České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Eldar Gabidullin**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Sopwith - retrohra pro Android**

Pokyny pro vypracování:

Navrhněte, naprogramujte, nasaďte, otestujte a dokumentujte retrohru pro telefon na platformě Android. Hra bude vycházet jednoduchostí grafiky i ovládaní z hry Sopwith (http://www.dosgamesarchive.com/download/sopwith). Vývoj provádějte iterativně.

Seznam odborné literatury:

[1] LARMAN, Craig a Chris RUPP. Applying UML and patterns: introduction to object-oriented analysis and design and iteractive development. 3rd ed. New Jersey: Prentice-Hall, 2005, xviii, ISBN 01-314-8906-2.
[2] FOWLER, Martin. Destilované UML. Praha: Grada, 2009, 173 s. ISBN 978-80-247-2062-3.
[3] Ken Schwaber and Mike Beedle. 2001. Agile Software Development with Scrum (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

Vedoucí: Ing. Martin Komárek

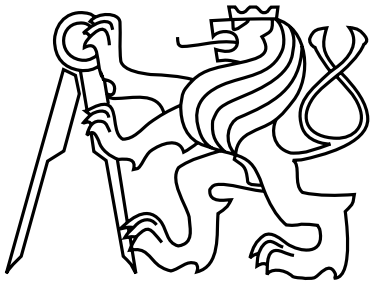
Platnost zadání: do konce letního semestru 2015/2016

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 12. 11. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Sopwith - retrohra pro Android

Eldar Gabidullin

Vedoucí práce: Ing. Martin Komárek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

6. ledna 2015

Poděkování

Rád bych poděkoval Martinu Komárkovi za vedení mé bakalářské práce, Ondřeji Mackovi za jeho cenné rady a svým kolegům za pomoc při testování aplikace. V neposlední řadě bych také chtěl velice poděkovat své rodině a přátelům za jejich podporu v období mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 5. 1. 2015 .....

Abstract

The bachelor thesis deals with creation of a game designed for the Android platform. This game has been inspired by the original game called Sopwith that was created in 1984. This piece of work contains three basic parts. The game has been step by step documented, implemented and finally deployed into the Google Play service in these parts.
The objective of this project is the iterative analysis, design, implementation and testing. The implementation of the game is divided into eleven particular iterations. In each of these parts, new functions and properties are added and so all the game is gradually evolved.
The goal of my work is to create a game which is as most as possible similar to the origin Sopwith videogame and bring back the memories to the fans of the game while playing it.

Abstrakt

Bakalářská práce se zabývá vytvořením hry pro platformu Android. Tato hra je inspirována původní hrou nazvanou Sopwith, která byla vytvořena roku 1984. Práce obsahuje tři základní části, ve kterých je postupně zdokumentována, naimplementována a finálně je funkční hra nasazena do služby Google Play.
Předmětem tohoto projektu je iterativní analýza, návrh, implementace a testování. Implementace samotné hry je rozdělena do jedenácti jednotlivých iterací. V rámci každé iterace dojde k přidávání nových funkcionalit a vlastností hry, a tím k jejímu postupnému rozvoji.
Cílem mé práce je vytvoření hry tak, aby byla co nejvíce podobná klasické videohře Sopwith a umožnila tak svým fanouškům při jejím hraní oživit vzpomínky na danou „retro“ hru.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

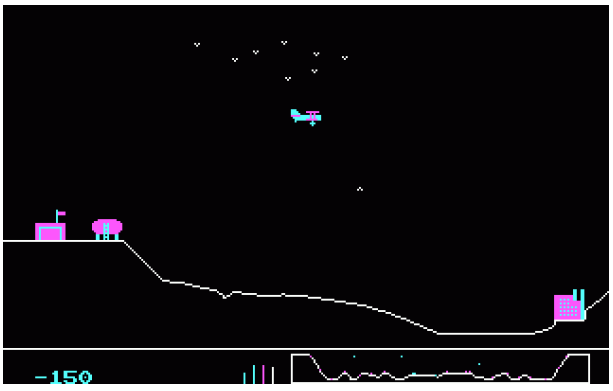
1.1 Záměr projektu

Projekt je zaměřen na vytvoření videohry pro platformu Android. Jedná se o „remake“ staré dvojrozměrné hry Sopwith. Důraz je kladen zejména na záchranu atmosféry originální hry, která zahrnuje jednoduchost, ohraničenou barevnou škálu a specifické ovládání.

Sopwith je hra ve stylu „side scrolling“, byla vytvořena Davidem L. Clarkem ze společnosti „BMB Compuscience“ v roce 1984. Původní hra byla spustitelná na platformě IBM PC. Ve hře uživatel řídí letadlo - biplan, cílem této hry je zničení nepřátelských letadel a budov. Více informací o původní videohře je možno získat na webové stránce wikipedie [16].

1.2 Cíl a obsah projektu

Cílem projektu je iterativně zanalyzovat, navrhnout, implementovat, otestovat a nasadit dokončenou hru do služby Google Play. Hra bude funkcionálně podobná druhé verzi originální hry - Sopwith II (viz [13]).



Obrázek 1.1: Snímek originální hry Sopwith II

1.3 Akceptační kritéria

- Projekt bude řádně zdokumentovaný.
- Projekt bude dokončen včas.
- Projekt bude nasazen do služby Google Play.
- Počet stažení ve službě Google Play bude větší než nula.

1.4 SWOT

1.4.1 Silné stránky

- Zkušenosti s vývojem her
- Zkušenost v syntéze zvuku

1.4.2 Slabé stránky

- Ohraničená znalost platformy Android
- Neznalost použitého herního frameworku
- Časové vytížení

1.4.3 Příležitosti

- Absence dobré verze na Google Play umožňuje přitáhnout milovníky originální hry.

1.4.4 Hrozby

- Možnost nezájmu uživatelů Google Play.
- Možnost zákazu aplikace na Google Play z důvodu porušení autorských práv.

Kapitola 2

Rešerše

2.1 Výběr frameworku

Důležitou částí při plánování budoucí hry je výběr správného nástroje. Hra bude určena pro platformu Android, proto hlavním kritériem výběru je podpora Android.

1. **OpenGL ES** - rozhraní pro vykreslování 2D a 3D grafiky, akcelerované za pomoci GPU a určené pro malé vestavěné systémy (více informací [11]). Je zároveň nejrychlejším a nejsložitějším nástrojem, umožňuje zpracování grafiky na nižší úrovni. Tento nástroj ovšem není vhodný, protože jeho nastudování i samotné použití je časově náročné.

2. **Unity 3D** - profesionální multiplatformní framework, určený na 3D hry (příp. 2D)(viz [15]). Podporuje velké množství platforem: Android, iOS, Windows, OS X. Pro akceleraci grafiky používá OpenGL a Direct X, také nabízí speciální vývojové prostředí. Pro použití tohoto frameworku je však nutná znalost jazyka C#. Tento framework poskytuje rozsáhlou funkcionalitu, která není příliš vhodná pro vývoj graficky jednoduché hry jako je Sopwith.

3. **Cocos2d-x** - open source multiplatformní framework, určený na 2D hry. Disponuje vlastním vývojovým prostředím a podporuje sadu platforem. Pro použití tohoto frameworku je vyžadována znalost jazyka C++ (viz [3]).

4. **AndEngine** – open source framework pro 2D hry, určený pouze pro platformu Android. Poslední verze používá OpenGL ES 2.0, proto je výkonný, navíc není nutná znalost OpenGL. Samotný framework není příliš rozsáhlý, nicméně podporuje sadu rozšíření jako například Box2D (viz [1]). AndEngine je nativní Java Android knihovnou a díky mé rozsáhlé znalosti jazyku Java tento framework považuji za vhodný nástroj pro použití.

Kapitola 3

Plán projektu

3.1 Základní části projektu

Projekt je rozdělen na několik základních částí:

- Dokumentace
- Implementace
- Nasazení

3.1.1 Dokumentace

Dokumentace obsahuje textové výstupy projektu. V každé iteraci je podrobně popsáno zadání, analýza, návrh, řešení a průběh uživatelského testování.

3.1.2 Implementace

Implementace hry pro platformu Android. Plán jednotlivých iterací je uveden v sekci 3.2.

3.1.3 Nasazení

Konečné nasazení aplikace do služby Google Play.

3.2 Plán iterací

Projekt bude kompletně vypracován po jednotlivých iteracích. Více informací o iterativním postupu naleznete v knize [19].

3.2.1 1. Iterace

- Jednoduchý reliéf krajiny (ve tvaru lomené přímky).
- Letadlo v podobě čtverce umístěné uprostřed displeje.
- Budou implementována 2 tlačítka určená pro posun krajiny doprava a doleva (podle počtu stisků se bude měnit rychlost pohybu, například 5 úrovní rychlosti), dále pak 2 tlačítka pro posun letadla nahoru a dolů.
- Kontrola letadla tak, aby se pohybovalo v prostoru mezi oblohou a reliéfem krajiny. Pokud se ve svém pohybu letadlo dostane k těmto mezníkům, dojde k jeho zastavení.

3.2.2 2. Iterace

- Letadlo v původní podobě čtverce bude nahrazeno obrázkem zjednodušeného letadla, u kterého bude jasně zřejmá přední a zadní část, stejně tak i vrchní a spodní část letadla.
- Tlačítka "doleva" a "doprava" budou implementována jako "zrychlení" a "zpomalení". Tlačítka "nahoru" a "dolů" budou nahrazena změnou směru letu, tj. "doleva" a "doprava".
- Bude doplněna funkce otočení letadla kolem vlastní osy o 180 stupňů.

3.2.3 3. Iterace

- Fyzický model posunu letadla.
- Při nedostatečné rychlosti letadla nebo po střetu letadla s oblohou dojde k jeho volnému pádu kolmo k zemi. Uživatel může letadlo před pádem zachránit použitím tlačítek pro změnu rychlosti nebo směru letu. Pokud tak neudělá, v okamžiku ztroskotání letadla hra začíná od začátku.

3.2.4 4. Iterace

- Letadlo bude moci střílet a házet bomby.
- Jakmile se bomba při dopadu dotkne krajiny, v daném okamžiku zmizí. Počet bomb bude omezený.
- Letadlo může být zasaženo a zničeno svou vlastní bombou, například v případě jeho otočení kolem své osy.

3.2.5 5. Iterace

- V krajině se budou nacházet domy a tanky, které budou součástí nepřátelských i vlastních jednotek.
- Bude vytvořena detekce střetu bomb a nábojů s domy a tanky. Při zásahu dojde k jejich destrukci a případnému přičtení či odečtení bodů.
- Při dopadu bomby kamkoliv v krajině dojde k její explozi. Střepiny mohou zasáhnout a zničit letadlo.

3.2.6 6. Iterace

- V krajině se budou nacházet krávy. V případě zabití krávy dojde k odečtení bodů.
- Krajina bude rozšířena o ptáky. Pokud dojde ke střetu letadla s ptákem, letadlo bude zničeno a hra začíná od začátku.

3.2.7 7. Iterace

- Nepřátelské tanky budou střílet na letadlo. Při zásahu letadla hra začíná od začátku.

3.2.8 8. Iterace

- Aplikace bude rozšířena o startovní menu. Toto menu bude obsahovat zatím jedinou možnost hry, a to hru bez nepřátelských letadel.

3.2.9 9. Iterace

- V této iteraci budou do aplikace přidány zvuky. Viz ukázkové video se zvukem[4].

3.2.10 10. Iterace

- Implementace nepřátelských letadel, která budou řízena samotnou aplikací. Tato letadla umí střílet náboje a budou se snažit se zničit letadlo, které řídí hráč.
- Startovní menu bude rozšířeno o další možnost.

3.2.11 11. Iterace

- Možnost hry mezi dvěma hráči přes Bluetooth[2].
- Startovní menu bude doplněno o další funkcionalitu.

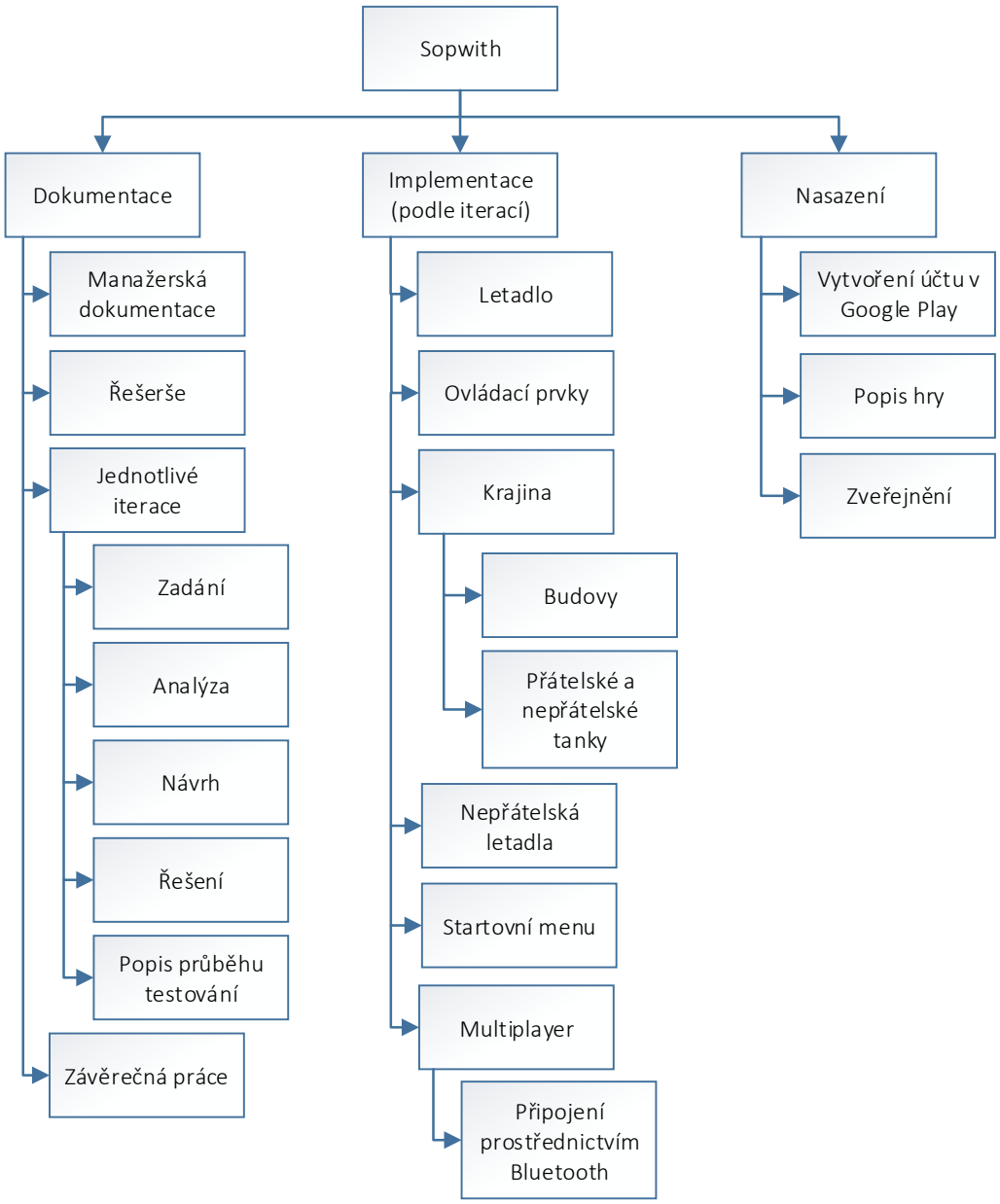
3.3 Harmonogram projektu

Níže v tabulce je popsán časový plán projektu. Celý projekt musí být dokončen do 7.12.2014, tedy do konce 11. týdne semestru. Iterace se realizují postupně. Každá iterace zahrnuje analýzu, návrh, implementaci a testování.

Část projektu	**Započetí [týden]**	**Finalizace [týden]**
Manažerská dokumentace	2	4
Rešerše	3	5
1. iterace: Krajina a letadlo	3	5
2. iterace: Řízení letadla	3	5
3. iterace: Fyzika letu letadla	3	5
4. iterace: Bomby a náboje	4	6
5. iterace: Domy a tanky	4	6
6. iterace: Krávy a ptáci	4	6
7. iterace: Střelba nepřátelských tanků	5	7
8. iterace: Startovní menu	5	7
9. iterace: Zvuky	5	7
10. iterace: Nepřátelská letadla	6	8
11. iterace: Multiplayer	7	9
Závěrečná práce	8	10
Nasazení do služby Google Play	9	10

Tabulka 3.1: Harmonogram projektu

3.4 WBS



Obrázek 3.1: WBS diagram

3.5 Rizika

3.5.1 Vyčerpání člena týmu

- **Pravděpodobnost:** Střední
- **Závažnost:** Velká
- **Řešení:** Vzhledem k tomu, že na projektu pracuji samostatně, hrozí fyzické i psychické vyčerpání z intenzivní činnosti na tomto projektu. Řešením vzniklé situace je odpočinek, relaxace.
- **Protiopatření:** Pravidelný odpočinek a relaxace, fyzické aktivity jako například sport.

3.5.2 Nedostupnost zařízení pro testování

- **Pravděpodobnost:** Malá
- **Závažnost:** Střední
- **Řešení:** Použití emulátoru
- **Protiopatření:** Disponovat vlastním zařízením.

3.5.3 Časová tíseň

- **Pravděpodobnost:** Velká
- **Závažnost:** Velká
- **Řešení:** Revize harmonogramu projektu
- **Protiopatření:** Vytvoření reálného časového plánu, pravidelná kontrola iterací a průběhu projektu.

3.5.4 Nedokončení všech určených iterací

- **Pravděpodobnost:** Velká
- **Závažnost:** Velká
- **Řešení:** Revize harmonogramu projektu, zajištění možnosti delšího termínu odevzdání projektu, akceptace sníženého hodnocení za odevzdaný projekt.
- **Protiopatření:** Pravidelná a systematická práce na projektu, dodržení naplánovaného harmonogramu.

3.6 Rozpočet projektu

3.6.1 Finanční plán

Na tomto projektu pracuji v rámci své bakalářské práce a předmětu Řízení softwarových projektů (A7B36SI2), proto není nijak finančně ohodnocen. Úspěšná finalizace projektu bude zároveň znamenat významný krok na cestě k úspešně ukončenému bakalářskému studiu.

3.6.2 Odhad časové pracnosti

V níže uvedené tabulce je zobrazen odhad časové pracnosti vypracování jednotlivých částí tohoto projektu.

Část projektu	**Čas [hod]**
Manažerská dokumentace	30
Rešerše	30
1. iterace: Krajina a letadlo	20
2. iterace: Řízení letadla	20
3. iterace: Fyzika letu letadla	30
4. iterace: Bomby a náboje	20
5. iterace: Domy a tanky	20
6. iterace: Krávy a ptáci	20
7. iterace: Střelba nepřátelských tanků	20
8. iterace: Startovní menu	20
9. iterace: Zvuky	30
10. iterace: Nepřátelská letadla	30
11. iterace: Multiplayer	30
Závěrečná práce	30
Nasazení do služby Google Play	4
Celkem	**354**

Tabulka 3.2: Předpoklad časové pracnosti jednotlivých částí projektu

3.6.3 Rozpočet

Dle odhadu časové pracnosti vypracování jednotlivých částí projektu (viz tabulka 3.2) předpokládám, že projekt zabere celkem 354 hodin. Hodinová sazba mé práce je 300 Kč. Celková cena tedy vychází na 106.200 Kč.

3.6.4 Termín dokončení

Projekt bude dokončen do 7.12.2014.

Kapitola 4

1. Iterace

4.1 Zadání

V 1. iteraci se dle plánu budeme zabývat řešením níže uvedených záležitostí:

- Jednoduchý reliéf krajiny (ve tvaru lomené přímky).
- Letadlo v podobě čtverce umístěné uprostřed displeje.
- Budou implementována 2 tlačítka určená pro posun krajiny doprava a doleva (podle počtu stisků se bude měnit rychlost pohybu, například 5 úrovní rychlosti), dále pak 2 tlačítka pro posun letadla nahoru a dolů.
- Kontrola letadla tak, aby se pohybovalo v prostoru mezi oblohou a reliéfem krajiny. Pokud se ve svém pohybu letadlo dostane k těmto mezníkům, dojde k jeho zastavení.

4.2 Analýza

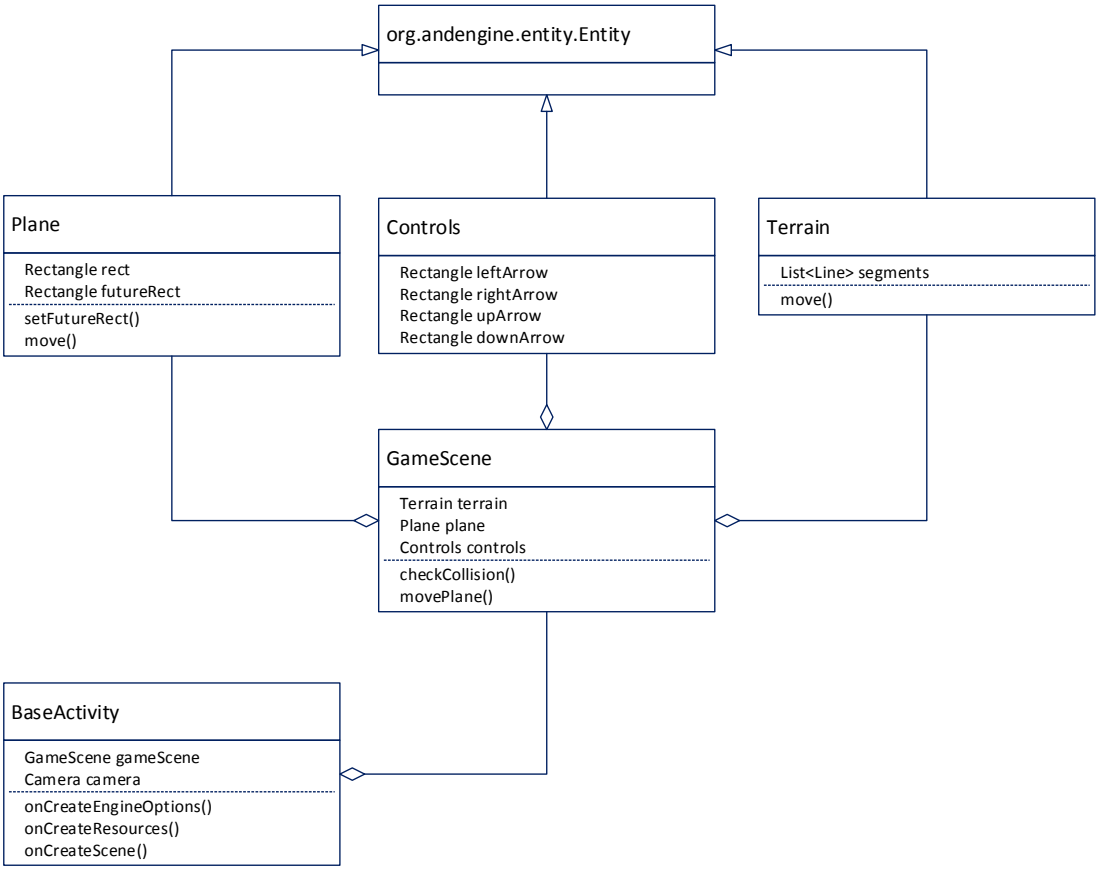
4.2.1 Krajina a letadlo

Definujeme velikost a tvar krajiny. V originální hře z roku 1984 má krajina tvar lomené přímky, která je umístěna v dolní třetině displeje zařízení, na kterém ji hráč spustí (viz [4]). Na obou koncích krajiny přímka stoupá tak, že ohraničuje krajinu. Výška krajiny je shodná s výškou displeje a její délka se rovná čtyřnásobku délky displeje hráčova zařízení.

V rámci první iterace bude letadlo prezentováno jenom malým čtvercem.

4.3 Návrh

4.3.1 Diagram tříd



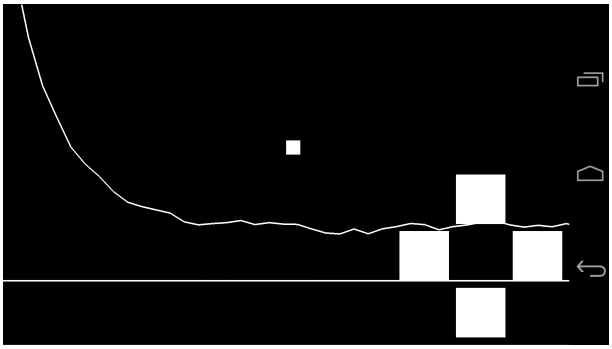
Obrázek 4.1: Diagram tříd

4.4 Implementace

4.4.1 Správa zdrojového kódu

Správa zdrojového kódu je realizována v Git-repozitáři (více informací na [8]). Vývoj všech iterací je realizován v *devel* větvi. Pro rozdělení commitu po iteracích používám Git-tagy tak, že v historii změn mohu jednotlivé iterace snadno rozdělit mezi sebou. Zde je odkaz na Git-repozitař[1].

[1] <https://bitbucket.org/indiegate/sopwith>



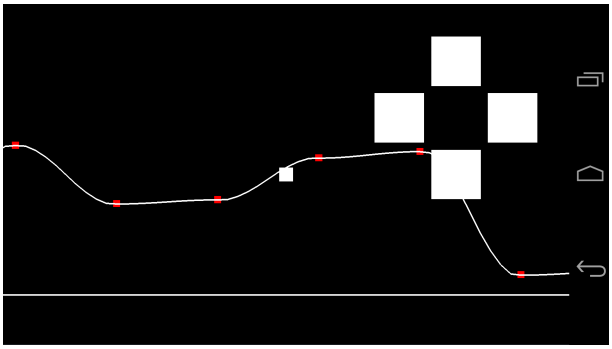
Obrázek 4.2: Snímek hry po 1. iteraci

4.4.2 Krajina

Krajina je tvořena množinou menších úseků[2], které na sebe navazují a zároveň se souběžně posouvají při letu letadla. Souřadnice vrcholů každého úseku se generují v několika krocích:

1. Náhodné vygenerování určitého množství základních bodů tak, aby byly rozmístěny přes stejný horizontální interval;
2. Mezi základními body se plynule vygenerují ostatní body dotvářející krajinu.

Tímto jsem dosáhl efektu, kdy krajina oplývá výšinami a nížinami.



Obrázek 4.3: Snímek hry, kde jsou označeny základní body

4.4.3 Řízení

Dotyky jsou zachycené pomocí metody **onSceneTouchEvent()** třídy **GameScene**.

```
@Override
public boolean onSceneTouchEvent(
    Scene pScene,
    final TouchEvent pSceneTouchEvent) {
```

[2]Objekt **org.andengine.entity.primitive.Line**

```
    synchronized (this) {
        if (controls.leftArrow.contains(
                    pSceneTouchEvent.getX(),
                    pSceneTouchEvent.getY()
                    )){
                        ...
                }
    }
    ...
    return true;
}
```

4.4.4 Řešení problémů

Výsledný *.apk* soubor aplikace jsem uložil do sekce Downloads služby Bitbucket. V mobilním zařízení se z tohoto umístění bohužel nepodařilo stažený soubor úspěšně nainstalovat. Proto jsem tuto situaci řešil změnou úložiště - instalační soubory jsou nyní ukládány do úložiště Dropbox.com.

4.5 Testování

V průběhu testování došlo k úspěšnému otestování všech testovacích scénářů, hra v každé situaci reagovala přesně podle požadavků zadání. Testování této iterace bylo tedy uzavřeno jako úspěšné.

Kapitola 5

2. Iterace

5.1 Zadání

Podle plánu bude v této iteraci vyřešeno:

- Letadlo v původní podobě čtverce bude zobrazováno obrázkem zjednodušeného letadla, u kterého bude jasně zřejmá přední a zadní část, stejně tak i vrchní a spodní část letadla.
- Tlačítka "doleva" a "doprava" budou implementována jako "zrychlení" a "zpomalení". Tlačítka "nahoru" a "dolů" budou nahrazena změnou směru letu, tj. "doleva" a "doprava".
- Bude doplněna funkce otočení letadla kolem vlastní osy o 180 stupňů.

5.2 Analýza

5.2.1 Změna směru letu

Pro změnu směru letu letadla je potřeba vypočítat nové souřadnice podle úhlu letu. Použijeme jednoduchý vzoreček[17]:

$$x_1 = x_0 + \cos\alpha * v$$

$$y_1 = y_0 + \sin\alpha * v$$

kde α je úhel náběhu letadla, a v je rychlost.

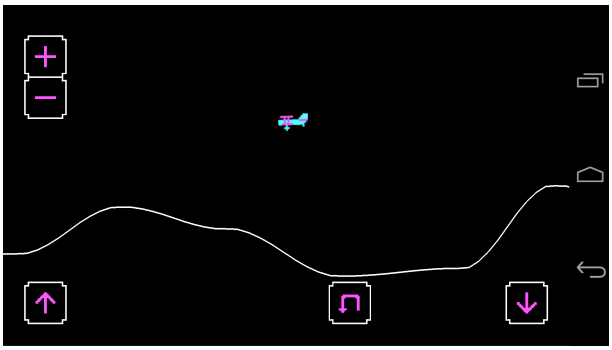
5.3 Návrh

5.3.1 Grafický návrh prvků

Obrázek 5.1: Grafický návrh letadla

Obrázek 5.2: Grafický návrh tlačítek

5.4 Implementace



Obrázek 5.3: Snímek hry po 2. iteraci

5.4.1 Pohybující se kamera

Pro usnadnění implementace pohybu entit jsem použil třídu **BoundCamera** z knihovny AndEngine. Návod na použití je uveden v knize [18]. V předchozí iteraci se letadlo mohlo pohybovat pouze nahoru, dolů a krajina se pohybovala směrem doleva, doprava. V současném stavu se krajina nepohybuje vůbec a kamera sleduje pohybující se letadlo.

```
BoundCamera = new BoundCamera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
boundCamera.setBounds(-Terrain.TERRAIN_WIDTH/2,
                      BaseActivity.CAMERA_HEIGHT/2,
                      Terrain.TERRAIN_WIDTH/2,
                      BaseActivity.CAMERA_HEIGHT/2);
```

```
boundCamera.setBoundsEnabled(true);
boundCamera.setChaseEntity(plane);
```

5.4.2 Detekce kolizí

Abych mohl zkontrolovat kolizi a předejít jí zastavením letadla, přidal jsem další neviditelnou entitu. Jedná se o čtverec, který se otáčí a pohybuje spolu s letadlem. Když chci nastavit letadlu novou pozici, nejdříve nastavím danou pozici neviditelnému čtverci. V případě, že v takovém okamžiku dojde k detekci kolize čtverce s krajinou, dojde k okamžitému zastavení letadla.

5.4.3 Otočení kolem vlastní osy

Pro otočení letadla kolem vlastní osy nepoužívám doplňující obrácený obrázek, ale původní obrázek obrátím během letu pomocí funkce **setScale()** knihovny AndEngine.

```
public void roll(){
    if (rolled) {
                planeSprite.setScale(1,1);
        } else {
                planeSprite.setScale(1, -1);
        }
        rolled = !rolled;
}
```

5.5 Testování

Během testování se vyskytl problém s velikostí tlačítek. Jsou příliš malá na dotyk a občas dojde k situaci, že se prst uživatele dotýká oblasti mimo uvedené tlačítko. Tento problém bude vyřešen v rámci další iterace.

Kapitola 6

3. Iterace

6.1 Zadání

Ve 3. iteraci bude hlavní náplní řešení níže uvedený seznam funkcionalit. První dva body jsou dle původního zadání, třetí bod vyplynul z testování druhé iterace.

- Fyzický model posunu letadla.
- Při nedostatečné rychlosti letadla nebo po střetu letadla s oblohou dojde k jeho volnému pádu kolmo k zemi. Uživatel může letadlo před pádem zachránit použitím tlačítek pro změnu rychlosti nebo směru letu. Pokud tak neudělá, v okamžiku ztroskotání letadla hra začíná od začátku.
- Z výsledků testování předchozí iterace je třeba vyřešit nedostatečnou velikost oblasti určené pro dotyk tlačítek.

6.2 Analýza

6.2.1 Fyzika letu

V reálném světě letadlo ovlivňují 4 síly[17]:

- odporová aerodynamická síla,
- vztlaková aerodynamická síla,
- zemská tíže,
- tažná síla motoru.

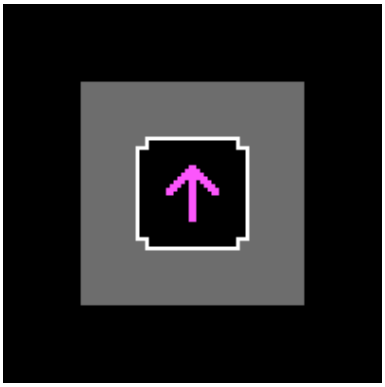
Ve hře je nejen zbytečné, ale i nežádoucí implementovat chovaní reálného světa, jelikož hra ztratí svou jednoduchost ovládaní. Proto na letadlo budou mít vliv jenom dvě síly, tedy zemská tíže a tah motoru. Bez vztlakové síly letadlo ihned spadne, proto je třeba udělat omezení - letadlo nebude ovlivňováno zemskou tíží v úhlech, které jsou závislé na rychlosti letadla. Například při úhlu náběhu 0° stačí minimální tah motoru pro udržení letadla ve vzduchu. Při úhlu náběhu 30° musí letadlo mít maximální tlak motoru. Přesné parametry je potřeba nastavit empirickým způsobem.

6.2.2 Záchrana letadla při pádu

Pokud hrozí pád letadla, v originální hře má hráč možnost jej ještě zachránit. V takovém případě je nutné letadlo otočit přední částí dolů. Je třeba podotknout, že letadlo se při pádu nekontrolovatelně otáčí do stran. V rámci tohoto projektu je daná funkcionalita ve hře implementována stejným způsobem.

6.3 Návrh

6.3.1 Návrh úpravy tlačítek



Obrázek 6.1: Schéma dotykové oblasti. Uprostřed - tlačítko, šedý čtverec - neviditelná oblast dotyku

6.4 Implementace

6.4.1 Řešení problémů

Při nastavování parametrů jsem na displej přidal několik řádků textu, které zobrazují aktuální fyzické parametry letadla: horizontální a vertikální rychlost, a také akceleraci letadla a úhel náběhu.

6.5 Testování

V průběhu testování jsem zjistil, že chybí zpětná vazba jednotky řízení. Je důležité znát aktuální tah motoru letadla. Dále chybí reakce na zmáčknutí tlačítek. Velikost oblasti určené pro dotyk tlačítek, která byla v rámci této iterace upravena, se nyní jeví jako zcela vyhovující.

Kapitola 7

4. Iterace

7.1 Zadání

Ve 4. iteraci se budeme zabývat řešením níže uvedených záležitostí:

- Letadlo bude moci střílet a házet bomby.
- Jakmile se bomba při dopadu dotkne krajiny, v daném okamžiku zmizí. Počet bomb bude omezený.
- Letadlo může být zasaženo a zničeno svou vlastní bombou, například v případě jeho otočení kolem své osy.
- Z předešlé iterace je nutné vytvořit zpětnou vazbu jednotky řízení.

7.2 Analýza

7.2.1 Střílení

Vzhledem k tomu, že tlačítek je na displeji již dostatečný počet, je nežádoucí implementace samostatného tlačítka určeného pro funkci střílení. Mohlo by dojít k situaci, kdy se uživatel začne v množství tlačítek ztrácet a hra tak ztratí svou jednoduchost. Funkce střílení se aktivuje vždy, jakmile dojde ke stlačení displeje herního zařízení kdekoli v místech mimo existující tlačítka.

Po stlačení displeje za účelem spuštění funkce střelby se před letadlem budou objevovat vystřelené náboje ve tvaru malých bílých čtverečků. Náboje létají ve směru, kterým letělo letadlo v momentě vystřelení a zaniknou po určitém čase nebo po střetu s jiným objektem (krajina, dům, tank, aj.) Čas, po kterém vystřelený náboj zanikne, pokud nedojde ke střetu s jiným objektem, je nutné nastavit empiricky.

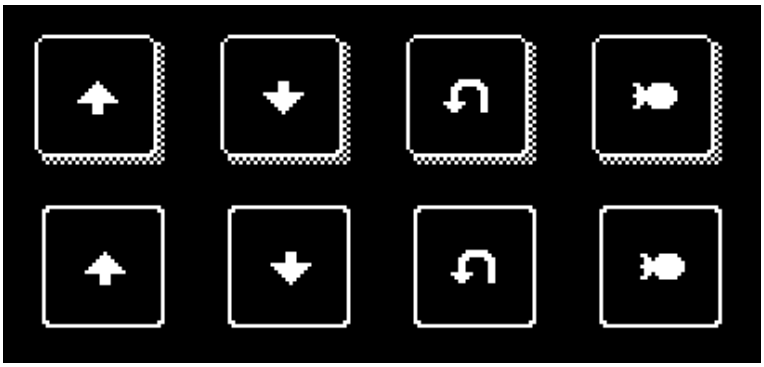
7.2.2 Bomby

V momentě, kdy letadlo vypustí bombu, se tato bomba objeví přímo pod letadlem a její počáteční směr bude totožný se směrem, který mělo letadlo v okamžiku jejího vypuštění. Působením gravitace a setrvačnosti bomba míří přímo k zemi, ovšem nikoli kolmo k zemi.

7.3 Návrh

7.3.1 Návrh obrázků dvoustavových tlačítek

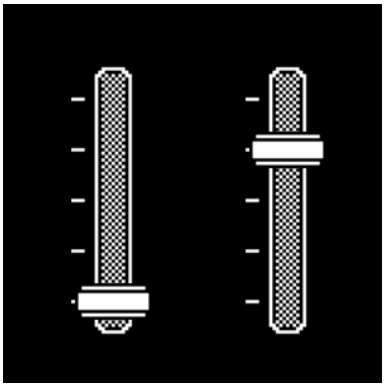
Každé tlačítko bude mít dva stavy: normální a stisknuté. Půjde o grafické rozlišení vzhledu tlačítka. Uživatel tak bude moci velice snadno rozlišit, v jakém stavu se tlačítko v daném okamžiku nachází.



Obrázek 7.1: Grafický návrh tlačítek pro 4. iteraci

7.3.2 Návrh pohybujícího se tlačítka

Jako přínosné považuji vytvořit tlačítko, které bude sloužit ke změně rychlosti. Uživatel bude moci tímto tlačítkem pohybovat na stupnici, kde bude umístěno, podle toho, zda chce rychlost zvýšit nebo snížit. Tlačítko bude prezentováno tzv. "knoflíkem".

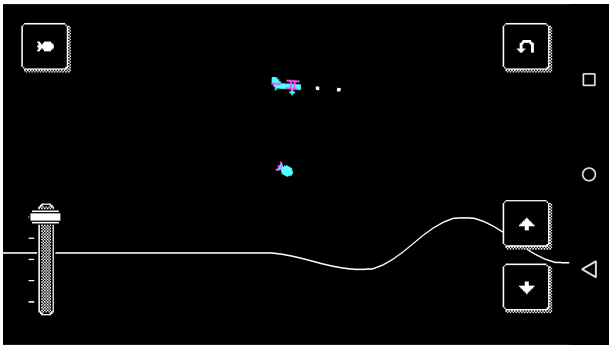


Obrázek 7.2: Grafický návrh pohybujícího tlačítka

7.4 Implementace

7.4.1 Automatické sestavení textur

V případě, kdy je obrázků více, není vhodné pro každý obrázek definovat souřadnice na mapě textur. Proto jsem použil automatické sestavení:



Obrázek 7.3: Snímek hry po 4. iteraci

```
BuildableBitmapTextureAtlas textureAtlas =
    new BuildableBitmapTextureAtlas(
        baseActivity.getTextureManager(), 1024, 1024);
textureAtlas.build(
    new BlackPawnTextureAtlasBuilder<
        IBitmapTextureAtlasSource, BitmapTextureAtlas>(0, 0, 0));
textureAtlas.load();
```

7.4.2 Pool objektů

Pro Java GC je velmi náročné pokaždé vytvářet nový objekt a následně ho mazat z paměti [18]. Z tohoto důvodu jsem vytvořil Pool pro náboje a bomby. Vice informací o objektových Poolech [12].

7.5 Testování

Naimplementovaná zbraň ve formě střílení a vypouštění bomb odpovídá zadání a při testování se projevila jako zcela funkční. Dále byla otestována jednotka řízení, která nyní poskytuje vhodnou zpětnou vazbu uživateli.

Kapitola 8

5. Iterace

8.1 Zadání

V 5. iteraci budeme řešit níže uvedené záležitosti:

- V krajině se budou nacházet domy a tanky, které budou součástí nepřátelských i vlastních jednotek.
- Bude vytvořena detekce střetu bomb a nábojů s domy a tanky. Při zásahu dojde k jejich destrukci a případnému přičtení či odečtení bodů.
- Při dopadu bomby kamkoliv v krajině dojde k její explozi. Střepiny mohou zasáhnout a zničit letadlo.

8.2 Analýza

8.2.1 Krajina

V předchozích iteracích se krajina generovala zcela náhodně. Vzhledem k tomu, že do krajiny potřebujeme přidat množinu objektů a zároveň je velmi důležité jejich správné rozmístění, je v tomto případě žádoucí implementovat načtení souřadnic úseků a jednotlivých objektů ze souboru.

K tomuto účelu bude v implementaci použit formát dat **JSON**. Více o formátu [9].

8.2.2 Druhy objektů

Ve hře se vyskytuje několik druhů objektů: hangáry, sklady paliva (cisterny), továrny a tanky. Všechny objekty, kromě továren, mohou být jak přátelské, tak nepřátelské. Továrny jsou pouze nepřátelské. V rámci této iterace se tanky nebudou pohybovat ani střílet.

8.2.3 Body

V případě zničení nepřátelského objektu hráč získá určitý počet bodů dle typu zničeného objektu [14]. Pokud ovšem dojde ke zničení přátelského objektu, hráči je odečten počet bodů, kterým je zničený objekt hodnocen. Definujeme body podle druhu objektu:

- Letadlo - 50 bodů
- Továrna, tank, hangár - 100 bodů
- Cisterna - 200 bodů

8.3 Návrh

8.3.1 Formát dat

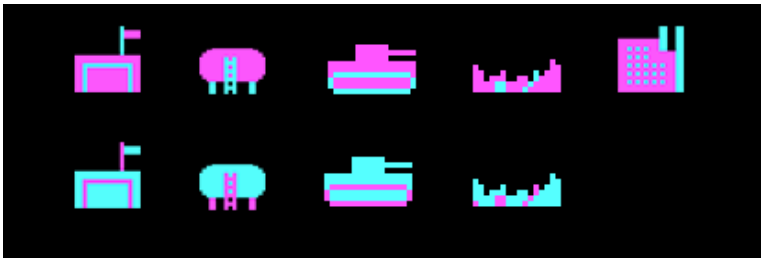
Tvar a parametry krajiny, včetně objektů v ní umístěných, se budou načítat ze souboru *json*. Níže je příklad struktury takového souboru. V první části (*objects*) jsou uvedeny informace typu druh objektu a vlastnost, tj. zda je objekt přátelský či nepřátelský, a také souřadnice každého objektu. Ve druhé části (*points*) jsou uvedeny souřadnice jednotlivých bodů, ze kterých se pak sestaví úseky krajiny.

```
{
    "map": {
        "objects": [
            {
                "type": "factory",
                "role": "enemy",
                "position": [290,420]
            },
            {
                "type": "tank",
                "role": "enemy",
                "position": [495,247]

            },

            ...

        "points": [
            [184,479],
            [208,460],
            [226,441],
            [238,424],
            [251,406],
            [266,398],
            [313,398],
```

```
            ...

        ]
    }
}
```

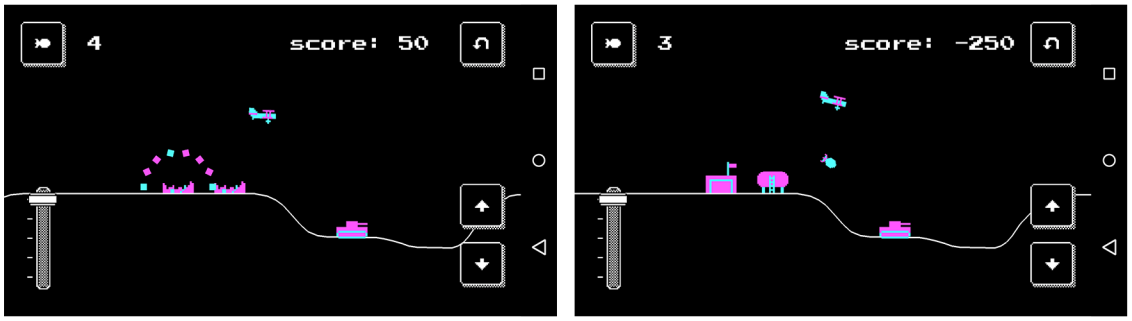
8.3.2 Návrh obrázků objektů

Přátelské i nepřátelské objekty jsou svým tvarem totožné, ke snadnému rozlišení slouží jejich barva. Přátelské objekty mají barvu azurovou, zatímco nepřátelské jsou barvy fialové.



Obrázek 8.1: Obrázky objektů

8.4 Implementace



Obrázek 8.2: Snímky hry po 5. iteraci

8.4.1 Načtení JSON

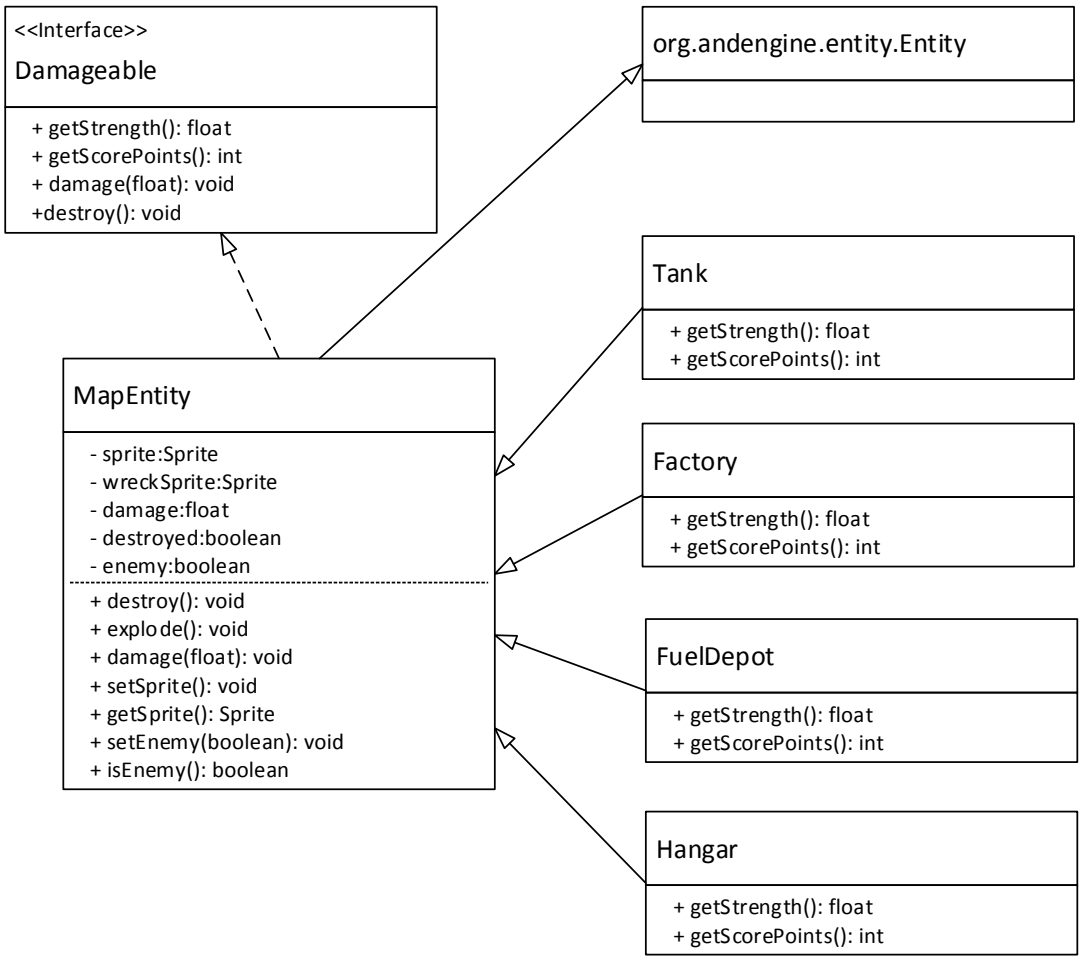
Příklad čtení souřadnic úseků krajiny ze formátu **JSON**. Více informací [10].

```
JSONObject jsonObject = new JSONObject(loadJSONFromAsset("lvl/map.json"));
JSONArray pointsArray = jsonObject.getJSONObject("map").getJSONArray("points");
```

```
int[][] terrainPoints = new int[pointsArray.length()][2];
for (int i = 0; i < pointsArray.length(); i++){
    JSONArray point = pointsArray.getJSONArray(i);
terrainPoints[i][0] = point.getInt(0);
terrainPoints[i][1] = point.getInt(1);
}
map.setPoints(terrainPoints);
```

8.4.2 Nové objekty

Každý nový objekt na mapě je prezentován zvláštní třídou.



Obrázek 8.3: Diagram tříd 5. iterace

8.4.3 Optimalizace

AndEngine lze nastavit tak, že objekty, které již nejsou viditelné na displeji zařízení, se automaticky nebudou vykreslovat[18]. Toto nastavení je defaultně vypnuté.

```
\\zapnutí redukování objektu mapEntity
mapEntity.setCullingEnabled(true);
```

Úseky krajiny se nyní nezobrazují jako jednotlivé objekty třídy *Line*, ale jako jeden objekt *LineStrip*. Tento objekt se zobrazuje jako jeden polygon, nicméně objekty *Line* jsou stále používány pro detekci kolizí.

```
\\Příklad vytvoření objektu LineStrip
LineStrip lineStrip = new LineStrip(0,0,LINE_THICK,source.length,
                                    resourceManager.getVBOM());
for(int i=0; i < source.length-1; i++){
    lineStrip.add(source[i][0], source[i][1]);
}
attachChild(lineStrip);
```

8.5 Testování

Po této iteraci se hra podstatně změnila. Jde především o vzhled krajiny a přidané objekty. V průběhu testování byl kladen důraz na příjemný vzhled, rozmístění a dosažitelnost objektů na mapě. Objevily se zde problémy s rychlostí vykreslení objektů, které jsem opravil vhodnou optimalizací.

Kapitola 9

6. Iterace

9.1 Zadání

Z důvodu časové tísně byla na základě domluvy s vedoucím práce provedena změna v plánu iterací. Dle původního zadání mělo být startovní menu přidáno až v 8. iteraci tohoto projektu. K jeho implementaci do hry dojde již v této iteraci.

- Aplikace bude rozšířena o startovní menu. Toto menu bude obsahovat zatím jedinou možnost hry, a to hru bez nepřátelských letadel.

Přidání startovního menu je pro hru velice zásadní prvek. Tímto hra získá více konzistentní tvar.

9.2 Analýza

Ve fázi, ve které se projekt nyní nachází, dojde po startu aplikace k okamžitému spuštění hry. Jakmile dojde ke zničení letadla, proběhne automatický restart a hra se spustí znovu. Hráč tak nemá šanci spouštění hry žádným způsobem ovlivnit, aniž by nemusel ukončit samotnou aplikaci.

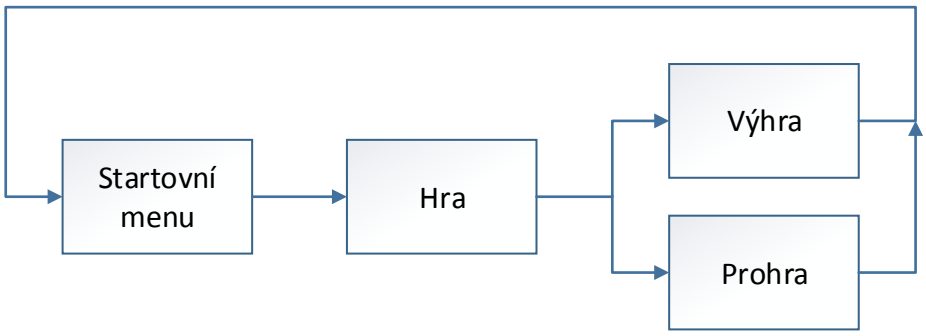
V originální hře bylo cílem, aby uživatel zničil všechny nepřátelské objekty dříve, než vyčerpá všech 5 životů, které měl k dispozici. V opačném případě nastane konec hry.

Logika hry implementované v tomto projektu bude postavena stejným způsobem jako původní hra. Aby uživatel mohl ve hře vidět počet zbývajících životů, je potřeba přidat další indikátor.

9.3 Návrh

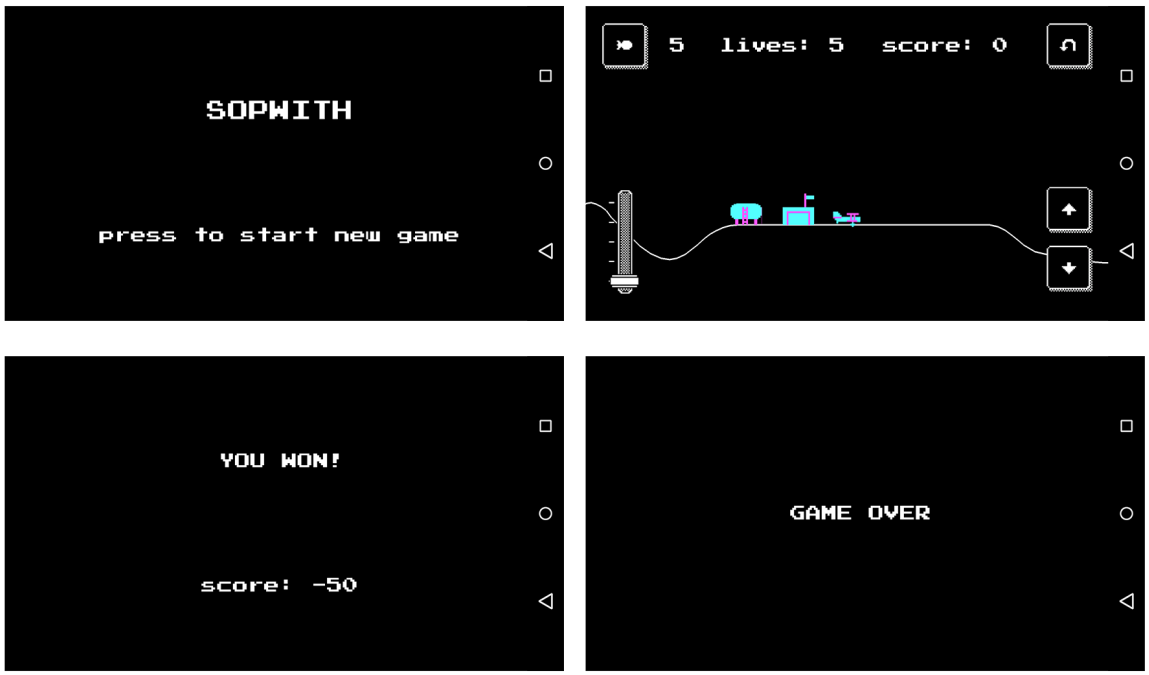
9.3.1 Plán scén

Na níže uvedeném obrázku je znázorněno schéma možných přechodů mezi scénami.



Obrázek 9.1: Schéma přechodů mezi scénami

9.4 Implementace



Obrázek 9.2: Snímky hry po 6. iteraci

9.4.1 Problém opakovaného spuštění

Po opakovaném spuštění aplikace zkrachovala, protože při uzavření aplikace singleton-objekt třídy **ResourceManager** neuvolňoval zdroje. Implementoval jsem metodu *onBackPressed()*, která uvolňuje zdroje v momentě, kdy je aktivní startovní menu nebo přepíná hru do startovního menu, když je aktivní herní scéna.

```
@Override
public void onBackPressed() {
    if (currentScene instanceof GameScene){
        setCurrentScene(new StartMenuScene());
    } else {
        currentScene = null;
        boundCamera.setHUD(null);
        ResourceManager.clean();
        super.onBackPressed();
    }
}
```

9.5 Testování

V průběhu testování této iterace jsem došel ke zjištění, že po zničení letadla hra začíná znovu příliš rychle. Může tak dojít k situaci, kdy se hra automaticky zrestartuje a hráč v daném okamžiku nepochopí, co se stalo. Tento nedostatek bude vyřešen v průběhu následující iterace.

Kapitola 10

7. Iterace

10.1 Zadání

V rámci předchozí iterace došlo ke změně plánu - osmá iterace byla realizována dříve než šestá a sedmá. A proto se nyní v 7. iteraci budu zabývat iterací číslo 6 dle původního zadání.

- V krajině se budou nacházet krávy. V případě zabití krávy dojde k odečtení bodů.
- Krajina bude rozšířena o ptáky. Pokud dojde ke střetu letadla s ptákem, letadlo bude zničeno a hra začíná od začátku.
- Řešení problému příliš rychlého restartování hry po zničení letadla. Tento problém se objevil v rámci testování předešlé iterace.

10.2 Analýza

10.2.1 Animace letadla po zničení

Jakmile je v originální hře letadlo zničeno, dochází k jeho volnému pádu k zemi. Při svém pádu letadlo ve vzduchu zanechává stopy kouře [4]. Do hry naimplementuji podobné chování. Zároveň bude přidána časová prodleva, která bude trvat od momentu, kdy letadlo dopadne na zem, až po okamžik, kdy se hra zrestartuje.

Když je letadlo ve vzduchu zničeno, přepne se do stavu, ve kterém už nebude nijak reagovat na uživatelská tlačítka a bude padat stejným způsobem jako bomba (Viz sekce 7.2.2).

10.2.2 Krávy a ptáci

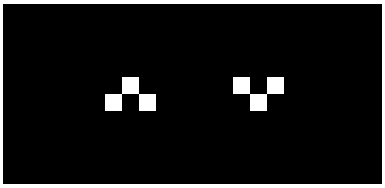
Krávy a ptáci zvyšují samotnou složitost hry. Hráč si musí dát pozor, aby nezasáhl krávu, jinak mu bude z jeho bodového účtu odečten vysoký počet bodů, a to 200. Chování objektů typu kráva je totožné s chováním ostatních objektů na mapě.

Ptáci nutí hráče k velice obezřetnému a pečlivému řízení letadla, jelikož po střetu letadla s tímto objektem dojde k okamžitému zničení letadla.

10.3 Návrh

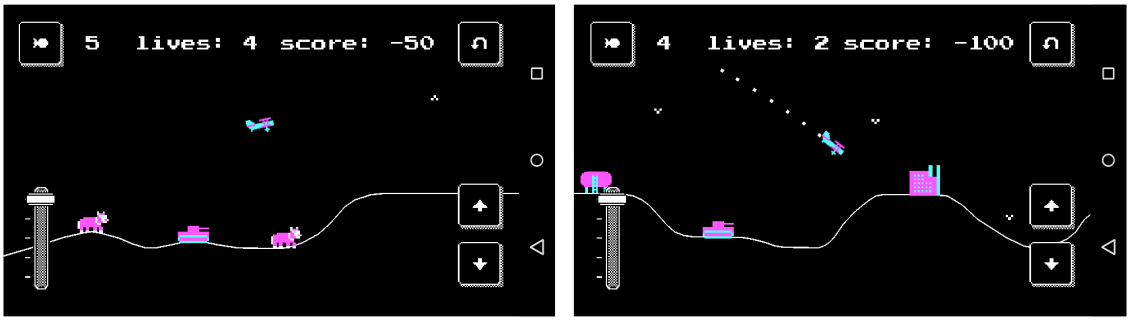
10.3.1 Návrh obrázků

Obrázek 10.1: Grafické znázornění krávy



Obrázek 10.2: Grafické znázornění ptáka

10.4 Implementace



Obrázek 10.3: Snímky hry po 7. iteraci

10.4.1 Ptáci

V každé hře se vyskytuje vždy přesně 10 ptáků, kteří jsou náhodně rozmístěni na celé mapě. Pokud letadlo zasáhne ptáka, ten se ihned objeví v jiném místě na mapě.

Ptáci se pohybují tak, že létají přímo zadaným směrem. Pokud se pták setká s krajinou, oblohou nebo jiným objektem na mapě, dojde k náhodné změně směru jeho letu.

Po implementaci ptactva do hry se zde objevil problém, kdy letadlo, které ještě nevyletělo z letiště, tj. na úplném začátku hry, bylo zničeno ptákem, který ve svém letu náhodně změnil směr a narazil do letadla hráče. Tato situace byla vyřešena provedením kontroly - pokud letadlo je stale na zemi a dojde ke střetu s ptákem, tak pták pouze odskočí a nezničí ho.

10.5 Testování

V průběhu testování letadla při střetu s objektem typu kráva nebo pták se reakce letadla projevila jako zcela vyhovující. Nově implementované objekty v této iteraci, ptáci a krávy, se chovají přesně podle zadání.

Kapitola 11

8. Iterace

11.1 Zadání

V 8. iteraci se budeme zabývat řešením níže uvedených záležitostí:

- Nepřátelské tanky budou střílet na letadlo. Při zásahu letadla hra začíná od začátku.
- Tato iterace bude realizována jako poslední, proto je důležité přidat ikonku aplikace.

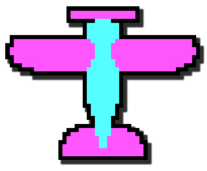
11.2 Analýza

Tank musí nejen střílet na letadlo, ale také musí umět na letadlo svým kanónem zamířit. Proto je potřeba vykreslovat tank a kanón zvlášť. Střílet na letadlo hráče mohou pouze nepřátelské tanky.

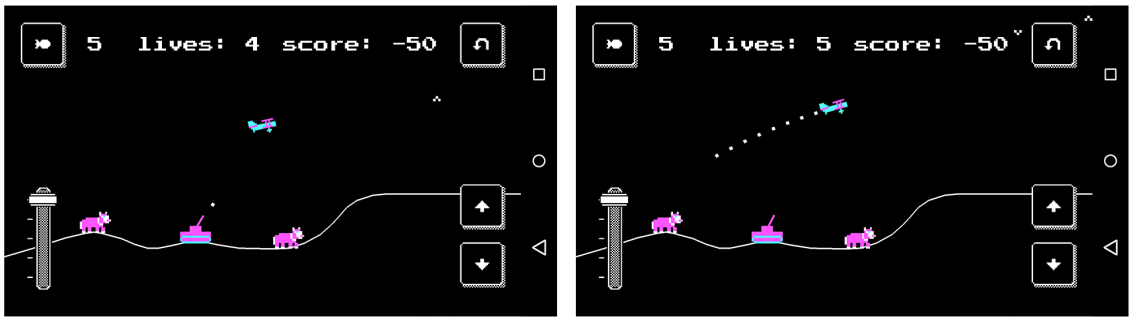
11.3 Návrh

11.3.1 Návrh ikonky aplikace

Ikonka se bude zobrazovat v menu zařízení.



Obrázek 11.1: Ikonka aplikace



Obrázek 11.2: Snímky hry po 8. iteraci

11.4 Implementace

11.4.1 Míření

Tank automaticky zaměřuje letadlo a míří na něho svým kanónem. Každou jednu sekundu tank na letadlo vystřelí.

Objevil se zde ale problém, kdy tank touto intenzivní střelbou velice často zasáhne budovy ve svém okolí. Tato vlastnost je zcela nežádoucí. Daná situace je ošetřena způsobem, kdy tank před každou střelou vytvoří pro hráče neviditelnou trajektorii směrem k letadlu. V případě, kdy do této trajektorie nezasahuje žádný objekt na mapě, tank vystřelí na letadlo.

11.4.2 Optimalizace

V případě, kdy tank na mapě ještě není viditelný, není žádoucí, aby byl aktivní, tj. aby střílel. Do metody *onManagedUpdate()* třídy **Tank** byla přidána kontrola tohoto stavu. V momentě, kdy tank úspěšně zasáhne letadlo, přestává na něho dále mířit.

```
@Override
protected void onManagedUpdate(float time) {
    if (gameScene.getBoundCamera().isEntityVisible(getSprite())
        && !gameScene.getPlane().isDestroyed()){
        ...
    }
}
```

Podobná kontrola byla také realizována pro ptáky ve třídě **Bird**.

11.5 Testování

Testování této iterace proběhlo úspěšně. Problémy s mířením kanónu tanku na letadlo, které se zde objevily, byly vyřešeny promptně v průběhu samotné implementace. Zvolená ikonka se jeví jako vhodná a odpovídající této hře. Finalizací této iterace hra získala komplexní tvar.

Kapitola 12

Nasazení

12.1 Zveřejnění ve službě Google Play

Zákazník dodatečně požádal o přidání do hry systému monitorování Yahoo Flurry (viz [7]). Také došlo ke změně způsobu publikace na službě Google Play - nemusím již vytvářet svůj vlastní účet v dané službě, ale mohu použít účet zákazníka CactooSoftware.

12.2 Monitorování Yahoo Flurry

Služba Yahoo Flurry slouží především pro analytické účely, dále umožňuje připojení reklamy a tím dává příležitost aplikaci zpeněžit. V rámci tohoto projektu bude ovšem použito pouze monitorování, které bude velmi užitečné pro další vývoj hry.

12.2.1 Název balíčku

Flurry pro spojení se službou Google Play potřebuje Android Market ID, což je cesta k balíčku, kde je uložena hlavní třída aplikace.

Hlavní balíček hry byl změněn na **com.cactoosoftware.sopwith**.

12.2.2 Logovaní událostí

Flurry také umožňuje logovat jednotlivé události, které se projevují u uživatelů hry.

Do hry jsem proto přidal odesílání událostí, když ve hře dojde k výhře nebo prohře. V případě výhry je také odesílán dosažený počet bodů. Viz návod [5].

```
Map<String, String> eventParams = new HashMap<String, String>();
eventParams.put("Score", ""+score);
FlurryAgent.logEvent("Game_Win", eventParams);
```

12.2.3 Povolení

Pro práci potřebuje Flurry přidat následující povolení do souboru *AndroidManifest.xml*:

- Přístup k síti Internet
- Kontrola připojení k síti Internet
- Přesná geolokační data

Více informací [6].

12.3 Google Play

Na službě Google Play je hra dostupná přes odkaz <https://play.google.com/store/apps/details?id=com.cactoosoftware.sopwith>.

Kapitola 13

Závěr

13.1 Výkaz stráveného času

V následující tabulce je uveden strávený čas oproti plánovanému kompletnímu vypracování.

Část projektu	**Plánovaný čas [hod]**	**Strávený čas [hod]**
Manažerská dokumentace	30	30
Rešerše	30	30
1. iterace: Krajina a letadlo	20	25
2. iterace: Řízení letadla	20	30
3. iterace: Fyzika letu letadla	30	45
4. iterace: Bomby a náboje	20	30
5. iterace: Domy a tanky	20	45
6. iterace: Startovní menu	20	30
7. iterace: Krávy a ptáci	20	30
8. iterace: Střelba nepřátelských tanků	20	25
9. iterace: Zvuky	30	-
10. iterace: Nepřátelská letadla	30	-
11. iterace: Multiplayer	30	-
Nasazení do služby Google Play	4	20
Závěrečná práce	30	20
Celkem	**354**	**360**

Tabulka 13.1: Výkaz stráveného času podle jednotlivých částí projektu

13.2 Možné pokračování práce

Zejména z důvodu časové náročnosti jsem se nedostal k implementaci posledních 3 iterací. Tyto iterace zahrnují následující části projektu:

- Nepřátelská letadla
- Zvuky
- Multiplayer prostřednictvím Bluetooth

Uvedené funkcionality by měly být realizovány nejdříve.

Dalším rozvojem hry je také možné přidání systému úrovní složitosti hry. Na začátku by hra mohla začínat na jednoduché úrovni a postupně se stávat složitější. Také může dojít k situaci, kdy hráči nemusí být zpočátku zcela jasný cíl hry a způsob ovládání letadla. Proto bych jako vhodnou část k rozšíření navrhl implementaci tutorialu v úvodu hry.

Dobrým zdrojem nápadů na zlepšení jsou analytická data ze systémů Yahoo Flurry a Google Play.

13.3 Závěrečné zhodnocení

V předkládané bakalářské práci jsem se věnoval vytvoření hry pro platformu Android, která svou jednoduchostí a grafickým zpracováním bude připomínat originální videohru Sopwith z roku 1984.

V první části této práce jsem se rozhodoval pro výběr vhodného frameworku, který použiji pro implementaci hry. Jako nejvhodnější jsem určil AndEngine, který je výkonný, nevyžaduje znalost OpenGL a zároveň podporuje sadu rozšíření. Jako významnou výhodu pro použití tohoto frameworku spatřuji svou hlubokou znalost programování v jazyku Java.

Další část projektu již tvořily samotné iterace. Nejprve byla ve hře naimplementována jednoduchá krajina, kterou tvořily náhodně generované body, a letadlo ve tvaru čtverce. Dále pak došlo k rozšíření hlavního objektu hry, tj. letadla. Nabylo už svou typickou podobu letadélka, a zároveň získalo vlastnost otočení se kolem vlastní osy, zrychlení, zpomalení a možnost změny směru letu. Následně byla letadlu naimplementována funkce střílení a házení bomb do krajiny.

Až do této doby byla krajina náhodně generována, ovšem z důvodu potřeby implementace dalších objektů do hry se nově krajina tvoří načtením souřadnic úseků a jednotlivých objektů ze souboru. Díky tomu mohly do krajiny přibýt další objekty jako jsou přátelské i nepřátelské tanky, hangáry, krávy a ptáci. Hra tím nyní získala na zajímavosti, protože letadlo již může ničit nepřátelské objekty, ale zároveň musí hráč být obezřetný, aby nezničil přátelský objekt nebo nezastřelil zvíře.

V neposlední řadě bylo do hry implementováno startovní menu a tlačítka pro ovládání letadla. Nepřátelské tanky získaly možnost střílet na hráčské letadlo, a tím došlo k dalšímu významnému posunu kvality hry.

Každá z iterací byla zanalyzována, průběžně zdokumentována a také otestována. V průběhu testování jsem se setkával s různými nálezy ke zlepšení hry, které jsem zvládl doprogramovat a hru tak vylepšit. Finální částí tohoto projektu je nasazení hry Sopwith do služby Google Play. Aplikace je zde nyní hráčům a fanouškům této hry k dispozici ke stažení.

Při plánování projektu jsem nedostatečně odhadl časovou náročnost a možné potíže s implementací. Také jsem strávil více času návrhem grafických prvků, což je ovšem pro grafickou

aplikaci velmi důležité. Proto došlo k situaci, kdy některé části projektu nebyly zrealizovány. Bez ohledu na to však hra získala komplexní tvar a je možné se jejím hraním zabavit.

Při rozhodnutí hru dále rozvíjet lze na tento projekt plynule navázat implementací iterací, které nebyly v této práci dokončeny a dále například navyšováním úrovní náročnosti hry nebo inspirací od samotných hráčů na Google Play, kteří zde mají možnost hru hodnotit a komentovat.

Literatura

[1] *Web AndEngine* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.andengine.org/>.

[2] *Bluetooth, Android developer* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://developer.android.com/guide/topics/connectivity/bluetooth.html>.

[3] *Web cocos3d-x* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.cocos2d-x.org/>.

[4] *Sopwith, Dos Games Archive* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.dosgamesarchive.com/download/sopwith/>.

[5] *Custom events, Flurry developer portal* [online]. [cit. 1. 1. 2015]. Dostupné z: <https://developer.yahoo.com/flurry/docs/analytics/gettingstarted/events/android/>.

[6] *Get Started With Android 5.0, Flurry developer portal* [online]. [cit. 1. 1. 2015]. Dostupné z: <https://developer.yahoo.com/flurry/docs/analytics/gettingstarted/android/>.

[7] *Flurry (Company), Wikipedia* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://en.wikipedia.org/wiki/Flurry_(company)>.

[8] *Git SCM* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://git-scm.com/>.

[9] *Web JSON* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.json.org/>.

[10] *Knihovna JSON* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.json.org/java/>.

[11] *OpenGL ES, Android developer* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://developer.android.com/guide/topics/graphics/opengl.html>.

[12] *Object pool pattern, Wikipedia* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://en.wikipedia.org/wiki/Object_pool_pattern>.

[13] *Popis Sopwith II* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.wingkong.net/sopwith2b/sopwith2.html>.

[14] *Sopwith readme file* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://www.wingkong.net/sopwith2b/files/sopwith-readme.txt>.

[15] *Web Unity 3D* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://unity3d.com/>.

[16] *Sopwith (video game), Wikipedia* [online]. [cit. 1. 1. 2015]. Dostupné z: <http://en.wikipedia.org/wiki/Sopwith_(video_game)>.

[17] BOURG, D. M. – BYWALEC, B. *Physics for Game Developers.* 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc., 2nd edition, 2013.

[18] SCHROEDER, J. *AndEngine for Android Game Development Cookbook.* Livery Place, 35 Livery Street, Birmingham B3 2PB, UK : Packt Publishing Ltd., 1st edition, 2013.

[19] SCHWABER, K. – BEEDLE, M. *Agile Software Development with Scrum.* Upper Saddle River, NJ, USA : Prentice Hall PTR, 1st edition, 2001.

Příloha A

Seznam použitých zkratek

2D Two-Dimensional

3D Three-Dimensional

GPU Graphics Processing Unit

JSON JavaScript Object Notation

OpenGL ES OpenGL for Embedded Systems

OpenGL Open Graphics Library

WBS Work Breakdown Structure

Příloha B

Obsah přiloženého CD

```
+apk.....................................Adresář s instalačním balíčkem aplikace
|__Sopwith.apk..................Instalační balíček aplikace Sopwith pro Android
+docs....................................Zdrojový kód textu bakalářské práce
+project
|__+AndEngine-AnchorCenter..................Eclipse projekt knihovny AndEngine
|__+Sopwith............................Eclipse projekt samotné aplikace Sopwith
+text
|__Sopwith.pdf............................Text bakalářské práce ve formátu PDF
readme.txt.............................................Stručný popis obsahu CD
```