na současném trhu převažují zejména takové oddychové hry, které nejsou složité, nevyžadují plánování do budoucna, ale dokáží zkrátit dlouhou chvíli. Mezi takové hry patří nyní populární Angry Birds[6], Cut the Rope[7] či Osmos[8]. Odlišná situace panuje u tabletů, u kterých některá média předpovídají, že nahradí současné konzole [12]. Pravé rozšíření tabletů však lze teprve očekávat. Podle statistik Googlu [11] jsou tablety s velkou obrazovkou mezi všemi zařízeními Android zastoupeny pouze necelými šesti procenty.

Posledním faktorem, ovlivňujícím tvorbu her na platformě Android, je možnost jejich monetizace (výdělek). Pokud někdo publikuje aplikaci na marketu, dělá tak téměř určitě ze snahy o výdělek. Ten je možno dosáhnout třemi základními cestami. První možnost je nabízet danou aplikaci zdarma s vloženými reklamami. Úspěch hry Angry Birds, která své firmě vydělává na reklamách jeden milion dolarů měsíčně [13], ukazuje, že prostor pro tuto metodu skutečně existuje. Výhoda tohoto způsobu je, že výdělek od jednoho uživatele není jednorázový a je zde potenciál

[6] `https://play.google.com/store/apps/details?id=com.rovio.angrybirds`
[7] `https://play.google.com/store/apps/details?id=com.zeptolab.ctr.paid`
[8] `https://play.google.com/store/apps/details?id=com.hemispheregames.osmos`

pro pravidelné výdělky, pokud se majitel o svou aplikaci stará a aktualizuje ji. Navíc se hra velmi rychle rozšíří, protože zde není žádná překážka v podobě platby.

Druhá možnost je aplikaci regulérně prodávat za předem stanovený finanční obnos. Nevýhodou tohoto způsobu je zejména počáteční bariéra, která brání uživatelům platit za neověřený produkt. Spousta uživatelů také zkrátka za aplikace platit nechce. Pokud si aplikaci již někdo koupí, je daná suma konečná a víc peněz z uživatele neplyne (samozřejmě to není zcela pravda – pokud si vytvoříme věrnou základnu platících uživatelů, lze je dále zhodnotit např. poměrně snadnou konverzí na novou aplikaci). Tuto možnost využila např. hra Cut the Rope, kterou si za cenu jednoho dolaru koupilo přes jeden milion uživatelů [16].

Třetí cesta jak nechat aplikaci vydělávat je v poslední době hojně rozšířený model Freemium. Aplikace je nabízena zdarma se základní funkčností. Využívá tak výhody prvního způsobu, tedy že si mnoho lidí hru stáhne a ta se může rychle rozšířit. Následně jsou hráči přímo ve hře nabízeny prémiové služby, ať už se jedná o lepší vybavení, přístup ke speciálním úrovním či nákup herní měny. Zástupcem této kategorie je hra Gun Bros, která na tomto modelu za první dva měsíce vydělala přes 600 tisíc dolarů [17].

Uvedené faktory mají za následek zejména to, že na Android cílí především vývojáři indie her[9]. Spousta her, původně zaměřených na desktop, byla na platformu portována. Mezi všemi lze zmínit např. World of Goo[10], již zmiňovaný Osmos nebo Swords and Soldiers[11].

2.3 Možnosti pro vývoj her

2.3.1 Předmluva

Jak již bylo řečeno, aplikace pro Android se píší v Javě. Přestože se tyto aplikace od standardní Javy mírně liší, např. v životním cyklu aplikace, stále je to skutečně plnohodnotná Java. To umožňuje, mimo jiné, využít všech dostupných knihoven v rámci Javy nebo dokonce externích knihoven, kterých je skutečně velké množství. Vývojáři mají také možnost využít nativního kódu v C++. Tato možnost je vhodná v případě výkonově náročných částí nebo pro snadný port existující aplikace napsané v C++ z jiné platformy. Např. populární hra Angry Birds od firmy Rovio je psaná v C++. To jí mj. umožnilo snadné rozšíření na různé platformy, např. iPhone či desktop. Navíc to dále rozšiřuje možnosti externích knihoven – některé jsou psány jen v C++.

Protože je Android svobodně šířen i se zdrojovými kódy, objevil se nespočet různých zařízení, na kterých tento systém běží. Na rozdíl od iPhone od Applu je tak třeba podporovat mnoho hardwarově odlišných zařízení. Toto břemeno je často ponecháno na programátorovi. Systém Android se mu snaží v některých úskalích pomáhat, např. ve správě aplikačních zdrojů běžných aplikací závislých

[9] Nízkorozpočtové hry tvořené zpravidla skupinou odhodlaných lidí

[10] `https://play.google.com/store/apps/details?id=com.twodboy.worldofgoofull`

[11] Aplikace není dostupná na Google Play, `http://www.swordsandsoldiers.com/`

na rozlišení [23]. Pro hry však tato výpomoc často pozbývá významu a vývojáři se musí s rozdílnými platformami vypořádat sami.

Následující části probírají jednotlivé aspekty vývoje her na Androidu. Budou probrány jak možnosti dostupné v základním Androidu, tak možnosti dostupné za pomoci externích knihoven.

2.3.2 Ovládání

Aplikace se primárně ovládají dotykem. V Androidu 2.0 byla tato možnost rozšířena o multi-touch, tedy dotyk více prsty zároveň [22]. Tento způsob ovládání zcela zásadním způsobem ovlivňuje koncepci hry. Přináší to jistá omezení, na druhou stranu však i prostor pro inovaci. Hry, které to pochopily a kam se řadí již zmiňované Angry Birds či Cut the Rope, se staly velmi populární.

Dotykové ovládání se dá využít několika způsoby. Je možno jím simulovat styl ovládání, na který jsou hráči zvyklí z jiných platforem. Nejběžnějším ovládáním tohoto druhu jsou tlačítka na obrazovce nebo *on-screen joystick*, tedy joystick na obrazovce. Ten hráč ovládá typicky jedním prstem a pohybuje díky němu herním avatarem. V jiné části obrazovky se pak nacházejí prvky pro akci, např. střelbu. Příkladem hry využívající tuto kombinaci je např. hra Gun Bros[12] na obrázku 2.1.

Obrázek 2.1: Virtuální joysticky ve hře Gun Bros

Příjem dotykových vstupních událostí je prováděn v metodě `onTouchEvent()` implementované v běžící `Activity`. Metoda předá v parametru typu `MotionEvent` informace o dotyku. Systém sám spravuje reference na jednotlivé prsty, které jsou nazývány *pointers*. Jejich *id* lze získat z `MotionEvent` funkcí `getPointerId()`.

[12] `https://play.google.com/store/apps/details?id=com.glu.android.gunbros_free`

Systém je částečně matoucí a pro plné pochopení je doporučeno přečíst si celý článek [22].

Mobilní zařízení s Androidem jsou typicky vybavena dalšími senzory – akcelerometrem, snímačem magnetického pole. První se mj. často používá k měření směru gravitační síly. Lze tedy jako ovládání použít náklonu telefonu. Toho je často využíváno u závodních her, např. Asphalt 6[13], nebo u populární hříčky Abduction[14]. Senzory se používají skrze třídy `SensorManager` u které se skrze metodu `registerListener()` registruje `SensorEventListener` a který poté v metodě `onSensorChanged()` přijímá události o změnách dat ze senzoru.

Posledním prvkem který se dá využít pro hry, ačkoliv to není ovládání v pravém slova smyslu, je fotoaparát. Ten je základem tzv. Augmented Reality her, ve kterých se na obrazovku promítá obraz snímaný kamerou a zároveň se v něm objevují herní objekty, avataři nebo monstra. Tento koncept zatím není příliš rozšířen a k dispozici je jen několik zajímavých, ale v celku nekvalitních her, např. Sky Siege[15] ve které hráč sestřeluje vrtulníky a jiné létající objekty či SpecTrek[16], ve které hráč hledá v okolí duchy a chytá je. Je to však poměrně perspektivní oblast a tak se v brzké budoucnosti možná dočkáme nějakého příjemného překvapení.

Ukázková aplikace podporuje multitouch i ovládání skrze akcelerometr.

2.3.3 Grafika

Pro grafickou část aplikace je možné využít standardního systémového API v podobě `Canvas` a `SurfaceView` [15]. Přestože je toto veskrze objektové API od verze Androidu 3.0 hardwarově akcelerováno, je vhodné spíše pro GUI aplikace a pro netriviální hry se příliš nehodí a proto se jím nebudeme dále zaobírat.

Druhou možností je využít OpenGL ES[17]. Podpora této knihovny ve verzi 1.0 a 1.1 je součástí Androidu již od verze 1.0. Ve verzi Androidu 2.2 přibyla podpora OpenGL ES 2.0, která je od předchozí verze poměrně odlišná. OpenGL ES je podmnožinou desktopového OpenGL přizpůsobenou pro účely přenosných zařízení a tak je znalost OpenGL pro vývoj v OpenGL ES výhodou. Podrobný popis knihovny je možno nalézt v knihách [19], [20] a [21].

Vývojář se musí před začátkem prací na aplikaci rozhodnout, zda bude využívat OpenGL ES 1.1 nebo OpenGL ES 2.0. Verze se od sebe totiž výrazně liší a přechod z jedné verze na druhou je poměrně komplikovaná záležitost. V první verzi OpenGL ES se využívá tzv. *fixed pipeline*. V podstatě to znamená, že API obsahuje nějakým způsobem funkce pro nastavení matic *model*, *view* a *projection*, funkce pro manipulace se světly a jiné. Svým způsobem tak programátorovi usnadňuje implementaci nabízením hotových komponent, na druhou stranu jej však omezuje.

[13] `https://play.google.com/store/apps/details?id=com.gameloft.android.ANMP.GloftA6HP`

[14] `https://play.google.com/store/apps/details?id=au.com.phil`

[15] `https://play.google.com/store/apps/details?id=madfirm.skysiege`

[16] `https://play.google.com/store/apps/details?id=com.spectrekking.full`

[17] OpenGL for Embedded Systems

Ve verzi OpenGL ES 2.0 je *fixed pipeline* nahrazena *programable pipeline*. API pozbylo zmíněné funkce a je na programátorovi, jakým způsobem je nahradí. Náhradou za *fixed pipeline* jsou tzv. *shadery*, což jsou programy, které běží přímo grafické kartě. Shadery se píší ve speciálním jazyce glsl, který je specifikován v dokumentu [18] nebo popsán srozumitelnější formou v knize [20]. V grafickém čipu jsou typicky spousty funkčních jednotek, které shadery paralelně zpracovávají. Různé grafické karty implementují shadery jinak a podporují v nich různé konstrukce. Některé například nepodporují indexaci pole pomocí proměnné[18]. Shadery jsou vždy ve dvojici – *vertex shader* a *fragment shader*, přičemž první zpracovává vertexy[19] a výsledky předává do fragment shaderu, kde se nastavuje výsledná barva každého vykresleného bodu.

Použití shaderů přenáší zodpovědnost za správné zobrazení z enginu na programátora, avšak poskytuje mu mnohem rozsáhlejší možnosti. Pomocí shaderů se dají vytvořit různé efekty, např. deformace objektů, nerealistické vykreslování v cartoon stylu či speciální efekty se světly.

Přístup k OpenGL ES je v Androidu podporován jak pomocí javového API frameworku, tak přímo z nativního kódu v C++. Pokud se programátor rozhodne využít nativní kód, musí zajistit veškeré náležitosti související s vykreslováním. Pokud ale využívá API frameworku, je mu přístup k OpenGL ES významně usnadněn. Programátor oddědí vlastní třídy od `GLSurfaceView` a `GLSurfaceView.Renderer`. První třída slouží jako kontejner `View` k vykreslování a také přijímá uživatelský vstup. V druhé třídě se implementuje metoda `onDrawFrame()`, která se vyplní výkonným kódem.

Ukázková aplikace demonstruje využití OpenGL ES 2.0.

2.3.4 Skripty

Skript je externí program či posloupnost příkazů, která určitým způsobem ovlivňuje samotnou hru nezávisle na hlavním kódu. Ačkoliv jsou systém skriptů ve hře nepovinný, výrazně rozšiřuje její možnosti. Pro vývoj složitějších her je nezbytný. Skripty umožňují dynamické chování hry, snadnější implementaci rozšíření a hlavně oddělují vývojáře od návrhářů. Vyvojáři píší kód hry, zatímco návrháři mohou pomocí skriptů vytvářet úrovně, příběh či herní setkání. Obě skupiny jsou nezávislé a mohou se soustředit jen na svůj úkol. Pokud je systém skriptů veřejně publikován, umožňuje to komunitě vytvářet rozšíření samostatně. Zájem komunity může hru rozvést daleko za původní představy vývojářské firmy a může rozhodnout o jejím úspěchu či zatracení.

Zájemci na toto téma je doporučeno přečíst si knihu [24]. Tato část probere některé možnosti, jak navrhnout skriptovací systém. Základní požadavky, které musí systém splňovat jsou:

- snadné psaní skriptů

[18] Např. SGX 540, který je součástí Samsung Galaxy S
[19] Vertex je v podstatě vrchol modelu

- snadná integrace do aplikace
- dostatečné možnosti ovlivnění průběhu hry

Možností jak implementovat skripty je několik. V zásadě jsou to:

- použít vlastní skriptovací jazyk
- použít nějaký skriptovací framework
- použít kompilovaný jazyk s možností externího spouštění

První možnost je pouze pro otrlé vývojáře a ty, kteří chtějí specifické vlastnosti[20]. Pokud se podíváme za hranice Androidu, je tato možnost použita např. v populární sérii her od BioWare Neverwinter Nights [25].

Druhá možnost je podstatně snažší. Hra Angry Birds používá jazyk Lua, který se prezentuje slovy: „Lua is a powerful, fast, lightweight, embeddable scripting language“ (Lua je mocný, rychlý, lehký a vložitelný skriptovací jazyk) [28]. V tomto skriptovacím jazyce je napsána prakticky celá logika hry, včetně menu apod. V roce 2008 se rozběhly práce na projektu ASE[21], což je v podstatě framework na jednoduché spouštění externích skriptů [26]. Projekt se poměrně rozrostl a v roce 2010 byl přejmenován na SL4A[22] [27]. Je stavěn přímo na Android a podporuje mnoho skriptovacích jazyků: BeanShell[23], Lua, Perl[24], Rhino[25] a Python[26]. Pomocí této knihovny může programátor vložit do své hry skriptovací podporu s minimálním úsilím.

Třetí možnost je použití kompilovaného jazyka a jeho provázání s původním kódem skrze rozhraní. Protože se aplikace pro Android vyvíjí v Javě a ta podporuje reflexi[27], je zde možnost použití Javy jako skriptovacího jazyka. Komunikace pak probíhá skrze rozhraní publikovaná hlavní aplikací. Skript implementuje rozhraní, díky kterému mu může aplikace zasílat události a aplikace exportuje rozhraní plnící roli *kontroléru*, skrze které zasílá události skript do ní. Skript, reprezentovaný Java třídou, pak může být dynamicky nahrán za běhu aplikace.

Návrh architektury systému skriptů dle třetí možnosti je na obrázku 2.2. V zásadě mohou být skripty spouštěny aplikací pomocí

a) pevně definovaných míst v kódu,

b) událostí na základě nějaké herní změny.

[20]Důvodem k použití vlastního jazyka může být zaměření na lidi, kteří nejsou orientováni na programování, tedy upřednostnění jednoduchosti před funkcionalitou.

[21]Android Scripting Environment

[22]Scripting Layer for Android

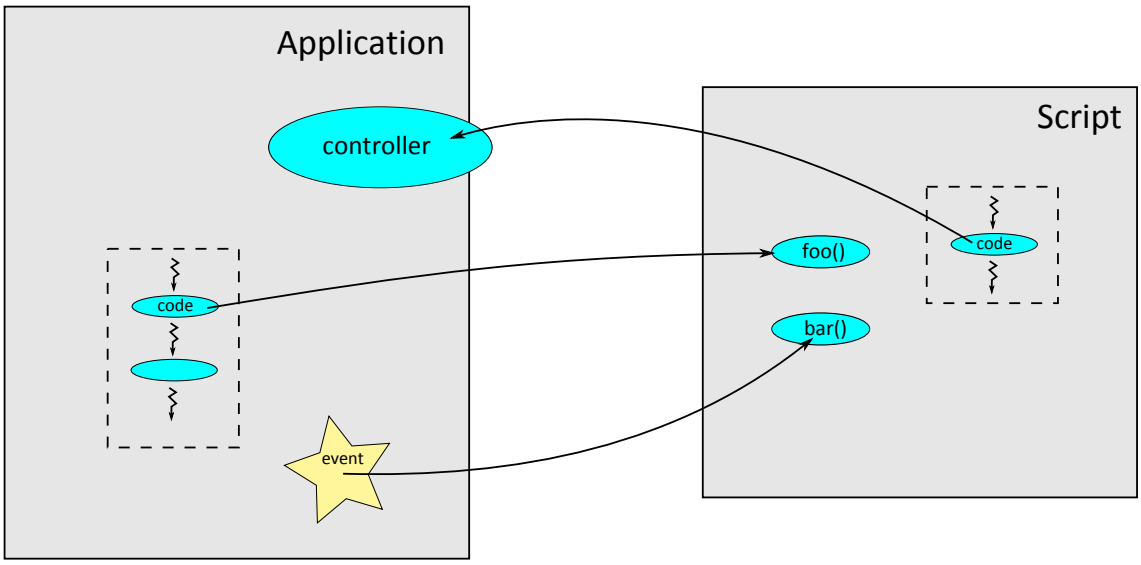
[23]Lightweight Scripting for Java, `http://www.beanshell.org/`

[24]The Perl Programming Language, `http://www.perl.org/`

[25]Rhino: Javascript for Java, `http://www.mozilla.org/rhino/overview.html`

[26]Python Programming Language, `http://www.python.org/`

[27]Reflexe je schopnost jazyka zjišťovat informace o kódu, pozměňovat jej nebo načítat jeho části za běhu.



Obrázek 2.2: Ukázka spojení skriptu s aplikací

První případ nastane např. při volání funkce `render()` pro každý vykreslení snímek nebo `init()` při inicializaci skriptu. Druhý případ nastane např. při vstoupení herního objektu do určité oblasti, při které se zavolá definovaná funkce skriptu. Této události se také říká *event*. Skript naopak komunikuje s aplikací přes *kontrolér* (*controller*).

V ukázkové aplikaci je využita tato třetí možnost.

2.3.5 Fyzika

Fyzikálně založené hry jsou z pohledu hráčů poměrně atraktivní, což dokazuje popularita her jako Labyrinth[28] nebo Apparatus[29]. Programátor má možnost si navrhnout a implementovat fyzikální systém sám, což se vyplatí v případě speciálních požadavků nebo pouze omezených funkcí (např. ve hře Osmos), nebo využít některého z fyzikálních enginů. Těch existuje pro Android hned několik a většinou jsou to porty z jiných platforem.

Fyzikální enginy můžeme rozdělit do dvou kategorií: 2D a 3D. V kategorii 2D je výběr veliký [33] – Box2D[30], APE[31], Glaze[32] nebo Chipmunk[33]. Za zmínku stojí zejména engine Box2D a Chipmunk, mezi kterými se vývojáři se často rozhodují [32]. Box2D je psán v C++ [34] a Chipmunk v C [35], což je umožňuje využít v Androidu skrze NDK. Nicméně pro oba enginy jsou k dispozici Java bindingy[34], které je umožňují využít i z Java prostředí [36] [37]. Obě knihovny jsou k dispozici

[28] `https://play.google.com/store/apps/details?id=se.illusionlabs.labyrinth.full`
[29] `https://play.google.com/store/apps/details?id=com.bithack.apparatus`
[30] `http://box2d.org/`
[31] `http://www.cove.org/ape/`
[32] `http://code.google.com/p/glaze/`
[33] `http://chipmunk-physics.net/`
[34] Java binding označuje třídy, metody, parametry a návratové hodnoty, které zaobalují spodní vrstvu nativního kódu, která je skrze něj volána

zdarma a jsou dobře zdokumentované. Chipmunk má menší komunitu, za to čistší API. Box2D je častěji aktualizovaný a objektově orientovaný [32].

V kategorii 3D se výběr pro Android notně zužuje. Je možné využít fyzikální engine firmy NVIDIA PhysX, který je pro komerční i nekomerční použití zdarma [29]. Kromě samotného enginu jsou k němu dostupné různé nástroje, např. pluginy pro grafické editory nebo analyzátor scén [30]. Další možností je engine Havok od stejnojmenné firmy, který však již pro Android zdarma není [38]. Jsou k němu k dispozici nástroje podobné těm od PhysX. Třetí možností je Bullet Physics, který je zdarma [39]. Navíc pro něj existuje Java port, který však nedrží tempo s vývojem původního projektu a nepodporuje všechny funkce [40]. Pro tvorbu fyzikálních modelů v tomto enginu je možné využít např. software Blender[35], ve kterém je podpora tohoto enginu zahrnuta [31].

Jako fyzikální engine je v ukázkové aplikaci využit 2D engine Box2D.

2.3.6 Sociální sítě

Hry mohou využít služeb sociálních sítí, díky kterým se mohou velmi rychle rozšířit mezi velkou masu uživatelů. Toho lze dosáhnout např. publikování skóre, achievementů[36] přímo ze hry na hráčův profil. Ten může obsahovat odkaz na původní hru a vidí ho všichni známí uživatele. Někteří uživatelé se rádi pochlubí svými výsledky a jsou za jejich automatické publikování rádi. Povědomí o hře se tam může velmi efektivně šířit. Pokud se navíc hra k tomuto prvku chytře postaví a obohatí jej např. o hráčské bonusy, čímž může být např. herní měna, zdarma za publikování příspěvků na hráčově profilu, může hráče dále motivovat a odbourat případný odmítavý postoj.

K usnadnění uvedeného postupu postytují sociální sítě často API. Hlavní sociální síť dneška, Facebook, poskytuje své Android SDK, které velmi zjednodušuje jeho používání v aplikaci [41]. S jeho pomocí je možno tuto síť bez problémů integrovat do hry. Pokud je uživatel na svém zařízení navíc regulérně do sítě přihlášen, nemusí při použití tohoto SDK znovu zadávat přihlašovací údaje a celý proces se pro něj zjednodušuje. Z pohledu uživatele je tak celá integrace otázkou několika kliknutí. Jedním z představitelů her, který tento systém využívají je např. hra Gun Bros, jíž se díky své provázanosti se sociálními službami a Freemium modelem zatím poměrně daří.

Pokud by se vývojář rozhodl ze své hry využívat služby Twitter, bude mít o trochu větší problémy než v případě Facebooku. Twitter totiž žádné oficiální SDK pro Android neposkytuje, pro přístup k němu se používá např. rozhraní REST[37]. K vyplnění této mezery vzniklo několik různých projektů, které se snaží SDK pro Android nahradit [44]. Knihovny jako je Twitter4J[38] nebo JTwitter[39] jsou psány

[35] `http://www.blender.org/`

[36] Achievement je herní cíl, který se odemkne po splnění určitých podmínek.

[37] Representational state transfer, technologie založená na HTTP

[38] `http://twitter4j.org/`

[39] `http://www.winterwell.com/software/jtwitter.php`

v Javě a lze je tak snadno v Androidu použít. Bohužel se u nich nedá očekávat propracovaná dokumentace a tutorialy.

Kromě těchto běžných sociálních sítí se rozšířila řada sítí zaměřených přímo na těsnou integraci s hrami. Tyto sítě poskytují ve svém API služby jako správa achievementů, žebříčků nebo skóre. Uživatelé mohou mezi sebou komunikovat a sdílet své statusy jako v běžných sociálních sítích, ale navíc jsou např. sdruženi podle her, které mají nainstalované. Příklady takových sítí je např. populární OpenFeint[40], který je využit např. ve hře Stupid Zombies[41], či Papaya se hrou Papaya Farm[42].

2.4 Herní frameworky

2.4.1 Přehled

Způsobů, jak vytvářet grafické hry pro mobilní platformu Android je spousta. Programátor může použít implementaci OpenGL ES poskytovanou již v základních knihovnách Androidu. Její použití je však poměrně nízkoúrovňové a dokončení nějaké smysluplné aplikaci by vyžadovalo mnoho úsilí, ačkoli to není nemožné – hra Replica Island je vytvořena bez využití jakéhokoliv externího frameworku [45].

Lepší způsob je využít pro svou práci jeden z dostupných herních frameworků. Ty programátora izolují od nízkoúrovňových záležitostí a povětšinou poskytují vysokoúrovňové API, s kterým se snadněji pracuje. Zastávají tak mnoho funkcí a zjednodušují práci. Na druhou stranu vyžadují určitý přístup a nutí vývojáře používat vlastní zavedené praktiky, což může být někdy omezující. Součástí těchto frameworků, nebo enginů, bývají často také různé nástroje. Ty dále zjednodušují práci a jsou důležitým kritériem pro výběr daného frameworku.

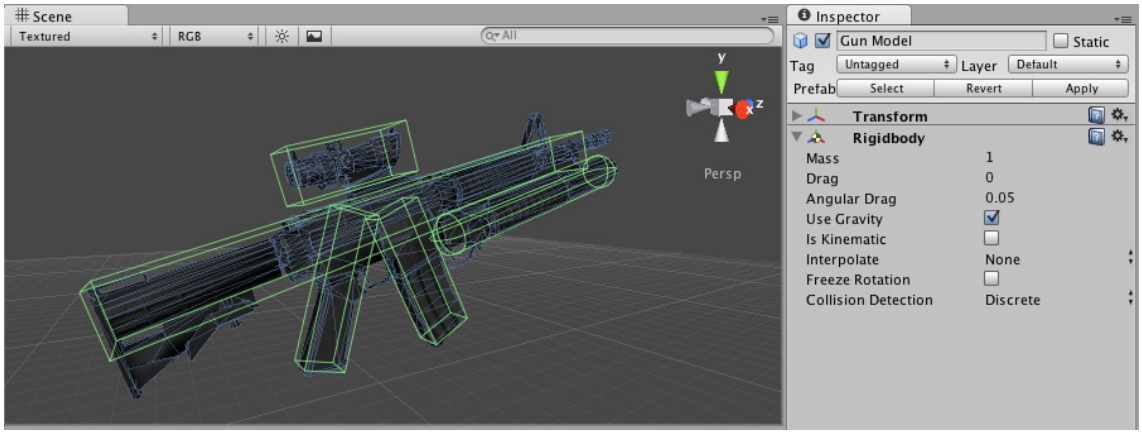
V následujícím přehledu uvedu několik dostupných frameworků pro tvorbu her na Androidu. Jsou rozděleny do dvou velkých kategorií: 2D a 3D. Ve výčtu budou uvedeny jejich vlastnosti jako je podpora ostatních platforem, možnosti, dostupné nástroje, obsažený fyzikální engine, uživatelská podpora a komunita a cena. Celkový přehled je uveden v tabulce A.1.

2.4.2 Unity

Unity[43] je vskutku masivní a profesionální framework pro tvorbu 3D her, který podporuje mnoho různých platforem, mj. Microsoft Windows, iPhone, Adobe Flash, Wii, PS3, dokonce i hraní v prohlížeči a od roku 2008 i Android [46]. Ačkoli je v základní verzi zdarma, licence pro Android stojí $400. Součástí Unity je jak samotný engine, tak i celé IDE[44]. Programy se mohou psát hned v několika jazycích [47],

[40] `http://openfeint.com/`
[41] `https://play.google.com/store/apps/details?id=com.gameresort.stupidzombies`
[42] `https://play.google.com/store/apps/details?id=com.papaya.papayaandroidfarm`
[43] `http://unity3d.com/`
[44] Integrated development environment, zkátka editor

např. .NET, UnityScript, nebo Boo[45]. Spíše se však zaměřuje na grafické rozhraní a editaci různých prvků. Unity je tedy velmi vysokoúrovňový nástroj. Ukázka grafické editace fyzikálních hranic objektu je na obr. 2.3.



Obrázek 2.3: Ukázka z vývojového prostředí Unity

Engine obsahuje množství nástrojů, mezi nimi např. editor animací, charakterů či částicových efektů. Unity obsahuje podporu AI[46], spoustu grafických efektů a možností, např. LOD[47], stíny, zaostřování scény, bloom. Unity dále podporuje multiplayer a obsahuje i moduly pro zpracování a nastavení zvuků. Unity má v sobě integrovaný PhysX fyzikální engine.

Unity má obrovskou uživatelskou základnu. Fórum registruje na 100 tisíc uživatelů [48], a na Unity Answers bylo položeno na 50 tisíc otázek [49]. Pořádají se živé online kurzy [50] i živé konference [51]. Pro tvorbu androidích her je i přes vyšší počáteční náklady velmi populární a v současné době je publikováno přes sto her postavených na tomto frameworku [52].

Tento framework může být díky svým možnostem poměrně dobrou volbou pro profesionální vývojáře mířící vysoko. Navíc umožňuje poměrně snadný port již napsané aplikace na ostatní podporované platformy a tak je dobrou investicí i do budoucna.

2.4.3 jPCT

jPCT[48] je další 3D framework, tentokrát poskytovaný zdarma. Knihovna sestává z několika Java tříd, které se integrují do Androidího projektu. To znamená, že se projekty píší v Javě, jako jakákoliv jiná aplikace.

Možnosti enginu nejsou obsáhlé a je poměrně nízkoúrovňový. Podporuje načítání grafických modelů, animací, textur či shaderů. Je možné jej využít jak v OpenGL ES 1.0 tak i s OpenGL ES 2.0. Dále umožňuje detekci kolizí a obsahuje funkce pro práci se světly. Knihovna neobsahuje žádný fyzikální engine, ale autor

[45] Více o Boo na `http://boo.codehaus.org/`
[46] Umělá inteligence
[47] Level of Detail
[48] `http://www.jpct.net/jpct-ae/`

nabádá k použití engine Bullet Physics, resp. jeho port jBullet, a jsou k němu k dispozici ukázky [57].

Engine není příliš často aktualizovaný [53], nicméně jako dobrý základ pro nízkoropočtový projekt by mohl posloužit. K enginu je dostupná dokumentace, několik tutorialů a ukázkové aplikace [56]. Komunita není veliká, na fóru je registrováno bezmála tisíc členů [55]. Na stránkách enginu je registrováno téměř dvacet her, které jej využívají a jsou publikovány na Google Play [54].

2.4.4 libGDX

LibGDX[49] je herní engine s otevřeným kódem. Je možné jej použít zdarma pro nekomerční i komerční účely. Engine podporuje jak Android, tak PC platformu. Kód se pro obě platformy píše pouze jednou a je okamžitě spustitelný na obou. To přináší jisté výhody, protože desktopové počítače jsou několikanásobně výkonější než mobilní zařízení a odpadá tak zdlouhavé nahrávání aplikace do zařízení a komplikovaný debugging. Ladění se tak stává rychlejší a pohodlnější. Tento framework je spíše zaměřený na 2D, ale není problém s ním vytvořit 3D hru.

Abstrakce od Androidí nebo PC platformy je možná díky tomu, že knihovna libGDX poskytuje své vlastní API, které zaobaluje rozdíly obou platforem. Poskytuje dále transparentní API pro práci s OpenGL, kde je pro každou funkci napsán wrapper, který ji umožní spustit nativně na Androidu nebo PC. Práce s OpenGL je na spodní úrovni napsána v nativním kódu, což zaručuje vysoký výkon [58]. Obsahuje rozhraní pro práci se grafickými prvy jako jsou shadery, textury, vertex buffery apod., dále vysokoúrovňové API pro tvorbu 2D scén a UI[50], API pro načítání 3D modelů z různých formátů. Dále zaobaluje práci se soubory, zvuky, uživatelským vstupem a obsahuje integrovanou fyzikální knihovnu Box2D. Ta je také psána v nativním kódu, pro práci s ní je ale k dispozici Java rozhraní.

Ke knihovně patří řada nástrojů – Particle Editor k snadnému generování částicových efektů. Výsledný efekt lze z nástroje exportovat a ihned použít ve hře. Další je Hiero, nástroj pro tvorbu bitmapových fontů a poslední TexturePacker pro umístění více textur do jednoho souboru.

Knihovna libGDX je vyvíjena hrstkou vývojářů [59], avšak je stále aktivně rozvíjena [60]. Engine má středně velkou uživatelskou základnu, fórum čítá na 2000 členů [63]. Her s touto platformou je publikováno již několik desítek [61] [62]. Dokumentace je stále roztroušená a neúplná, ale přesto poměrně použitelná.

2.4.5 cocos2d-x

Cocos2d-x[51] je herní framework zaměřený na 2D a podporující zejména platformy Windows, Android, iPhone, ale i další [64]. Je licencován pod MIT licencí[52], takže

[49] `http://libgdx.badlogicgames.com/`
[50] uživatelské rozhraní
[51] `http://www.cocos2d-x.org/`
[52] `http://www.opensource.org/licenses/MIT`

je zdarma a s volně dostupnými zdrojovými kódy. Programy se píší v C++, na Androidu tedy prostřednictvím NDK, čímž ztrácí výhodu oproti ostatním frameworkům založených na Javě. Díky podpoře desktopových platforem tento engine sdílí stejnou výhodu paralelního vývoje a ladění jako libGDX.

Cocos2d-x obsahuje abstrakční API pro práci s grafikou, zvuky, sítí, soubory, atd. Obsahuje podporu pro spouštění externích skriptů v jazyce LUA a podporu pro fyzikální enginy Box2D [67] a Chipmunk [68].

Pro tuto knihovnu existuje řada externích vysokoúrovňových nástrojů, např. editor částicových efektů, editor celých úrovní, tilemap editor[53] a další [66]. Tyto externí nástroje nejsou přímou součástí frameworku, jsou často vyvíjeny někým jiným a někdy i zpoplatněny.

Velikost komunity by se dala označit za střední. Na oficiálním fóru cocos2d-x je registrováno něco přes dva tisíce témat, počet uživatelů není zobrazen [69]. Existují však i jiné varianty tohoto enginu, např. cocos2d-iphone[54] či cocos2D[55] s různou dokumentací, tutorialy a komunitou. Pro iPhone byla dokonce vydána kniha [70], která může případnému zájemci osvětlit principy frameworku využitelné i při programování her pro Android. Na tomto frameworku již bylo postaveno na 150 androidích her [65].

2.4.6 AndEngine

AndEngine[56] je další populární herní engine, tentokrát zaměřený pouze na Android. Je licencovaný pod LGPL[57], která by efektivně bránila jeho komerčnímu využití, kdyby autor veřejně tuto licenci nerozšířil tak, aby odvozený produkt nebylo nutné šířit pod tou samou licencí [74]. Engine je tedy dostupný zdarma i se zdrojovými kódy. Zkompilovaná knihovna se přiřadí k projektu a dále se vyvíjí jako jakákoliv jiná běžná androidí aplikace v Javě. Stejně jako cocos2d-x je tento framework zaměřený na tvorbu 2D her.

Ve srovnání s libGDX je tento engine poměrně vysokoúrovňový. Kromě běžného API pro práci s grafikou, zvuky atp. obsahuje vysokoúrovňové rozhraní pro podporu paralax scrolling, což je technika simulující hloubku obrazu ve 2D pomocí různou rychlostí pohybujících se vrstev [75], HUD[58] prvků jako je on-screen joystick [77] či pro zobrazení herního menu [76]. Framework obsahuje fyzikální engine Box2D, jehož implementaci si autor vypůjčil z konkurenčního projektu libGDX [78].

Projekt je stále aktivně ve vývoji [72] a fórum čítá na šest tisíc členů [71]. Skrze tento framework bylo vydáno již téměř 200 her pro Android [73]. Dokumentace je neúplná, ale jsou k dispozici různé tutorialy a spousta ukázkového kódu [79]

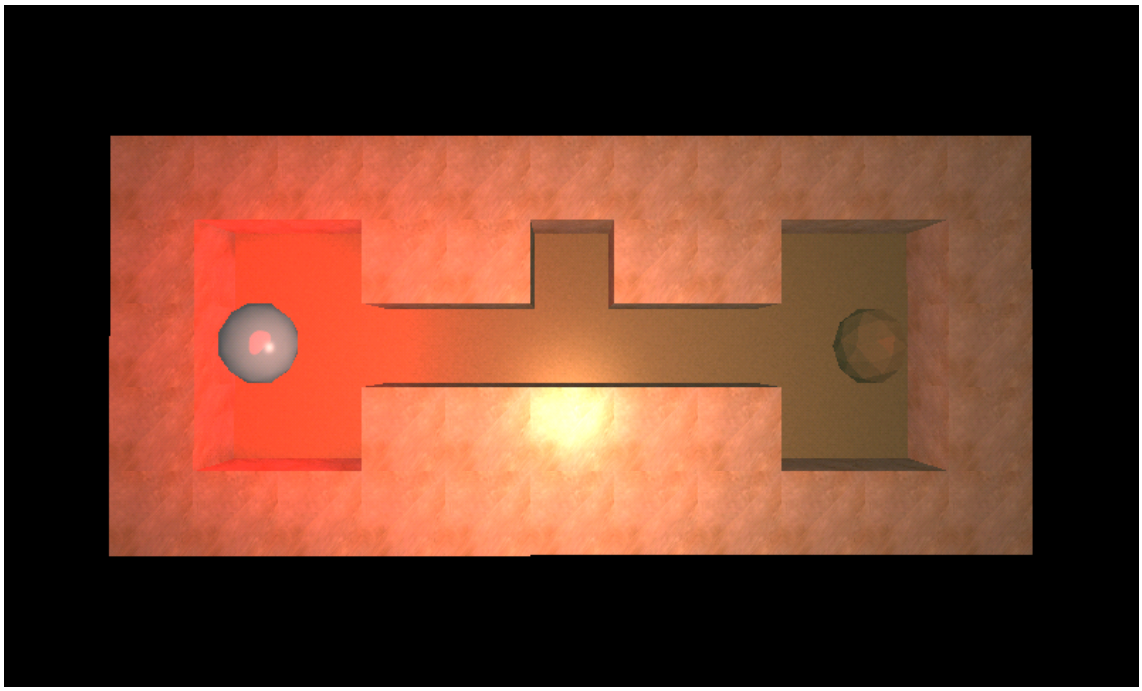
[53] Tilemap vystihuje mapu složenou ze čtvercových dlaždic
[54] `http://www.cocos2d-iphone.org/`
[55] `http://cocos2d.org/`
[56] `http://www.andengine.org/`
[57] `http://www.gnu.org/copyleft/lesser.html`
[58] Zobrazení ovládacích či informačních prvků na obrazovce při hře

Kapitola 3

Realizační část

3.1 První setkání se hrou

Hmatatelným výsledkem této práce je funkční prototyp hry primárně zaměřené na systém Android s přiléhavým názvem *Ball2Hole*. Hra se řadí mezi logické arkády. Herní prostředí je zobrazené ve 3D grafice, plně ozvučené a fyzikálně simulované. Hra je dostatečně flexibilní díky integrovanému systému skriptů.



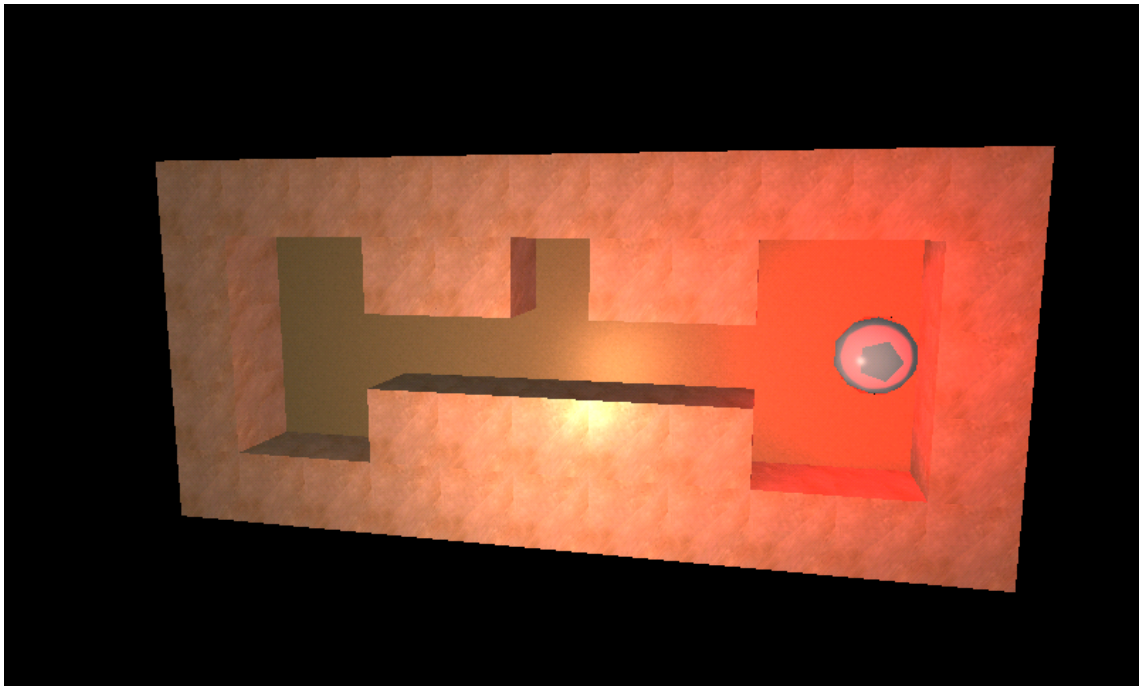
Obrázek 3.1: Ukázka ze hry

Na obrázku 3.1 si lze prohlédnout herní prostředí v jedné z úrovní. Je nutné v této fázi čtenáře upozornit, že vše v dané úrovni je dynamicky vytvořeno na základě aplikačních zdrojů – definičních souborů úrovně, modelů, objektů, zvuků či shaderů. Více o dané problematice je zmíněno v kapitole 3.6.4.

Hlavním principem celé hry je dostat *kuličku* (*ball*) do *díry* (*hole*). Na obrázku 3.1 si lze povšimnout kuličky na levé straně a díry na straně pravé. Na této cestě

mohou vyvstat různé překážky, například v podobě jiných objektů blokujících cestu, jiných necílových děr či jen zkrátka úrovně vykonstruované do sofistikovaného bludiště.

Herní objekty na cestě mohou být ovladatelné hráčem nebo mohou jenom reagovat na gravitaci. Díky tomu, že většina dostupných mobilních zařízení se systémem Android disponuje akcelerometrem, lze hru rozšířit o prvek gravitace. Nakláněním zařízení se herní objekty přesouvají na základě simulované gravitační síly, která směřuje vždy k zemi. Na obrázku 3.2 je vidět kulička správně umístěná v díře a nakloněné prostředí, které reflektuje fyzický stav zařízení.



Obrázek 3.2: Ukázka kuličky v díře a nakloněného prostředí

Hra umožňuje libovolně definovat jakýkoliv herní prvek, lze tak vybudovat prostředí z ledových plání, travnatých krajin či pouští, libovolně měnit tvar i vlastnosti objektů či naskriptovat různé herní cíle. Lze tak vytvořit úrovně, kde se nehraje s kuličkou, ale s jiným objektem či dokonce bez ní[1]. Je dokonce možno mít kuliček více a cílem pak může být dopravit je všechny do děr ve správném pořadí. Hra je skutečně flexibilní a záleží na vkusu a kreativitě herního designera, jaké úrovně vytvoří.

[1]Příkladem pro zvídavého čtenáře může být například úroveň, kde je třeba herní objekty poskládat do určitého tvaru

3.2 Uživatelská příručka

3.2.1 Spuštění

Na přiloženém disku naleznete zdrojové kódy aplikace a rovněž aplikaci připravenou ke spuštění na počítači či systému Android verze 2.2 či vyšší. Ke spuštění aplikace na počítači je nutno mít nainstalovánu Javu ve verzi 1.6. Grafická karta počítače musí podporovat OpenGL alespoň verze 2.0.

Pro spuštění jsou připravené skripty `run.bat` jak pro PC, tak Android, které některé dále popsané kroky provedou automaticky. V případě PC verze jsou k dispozici dva skripty, každý z nich spustí jeden ukázkový level.

Spuštění na PC se pak provede následovně:

```
> java –jar ball2hole.jar <episodeId> <levelId>
```

K otestování je připraveno několik úrovní. Které to jsou je možno vidět v tabulce 3.1. Je potřeba zajistit, aby byly dostupné veškeré zdroje v adresářích `res` a `episodes`. Kromě úrovní v tabulce jsou k dispozici testovací úrovně vytvořené pro testy výkonu. Pro jejich spuštění se podívejte do adresáře `assets/episodes/testPerformance/levels/` v projektu `Ball2Hole-android`, kde uvidíte jejich názvy.

episodeId	**levelId**
testEpisode	BrickAndGrass
testEpisode	Sand

Tabulka 3.1: Testovací úrovně

Pro spuštění aplikace na systému Android je nejprve potřeba nainstalovat soubor `apk`[2] do zařízení. Uvedený postup instalace není jediný možný, avšak je spolehlivě funkční. V zařízení je nejprve nutné aktivovat volbu *Settings>Applications>Development>USB debugging* a volbu *Settings>Applications>Unknown sources*, jak je zobrazeno na obrázku 3.3.

Takto nastavené zařízení připojíme k počítači pomocí USB rozhraní. V počítači musí být nainstalován *Android SDK*[3] a USB ovladače daného zařízení pro `adb`[4]. Správně připojené zařízení ověříme příkazem:

```
> adb devices
List of devices attached
884012B4410423BC        device
```

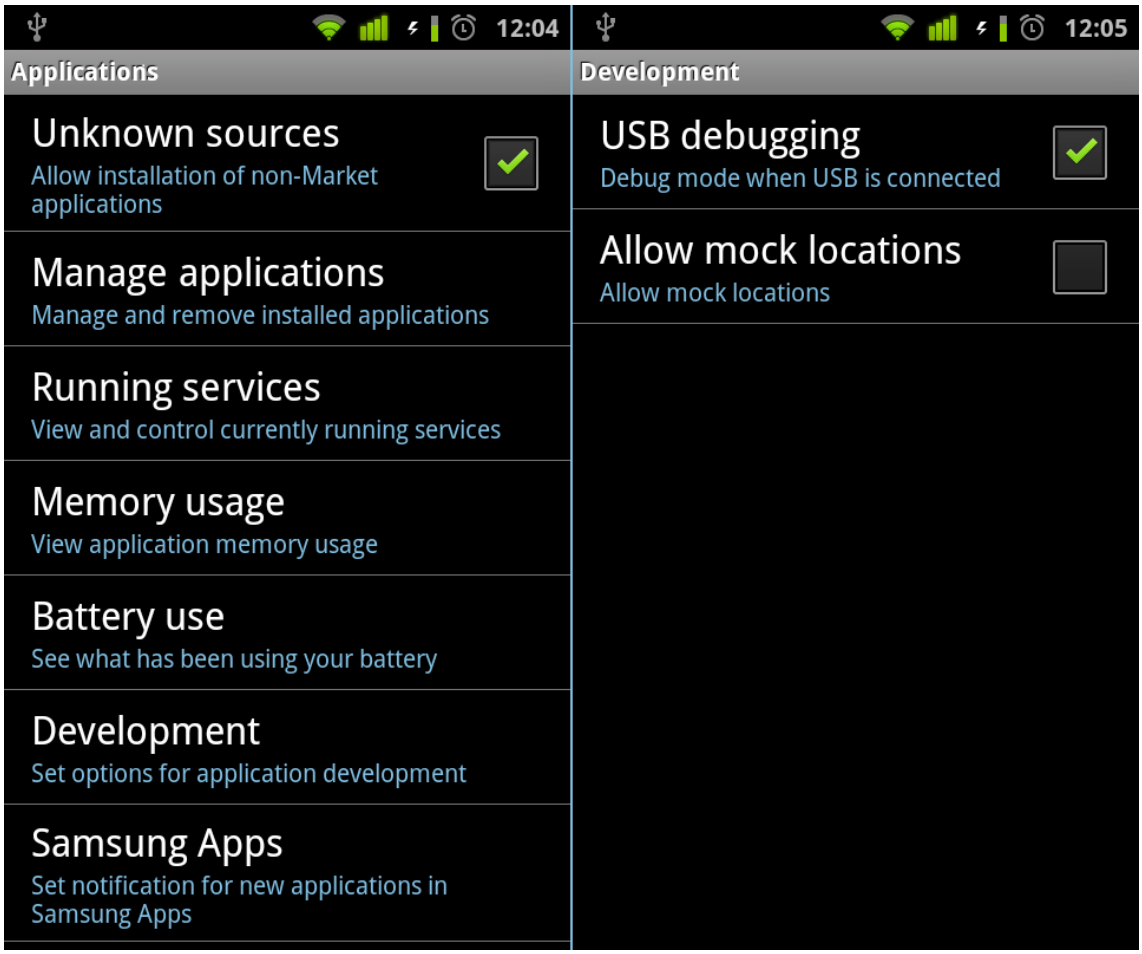
Poté již stačí spustit příkaz:

```
> adb install ball2hole.apk
```

[2] `apk` soubor obsahuje spustitelnou aplikaci na platformě Android včetně všech zdrojů

[3] Android SDK lze stáhnout z `http://developer.android.com/sdk/index.html`

[4] Android Debug Bridge (adb), `http://developer.android.com/guide/developing/tools/adb.html`



Obrázek 3.3: Nastavení zařízení pro instalaci

V zařízení již aplikaci spustíme z menu standardním způsobem. Přivítá nás obrazovka výběru úrovně (viz obrázek 3.4). Zde stačí kliknout na požadovný level a hra se spustí.

Emulátor není možné použít, protože v něm není podporováno OpenGL ES 2.0 [81].

3.2.2 Sestavení

Pokud chce uživatel sám sestavit aplikaci ze zdrojových kódů, je potřeba použít prostředí Eclipse[5] s nainstalovaným ADT pluginem[6]. Celá aplikace se skládá ze 4 základních projektů: `Ball2Hole`, `Ball2Hole-android`, `Ball2Hole-desktop` a `Ball2Hole-scripts`.

Projekt `Ball2Hole` (hlavní projekt) obsahuje hlavní zdrojové soubory aplikace. Další dva projekty obsahují platformě závislé zdrojové soubory. Tyto dva projekty jsou závislé na hlavním a obsahují zejména spouštěč dané platformy. Projekt

[5] `http://download.eclipse.org/eclipse/downloads/drops/R-3.7.1-201109091335/index.php`, použijte verzi 3.7.1

[6] `http://developer.android.com/sdk/eclipse-adt.html`

Obrázek 3.4: Výběr úrovně

`Ball2Hole-android` taktéž v adresáři `assets` obsahuje veškeré zdroje, které jsou vloženy do výsledného `apk` souboru.

Poslední projekt obsahuje standardní skripty a není závislý na žádném z předchozích. Obsahuje veřejná rozhraní zkopírované z hlavního projektu a poté samotný skript.

Pro kompilaci importujte všechny projekty do svého workspace. Dbejte na správnou provázanost projektů a závislosti na knihovnách.

Pro vytvoření spustitelného souboru pro PC exportujte projekt `Ball2Hole-desktop` jako *Runnable JAR file* a ve formuláři zvolte *Package required libraries into the generated JAR*. Ve výsledném souboru nebudou zahrnuty žádné aplikační zdroje, je třeba je zkopírovat do pracovního adresáře z adresáře `Ball2Hole-android/assets`.

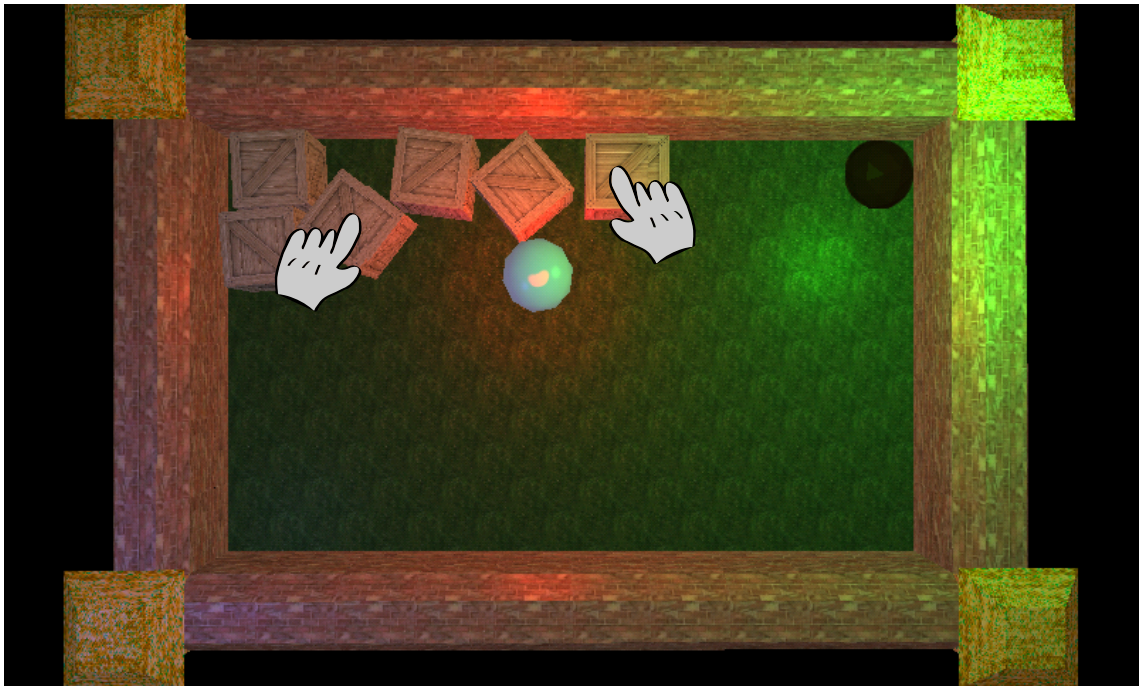
K vytvoření `apk` souboru spustitelného na Androidu je z kontextové nabídky (klikněte pravým tlačítkem) projektu `Ball2Hole-android` zvolit *Android Tools>Export Unsigned Application Package*.

3.2.3 Ovládání

Ovládání na PC se provádí myší, kliknutí simuluje dotyk na obrazovku. Pokud nepustíte tlačítko myši, simuluje se trvalý dotyk prstu. Na PC bohužel není dostupné ovládání multi-touch[7] , ani akcelerometrem. Až na tyto rozdíly je funkčnost aplikace na PC i na Androidu totožná. Následující text bude předpokládat použití fyzického zařízení se systémem Android.

[7]Dotyk více prsty najednou

Některé objekty v úrovni mohou být ovladatelné prstem. Objektů lze ovládat i několik naráz jednoduše použitím více prstů. Pokud objekt prstem chytíte, bude se snažit dosáhnout jeho pozice, pokud mu to ostatní objekty a prostředí dovolí. Viz obrázek 3.5.



Obrázek 3.5: Ovládání prstem

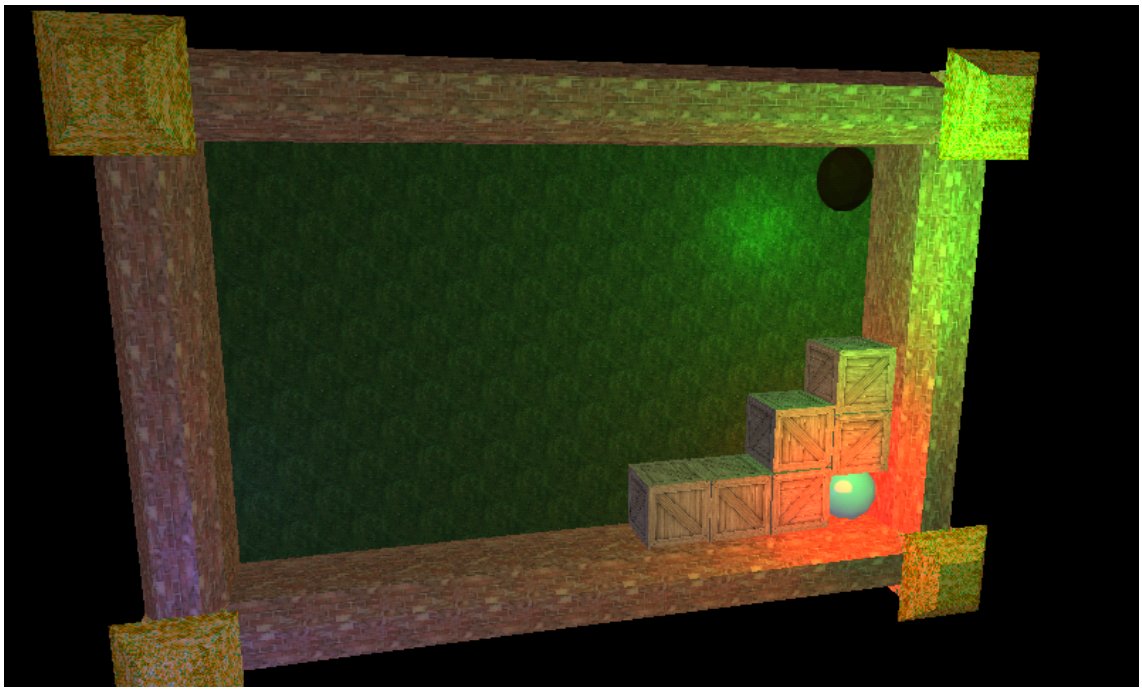
Dalším způsobem ovládání je náklon zařízení. Objekty budou reagovat na přitažlivou sílu a padat k zemi. Viz obrázek 3.6.

V testovacích úrovních je vždy cílem dostat kuličku do díry.

3.3 Technické detaily

Na základě poznatů nabytých v teoretické části byla implementována tato ukázková hra. Aplikace je postavena na platformě Java. Díky použití herního enginu *libGDX*, který bude podrobně probrán v kapitole 3.4, je možno ji spustit jak na desktopovém počítači, tak na zařízení se systémem Android. Dalším důvodem pro volbu tohoto enginu je, že je přehledný, stále aktivně vyvíjený, a jeho poměrně nízkoúrovňové API poskytuje poměrně velký prostor pro manipulaci a programátorskou kreativitu. V neposlední řadě byl vybrán také proto, že s ním má autor ze všech frameworků největší zkušenosti. Další knihovnou, kterou aplikace využívá, je *Simple*[8], která zpracovává XML soubory. Slouží k serializaci a deserializaci XML na JavaBeans. XML je hojně využito k popisu aplikačních zdrojů, např. herních objektů nebo úrovní. XML se pro popis objektů velmi vhodné a knihovna je dostatečně jednoduchá a odlehčená.

[8] `http://simple.sourceforge.net/`



Obrázek 3.6: Reakce na náklon zařízení

Grafická část využívá OpenGL ES 2.0. Tato verze je podporována od Androidu verze 2.2, toto je tedy zároveň minimální verze Androidu, na které lze aplikace spustit. OpenGL ES je podmnožinou OpenGL přizpůsobenou pro potřeby mobilních zařízení. OpenGL ES 2.0 lze na desktopovém počítači simulovat, pokud grafická karta podporuje alespoň OpenGL 2.0. Aplikace pro vykreslování využívá shadery[9]. OpenGL ES 2.0 bylo vybráno k demonstraci *programmable pipeline* za pomoci shaderů a také protože oproti verzi 1.0 či 1.1 dosahuje lepších výkonů [81].

Aplikace běží ve 4 vláknech, která paralelně obsluhují vykreslování, fyziku, uživatelský vstup a skripty. Hra dále demonstruje využití ovládání multitouch a akcelerometru.

Skripty jsou psány v jazyce Java, a jsou pomocí reflexe dynamicky načteny v průběhu hry. Kompilace je provedena dvakrát - jednou do `class` pro desktopovou Javu a podruhé do formátu `dex`, který využívá Dalvik VM[10]. Tato možnost byla zvolena pro svou jednoduchost, není potřeba žádných externích knihoven a funguje poměrně spolehlivě.

Jako vývojové prostředí posloužil Eclipse ve verzi 3.7.1 s nainstalovaným ADT pluginem.

3.4 Stručný popis knihovny libGDX

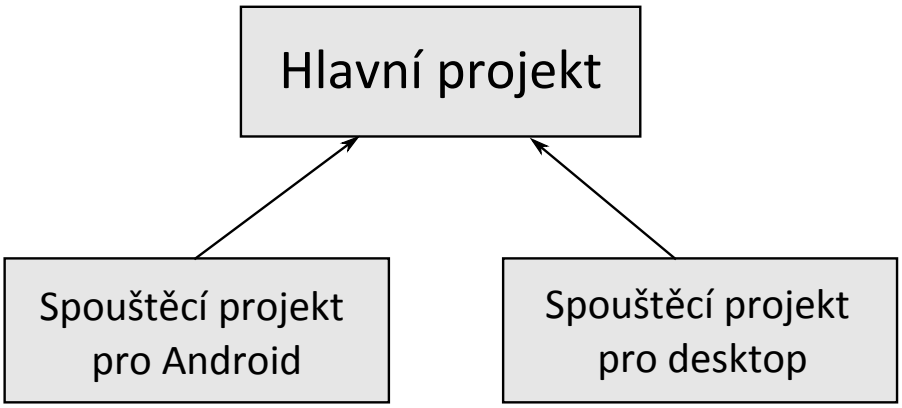
Celá aplikace je postavená na knihovně libGDX. Knihovna je v podstatě napsána dvakrát – jednou pro Android OpenGL ES a jednou pro OpenGL na desktopovém

[9] Shader je nativní program pro GPU

[10] Dalvik je virtuální stroj pro programy napsané v jazyce Java pro platformu Android

počítači. Pro všechny standardní konstrukce Androidu (např. práce se soubory, logem, uživatelskými vstupy a jinými zdroji) jsou implementovány wrappery, které zajistí správnou funkčnost na obou platformách. Programátor poté používá pouze tyto konstrukce. Díky tomuto vhodnému využití rozhraní jsou před programátorem skryty rozdíly obou platforem a hlavní část aplikace může napsat pouze jednou. Jediný rozdíl je ve spouštěcí části, která inicializuje část hlavní a která musí být napsána pro každou platformu zvlášť. Použití knihovny libGDX vynucuje použití určitých konstrukcí a pro porozumění následujících částí je důležité, aby byly stručně uvedeny nejdůležitější koncepty.

Prvním z nich je použití tří samostatných projektů. První z nich je projekt hlavní, kde jsou uvedeny všechny zdrojové kódy samotné aplikace. V tomto projektu se striktně využívá pouze platformně nezávislých konstrukcí knihovny libGDX. Další dva projekty obsahují spouštěče hlavního kódu způsobem standardním pro daný systém. Tyto dva projekty jsou závislé na hlavním, jak je uvedeno na obr. 3.7.



Obrázek 3.7: Závislost projektů

Spouštěč na desktopu je pouze třída s metodou `main`, která inicializuje hlavní kód. Příklad je uveden ve výpise 3.1.

```
1 public class MaiDesktop {
2    public static void main(String[] args) {
3       LwjglApplicationConfiguration config = new
           LwjglApplicationConfiguration();
4       // ... nastaveni konfigurace config
5
6       // inicializace libGDX a kodu aplikace v tride MyGame
7       new LwjglApplication(new MyGame(), config);
8    }
9 }
```

Výpis 3.1: Spouštěč desktopu

Na Androidu je spouštěčem aktivita dědící od `AndroidApplication`. Příklad je ve výpise 3.2.

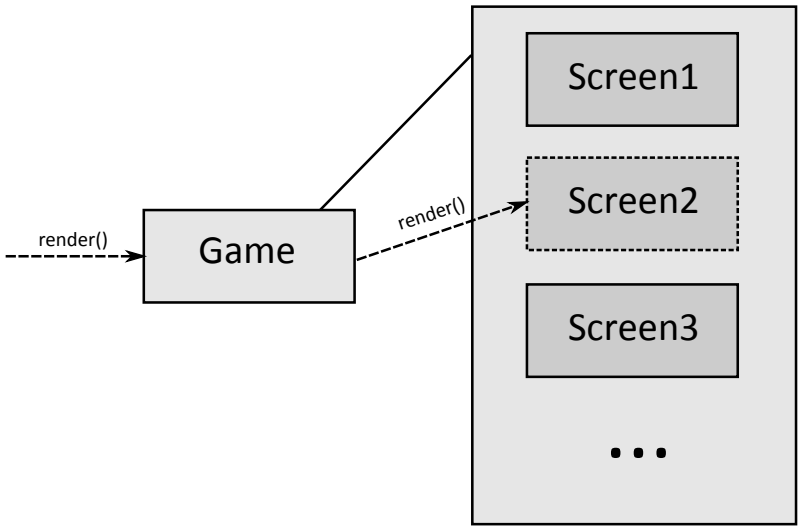
```
1  public class MainAndroid extends AndroidApplication {
2     protected void onCreate(Bundle savedInstanceState) {
3        super.onCreate(savedInstanceState);
4
5        AndroidApplicationConfiguration config = new
            AndroidApplicationConfiguration();
6        // ... nastaveni konfigurace config
7
8        // inicializace libGDX a kodu aplikace v tride MyGame
9        initialize(new MyGame(), config);
10    }
11 }
```

Výpis 3.2: Spouštěč Androidu

Je vidět, že platformě závislého kódu je skutečně minimum.

Po inicializaci jsou do třídy implementující `ApplicationListener` zasílány události o stavu. Jedná se zejména o události `create`, `render`, `pause`, `resume` a `dispose`. Poslední tři se vztahují k životnímu cyklu androidí aplikace a jsou volány při změně aktuální aplikace uživatelem či ukončení. První událostí, kterou aplikace přijme je `create`. Zde se provedou skutečné inicializační kroky samotné aplikace. Poté je periodicky volána událost `render`, kde aplikace vykresluje dle potřeby na obrazovku. V této metodě aplikace definuje pouze svou business logic[11], okolní věci jako je buffering a inicializace řeší knihovna libGDX.

V novějších verzích knihovny libGDX přibyla nová třída `Game` implementující rozhraní `ApplicationListener` a stala se novým základem aplikace. Přinesla zejména možnost použití obrazovek třídy `Screen`, které lze dynamicky měnit. Každá obrazovka je tak specializována na jednu konkrétní činnost – např. hlavní menu, nahrávací obrazovka či obrazovka samotné hry. Předávání události `render` je ukázáno na obr. 3.8



Obrázek 3.8: Předávání událostí do aktuální `Screen`

[11] Bussiness logic je termín označující logiku aplikace vztahující se přímo k problému. Neobsahuje věci kolem jako je logování, řešení přístupových práv, inicializaci apod.

Knihovna za programátora řeší správu textur, shaderů, hudby, zvuků i uživatelského vstupu. Aplikace využívá třídy `Texture`, `Shader`, `Music`, `Sound` či `Input` transparentně a nemusí se starat o záležitosti související se ztrátou kontextu při přepnutí do jiné aplikace, reloading ztracených zdrojů a další.

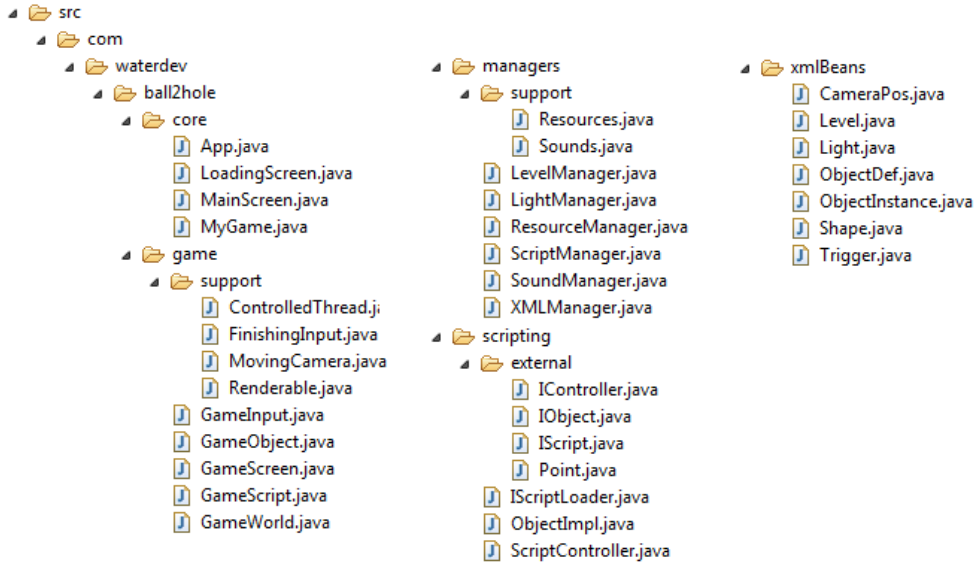
Knihovna dále obsahuje spoustu pomocných tříd, např. `Matrix4`, `Vector3`, třídu `MathUtils` obsahující základní často využívané matematické metody nebo dokonce algoritmy jako je `EarClippingTriangulator` na zpracování polygonů.

Velmi důležitou součástí knihovny libGDX je fyzikální engine *Box2D*. Tento engine je napsán v jazyce C++, knihovna libGDX však pro každou jeho funkci poskytuje Java wrapper, díky kterým je engine Box2D snadno použitelný i v Java prostředí. Ve stručnosti tento engine funguje tak, že programátor definuje fyzikální objekty a jejich vlastnosti a poté periodicky volá funkci `World.step()`, která zajistí správnou simulaci.

3.5 Celkový koncept

3.5.1 Přehled balíků

Hlavní kód aplikace je obsažen v projektu `Ball2Hole`. Všechny zdrojové kódy jsou součástí balíku `com.waterdev.ball2hole`, který je následně dále členěn. Přehled všech balíků a existujících tříd je na obrázku 3.9. Hlavními balíky, které jsou nyní důležité, jsou `core`, `game`, `managers` a `scripting`. Ostatní balíky budou zmíněny později.



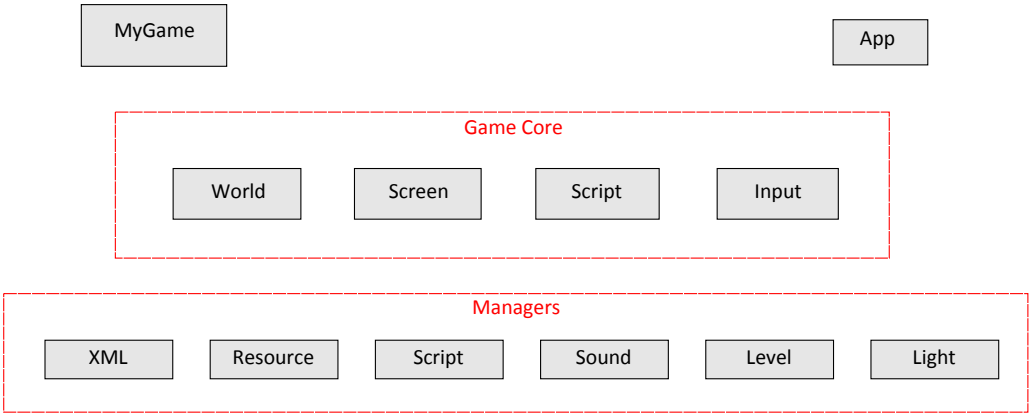
Obrázek 3.9: Přehled všech balíků a tříd

Na obrázku A.1 je zobrazen diagram tříd. Spíše než samotné třídy je třeba pochopit, jak jsou poskládány do jednotlivých modulů a o tom pojednává následující

sekce. V části 3.6 jsou jednotlivé komponenty probrány a je vysvětleno, z jakých tříd se skládají.

3.5.2 Stručný popis modulů

Celá aplikace je poměrně složitá kompozice jednotlivých modulů. Určitou představu o nich dává obrázek 3.10.



Obrázek 3.10: Přehled modulů aplikace

Začněme modulem `MyGame`. Skládá se ze stejnojmenné třídy a jeho úkolem je zajistit základní činnost aplikace a delegovat události do příslušné aktivní `Screen`. Třída `MyGame` dědí od třídy `Game` poskytované knihovnou libGDX a je tedy hlavním prostředníkem mezi knihovnou a aplikací. Dále je zde pomocná třída `App`, která slouží pro uchování statických referencí na důležité objekty v rámci aplikace. Protože provázanost modulů je velká, nebylo by efektivní ani jednoduché předávat si odkazy mezi sebou. K přístupu k instancím jednotlivých modulů proto slouží třída `App`.

Další kategorií je *Game Core*, ve které jsou moduly přímo se vztahující k probíhající hře. Každý z nich obsluhuje právě jedno vlákno. Modul `World` zajišťuje simulaci herního světa skrze engine Box2D a případné kolize. Modul `Screen` vykreslování objektů na obrazovku. `Script` zajišťuje spouštění *triggerů*[12]. Poslední modul `Input` zajišťuje správné zpracování uživatelského vstupu, tedy výběr objektů dotykem.

V kategorii *Managers* jsou podpůrné moduly, které zajišťují služby pro ostatní moduly. Modul `XML` poskytuje funkce pro snadný přístup ke zdrojům v XML. Modul `Resource` spravuje veškeré aplikační zdroje a poskytuje služby pro jejich načtení a vyhledání. Manager `Script`[13] zajišťuje správné načtení zkompilovaných skriptů. Module `Sound` poskytuje služby pro přehrávání zvuků a hudby. Modul `Level` zajišťuje inicializaci úrovně, vytvoření všech herních objektů, včetně jejich fyzikálních částí pro engine Box2D, a dalších věcí souvisejících s inicializací. Poslední modul `Light` spravuje světla ve hře umožňuje předání jejich parametrů do shaderu.

[12] Trigger je oblast, do které když vejde herní objekt, spustí se odpovídající skript
[13] Ačkoliv nese stejné jméno jako `Script` z Game Core, nejedná se o stejný modul

3.5.3 Jak to celé funguje

Pro pochopení základních principů fungování aplikace se podívejme na obrázek A.2. Obrázek sice nevystihuje dokonale realitu, nicméně přehledně ukazuje, jak spolu jednotlivé komponenty komunikují.

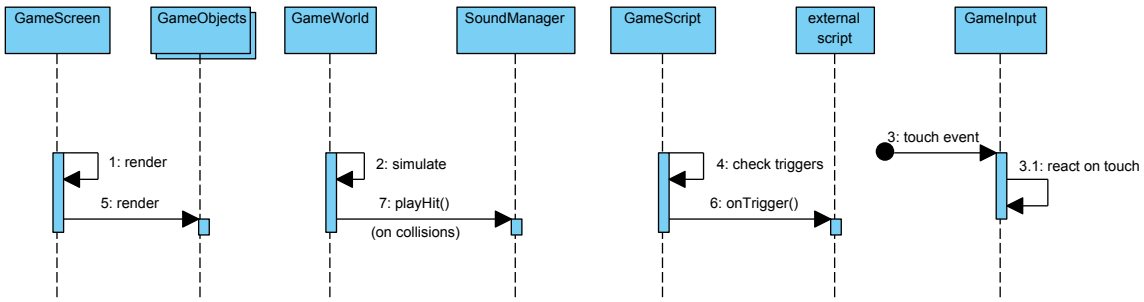
Jako první se při požadavku na spuštění levelu přepne obrazovka na `LoadingScreen`. Ve většině případů je inicializace levelu poměrně rychlá a tak uživatel nahrávací obrazovku ani nepostřehne. Při nahrávání složitějších úrovní nebo na pomalejších zařízení však tato obrazovka dobře poslouží jako indikátor, že se stále něco děje.

V druhém kroku se nahrají zdroje přiřazené dané úrovni. O tom, jak jsou zdroje strukturovány a jak probíhá samotné načítání, pojednává část 3.6.4. Je vidět, že `ResourceManager` spolupracuje s moduly `XMLManager` a `ScriptManager`, které mu pomáhají v načítání.

Když jsou zdroje načteny, provede se inicializace levelu v modulu `LevelManager`. Ten si vyžádá načtené aplikační zdroje a definici samotného levelu. Z nich vytvoří objekty `GameObject`, které slouží jako základní kámen s kterými aplikace pracuje. Jednotlivé instance udržují informace o jejich stavu - reference na *body*[14], informace o pozici, modelu, shaderu, texturách, světlech, triggerech, zvucích a dalších vlastnostech. Nakonec předá do modulu `LightManager` informace o světlech ve scéně a modulu `SoundManager` informace o hudbě, která bude v úrovni hrát.

Když je vše připraveno, inicializují se základní herní moduly z kategorie Game Core: `GameScreen`, `GameWorld`, `GameScript` a `GameInput`. Moduly se samy postarají, ve spolupráci s modulem `LevelManager`, o správnou inicializaci. Když je vše připraveno, spustí se jednotlivá vlákna a aktivní obrazovka se přepne na `GameScreen`.

Průběh samotné hry je znázorněn na obrázku 3.11. Uvedené činnosti se periodicky a souběžně opakují, dokud hra neskončí.



Obrázek 3.11: Sekvenční diagram průběhu hry

Vzhledem k tomu, že v aplikaci běží paralelně čtyři vlákna pracující se stejnými objekty, bylo nutné zavést synchronizační prvky. Synchronizace je zajištěna instancí objektu `ReadWriteLock`, který je součástí běžné Javy. Umožňuje zamykání pro čtení a pro zápis, přičemž zámky pro čtení mohou existovat paralelně, ale zápis má exkluzivní přístup. Použití zamykání pro zápis a čtení na rozdíl od synchronizace pomocí konstrukce `synchronized` zvyšuje výkon aplikace.

[14] *Body* je termín enginu Box2D pro fyzickou schránku objektu

Synchronizace je vztažena na simulaci herního světa, kterou provádí modul `GameWorld`. Tento jediný v podstatě může použít zámek pro zápis. Ostatní moduly zamykají používají pouze zámek pro čtení a pro zápis jen ve výjimečných případech[15].

Nyní by čtenář měl chápat základní principy fungování hry. V dalších částech bude toto porozumění hlouběji rozšířeno.

3.6 Popis komponent

3.6.1 XML

K pochopení následujících částí je důležité vysvětlit, jak aplikace pracuje se zdroji zapsanými ve formátu XML. Ke jejich snadné manipulaci slouží třída `XMLManager`. Tato třída jako jediná přímo využívá knihovnu Simple a všechny dotazy do světa XML směřují přes ni. Knihovna Simple poskytuje velmi jednoduché a zároveň mocné rozhraní pro serializaci a deserializaci XML do JavaBeans a obráceně. V aplikaci je využit pouze převod XML do podoby JavaBeans.

Při definování jednotlivých JavaBeans se využívají anotace, které říkají, jak vypadá protějšek daného prvku v XML. Abychom si dovedli lépe představit, jak takový JavaBean vypadá, podívejme se na výpis 3.3. Ukázka zároveň slouží k demonstraci definice objektu.

```
1  @Default(required=false)
2  public class ObjectDef {
3     @Attribute(required=true)
4     public String id;
5
6     @Attribute(required=false)
7     public String base;
8
9     @Path("graphics")
10    public String modelId;
11
12    @Path("graphics")
13    public String textureId;
14
15    @Path("graphics")
16    public String shaderId;
17
18    @Path("graphics")
19    public Light light;
20
21    @Path("physics")
22    public Shape shape;
23
24    @Path("physics")
25    public Boolean isTouchable;
```

[15] Jediný případ v celé aplikaci zamykání pro zápis je ve funkci `ObjectImpl.moveTo()`, která je volána z externího skriptu. Protože metoda způsobuje viditelnou změnu ve fyzice, je třeba uzamknutí pro zápis.

```
26
27      @Path("physics")
28      public Boolean isStatic;
29
30      @Path("physics")
31      public Float density;
32
33      @Path("physics")
34      public String hitSound;
35
36      public Trigger trigger;
37  }
```

Výpis 3.3: Ukázka JavaBeanu připraveného pro zpracování knihovnou Simple

Odpovídající soubor XML, který může být do takto definovaného JavaBeanu načten je ukázán na výpise 3.4

```
1  <objectDef id="ball">
2      <graphics>
3          <modelId>ball</modelId>
4          <shaderId>default</shaderId>
5          <textureId>earth</textureId>
6      </graphics>
7      <physics>
8          <shape type="Circle">
9              <radius>0.45</radius>
10         </shape>
11         <isTouchable>false</isTouchable>
12         <isStatic>false</isStatic>
13         <density>5</density>
14
15         <hitSound>metalHit</hitSound>
16     </physics>
17 </objectDef>
```

Výpis 3.4: Ukázka definice objektu v XML převeditelného na JavaBean

Pro úplnost je ještě ve výpise 3.5 ukázáno, jak může vypadat definice velice jednoduchého levelu.

```
1  <level id="test">
2      <!-- level definition -->
3      <description>Test level</description>
4      <camera x="5.5" y="3.5" z="12" />
5      <light a="#6f6f6f" ds="#1f1fcc" x="0" y="0" z="5" />
6      <light a="#000000" ds="#1fff11" x="10" y="5" z="3" />
7      <backgroundMusic>carefulNow</backgroundMusic>
8
9      <!-- triggers -->
10     <trigger id="levelTrigger" x="1" y="1">
11         <shape type="Circle">
12             <radius>1</radius>
13         </shape>
14     </trigger>
15
16     <!-- objects -->
17     <object def="shinyBall" id="ball">
```

```
18        <position x="8" y="1" />
19     </object>
20     <!-- ... -->
21
22     <!-- ground -->
23     <object id="hole" def="tile_hole">
24        <position x="10" y="6" />
25     </object>
26     <!-- ... -->
27
28     <!-- walls -->
29     <object def="wall">
30        <position x="0" y="0" />
31     </object>
32     <!-- ... -->
33  </level>
```

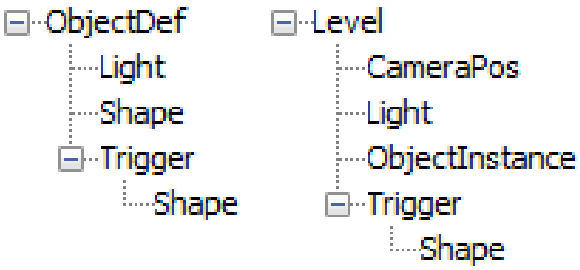
Výpis 3.5: Ukázka XML definice úrovně

Představu o jednoduchosti použití knihovny simple dá výpis 3.6. Ačkoli je příklad ilustrativní a neřeší např. výjimky, je vidět, že s pouze několika řádky kódu lze z prostředí Java pohodlně programovat za použití XML.

```
1 public ObjectDef loadObjectDef(FileHandle handle){
2    Serializer serializer = new Persister();
3    return serializer.read(ObjectDef.class, handle.read());
4 }
```

Výpis 3.6: Příklad načtení JavaBeanu z XML

Třída XMLManager poskytuje dvě funkce: `loadObjectDef()` sloužící k načtení definice objektu a `loadLevel()`, která načte definici úrovně.



Obrázek 3.12: Hierarchie JavaBeans

Přestože může aplikace načíst pouze dvě JavaBeany, `ObjectDef` a `Level`, existují i další, s jejichž pomocí jsou ony dvě hlavní definovány. Vzniká tak hierarchie JavaBean, která je ukázána na obrázku 3.12. V tabulce A.2 je uvedeno, jaké JavaBeany aplikace obsahuje včetně proměnných a jejich typů. Některé JavaBeany obsahují i metody, které dále usnadňují jejich zpracování.

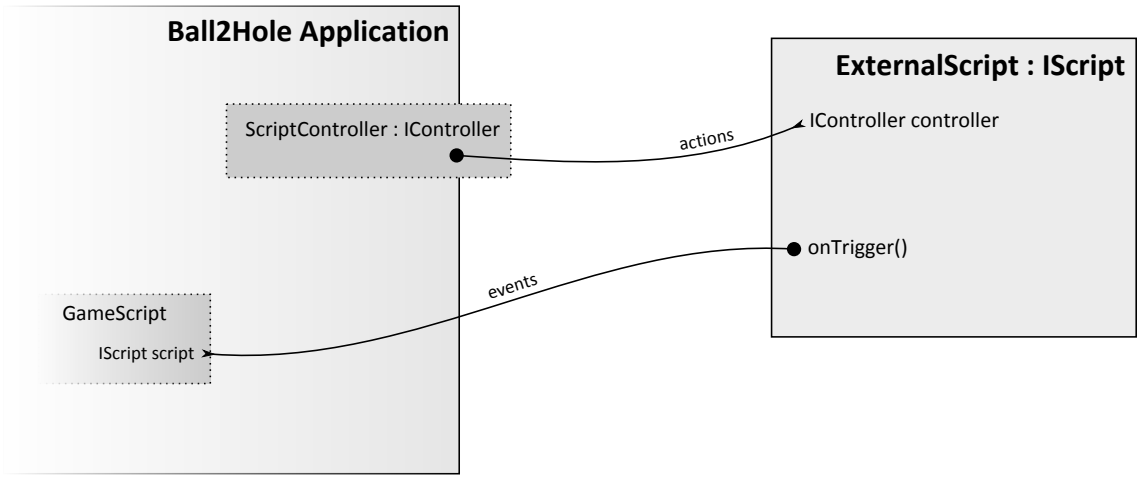
3.6.2 Skripty

Součástí aplikace, která výrazně rozšiřuje její flexibilitu, je systém skriptů. Skripty se píší v Javě a jsou aplikací za běhu dynamicky načítány. Programování v Javě

přináší výhody: Java má obrovskou vývojářskou podporu, programátor se nemusí učit další jazyk a může pracovat v prostředí na které je zvyklý a v neposlední řadě není potřeba žádných dalších speciálních knihoven pro načtení a běh skriptů.

Aplikace obsahuje externí rozhraní a sadu standardních[16] objektů, které jsou publikovány samostatně pro vývojáře skriptů. Díky tomu může programátor skriptů pracovat bez zdrojových kódů celé aplikace. Tyto rozhraní jsou tři: `IController`, `IScript` a `IObject`. V sadě standardních objektů je pouze jeden objekt `Point`, který slouží pro předání bodu na ploše v návratové hodnotě funkce.

Na obrázku 3.13 vidíme propojení skriptu s aplikací. Začněme rozhraním `IScript`. Toto rozhraní je implementováno výsledným skriptem. Díky němu může aplikace transparentně pracovat s jakýmkoliv skriptem. Nejdůležitější metodou tohoto rozhraní je `onTrigger()`, která slouží jako příjemce stejnojmenné události. Ta přichází z modulu `GameScript` v případě spuštěného triggeru[17]. V modulu `GameScript` běží vlákno, které po definovaných časových intervalech vyhodnocuje pozici objektů a případně vyšle událost spuštění triggeru. Tato kontrola je prováděna 10krát za vteřinu.



Obrázek 3.13: Propojení skriptu s aplikací

Jako prostředník pro komunikaci skriptu s aplikací slouží rozhraní `IController`, které je implementováno na straně apliace v třídě `ScriptController`. Obsahuje metody `getObject()`, která podle zadaného `id` najde a vrátí herní objekt. Tento herní objekt je vrácen skrze rozhraní `IObject`. Další metoda, kterou rozhraní `IScript` obsahuje je `showText()`[18], která umožňuje skriptu zobrazit text na obrazovce. Poslední metodou je `win()`, která slouží pro indikaci, že hráč již dosáhl herního cíle a hra končí.

Skript může získat pomocí metody `IController.getObject()` rozhraní

[16] Standardem se v tomto případě myslí vlastní konvence definované pro vývoj skriptů ve hře Ball2Hole

[17] Trigger je spuštěn v případě, že se nějaký objekt (popř. konkrétní objekt) dostane do oblasti triggeru

[18] Poznámka: tato metoda není v aplikaci plně implementována, text se vypíše pouze do konzole (logu)

`IObject` určené k manipulaci s objekty. Toto rozhraní publikuje několik metod: `getId()` vrací id objektu. Metoda `setActive()` umožňuje skriptu nechat objekt „zamrznout.“ Metoda `moveTo()` slouží k manuálnímu přesunu objektu. Metoda `setZPosition()` se užije, pokud je třeba objekt vyzvednout do výšky nebo do potopit do hloubky[19]. Poslední je metoda `getPosition()`, díky které může skript zjistit pozici objektu.

Už s těmito několika metodami lze definovat plnohodnotný skript, který vyhodnotí základní cílové podmínky hry – dostat kuličku do díry. Příklad takového skriptu je na výpisu 3.7.

```
1  public class Default implements IScript {
2     private IController controller;
3     private boolean won = false;
4
5     public void setController(IController controller) {
6        this.controller = controller;
7     }
8
9     public void onTrigger(String triggerId, String objectId) {
10       if(triggerId.equals("hole") && objectId.equals("ball") &&
            !won){
11          won = true;
12
13          IObject ball = controller.getObject("ball");
14          IObject hole = controller.getObject("hole");
15
16          Point holePos = hole.getPosition();
17
18          ball.setActive(false);
19          ball.moveTo(holePos.x, holePos.y);
20          ball.setZPosition(-0.5f);
21
22          controller.win();
23       }
24    }
25 }
```

Výpis 3.7: Defaultní skript

Tento skript je zkompilován a aplikací načten při nahrávání úrovně. Bohužel se formát zkomilovaných java souborů liší pro desktopovou a androidí Javu. Na desktopu probíhá překlad do *.class* souborů, na Androidu jsou to soubory *.dex* obsahující kód pro Dalvik VM. Aby byla zachována možnost aplikaci spouštět na desktopu i na Androidu, bylo zvoleno následující řešení.

Oba dva formáty, *.class* i *.dex*, jsou součástí souboru *.jar*, který je distribuován jako hlavní soubor skriptu. Pro překlad do formátu *.class* se používá standardní postup. Pro překlad do souboru *.dex* se používá skript *dx*, který je součástí Android SDK. Pro snadný překlad je v projektu `Ball2Hole-scripts` připraven build skript, který zajistí správnou kompilaci a připravení výsledného *.jar* souboru.

S dynamickým načítáním *.dex* souboru se však pojí další nepříjemnost – kód je závislý na Androidu. Nelze jej proto využít v hlavním projektu

[19] Výsledný efekt je pouze optický – herní prostředí je dvourozměrné

`Ball2Hole`. K účelu načtení zkompilovaného skriptu byly vytvořeny dvě třídy: `AndroidScriptLoader` a `DesktopScriptLoader`, které jsou umístěny do příslušných projektů `Ball2Hole-android` a `Ball2Hole-desktop`. Obě implementují rozhraní `IScriptLoader` a obsahují metodu `loadScript()`. Každá třída zajistí správné načtení platformě závislého formátu ze souboru skriptu *.jar*. Manager `ScriptManager` zajišťuje správnou inicializaci těchto loaderů v závislosti na platformě, na které aplikace běží. Jeho metoda `getScript()` umožňuje aplikaci jednoduše načíst skript bez starostí ohledně platformy.

3.6.3 Zvuky a hudba

Systém zvuků a hudby je, na rozdíl od ostatních, poměrně jednoduchý. Knihovna libGDX podporuje dva druhy zvuků: `Sound`, který slouží pro krátké zvuky, které mohou znít zároveň ve velkém množství a které jsou pro tento účel kompletně nahrány v paměti a `Music`, který slouží pro postupně streamovaný hudební doprovod a který může jako celek zabírat i několik MB.

V aplikaci se se třídou `Music` pracuje přímo, kdežto pro krátké zvuky slouží třída `Sounds`. Obsahuje seznam zvuků `Sound` spadajících do jedné kategorie[20]. Její metoda `play()` vybere ze seznamu náhodně jeden zvuk, který přehraje. Je-li takto pro jeden zvuk definováno několik variant, odstraňuje se jednotvárnost a zvyšuje se uživatelský zážitek.

Hlavním prostředníkem, skrze kterého aplikace ke zvukům přistupuje, je manager `SoundManager`. Obsahuje následující metody: Metoda `playMusic()` spustí přehrávání zadané hudby. Metoda `stopMusic()` ukončí přehrávání. Metoda `playSound()` jednorázově přehraje konkrétní zvuk. Poslední metoda `playHit()` slouží pro přehrání zvuku kolize mezi dvěma herními objekty.

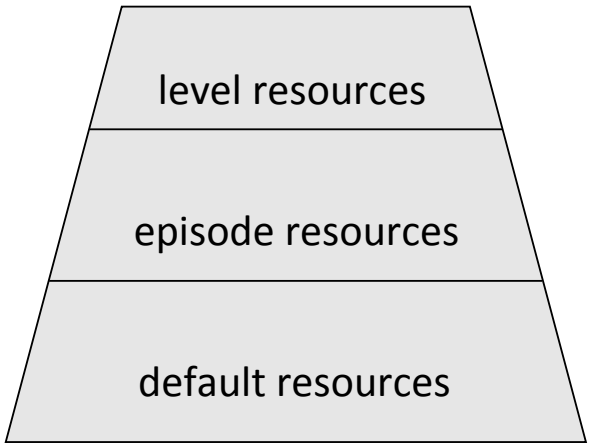
3.6.4 Aplikační zdroje

Aplikace využívá velké množství zdrojů, které jsou distribuovány společně s ní. Aplikačními zdroji se rozumí 3D modely, textury, shadery, skripty, definice objektů, zvuky a hudba. Na platformě Android jsou tyto zdroje distribuovány v souboru `apk` v adresáři `assets`, zatímco na desktopu jsou distribuovány volně v pracovním adresáři aplikace.

Aplikační zdroje jsou rozděleny do tří vrstev, jak je vidět z obrázku 3.14. Ve spodní vrstvě jsou defaultní zdroje platné v celé aplikaci. Ve střední vrstvě jsou zdroje platné pro právě aktivní epizodu a nahoře jsou zdroje dané úrovně.

Vrstvení zdrojů je důležité zejména při jejich vyhledávání. To tudíž probíhá shora dolů a použije se zdroj z první úrovně, ve které je definován. Zdroje úrovně tak mohou překrýt obecnější zdroje epizody nebo defaultní zdroje. Zdroje epizody mohou překrýt zdroje defaultní. Úrovně a epizody se tak mohou lišit a zároveň nevyžadují zásah do zdrojů ostatních epizod a úrovní. Výhodou je mj. snadnější distribuce nové epizody či úrovní.

[20] To může být např. cinknutí kovu nebo klapnutí dřeva



Obrázek 3.14: Tři vrstvy zdrojů

V kódu jednu vrstvu aplikačních zdrojů reprezentuje třída `Resources`. Obsahuje výčet `Type`, který slouží jako parametr funkcí vztahujících se k vyhledávání zdrojů. Výčet obsahuje následující typy zdrojů: `Model`, `Texture`, `Shader`, `ObjectDef`, `Script`, `Sounds` a `Music`. Hlavní součástí je sada `HashMap`, pro každý druh zdroj existuje jedna mapa. Indexace jednotlivých zdrojů je v těchto mapách prováděna skrze jedinečné *id*. Toto *id* je zpravidla odvozeno od názvu souboru, nicméně v některých případech tomu tak není a přesný způsob získávání *id* bude probrán později. Hlavní metodou třídy `Resources` je metoda `getResource()`, která podle požadovaného typu zdroje a jeho *id* daný zdroj nalezne v příslušné mapě a vrátí ho.

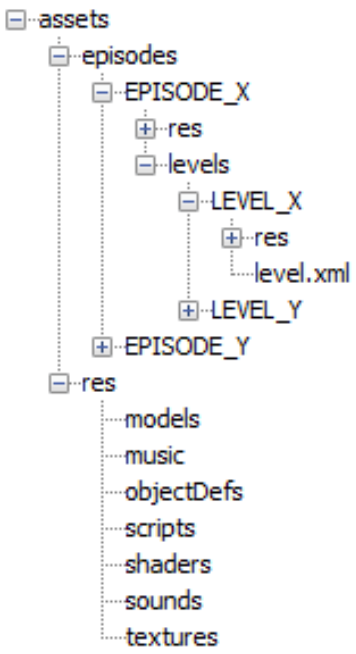
Prostředníkem přístupu ke zdrojům je `ResourceManager`. Obsahuje tři vrstvy zdrojů v podobě tří instancí třídy `Resources`. Jeho metody `loadDefaultResources()`, `loadEpisodeResources()` a `loadLevelResources()` umožňují aplikaci nahrávat příslušné zdroje. Samotné nahrávání probíhá ve funkci `loadResources()`, kde se jednotlivé zdroje vyhledávají v adresářové struktuře a načítají. Adresářová struktura zdrojů je vidět na obrázku 3.15. Je zde vidět, že každá epizoda a každý level může mít vlastní zdroje a stejně tak jsou zde vidět zdroje defaultní. Každý typ zdroje má svůj adresář. Metoda `getResource()` vyhledá postupně v jednotlivých vrstvách zdrojů požadovaný zdroj a vrátí ho.

Inicializace zdrojů je prováděna postupně. Za *id* je považován ve většině případů název souboru bez přípony. Na případné výjimky bude upozorněno.

Nejdříve jsou inicializovány skripty za využití třídy `ScriptManager`. Skripty se nalézají v souborech *.jar*, které obsahují přeložené soubory ve formátu *.class* a *.dex*. Podrobněji o tom pojednává kapitola 3.6.2.

Dále jsou inicializovány zvuky. Název souborů se zvuky se skládá z *id*, jednoho dalšího znaku [0-9a-z] a přípony. Onen speciální znak od sebe odlišuje jednotlivé variace jednoho zvuku. Pro každý zvuk je vytvořena instance `Sound` a všechny zvuky se stejným id jsou umístěny v jedné instanci třídy `Sounds`. Podporované zvukové formáty jsou *mp3*, *ogg* a *wav*.

Následně je inicializována hudba. Podporované zvukové formáty jsou stejné jako u zvuků. Hudba je reprezentována instancemi třídy `Music`.



Obrázek 3.15: Adresářová struktura zdrojů, epizod a úrovní

Dalším inicializovaným zdrojem jsou 3D modely. Ty jsou uloženy v souborech formátu *Wavefront obj* s příponou *.obj*. Tento formát je podporován jak knihovnou libGDX, tak spoustou editačních programů, mj. Blendrem[21]

Poté jsou inicializovány textury. Ty mohou v různých obrazových formátech, např. *png* a *jpg*. U textur platí, že jejich rozlišení by mělo být rovno mocnině dvou. Dle zamýšleného použití a cílového zařízení se zvolí vhodná velikost textury.

Inicializace definic objektů není tak přímočará jako u ostatních zdrojů. Definice objektů totiž od sebe mohou dědit jednotlivé parametry. Tato generičnost umožňuje od již vytvořeného objektu definovat další variace. Tato dědičnost je podporována metodou `ObjectDef.inherit()`, která zajistí převzetí všech parametrů, které nejsou definovány, od rodiče. Protože načtené objekty mohou mít mezi sebou různé závislosti a je třeba inicializovat je ve správném pořadí, odkládá se inicializace objektů s dědičností do zvláštního seznamu. V následující smyčce jsou tyto závislosti vyřešeny a objekty správně inicializovány. Definice objektů jsou uloženy ve formátu XML a jsou načítány do třídy `ObjectDef`. O načítání XML pojednává kapitola 3.6.1. Identifikátor je v případě definic objektů uložen přímo v definici a je načten z XML souboru.

Posledním inicializovaným zdrojem jsou shadery. Shadery jsou programy ve speciálním jazyce a jejich zkompilovaný kód je vykonáván přímo grafickou kartou. Více o shaderech bude součástí sekce 3.6.6. Shadery jsou dvojího typu – vertex shadery a fragment shadery. Jsou uloženy v podobě zdrojového kódu v souborech[22] *.vsh* a *.fsh*. Tyto soubory se vyskytují vždy ve dvojicích a sdílejí svůj název, mimo

[21] Blender je populární open-source software pro tvorbu 3D modelů, `http://www.blender.org/`

[22] Kód shaderů je uložen ve zdrojových souborech, protože zkompilovaný kód se u různých výrobců grafických čipů může lišit

příponu. Tento název je zároveň identifikátorem. Dvojice shaderů je na místě zkompilována do `ShaderProgram`, který je poskytován knihovnou libGDX.

3.6.5 Simulace fyziky

Fyzika je simulována knihovnou Box2D, která je součástí knihovny libGDX. Tento engine umožňuje simulaci fyziky v dvourozměrném prostoru. Použití je poměrně jednoduché. Aplikace vytvoří pro každý fyzikální objekt instanci `Body`, která v sobě nese informace o tvaru, pozici, natočení a hmotnosti objektu, propojení s jinými objekty a další vlastnosti. Obecné vlastnosti objektů jsou definovány v definicích objektů v aplikačních zdrojích. Pozice a natočení každého objektu je definováno v definici úrovně (viz výpisy 3.4 a 3.5 na str. 34). Objekty dělíme na *statické*, které se nepohybují a mezi sebou nekolidují (např. zdi) a dynamické, což jsou běžné objekty, které jsou plně simulovány. Existuje ještě kategorie kinematických objektů, které však v aplikaci nejsou využity. Dále je vytvořena instance třídy `World`, do které jsou všechny fyzikální objekty přiřazeny.

Vytváření objektů se provádí v `LevelManager` metodou `addObject()`, která zajistí správnou inicializaci herního objektu. Samotné vytvoření fyzikální části je provedeno ve funkci `createBody()`.

Při hře je simulace prováděna v modulu `GameWorld()`, jehož vlákno periodicky spouští metodu `World.step()` z knihovny Box2D. Metodě je předáván časový krok, který by v ideálním případě byl konstantní. Časový krok je v aplikaci totožný s reálným časem od posledního volání. Aplikace se snaží docílit frekvence volání metody `World.step()` 60krát za vteřinu. Modul `GameWorld` také obsahuje metodu `beginContact()` která je volána v případě kolize dvou objektů. Metoda zjistí, které objekty to jsou a rychlost vzájemného nárazu a přehraje zvuk kolize skrze `SoundManager.playHit()`.

Tato třída také zajišťuje správnou reakci na náklon zařízení zjišťováním aktuálního stavu akcelerometru a aktualizace gravitace pomocí `World.setGravity()`.

3.6.6 Rendering

Vykreslování je zajištěno třídou `GameScreen`. Při spuštění hry aplikace přepne na tuto obrazovku, což způsobí volání funkce `show()`, která zajistí správné nastavení prostředí OpenGL a kamery. Samotné vykreslování probíhá v metodě `render()`, která je periodicky volána knihovnou libGDX.

V první části metody se pro každý herní objekt volá funkce `GameObject.update()`, která zjistí aktuální polohu a natočení z fyzikálního body a lokálně si ji uloží. Dále se nastavuje kamera. Její pozice je závislá na náklonu zařízení. Kamera se posouvá v rovině rovnoběžné se zemí a míří vždy na střed. Je tak vytvořena iluze náklonu herního prostředí. Kamera je implementována ve třídě `MovingCamera`, která obsahuje metody `moveTo()`, `lookAt()` a `updatePosition()`. Metodou `moveTo()` se nastaví žádaná pozice kamery a v

metodě `updatePosition()` dochází k vyhlazení pohybu tak, aby kamera neskákala chaoticky a trhaně z místa na místo.

V poslední části metody `render()` se pro každý herní objekt volá metoda `GameObject.render()`. Tato metoda zajistí vykreslení samotného objektu. Nejdříve se vypočtou translační a rotační matice nezbytné pro vykreslení. Poté se nastaví textura a zavolá se metoda `LightManager.initShader()`, která nastaví světla do shaderu. Shaderu se dále nastaví patřičné proměnné a nakonec se zavolá metoda `Model.render()`, která vykreslí objekt na obrazovku.

Vysvětlení principů shaderů je součástí kapitoly 2.3.3 na straně 12. Protože je přeložený kód závislý na dané grafické kartě, jsou shadery součástí aplikace v podobě zdrojových kódů. Byly napsány co nejobecněji a pomocí direktiv překladače bylo zajištěno, že jsou přeložitelné jak na mobilním zařízení, tak na desktopu. Defaultní dvojice shaderů v aplikaci zajišťuje osvětlení dle Blin-Phongova osvětlovacího modelu, popsaného např. v [80], a podporuje až 3 světla ve scéně.

3.6.7 Uživatelský vstup

Uživatelský vstup je zpracováván ve třídě `GameInput`, která implementuje rozhraní `InputProcessor`. Třída obsahuje metody `touchDown`, `touchUp` a `touchDragger` a je zaregistrována při spuštění hry jako příjemce událostí vstupu. Metody třídy jsou volány při příslušné události knihovnou libGDX.

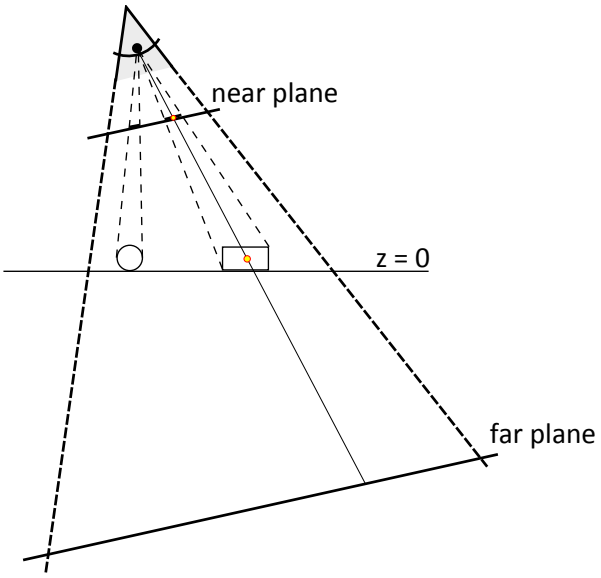
Třída má na starost zásadní úkol – zajistit reakci herních objektů na dotyk uživatele. K určení objektu, kterého se uživatel dotýká se využívá objektu kamery z `GameScreen` a instance `World` z `GameWorld`. Protože kamera umožňuje zjistit pozici bodu v prostoru na své *near plane* a *far plane*[23], lze jednoduchými matematickými postupy zjistit, který objekt leží na přímce definované místem dotyku, oběma rovinami a rovinou herních objektů[24]. Postupu se mj. říká *ray casting* a postup nalezení vybraného objektu je na obrázku 3.16.

Tímto postupem získáme bod v rovině herních objektů, tedy bod v herním světě knihovny Box2D. Na instanci herního světa lze volat metodu `World.QueryAABB()`, která zjistí, který objekt se nachází na daných souřadnicích. Instance herního objektu `GameObject` se získá z reference, kterou s sebou každé fyzikální body nese. Pro svázání objektu s prstem uživatele je použit objekt `MouseJoint`, který je součástí knihovny.

Aplikace reaguje až na pět různých dotyků zároveň, kde omezení plynou spíše z hardwarové a softwarové implementace samotné platformy.

[23] Kamera zobrazuje oblast mezi rovinami near plane a far plane. Na near plane zároveň prováděna projekce herního prostředí pro pozdější rasterizaci.

[24] Nezapomeňme, že ačkoliv je herní prostředí vykreslováno třírozměrné, ve skutečnosti se herní objekty nacházejí jen na jedné rovině



Obrázek 3.16: Ray casting

3.7 Vyhodnocení

3.7.1 Testy na zařízeních

Aplikace byla otestována na několika různých zařízeních a ve verzi pro PC. Aplikaci není možné spustit v emulátoru, protože v něm není podporováno OpenGL ES 2.0 [81]. Pro testy byl využit level *BrickAndGrass* z epizody *testEpisode* Výsledky testů jsou v tabulce 3.2. Výsledky ukazují počet iterací za vteřinu (dále FPS) daného modulu ve formátu minimum/maximum/průměr. Aplikace se snaží dosáhnout 60 FPS ve fyzice, 10 FPS u skriptů a maximálního možného v grafice, zde je za optimum považováno 30 FPS.

Jednotlivé moduly zaznamenávají do logu zařízení údaje o FPS. Ty se dají poté dále automaticky zpracovat skriptem. Metodika testování byla následující:

1. Spusť na daném zařízení příslušnou úroveň
2. Přibližně po 30 vteřinách interakce se hrou tak, aby nedošlo k dosažení cíle, hru ukonči
3. Z logu zařízení vyjmi příslušné údaje o FPS

Testy PC verze proběhly na stroji s operačním systémem Microsoft Windows Vista 64bit, a následujícím hardwarovým vybavením: Intel Core2 4300 @ 1.8GHz, 5GB RAM, NVIDIA GeForce 9600 GSO 512. Verze na PC slouží především k poskytnutí referenčních výsledků. Aplikace byla primárně vyvíjena pro Android, ale PC verze nám dává představu o ideálním stavu. Dle očekávání je PC verze skutečně ve vedoucí pozici: plynule zobrazuje grafiku, simuluje fyziku a spouští skripty.

Na zařízení Samsung Galaxy S, vybaveným čipy CPU 1 GHz Cortex-A8, GPU PowerVR SGX540 a pamětí RAM 512 MB [82], trpí nejvíce grafika, která vykazuje

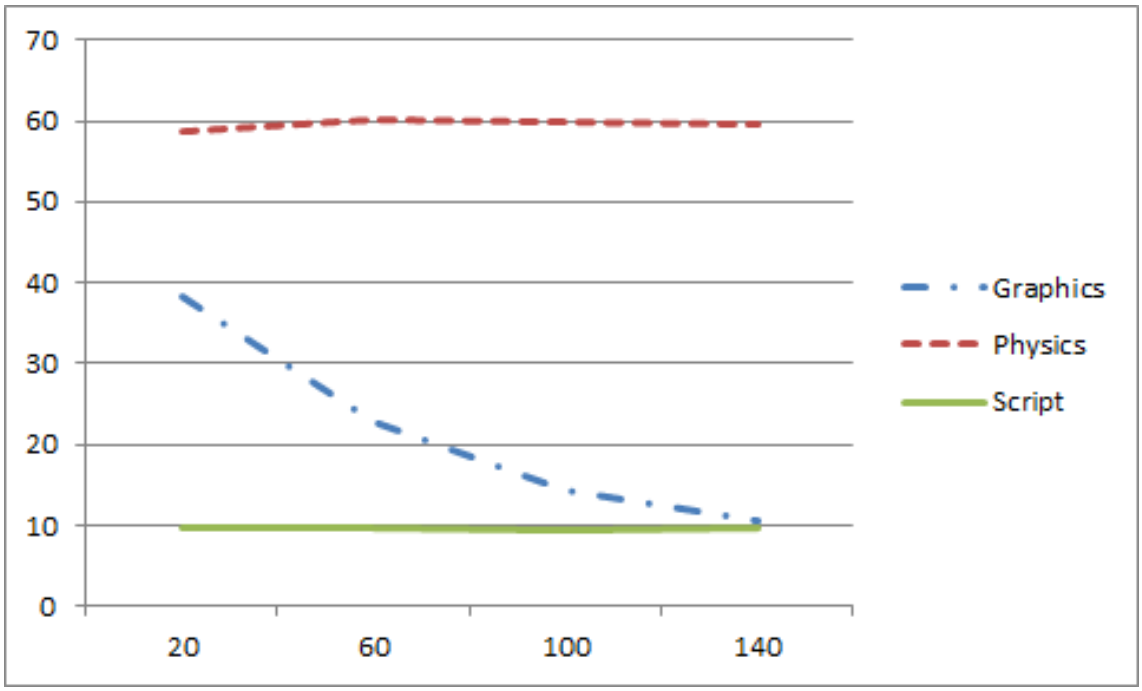
	Grafika	*Fyzika*	*Skripty*
Samsung Galaxy S	6.0/16.0/13.5	50.0/60.0/57.0	9.6/10.0/9.9
Samsung Galay Mini	0.6/6.0/3.0	3.6/11.1/6.3	2.09/7.0/4.3
HTC Desire	2.2/3.9/3.0	49.8/60.0/56.5	9.6/10.0/9.7
PC verze	39.0/63.0/60.0	46.0/62.5/59.0	9.9/10.0/9.9

Tabulka 3.2: Výsledky testů

nízkou hodnotu FPS, což může zásadním způsobem ovlivnit hratelnost. Systém skriptů a fyziky bez problémů funguje.

Na slabší variantě, zařízení Samsung Galaxy Mini, která je vybavená čipy CPU 600 MHz ARMv6, GPU Adreno 200 a jen 384 MB paměti RAM [84] je hra naprosto nehratelná. Spustí se, ale nestíhá ani grafika, tak ani fyzika a skripty. Hodnoty jsou hluboko pod úrovní přijatelné hratelnosti.

Na zařízení HTC Desire je situace obdobná jako u zařízení Samsung Galaxy S. Zařízení je vybaveno procesorem 1 GHz Scorpion, grafickým čipem Adreno 200 a 576 MB paměti RAM. Výpočetní síla CPU bez problému stačí ke zvládnutí simulace fyziky a běh skriptů, ale slabý grafický čip, stejně jako v případě Samsung Galaxy Mini, nedostačuje svým výkonem k plynulému běhu.



Obrázek 3.17: Výsledky testů grafické části

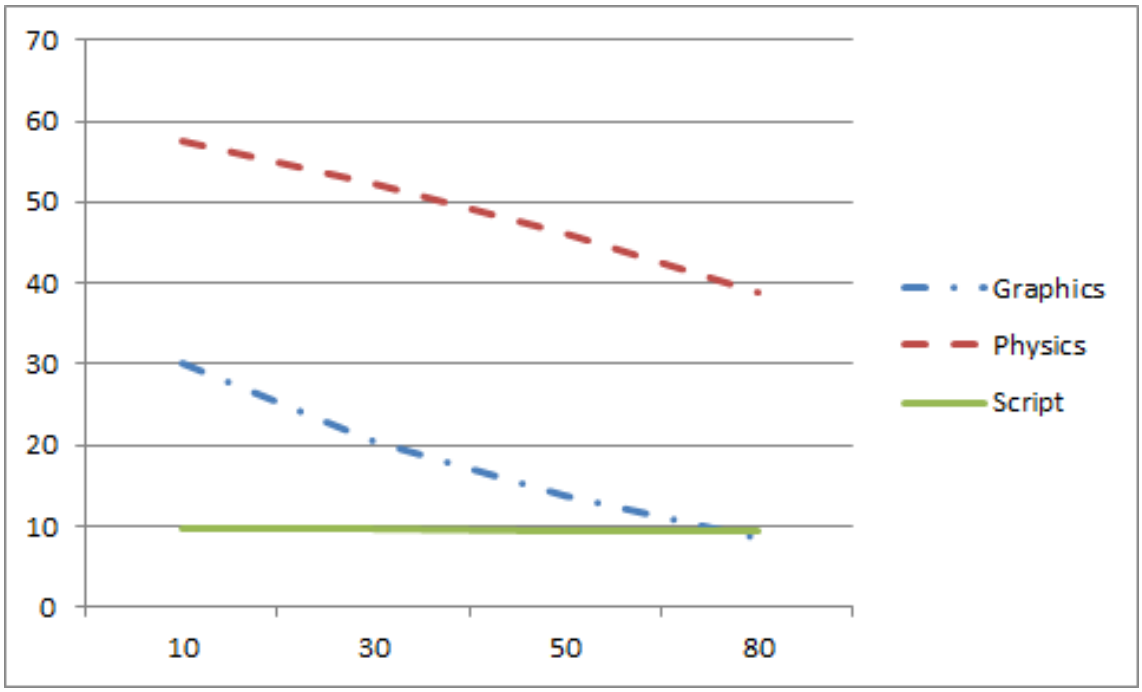
3.7.2 Testy výkonu jednotlivých modulů

Bylo vybráno referenční zařízení Samsung Galaxy S, na kterém se prováděly další testy. Ty mají za cíl zjistit, jak závisí rychlost jednotlivých modulů na počtu určitých entit. Ty jsou určeny tak, aby ovlivňovaly co nejvíce daný modul a co nejméně moduly ostatní. Zároveň byla sledována závislost jednotlivých modulů mezi sebou. Pro tyto testy byly navrženy speciální úrovně a jsou umístěny do epizody *testPerformance*. Číslo na konci názvu úrovně udává počet simulovaných entit.

Prvním testem byl výkon grafické části. K tomuto účelu slouží úrovně *testGraphics**, které obsahují pouze grafické prvky v počtech: 20, 60, 100 a 140. Průměrný level bude obsahovat kolem 60 až 80 grafických prvků, takže nemá smysl simulovat vyšší hodnoty. Výsledky jsou v grafu 3.17.

Vidíme, že již při 40 objektech výkon na testovacím zařízení spadl pod optimální hranici 30 FPS a přibližně při 70 objektech pod hranici plynulosti 20 FPS. Výkon zařízení zřejmě nedostačuje k hraní průměrných úrovní, pokud by nebylo přistoupeno na výkonové ústupky a optimalizace. Dále je vidět, že moduly fyziky a skriptů jsou na grafické části v podstatě nezávislé. To je způsobeno zřejmě oddělením jednotlivých vláken, přičemž grafické vlákno pracuje s GPU a zbylé dvě s CPU.

Dalším testem byl výkon fyziky. Aby nedocházelo k ovlivnění grafické části vykreslováním objektů, byla grafická část potlačena na minimum – objekty jsou pouze dvourozměrné jednoduché textury. Nebyl použit ani standardní shader, pouze shader vykreslující texturu. Tento test byl prováděn na úrovních *testPhysics**, které obsahují dynamické fyzikální objekty v počtech: 10, 30, 50 a 80. Průměrná úroveň může obsahovat kolem 20-30 dynamických fyzikálních objektů. Výsledky jsou v grafu 3.18.

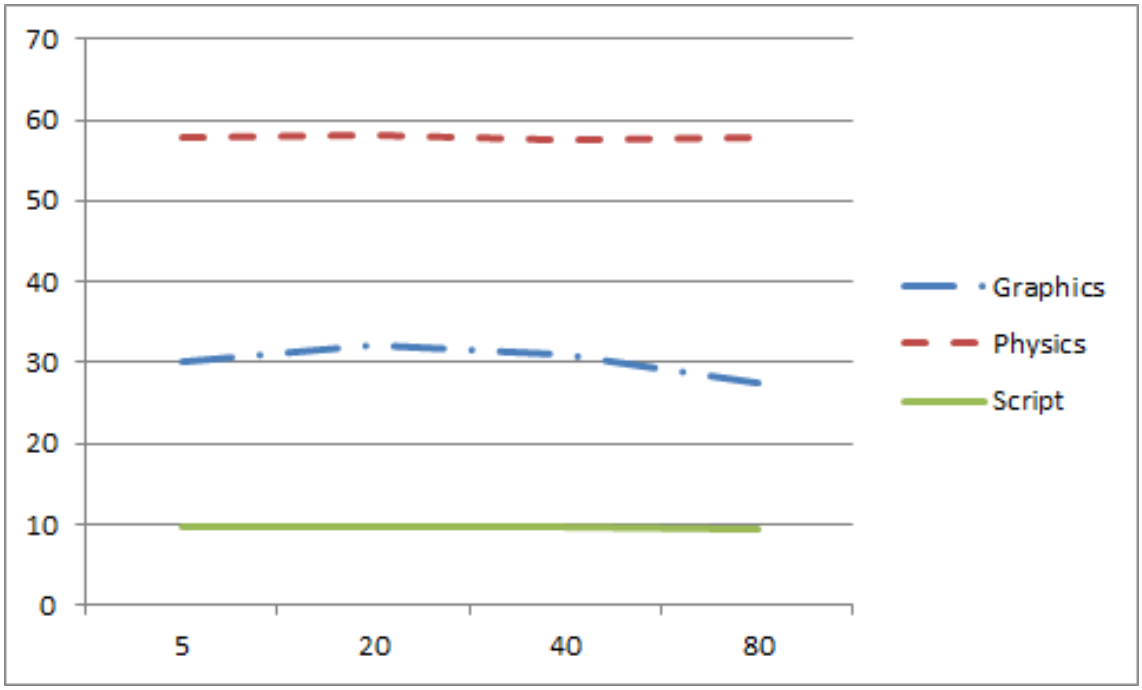


Obrázek 3.18: Výsledky testů fyzikální části

Je patrné, že výkon i při velkých množstvích objektů trpí jen málo. Hranice 40 FPS, která je však dostatečná a stále ještě vysoko nad hranicí nehratelnosti, byla překonána až při téměř 80 fyzykálních objektech. Výkon fyzikálního enginu, který je závislý na CPU, dostačuje k optimálnímu hraní.

Dále si povšimněme, že křivka výkonu grafické části klesá podobným tempem jako křivka fyzikální části. Mohlo by se zdát, že je grafická část částí fyzikální ovlivněna. To bylo vyvráceno dalším testem, kdy se pouze upravila definice pokusného objektu tak, aby nebyl fyzikálně simulován. Grafiký výkon zůstal stejný a pokles je tedy, přes všechny snahy, stále ovlivněn zobrazováním objektů.

Posledním testem byl test modulu skriptů. Ten byl otestován vložením množství triggerů do úrovní *testScript**, konkrétně v množstvích 5, 20, 40 a 80. Průměrné množství triggerů v úrovni může být kolem 1-10, hodnoty jsou tedy dostatečně naddimenzovány. Testuje se v podstatě vyhledávání objektů, které mohou trigger spustit, a případné volání skriptu. K testům byl použit defaultní skript, který v příslušném kontextu neplní žádnou funkci. Resp. ověřuje zda je spuštěný trigger `hole` a objekt `ball`, k čemuž nikdy nedojde. Výsledky jsou v grafu 3.19.



Obrázek 3.19: Výsledky testů skriptovací části

Navzdory očekávání zůstává výkon počtem triggerů neovlivněn. Toto skvělé zjištění znamená, že se případný designer v podstatě nemusí obávat ztrátám výkonu při rozmisťování triggerů v úrovni. Vidíme také, že ostatní moduly jsou po výkonové stránce též nedotčeny.

3.7.3 Shrnutí

Na provedených testech je vidět, že úzkým hrdlem celé aplikace je grafika. Za předpokladu, že je výkon zařízení dostatečný na straně CPU, což u většiny zařízení

je[25], je celková hratelnost ovlivněna pouze výkonem GPU. Na testovacím zařízení Samsung Galaxy S s grafickým čipem PowerVR SGX540 nastává tento případ. Výkon CPU je dostatečný, ale výkon GPU nikoliv. Můžeme usuzovat, že u zařízení s lepším grafickým čipem, a ty jsou už momentálně na trhu, bude tato otázka bezpředmětná a hra naprosto hratelná.

[25]Pomineme zařízení Samsung Galay Mini, které je svými specifikacemi v kategorii low-end

Kapitola 4

Závěr

Diplomová práce prezentovala možnosti tvorby her na platformě Android. Ukazuje se, že hry pro tuto platformu jsou schopny generovat nezanedbatelný zisk. Stále však převládají hry určené pro krátkodobé zabavení na cestách či před spaním. Hry, jaké známe z PC či konzolí, snad přijdou až s masivním rozšířením tabletů.

Byla popsána problematika ovládání, grafiky, fyziky a skriptů pro tuto platformu z obecného i implementačního hlediska. Platforma je dostatečně zralá na tvorbu kvalitních her. Dále práce prezentovala a srovnala několik dostupných herních frameworků. Byl ukázán jeden komerční a několik dalších enginů poskytovaných zdarma. V přehledu bylo uvedeno množství odkazů na příslušné zdroje, z nichž se případný zájemce může o příslušných knihovnách dozvědět více.

Na základě nabytých poznatků byl vybrán herní framework libGDX s fyzikální knihovnou Box2D a na jejich základě byla implementována 3D grafická interaktivní hra. Tato hra je součástí práce ve formě zdrojových kódů i přeložených spustitelných balíků a je k dispozici na přiloženém médiu. Výslednou hru je možné spustit na většině nyní dostupných mobilních zařízeních. Při testech bylo zjištěno, že běh komponent závislých na CPU, tedy fyzika a skripty, je na zařízeních z roku 2010 dostatečně plynulý. Grafická část však na testovacích zařízeních dosahuje v průměru pod 20 FPS. Na trhu jsou však již nyní rozšířeny zařízení po grafické stránce mnohem výkonnější než ty, které byly k dispozici pro testování. Z toho se dá usuzovat, že hra je na nich plně hratelná a plynulá.

Přehled zkratek a termínů

API	Application Programming Interface, aplikační rozhraní
aplikační zdroje	Soubory obsahující definice modelů, objektů, zvuky, shadery apod.
body	Termín enginu Box2D, označuje fyzickou schránku objektu
CPU	Central Processor Unit, procesor
Game Core	Skupina aplikačních modulů tvořící jádro běžící hry
GPU	Graphics Processor Unit, grafický čip
GUI	Graphical User Interface, grafické uživatelské rozhraní
HUD	Head-up display, zobrazení ovládacích a informačních prvků na obrazovce
id	identifikační řetězec
IDE	Integrated development environment, vývojové prostředí
LOD	Level of Detail, bližší objekty zobrazují detailněji než vzdálenější
Managers	Skupina podpůrných modulů, poskytujících služby ostatním
UI	User Interface, uživatelské rozhraní

Použité zdroje a literatura

[1] K, Thimmarayaswamy and Dsouza, Mary M. and Varaprasad, G. (2011), *Low power techniques for an android based phone*, ACM SIGARCH Computer Architecture News (Vol. 39, Iss. 2, May 2011)

[2] *Android Licenses* [online, cit. 19.4.2012], `http://source.android.com/source/licenses.html`

[3] DealExtreme, *Cheap Android Phones for Sale* [online, cit. 19.4.2012], `http://www.dealextreme.com/c/android-phones-526`

[4] Digitaldesi.com, *Kindle Fire Review* [online, cit. 19.4.2012], `http://digitaldesi.com/`

[5] *Samsung Galaxy S II – Specification* [online, cit. 13.3.2012], `http://www.samsung.com/global/microsite/galaxys2/html/specification.html`

[6] Rejthar, J. (2010), *Programování aplikací s využitím API Google Android*, bakalářská práce na Fakultě aplikovaných věd ZČU. Vedoucí práce Ing. L. Pešička.

[7] Murphy, M. *Beginning Android 2*. 1. vyd. Apress, 2010. ISBN 1430226293

[8] Gartner Inc. (2011), *Gartner Says 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011, a 19 Percent Increase Year-on-Year* [online, cit. 18.4.2012], `http://www.gartner.com/it/page.jsp?id=1689814`

[9] *Comparison of Android devices* [online, cit. 20.4.2012], `http://en.wikipedia.org/wiki/Comparison_of_Android_devices`

[10] Xu, Qiang and Erman, Jeffrey and Gerber, Alexandre and Mao, Zhuoqing and Pang, Jeffrey and Venkataraman, Shobha (2011), *Identifying Diverse Usage Behaviors of Smartphone Apps*, Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, pp. 329–344, New York, USA

[11] Android Developers, *Screen Sizes and Densities* [online, cit. 20.4.2012], `http://developer.android.com/resources/dashboard/screens.html`

[12] GeekWire, *Tablets are the new game consoles, and other game industry predictions for 2012* [online, cit. 20.4.2012], http://www.geekwire.com/2012/video-games-2012-predictions-tablets-cloud/

[13] Techcrunch.com, *Angry Birds On Android Projected To Generate $1 Million Per Month In Advertising* [online, cit. 20.4.2012], http://techcrunch.com/2010/12/03/angry-birds-android-1-million-advertising/

[14] Prensky, M. *Fun, Play and Games: What Makes Games Engaging*. V knize Prensky, M. *Digital Game-Based Learning*. kapitola 5, 1. vyd. McGraw-Hill, 2000, ISBN 0071363440

[15] Android Developer, *Graphics* [online, cit. 20.4.2012], http://developer.android.com/guide/topics/graphics/index.html

[16] Google Play, *Cut the Rope* [online, cit. 20.4.2012], https://play.google.com/store/apps/details?id=com.zeptolab.ctr.paid

[17] Examiner.com *'Gun Bros' finds success as a free game for the iPhone and Android*, http://www.examiner.com/article/gun-bros-finds-success-as-a-free-game-for-the-iphone-and-android

[18] The Khronos Group Inc. (2009), *The OpenGL ES Shading Language* [online, cit. 14.3.2012], http://www.khronos.org/files/opengles_shading_language.pdf

[19] Shreiner, D. *OpenGL Programming Guide*. 7. vyd. Addison Wesley, 2009, ISBN 0321773039

[20] Munshi, A., Ginsburg, D., Shreiner, D. *OpenGL ES 2.0 Programming Guide*, 1. vyd. Addison-Wesley, 2008. ISBN 0321502795

[21] Astle, D., Durnil, D. *OpenGL ES Game Development*. 1. vyd. Thomson Course Technology, 2004. ISBN 1592003702

[22] Android Developers Blog *Making Sense of Multitouch* [online, cit. 20.4.2012], http://android-developers.blogspot.com/2010/06/making-sense-of-multitouch.html

[23] Android Developers *Application Resources: Providing Resources* [online, cit. 20.4.2012], http://developer.android.com/guide/topics/resources/providing-resources.html

[24] Varanese, A. *Game scripting mastery*. 1. vyd. Premier Press, 2003. ISBN 1931841578

[25] Wikipedia *NWScript* [online, cit. 23.4.2012], http://en.wikipedia.org/wiki/NWScript

[26] Open Source at Google, *Introducing Android Scripting Environment* [online, cit. 22.4.2012], http://google-opensource.blogspot.com/2009/06/introducing-android-scripting.html

[27] *ASE is now SL4A* [online, cit. 22.4.2012], http://www.damonkohler.com/2010/07/ase-is-now-sl4a.html

[28] *About LUA* [online, cit. 22.4.2012], http://www.lua.org/about.html

[29] *PhysX Downloads* [online, cit. 22.4.2012], http://developer.nvidia.com/physx-downloads

[30] *PhysX Tools* [online, cit. 26.4.2012], http://developer.nvidia.com/physx-tools

[31] *Blender – Bullet Physics* [online, cit. 26.4.2012], http://www.blender.org/development/release-logs/blender-240/bullet-physics/

[32] *Choosing a 2D Physics Game Engine* [online, cit. 23.4.2012], http://www.gandogames.com/2010/05/choosing-a-2d-physics-engine/

[33] Stackoverflow.com *Are there any decent physics engines for Android?* [online, cit. 23.4.2012], http://stackoverflow.com/questions/1034253/are-there-any-decent-physics-engines-for-android

[34] *Box2D – About* [online, cit. 23.4.2012], http://box2d.org/about/

[35] *Chipmunk Physics* [online, cit. 23.4.2012], http://chipmunk-physics.net/

[36] *JBox2D: A Java Physics Engine* [online, cit. 23.4.2012], http://www.jbox2d.org/

[37] *Chipmunk for Java* [online, cit. 23.4.2012], https://github.com/johang/chipmunk-for-java

[38] *Havok Sales* [online, cit. 23.4.2012], http://havok.com/sales

[39] *Bullet Physics – Downloads* [online, cit. 23.4.2012], http://code.google.com/p/bullet/downloads/list

[40] *JBullet – Java port of Bullet Physics Library* [online, cit. 23.4.2012], http://jbullet.advel.cz/

[41] *Facebook SDK for Android* [online, cit. 23.4.2012], https://github.com/facebook/facebook-android-sdk/

[42] *Facebook Android Tutorial* [online, cit. 23.4.2012], https://developers.facebook.com/docs/mobile/android/build/

[43] *Twitter Developer Documentation* [online, cit. 26.4.2012], https://dev.twitter.com/docs

[44] *Is there any Twitter API SDK for Android?* [online, cit. 26.4.2012], http://stackoverflow.com/questions/4571352/is-there-any-twitter-api-sdk-for-android

[45] *Replica Island Developement Diary* [online, cit. 14.3.2012], http://replicaisland.blogspot.com

[46] *Unity – Publishing* [online, cit. 24.4.2012], http://www.gamasutra.com/php-bin/news_index.php?story=27558

[47] Unity Community, *Best programming language in Unity* [online, cit. 24.4.2012], http://forum.unity3d.com/threads/43150-Best-programming-language-in-Unity

[48] Unity Community, *Home* [online, cit. 24.4.2012], http://forum.unity3d.com/

[49] Unity Answers, *Index* [online, cit. 24.4.2012], http://answers.unity3d.com/index.html

[50] Unity3D.com, *Live Online Training Courses* [online, cit. 24.4.2012], http://unity3d.com/support/online-training/

[51] Unity Blog, *London Unity Usergroup 10* [online, cit. 24.4.2012], http://blogs.unity3d.com/2012/04/02/london-unity-usergroup-10/

[52] Unity Community, *Games released with Unity Android* [online, cit. 24.4.2012], http://forum.unity3d.com/threads/59697-Games-released-with-Unity-Android

[53] *jPCT Changes* [online, cit. 26.4.2012], http://www.jpct.net/changes.html

[54] *jPCT Projects* [online, cit. 26.4.2012], http://www.jpct.net/projects.html

[55] *jPCT Forum* [online, cit. 26.4.2012],
http://www.jpct.net/forum2/

[56] *jPCT Forum* [online, cit. 26.4.2012],
http://www.jpct.net/wiki/index.php/Main_Page

[57] *jPCT Physics* [online, cit. 26.4.2012],
http://www.jpct.net/wiki/index.php/Physics

[58] *libGDX Features* [online, cit. 25.4.2012],
http://libgdx.badlogicgames.com/features.php

[59] *libGDX People* [online, cit. 25.4.2012],
http://code.google.com/p/libgdx/people/list

[60] *libGDX Commited Changes* [online, cit. 25.4.2012],
http://code.google.com/p/libgdx/source/list

[61] *Tons of new libGDX games!* [online, cit. 25.4.2012],
http://www.badlogicgames.com/wordpress/?p=1949

[62] *libGDX games round-up #2* [online, cit. 25.4.2012],
http://www.badlogicgames.com/wordpress/?p=2009

[63] *Badlogic games Forum* [online, cit. 25.4.2012],
http://www.badlogicgames.com/forum/

[64] *Platforms supported by cocos2d-x* [online, cit. 25.4.2012],
http://www.cocos2d-x.org/projects/cocos2d-x/wiki/Platforms_supported_by_cocos2d-x

[65] *Cocos2d-x Android Games* [online, cit. 25.4.2012],
http://www.cocos2d-x.org/projects/cocos2d-x/apps/all?p=8

[66] *Editors for cocos2d-x Texture, Tilemap, Particle, Action, Level etc* [online, cit. 25.4.2012],
http://www.cocos2d-x.org/projects/cocos2d-x/wiki/Editors_for_cocos2d-x_TextureTilemapParticleActionLevel_etc

[67] *Cocos2d-x: Adding Box2D to your Project* [online, cit. 25.4.2012],
http://www.gmtdev.com/blog/2011/08/18/cocos2d-x-adding-box2d-to-your-project/

[68] *Adding chipmunk to new project* [online, cit. 25.4.2012],
http://www.cocos2d-x.org/boards/10/topics/586

[69] *Cocos2d-x Forums* [online, cit. 25.4.2012],
http://www.cocos2d-x.org/projects/cocos2d-x/boards

[70] Itterheim, S. *Learn IPhone and IPad Cocos2d Game Development*, 1. vyd., Apress, 2010. ISBN 1430233036

[71] *Cocos2d-x Forums* [online, cit. 25.4.2012], http://www.andengine.org/forums/

[72] *AndEngine Commit History* [online, cit. 25.4.2012], https://github.com/nicolasgramlich/AndEngine/commits/GLES2

[73] *AndEngineWiki List Of Apps And Games* [online, cit. 25.4.2012], http://wiki.andengine.org/List_of_Apps_and_Games

[74] *AndEngine and the LGPL – clarification!* [online, cit. 25.4.2012], http://www.andengine.org/blog/2010/11/andengine-and-the-lgpl-clarification/

[75] *andengineexamples - AutoParallaxBackgroundExample.java* [online, cit. 25.4.2012], http://code.google.com/p/andengineexamples/source/browse/src/org/anddev/andengine/examples/AutoParallaxBackgroundExample.java

[76] *andengineexamples - MenuExample.java* [online, cit. 25.4.2012], http://code.google.com/p/andengineexamples/source/browse/src/org/anddev/andengine/examples/MenuExample.java

[77] *andengineexamples - AnalogOnScreenControlsExample.java* [online, cit. 25.4.2012], http://code.google.com/p/andengineexamples/source/browse/src/org/anddev/andengine/examples/AnalogOnScreenControlsExample.java

[78] *AndEngine – Fully Featured Box2D Physics Wrapper* [online, cit. 25.4.2012], http://www.andengine.org/blog/2010/07/andengine-fully-featured-box2d-physics-wrapper/

[79] *AndEngine - Examples* [online, cit. 25.4.2012], http://code.google.com/p/andengineexamples/

[80] *Blin-Phongovo stínování* [online, cit. 18.4.2012], http://en.wikipedia.org/wiki/Blinn-Phong_shading_model

[81] *Android Developers – OpenGL* [online, cit. 14.3.2012], http://developer.android.com/guide/topics/graphics/opengl.html

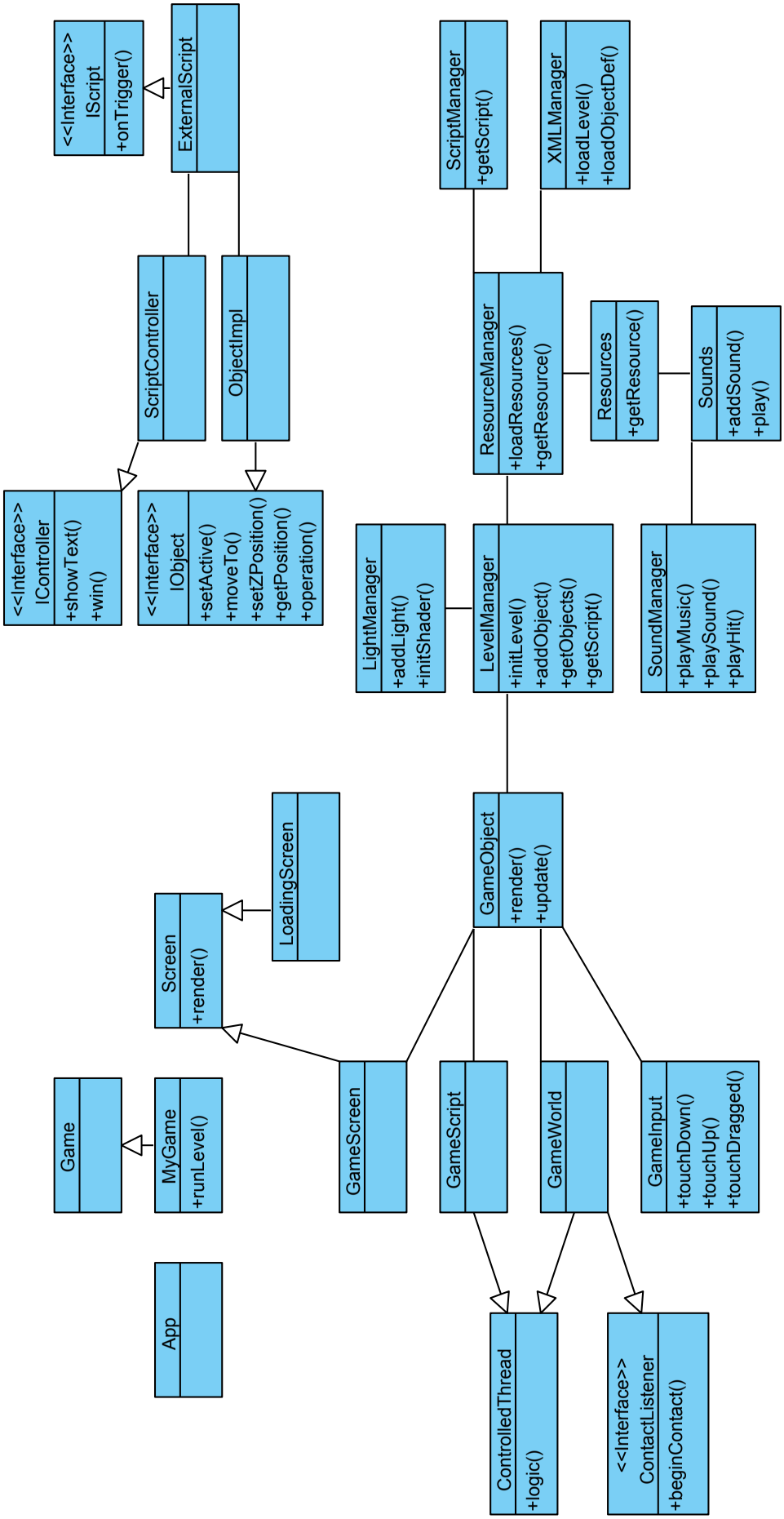
[82] *Samsung I9000 Galaxy S* [online, cit. 30.4.2012], http://www.gsmarena.com/samsung_i9000_galaxy_s-3115.php

[83] *Samsung Galaxy Mini S5570* [online, cit. 30.4.2012], http://www.gsmarena.com/samsung_galaxy_mini_s5570-3725.php

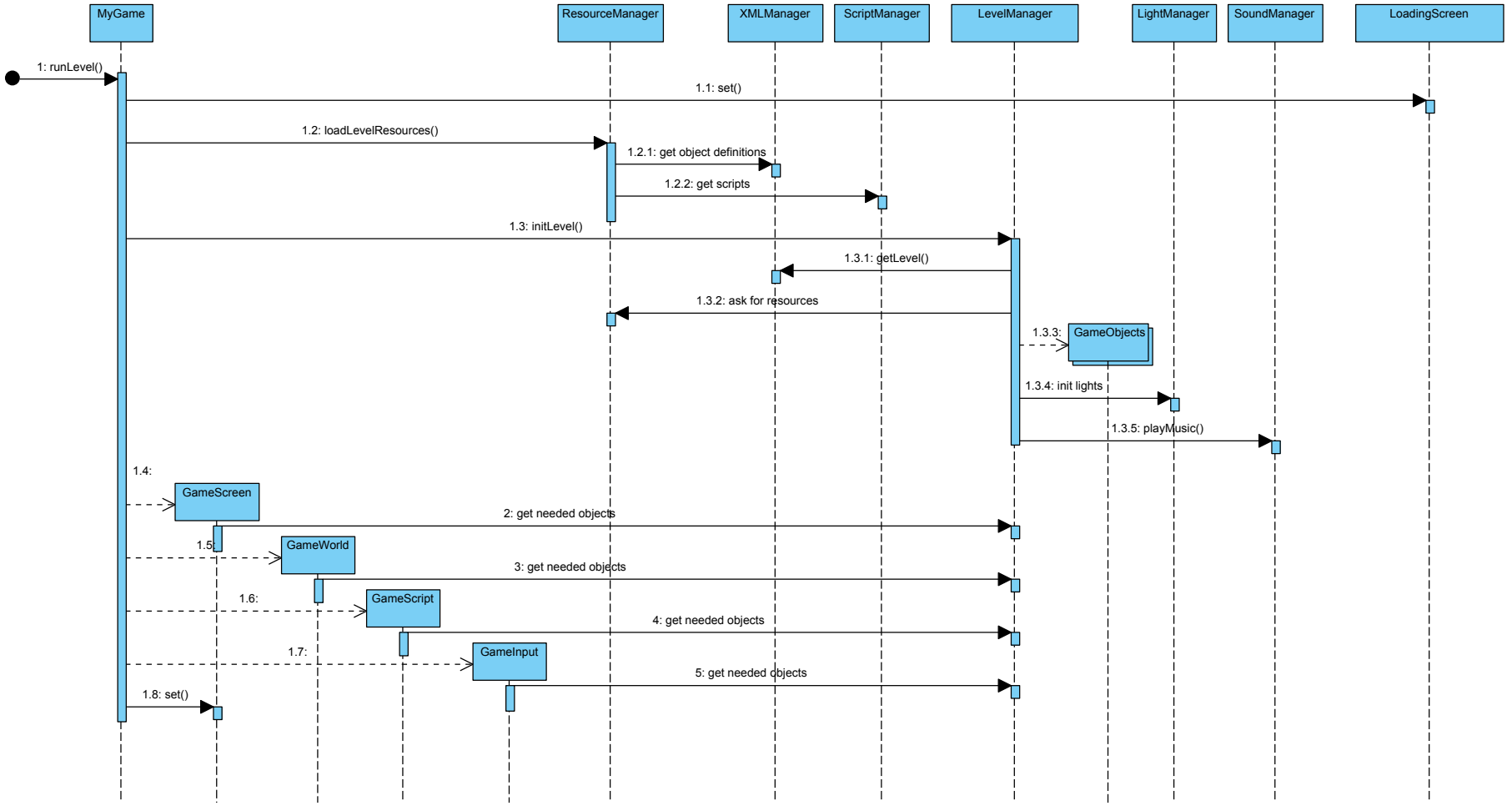
[84] *HTC Desire* [online, cit. 30.4.2012],
`http://www.gsmarena.com/htc_desire-3077.php`

Příloha A

Diagramy a tabulky



Obrázek A.1: Diagram tříd



Obrázek A.2: Sekvenční diagram akce Spuštění levelu

	Poporované platformy	Typ	*Programovací jazyk*	*Fyzikální engine*	*Komunita*	*Podpora*	*Cena*	*Poznámka*
Unity	Android, iPhone, Win*, ...	3D	.NET, UnityScript, Boo	PhysX	obrovská	ano	$400	engine + IDE
jPCT	Android	3D	Java	žádný	malá	ne	zdarma	–
libGDX	Android, Win*	2D/3D	Java	Box2D	střední	ne	zdarma	nízkoúrovňový
Cocos2D	Android, iPhone, Win*, ...	2D	C++	Box2D, Chipmunk	střední	ne	zdarma	–
AndEngine	Android	2D	Java	Box2D	velká	ne	zdarma	–

Tabulka A.1: Přehled herních frameworků

*Microsoft Windows

ObjectDef		**Level**		**Light**		**Shape**		**Trigger**		**CameraPos**		**ObjectInstance**	
String	id	String	id	String	a†	ShapeType	type	String	id	float	x	String	id
String	base	String	id	String	ds‡	Float	radius	Shape	shape	float	y	String	def
String	modelId	String	description	float	x	float[][]	pointList	float	x	float	z	float	x
String	textureId	String	script	float	y			float	y			float	y
String	shaderId	Light*	lights	float	z			String	reactsOn			float	angle
Light	light	ObjectInstance*	objects										
Shape	shape	CameraPos	camera										
Boolean	isTouchable	Trigger*	triggers										
Boolean	isStatic	String	backgroundMusic										
Float	density												
String	hitSound												
Trigger	trigger												

Tabulka A.2: Přehled JavaBean

*Proměnné jsou typu List
†ambient
‡diffuse a specular