Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Mobilní hra typu "rage game" na platformě Android

Plzeň 2015 Lukáš Mařík (A12B0122P)

Prohlášení

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

V Plzni dne 18. června 2015

Lukáš Mařík

Poděkování

Rád bych touto cestou poděkoval vedoucímu své bakalářské práce panu Ing. Ladislavu Pešičkovi za vstřícný přístup, odbornou konzultaci, metodické vedení, které se staly podkladem pro vypracování této práce.

Abstract

Abstrakt

Tato bakalářská práce se zabývá programováním herních aplikací do chytrých mobilních telefonů, které běží na platformě Android. Prozkoumává jednotlivé možné herní typy a jejich představitele v telefonním světě. Dále se zaměřuje herní typ zvaný „Rage Game“, popisuje jednotlivé prvky, které musí tento typ splňovat, a ukazuje několik jejich představitelů v herním světě. V následující řadě se dozvíte jakým způsoben a jakými prostředky se naimplementují jednotlivé herní prvky (herní smyčka, animace, kolize atd.). Poté se můžete podívat na implementaci mé vlastní hry typu „Rage Game“ pro platformu Android.

Abstract

This paper is based on programming game applications to the smart phones, which are running on android. It search individually possible types of games and their representative in the Word of smart phones. Further it focus on types of games called „rage game“ , it describes simple elements , which must this fulfil this type and it shows lot of their representative the a game Word. Next you will know in which way and which means of game elements are implementing (game loop, animation, collision etc.) Then you can look on implementation of my own game type „ Rage Game“ for platform Android.

Obsah

1 Úvod

Úkolem mé práce je vytvořit mobilní hru typu „rage game“, která bude spustitelná na platformě Android. Hry by měla být spustitelná a hratelná na různých moderních telefonech s různou verzí operačního systému Android. První úkolem tedy bude se seznámit s platformou Android a s programováním her pro tuto platformu.

Vytvořená hra má mít příjemnou 2D grafiku a k ovládání by měly sloužit pohybové a dotykové senzory. Hra by měla mít více než dvacet úrovní. Postupně se bude v úrovních zvyšovat obtížnost (těžší překážky, soupeři, délka mapy).

Vybral jsem si naprogramovat klasickou 2D plošinovou arkádu. Uživatel tedy bude ovládat jednu postavu, která se v dané úrovni bude muset dostat do cíle. Pomocí několika protivníků a různých překážek, kteří budou v dané úrovni rozmístěny tak, aby splňovaly prvky pro hru daného typu „rage game“.

2 Herní svět

Dnešní herní svět obsahuje již mnoho her, které se rozdělují do několika obecných žánrových skupin. Jednotlivé žánry obsahují mnoho podžánrů, které jsem charakterizoval v následujících kapitolách.

2.1 Typy her

2.1.1 Adventura:

Je typ hry, kde hlavní hrdina (hráč) řeší rozličné úkoly. Hra je většinou postavena na složitém a propleteném příběhu. Hlavní hrdina většinou postupuje příběhem a řeší jednotlivé logické úkoly. Jen málokdy je připojena akce v podobě souboje.

- Textové adventury a grafické textové adventury:

Starší typy her se v dnešní době již nevyskytují. Hráčovi bylo vypisováno dění hry do konsole a poté hráč pomocí příkazů určoval další průběh hry. Mezi nejznámější firmy, které se zaměřovaly na tento žánr, patří česká firma *Golden Triangle*. Jedna z jejích nejznámějších textovek, na motivy románů J. R. R. Tolkiena, byla hra *Belegost*. Ve světě Androidů se tyto hry nevyskytují.

- Point and Click adventura:

Hra již obsahuje pestré grafické rozhraní pro uživatele. Ovládá se pouze kurzorem (myší), kde se klikáním na grafické předměty udává příkaz, který náše postava (avatar) poté programově vykoná. Jedna z nejznámějších českých „Point and Click“ adventur je *Polda (1,2,3,4,5,6)*. Existuje i mnoho typů těchto her pro Android: (např.: *Leisure Suit Larry: Reloaded*, *The Lost City*).

- 3D adventury:

Hra obsahuje klasické ovládání pohybu a akcí. Dále se mohou dělit na tzv. akční 3D adventury, které v sobě nemají pouze řešení logických úkolů a postupování příběhem, ale navíc mají i několik akčních situací v příběhu. Mezi

nejznámější akční 3D adventury bych zařadil určitě sérii *Tomb Raider/Larra Croft*. Herní svět Androidu obsahuje velké množství takových her.

2.1.2 Akční hry:

V dnešním herním světě, kde převládá 3D grafika rozlišujeme dva základní typy těchto her a to jsou:

- **FPS (First Person Shooter)**: Střílečka z první osoby, kde kamera je postavena na úrovni očí naší postavy. Hráč na obrazovce vidí ze své postavy pouze ruku se zbraní.

- **TPS (Third Person Shooter)**: Střílečka z tzv. třetí osoby, kde kamera je postavena mimo hlavního hrdinu, přičemž se kamera pohybuje za zády hlavního hrdiny.

Svět Androidu obsahuje mnoho obou typů her. Spousta známých počítačových her se dočkala také svého předělání pro Android: např.: *Grand theft auto*, *May Payne*, *Duke Nuke* a spousta dalších.

2.1.3 Arkáda:

Tento typ her je založen na jednoduchém příběhu. Hry většinou obsahují několik úrovní se zvyšující se obtížností. Některé z těchto her mohou mít i časově omezená kola. Hry jsou založeny na šikovnosti a rychlosti uživatele. Dále se rozdělují na:

- Bojové hry.
- Plošinové hry.
- Sportovní hry.
- Závodní hry.
- Logické hry.

Za nejznámější arkádovou hru, jak počítačovou i mobilní, lze považovat sérii her *Super Mario*.

2.1.4 Strategie:

V těchto hrách většinou ovládáme skupinu vojsk nebo osadníků a stavíme infrastrukturu naší osady. Existuje několik podžánrů:

- **Tahové strategie:** Hra je rozdělena na kola, kde v jednom kole hraje jeden z hráčů (s omezeným počtem pohybů a staveb). Nejznámější hra jak pro PC, tak pro Android je sága *Heroes of Might and Magic*.

- **Realtimové strategie:** Průběh hry není rozdělen na kola, ale probíhají tu veškeré akce hráčů a objektů v reálném čase (např.: série *Warcraft*, *StarCraft*). Pro Android se vyskytuje i několik her tohoto žánru jako například *Autumn Dynasty*. K tomuto typu strategie lze přiřadit další podžánr, který je velice oblíben v mobilním světě a tímto žánrem je tzv. **Tower Defense**. Na určených cestách se zde protivníci pohybují z bodu A do bodu B a uživatel má za úkol stavět objekty zneškodňující tyto protivníky. Během následujících levelů se hodnota protivníkova zdraví zvětšuje a uživatel má možnost upgradu svých věží.

- **Budovatelské strategie:** Hráč se zde soustředí spíše na budování a ekonomiku nežli na boj. Ve většině případů systém boje v těchto hrách ani zabudovaný není. Hráč se zde může stát starostou města, majitelem zoo parku nebo zábavného parku a dokonce se může stát i bohem starající se o své věřící na planetě Zemi. Pro platformu Android si můžeme najít nespočet her tohoto typu. Mezi nejznámější jak mobilní tak i počítačovou je hra *SimCity*. První díl této veleúspěšné herní série vyšel již roku 1989. Jednou z neznámějších dílů pro iPhone je *SimCity IPhone(3000)*, který vznikl kolem roku 2008. Také Android se dočkal svého zpracování a to roku 2010 hrou *SimCity Deluxe*.

2.1.5 Simulátory:

V simulátorech si může každý uživatel v pohodlí domova vyzkoušet řídit skoro všechny dopravní prostředky.

Simulátory v dnešní době nemusí být jenom pro nějaký dopravní prostředek, ale existují i simulátory kde se uživatel může stát kozou, koněm, psem a dalšími zvířaty.

V obchodě GooglePlay si můžeme v podkategorii Simulátory najít simulátor pro Android, který chceme.

2.1.6 Hra na hrdiny:

Známější ve zkratce RPG (Role playing game), kde hráč je přenesen do otevřeného virtuálního světa. Hráč zde hraje za svojí postavu (avatara), které zvyšuje pomocí nejrůznějších úkolů její herní level. S herními zkušenostmi (XP) se zvyšují schopnosti naší postavy. Tyto schopnosti se dají také zvyšovat pomocí aktuálního brnění nebo zbraní jaké má naše postava na sobě. Brnění, zbraně a různé doplňky hráč nachází putováním po virtuálním světě a nebo je dostává z úkolů.

Tento typ se dá rozdělit to tří odvětví:

-**Akční RPG:** Neboli tzv. „diablovky“, kde tento název je od první hry tohoto typu naprogramovanou firmou *Blizzard* a to hrou *Diablo (1996)*. V těchto hrách je kladen důraz spíše na akční stránku hry nikoliv na příběhovou.

-**Epické RPG:** Oproti akční RPG se to v těchto hrách má přesně naopak. Tedy velkou stránkou her je příběhová linie.

-**MMORPG (Massive multiplayer online role playing game):** Jak již z názvu vyplývá, jedná se pro více hráčů běžící po počítačové síti. Hráč zde v reálném čase potkává ostatní hráče, se kterými může různě smlouvat, vyměňovat věci anebo i bojovat. Hráč se zde může zaměřovat buď'to na PvP (Player vs. Player), kde si svoji postavu záměrně trénujeme na boj s ostatními hráči, a nebo na PvE (Player vs. Enemy), kde naší postavu přizpůsobujeme pro boj s různými Bossy (nejtěžší, finální nepřátelé) a procházením všelijakých tzv. Dungeonů s ostatními hráči. Lze také samozřejmě si postavu připravit na obě tyto cesty. Myslím si, že nejhranější hrou tohoto žánru na PC je hra *World of Warcraft* od společnosti *Blizzard*, kdy první datadisk, který rozšiřuje původní hru, vyšel v roce 2004.

Akční a Epické RPG je zastoupeno v několika hrách v obchodě Google-Play. MMORPG hry pro platformu Android také existují. Mezi nejlepší free-to-play (zdarma dostupné) hry tohoto typu patři:

- ***Arcane Legends:*** Tato hra je ve 3D grafice. Hráč ji může hrát jak sám, tak s ostaními připojenými hráči.
- ***Avabel online:*** Herní svět obsahuje spoustu herních úkolů, které hráče vedou dějem hry.

- ***Brave Trials:*** Hra obsahuje více jak 120 schopností, které se vaše herní postava může naučit.
- ***Dungeon Hunter:*** Tato hra má hodně verzí (poslední je verze 5) a je jedna z nejpopulárnějších her mezi hráči.[5]

2.2 Hry typu „Rage game“

Tento typ her se začal rozvíjet v nedávné době. Zatím je známo mezi hráčskou komunitou jenom pár titulů.

Hry jsou většinou Plošinové Arkády (viz kapitola 2.1). Vyznačují se vysokou obtížností. Jako hráč těchto her nemáte žádné pomocné atributy hry, jako jsou životy a bonusy. Některé hry ale poskytují buď to menší pomoc formou tzv.“checkpointů“ anebo uložením dané hry po skončení jednoho kola. Hráč by se měl připravit, že bude neustále umírat a opakovat jednotlivá kola.

2.2.1 Super Meat Boy.

Tato hra je asi největší inspirací pro moji práci. Super Meat Boy je nezávislá video hra, kterou navrhnul Edmund McMillen a Tommy Refenes. Vývoj na této hře začal už v roce 2009. Hra vyšla na krátkou dobu v roce 2010, ale byla vývojáři vzata zpět pro dodělání. Finální verze tedy vyšla až o rok později a to v roce 2011, screenshot hry můžete vidět na obrázku č. 2.1. Svého zastánce ve světě mobilních telefonů ale stále nemá.

V roce 2010 také hra dostala ocenění za nejnáročnější hru z IGN (Imagine Games Network). Kritici velmi chválili přesné herní ovládání, retro grafiku a soundtrack. Od roku 2012 se prodalo kolem dvou miliónů titulů.

Jedná se o plošinovou arkádu skládajících se z několika herních levelů a jejich finálních Bossů (viz Obrázek 2.2). Jednotlivé úrovně zvyšují stále svoji obtížnost a rozlehlost. Hráč má pouze jeden jediný úkol a to proskákat se od bodu A do bodu B s tím, že uprostřed jedné úrovně není možné použit jakýkoliv checkpoint, takže při úmrtí hráč musí opakovat celé kolo od začátku. Po úspěšném zdolání úrovně se uloží do paměti jako dokončená.

Spoustu hráčů se také snaží dokončit dané kolo ve stanoveném čase, který

Obrázek 2.1: SuperMeatBoy

nám otevře tu samou úroveň, ale ještě ve větší obtížnosti. Tím se stává velice obtížné dokončit hru na sto procent.

Obrázek 2.2: SuperMeatBoy-Boss[13]

2.2.2 Shobon Action

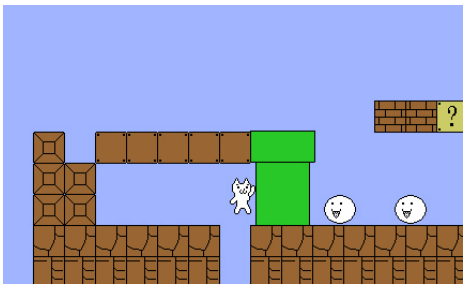
Jedná se o 2D arkádu vyrobenou japonským programátorem Chiku, který ji roku 2007 naprogramoval za tři dny. Mezi hráčskou komunitou je známá spíše pod názvem *Cat Mario* (viz Obrázek 2.3). Hra se dá považovat za parodii hry *Super Mario Bros (1985).*

Hra má málo úrovní (cca. 5 úrovní) a v každé úrovni je jeden záchytný bod. Hra nemá úložní systém, takže jediná možná cesta jak hru dohrát je, hrát hru od začátku až do konce v jednom kuse. To je i při velmi malém počtu úrovní velice obtížné.

Hráč představuje kočku, která prochází úrovněmi velmi podobným jako ve hře Super Mario Bros. Jednotlivé úrovně se dají rozdělit do několika částí,

Obrázek 2.3: Shobon Action

kde jednotlivá část obsahuje několik skrytých hrozeb (viz Obrázek 2.4), k jejich nalezení a následnému zjištění jak danou část udělat, hráč jistě obětuje několik svých životů. Hráč začíná se třemi životy. Počet životů se sice odečítá, ale nemá vliv na běh hry. Pouze nás informuje, do jakého velkého mínusu se hráč dostal v počtu životů.



Obrázek 2.4: Shobon Action - Past [10]

Hru je možné si stáhnout i pro platformu Android v obchodě GooglePlay. V tomto obchodě lze vyhledat pod názvem Cat Mario několik titulů, ale pouze jeden titul odpovídá první verzi hry pro osobní počítače.

2.2.3 I wanna Be the Guy (IWBTG)

Jako poslední rage-game hrou, kterou zde uvedu, je titul naprogramovaný nezávislým developerem Michaelem ”Kayin”O’Reilliem. Hra spatřila světlo světa roku 2007 a zdrojový kód pro hru byl dostupný roku 2011. Její úvodní image můžete vidět na obrázku 2.5.

Dá se říci, že hra bere prvky z obou dvou zmiňovaných her v předcho-

Obrázek 2.5: IWBTG

zích kapitolách. Hra disponuje spoustou těžkých překážek, a také má plno skrytých hrozeb.

Hra se skládá z několika úrovní, kde na konci každé úrovně čeká Boss (poslední překážka, jejíž zdolání znaméná úspěšné dokončení úrovně). Bossové jsou postavy ze známých starých video her, jako např.: Mike Tyson (*Punch-Out*), Dracula (*Castlevanie*) nebo Bowser (finální boss ze hry *Super Mario*). Při neúspěšném zdolání překážky hráč umírá a musí znovu opakovat danou úroveň (viz Obrázek 2.6).

Obrázek 2.6: IWBTG Game over

Originální hru na platformu Android sehnat nelze, ale existuje spousta předělávek jako například:

- *I wanna Be The Challenger*
- *I wanna Be The Phoenix*
- *I wanna Be The Four Elements*

3 O platformě Android

Android je jeden z nejrozsáhlejších operačních systémů pro chytré telefony, PDA a tablety. Je založený na open source platformě neboli software s otevřeným kódem.[6]

3.1 Používané verze

První verzi systému Android, kterou vydala společnost Google dne 5. listopadu 2007, byla verze 1.0.

Nejnovější a zároveň nejpoužívanější verze systému Android jsou **Android 4.0/4.0.1/4.0.2(Ice Cream Sendwich)** a **Android 5.0 (Lollipop)**. [8, 6]

3.2 Vývojové nástroje a prostředí

Android aplikace jsou primárně určeny pro programovací jazyk Java. Proto jsem si k implementaci vybral vývojové prostředí Eclipse, které umožňuje zdarma si stáhnout nástroje pro vývoj aplikací pro Android.[8]

3.2.1 Java Development Kit (JDK)

JDK je základní nástroj pro práci v programovacím jazyku Java. Obsahuje základní nástroje a knihovny pro vývoj aplikací. Základní součást je JRE sloužící pro spuštění, dále také překladač, debugger atd.

3.2.2 Android SDK

Další důležitá část je Android SDK balíček, se kterým komunikuje přímo Eclipse. Tento balíček obsahuje vše potřebné pro vývoj aplikací pro tuto platformu. Obsahem toho instalačního balíčku je **SDK Manager**, se kterým

můžeme stáhnout odpovídající platformy Androidu. Pomocí SDK manageru můžeme nastavit tzv. **Virtuální mobilní zařízení AVD**, které nám umožnují vytvářet emulátory jednotlivých typů zařízení a testovat na nich naší aplikaci. [8]

3.2.3 Android Studio

Jedná se o vývojové prostředí specializované na tvorbu aplikací pro Android. Google jej uvedl v červnu roku 2013 a potom dále rozvíjel až do dnešní doby. Celé prostředí obsahuje vše, co je pro vývoj potřeba a celkově se zde pracuje lépe než v Eclipse. Pro implementaci své hry jsem ho však nepoužil, zůstal jsem u známého vývojového prostředí Eclipse.

Pro vývoj mobilních aplikací pro Android je dnes tou nejlepší cestou a to hlavně z toho důvodu, že Google ukončil práce na Eclipsu s rozšířením Android Development Tools. [15]

3.2.4 Frameworky pro vývoj her

Na internetu je mnoho frameworků, které programátorovi pomáhají s tvorbou hry pro platformu Androidu pomocí programovacího jazyka Java či jiných jazyků. Frameworky se starají o fyzikální stránku hry, optimalizaci hry pro lepší výkon, UI a mnoho dalších. Některé z nich dokonce umožní vývoj her bez jaké koliv znalosti programování (např. GameMaker)[12].

- **Corona SDK:** široce používaný framework u herních vývojářů. Tvoří jednoduchou implementaci a vývoj pomocí tohoto frameworku je velice rychlý. Používá programovací jazyk Lua. Podporuje IOS, Android a Windows Mobile.[9]
- **Edgelib:** 2D a 3D herní engine. Framework je multiplatformní, podporuje Android, iOs, Windows Phone a Symbian.[3]
- **Unity Mobile** mobilní verze populárního frameworku pro vývoj her Unity3D. Od ostatních frameworků podporuje i BlackBerry a Tizen.[14]
- **Emo** framework založený na OpenGL ES a OpenGL AS. Umožňuje vytvořit hru jak pro Android tak pro IOs. [11]

Žádný z internetových frameworků jsem v implementaci nepoužil. Jeden z hlavních důvodů bylo kompletní pochopení a naučení implementace všech potřebných herních prvků (např. tvorba animace, fyzikální podstatu hry, kolize mezi hernímy prvky a jiné).

4 Android aplikace

Aplikace, které se dají vytvořit v tomto operačních systému, se dělí do tří kategorií :

- **Aplikace na popředí:** Typickým příkladem jsou hry nebo aplikace pracující s mapami (např. navigace). Tyto aplikace se dají kdykoliv pozastavit, z důvodu nějaké události (hovor, SMS atd..).
- **Aplikace na pozadí:** Běží mimo běh zařízení. Jsou to například aplikace sledující telefonní hovory a SMS.
- **Aplikace s přerušovanou činností:** Jedná se vlastně o spojení předchozích dvou typů. Běží na pozadí, ale očekává se komunikace s uživatelem. Typickým příkladem je přehrávání médií.[8]

4.1 Základní komponenty aplikace

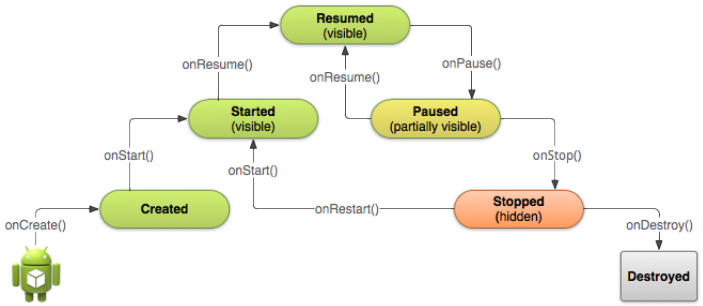
Každá aplikace je složena z několika komponentů. Hry reprezentují aplikace na popředí, protože používají grafické rozhraní. Pro implementaci hry bude tedy nejdůležitější pochopit, jak funguje první komponenta Aktivity, která je popsána v následující kapitole. [8]

4.1.1 Aktivity (Activity)

Aktivita nám poskytuje dialogové okno, které typicky vyplňuje celou obrazovku nebo představuje tzv. „plovoucí okno“. Aplikace je většinou sestavena z několika aktivit, kde hlavní aktivita se stará o běh celé aplikace. Tato hlavní aktivita je spuštěna jako první.

Mezi jednotlivými aktivitami se dá přepínat. Po přepnutí se předchozí aktivita uloží do zásobníku, který funguje na systému LIFO (last in, first out), tím je zajištěno, že se vždy budeme moci vrátit z jedné aktivity přes předchozí až na tu hlavní.[8]

Aktivita má svůj životní cyklus, který se skládá z několika důležitých metod (viz Obrázek 4.1).



Obrázek 4.1: Životní cyklus Aktivity [1]

- **onCreate():** Metoda volána při vytvoření aktivity.
- **onStart():** Metoda volána při spuštění dané aktivity, anebo když se uživatel chce vrátit k této aktivitě.
- **onRestume():** Metoda je volána, když se aktivita dostává na popředí.
- **onPause():** Metoda volána, když se uživatel přesune do jiné aktivity.

4.1.2 Android Manifest

Jedná se o XML soubor, který operačnímu systému sděluje, jaké komponenty jsou k dispozici. Dále specifikuje parametry aplikace – název, verze. Pomocí android manifestu můžeme dávat aplikaci různé oprávnění (např. přístup k fotografickému zařízení, nebo nadefinovat různé atributy aplikace (náklon obrazu, název aplikace, šířka a výška layoutu atd.). [8, 6]

5 Tvorba hry

Pro tvorbu hry jsem si vybral prostředí Eclipse. Důvodem volby bylo, že je mi dobře známé, dobře se mi s tímto prostředí pracuje, a na mojí 2D bylo dostatečné. Při složitějších 3D hrách už se doporučuje používat nejlépe OpenGL ES.

Nejdříve je třeba se naučit, jak se hry v tomto prostředí dělají - jak vykreslovat jednotlivé objekty a kam, tvoření animace a kolizí mezi objekty. Jelikož je hra ovládána dotykem a náklonem, proto jsem se musel naučit pracovat s těmito ovládacími prvky.

5.1 Popis hry

Rozhodl jsem se udělat hru herního typu „Rage Game“ (viz kapitola 2). Hra by měla být ovládána náklonem zařízení. Podle velikosti náklonu by měla postava, za kterou hráč hraje chodit rychleji nebo pomaleji. Postava by měla umět skákat a tato akce by se vyvolala kliknutím na obrazovku telefonu.

Hra by měla mít velký počet úrovní (přes dvacet úrovní), kde v každé úrovní by měl hráč za úkol se dostat ze startu do cíle. Dvě z dvaceti kol by obsahovaly souboj postavy s Bossem, kde by se uživatel pomocí nástrojů na mapě snažil Bosse zneškodnit.

5.2 Vykreslování jednotlivých objektů

Nyní se podíváme na dvě možnosti, jak vykreslovat naše objekty na obrazovku telefonu.

5.2.1 View

View je základním grafickým prvkem systému Android. Stará se o vykreslování objektů a zachytávání událostí. Pro hry je však nepoužitelný, a to z jednoho prostého důvodu. Metoda vykreslování View je totiž volána na

```
holder.addCallback(new SurfaceHolder.Callback() {

      @Override
       public void surfaceDestroyed(SurfaceHolder holder) {
       }

      @Override
       public void surfaceCreated(SurfaceHolder holder) {
        }

       @Override
        public void surfaceChanged(SurfaceHolder holder,
         int format,int width, int height) {
           }
 });
```

Ukázka zdrojového kódu 5.1: Zpětné volání [4]

stejném GUI vlákně (Thread). Proto je potřeba najít takový grafický prvek, který lze spustit na vlastním vlákně, aby se překreslování dělalo rychleji a nezpomalovalo ostatní činnosti naší aplikace.

5.2.2 SurfaceView

Je to potomek třídy *View*. Jeho jedním z cílů je poskytnout *View* se sekundárním Vláknem (*Thread*). Přístup je zajištěn pomocí třídy *SurfaceHolde*, který lze získat voláním metody *getHolder()*[4]. K volání metody *onDraw()* je zde třeba předat jako parametr třídu *Canvas*. *Canvas* lze chápat jako jakousi tabuli, kde můžeme kreslit prakticky, co chceme. Když je vše připraveno pro začátek vykreslování, je třeba ještě v *SurfaceHolder* nastavit zpětné volání (viz Ukázka 5.1), sloužící k nastavení následujících tří základních metod.

- *surfaceDestroyed (SurfaceHolder holder):* metoda volána při zničení našeho povrchu.

```
public void surfaceCreated(SurfaceHolder holder) {
    Canvas c = holder.lockCanvas(null);
    onDraw(c);
    holder.unlockCanvasAndPost(c);
}
```

Ukázka zdrojového kódu 5.2: Implementace metody surfaceCreated[4]

- *surfaceChanged (SurfaceHolder holder, int format,int width, int heigh) :* metoda volána při strukturální změně.
- *surfaceCreated(SurfaceHolder holder) :* metoda volána při prvním spuštění.

Před začátkem vykreslování na plátno je potřeba dané plátno uzamknout pomocí našeho *SurfaceHolder* a po dokončení volání metody *onDraw()* opět odemknout (viz Ukázka 5.2).

Když to shrneme je tedy potřeba vytvořit třídu *GameView* , která bude potomkem *SurfaceView*. V této třídě vytvořit instance *SurfaceHolder* a přidat jí a nastavit metody zpětného volání.

5.3 Tvorba Textur

Veškeré textury jsem kreslil sám a pro jejich tvorbu jsem se rozhodl použít program pro tvorbu vektorových obrázku Inkscape. Dále pro export obrázku do bitmapového formátu (.png a .gif) jsem použil grafického nástroje GIMP. Celkem jsem vytvořil 44 obrázků, některé z nich jsou obsaženy v příloze.

5.4 Animace

V této kapitole bude popsána základní herní smyčka. Dále se podíváme na vše potřebné pro tvorbu animace ve 2D hře.

5.4.1 Game Thread

Z předchozí kapitoly bylo řečeno, že pro vykreslování bude používán *SurfaceView*, který může pracovat na sekundárním vlastním vláknu. Je třeba vytvořit naše vlastní vlákno, které pak můžeme spustit v metodě *surfaceCreated* kódem *gameLoopThread.start();* a při příchodu přerušovací události i zastavit pomocí metody *surfaceDestroyed* kódem *gameLoopThread.join();*.

V metodě *run* následně bereme *Canvas* z našeho *SurfaceHolderu*, a poté jej uzamkneme a zavoláme v našem *GameView* metodu *onDraw()*.

Dále toto vlákno můžeme uspat na určitou dobu a tím určit, kolikrát nám může proběhnout za sekundu. Tak se ve hře určuje tzv. FPS (počet snímků za sekundu).

5.4.2 Game Loop (Herní smyčka)

Každý objekt ve hře (hlavní hrdina, protihráči, střely, atd.) má svoje základní dvě metody *onDraw()* a *update()*.

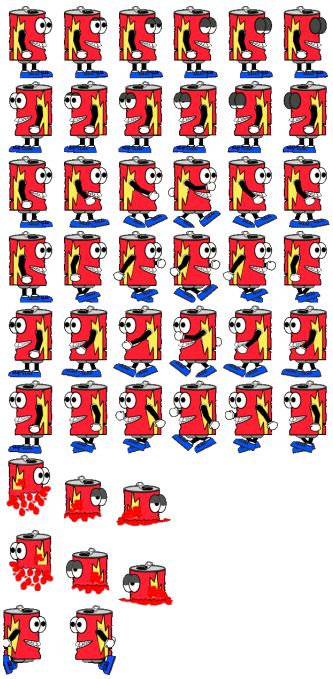
- *onDraw(Canvas c):* vykresluje nám náš objekt na plátno zadané v parametru.
- *update():* provádí veškeré změny v objektu, například: změna x a y souřadnic a pozici ve Sprites (viz další podkapitola).

5.4.3 Tvorba animace

Textura objektu je ve formě tzv. Sprites. Jedná se vlastně o jeden obrázek, kde jsou přesně vedle sebe postaveny možné polohy našeho objektu (viz Obrázek 5.1).

Na obrázku můžeme vidět celkový počet sloupečků a řádků, které tento obrázek rozdělí do pomyslných obdélníků. Pomocí těchto údajů si můžeme spočítat opravdovou šířku a výšku našeho objektu.

Animace se provádí tak, že je v jednom čase vykreslena pouze část našeho Sprites s jednou polohou objektu. Máme Sprites pozice s číslem řádku



Obrázek 5.1: Sprites Tiny

(animR) a sloupečku (animC), kterou měníme. Tím vykreslujeme stále jinou polohu našeho obrázku (viz Ukázka 5.3).

```
int srcX = animC*width;
int srcY = animR*height;

//odkaz na vybraný obdélník z textury

Rect src = new Rect(srcX, srcY, srcX + width, srcY + height);

//Obdelník na který se vykreslí vybraný obdélník z textury

Rect dst = new Rect(x, y, x+ width, y+  height);
canvas.drawBitmap(btm,src ,dst,null);
```

Ukázka zdrojového kódu 5.3: Animace[4]

Například pro pohyb doprava mého hlavního hrdiny Tinyho v metodě *update()* nastavíme *animR* nejdříve na hodnotu tři pro pohyb levé nohy dopředu a pravé ruky nahoru, a pak *animC* na počátek animace (hodnota 0). Při dalším volání update metody inkrementuji hodnotu *animC* dokud nedosáhne hodnoty pět, poté se nastaví *animR* na čtyři a *animC* zase na nulu. Vlastně stále přepínáme mezi dvěma řádky Sprites a stále inkrementujeme *animC* do určité hodnoty. Tím docílíme celkem pěkné animace pro pohyb doprava s pohybem všech končetin.

5.4.4 Náklon a detekce dotyku

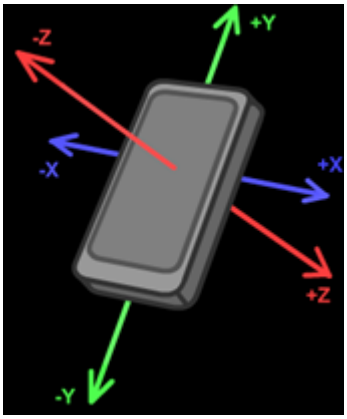
Pro získání souřadnic náklonu telefonu je potřeba dvou tříd *SensorManager* a *SensorEventListener*.

Nejdříve je potřeba si nastavit*SensorEventListener*, kde při inicializaci je potřeba implementovat dvě metody:

- *onSensorChanged(SensorEvent se)*
- *onAccuracyChanged(Sensor sensor, int accuracy).*

Stačí implementovat první z metod, která je volána, když se hodnoty senzoru změní. Pro náklon je potřeba senzor Akcelerometr. Nejdříve, než za-

čneme číst hodnoty senzoru, musíme přes metodu *se.sensor.getType()* ošetřit, jestli nám tuto metodu zavolal právě ten senzor, od kterého data chceme. Akcelerometr nám vrací hodnotu souřadnice svého náklonu. (viz Obrázek 5.2).



Obrázek 5.2: Akcelerometr[7]

Hra je nastavena na orientaci tzv. „landscape“ *android:screenOrientation=”landscape”* (aplikace běží pouze na orientaci na šířku). Orientace se nastavuje v Android Manifestu (viz. Kapitola 4.1.2).

Když nastavíme tuto orientaci v manifestu, zamezíme aplikaci, aby měnila svůj obraz při změně orientace telefonu (další orientace je tzv. „portrait“, neboli orientace na výšku). Při našem nastavení nás tedy bude zajímat pouze y-ová souřadnice. Pomocí této souřadnice můžeme zvyšovat rychlost našeho hrdiny. Hodnota souřadnice určuje, na jakou stranu se bude naše postava pohybovat - vlevo či vpravo. Pro detekci dotyku na obrazovku stačí přepsat metodu *onTouchEvent(MotionEvent event)*. Tu můžeme překrýt ve třídě dědící Activity a nebo jí můžeme dokonce překrýt i ve třídě *gameView* dědící *SurfaceView* (viz Kapitola 5.1.2). V metodě pomocí třídy *MotionEvent* dostaneme x-ové a y-ové souřadnice zachyceného dotyku na display (viz Ukázka 5.4).

5.4.5 Kolize mezi objekty

Při řešení herních kolizí lze použít tři základní herní algoritmy. Všechny tyto algoritmy jsem využil ve své práci, protože pro každý objekt a situaci se hodí algoritmus jiný.

```
public boolean onTouchEvent(MotionEvent event) {
Log.i("Tiny touch:", "x: "
+ event.getX() + " y: " + event.getY());
}
```

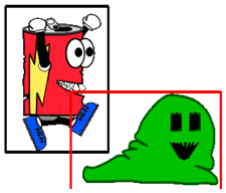
Ukázka zdrojového kódu 5.4: Zachycení náklonu

- Bounding Rectangle/Box Collision:

Tento algoritmus využívá pro detekci kolize obdélník (Bounding Rectangle Collision) nebo souřadnice obdélníku kolem objektu (Bounding Box Collision).

Pomocí metody *rect1.intersect(rect2)* nebo pomocí souřadnic se zjistí detekce dvou objektů.

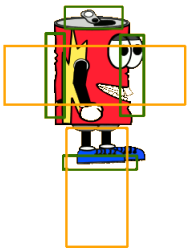
Pro detekci našeho hrdiny s nepřáteli je tento algoritmus nevhodný (viz Obrázek 5.3), protože program vyhodnotí kolizi a přitom se dva objekty nedotkly.



Obrázek 5.3: Kolize - Bounding Rec. Collision

Tuto metodu jsem použil při detekci pohybujících se objektů s podlahou nebo stěnou. Jednotlivý objekt je popsán celkem šesti obdélníky - horním, dolním, levým, pravým, spodním a okolním (viz Obrázek 5.4).

Spodní a okolní obdélník (na obrázku vyobrazeny oranžovou barvou) slouží k tomu, když se náš objekt nějakou rychlostí přibližuje k podlaze nebo ke stěně, tak se jeho rychlost postupně zmenšuje a náš objekt dopadne na podlahu bez vniknutí do textury podlahy nebo stěny. Rychlosti jsou zmenšovány a přizpůsobovány tak, aby zpomalení nebylo zaznamenáno lidským okem a uživateli se zdálo, že dopad byl přesný.



Obrázek 5.4: Kolize obdélníky

- Per-Pixel Collision:

Jak z názvu vyplývá, pro detekci se bude pracovat s jednotlivými pixely objektu. Nejdříve se zjistí pomocí Bounding Rectangle Collision (popsán výše) jestli je potřeba vyhodnotit tento algoritmus, a poté se projedou všechny pixely obou objektů. Kolize se vyhodnotí jako pravdivá, když se najdou na stejném místě dva pixely, které nemají průhlednou barvu. Tato metoda je sice velice náročná na systém, ale za to nejpřesnější pro detekci kolize. Z toho důvodu jsem tuto metodu použil pro detekci hlavního hrdiny „Tiny“ s nepřáteli. Bohužel se tato metoda ukázala nevhodná pro detekci s kterýmkoliv rotačním objektem, jako je ve hře nepřítel Reaper (viz kapitola B.5). Při rotování objektu se stále v paměti udržuje bez rotace, a tak nám metoda pracuje s nerotovaným obrázkem. Proto jsem pro rotační objekty musel použít následující metodu detekování kolize.

- Separating Axis Theorem:

Metoda určena pro způsob jak zjistit jestli se dva konvexní tvary protínají.

Objekty jsou popsány okolními vektory, které jsou přepočítávány při změně polohy nebo při rotaci objektu. Při kontrole kolize se matematicky spočítá, jestli některý z vektorů jednoho objektu protnul některý vektor druhého objektu [2].

Jak jsem již zmínil, tuto metodu jsem použil pro detekci kolize hlavní postavy „Tiny“ s překážkou Reaper (viz Kapitola B.5).

5.4.6 Tvorba mapy

Při tvorbě mapy pro 2D hru je potřeba si nejdříve rozdělit display na několik tzv. „buněk“. Při mé aplikaci jsem si rozdělil display na 25 buněk na šířku a 10 buněk na výšku.

Po dokončení dostanu velikost jednotlivých buněk, které budou různě veliké v závislosti na displeje daného zařízení. Pokud tedy chceme, aby na každém zařízení vypadala naše hra stejně velká, můžeme využít těchto hodnot pro přepočítání a změnění velikosti jednotlivých textur při načítání hry.

Při tvorbě mapy pak stačí do jakéhokoliv souboru uložit matici znaků. Každý znak představuje herní objekt, jako například:

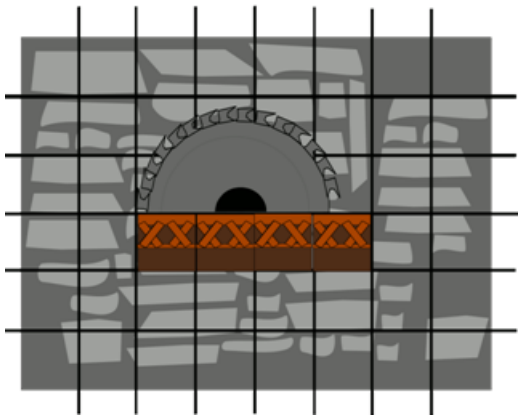
- **0:** volná plocha
- **1:** podlaha
- **2:** stěna
- **S:** start
- **F:** cíl
- **T:** začáteční souřadnice pro hlavního hrdinu „Tiny“
- **3:** pozice pro Slim (viz Kapitola B.5)
- **4:** pozice pro pilu (Saw)

Jednotlivé objekty a jejich určovací znaky jsou více popsány v přílohách (viz Příloha B.5).

Pomocí pozice znaku v matici a údajích o velikosti buňky se spočítá souřadnice objektu. Například pro mapu (viz Obrázek 5.5) by soubor s mapou vypadal takto:

00000000
00000000
00400000
00111100

00000000
00000000



Obrázek 5.5: Mapa

Veškeré objekty mají jinou velikost, tak je potřeba po získání pozice ještě přepočítat souřadnice, aby se nám objekt umístill přesně tam, kde potřebujeme (např. pila na obrázku 5.5).

5.4.7 Hudba a zvukové efekty

Jednotlivé zvukové efekty i hudba na popředí by měly být uloženy v systémové složce aplikace raw, která je read-only (pouze pro čtení).

Pro hudbu využijeme třídu *MediaPlayer*. Tento přehrávač se vytváří metodou *create(Context context, int resid)*.

- **context:** Rozhraní pro globální informace o aplikačním prostředí. Je to abstraktní třída, jejíž realizace je zajištěna Android systémem.
- **resid:** Id zvukového souboru získané z *R.raw.<název>*.

Po vytvoření přehrávače metodou *mp.start()* a *mp.stop()* pouštíme nebo vypínáme hudbu. Opakování písničky po dohrání je zajištěno v metodě *update()* (viz Kapitola 5.2.2.). V této metodě se pomocí metody *mp.isPlaying()* zjistí, jestli hudba stále pořád hraje, jestli ne je puštěna znovu.

```
public final int play (int soundID, float leftVolume,
 float rightVolume, int priority, int loop, float rate)
```

Ukázka zdrojového kódu 5.5: Metoda pro přehrávání

Zvukových efektů máme ve hře samozřejmě několik. Na to je potřeba jiná třída než *MediaPlayer* a to třídy *SoundPool*. Jedná se vlastně o kolekci zvukových efektů, které lze vložit pomocí metody *load(Context kontext, int id)* podobně jako u *Media player*, do paměti ze zdroje uvnitř aplikace nebo ze souboru. Tedy hlavní výhodou oproti *MediaPlayer* je počet zvukových souborů.

Další výhodou je přehrávání s tím, že se dá nastavit pravá i levá hlasitost zvuku (viz Ukázka 5.5).

Při volání metody play se musí zadat ID našeho zvukového efektu, který dostaneme pomocí *R.raw.<název efektu>*. Poté zadám prioritu přehrávání a frekvenci přehrávání.

Hudba na pozadí a veškeré zvukové efekty jsou volně dostupné ke stažen.

5.4.8 Uložení stavu hry

Pro uložení stavu hry a jejího zapamatování i po vypnutí aplikace se dá použít několik způsobů. Jedním takovým způsobem je využít třídu *SharedPreferences*, která se hojně používá při zapamatování přihlašovacích údajů, hesel, apod. Tato třída využívá formátu klíč-hodnota k uložení jednoduchých dat do vnitřní paměti aplikace.

Já jsem se však rozhodl jít jinou cestou, vytvořil jsem si do interní paměti telefonu soubor, který pomocí příslušných utilit *FileOutputStream* a *InputStreamReader* přepisuji nebo čtu. Při vytváření instance třídy *FileOutputStream* se nastaví cesta k souboru a mód *Context.MODE_PRIVATE*, který umožní přístup k souboru pouze naší aplikaci. Struktura souboru se skládá z několika řádků, kde na každém z nich jsou data ve formátu:

<číslo_kola>;<minuty_dokončení>;<sekundy_dokončení>

Při ještě nedohraném kole jsou hodnoty *minuty_dokončení* a *sekundy_dokončení*

nastaveny na *N:N*. Počet řádků souboru mi určuje počet otevřených kol ve hře. Při prvotním spuštění se tedy vytvoří soukromý soubor *score.txt*, který bude mít ze začátku pouze jeden řádek: *1:N:N*. Tento řádek aplikaci sdělí, že je otevřené pouze jedno kolo a zatím nebylo nikdy dohrané. Při dohrání jednoho kola se změní čas (nebo nezmění, podle toho jestli je větší, než čas který byl už někdy zahrán). Jestli se přepisuje řádek, s hodnotou *N:N*, přidá se do souboru nový řádek a tím se uživateli otevře nová úroveň.

5.4.9 Přerušení hry při změně aktivity

Při jakékoliv aplikaci je důležité ošetřit, co se stane, když běh naší aplikace naruší některá z jiných aktivit (například příchozí hovor).

V aplikacích, kde na pozadí neběží žádné další vlákno nebo některý výpočet, se stopnutí v podstatě řešit nemusí, ale například ve hrách vede narušení k pádu aplikace při snaze jejího znovu najetí.

Tento pád způsobují právě běžící vlákna, protože při znovu najetí aplikace se volá metoda *onResume()* (viz Obrázek 4.1). Tato metoda se v našem *SurfaceHolder* (viz Kapitola 5.1.2) bude snažit opět zapnout vlákno, o kterém si systém myslí, že stále běží.

Když se podíváme na náš *SurfaceHolder*, kde jsme nastavovali metodu jako *surfaceCreated*, a zároveň metodu *surfaceDestroyed*, která by měla být volána při přerušení aplikace zjistíme, že tato metoda není volána při všech druzích přerušení. Je tedy vhodné přesunout vypnutí vláken a tím nastavit stopnutí hry do metody *onPause()*, která je volána vždy.

Tím se ošetří nepříjemný pád hry při příchozím hovoru či jiných narušení běhu naší aplikace.

6 Programátorská dokumentace

Program se skládá celkem z 29 tříd. Všechny třídy jsou popsány v Javadoc dokumentaci, kterou lze nalézt na přiloženém CD ve složce Javadoc. Následuje popis nejvýznamnějších tříd. Nejvýznamnější třídy je možné vidět i v UML diagramu v přílohách.

6.1 GameView

Třída, která se stará o celkové dění hry. Nejprve načte všechny textury a upraví jejich velikost podle používaného rozlišení displeje. Zobrazuje aktuální herní scénu, spouští jednotlivé herní efekty a hudbu. Volá metody *update()* a *onDraw(Canvas canvas)* všech herních prvků. V poslední řadě spouští a pozastavuje herní smyčku.

Vybrané metody:

- **update():** Zavolá metodu *update()* všech aktuálně používaných herních prvků.
- **onDraw():** Zavolá metodu pro vykreslení všech aktuálně používaných herních prvků *onDraw(Canvas canvas)*. Parametr *canvas* je instance plátna, na který se pvky vykreslí.
- **setInGame(int inGame):** Přepne do části herní aplikace podle parametru *inGame* (0 - hlavní menu, 1 - výběr mapy, 2 - tutorial, 3 - hra a 4 - seznam skóre).
- **playSoundEffect(int i):** Spustí zvukový efekt podle indexu předaném v parametru *i*.
- **startGame(String mapa):** Načte mapu pro vybranou úroveň ze souboru předaném v parametru *mapa*. Zavolá metodu třídy Map (viz Kapitola 6.3) pro vytvoření mapy. V poslední řadě nastaví souřadnice hlavní postavě a spustí danou úroveň.
- **finishGame():** Metoda je zavolána, když hlavní hrdina dokončí úroveň. Vytvoří a spustí animaci tabulky, která se postupně objevuje na

obrazovce. Tabulka obsahuje informace o čase dokončení úrovně a tlačítka pro návrat do menu hry, pro restartování úrovně a spuštění další úrovně.

6.2 GameMenu

Třída zobrazující část pro výběr levelu. Na obrazovce se zobrazí šest oken. Jednotlivá okna mají v sobě buď číslo (číslo mapy) nebo zámek. To znamená, že je daná úroveň zamčena a hráč musí dohrát předchozí kolo, aby ji odemknul. Jelikož ve hře je celkem dvacet kol a oken je zobrazeno pouze šest, je v pravém dolním rohu tlačítko, kterým zobrazíme dalších šest oken.

Vybrané metody:

- **loadScore(LinkedList<Integer> finished):** V parametru je předán seznam dokončených kol. Tento seznam poté určí, jaká okna budou mít číslo a jaká budou mít zámek.
- **onDraw(Canvas canvas):** Vykreslí šest oken na plátno předané v parametru. Dále v každém okně vykreslí číslo mapy, a nebo zámek. Číslo se zobrazí, když je index okna v seznamu dokončených kol, nebo když je index o jedno větší než poslední dokončené kolo.
- **getTouch(float x,float y)** Podle x-ové a y-ové souřadnice dotyku na displej metoda zkontroluje, zda se uživatel nedotkl některého okna s číslem. Jestli ano, spustí metodu pro start dané úrovně ve třídě GameView.

6.3 Map

Tato třída se stará o vytvoření a vykreslení mapy. Uchovává v sobě dva seznamy, seznam podlah a stěn vyskytujících se na mapě a seznam překážek a protivníků.

Vybrané metody:

- **loadMap(InputStreamReader inputStream)):** Načtě všechny herní prvky dané mapy.
- **reset():** Vymaže data v obou seznamech.

6.4 Score

Třída spravující soubor *score.txt*, kde jsou uložené časy dokončení jednotlivých úrovní.

Vybrané metody:

- **checkScore():** Zkontroluje jestli soubor *score.txt* existuje pokud ne, vytvoří jej.
- **getOpenLevel():** Metoda vrací seznam dokončených kol.
- **updateList(int mapa,float finishedtime):** Metoda je volána, když uživatel dokončí úroveň (index úrovně je předán v parametru *mapa*). Zkontroluje se, jestli uživatel nedosáhl nového rekordu dané úrovně. Jestli ano, zapíše nový čas dokončení do souboru.

6.5 AttackController

Třída kontroluje kolize mezi hlavní postavou a nepřáteli.

Vybrané metody:

- **controlTinyEnemyCollisions():** Spustí kontrolu kolizí mezi hlavní postavou a všemi nepřáteli.
- **isPixelCollision(int width1,int height1,Bitmap btm1,Rect rec1,int width2,int height2,Bitmap btm2,Rect rec2):** Kontroluje pixelovou kolizi (viz Kapitola 5.4.5) mezi dvěma zadanými texturami.
- **isRotateObjectCollision(Enemy enemy,Tiny tiny):** Spustí kontrolu kolize mezi rotačním objektem a hlavním hrdinou. Kolize se kontroluje pomocí algoritmu Separating Axis Theorem (viz Kapitola 5.4.5).

6.6 FloorController

Třída kontroluje kolize pohybujících se herních objektů s podlahou a stěnami.

Vybrané metody:

- **controlCollisions():** Spouští kontrolu kolize všech pohybujících se herních objektů s podlahou a stěnami. Herním objektům při zachycení kolize řekne, kam se mohou a nemohou pohybovat.

6.7 Tiny

Třída představující funkcionalitu hlavní postavy "Tiny".

Vybrané metody:

- **update():** Spustí update daného objektu. Posune souřadnice objektu podle x-ové a y-ové rychlosti. Podle pohybu nastaví proměnné *animC* a *animR* a určí, která část Sprites se má vykreslit (viz Kapitola 5.4.3).
- **setTiltThePhone(float y):** Nastaví x-ovou rychlost hlavního hrdiny podle y-ové souřadnice náklonu zařízení, předané v parametru.
- **setDead(boolean dead):** Nastaví část Sprites na úmrtí. Dále vytvoří krev a spustí její animaci.
- **setJump():** Zkontroluje, jestli postava může provést skok (podle hodnoty proměnné *number_jump*, která nesmí být větší než 2) a popřípadě spustí schopnost skoku.
- **onDraw(Canvas canvas):** Vykreslí hlavní postavu na plátno předaném v parametru. Jestli existuje nějaká krev, vykreslí i ji.
- **nearDownFloor(Floor fl):** Metoda je volána, když postava padá na podlahu zadanou v parametru. Kontroluje y-ovou rychlost jestli není větší než vzdálenost mezi postavou a podlahou, popřípadě zmenšuje y-ovou rychlost.

6.8 GameThreadGraphics

Herní vlákno nejdříve spustí kontrolu kolizí herních objektů s podlahou a stěnami pomocí třídy *FloorControler*. Dále uzamyká plátno, na které se ve tříde *GameView* vykreslují herní objekty. V poslední řadě volá metody *update()* a *onDraw(Canvas canvas)* třídy *GameView* a znovu odemyká plátno.

6.9 GameThreadLogic

Další herní vlákno, které spouští kontrolu kolizí mezí hlavním hrdinou a nepřáteli třídy *AttackController*. Vytvoření druhého herního vlákna bylo důležité, protože kontrola kolizí mezi hlavním hrdinou a nepřáteli je velice náročná na systém. Když byla veškerá kontrola kolizí společně s voláním metod *update()* a *onDraw(Canvas canvas)* v jednom vlákně, tak hra na některých zařízení běžela pomalu.

7 Testování

Moje aplikace byla testována na několika zařízeních (viz Tabulka 7.1). Testování proběhlo na verzí Androidu 2.3.6, 4.0,4.1.2, 4.2.2 a 4.4.2. Až na jeden problém, který vznikl u staršího telefonu s verzí 2.3.6 (viz. Podkapitola 7.2), běžela hra bez problému.

Zařízení	**Android verze**	**Display rozlišení**
Galaxy Samsung Xcover	2.3.6	320x480
HTC One V T320e	4.0	480x800
Samsung Nexus S	4.1.2	480x800
HTC One X	4.2.4	720x1280
Sony Xperia T	4.0	720x1280

Tabulka 7.1: Přehled testovacích zařízení

7.1 Popis testování

Testováním aplikace se věnovala skupina uživatelů. Hlavním předmětem testování bylo odzkoušet hratelnost. Nejdříve se testoval pohyb postavy, jestli se na všech zařízeních chová stejné (chodí stejně rychle a skáče stejně vysoko). Dále se zkoušelo, jestli hlavní hrdina při pohybu neproniká do textur podlahy a stěn, tím by ve hře vznikaly tzv. herní bugy, které mají velký dopad na hratelnost.Také jsem kontroloval vzhled aplikace v jednotlivých zařízení (stejná velikost textur na různých rozlišeních).

V poslední řadě se testovalo, jestli pro některý typ přístroje není hra příliš náročná, koukalo jestli hra běží plynule bez sekání.

7.2 Výsledky testování

Testování přineslo základní informace o chodu aplikace. Pomohlo objevit problémy například s hernímy bugy (pronikání postavy do textu podlah a stěn) a náročnost pro některá zařízení. Tyto problémy se podařilo opravit a nyní by měla aplikace bez sebemenších závad fungovat. U jednoho zařízení se ale

objevil problém s příliš malým displayem (viz Podkapitola 7.3), který se mi nepodařil vyřešit tak, aby to i na tomto zařízení běželo stejně jako na ostatních.

7.3 Odlišné chování na některých zařízení

Jeden z vážných problémů vznikl na zařízení Galaxy Samsung Xcover GT-S5690. Důvodem je příliš malé rozlišení a to pouze 320x480. Jelikož jsem se snažil, aby hra vypadala a byla hratelná na všech zařízení, stejně tak je třeba před spuštěním samotné herní smyčky nastavit velikosti obrázků a rychlosti pohybů jednotlivých objektů. Zjistil jsem, že problém vznikl u rychlosti.

Nové rychlosti se přepočítávají od defaultních hodnot, které jsou pro rozlišení 720x1280. Ku příkladu y-nová počáteční rychlost našeho hlavního hrdiny je 20px a celková gravitace hry je nastavena na 3px pro toto rozlišení. Po přepočítání například na rozlišení 480x800 je gravitace snížena na 2px a y-nová rychlost na 14. Při malém rozlišení jako je 320x480 vznikají po přepočítání desetinná čísla (gravitace nemůže být 1.5px potřebujeme celé číslo). To má zásadní vliv na hratelnost hry a to například, tím že náš hlavní hrdina skáče výše než by měl.

Myslím si, že tento problém nevadí, protože všechna novější zařízení mají větší rozlišení než 320x480.

8 Závěr

Hry jsou součástí mobilních telefonů od dob, kdy uměly více než jenom telefonovat. Hry jsou dobré na uvolnění od každodenního stresu anebo také dobře poslouží ke zkrácení času při cestě v dopravních prostředcích. Postupem času se mobilní zařízení zdokonalovala a stále se zdokonalují, stejně i hry v nich nainstalované. V dnešní době chytrých telefonů, dosáhly hry dokonce úrovně her pro domácí počítače před pár lety.

Myslím si, že hra do mobilních zařízení by měla mít nenáročnou ovladatelnost, jednoduchý příběh a chytlavou grafiku, aby zaujala co největší počet uživatelů.

Cílem mé bakalářské práce bylo prozkoumání a analyzování her typu, který se v mobilních zařízeních téměř nevyskytuje, a to žánru „Rage Game“. Dalším cílem bylo vymyslet a vytvořit herní aplikaci pro mobilní platformu Android. Navrhl jsem a zrealizoval hru pro platformu Android ovládanou náklonem zařízení, která zahrnuje prvky ”Rage Game”. Hra obsahuje více jak 20 herních úrovní. K testování hry bylo přizváno několik dalších uživatelů a také se testovalo na několika mobilních zařízení s různou verzí operačního systému Android. Bylo testováno chování aplikace v různých situacích (např. shození aplikace, reagování na příchozí hovor atd.).

Herní aplikace byla úspěšně naimplementována a je dostupná na zařízení s verzí operačního systému Android vyšší než 2.3.3 a splňuje všechny požadavky, které byly v zadání práce.

Literatura

[1] Šajboch, A.: Rozšíření informačního systému EPI, s.r.o. na mobilní platformu Android. URL:<http://www.kiwiki.info/index.php/>, [online], cit. 2009-01-19.

[2] Bittle, W.: SAT (Separating Axis Theorem). URL:<http://www.dyn4j.org/2010/01/sat/#sat-intro3l/>, [online], cit. 2015-06-10.

[3] B.V., E. I. M.: Cross-platform mobile development at your fingertips. URL:<http://www.edgelib.com/index.php/>, [online], cit. 2014-08-10.

[4] edu4java: Android Game Programming. URL:<http://www.edu4java.com/en/androidgame/androidgame1.html/>, [online], cit. 2015-06-10.

[5] Hindy, J.: 15 best Android MMORPGs. URL:<http://www.androidauthority.com/best-android-mmorpgs-578250/>, [online], cit. 2015-05-1.

[6] Hoog, A.: *Android Forensics*. 225 Wyman Street,USA: Elsevier Inc., 2011, ISBN 978-1-59749-651-3.

[7] Ivar, L.: Android Accelerometer App. URL:<http://larsivar.com/wp/?p=87>, [online], cit. 2011-08-20.

[8] Jíří Vávrů, M. U.: *Programujeme pro Android*. U Průhonu22,Praha: Graga Publishing a.s., 2013, ISBN 978-80-247-4863-4.

[9] Labs, C.: Build 10x Faster. URL:<https://coronalabs.com/>, [online], cit. 2015-06-1.

[10] Neoblast: Open Syobon Action DC Released! URL: <http://dreamcast-talk.com/forum/viewtopic.php?f=5&t=4635y/>, [online], cit. 2009-01-19.

[11] emo-framework project: Emo open source framework for games. URL:<http://www.emo-framework.com/>, [online], cit. 2011-04-5.

[12] Reynolds, C.: Top IOS and Android Mobile Game Development Tools, Frameworks, Engines and Resources. URL: <http://www.mobyaffiliates.com/>, [online], cit. 2012-10-22.

[13] Ryuchi: Super Meat Boy. URL: <http://gameload.blog.cz/1305/super-meat-boy/>, [online], cit. 2013-05-14.

[14] Technologies, U.: Mobile Development. URL:<https://unity3d.com/>, [online], cit. 2014-04-27.

[15] Zapata, B. C.: *Android Studio Application Development*. 35 Livery Street,UK: Packt Publishing, 2013, ISBN 978-1-78328-527-3.

Přílohy

A Instalace

Před instalací aplikace je nutné si v nastavení telefonu povolit instalaci z neznámých zdrojů. Většinou se nachází v *Nastavení -> Zabezpečení -> Neznámé zdroje*.

V přiloženém CD lze nalézt soubor Tiny.apk, který si přesunuté do jakéhokoliv adresáře ve vašem mobilním zařízení se systémem Android. Dále se v přiloženém CD nachází soubor TinyTest.apk, který slouží k testování a po jeho nainstalování dostaneme hru, která má od začátku dostupné všechny úrovně. Pomocí některé utility na prohledávání složek (např. ES File Explorer) najdete soubor Tiny.apk. Po kliknutím na soubor se nejdříve zařízení zeptá jestli chceme tuto aplikaci nainstalovat a poté proběhne instalace. Po úspěšné instalaci můžeme spustit aplikaci Tiny a začít hrát.

B Uživatelská dokumentace

Vytvořená hra, je podle zadání, typu „Rage Game“. Jedná se o arkádu (viz Kapitola 3.) s vysokou obtížností. Ve hře se nevyskytují skryté prvky jako je to mu ve hře Shobon Action (viz Kapitola 3.2.2), ale aby hra splňovala prvky „Rage Game“ jsou jednotlivé překážky dány, tak aby jejich překonání bylo obtížné.

Hráč ve hře hraje za plechovku „Tiny“, která se v jednotlivých kolech snaží dostat ze startu do cíle. Ve hře na hráče čeká více než dvacet kol. Dvě z těchto kol jsou kola s Bossem, kterého musí hráč zneškodnit pomocí objektů na mapě (napři. bodák, který při dotyku hlavního hrdniny, spadne na Bosse).

V jednotlivých úrovních musí hráč překonávat množství nástrah (viz Podkapitola 6.5). Při střetnutí s těmito nástrahami hlavní hrdina umírá a hráč je nucen dané kolo opakovat.

Veškeré textury a grafické prvky jsem vytvořil sám s použitím programu pro vektorovou grafiku Inkscape a dále program pro práci s rastrovou grafikou GIMP.

B.1 Menu hry

Jako první se hráči objeví menu hry, které je animované a hráč se pomocí něho může dostat do dalších částí hry (Výběr mapy, tutoriál, Skóre) anebo může pomocí tlačítka „Exit“ ukončit celou hru (viz Obrázek B.1).

Dále může pomocí tlačítek v pravém horním rohu vypnout zvuk pro efekty a pro hudbu.

B.2 Ovládání

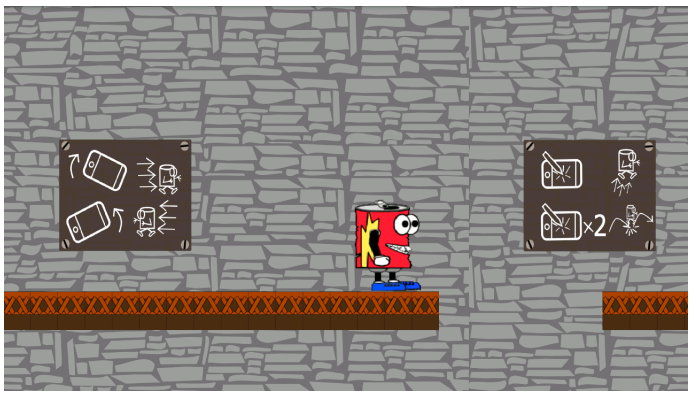
Pohyb hlavního hrdiny se ovládá náklonem telefonu s tím, že čím víc je telefon nakloněn, tím se náš hrdina pohybuje rychleji (doprava nebo doleva podle náklonu).

Obrázek B.1: Menu

Skok a následný tzv. „dvoj-skok“ se udělá dotykem na jakoukoliv část display.

B.3 Tutoriál

Tuto část hry bych doporučoval začínajícímu hráči pustit jako první. Tato mapa je velice krátká, čeká zde na hráče jeden přeskok přes propast a jedna velká zeď. Ale hráč se tu pomocí jednoduchých informačních tabulek (viz Obrázek B.2) naučí ovládat hru a jednotlivé schopnosti, které může s hlavním hrdinou dělat.



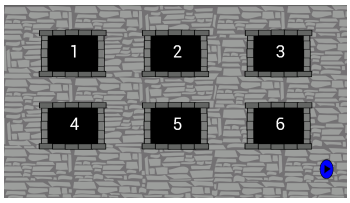
Obrázek B.2: Menu

Mezi jednotlivé schopnosti hrdiny patří dvoj-skok a držení o zeď. Když se hrdina chytí o zeď padá po ní pomaleji, ale obnoví se mu počet možných

skoků, takže touto schopností může vyskákat i na velikou zeď, kterou by dvoj-skokem nepřeskočil.

B.4 Menu výběru mapy

Do této sekce se hráč dostane při zmáčknutí tlačítka „Play“ v herním menu. Je zde vidět šest oken a jednotlivá okna mají v sobě buď to číslo (číslo mapy) nebo zámek, to znamená že je daná úroveň zamčena a hráč musí dohrát předchozí kolo, aby odemknul tuto úroveň. Jelikož ve hře je celkem dvacet



Obrázek B.3: Menu

kol a oken je zobrazeno pouze šest je pravém dolním rohu tlačítko, kterým zobrazíme dalších šest oken. Po zobrazení dalších šesti oken se v levém dolním rohu zobrazí tlačítko, pomocí něhož se zase vrátíme na původní (viz Obrázek B.3).

B.5 Překážky a protivníci

Dotyk našeho hrdiny s vypsanými protivníky je, až kromě střetnutí se střelcem (Shoter), smrtelné. Po úmrtí musí hráč absolvovat celou úroveň znovu.

Slim

Obrázek B.4: Slim

Slim je protivník, který jde pouze jedním směrem a nevadí mu ani pád do propasti. Jeho zdolání je poměrně jednoduché, ale v některých situací může znepříjemnit hraní (viz Obrázek B.4).

Saw

Obrázek B.5: Saw

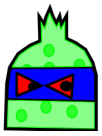
Pila, která se neustále točí se může vyskytovat ve vodorovné poloze anebo na stěnách. Ve vodorovné poloze se dá překonat nejlépe dvoj-skokem anebo skočením s velkým náklonem telefonu. Na stěnách jsou už pily o dost menší (viz Obrázek B.5).

Bayonet

Obrázek B.6: Bayoner

Jedná se o jednoduchou snadno překonatelnou překážku. Vyskytuje se na místech kdy je potřeba hráči omezit seskok dolů anebo ztížit danou část mapy(viz Obrázek B.6).

Bottle



Obrázek B.7: Bottle

Tento protivník střílí v pravidelných intervalech nad sebe střely. Jednotlivé střely mají pokaždé náhodnou x-ovou a y-ovou rychlost(viz Obrázek B.7).

Slarm

Obrázek B.8: Slarm

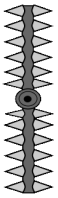
Pohybuje se jak doprava tak doleva v určité vzdálenosti. V pravidelných časových intervalech vystřelí do obou stran. Hráč může poznat kdy Slarm střílí tím, že Slarm se před vystřelením přikrčí(viz Obrázek B.8).

Shoter

Obrázek B.9: Shoter

Neustále sleduje hlavního hrdinu. Když je hrdina v dosahu tak po něm vystřelí rychlou ránu a chvíli nabíjí, poté střílí znovu. Jako u jediného protivníka se o něj nemůže Tiny zranit(viz Obrázek B.9).

Reaper



Obrázek B.10: Reaper

Neustále se točící tyč s bodáky. Hráč tedy musí dobře vyčasovat kdy tuto překážku přeskočit. Ve hře se většinou vyskytují až tři za sebou pro obtížnější zdolávání(viz Obrázek B.10).

B.6 Bossové

Každá desátá úroveň ve hře je určená pro překonání bosse. V každé mapě jsou objekty pomocí niž lze bosse poranit anebo i zničit.

První Boss

Obrázek B.11: První Boss

Ohromný boss usídlený uprostřed mapy, který neustále okolo sebe střílí střely. Hráč zde má za úkol vyskákat jak na pravé tak i na levé straně mapy nahoru. Nahoře se hráč dostane na plošinu kde je také chráněn proti střelám. Na konci této plošiny je bodák, který hráč musí shodit na bosse (tím že na něj skočí) a tím bosse zranit (viz Obrázek B.11).

Po zranění boss ztratí půlku svého těla (kde byl trefen) a zároveň na danou stranu již nemůže střílet a uživatel se může v klidu dostat dolů.

Po zranění na obou stranách boss umírá a hráč vyhrává hru.

Druhý Boss

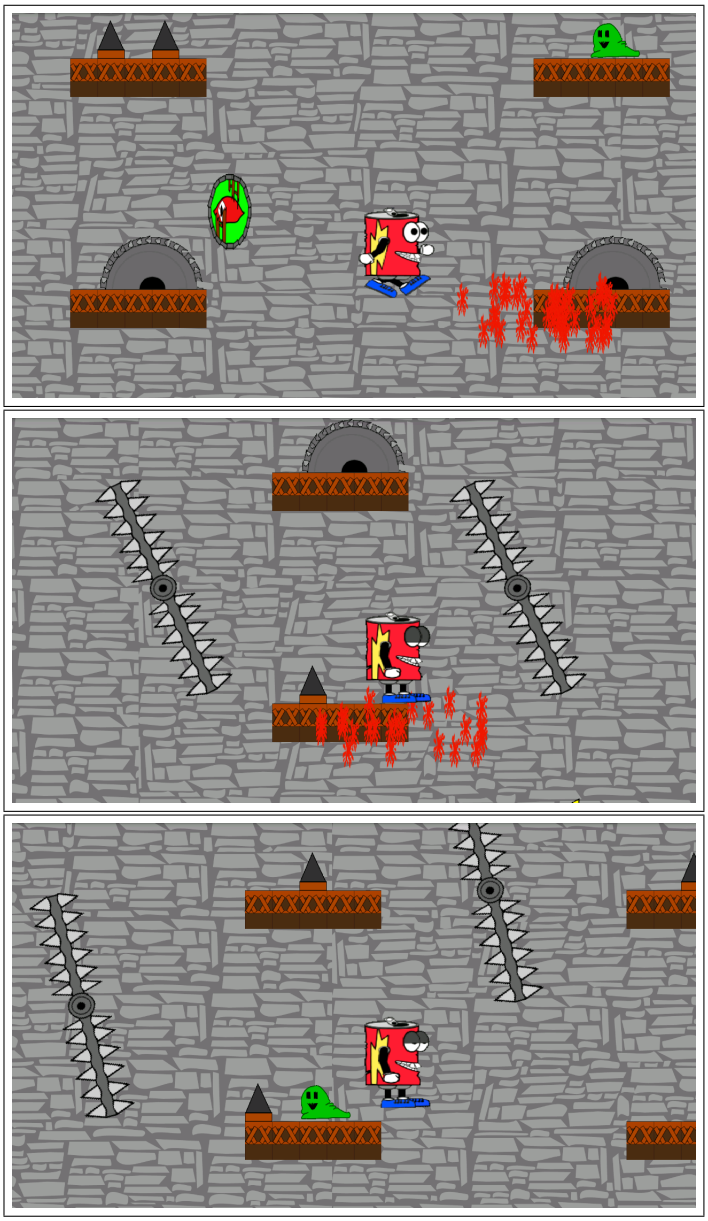
Mapa úrovně se skládá z několika pater. Hráč spolu s Bossem začíná na vrchním patře. Boss (viz Obrázek B.12) je dvakrát větší než hráčova postava. Boss neustále skáče ze strany na stranu. Při dopadu zničí pod sebou část podlahy patra. Úkolem hráče je neustálé se vyhýbání a snažení se nespadnout do proprasti pod poslední podlahou dříve, než Boss. Když spadne Boss do propasti dříve než hráč, tak hráč vyhrává danou úroveň.

Obrázek B.12: Druhý Boss

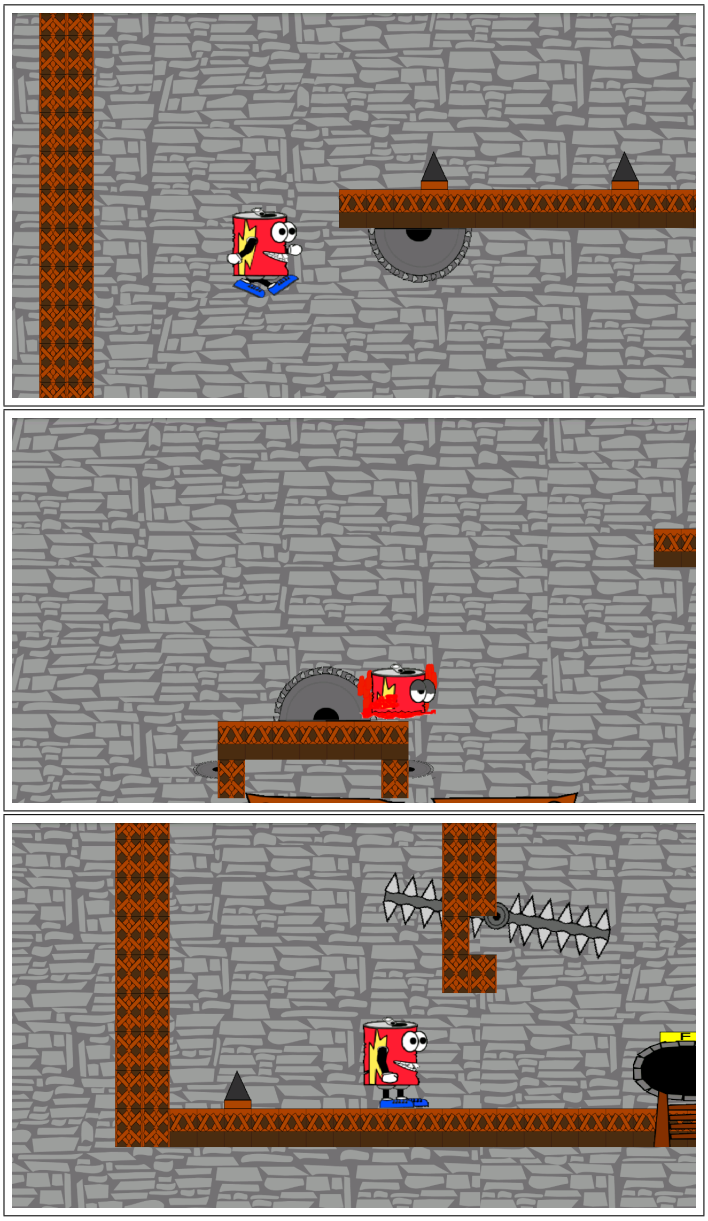
Tento Boss je finální části mé hry. Po dokončení této úrovně je hra dohrána a hráči se objeví obrazovka s gratulací a poděkováním.

C ScreenShoty

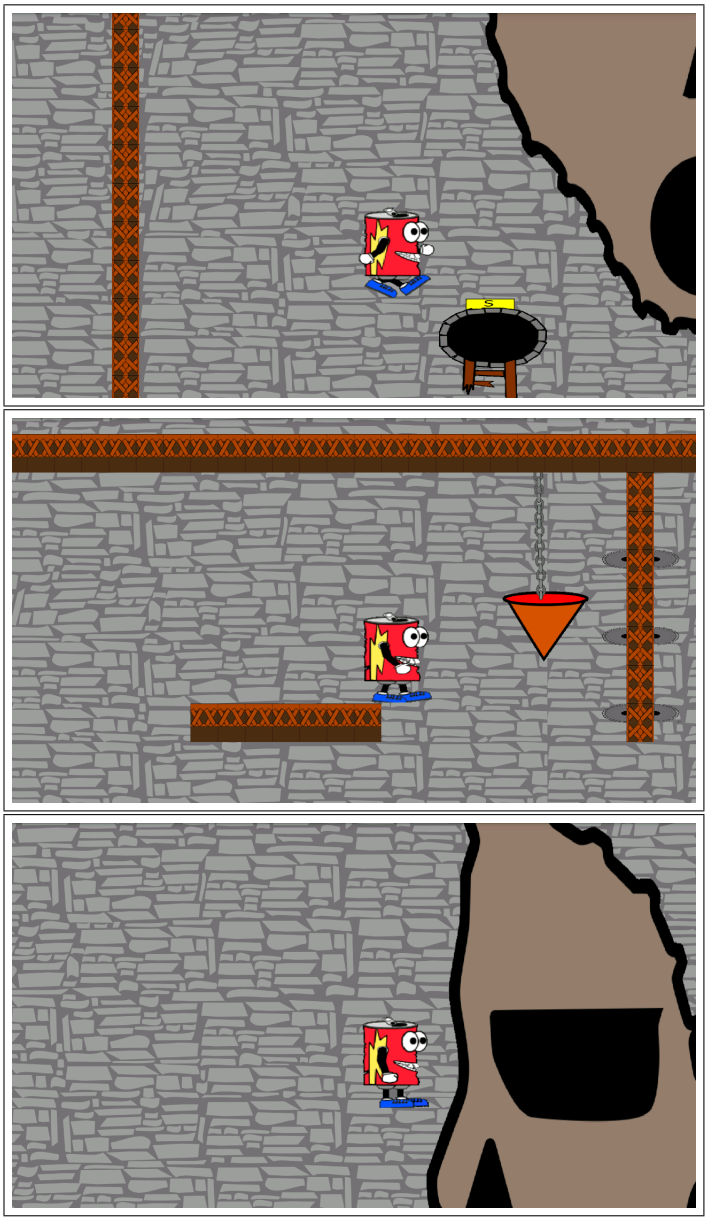
- Úroveň 7.



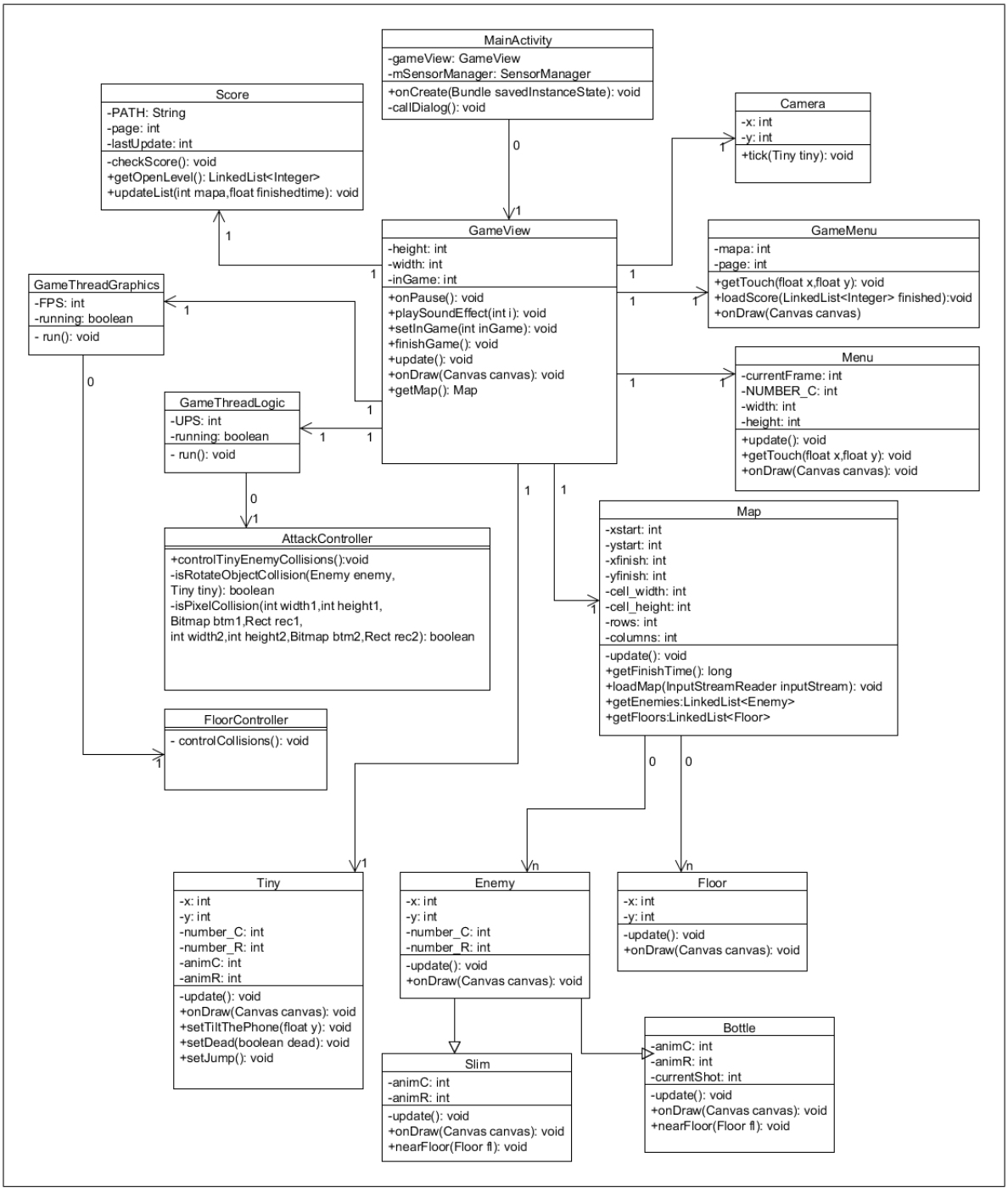
- Úroveň 12.



- První Boss.



D UML diagram nejvýznamnější tříd



E Struktura přiloženého CD

1. *doc:* - obsahuje dokument bakalářské práce ve formátu PDF.
2. *projekt:* - Projekt aplikace v Eclipse (zdrojové kódy, textury, apod.).
3. *apk:* - má v sobě dva soubory Tiny.apk a TinyTest.apk, pomocí nich můžeme nainstalovat aplikaci na mobilní zařízení. TinyTest.apk slouží k testování a po nainstalování dostaneme hru, která má od začátku dostupné všechny úrovně.
4. *tex:* - Zdrojový kód dokumentu se seznamem literatury a obrázky.
5. *Javadoc:* - Javadoc dokumentace k projektu.