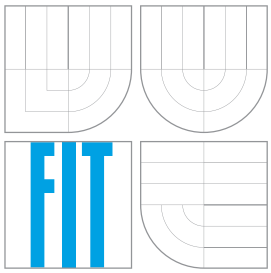
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LETECKÁ HRA PRO ANDROID

FLIGHT GAME FOR ANDROID

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DAVID ŠABATA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2013

Abstrakt

Tato práce se zabývá vývojem letecké hry pro platformu Android. Nejdříve budou popsány možnosti nativního vývoje a vývoje s využitím knihovny Libgdx. Dále budou vysvětleny principy, které se uplatňují při letu skutečného letadla, a jejich zjednodušené použití v leteckých hrách. Práce také shrne současné trendy v ovládání leteckých her na mobilních zařízeních a navrhne novou metodu založenou na dotykovém ovládání. S využitím tohoto ovládání bude navržena a implementována letecká hra. V závěru bude popsán proces testování a publikování, stejně jako možnosti dalšího vývoje této hry.

Abstract

This work deals with flight game development on Android platform. Firstly the possibilities of native development and development using Libgdx library will be discussed. Then flight mechanics of a real aircraft and simplified mechanics used in flight games will be explained. The work will also summarize current trends in mobile flight game controls and will propose a new control method based on touch input. Using this method a flight game will be designed and implemented. In the end of this work the process of testing and publishing will discussed, as well as possibilities of further development.

Klíčová slova

vývoj her, mobilní zařízení, mobilní aplikace, Android, mechanika letu, dotykové ovládání

Keywords

game development, mobile devices, mobile applications, Android, flight mechanics, touch input

Citace

David Šabata: Letecká hra pro Android, diplomová práce, Brno, FIT VUT v Brně, 2013

Letecká hra pro Android

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Adama Herouta, Ph.D.

.....
David Šabata
19. května 2013

Poděkování

Na tomto místě bych rád poděkoval Doc. Ing. Adamu Heroutovi, Ph.D. za odborné vedení a cenné rady při tvorbě této práce. Dále také firmě CUKETA, s. r. o. a ostatním, kteří se podíleli na testování a svými připomínkami přispěli ke kvalitě výsledné aplikace. V neposlední řadě pak děkuji své rodině a přítelkyni za neutuchající podporu při studiu.

Obsah

Kapitola 1

Úvod

Hry pro mobilní zařízení, zejména telefony a tablety, v současnosti zažívají nebývalý rozmach. Dalo by se říct, že jsou nyní tam, kde byly před patnácti až dvaceti lety hry počítačové. Zatímco typický tým vyvíjející hru pro počítač nebo herní konzoli čítá desítky, ale často i stovky členů, hru pro mobilní telefon je stále možné stvořit jen v několika málo lidech.

Neustále narůstající výkon dovoluje přiblížit se kvalitě počítačových her. Těmi se nově díky mobilním zařízením může hráč zabavit například na cestách anebo při čekání na autobus. Velká většina lidí s sebou neustále nosí výkonný počítač ve velmi malém balení, který je navíc vybaven množstvím sensorů a často i soustavným připojením k internetu. To nabízí prostor pro velké množství inovací a je dobře, že se takové inovativní přístupy skutečně objevují.

Tato práce se pokusí přispět svým dílem navržením nové metody ovládání leteckých her pro mobilní zařízení. Nejdříve bude popsána platforma na které bude implementace probíhat, dále budou vysvětleny principy, díky kterým letí a manévruje skutečné letadlo. Tyto principy budou poté zjednodušeny pro použití v prostředí her. Dále budou shrnuty současné možnosti ovládání herních letadel a navržena nová metoda jejich ovládání, založená na dotyku. Nad touto metodou bude vystavěna plnohodnotná letecká hra, jejíž vývoj bude popsán od prvotního návrhu až po její implementaci a publikování v internetovém obchodě Google Play. Součástí práce bude také průzkum reakcí hráčů a jeho zhodnocení. Na závěr bude nabídnuta také kapitola shrnující další možný vývoj této hry.

Kapitola 2

Vývoj her na platformě Android

Andriod je operační systém pro mobilní zařízení, založený na Linuxovém jádře a poskytovaný ve formě otevřeného zdrojového kódu. Poprvé byl představen v roce 2007, první mobilní telefon s Androidem se dostal na trh o rok později. Aktuálně dle statistik za první čtvrtletí roku 2013 má na poli *chytrých mobilních telefonů* (smartphone) Android jakožto operační systém podíl přes 38% [1] (měřeno na základě navštěvovanosti webových stránek z mobilních telefonů) se zřejmým vzestupným trendem. Ačkoliv je Android použitelný pro množství různých zařízení (a implementace mimo pole mobilních telefonů se už nyní objevují), bude se tato práce soustřeďovat pouze na jeho využití v mobilních zařízeních typu mobilního telefonu nebo tabletu. Informace v následujících kapitolách budou vycházet z oficiálního vývojářského portálu [2], knihy o vývoji her pro Android [3] a z vlastních zkušeností a poznatků. Kapitola 2.5 pak bude čerpat z dokumentace knihovny Libgdx [4].

2.1 Architektura systému

Architekturu systému lze rozdělit do několika úrovní. Na té nejvyšší jsou aplikace psané v jazyce Java, které používají systémové funkce a rozhraní. Distribuují se v balíčcích v tzv. *mezikódu* (Java bytecode). Mezikód je výstup kompilace a jako takový je přenositelný mezi platformami. Pro spuštění však potřebuje interpret, kterým je v tomto případě virtuální stroj pojmenovaný Dalvik.

Dalvik samotný se nachází na nižší úrovni, tedy mezi programy a jejich součástmi v nativním kódu, psanými v jazyce C, C++ či jazyce symbolických instrukcí. Zde můžeme najít především sadu systémových knihoven, od kterých je vyžadován vysoký výkon. Jsou jimi například knihovny pro vykreslování (OpenGL), databáze (SQLite), šifrování (SSL) anebo multimédia. Jejich rozhraní je dostupné jak v prostředí Javy, tak skrze *JNI* (Java Native Interface). JNI je rozhraní umožňující psát programy v některém z výše uvedených jazyků s možností přímého propojení se součástmi psanými v Javě. Aplikace jako celek je pak rozdělená do dvou samostatně kompilovaných částí, jejichž spolupráci zajišťuje Dalvik.

Do nativního kódu je vhodné převést především provádění náročných operací v kritickém čase, například simulaci fyziky ve 3D scéně anebo výpočty pro umělou inteligenci. Není ovšem vhodné, ačkoliv by se to mohlo zdát, implementovat zde celou aplikaci. Výkon nemusí být oproti interpretovanému kódu nutně vyšší, a naopak mohou nastat problémy s přenositelností a optimalizací pro konkrétní zařízení [5]. Tu by běžně prováděl Dalvik, zde ale záleží plně na programátorovi. Konkrétní přínosy i slabiny použití JNI budou dále rozvedeny v kapitole 2.3.

Na nejnižší úrovni je již zmíněné jádro psané v jazyce C. To zajišťuje přístup k hardwaru, multitasking, správu souborového systému a další základní úlohy. Přímo k jádru však programátor v drtivé většině případů nepotřebuje přistupovat. Funkce jádra programátor využije jen při výjimečných případech kdy vytváří systémovou aplikaci. Taková aplikace vyžaduje pro svůj běh vyšší oprávnění a běžný uživatel ji nemůže spustit. Své uplatnění ale nachází při modifikacích systému, kde jakožto předinstalované aplikace získávají vyšší oprávnění automaticky. Jelikož je Android otevřená platforma, je možné její součásti libovolně upravovat a takto vzniklý systém distribuovat.

2.2 Verze systému a cílová zařízení

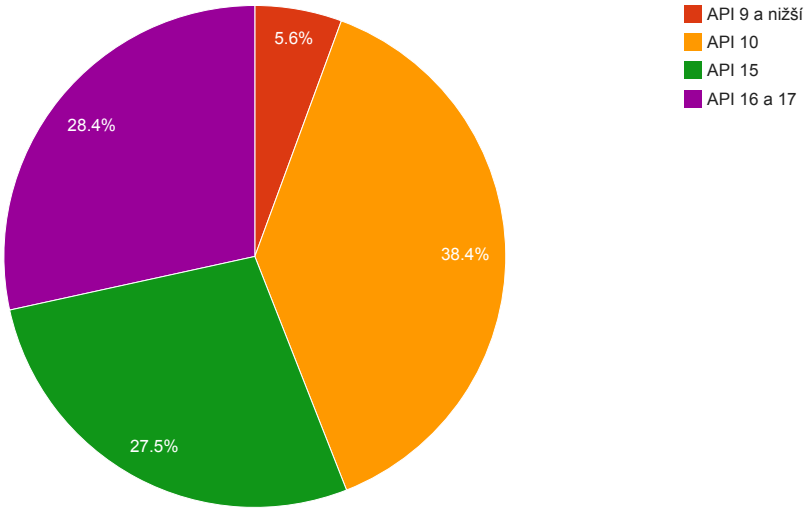
Vývoj operačního systému vede firma Google, která vydává zdrojové kódy nových verzí. Ty se ovšem přímo do zařízení dostávají jen velmi zřídka (jde o referenční modely telefonů a tabletů které si nechává vyrábět přímo Google). Typicky přebírají kód jednotliví výrobci a upravují či doplňují jej tak, aby fungoval pro konkrétní modely (nekompatibilita hrozí především u specifického hardwaru jako je 3D displej a další). Často také výrobce přidává dodatečné aplikace, ať už pro plné využití možností zařízení nebo čistě pro reklamní účely. Jak bude výsledný systém vypadat, jaké funkce bude obsahovat a jaká zařízení bude podporovat je plně v zodpovědnosti výrobců. Právě kvůli tomu jsou nejčastěji aktualizovány modely z vyšší cenové třídy a levnější zařízení bývají opomíjena.

Důsledkem otevřenosti Androidu je jeho roztříštěnost co se verzí týče. Aktuální rozložení verzí *API* (programátorské rozhraní, Application Programming Interface) k dubnu 2013 je na obrázku 2.1, data pocházejí z oficiálního zdroje [6] a jsou měřena podle toho, jak se zařízení s danými verzemi připojují k obchodu Google Play. Statistika tedy nezobrazuje podíly verzí prodaných zařízení, nýbrž těch, která alespoň jednou navštívila oficiální obchod s aplikacemi. Z hlediska vývojáře má tato metodika mnohem vyšší vypovídací schopnost. Je vidět, že více než třetinový podíl má verze 10 (v uživatelském značení jde o Android verze 2.3.x) která je i nyní, po téměř dvou letech od prvního vydání, stále nasazována do nově prodávaných zařízení. Jde však především o zařízení nejnižší cenové a výkonnostní třídy.

Je často připomínanou pravdou, že právě zpětná kompatibilita se staršími verzemi systému znamená pro tvůrce aplikací množství práce navíc. V případě vývoje her však toto neplatí, neboť ve skutečnosti využívají jen malou část API. Zde je z programové části důležitá především podpora knihovny OpenGL ES ve verzi 2.0. Ta je nyní dostupná na více než 90% zařízení a můžeme ji tedy považovat za standard. Mnohem důležitější je však technická vybavenost zařízení, zejména pak výkon procesoru, výkon grafického čipu a velikost jeho paměti. Zde jsou parametry často velmi rozdílné a relevantní srovnání či statistiky bohužel chybí. Nezanedbatelným parametrem jsou také rozměry displeje. Vzhledem k různým hustotám zobrazení není vhodné uvádět pouze rozlišení. Větší vypovídací hodnotu z hlediska dotykového ovládání má fyzická úhlopříčka displeje, která je v současnosti nejvíce zastoupená v rozměrech okolo 4 palců. [6]

2.3 Nativní vývoj her

Při zaměření konkrétně na vývoj grafických her se ukazuje, že hluboké znalosti systému nejsou nezbytné. Android umožňuje navrhovat velmi složitá a propracovaná uživatelská rozhraní, hernímu vývojáři však bude stačit jen nezbytné minimum. Podobně je tomu s dalšími vymoženostmi systému.



Obrázek 2.1: Rozložení verzí systému Android k dubnu 2013. Obecné aplikace musejí řešit kompatiblitu s co nevětším množstvím verzí systému. Hry však používají pouze minimum systémových funkcí, a lze tak často pokrýt více než 94% zařízení. Zbývající procenta představují nejméně výkonná zařízení s nejstaršími verzemi systému. Zdroj dat: oficiální portál Android [6]

Základním stavebním kamenem každé aplikace je tzv. *Aktivita* (třída `Activity`). Zjednodušeně lze říct že Aktivita je okno aplikace, typicky zobrazované přes celý prostor displeje (s výjimkou informační lišty a případných virtuálních kláves), se kterým může uživatel interagovat. Aktivita je rodičem jednotlivých elementů uživatelského rozhraní, případných dialogů, nabídek a dalších prvků. Pro potřeby hry běžně postačuje jediná aktivita jejímž potomkem je speciální oblast do které může vykreslovat OpenGL (oblast je třídy `GLSurfaceView`). Při vytoření této oblasti systém současně spustí speciální vlákno, které má přístup k OpenGL kontextu, a může tedy provádět grafická volání. Toto vlákno je poté systémem pravidelně probouzeno tak, aby vykreslování probíhalo v rychlosti co nejbližší ideálním 60 snímkům za sekundu. S uživatelským rozhraním systému běžící hra až na výjimky nepracuje, vše probíhá v oblasti vyhrazené pro vykreslování.

Kromě zobrazování je Aktivita důležitá také pro příjem vstupních událostí. Má možnost *přihlásit* se u systému ke sledování dotyků na obrazovce, stisků tlačítek, naklonění zařízení, případně stavů dalších sensorů. Tyto události přicházejí v odděleném vlákně které nemá možnost vykreslování. Vlákno typicky upraví vnitřní stav hry a poté se opět uspí a čeká na další událost. Platí zde samozřejmě obdobné principy jako u jiných vícevláknových prostředí. Jak vlákno pro vykreslování, tak i vlákno pro zpracování události nesmějí provádět blokující volání a neměly by provádět časově náročné výpočty. Pro takové případy nabízí Android možnost spuštění odděleného vlákna a postupy jak s takovým vláknem asynchronně komunikovat, aniž by docházelo k narušení plynulosti běhu aplikace, v tomto případě poklesu počtu vykreslených snímků za vteřinu.

2.3.1 Negativa nativního vývoje

V případě že zvolíme tvorbu hry čistě na základě Androidu bez mezičlánků typu frameworku nebo herního engine, můžeme narazit na některé nepříjemnosti. Především nemůžeme počítat s jakýmkoliv rozšířením nad možnosti OpenGL ES. Je plně v kompetenci programátora aby si připravil potřebné datové struktury a pomocné třídy. Systém jako takový sice obsahuje několik základních tříd pro popis vektorů, matic a operace s nimi, ty ale nebyly navrženy pro práci s OpenGL a dochází zde k nesrovnalostem (příkladem může být vnitřní reprezentace matice, která je jednou řádková a jindy sloupcová). Na většinu těchto problémů poukazuje autor knihy o vývoji her pro Android [3], který je současně autorem knihovny Libgdx, která tyto nedostatky napravuje.

Jako jedna z výhod aplikací v jazyce Java bývá uváděna automatická správa paměti použitím tzv. *garbage collectoru*, tedy součásti virtuálního stroje, která počítá reference na každý alokovaný objekt a v případě odstranění posledního výskytu tento objekt uvolní z paměti. Ačkoliv může být tento přístup pohodlný, přináší množství rizik. Vzhledem ke složitosti operačního systému (a často i neznalostí jazyka Java) je pro programátora relativně snadné držet referenci na objekt, který již nepoužívá a o kterém předpokládá, že již byl z paměti uvolněn. Tím sice nedochází k nenávratně ztraceným blokům paměti jako při práci v nativním kódu, paměťové nároky aplikace však zbytečně rostou. Čím více paměti pak aplikace používá, tím častěji se bude garbage collector obracet právě na ni a její paměť se bude snažit uvolňovat. A právě při úklidu paměti dochází ke krátkému pozastavení všech vláken dané aplikace. Toto pozastavení běžně trvá jednotky až desítky milisekund a nemusí být pro uživatele viditelné. Zda se ale aplikace pozastaví a jak dlouho tento stav potrvá závisí na velkém množství faktorů které programátor nemá pod kontrolou. Je tedy velmi důležité zvolit vhodnou taktiku práce s pamětí a té se striktně držet. Jednou z možností je předběžné vytvoření všech objektů které mohou být potřeba s tím, že v průběhu hry již nové objekty nevznikají. Toto je však ideální stav, kterého není snadné dosáhnout. Sofistikovanější správa paměti bude popsána v kapitole 2.5.

S problematikou paměti souvisí také načítání dat potřebných ve hře, zejména pak geometrie scény a textur. V případě 3D modelů opět narazíme na neexistenci jakýchkoliv pomocných tříd. Ve vývojářské komunitě navíc není úplná shoda na tom, který z formátů uložení modelů používat. Ve výsledku je opět na programátorovi, aby zvolil vhodný formát, ten nastudoval a implementoval jeho načítání. To vzhledem ke správě paměti popsané v předchozí části rozhodně není triviální úkol. (U mnohých formátů existují více či méně zdařilé implementace jejich načítání, téměř vždy ale diktují v jakých strukturách budou načtená data uložena, což nemusí být optimální.)

V případě textur je situace o něco jednodušší. Jelikož Android běžně využívá rastrová data obvyklých formátů ve svém uživatelském rozhraní, umí takové soubory přirozeně i načítat. Nedostatkem zde mohou být zvýšené paměťové nároky ve chvíli kdy se textura dekóduje ze svého formátu do interní reprezentace (velmi podobné bitmapě) a následně přesouvá do grafické paměti. V tento okamžik jsou současně alokovány dva až tři objekty reprezentující stejnou texturu, ovšem s rozdílnou vnitřní reprezentací. S velkou pravděpodobností bude opakovaně docházet ke spuštění automatického úklidu paměti. V krajních případech při použití rozměrných textur může dojít i k ukončení aplikace z důvodu překročení maximálního množství přidělené paměti.

Zmíněné slabiny nízkoúrovňového přístupu k tvorbě her jsou jen jedny z mnoha. Zcela jistě záleží na subjektivním přístupu a zkušenostech, zda se můžou rozvinout do skutečných problémů či nikoliv. Drtivá většina těchto i jiných nedostatků je samozřejmě řešitelná a

vývojová prostředí nabízejí množství pomocných nástrojů. V extrémním případě může programátor zajít až tak daleko, že vytvoří vlastní framework či engine. Je však otázkou, zda by měl své síly soustředit na ladění a obcházení záludností systému nebo použít existující řešení a své síly zaměřit na konkrétní prvky, které odliší jeho hru od konkurence.

Z výše uvedeného je zřejmé, že u většiny grafických aplikací je žádoucí použít existující a ověřený kód na kterém se bude dále stavět. Není podstatné zda půjde o malou knihovnu, framework či herní engine – názvosloví zde není ustálené (v dalším textu se proto budu držet obecného pojmu *knihovna*). Důležitý je přínos pro programátora, úspora času a prostředků, možnost soustředit se na konkrétní produkt a ignorovat podružné problémy. V neposlední řadě také použití knihovny znamená přenesení odpovědnosti za část aplikace na někoho jiného. To může znamenat zefektivnění vývojového procesu, ale také jeho zásadní zpomalení v případě že se vývoj knihovny zastaví nebo obsahuje chyby které autor nechce či nemůže opravit.

2.4 Komplexní vývojová prostředí

Jistou ochranou před hrozbou budoucích problémů je využití komerčně nabízených řešení. Dva nejrozšířenější zástupci této kategorie jsou Unity 3D a Unreal Engine, kde oba sami sebe označují za herní engine. V obou případech jde o specializovaná vývojová prostředí která se snaží upozadit programování ve smyslu psaní kódu a naopak upřednostňují přímé sestavování scény a definice chování. Vývojář může hru hrát a souběžně v reálném čase upravovat. K tomu slouží interaktivní pohled do scény, kde je možné pracovat na úrovni objektů. Samozřejmostí je množství předdefinovaných chování, fyzikálních vlastností, materiálů a často i celých objektů se všemi vlastnostmi (například celé pojízdné auto). Na skutečné programování přichází čas pouze ve chvíli, kdy je potřeba nadefinovat vnitřní logiku hry, případně vytvořit nové chování některého z objektů. V takovém případě používá engine vlastní skriptovací jazyk. Jak Unity 3D, tak i Unreal Engine jsou komplexní vývojová prostředí produkující multiplatformní aplikace a Android je pouze jedním z mnoha výstupů. Vývojář ve skutečnosti přichází s architekturou Androidu do kontaktu jen minimálně nebo vůbec. I přesto není vhodné tento přístup opomíjet, neboť s použitím právě Unity 3D vznikají v současnosti nejlépe hodnocené a po vizuální stránce nejpropracovanější hry. Záporem je zde vysoká počáteční investice, a to v řádu minimálně stovek dolarů.

2.5 Knihovna Libgdx

Alternativou k placeným vývojovým nástrojům jsou menší knihovny s těsnější vazbou na systém Android. Nejpoužívanější z nich jsou AndEngine a Libgdx. První z jmenovaných se nazývá herním engine, druhý se označuje za framework. Oba jsou však k dispozici zdarma ve formě otevřených zdrojových kódů s možností komerčního i nekomerčního využití. Jelikož jsou si obě knihovny velmi podobné, soustředím se v dalším popisu pouze na Libgdx, kterou také použiji při implementaci. Zdrojem bude z velké části oficiální dokumentace [4] a také již zmiňovaná kniha o vývoji her pro Android [3].

Na první pohled vypadá knihovna Libgdx podobně jako dříve zmíněné nástroje – stejně jako ony je schopná produkovat aplikační balíčky pro více platforem. Původně však vznikala jako knihovna pro tvorbu 2D her pro Androida a až postupem času došlo rozšíření o podporu dalších platforem. Aktuálně je kromě systému Android podporovaná i tvorba webových

Obrázek 2.2: Hra Shadowgun od brněnského studia Madfinger Games, vytvořená v prostředí Unity 3D, zdroj: www.madfingergames.com

appletů a klasických aplikací pro počítače. Jak konkrétně multiplatformní vývoj probíhá bude popsáno v kapitole 5.2.1.

Narozdíl od jiných řešení knihovna neobsahuje žádné vývojové prostředí ani předpřipravená data. Zaměřuje se více na usnadnění vývoje nabízením pomocných algoritmů a tříd. Je pak čistě na programátorovi, zda výhod knihovny využije. Díky tomu má prostor pro detailní optimalizace na nízké úrovni pro dosažení maximálního výkonu hry, stejně jako se může pohybovat na vyšší úrovni, která umožní velmi rychlé prototypování a vytvoření funkční aplikace. Mezi přednosti Libgdx patří vysoko- i nízkoúrovňové rozhraní pro 2D a 3D grafiku, podpora fyziky, propracovaný systém uživatelského rozhraní, načítání množství formátů, práce se zvuky a mnoho dalších. V neposlední řadě se pak snaží stírat rozdíly mezi jednotlivými zařízeními a verzemi systému Android.

Samozřejmostí je řešení výše zmiňovaných nedostatků při vývoji bez pomocných knihoven. Velká síla Libgdx spočívá v tom, že je implementována částečně přímo v Javě a částečně v nativním kódu. Tím dosahuje vyššího výkonu zejména u již zmiňovaných fyzikálních simulací, ale především má plnou kontrolu nad alokací paměti a jejím následným uvolňováním. Automatické uvolnění přirozeně probíhá pouze v části aplikace běžící ve virtuálním stroji, zatímco v nativním kódu je práce s pamětí plně v režii programátora. Vhodným propojením těchto dvou částí lze bezpečně alokovat paměť kterou garbage collector *neuvidí*, a dokonce dosáhnout stavu kdy framework jako takový uvnitř hlavní herní smyčky žádnou paměť nealokuje. To samozřejmě nemůže úplně zamezit spouštění úklidu paměti, příčinou

však bude běh ostatních aplikací a systému, nikoliv samotné hry. V praxi se pak paměť uklízí jen velmi výjimečně, probíhá velmi rychle (garbage collector spravuje jen velmi malé množství paměti), a hra tedy může běžet plynule.

2.5.1 Struktura aplikace

Vstupním bodem aplikace je instance třídy `Application`, která obsahuje základní herní smyčku. Ta je volaná systémem právě na dříve zmiňované frekvenci blízké 60 snímkům za sekundu. Při tvorbě grafických aplikací se často využívá oddělené volání pro aktualizaci scény a její vykreslení, Android však takové rozdělení nepodporuje (z důvodu návaznosti na systémový element s OpenGL kontextem, popsaný v kapitole 2.3. Je samozřejmě možné aktualizaci scény a její vykreslení oddělit ručně, například v případě že není nutné se scénou manipulovat příliš často, ale chceme zachovat plynulost vykreslování. Typické využití může být u tahových her, kdy se herní logika provádí jen v okamžiku tahu, zatímco zobrazování probíhá v každém snímku, například kvůli probíhajícím animacím. Je samozřejmě možné optimalizovat ještě více, animace nepoužívat a vykreslovat jen ve chvíli, kdy se scéna změní. V takovém případě dochází k maximální úspoře energie. Aplikace ale může působit značně statickým dojmem a pro akční hry tento přístup není vhodný vůbec.

Zatímco samostatná třída aplikace je vhodná jen pro nejtriviálnější aplikace, u složitějších bude žádoucí rozdělit aplikaci do několika částí. Zde knihovna nabízí členění do tzv. *obrazovek* (potomci třídy `Screen`), které mají vlastní životní cyklus. Hlavní třída aplikace pak obstarává pouze přepínání mezi jednotlivými obrazovkami. Ty lze považovat za oddělené logické celky a často je možné je i samostatně vyvíjet, což bude přínosem zejména pro vícečlenný tým vývojářů. Obrazovka jako taková však žádné vyšší funkce nemá a kromě zmíněného zpřehlednění kódu nabízí jen možnost jednoduchého vykreslování texturovaných obdélníků (tzv. *sprite*).

2.5.2 Zpracování vstupních událostí

Vstupní události jsou závislé na aktuální platformě, a proto jsou v Libgdx skryty pod jednotným rozhraním, kdy konkrétní implementaci doplňuje sama knihovna právě podle aktuálního prostředí. Programátor díky tomu nemusí řešit, zda bude aplikace ovládaná dotykem či myší, klávesnicí nebo hardwarovými tlačítky mobilního telefonu nebo třeba ovladačem typickým pro herní konzole. Je zřejmé, že všechny typy událostí nebudou podporovány na všech platformách. Knihovna však zavádí vhodné substituce a sjednocení, kde například kliknutí na dotykovém displeji odpovídá kliknutí levého tlačítka myši. Na vyšší úrovni je pak možné detekovat celá gesta jako chycení a táhnutí nebo přiblížení použitím dvou prstů.

Přirozeně lze definovat i vlastní zpracování vstupu. Příkladem může být křížový ovladač vykreslovaný na obrazovku v případě, že aplikace běží na dotykovém zařízení. Vnitřně pak může takový ovladač *překládat* dotykové události na události stisku klávesy. Ty pak aplikace obslouží jako běžné události a programátor tak velmi snadno získává podporu pro dotyková zařízení, počítačové klávesnice i konzolové ovladače.

Toto chování je umožněno tzv. *vstupními procesory* (třída `InputProcessor`). Vstupní procesor je objekt obsahující kód pro zpracování vstupních událostí. V okamžiku zpracování v procesoru už je známo o jaký typ události jde. Procesor jako takový tedy žádnou detekci neprovádí. Typicky obsahuje referenci na objekty aplikační logiky a s těmi nějak pracuje. Na konci svého běhu vrací návratovou hodnotu podle toho, zda událost zpracoval (a měla by tedy zaniknout) anebo ne (což nutně neznamená že na událost nijak nereagoval).

Vstupních procesorů může být v aplikaci několik, aktivní je ale vždy pouze jeden. Pro komplexní zpracování vstupů se proto zavádí tzv. *multiplexer* (třída `InputMultiplexer`), který představuje kontejner vstupních procesorů, přičemž rozhraní procesoru také sám implementuje. Při příchodu události ji pouze předává jednotlivým procesorům do té doby, dokud ji některý z nich nezpracuje.

Tím je možné vytvářet kaskády vstupních procesorů ať už pro zmíněné sjednocení vstupů napříč platformami anebo pro obsluhu událostí ze složitějších scén. Příkladem může být dotykem ovládaná hra využívající doplňkové ovládací prvky na obrazovce. Vhodným řešením by mohl být multiplexer zastřešující vstupní procesor pro doplňkové prvky (přicházející na řadu jako první, tedy může událost zpracovat a zahodit) a procesor pro ovládání samotné hry. Takto oddělené zpracování vede nejen k přehlednějšímu kódu, ale i k jednodušší obsluze událostí ve chvílích, kdy existuje více prvků souběžně či postupně reagujících na stejný vsutp.

2.5.3 Tvorba uživatelského rozhraní

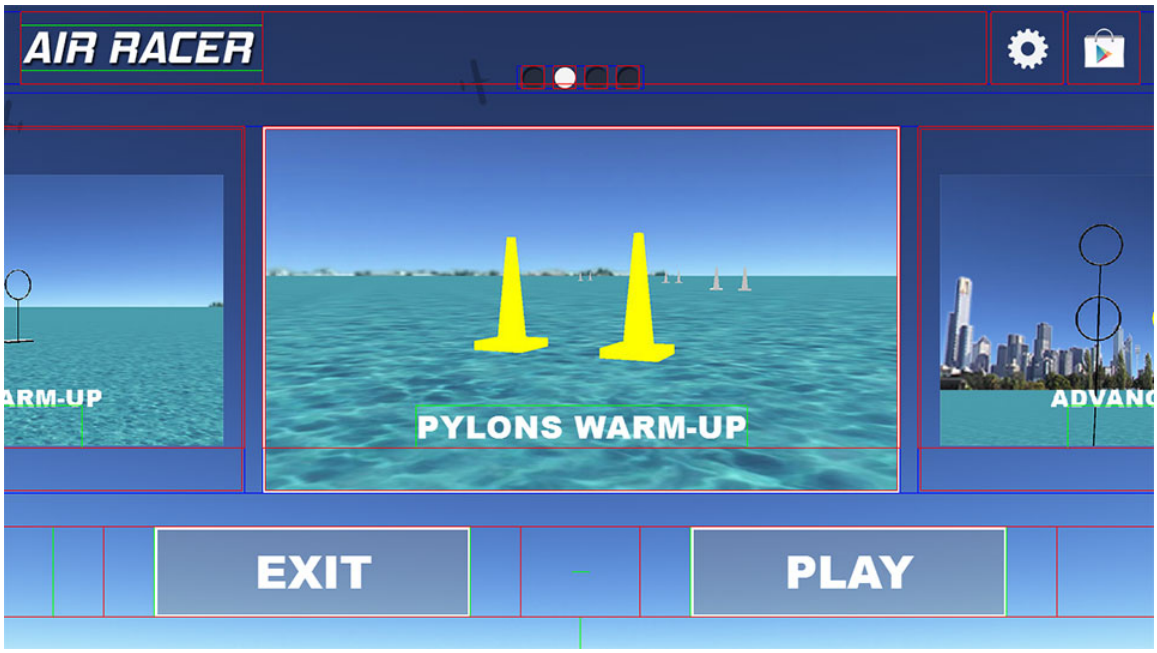
Velmi silnou součástí Libgdx z hlediska tvorby uživatelského rozhraní je knihovna Scene2D.ui [7], která je použitelná i samostatně v obecných projektech v jazyce Java. Jejím základním prvkem je tabulka (třída `Table`), pro kterou se definují jednotlivé buňky a jejich rozdělení do řádků. Chování je podobné tabulkám vytvořeným v jazyce HTML v kombinaci s CSS. Buňkám lze nastavovat pevné či poměrné rozměry, vnitřní okraje, pozadí a některé další atributy. Je také možné buňky v rámci jednoho řádku slučovat; vertikální slučování podporované není, ale lze jej dosáhnout jinou cestou. Specifickým parametrem buněk je pak *rozpínavost* v rámci řádku a v rámci sloupce. Vhodnou kombinací je možné vytvořit dynamickou, tzv. *fluidní* tabulku, která se bude přizpůsobovat svému obsahu i rozměrům zobrazovací plochy.

Nad tabulkami jsou dále vystavěny základní prvky uživatelského rozhraní jako tlačítka, dialogy, pole pro zadání textu, obrázky, bloky textu, posouvatelné panely a další. Využití tabulky lze demonstrovat na tlačítku. To je ve skutečnosti tabulka o třech řádcích a třech sloupcích, kde prostřední buňka obsahuje textový popisek čí obrázek. Všechny jeho buňky pak mají vhodně nastavené pozadí, čímž lze vytvořit například i tlačítka se zakulacenými okraji. Samozřejmě si opět zachovává schopnost přizpůsobit rozměry svému obsahu i okolí. Složitější prvky vznikají obdobně, přičemž často využijí i možnost tabulky do sebe libovolně vnořovat.

Prvky rozhraní také nemusejí být nutně statické. Každý prvek může mít nadefinováno své vlastní chování a v rámci scény se chovat zcela autonomně. Současně knihovna nabízí množství předdefinovaných efektů, které lze prvkům přiřazovat. Typicky jde o geometrické transformace nebo práce s barvami či průhledností. Efekty dostávají jako parametr dobu trvání a krajní hodnoty, mezi kterými v čase interpolují. Podobně jako při zpracování událostí i zde existují speciální efekty chovající se jako kontejnery efektů běžných. Ty umožňují nechat efekty probíhat současně anebo v řadě za sebou. Samozřejmostí je opět libovolné vnořování, čímž lze vytvořit velmi komplexní chování.

Kořenovou komponentou uživatelského rozhraní je tzv. *Stage* (jelikož by český výraz mohl být zavádějící, ponechávám bez překladu a použiji přímo název třídy `Stage`). Představuje kontejner na výše popsané prvky rozhraní. Ty sice své vykreslování řídí samy, ale předpokládají korektně nastavené parametry vykreslování, především pohledových transformací. K tomu slouží Stage, která udržuje kameru s ortogonální projekcí a některé další

struktury sloužící k vnitřní optimalizaci vykreslování. Implicitně Stage pokrývá celou zobrazovací plochu, ale toto chování je možné změnit. Stará se také o pořadí vykreslování prvků rozhraní, čímž řídí jak se budou prvky překrývat. Na samotnou Stage můžeme také, podobně jako na jednotlivé její prvky, aplikovat některé efekty. Stejně tak je možné vložit do jedné obrazovky několik instancí Stage a provádět navigaci mezi nimi, v rámci jediné obrazovky.



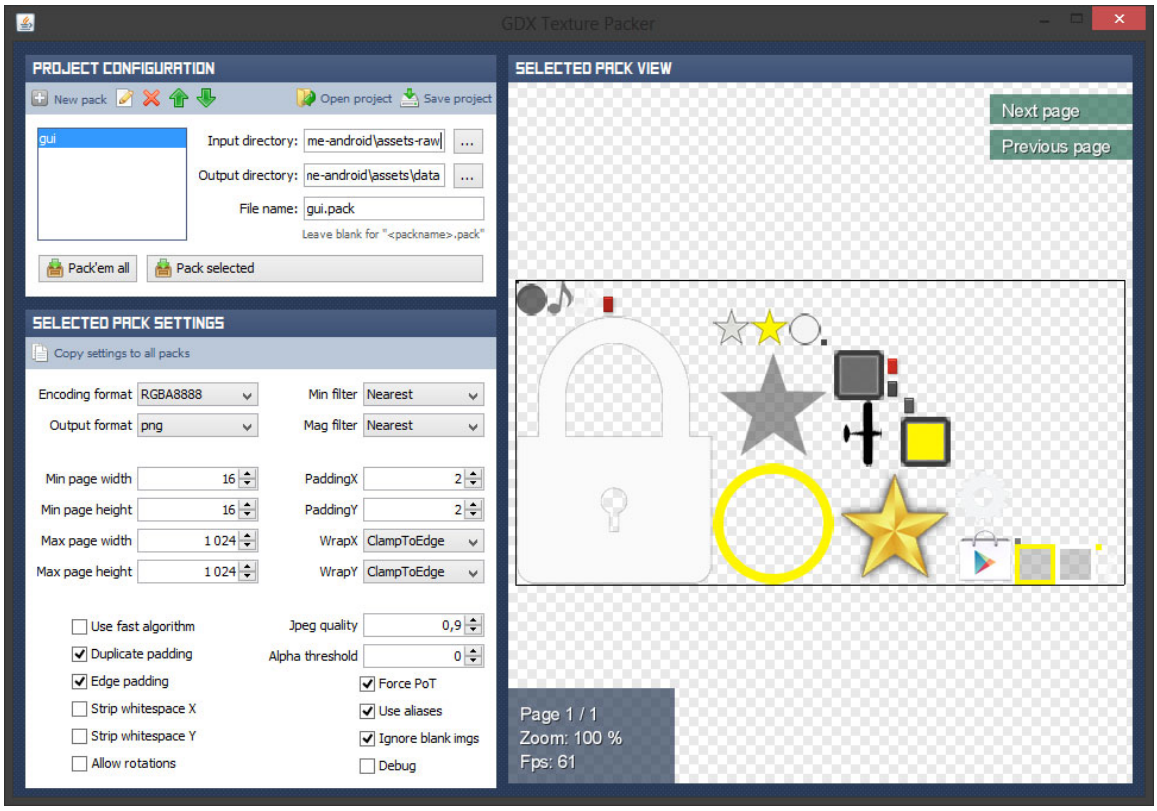
Obrázek 2.3: Součásti rozhraní hlavního menu hry. Červeně jsou orámovány hlavní komponenty, modře kontejnery a zelenou barvou je vyznačeno odsazení vnořené komponenty od okraje svého rodiče.

2.5.4 Pomocné nástroje a součásti knihovny

Knihovna Libgdx obsahuje několik velmi užitečných nástrojů. Některé z nich jsou přímo součástí kódu knihovny a jsou použitelné při běhu aplikace, jiné jsou distribuovány jako samostatné programy vhodné pro předpřipravení dat do vhodného formátu. Jejich použití či nepoužití je vždy rozhodnutí programátora a knihovna jako taková použití žádné z níže popsaných součástí nevyžaduje.

První zajímavý nástroj se jmenuje *Texture Packer*. Slouží k vytváření tzv. *atlasů textur*, což je jediný obrázkový soubor, který v sobě obsahuje vhodně seskládané jednotlivé textury. Jde obecně používaný přístup, který značně urychluje načítání textur i jejich vykreslování zejména v případech, kdy aplikace využívá množství malých textur. Z pohledu načítání se ušetří přístupy do souborového systému, v okamžiku vykreslování je pak velká textura uložená v jediné texturovací jednotce. Pokud by se atlas textur nepoužil bylo by nutné v průběhu vykreslování jediného snímku texturovací jednotky nejen přepínat, ale vzhledem k jejich omezenému počtu i měnit k nim připojené textury. Texture Packer současně s grafickým výstupem ukládá i soubor ve formátu JSON, který knihovna při použití atlasu

automaticky načte a podle informací v něm obsažených vytvoří pojmenované texturové oblasti (odpovídající dílčím texturám). S těmi je následně možné v aplikaci pracovat jako s běžnými texturami, zatímco popsané optimalizace probíhají automaticky bez přičinění programátora. Užitečnou vlastností nástroje Texture Packer je také možnost spouštění z příkazové řádky, respektive jeho přímého zapojení do procesu sestavování aplikace. Programátor tedy udržuje adresář s jednotlivými texturami (jelikož je tento způsob mnohem výhodnější pro případné úpravy), z toho se při sestavení aplikace vytvoří atlas, knihovna jej načte a programátor v aplikaci dostává přísutp opět k jednotlivým texturám.



Obrázek 2.4: Rozhraní nástroje Texture Packer, který z dílčích textur vytváří tzv. *atlas*, uložený v jediném souboru. V pracovní oblasti programu na pravé straně je atlas textur použitý ve hře.

Dalším užitečnou součástí knihovny je podpora bitmapových písem. K používání písem a vypisování textů není v Open GL jednotný přísup. Bitmapová písma se však jeví jako rozumný kompromis mezi vektorově definovanými písmy, které je před vykreslením potřeba rasterizovat, a bitmapami s pevně uloženými řetězci a texty. Bitmapové písmo se skládá z obrázkového souboru, kde jsou jednotlivé znaky a symboly uloženy podobně jako textury v atlasu textur, a definičního souboru ve formátu FNT. Tento soubor však kromě informací pro rozsekání obrázku na jednotlivé znaky obsahuje informace o tom, jak znaky skládat do slov, o rozestupech mezi nimi, o názvu a charakteristikách písma a další. K vytvoření těchto souborů lze využít některý z volně dostupných nástrojů (bitmapová písma jsou opět obecně používanou technikou). Knihovna Libgdx tyto soubory načítá a může s jejich použitím

vykreslovat jednořádkový či víceřádkový dynamický text. Taková funkcionalita je velmi výhodná při překladu aplikace do jiných jazyků. V takovém případě není potřeba udržovat popisky v grafických souborech (čímž rapidně vzrůstá velikost celé aplikace, ale i nároky na její údržbu a další vývoj), ale je možné použít běžný textový soubor s řetězci. Přidání podpory dalšího jazyka pak znamená pouhé přeložení sady řetězců, vše ostatní proběhne automaticky, včetně například správného zarování popisků v uživatelském rozhraní, které plyne z chování prvků rozhraní popisovaných v kapitole 2.5.3.

Specialitou knihovny Libgdx je použití vizuálních témat pro celou aplikaci (základem je třída `Skin`). Při návrhu uživatelského rozhraní zavádí vrstvu abstrakce, kde každému typu prvku rozhraní přiřazuje definici vizuálního stylu. Definuje se tedy obecný vzhled tlačítek, polí, posuvníků a dalších prvků. Je také možné zvolit ještě jemnější dělení v rámci typu, například zavést několik skupin tlačítek a jejich vzhled definovat odděleně. Pro popis vizuálních témat slouží soubor ve formátu JSON obsahující konkrétní názvy tříd prvků rozhraní a poté pro každý z nich sadu atributů jako je barva, pozadí, použité písmo a další. Právě zde lze vhodně využít atlasy textur a bitmapová písma popsaná v předchozích odstavcích. Po načtení vizuálních stylů do knihovny jsou automaticky aplikovány. Knihovna pozná že je v rozhraní použitý prvek, jehož třída má přiřazenou definici, a tuto použije. Změnou definičního souboru lze velmi snadno změnit celý vzhled aplikace. Samozřejmě je možné s aplikací dodávat i několik vizuálních témat a výběr ponechat na uživateli; z hlediska programátora je toto jen úprava jediného řádku, kde se definiční soubor načítá.

Poslední popisovanou součástí knihovny je *Assets Manager*, který je možné volně přeložit jako *správce obsahu*. Přestože je jeho použití volitelné, je velmi silně doporučováno. Správce obsahu totiž představuje jednotné místo v rámci aplikace, přes které se bude přistupovat k datovým souborům. Těmi můžou být textury, zvuky, různé definiční soubory, a teoreticky jakýkoliv obsah. Správce dostane seznam datových souborů které jsou vyžadovány pro běh aplikace a tyto soubory bude načítat. Jeho rozhraní je vhodně navržené tak, aby bylo použitelné pro vytvoření načítací obrazovky s možností zobrazení reálného průběhu načítání. Zajímavý je také způsob, jakým ze souborů vznikají datové objekty. Aplikace definuje soubor a cílovou třídu, jejíž instanci chce získat. Pro každou takovou třídu pak musí existovat speciální načítací třída (tzv. *loader*), která umí ze souboru konkrétní objekt vytvořit. Pro všechny základní formáty už knihovna tyto třídy nabízí, stejně jako možnost načítání parametrizovat. To se využije například pro volbu způsobu filtrování textur. Spolu s načítáním také správce hlídá, zda jsou data ještě používána a může je průběžně uvolňovat.

2.6 Knihovna Bullet Physics

Nezbytnou součástí zejména leteckých her bude jistá forma fyzikální simulace. K tomuto účelu popíši volně dostupnou knihovnu Bullet Physics. Ta může fungovat v několika režimech, přičemž dva nejvýznamnější jsou režimy tzv. *kolizního světa* (třída `btCollisionWorld`) a *simulačního světa* (třída `btDiscreteDynamicsWorld`). Základní principy jednoduché 2D a 3D simulace popisuje již zmiňovaná kniha o vývoji her [3], informace specifické pro knihovnu Bullet Physics pak vycházejí z její oficiální webové stránky [8] (manuál ve formátu PDF je součástí stažitelného vývojářského balíčku).

Jak již název napovídá, kolizní svět obsahuje entity reprezentované svými kolizními tvary a signalizuje, pokud dojde k jejich kontaktu. Na tento kontakt lze následně reagovat v rámci herní logiky. Entity se ale dále mohou pohybovat a v kolizi pokračovat. Knihovna

zde slouží čistě k detekci kolizí a nikoliv k jejich další obsluze. Detekce probíhá ve dvou fázích – tzv. *široké* a *úzké*. Při široké fázi jsou namísto skutečných obalových těles porovnávány kolize osově zarovnaných obalových kvádrů (metoda *AABB - Axis Aligned Bounding Boxes*). Tímto krokem je možné rychle odstranit velké množství dvojic objektů, které spolu nemohou v žádném případě kolidovat. V úzké fázi se následně hledá kolize mezi skutečnými tvary entit. Tento krok je značně náročnější, neboť obalové tvary můžou být jednoduché (základní tvary jako koule, kvádr, plocha), ale také složené z obecně libovolného množství základních a složených tvarů. Se složitostí tvaru přirozeně roste i náročnost ověření kolize. Z tohoto důvodu je zavedeno rozdělení do dvou fází, které toto složité porovnání provede jen v případech, kde může kolize reálně nastat.

Nad kolizním světem je vystavěný svět simulační. V tomto světě obsahují entity více informací. Kromě kolizního tvaru je nutná znalost hmotnosti, rychlosti a zrychlení v čase a volitelně dalších vlastností jako například tvrdost či pružnost povrchu. Tyto entity jsou pak součástí plnohodnotné fyzikální simulace, která sama řídí jejich pohyb a případné kolize. V praxi funguje zapojení simulace do hry nejčastěji tak, že herní logika dodává impulsy (například sílu vzniklou vysunutím klapky letadla) a o zbytek se stará právě fyzikální knihovna.

Provádění úplné fyzikální simulace je ve srovnání s pouhou detekcí kolizí řádově složitější. Dokonce natolik, že se v současných hrách pro mobilní zařízení používá jen minimálně, případně jen pro velmi malé množství entit. Takto lze dosáhnout dobrého dojmu ze hry, kdy hráč vnímá hru jako reálnou, přestože fyzikální simulace probíhá jen u hlavního objektu a chování ostatních objektů může být pevně přednastaveno. I v případě simulace jediného objektu je vždy nutné co nejpřesněji fyzikálně popsat působící síly i objekt samotný. Současně není možné do této simulace zasahovat a zavádět vlastnosti, které by realitě neodpovídaly. Ze zmíněných důvodů plyne, že pro tuto práci bude dále relevantní pouze svět kolizní.

2.7 Distribuce aplikací

Aplikace pro systém Android lze distribuovat několika různými cestami. Tou nejzákladnější je přímá distribuce binární podoby aplikace (v souboru s příponou APK). Tato volba však znemnožní jakoukoliv další automatickou aktualizaci (manuální aktualizaci lze stále provést získáním nové verze souboru) a současně nedovoluje mít kontrolu nad tím, kteří uživatelé, na jakých zařízeních a za jakých podmínek, budou aplikaci používat. Navzdory tomu je tato metoda používána, například pri testování prototypu aplikace a jeho kompatibility.

Důmyslnějším způsobem distribuce je využití některého z internetových obchodů. Těch existuje hned několik s různě velkou uživatelskou základnou a s různými podmínkami publikování. Dominantní postavení zde má obchod Google Play, představující oficiální katalog aplikací platformy Android. Jeho hlavní předností je, že je přednastaven v drtivé většině prodávaných zařízení a má tak možnost oslovit maximální množství potenciálních uživatelů. Vývojáři současně nabídne možnost aplikaci průběžně vzdáleně aktualizovat, řídit která zařízení bude aplikace podporovat, sledovat statistiky instalací a také sbírat zpětnou vazbu od uživatelů. Aplikaci zde lze zveřejňovat bezplatně, ale i za poplatek, přičemž si provozovatel obchodu nárokuje 30% podíl z provedené transakce. Zmíněné přednosti jsou však vykoupeny jednorázovým poplatkem za registraci vývojáře, který v době psaní této práce činí $25. Dalším nedostatkem obchodu Google Play je nemožnost rozdělit konkrétní uživatele do skupin a těm následně nabídnout jinou verzi aplikace nebo například možnost bezplatného testování placené varianty. Stejně tak by mnoho vývojářů ocenilo možnost vygenerování speciálních poukazů, kterými by mohli zdarma zpřístupňovat propagační verze svých aplikací.

Kapitola 3

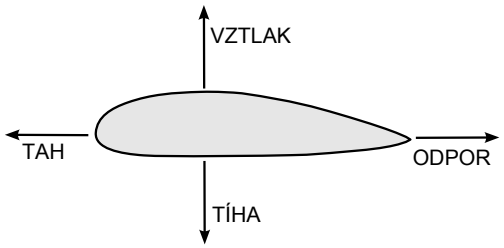
Ovládání skutečného a herního letadla

Tato kapitola popíše principy ovládání letadel a leteckých her. Shrne existující metody a představí novou metodu zaměřenou na ovládání akčních leteckých her v mobilních zařízeních.

Před návrhem herního ovládání letadla je vhodné se seznámit s tím, jak létají skutečná letadla. Více než o konkrétní rovnice zde půjde o základní principy díky kterým se letadlo udržuje ve vzduchu. Rovnice by totiž nebylo možné vyčíslit bez znalosti konkrétních koeficientů, které jsou pro každou konstrukci odlišné. I s nimi by však bylo exaktní matematické popsání letícího letadla velmi obtížné. Následující kapitoly budou čerpat především z knihy Umění létat [9] a studijních materiálů americké vládní agentury pro letectví (NASA) [10].

3.1 Síly působící za letu

Základní pojem pro další popis bude takzvaný *ustálený vodorovný let*. V tuto chvíli se letadlo pohybuje konstantní rychlostí s nulovým zrychlením, nestoupá, neklesá a není v náklonu. Působí na něj síly vyznačené na obrázku 3.1. Jsou jimi vzlak, gravitační tíha, tah a odpor. Při ustáleném letu jsou tyto síly v rovnováze. Ne však všechny navzájem, jak je někdy mylně uváděno, ale pouze dvojice sil působících proti sobě. Pro ilustraci můžeme použít namísto celého letadla pouze křídlový profil, na kterém jsou veličiny více zřejmé.



Obrázek 3.1: Síly působící na letadlo při ustáleném vodorovném letu

Nejdůležitější silou je vztlak. Ten působí kolmo vzhůru od křídla a při vodorovném letu vyrovnává tíhu letadla. Dochází k němu díky typickému profilu křídla, kdy na jeho spodní straně vzniká vyšší tlak vzduchu a současně na horní se tlak vzduchu snižuje. Pro zjištění

celkového vztlaku křídla je potřeba znát množství parametrů, jako je hustota prostředí či rychlost pohybu křídla. Především ale záleží na koeficientech popisujících tvar křídla, jeho plochu, rozpětí a tvar jeho profilu. Pro zjištění celkového vztlak letadla, bylo by samozřejmě nutné započítat dílčí vztlakové síly na celém jeho povrchu.

Další z hlavních sil je tah motoru. Ten je tvořen jednou či více vrtulemi. Lopatky vrtule svým tvarem ženou vzduch za sebe (tedy směrem na trup a křídla letadla), čímž vzniká síla táhnoucí vrtuli, a tím i celé letadlo, dopředu. Tažná síla opět závisí na množství parametrů které popisují stavbu vrtule, počet lopatek a jejich aerodynamické vlastnosti, a samozřejmě také na rychlosti jakou se vrtule otáčí. S vyšší rychlostí stoupá tah a současně se zvyšuje rychlost proudění vzduchu v oblasti za vrtulí. Tím se nepřímo zvyšuje vztlak křídel, ale také se mění citlivost ovládacích prvků letadla.

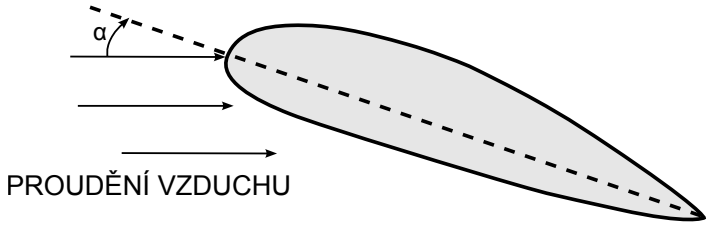
Gravitační síla působící na letadlo je zřejmá a vždy směřuje ke středu Země.

Poslední silou je odpor který klade vzduch letícímu letadlu a působí vždy v opačném směru než je směr jeho pohybu. Je složený ze dvou hlavních složek. Jednou je odpor způsobený tvarem křídla, potažmo celého letadla, a jeho pohybem v prostředí. Druhou složkou je odpor vznikající na koncích křídel. Jak již bylo zmíněno, u horní části křídla se nachází podtlak, zatímco u spodní části je naopak přetlak. V místě kde je křídlo spojeno s trupem letadla se tyto tlaky *nepotkávají*, na opačném konci křídla však ano. Dochází k jejich přirozenému vyrovnávání a vznikají víry, které pokrucují proudění vzduchu a mění směr vztlakové síly křídla. Její pomyslný vektor pak není kolmý ke křídlu, ale mírně sklopený, čímž přispívá k velikosti odporové síly. Čím vyšší vztlak křídlo tvoří, tím více odporové síly současně vzniká. Obecně je pro zjištění odporové síly důležitá rychlost pohybu a charakteristika prostředí, především pak samotný tvar křídel a celého letadla.

Výsledné chování letadla záleží vždy na poměru zmíněných sil. Pokud by křídla netvořila dostatečný vztlak (z důvodu konstrukce nebo nízké rychlosti pohybu), letadlo se nevznese, případně spadne. Je však nutné si uvědomit, že s výjimkou tahu motoru nemá pilot letadla možnost žádné z uvedených parametrů změnit, neboť jsou dány konstrukcí letadla.

3.2 Ovládací prvky a manévry

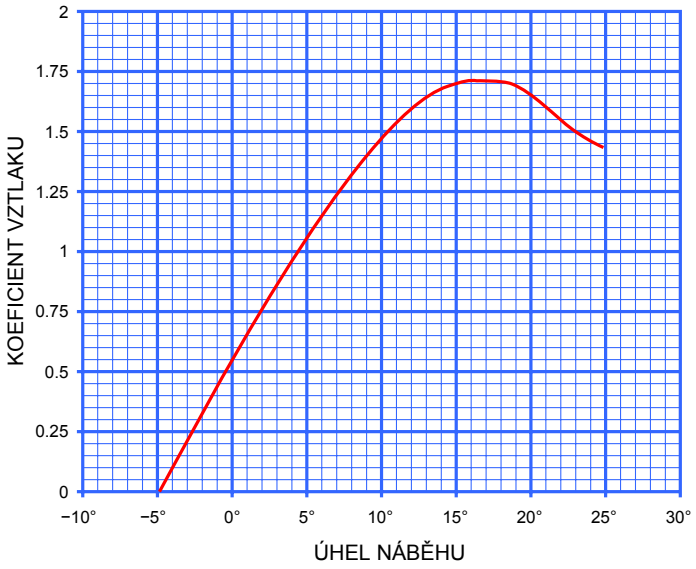
Pro popis jiného než vodorovného letu je potřeba zavést nový pojem, a sice *úhel náběhu*. Jde o úhel, který svírá tětiva profilu křídla s proudem vzduchu, který na ni dopadá. Úhel náběhu je vyznačený na obrázku 3.2.



Obrázek 3.2: Úhel náběhu křídla. Jeho výchozí hodnota je dána konstrukčními vlastnostmi letadla a křídel. Za letu jej lze pak regulovat vysouváním či zasouváním klapek a samozřejmě změnou orientace celého letadla vůči směru proudění vzduchu.

Navzdory časté představě ve skutečnosti neplatí, že by směr letu letadla byl vždy totožný

s osou jdoucí jeho trupem. Při dostatečné rychlosti může letadlo letět vodorovným letem, přičemž trup bude skutečně souběžný s dráhou letu. Vztlak křídla bude v tomto případě dostatečný právě díky vysoké rychlosti proudění. Pokud je ale cílem nižší letová rychlost (například z důvodu úspory paliva), bude pro zachování vodrovného směru nutné aby pilot zvýšil úhel náběhu křídel (použije k tomu výškové kormidlo – viz dále). Tím zvýší i jejich koeficient vztlaku a zamezí klesání letadla. V této chvíli se letadlo pohybuje vodorovně, ale jeho příď je výš než záď a míří tedy mírně nahoru (úhel o který je letadlo nakloněno odpovídá úhlu náběhu). Přibližný vztah mezi úhlem náběhu a vztlakovým koeficientem křídla je znázorněn na obrázku 3.3. Přesný tvar průběhu je opět dán konstrukčními vlastnostmi konkrétního křídla.



Obrázek 3.3: Závislosti vztlakového koeficientu na úhlu náběhu u neidentifikovaného křídla. Při úhlu vyšším než 25° křídlo přestává generovat vztlak a letadlo se začíná propadat. Zdroj: Wikimedia Commons na základě dat z programu XFOIL

Nejvyšší hodnota vztlakového koeficientu je v případě křídla na obrázku vidět v oblasti kolem 17° (opět je nutné připomenout že tento průběh je jen jedním z mnoha možných a například sportovní letadla mohou efektivně využívat i vyšších úhlů náběhu). Při dalším zvyšování úhlu náběhu se mění proudění kolem horní části křídla z *laminárního* (proudnice jsou rovnoběžné a na pohled vypadají jako by byly rozdělné do vrstev) na proudění *turbulentní* (proudnice se trhají a vzájemně prolínají, vznikají vzdušné víry). V důsledku vznikajících turbulencí klesá vztlak křídla a současně roste jeho povrchové tření, čímž stoupá i jeho odpor. Na hranici blízké 25° pak křídlo ztrácí vztlak úplně a letadlo se začne propadat. Této situaci se často říká *přetažení*.

Pilot k řízení letadla používá takzvané *primární ovládací prvky*. Jsou jimi klapky na křídlech a výškové kormidlo, obojí ovládané kniplem, a dále směrové kormidlo ovládané pomocí pedálů. Všechny tyto ovládací prvky ve skutečnosti plní jedinou činnost, a sice že mění úhly náběhu jednotlivých částí letadla. Přestože jsou křídla pevně spojená s trupem, jsou na nich pohyblivé klapky. Stejně tak vodorovné výškové i svislé směrové kormidlo má tvar křídla, jehož část u odtokové hrany je pohyblivá.

Pokud nyní přestaneme dosud popisované síly vztahovat k celému letadlu, ale budeme je uvažovat jako samostatné síly vznikající vždy na konkrétním křídle (za křídla zde považujeme i výškové a směrové kormidlo), můžeme snadno popsat základní letecké manévry.

Proces stoupání či klesání pilot reguluje nastavením klapek výškového kormidla. V případě stoupání klapky zvedne, čímž sníží úhel náběhu kormidla a tím i jeho vztlak. Záď letadla začne klesat, úhel náběhu hlavních křídel letadla se začne zvyšovat a s ním i jejich vztlak, potažmo vztlak letadla jako celku. Při klesání je situace analogická, pouze se veličiny pohybují opačným směrem.

U zatáčecího manévru často panuje přesvědčení, že k jeho provedení slouží pouze směrové kormidlo, od kterého se už podle názvu očekává že bude řídit směr letu. Z předchozích odstavců již můžeme tušit že tomu tak není. Pokud by pilot při vodorovném letu použil směrové kormidlo, záď letadla by se sice vůči směru letu natočila, samotný směr letu by se však změnil jen málo. Tím jak se osa letadla natáčí vůči směru letu současně roste proudění kolem křídla které je více vystaveno přímému směru letu. Tím na křídle vzniká větší vztlak a letadlo začíná podélně rotovat, ačkoliv je knipl stále ve výchozí poloze.

Správný průlet zatáčkou probíhá v náklonu. Pilot nejdříve zatočením kniplu nakloní klapky na křídlech (které se narozdíl od výškového a směrového kormidla pohybují navzájem protichůdně) tak, že na jednom z křídel se vztlak zvýší, zatímco na druhém se sníží. Letadlo tedy letí v náklonu a vztlak vznikající na křídlech nepůsobí kolmo vzhůru, ale kolmo k horním stranám křídel (vztlak vzniká stále stejně, pouze jeho směr již není přímo opačný ke gravitační síle). Můžeme tedy vztlakovou sílu pomyslně rozložit na skutečný vztlak držící letadlo ve vzduchu a složku vodorovnou, která působí jako dostředivá síla a nutí letadlo skutečně zatáčet. Směrové kormidlo při průletu zatáčkou slouží regulaci vzájemné polohy přídě a zádi letadla. Některá letadla mají totiž tendenci v důsledku různě velkého odporu na křídlech (například pokud letadlo automaticky nenastavuje klapky do vzájemně opačných poloh) *sklouzávat* po přídi nebo zádi do středu pomyslného kužele po jehož vnitřní stěně letadlo zatáčkou prolétává. Výškové kormidlo může zatáčecí manévr doplnit na vzestupný či sestupný.

3.3 Stabilita letadla

Každé letadlo má svou konstrukcí dané stabilizační schopnosti, které se uplatňují aniž by pilot zasahoval do řízení. Stabilitu se dělí na statickou a dynamickou.

Statická stabilita popisuje reakci letadla ve chvíli kdy dojde k narušení jeho dosavadního směru letu například poryvem větru. U staticky stabilního letadla vznikne opačná síla snažící se letadlo vrátit do původní polohy. Staticky neutrální letadlo bude pokračovat v pozměněném směru a u staticky nestabilního letadla se dokonce objeví síla působící spolu s rušivým vlivem, která odklon od původního směru ještě více posílí.

Dynamická stabilita popisuje stabilitu letadla v dlouhodobém horizontu. Dynamicky stabilní letadlo sice bude při vyrovnávání svého směru oscilovat kolem rovnovážné polohy (vodorovného letu), ale po čase se v něm ustálí. Dynamicky neutrální letadlo bude oscilovat s pořád stejnou sílou a dynamicky nestabilní letadlo bude od rovnovážné polohy dokonce divergovat.

Letadlo které je staticky i dynamicky stabilní je většinou snáze ovladatelné a vhodné pro začínající piloty. Pokud je však tendence letadla dosáhnout stabilní polohy příliš velká, může to být ke škodě, neboť bude k ovládání letadla nutné vynaložit vyšší sílu. Ač se to nemusí zdát, není vždy cílem zkonstruovat dokonale stabilní letadlo a v některých případech může být vhodné, pokud se letadlo chová neutrálně nebo dokonce nestabilně.

3.4 Herní ovládání letadla

V následujících podkapitolách bude popsáno zapojení výše vysvětlených mechanik letu do leteckých her. Zaměřím se přitom především na takzvané *causual* hry, tedy oddechové hry které od hráče nevyžadují vysokou úroveň zkušeností. U her a programů profilujících se jako simulátory je zřejmá snaha co nejvíce se přiblížit reálné předloze jak u letové dynamiky, tak i z hlediska ovládání letadla. I přesto lze u simulátorů pozorovat využití některých zjednodušení zejména z oblasti ovládání.

3.4.1 Chování letadla

V předchozích kapitolách byla zmíněna složitost numericky přesného výpočtu všech sil působících na letadlo. Nejen že by bylo potřeba znát až desítky reálných koeficientů pro každý jeden typ letadla, ale požadavky na výpočetní výkon by byly velmi vysoké (tím spíše pokud uvažujeme mobilní zařízení). Proto se v případě leteckých simulátorů a her používají matematické modely s různou úrovní věrnosti fyzikální předloze [11].

Je pochopitelné, že v případě simulátorů pro výcvik pilotů je maximální věrnost simulace nezbytná. V případě leteckých her se však ukazuje, že příliš reálně ovládané letadlo je spíše na obtíž. Hráč causual her neumí a často ani nechce umět plnohodnotně ovládat letadlo. Jeho prvotním cílem je splnění herních úkolů, typicky sestřelení protivníka nebo let na dané místo. Letadlo je v této kategorii her pouze prostředkem k dosažení cíle, nikoliv cílem samotným.

Do matematických modelů se proto vkládají zjednodušující předpoklady, jako například stabilita letadla při zatáčení. To potom nemá tendence v zatáčce sklouzávat do jejího středu (tento jev byl detailněji popsán v kapitole 3.2). Častým zjednodušením je také nemožnost dosáhnout bodu přetažení (opět viz kap. 3.2). Pomyslný omezovač nedovolí další zvyšování úhlu náběhu a letadlo přetažení nikdy nedosáhne. Tím odpadne část simulace, kdy se letadlo ocitá ve vzduchu s minimálním nebo žádným vztlakem a mělo by začít padat. Jelikož však běžný hráč nemá nejmenší ponětí o úhlu náběhu a jeho důsledcích na let, je tohoto zjednodušení často využíváno. Nevýhodou tohoto přístupu může být pro hráče viditelné omezení manévrovacích možností. Pokud se ale stále držíme v kategorii her které se nesnaží býti skutečnými simulátory, není tento problém nijak závažný. Navíc lze vhodným návrhem hry takové omezení ospravedlnit, například použitím grafického modelu letadla který v hráči vyvolá dojem těžkopádnosti a nízkého výkonu. Od takového letadla potom podvědomě nebude očekávat velké manévrovací schopnosti a snáze pochopí, když narazí na jejich hranice, které ve skutečnosti slouží primárně ke zjednodušení celého výpočtu.

Podobně lze mlčky předpokládat existenci nějakého inteligentního řízení, které hráči asistuje. To může kontrolovat další ovládací prvky letadla sloužící zejména ke zjednodušení práce pilota v některých specifických situacích. (Jde mimo jiné o tzv. *sekundární ovládací prvky*, jejichž popis je však nad rámec této práce.) Takový pomocný systém je zcela jistě reálný a můžeme na něj nahlížet jako na jistou formu autopilota.

Z pohledu ovládání letadla se hry v drtivé většině omezují výhradně na klapky na křídlech a výškové kormidlo. Pokud uvažujeme zmíněné *asistované řízení*, lze těmito dvěma prvky letadlo plnohodnotně (stále pouze z hlediska hry, nikoliv simulátoru) ovládat. V některých případech bývá doplněna i možnost ovládání směrového kormidla. U něj však nastává časté nepochopení jeho funkce a bývá implementováno jako možnost zatáčení letadla bez náklonu při zachování vodorovného letu. Takové letadlo pak zatáčí podobně jako například

ponorka. Je otázkou, zda toto pramení z neznalosti vývojáře hry anebo zda jde o objednávku samotných hráčů. Průlet zatáčkou je totiž při použití *nepochopeného* směrového kormidla mnohem snazší, dává hráči větší kontrolu nad letadlem a přirozeně snižuje požadavky na hráčovu zručnost. Ve výsledku záleží na herním vývojáři a na návrhu a zaměření hry, zda se bude držet reálného (byť zjednodušeného) ovládání letadla anebo se podvolí poptávce po snadno ovládaném letadle, přestože by tím popíral fyzikální zákonitosti.

V neposlední řadě u většiny leteckých her chybí možnost regulovat tah motoru. To je veličina se kterou pilot skutečného letadla v průběhu letu a především pak manévrů aktivně pracuje. Je ale také jednou z veličin ovlivňujících celkový vztlak letadla a při neznalosti všech důsledků by pro hráče bylo snadné ztratit nad letadlem kontrolu. Stejně tak se často ignoruje dříve zmiňovaná změna citlivosti ovládacích prvků letadla v závislosti na tahu motoru. Obvyklým řešením bývá opět předpoklad existence nějakého palubního systému který tah motoru automaticky reguluje. Ve výsledku bývá v průběhu letu tah konstantní a k jeho změnám dochází pouze při vzletu nebo přistávání. Často také informace o otáčkách motoru úplně chybí a jedinou zpětnou vazbou hráči je zvuk který motor vydává.

3.4.2 Vstupní metody

Pokud se podíváme na letecké hry pro počítače, případně pro herní konzole, jsou ovládací možnosti vcelku omezené. Prakticky vždy se omezují na ovládání ve dvou osách, kdy pohyb v jedné ose řídí klapky na křídlech letadla a pohyb v druhé ose ovládá výškové kormidlo. Subjektivně je asi nejméně pohodlnou a současně nejméně přesnou metodou ovládání klávesnice počítače. Reakce na řízení jsou z povahy kláves skokové a nelze ovlivňovat jejich intenzitu. Pozitivem použití klávesnice může být velké množství kláves, které můžou ovládat další prvky letadla. Tyto možnosti však využijí spíše simulátory.

O mnoho lepší kontrolu nabízí joystick, případně jeho zmenšená varianta na ovladačích současných herních konzolí. Ovládání je opět dvouosé, ovšem s možností plynule regulovat intenzitu. Nejdůležitější vlastností je ale podobnost se skutečným řízením letadla pomocí kniplu. To umožňuje především v případě joysticku mnohem vyšší vtáhnutí hráče do hry. Nevýhodou může být omezený počet doplňkových ovládacích prvků, typicky je na ovladači pouze několik málo dalších tlačítek. Pokud by hra nabízela možnost detailnějšího ovládání letadla (například vysouvání podvozku nebo vypouštění světlic pro odlákání nepřátelských střel), mohl by to být problém.

Rozšíření výkonných mobilních zařízení umožnilo příchod nových metod ovládání. Zcela nepoužitelná byla první generace mobilních telefonů s klávesnicí, jelikož ty ještě neměly dostatek výkonu a k rozšíření leteckých her u nich nedošlo. Další popis se tedy zaměří na zařízení s velkým dotykovým displejem. Taková zařízení často nemají hardwarová tlačítka nebo taková tlačítka slouží systému a hra je pro své potřeby nemůže využít. Novým prvkem je ale zapojení sensorů. Konkrétně gyroskopu a akcelerometru, které sledují orientaci, respektive zrychlení zařízení v trojrozměrném prostoru. Hráč ovládá letadlo nakláněním zařízení přičemž se typicky neuvažuje svislá osa. Natočení do stran (pohyb podobný zatáčení volantem) ovládá klapky letadla a naklonění od sebe či k sobě ovládá výškové kormidlo.

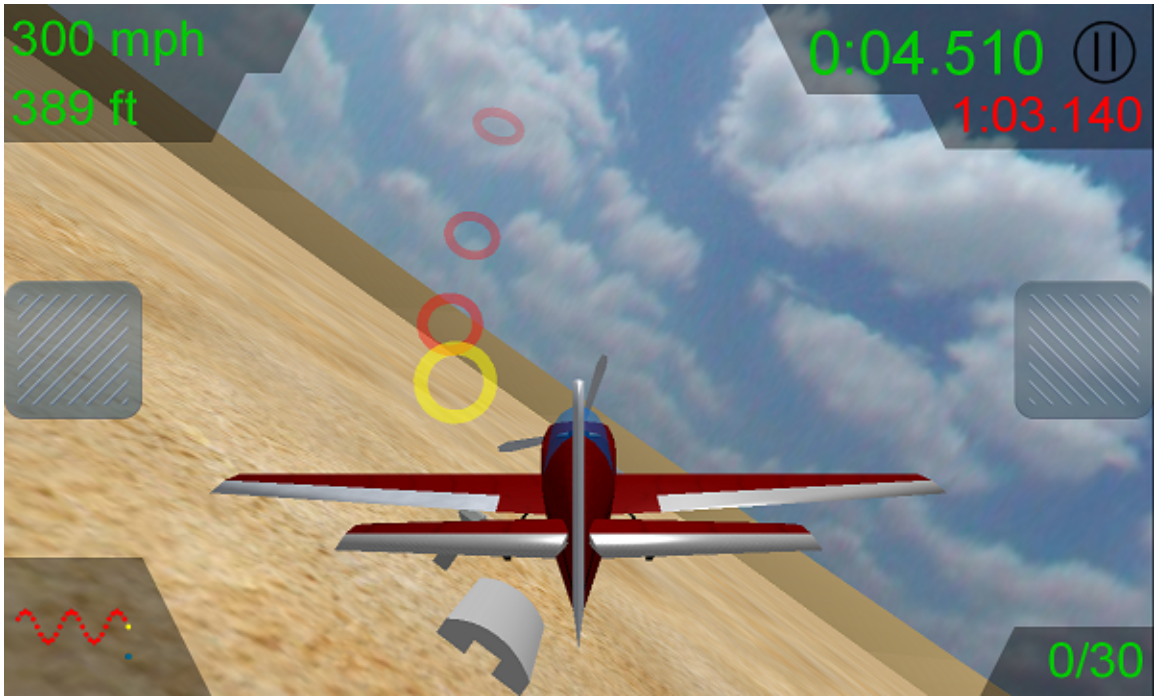
Ovládání her pomocí sensorů se rychle rozšířilo, pravděpodobně díky své inovativnosti, a v současnosti si udržuje přinejmenším u leteckých her většinový podíl. Má však také několik nedostatků. Asi nejzásadnějším je výchozí pozice zařízení. Při používání sensorů je potřeba stanovit referenční bod ke kterému naměřené hodnoty vztahujeme. Častým řešením je nastavit bod na začátku každého letu, poté se již měnit přirozeně nemůže. Problém

nastává ve chvíli, kdy se hráč hrající hru sám pohybuje ve skutečném světě. Může jít například o houpání se v křesle anebo přesun v nějakém dopravním prostředku. Hry se v takovém případě stávají nehratelnými a zatím se neobjevila metoda, která by uměla tento jev odstranit.

Druhý nedostatek stojící za zmínku je změna úhlu pohledu při pohybu se zařízením. Displeje mají většinou relativně velký pozorovací úhel, i tak ale může docházet ke snižování viditelnosti pokud je zařízení až moc přikloněno, respektive odkloněno od pohledu hráče. Řešením může být vhodná volba citlivosti tak, aby stačilo zařízení naklánět jen v *pohledově bezpečném* rozsahu. Stejně tak může být problémové i otáčení displeje. Přestože zde se úhel pohledu nemění, dochází k otáčení celého obrazu včetně uživatelského rozhraní. Na tuto skutečnost je nutné myslet při návrhu hry tak, aby uživatel špatnou čitelností prvků rozhraní nepřicházel o důležité informace.

Dalším problémem je relativně pomalá odezva sensorického vstupu a tvar jeho průběhu. Zpoždění není zásadní, ale je pozorovatelné. Hru je navíc možné navrhnout tak, aby pro hráče bylo pochopitelné. Při letu s velkým těžkopádným letadlem asi nikdo nebude očekávat reakce v řádu zlomků sekundy. U rychlých a akčních her by však toto mohlo být omezující. Průběh signálu vznikajícího v sensoru navíc není ideální a běžně obsahuje výkyvy a šum. Je proto nutné jej filtrovat, čímž může docházet ke ztrátě přesnosti.

Poslední slabinou ovládání pomocí sensorů je společenská vhodnost. S tím jak se herní zařízení stala mobilními se rozšířila i místa kde může člověk hry hrát. Ne vždy a všude bude okolí tolerovat hráče který před sebou mává mobilním telefonem či tabletem. Vývojář sice nemůže ovlivnit na jakých místech budou hráči jeho hru hrát, může se však přinejmenším zamyslet kdo bude cílovou skupinou a nakolik jsou členové této skupiny ochotní se ve společnosti tímto způsobem zviditelňovat.



Obrázek 3.4: Screenshot hry Race Pilot ovládané pomocí sensorů. Šedé čtverce po stranách ovládají směrové kormidlo, které je však implementováno nerealisticky; zdroj: Google Play

Z pohledu mobilních zařízení je asi nejzásadnější možnost ovládání dotykem. Ta se neomezuje pouze na tlačítka a polohovací páčky a je zde možné navrhnout téměř libovolné ovládací prvky. K tomu lze využít položení prstu na displej, který odpovídá stisku klávesy, a může tedy řídit dvoustavové veličiny. Současně je možné sledovat pohyb jednoho či více prstů v nějaké předem dané oblasti a získávat plynulé změny hodnoty (například vzdálenost bodu dotyku od referenčního bodu). Dalším velmi silným prostředkem je vzájemná poloha dvou a více bodů dotyku.

Ve srovnání se sensorickým ovládáním netrpí zmiňovanými nedostatky jako je nutné natáčení a naklánění zařízení a nepoužitelnost v případě že je hráč v pohybu. Při návrhu dotykového ovládání je však nutné počítat s tím, že hráč může svými prsty překrývat značnou část displeje. To se může projevit zejména u displejů s menší úhlopříčkou. Stejně tak je vhodné se při návrhu ovládání zamýšlet nad tím, kdo bude typickým hráčem hry. Zcela jistě bude rozdíl mezi prstem dítěte a dospělého jedince, navíc s přihlédnutím k vykonávané profesi, a je nutné tomuto ovládací metodu přizpůsobit.

Přestože se zdá že právě dotykové ovládání nabízí široké možnosti pro inovace, herní vývojáři je zatím spíše nevyužívají. V případě leteckých her drtivě převládá využití sensorů, případně dotykového ovladače na displeji – bodu který hráč dotykem posouvá a simuluje tak pohyb joysticku.

Kapitola 4

Návrh dotykového ovládání a letecké hry

V této části práce popíši nově navrhovanou metodu ovládání a varianty její implementace. Dále rozvedu jak by měla výsledná hra vypadat a jaké budou její herní mechanismy. Stále je potřeba mít na paměti, že se pohybujeme v oblasti her pro mobilní zařízení. Rozměry obrazovek, tedy ovládacích ploch, budou omezené, stejně jako výkon zařízení. Stejně tak povaha práce s mobilním zařízením je diametrálně odlišná od povahy práce s běžným počítačem nebo herní konzolí. Tyto faktory bude nutné uvážit.

Při navrhování hry jsem vycházel zejména z knih Game Industry, které sepsali čeští herní vývojáři [12, 13].

4.1 Nově navrhovaná metoda ovládání

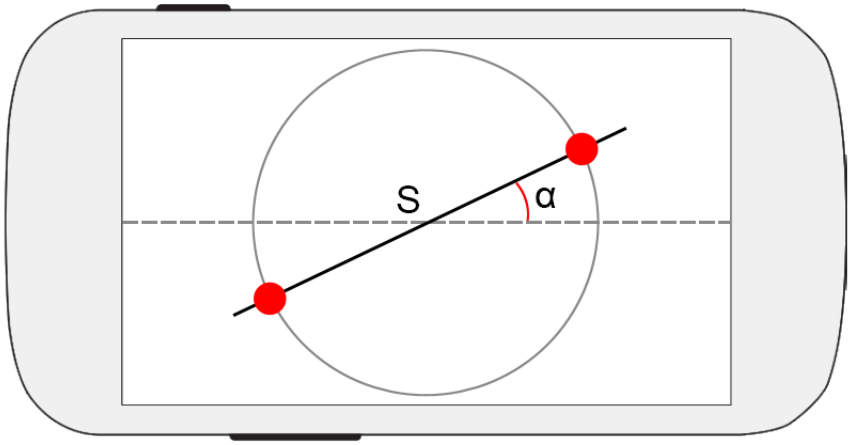
Při uvažování o inovativním přístupu k ovládání letadla jsem téměř okamžitě vyřadil vstup pomocí sensorů. Důvody zmíněné v kapitole 3.4.2 mi přijdou příliš omezující. Současně v této oblasti nejsou velké možnosti k dalšímu rozšiřování tak, aby pro hráče zůstala zachovaná pohodlnost a intuitivnost ovládání.

Navrhované ovládání tedy bude čistě dotykové. Tím bude možné jej používat kdekoliv a jeho hlavními přednostmi bude vysoká přesnost a rychlá odezva. Díky tomu na něm půjde vystavět i velmi rychlou a akční hru. Oproti existujícímu dotykovému vstupu by mělo hráči nabídnou větší vtažení do děje a snad i navodit pocit jisté jedinečnosti, tedy že hráč používá něco nového a neobvyklého. Proti této myšlence jde částečně požadavek na nízké znalosti a zkušenosti. Hráč by měl princip ovládání pochopit prakticky ihned, v nejlepším případě pak bez jakékoliv nápovědy nebo předchozího proškolení. Současně musí být ovládání maximálně pohodlné. To totiž opět není cílem hry, ale pouze prostředkem k jeho dosažení. V ideálním případě by mělo být pro hráče natolik přirozené, že nad ním nebude muset přemýšlet či ještě lépe vůbec si uvědomovat jeho existenci. Pochopitelně jej také nesmí omezovat a mělo by postihnout všechny podstatné funkce ovládaného objektu, v tomto případě letadla.

Pro další popis rozdělím ovládání do jednotlivých os. První z nich bude osa ovládající klapky na křídlech a tím i náklon celého letadla, tedy rotaci kolem osy definované směrem letu. V případě joysticku a z něj vycházejících metod jde o pohyb ukazatele či ovladače do stran. Navrhovaná metoda ovládání vyžaduje aby existovaly právě dva body ve kterých se hráč dotýká displeje. Tyto se použijí k sestrojení pomyslné kružnice, jejíž střed bude ležet

na středu úsečky definované oběma body. Úhel náklonu letadla pak odpovídá úhlu o který se spojnice bodů odchyluje od vodorovné osy kružnice. Situace je zachycená na obrázku 4.1. Ačkoliv to z popisu nemusí být zřejmé, na poloměru kružnice ve skutečnosti vůbec nezáleží. Hráč může průběžně vzdálenost bodů měnit a pokud oba body zůstanou na původní úsečce, náklon letadla se nezmění. Vedlejším efektem je pak možnost takto měnit citlivost ovládání. Zatímco u kružnice o velkém poloměru je možné dosáhnout velmi jemné regulace náklonu, při přibližování bodů a zmenšování poloměru se citlivost přirozeně zvyšuje. Důležité je, že nezáleží na pozici kružnice v rámci dotykové plochy a v každém okamžiku se vytváří tak, aby její střed ležel ve středu spojnice mezi oběma body dotyku. Všechny dvojice protilehlých bodů takové kružnice pak definují manévrovací možnosti v dané ose rotace. Ty budou bez dalších omezení odpovídat 360°. Pro ovládání letadla však bude vhodnější použít rozsah $-180°$ až 180°, kde nulový úhel náklonu bude odpovídat vodorovné ose obrazovky. (Přestože se tato práce soutřeďuje na letecké hry, je jistě možné navrhovanou metodu ovládání použít pro ovládání zcela jiných objektů. V takovém případě můžou být vhodné rozsahy vstupů, stejně jako neutrální poloha, pochopitelně odlišné.) V zájmu sjednocení rozsahů vstupních hodnot z obou os bude posledním krokem převod do rozsahu -1 až 1, kde opět 0 představuje neutrální hodnotu.

Je vhodné zde také zdůraznit, že rozsah hodnot ovládání nemusí přímo odpovídat silám aplikovaným na ovládaný objekt. V případě letadla si lze hodnoty ovládání představit jako úroveň přitažení kniplu nebo sešlápnutí pedálu. Jde o bezrozměrné hodnoty a teprve jejich *dosazením* na vstup konkrétního matematického modelu letadla vzniknou síly na něj aplikované. To odpovídá skutečnosti, kde malé letecké letadlo používá rámcově stejné základní ovládací prvky jako například velký dopravní letoun. A teprve rozměry a ostatní charakteristiky ovládacích ploch letadla určují, jak moc letadlo změní směr svého letu.



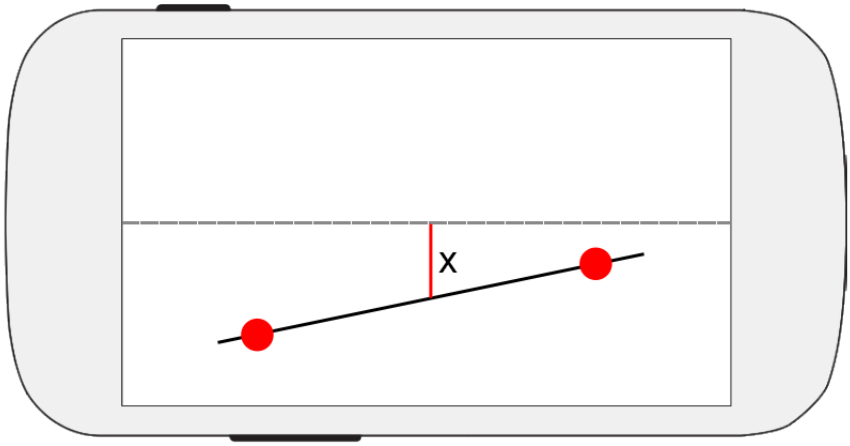
Obrázek 4.1: Ovládání náklonu letadla. Červená místa značí body dotyku prstů. Intenzita změny náklonu odpovídá úhlu, který svírá úsečka ohraničená těmito body s vodorovnou osou obrazovky. Body je možné podél vodorovné osy libovolně posouvat bez vlivu na řízení.

Druhou z ovládaných veličin je výškové kormidlo. To bývá u joysticku a podobných metod reprezentováno přitáhnutím nebo odtlačením ovládací páčky. Princip nastiňuje obrázek 4.2. Opět zde pracujeme se středem úsečky mezi dvěma body dotyku a odečítáme jeho vzdálenost od referenčního bodu, který je na obrázku znázorněn uprostřed displeje. Střed displeje je zvolen pouze pro ilustraci a ve skutečnosti se referenční bod přemístí pokaždé

když se displeje začnou dotýkat právě dva prsty. V okamžik prvotního dotyku se referenční bod nastaví a nemění se dokud se oba prsty nepřestanou displeje dotýkat. Jakmile se znovu dotknou, referenční bod se přesune na novou pozici na které opět po dobu kontaktu prstů s displejem zůstává. Hráč tak při pouhém položení prstů na displej bez dalšího pohybu nijak nezasahuje do ovládání letadla. Současně tento přístup umožňuje hráči začít letadlo ovládat dotykem v libovolné části displeje, což může být výhodné zejména u tabletů a obecně zařízení s větší úhlopříčkou dotykové plochy.

Funkční rozsah tohoto vstupu bude omezený hranami obrazovky. Zde se nabízí více řešení problému. Je možné jako rozsah definovat celou výšku obrazovky. V místě prvotního dotyku se pak nastaví neutrální hladina s tím, že obě oblasti (kladná i záporná) budou mít stále výšku poloviny obrazovky, třebaže by zasahovaly mimo oblast obrazovky. Takový přístup nabízí optimální chování v situaci, kdy prvotní dotyk nastává právě v polovině výšky obrazovky. V ostatních pozicích dochází k oříznutí rozsahu a tím ke zhoršení manévrovacích možností. Jiným přístupem může být zmenšení výšky oblastí tak, aby žádná z nich nepřesahovala oblast obrazovky. Obě ovládací oblasti by samozřejmě měly být shodně velké, a rozměr by se tedy řídil podle menší z nich. Slabinou tohoto přístupu je různá citlivost vstupu v závislosti na místě prvotního dotyku. Optimální chování je opět v oblasti kolem středu obrazovky, s klesající vzdáleností k jejímu okraji pak roste citlivost až do místa, kde vstup přestává být použitelný.

Konkrétní implementace výškového kormidla může opět záviset na povaze ovládaného objektu. V případě této práce jsem experimentoval s více variantami. Více k tomuto tématu bude popsáno v kapitole 5.1.



Obrázek 4.2: Ovládání vzestupu a poklesu letadla. Červená místa značí body dotyku prstů a současně definují úsečku. Intenzita změny výšky letadla závisí na vzdálenosti středu této úsečky od vodorovné osy obrazovky. Body je možné podél vodorovné osy libovolně posouvat bez vlivu na řízení.

Kombinací těchto dvou principů vzniká ovládání schopné pokrýt stejný rozsah veličin jako zmiňovaný *klasický dotykový* anebo sensorový vstup. Skládá se přitom ze známých prvků – ovládání náklonu může hráči evokovat zatáčení volantem, změna výšky pak gesto běžně používané pro svislé posouvání obsahu. Je proto možné předpokládat, že navrhované ovládání bude pro hráče skutečně intuitivní a k jeho vysvětlení nebude potřeba více než

dva nákresy uvedené v této kapitole.

Navrhované ovládání by bylo možné dále rozšířit o možnost ovládání směrového kormidla. To by představovalo třetí ovládací osu, tedy svislou osu kolem které může letadlo rotovat. Veličinu by bylo možné odečítat podobně jako je odečítána poloha výškového kormidla. Šlo by o vzdálenost středu úsečky tvořené dvěma body dotyku od svislé osy obrazovky, procházející bodem počátečního dotyku. (Výškové kormidlo zde měří vzdálenost od osy vodorovné.) Využití této třetí ovládací osy však nemusí být ideální a navrhované ovládání jako celek by mohlo na hráče působit příliš složitě. Ideálním řešením by bylo tuto variantu implementovat a získat zpětnou vazbu od prvních zkušebních hráčů.

4.2 Žánrové zařazení a herní mechanismy

Hra se bude odehrávat v prostředí leteckých závodů. Hráč je v roli pilota akrobatického letadla který navštěvuje jednotlivé závody. Samotný závod pak spočívá v průletu branek ve stanoveném pořadí (přesná dráha letu se nevyžaduje), vzlet ani přistávání se neuvažuje. Branky můžou být různého typu: obyčejná, jednosměrná nebo branka vyžadující průlet v předem dané poloze letadla. Právě poslední typ branky je typickým prvkem leteckých závodů, kdy nezáleží pouze na času letu, ale hodnotí se také přesnost se kterou pilot brankou proletí. Rozlišuje se průlet vodorovný, svislý (takzvaný *nožový let*) a průlet vzhůru nohama.

Cílem hráče je proletět všemi brankami v co nejkratším čase. Při průletu brankou vyžadující konkrétní polohu letadla se kontroluje přesnost průletu. Pokud je nízká, přičte se k času průletu postih závisející na velikosti odchylky. Průlet brankou se však započítá i v případě, že bude odchylka maximální. To je rozdíl oproti jednosměrné brance, která se považuje za splněnou pouze při průletu ze správného směru. U obyčejné branky žádná omezení ani postihy nejsou. U každého závodu jsou definovány tři časy průletů, které se hráč snaží překonat. Tyto časy vzniknou pravděpodobně v prvotní fázi odhadem, později se upraví na základě testování a zpětné vazby od hráčů.

Celá hra se pak sestává s množství závodů, kde každý se může odehrávat v jiném grafickém prostředí (ne však nutně vždy unikátním). Jednotlivé závody spolu ale nijak nesouvisejí. Na začátku má hráč zpřístupněný jediný závod do kterého může vstoupit, ostatní závody jsou uzamčené. Ve chvíli kdy hráč v některém ze závodů překoná alespoň jeden z předdefinovaných časů, považuje se závod za splněný a zpřístupní se jeden další závod. Nezbytným předpokladem pro plynulý průchod všemi závody bude postupně se zvyšující obtížnost. Tak jako porostou zkušenosti a dovednosti hráče, musí stoupat i velikost výzvy před kterou je postaven. V případě příliš jednoduchých závodů by hráč procházel bez nejmešího odporu a hra by byla nudná. U příliš obtížných závodů by se naopak hráč mohl zaseknout bez možnosti pokračovat dál. Motivací pro nejlepší možné splnění závodu, tedy překonání nejnižšího z časů, pak mohou být speciální odměny, jako například nové letadlo nebo jiné dekorativní prvky.

V počáteční fázi (tedy prvních vydaných verzích hry) není žádoucí zpřístupnit hráči více letadel s různým výkonem, třebaže by se takové letadlo nabízelo jako vhodná odměna za vynikající výkony. Taková možnost by totiž zvýšila obtížnost odhadu přiměřené náročnosti závodu. Lze jistě předpokládat, že hráči nováčkovi potrvá nějakou dobu, než dostane letadlo plně pod kontrolu a naučí se jej ve všech situacích bezpečně ovládat. V takové situaci není nutné hráče zatěžovat úvahami o rychlosti či ovladatelnosti letadla. Naopak bude vhodné nabídnout pouze jediné letadlo a hráče odměňovat právě formou dekorativních úprav, například jiného laku (tedy textury) letadla.

Uvedení různě výkonných a ovladatelných letadel může přijít ve chvíli, kdy bude možné předpokládat jistou zkušenost hráče. Takový hráč už letadlo plně ovládá a hlavná výzva závodů se přesouvá od schopnosti proletět všechny branky (v téměř libovolném čase) k cíli proletět je co nejrychleji. Zde bude možné zpřísnit požadavky na rychlost průletu závodem, včetně možnosti definovat časy na první pohled nesplnitelně. To hráče přinutí vrátit se k dřívějším závodům, dosáhnout v nich nejlepšího hodnocení (to by v této chvíli měl zvládnout, neboť jeho zkušenosti vzrostly) a odměnou získat rychlejší letadlo, se kterým již bude možné plnit obtížnější závody.

Celkově by hra měla působit akčně a rychle, čemuž musí odpovídat i metoda ovládání. Právě u akrobatických letů je potřeba aby odezva na vstup byla minimální a hráč neustále cítil že má letadlo plně pod kontrolou. Takovouto hru by zcela jistě bylo velmi obtížné ovládat pomocí sensorického vstupu, a je proto dobrým prostředkem pro prezentaci předností nově navrhovaného ovládání.

4.3 Rozšiřitelnost hry

Podstatným faktorem je také následná rozšiřitelnost hry. Aktualizace aplikace se nemusí zaměřovat pouze na opravování chyb, ale mohou současně přidávat další herní obsah. To je z pohledu hráče velmi žádoucí a právě informaci o aktualizaci mu může připomenout dávno dohranou hru. Aktualizace se pak stávají menšinovým propagačním nástrojem a je velmi žádoucí s nimi takto pracovat. Příliš časté vydávání nových verzí může být obtěžující, zejména pokud hráč většinu herního obsahu zkonzumoval a nového se mu nedostává. I v případě že má vývojář větší množství nevydaného obsahu, je vhodné jeho zveřejňování dávkovat a snažit se tak udržovat dlouhodobý zájem o hru.

Konkrétně u navrhované hry je relativně jednoduché doplnit další herní režimy. Ve chvíli kdy je implementováno ovládání letadla a jeho interakce s herním světem, je jen otázkou téměř triviální logiky, zda se budou počítat body za prolétnuté branky nebo zda bude cílem proletět trasu v co nejkratším čase, případně cokoliv jiného.

Podobně je tomu v případě vytváření nových herních úrovní, tedy závodů. Hra nepočítá s unikátním prostředím pro každý jeden závod a scenérie se nutně budou opakovat. Nejjednodušším doplněním obsahu tedy může být využití existujícího prostředí a vytvoření pouze nové sady branek. O něco náročnějším rozšířením by pak bylo doplnění nového prostředí. Tam je náročnost zejména na straně grafických prací. Tímto způsobem je možné připravovat tématické balíčky s novými úrovněmi k různým významným příležitostem, což může být opět zajímavý propagační prvek.

Z tohoto hlediska nebude nutné implementovat všechny herní mechanismy z předchozí kapitoly ihned. S ohledem na časové možnosti a zmíněného dávkování obsahu může být dokonce vhodné začít s jednoduchou hrou a tu následně rozšiřovat. Zpětná vazba od hráčů a informace získané ze statistik užívání pak mohou poskytnout indicie k tomu, kterým směrem hru dále vyvíjet. Současně se v případě neúspěchu minimalizuje úsilí vynaložené k vytvoření hry, a tím tedy i ztráty.

V neposlední řadě je potřeba uvažovat o monetizaci hry, tedy o její ziskovosti. Vzhledem k výše popsaným mechanismům se nabízí jako vhodné řešení zpoplatnění některých závodů, případně možnost zakoupit si předčasně některé odměny. Opět ale bude potřeba dbát na vyváženost hry a zachování její zábavnosti. Nelze proto hráči ihned nabídnout vykonnější letadlo, třebaže za poplatek. Vhodnější bude počkat do chvíle, kdy by toto letadlo mohl získat přirozeně a teprve poté jej nabídnout ke koupi.

Kapitola 5

Implementace a publikování hry

V této části popíši implementaci navrženého ovládání a hry. Veškerá práce probíhala ve vývojovém prostředí Eclipse, doplněné o moduly specifické pro platformu Android. Tyto doplňky jsou součástí podkladů pro vývojáře a jsou dostupné z oficiálních webových stránek [2].

Poslední podkapitola bude věnovaná publikování vytvořené hry nejdříve v omezené skupině lidí a následně na internetovém obchodě Google Play. Zhodnotím také výsledky testování, průzkumů a poznatky získané sběrem statistik užívání. V závěru kapitoly také shrnu možnosti dalšího vývoje.

5.1 Prototyp a prvotní testování

V počátku práce na aplikaci jsem navrhl a impementoval prototyp založený čistě na platformě Android, bez použití dalších knihoven. Potýkal jsem se s problémy, které byly popsány v kapitole 2.3.1, a zpětně lze prohlásit, že vývoj bez pomocných nástrojů či knihoven bude přinejmenším velmi obtížný.

Cílem prototypu byla prvotní implementace navržného ovládání a validace jeho funkcionality. Obecně by mělo jít o maximálně zjednodušenou verzi aplikace, kde bude možné velmi rychle doplnit libovolnou funkcionalitu, především z hlediska vstupních metod. Přestože měl být prototyp takřka jednoúčelovou aplikací, jeho vnitřní struktura je v mnohém podobná struktuře aplikace používající knihovnu Libgdx. To je pravděpodobně z velké části dáno přímo platformou Android, která se návrhem svého rozhraní (a samozřejmě i všemi ukázkovými příklady) snaží vést programátora ke správně navržené struktuře aplikace. Správná struktura v tomto kontextu znamená aplikaci, která je vnitřně přehledná a efektivní, její součásti pak znovupoužitelné a dobře testovatelné. Tyto požadavky nemají v případě prototypu velkou prioritu (snad s výjimkou znovupoužitelnosti součástí), přesto je vhodné zmínit, že jsem jich při práci na prototypu bez velkého úsilí dosáhl anebo se jim alespoň přiblížil.

Z pohledu uživatele je prototyp velmi omezenou verzí hry. Neobsahuje uživatelské rozhraní, vizuální stránka je strohá a stejně tak nejsou implementovány ani herní mechanismy týkající se závodů. Dalo by se říct že jde pouze o letadlo nekonečně se pohybující v prostoru, neboť detekce kolizí, ať už s terénem anebo průlet kruhy, v prototypu chybí také. Pro představu o prvotní verzi vizte obrázek 5.1. Později do hry přibylo více vizuálních prvků, především model ostrova pro experimenty s terénem. Pro lepší pocit ze hry při jejím testování jsem také nahradil hlínu texturou vody. Závěrečná podoba prototypu je na obrázku

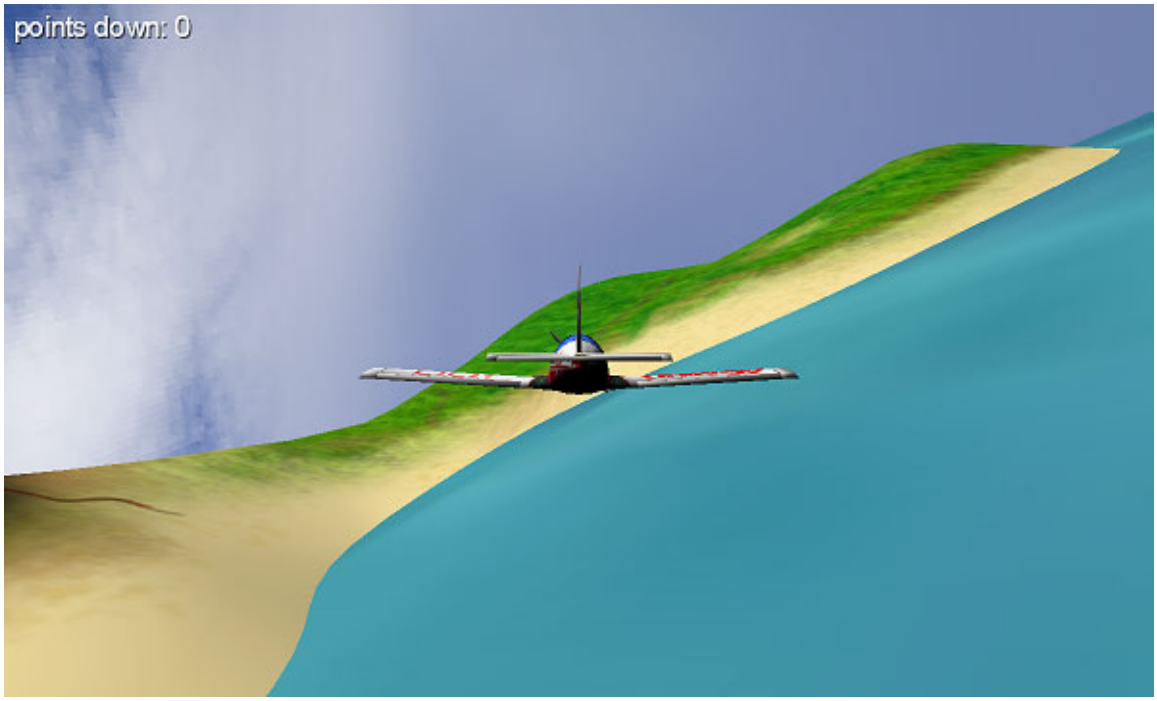
5.2. Kromě zmíněného ostrova bylo v prostoru rozmístěno i několik kruhů, na kterých bylo možné testovat ovladatelnost letadla zvolenou metodou.

Obrázek 5.1: Jedna z prvních verzí prototypu vyvíjeného bez použití pomocných knihoven. Primárním cílem bylo testování ovládání. Vizuální stránka proto byla zpočátku velmi slabá.

Ovládacích metod obsahoval prototyp hned několik. V zájmu rychlé implementace se k jejich výběru používá nativní menu platformy. V základu je zvolená již popsaná nová metoda ovládání a dále je možno přepnout na ovládání pomocí akcelerometru nebo virtuálního joysticku na obrazovce. Takto mohlo dojít k přímému porovnání ovládacích metod. Současně vznikaly i varianty jednotlivých metod s různou citlivostí nebo lehce odlišným chováním. To se týká především navrhované metody, kde existuje více přístupů, jak jsem popsal v kapitole 4.1. Ukázalo se, že je vhodnější použít variantu, kdy mají ovládací oblasti pevně danou velikost v závislosti na rozlišení obrazovky, přičemž případný přesah mimo zobrazovací oblast se ignoruje. Zjednodušeně lze říct, že tato metoda udržuje konstantní citlivost na úkor možnosti dosáhnout vždy plného rozsahu ovladatelnosti. Ideální místo prvotního dotyku tedy bude přesně v polovině výšky obrazovky. Tuto informaci bude potřeba vhodným způsobem předat uživateli a více tento problém popíši v kapitole 5.4.

Druhá popisovaná varianta měnila rozměry obládacích oblastí (a tím i citlivost) v závislosti na poloze prvotního dotyku tak, aby nikdy nepřesáhly okraje obrazovky. Tato verze se ukázala jako méně vhodná. Původní předpoklad, že hráč bude mít v průběhu celého letu trvale položené prsty na obrazovce, se ukázal jako nesprávný. Hráči mají naopak tendence prsty zvedat a opětovně se obrazovky dotýkat, což mění refrenční hladinu pro ovládání. V případě této varianty se navíc měnila i citlivost a hráč ztrácel přehled o tom, jakou odezvu pohyb jeho prstů vyvolá. Ovládání se tím stávalo složité a nepřirozené, proto jsem tuto variantu zavrhl.

Podobně jsem experimentoval i s ovládáním směrového kormidla, jakožto třetí ovládané osy, jak bylo opět popsáno v kapitole 4.1. Tato varianta, byť návrhově čistá a konzistentní s ostatními osami, se také ukázala jako nevhodná. Problémem zde nebyla nepřirozenost,



Obrázek 5.2: Závěrečná podoba prototypu obsahovala několik variant ovládání. I v grafickém zpracování nastal posun, neboť testující hráče příliš zjednodušené prostředí rušilo od hlavního cíle, tedy testování ovládání.

ale především komplexnost ovládání. Opět připomínám, že hra necílí na profesní piloty či letecké modeláře, ale na běžné hráče. Ti bohužel v obecném měřítku nejsou schopní ovládat letadlo ve více než dvou osách. Na druhou stranu se ukázalo, že někteří jednotlivci toto ovládání považují za vhodné, a dokonce by jej preferovali před ovládáním dvouosým. Lze o něm proto uvažovat jako o možném volitelném rozšíření.

Zmíněné testování probíhalo v průběhu vývoje prototypu a jednotlivých metod na vesměs náhodném vzorku testerů. Nebylo však nijak organizované a neexistují z něj žádné záznamy, právě kvůli plynulému vývoji nezřetelným iteracím. Ne vždy tak bylo možné nechat stejnou osobu srovnat varianty před a po úpravě. V této fázi by byl jistě prostor pro zlepšení. Detailnější testování jsem provedl později před zveřejněním finální hry a blíže jej popíši v kapitole 5.6.1. Pro úplnost doplním, že vstup pomocí akcelerometru nebo virtuálního joysticku byl v drtivé většině případů označen za méně pohodlný a vzhledem k povaze hry také méně vhodný.

5.2 Struktura aplikace

Tato kapitola popíše strukturu aplikace, která má v případě použití knihovny Libgdx jistá specifika. Budou také představeny některé diagramy tříd, zachycující nejdůležitější vazby mezi herními objekty. Úplny diagram tříd tato práce neobsahuje, neboť by od čtenáře vyžadoval hlubokou znalost použité knihovny a pro běžného čtenáře by nebyl příliš popisný.

5.2.1 Oddělení platforem

Jak již bylo naznačeno v popisu knihovny Libgdx v kapitole 2.5, jednou z jejích výhod je přenositelnost mezi platformami. Pro vytvářenou hru budou relevantní platformy Android a PC (bez rozdílu operačního systému). V době psaní této práce existuje v knihovně Libgdx také podpora platformy iOS, která se nachází na mobilních zařízeních firmy Apple. Tato podpora je však ve velmi ranném stádiu a proces sestavení aplikace vyžaduje proprietální software, vývojářskou licenci a fyzické zařízení této platformy. Kvůli těmto okolnostem jsem podporu iOS vyloučil. Z povahy používané knihovny by ovšem bylo přidání podpory této platofrmy do již hotové hry triviální záležitostí.

Pro další popis budu využívat pojem *projekt* tak, jak je chápán z hlediska vývojového prostředí, zde programu Eclipse. Jeden projekt lze sestavit do jedné spustitelné aplikace či knihovny. Doporučeným postupem je zde vyčlenění veškeré aplikační logiky do jednoho projektu, ze kterého sestavením vznikne knihovna. Tuto knihovnu využívají ostatní projekty, vždy jeden pro každou cílovou platofrmu. Existence projektu pro platformu Android je vyžadována. Z historických a praktických důvodů jsou právě zde uloženy veškeré datové soubory budoucí aplikace a varianty pro ostatní platformy se na tento obsah pouze odkazují (ke kopírování dat dochází až při sestavování konkrétních aplikací). Projekty pro jednotlivé platformy ale typicky plní úlohu triviálního zavaděče, který pouze připraví prostředí pro běh a následně předá řízení hlavní aplikační logice. Podoba těchto zavaděčů je pevně daná a v průběhu vývoje aplikace se již prakticky nemění.

Ukázalo se jako velmi výhodné vedle platformy Android použít i platformu PC a provádět maximální možné množství vývoje právě zde. Narozdíl od mobilní aplikace, kde probíhá typický proces sestavení a následné instalace do zařízení, zde dochází k sestavení a spuštění přímo v počítači vývojáře. To probíhá řádově v jednotkách sekund, tedy zásadně rychleji než na mobilním zařízení. Díky knihovně Libgdx jsou přitom rozdíly mezi platformami z pohledu vyvíjené hry minimální a lze takto vyvíjet celé uživatelské rozhraní i veškeré vykreslování 3D scény. Problém nastává při implementace ovládání, které pro svoji funkčnost vyžaduje současný dotyk dvou prstů. Tuto část bylo tedy nutné vyvíjet přímo na skutečném mobilním zařízení. Díky důmyslnému systému práce se vstupem (který je popsán v kapitole 2.5.2) lze nahradit dotykový vstup vstupem z klávesnice. Takové řešení sice není ideální (řízení letadla je skokové), ale pro vývoj částí se vstupem přímo nesouvisejících je více než postačující.

Dalším důvodem k vytvoření projektu pro platformu PC může být také tvorba propagačních materiálů, typicky videí. V případě mobilních zařízení je snímání obrazovky a současné ukládání ve formě videa velmi omezené nebo úplně nemožné. Je to dáno jak výkonovými možnostmi, tak i potřebou speciálního oprávnění takové snímací aplikace. U běžného počítače by v současnosti požadavky na výkon neměly představovat zásadní problém a stejně tak je snadno dostupný i software pro záznam. Opět díky sjednocenému chování napříč platformami je možné vytvořit video přímo ze hry, aniž by neznalý divák rozeznal rozdíl. Stejně jako při vývoji zde může nastat problém se vstupem. Je však možné při hraní na mobilním zařízení veškerý vstup zaznamenávat a následně jej v počítači přehrávat do běžící aplikace. Tímto způsobem lze velmi efektivně vizualizovat i místa dotyku a vytvořit například názornou ukázku, jak může hráč svými dotyky letadlo ovládat.

5.2.2 Objektová struktura aplikace

Nyní se v popisu přesunu od prototypu k finální aplikaci vystavené nad knihovnou Libgdx. V dřívějších kapitolách již bylo zmíněno, že knihovna umožňuje rozčlenění aplikace do tzv. *obrazovek*. Vyvíjená hra této možnosti využívá, jak je vidět z obrázku 5.3. Do diagramu jsou zaneseny jen základní třídy aplikace a jejich vazba na třídy z knihovny. Každá z obrazovek má pak vazby další, které v zájmu přehlednosti v diagramu znázorněny nejsou.

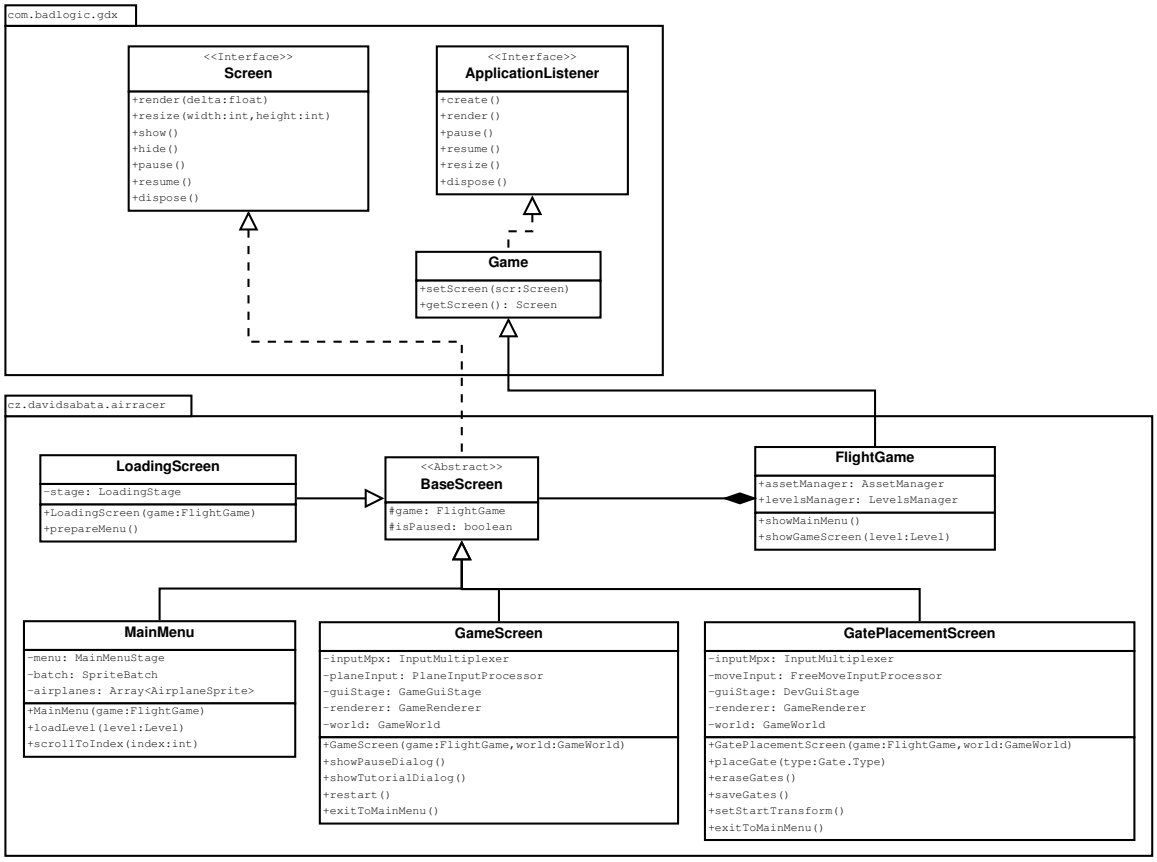
Základní třídou aplikace je `FlightGame`, která po spuštění vytváří jednotlivé obrazovky a zajišťuje mechanismy pro jejich přepínání. Současně vytváří a zpřístupňuje objekty `AssetManager` a `LevelsManager` pro načítání obecných datových souborů a definičních souborů pro jednotlivé závody.

První obrazovka která se po spuštění aplikace zobrazí je `LoadingScreen`. Ta obsahuje triviální animaci informující uživatele o načítání hry. Tato obrazovka musí být velmi jednoduchá, aby v době mezi spuštěním aplikace a jejím zobrazením nenabýval uživatel dojmu, že aplikace přestala pracovat. Ihned po zobrazení načítací obrazovky se volá `AssetManager`, který načte podklady pro ostatní obrazovky, včetně samotné hry. Tato funkcionalita je již součástí Libgdx. V závěru načítání je volán `LevelsManager`, který ze souborů ve formátu JSON načte definice jednotlivých závodů, aby na jejich základě mohla obrazovka hlavního menu vytvořit navigační rozhraní. Přepnutí na obrazovku samotného závodu pak v rámci jednoho spuštění aplikace probíhá vždy téměř okamžitě. Proběhne totiž pouze přesunutí již načtených herních objektů na správné pozice a žádným dalším náročným operacím nedochází.

Datová reprezentace světa je znázorněná na obrázku 5.4. Diagram opět některé souvislosti v zájmu přehledosti zjednodušuje. Klíčovou třídou je zde `GameWorld`, reprezentující právě herní svět. Z hierarchie dědičnosti lze vyčíst, že třída rozšiřuje třídy `BulletWorld` a `BaseWorld`. Druhá zmíněná, tedy `BaseWorld`, představuje základní svět obsahující vykreslitelné entity. (Samotné vykreslování však provádí třída `GameRenderer`, spravovaná třídou herní obrazovky.) Každá entita vzniká ze svého *konstruktoru* (ve smyslu třídy `Constructor`). Konstruktor definuje 3D model entity a v případě, že má být entita součástí kontroly kolizí, definuje i její kolizní tvar. Entita je pak instancí svého konstruktoru na určitém místě ve scéně (přesněji řečeno s určitou transformací). Tento princip umožňuje sdílet zdrojová data všech entit stejného typu a snižovat tak paměťové i výkonové nároky.

Základní svět `BaseWorld` tedy podporuje pouze vykreslení entit. Jeho potomek `BulletWorld` rozšiřuje toto chování o možnost provádění detekce kolizí. K tomu rozšiřuje i konstruktor na `BulletConstructor`, obsahující právě informaci o kolizním tvaru a povaze tělesa jak byla popsána v kapitole 2.6. Stejně tak rozšiřuje entitu na `BulletEntity`, se kterou může knihovna detekci provádět. Poslední rozšíření světa je právě na nejdůležitější třídu `GameWorld`, která nad běžící simulací spouští samotnou hru ve smyslu dodání sémantiky kolizím objektů. Z tohoto důvodu zavádí třídu `GateEntity`, která reprezentuje entitu, s níž je kolize žádoucí, tedy branku nebo kruh k prolétnutí.

Třídy spadající pod knihovnu Bullet Physics obsahují ve svém názvu prefix `bt`. Pro použití této knihovny není nutná detailní znalost její vnitřní struktury, proto i já popíši tyto třídy jen velmi zběžně. Základní třídou je `btCollisionShape`, tedy kolizní tvar objektu. Podobně jako vytvořením instance konstruktoru vzniká entita, tak vytvořením instance kolizního *tvaru* vzniká kolizní *objekt*, který je součástí entity a reprezentuje jej třída `btCollisionObject`. Z hlediska výkonnosti detekce kolizí je důležité, že opět objekty vzniklé ze stejného tvaru sdílejí stejné datové struktury. Dále fyzikální svět obsahuje

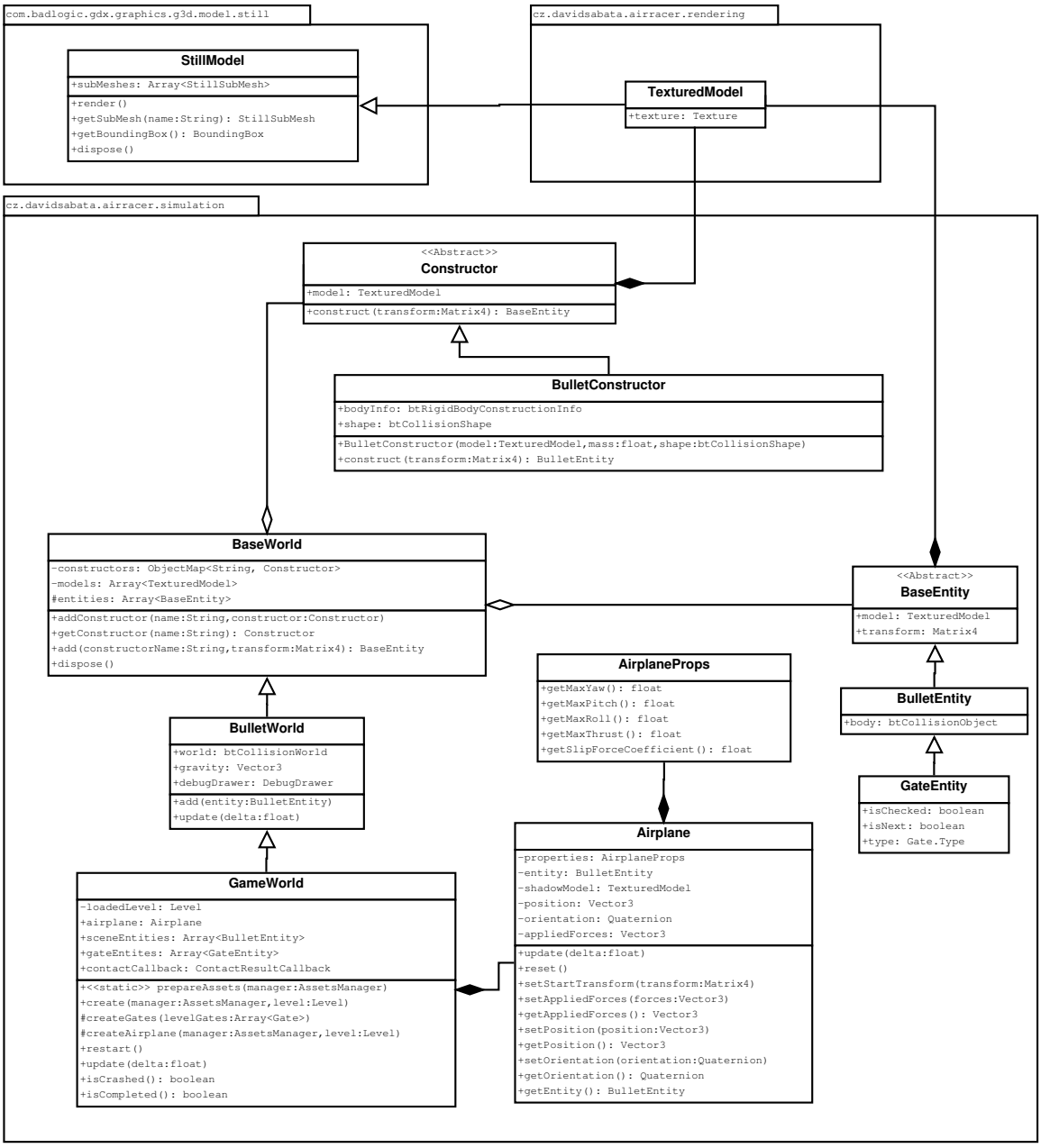


Obrázek 5.3: Diagram klíčových tříd aplikace. Vstupním bodem je třída `FlightGame`, která vytváří a udržuje jednotlivé obrazovky, potomky třídy `BaseScreen`.

instanci třídy `btCollisionWorld`, což je knihovní reprezentace prostředí, ve kterém detekce probíhá. (Z důvodů kompatibility rozhraní je do třídy `BulletWorld` pouze vložen a třída z něj přímo nedědí.) Toto prostředí představuje zjednodušenou verzi fyzikálního světa, kde je možné pouze testovat zda dochází ke kolizi některých objektů. Fyzikální simulace ve smyslu například volného pádu těles nebo jejich vzájemná interakce zde neprobíhá. Knihovna Bullet Physics toto samozřejmě nabízí, ovšem za cenu značného nárůstu požadavků na výkon. Vyvíjená hra je navíc záměrně navržena tak, aby plnohodnotná simulace probíhat nemusela.

5.3 Herní logika a průběh simulace

Řídicí funkci z hlediska herní logiky má herní obrazovka, tedy instance třídy `GameScreen`. Ve chvíli svého vytvoření dostává v konstruktoru připravený herní svět (tedy `GameWorld`) a v něm načtený závod. Obrazovka vytvoří objekt letadla (součástí definice světa je pouze jeho počáteční transformace), nastaví zpracování vstupů tak, aby se dotykové události dostaly na vstup letadla, nechá probíhat simulaci a současně scénu vykresluje pomocí třídy `GameRenderer`. Samotný pohyb letadla ve scéně probíhá velmi jednoduše, vyčíslováním vztahu pro rovnoměrný přímočarý pohyb, kde jako časový parametr slouží doba uplynulá mezi dvěma vykreslovanými snímky. Díky tomu se bude letadlo pohybovat stále stejně



Obrázek 5.4: Diagram tříd reprezentujících herní svět. Hlavní třída `GameWorld` je po vytvoření předána obrazovce, která řídí průběh hry. Herní svět obsahuje entity reagující na kolize a také speciální objekt letadla `Airplane`, který implementuje autonomní chování v závislosti na dotykovém vstupu.

rychle bez ohledu na to, jaký je aktuální počet snímků za sekundu.

Vykreslovací třída je velmi jednoduchá, a proto se jí nebudu hlouběji zabývat. Z externích souborů načítá shadery ve formátu GLSL, v každém snímku nastavuje jejich parametry a jejich pomocí vykresluje jednotlivé entity daného světa. K tomu by jí postačovaly instance `BaseWorld` a `BaseEntity`. Takový objekt je však ve hře pouze jediný, a sice tzv. *skybox*, tedy krychle obalující celou scénu, na jejíž vnitřních stěnách je textura nebe. Všechny ostatní

objekty jsou zapojeny do fyzikální simulace, kvůli čemuž musejí být v hierarchii dědičnosti níže.

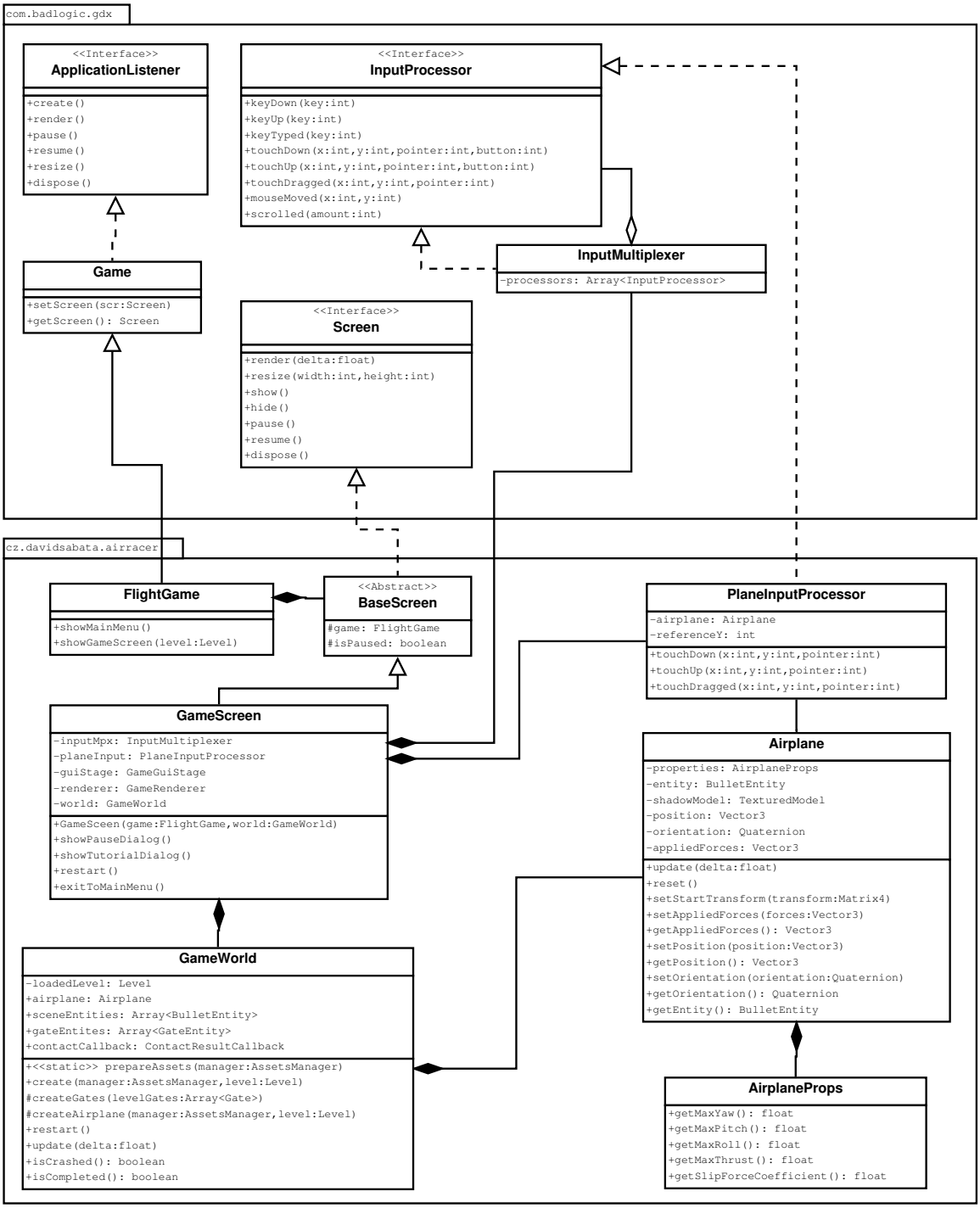
Z hlediska vstupů je situace taková, že herní obrazovka nastaví jako jeden ze vstupních procesorů právě `PlaneInputProcessor`, který je svázán s objektem letadla. Každý vstup zpracovaný tímto procesorem se převede na intenzitu sil způsobujících rotace v jednotlivých osách. (Rotace kolem osy Y, anglicky pojmenovávaná *yaw*, je předávána jen pro úplnost; vstup ji však přímo ovlivňovat nemůže.) V okamžiku příchodu události (který probíhá asynchronně) se tedy do objektu letadla dostává informace o tom, v jaké poloze jsou které jeho ovládací prvky, neboť právě taková je sémantika působících sil. Při aktualizaci scény v každém snímku pak letadlo tyto síly zohledňuje při svém dalším pohybu, upravením jeho směru a případné rotace samotného letadla. Je tedy možné říct, že letadlo ve scéně pohybuje autonomně.

Herní obrazovka před vykreslením každého snímku vyvolá aktualizaci scény, která se skládá právě z aktualizace pozice letadla a dále z provedení detekce kolizí. Výsledkem této aktualizace může být informace o tom, že nastala kolize. V takovou chvíli herní svět rozliší, zda jde o kolizi žádoucí (s brankou či kruhem) nebo nežádoucí (s ostatními objekty ve scéně) a nastaví příslušné příznaky. Obrazovka pak pouze kontroluje, zda se po provedení aktualizace změnil stav světa, což znamená že letadlo buď prolétlo poslední brankou anebo se vybouralo. Na toto obrazovka reaguje zobrazením příslušných dialogů.

Je zde také vhodné zmínit závislost výkonu aplikace na použitých kolizních tvarech, kterou jsem při vývoji vypozoroval. Dostatečně nízké výkonové nároky tak, aby hra běžela plynule na cílených zařízeních je nezbytností. Výkonovou náročnost hry lze snižovat optimalizací vykreslování scény anebo optimalizací probíhající fyzikální simulace. Narozdíl od způsobu vykreslování má úprava fyzikální simulace přímý vliv na hratelnost výsledné hry. Ve hře která používá fyzikální knihovnu pouze pro detekci kolizí není mnoho prostoru pro optimalizace. Prakticky jedinou proměnnou jsou zde kolizní tvary jednotlivých herních objektů, a právě jejich volba má dopad jak na výkon, tak i na hratelnost. Ve chvíli, kdy jsou kolizní tvary maximálně zjednodušené (koule, krychle, plochy) jsou sice požadavky na výkon nízké, ale kolize může být detekována (nebo naopak nedetekována) velmi nepřesně. Dokonce natolik, že si problému všimne i sám hráč a představuje pro něj chybu. Příkladem může být aproximace kolizního tvaru kruhu pouhým čtvercem. Zde tvar nepokryje některé krajní oblasti kruhu, které však hráč může cíleně využívat pro dokončení závodu v lepším čase. Knihovna Bullet Physics podporuje kromě základních geometrických tvarů také tvary složené. Právě jejich pomocí jsem implementoval kolizní tvar kruhu, složený ze dvou čtverců, které jsou vzájemně pootočené. Optimálním řešením by jistě bylo použít čtverců mnohem více, případně pak přímo obecný polygon, v takovém případě pak ale začínají nároky na výkon značně narůstat. Již při použití čtyř čtverců bylo možné pozorovat zpomalení hry (testováno na přístroji Samsung Galaxy Nexus, který se v době psaní této práce řadí mezi zařízení vyšší třídy). Podobně se ukázalo vhodnější použít pro kolizní tvar letadla jednoduchý kvádr, namísto dvou kvádrů představujících trup a křídla. Oba zmiňované kolizní tvary jsou znázorněny na obrázku

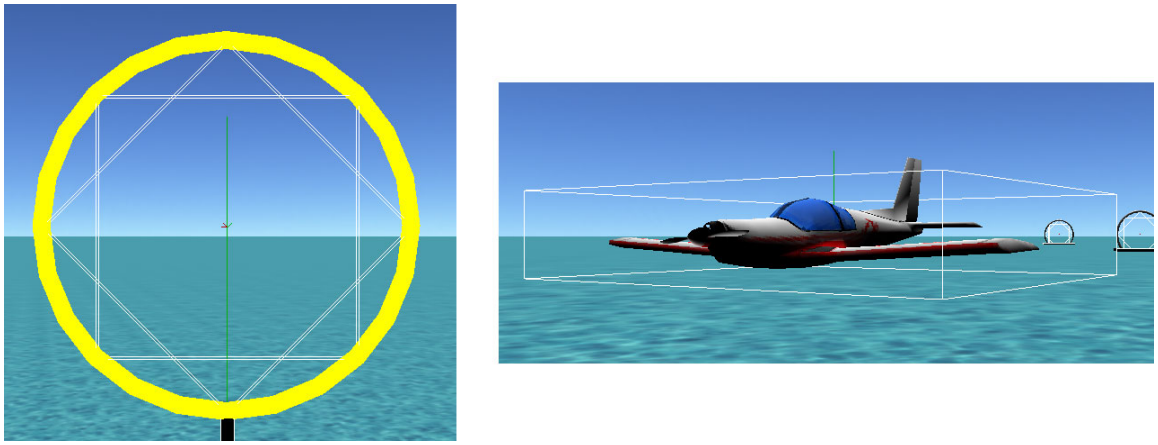
5.4 Uživatelské rozhraní

Z hlediska uživatelského rozhraní lze hru rozdělit na dvě části – hlavní menu a samotnou hru. Ty jsou implementovány v již popisovaných třídách `MainMenu` a `GameScreen`. Vazby součástí uživatelského rozhraní jsou znázorněny na obrázku 5.7. Každá z obrazovek obsahuje jednu



Obrázek 5.5: Diagram tříd z hlediska vstupních událostí. `PlaneInputProcessor` události zpracovává a vytváří hodnoty intenzit, které se ve třídě `Airplane` aplikují na ovládací prvky letadla.

instanci potomka třídy `BaseStage` (konkrétně `MainMenuStage` a `GameGuiStage`), kde jsou vytvářeny jednotlivé prvky rozhraní a definováno jejich rozložení. Využity jsou převážně předdefinované komponenty, jejichž vizuální podoba je upravená tak, jak bylo popsáno

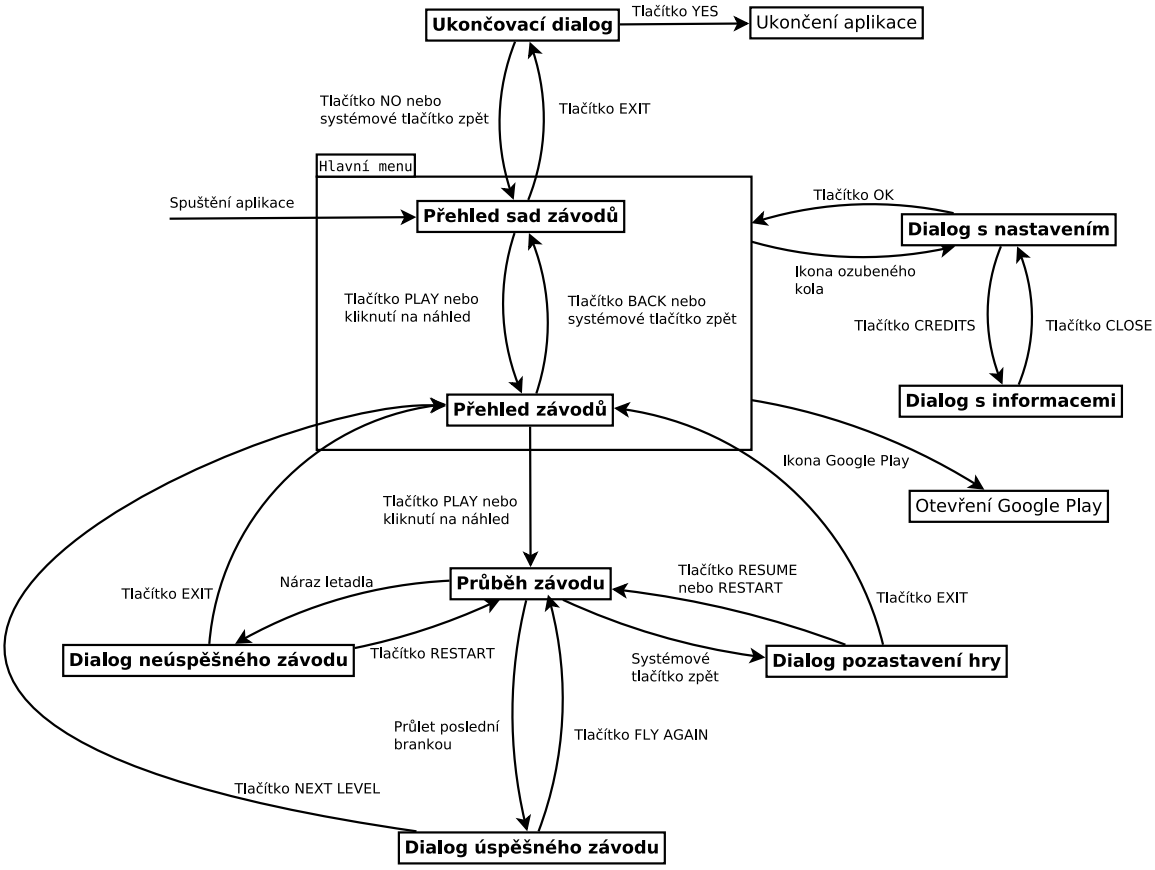


Obrázek 5.6: Kolizní tvary některých herních objektů. S rostoucí složitostí strmě rostou i nároky na výkon. Proto je kruh reprezentován pouze dvěma pootočenými čtverci.

v kapitole 2.5.3. Kromě těchto dvou obrazovek hra využívá několik dialogů. Ty jsou ve své výchozí podobě ztvárněny použitím plovoucího okna. Tuto podobu jsem upravil tak, že dialogy okno nepoužívají (lépe řečeno je vykresleno průhlednou barvou) a namísto toho částečně ztmaví celou oblast obrazovky. Toto řešení působí méně technicistním dojmem a lépe zapadá do vizuální podoby celé aplikace.

Navigace v menu je rozdělená do dvou úrovní. Na první úrovni lze vybírat mezi sadami závodů, na úrovni druhé pak mezi jednotlivými závody. Vizuálně výběr realizuje horizontálně posuvná oblast s náhledy jednotlivých sad či závodů. Položka nejblíže středu obrazovky se považuje za aktuálně vybranou a kliknutím na její miniaturu nebo na tlačítko *PLAY* se sada či závod vybere a spustí. Pro zřetelost vybrané položky se ostatní položky s rostoucí vzdáleností od středu obrazovky zmenšují. Při posouvání se pak panel vždy zafixuje tak, aby byla aktivní položka právě ve středu obrazovky. Pro dosažení tohoto chování jsem rozšířil knihovní třídu `ScrollPane`, která podporuje pouze plynulý horizontální či vertikální posun obsahu v panelu bez dalšího možného nastavení nebo funkcionality. Výsledné chování implementuje třída `MyScrollPane`. Z panelu je také odstraněn klasický posuvník, který nahrazuje skupina ikon, jejichž počet je shodný s počtem položek v panelu. Ikona reprezentující aktuálně vybranou položku je pak vizuálně odlišená. Situaci ilustruje dříve uvedený obrázek 2.3, který současně zachycuje rozdělení obrazovky hlavního menu do jednotlivých komponent.

V kapitole 5.1 jsem zvolil jako optimální variantu ovládání letadla takovou, kdy nevhodné místo prvotního dotyku může omezit manévrovací schopnosti letadla. Z tohoto důvodu a současně proto, že je metoda ovládání zcela nová a neznámá, jsem vytvořil speciální dialog pro zaučení nového hráče. Zobrazí se při spuštění prvního závodu ve hře a namísto potvrzovacích tlačítek jsou vykresleny dva žluté kruhy (viz obrázek 5.8). Hráč je instruován, aby se současně dotkl obou kruhů, teprve poté se závod zahájí. Kruhy jsou záměrně zarovnány svým středem na střed výšky obrazovky s cílem naučit hráče tuto pozici prstů jako výchozí. Tento postup se osvědčil. Při testování se ukázalo, že v drtivé většině případů hráčům stačí pro pochopení ovládání pouze informace zobrazená v tomto dialogu. Někteří navrhovali, že by se dialog mohl zobrazit před každým závodem. Jiní by prefero-

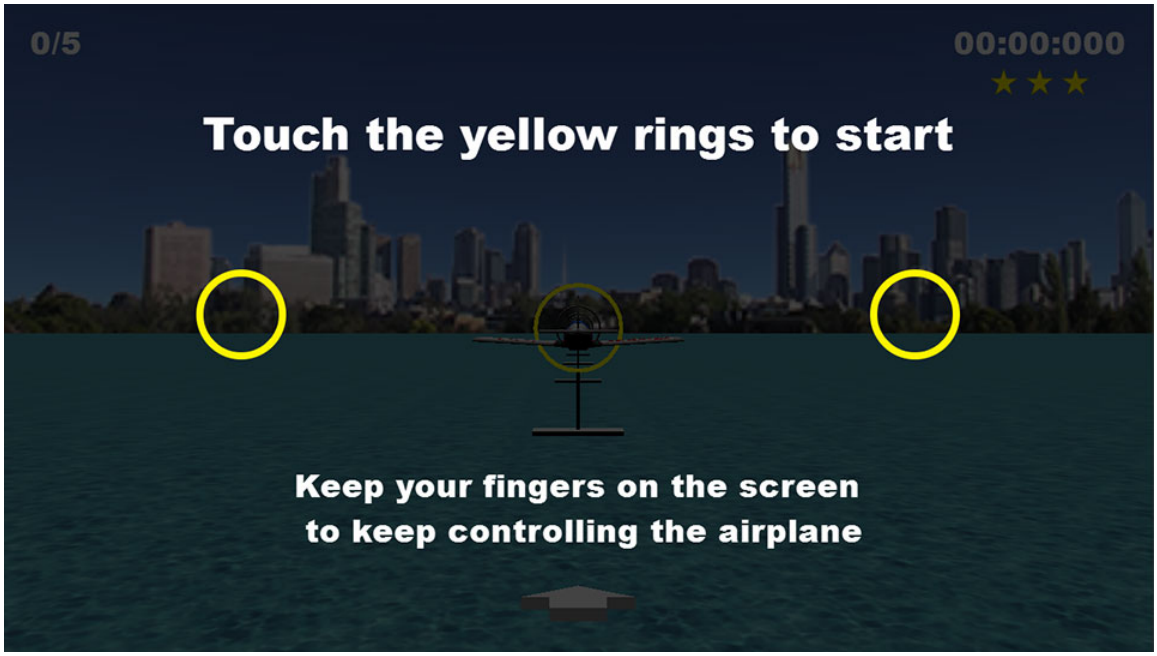


Obrázek 5.7: Jednotlivé obrazovky a dialogy v aplikaci. Pro navigaci mezi nimi slouží prvky uživatelského rozhraní nebo volitelně systémová tlačítka zařízení.

vali, kdyby se místa optimálního dotyku zobrazovala v průběhu celého závodu ve chvílích, kdy se hráč obrazovky přestane dotýkat. Především druhý zmíněný námět by mohl pomoci hráčům rychleji se v ovládání zorientovat například v případech, že se ke hře vrátili po delší době. Ani jeden z těchto námětů však v aktuální verzi zahrnut není.

5.5 Vývojářské rozhraní

Součástí hry je také speciální vývojářský režim. Ten je pro běžné hráče přirozeně nedostupný a v publikované verzi je odstraněn. Slouží k jednoduchým úpravám existujících závodů a k vytváření závodů nových. Jeho rozhraní je spíše strohé, avšak účelné. Zachyceno je na obrázku 5.9. Vývojářský režim implementuje obrazovka `GatePlacementScreen`, uživatelské rozhraní pak třída `OverlayStage`. Obrazovka načte objekty herního světa, podobně jako při spuštění závodu, s tím rozdílem, že neprobíhá detekce kolizí ani vyhodnocování stavu závodu. Scéna neobsahuje ani letadlo a kamera se může volně pohybovat v prostoru. K tomu slouží tlačítka uživatelského rozhraní pro posun o pevně danou vzdálenost, případně pak speciální vstupní procesor `FreeMoveInputProcessor` umožňující plynulé rozhlížení a průlet scénou. Je možné přidávat a odebírat jednotlivé kruhy a branky závodu a také nastavit počáteční transformaci letadla.



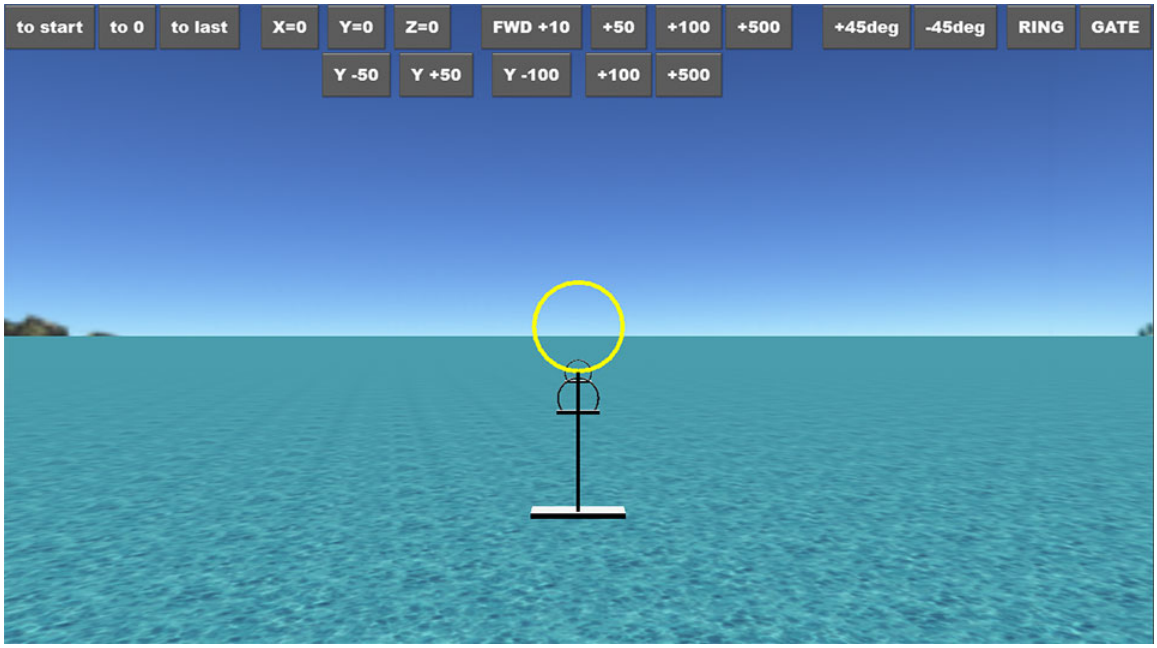
Obrázek 5.8: Dialog pro zaučení nového hráče, zobrazovaný u prvního závodu ve hře. Pro spuštění závodu se musí hráč dotknout současně obou žlutých kruhů. Tím by měl pochopit, že k ovládání letadla je potřeba vždy dvou prstů.

Z důvodu zachování přenositelnosti mezi platformami bohužel nelze závod přímo uložit a ihned jej hrát. Aktuálně se závod vypíše do konzole vývojového prostředí jako řetězec ve formátu JSON. Ten je nutné ručně uložit do souboru v adresáři s ostatními datovými soubory aplikace. Toto chování vzniká proto, že v okamžiku sestavování aplikace jsou veškeré její soubory zabaleny do jediného balíčku, který se dále zpřístupňován pouze pro čtení. Alternativně by bylo možné ukládat vytvářené závody do uživatelské paměti zařízení, ovšem tyto by pak byly volně přístupné a upravitelné, a tudíž by nebylo možné zaručit jejich konzistenci. Časy jednotlivých závodů jsou vkládány také ručně, na základě několika zkušebních letů. V budoucnu budou časy upravovány na základě dat získaných anonymním sběrem statistik.

5.6 Zveřejnění hry a zpětná vazba

Ve chvíli, kdy se hra dostala do finální podoby, bylo možné ji začít veřejně distribuovat. Současně jsem pro dosud bezejmennou hru zvolil název Air Racer. Do té doby jsem testování prováděl osobními schůzkami v různých stádiích vývoje, kde bylo možné okamžitě reagovat na chyby a jiné nedostatky. Tento přístup současně nabízel mnohem lepší zpětnou vazbu od hráčů, kterou by bylo později obtížné získat. Dalším přínosem bylo také přímé testování hry na různých zařízeních. Zde žádné problémy nenastaly a hra bez problémů běžela na všech zařízeních, se kterými deklarovala kompatibilitu. Hodnotné reakce přinesly také pravidelné konzultace ve firmě CUKETA, s. r. o. , která se herním vývojem profesionálně zabývá.

Posledním krokem před zveřejněním bylo zapojení statistik. Tuto službu poskytuje přímo firma Google formou speciální knihovny. Původně byla navržená pro sledování klasic-



Obrázek 5.9: Rozhraní pro tvorbu a úpravu závodů. Slouží pouze pro vývoj a hráči do něj přístup nemají. Ve scéně se lze pohybovat skokově (tlačítky) nebo volně pomocí dotykového vstupu.

kých aplikací, které jsou založeny na výchozích komponentách platformy, jako jsou Aktivity, dialogy a další. Díky poslední verzi jejího rozhraní je však možné sledované události spouštět manuálně, a tím ukládat statistiky i pro hru, která má z pohledu sledující knihovny jen jedinou obrazovku.

5.6.1 Veřejné testování a průzkum

První veřejnou verzi hry jsem vystavil na internet ve formě samostatného binárního souboru (tedy bez možnosti automatické aktualizace) a nasdílel na některých sociálních sítích. Současně jsem vytvořil dotazník, o jehož vyplnění jsem hráče požádal. Plné znění dotazníku se nachází v příloze A této práce. Struktura dotazníku je inspirována metodou *Likert Scale* [14]. Jejím cílem je nepokládat přímé otázky, ale představovat tvrzení a měřit, jak silně se s nimi hodnotící ztotožňuje. Tím by mělo být hodnocení jednodušší a spontánnější. Současně se omezuje nežádoucí kreativita hodnotícího a zjednodušuje se i vyhodnocení takového dotazníku. Nejčastěji se používá stupnice o 3, 5 nebo 7 hodnotách. Důležitý je vždy lichý počet možností, umožňující hodnotícímu zaujmout neutrální postoj. Mohou nastat výjimky, kdy je žádoucí použít sudý počet možností, a tím si odpověď vynutit. Tento přístup ale pro vyvíjenou hru vhodný není. Další z doporučení při tvorbě dotazníku metodou Lickert Scale je nezaujatost tvrzení. Ta by měla být ideálně předkládána tak, aby nebylo zřejmé, která z možností je správná. Hodnotící hráč pak mnohem méně podléhá společenským a jiným tlakům.

Dotazník vyplnilo 67 hráčů a z jeho výsledků dotazníku vyplývá, že první dojem z aplikace byl pozitivní, hlavní menu hráčům připadá přehledné a je dobře ovladatelné. To po-

tvrzuje, že implementace vlastního chování posuvného panelu v menu byla úspěšná. Stejně tak se osvědčila ikona ozubeného kola reprezentující dialog s nastavením. Zajímavostí je, že z hráčů odpovídajících na dotazník tento dialog otevřela asi jen čtvrtina z nich (dle údajů ze statistik). Přesto většina odpověděla, že by uměla zapnout či vypnout zvuky ve hře. Příčinou může být buď nedbalost při vyplňování dotazníku anebo fakt, že jde o silně zažitý prvek uživatelského rozhraní ve hrách, od kterého hráči očekávají chování, aniž by jej ověřovali.

Prokázala se také vhodnost logické stavby menu. Hodnotící ve hře rozpoznali, že má menu více úrovní a že pro zpřístupnění uzamčeného závodu bude pravděpodobně nutné splnit závod předchozí. V případě dotazu na počet položek závodů ve hře však byly odpovědi sporné. To mohlo být zapříčiněno nevhodně formulovaným tvrzením (*Mám představu o tom, kolik je ve hře levelů*), které vyžaduje zpřístupnění všech sad závodů. Sady jsou na sebe navázány stejně jako jednotlivé závody a do odemčení do nich nelze ani nahlédnout. Hráč který by chtěl na toto tvrzení korektně odpovědět by tedy musel nejdříve všechny závody zpřístupnit, což však podle statistik neudělal nikdo. Dalším sporným tvrzením byla možnost spuštění závodu. Zde šlo čistě o experiment – závod lze spustit jak tlačítkem PLAY, tak i kliknutím na jeho náhled. Téměř třetina odpovídajících využila kliknutí na náhled. Pravděpodobně se tak chovali proto, že jde opět o chování které lze od posuvného panelu obecně očekávat.

Další část dotazníku se věnovala samotnému průběhu závodu a chování letadla. Téměř 80% hráčů odpovědělo, že jsou schopní letadlo bez předchozího vysvětlení ovládat (myšleno jiného vysvětlení než dříve zmiňovaného dialogu před prvním závodem). To potvrzuje intuitivnost navrhovaného ovládání. Většina hráčů považuje ovládání za dostatečně přesné, přičemž lze s letadlem stále provádět ostré zatáčky. Stejně tak hráči odpovídají, že vědí, jak pohyby jejich prstů ovlivní letadlo. Tím opět potvrzují dobrý návrh ovládání. Současně však hráči nebyli schopni jasně identifikovat, zda chování letadla odpovídá letu skutečného letadla. S tím souvisí i tvrzení ohledně předvídatelnosti chování letadla, kde jsou odpovědi také vesměs neprůkazné. Již při návrhu hry (v kapitole 4.2) jsem deklaroval, že cílem hry není maximálně věrné napodobení letu skutečného letadla, nýbrž navržení chování tak, aby vyústilo v co nejvíce zábavnou a obecným hráčům pochopitelnou hru. Z tohoto pohledu nejsou zmíněné reakce ani pozitivní ani negativní. Bylo by možné říct, že se chování pohybuje na hraně s lehkým přesahem do nerealističnа, což aktuálně považuji za vhodný kompromis. Posledním tvrzením na téma ovladatelnosti byl dotaz na ovládání směrového kormidla. Zde se hráči rozdělili na téměř přesné poloviny, kdy jedna strana preferuje ovládání přímé a druhá inverzní. Tento jev se projevil ještě před veřejným testování a hra již v době vyplňování dotazníku obsahovala možnost převrácení svislé ovládací osy. Tato volba je u leteckých her velmi často používaná a měla by uspokojit obě skupiny hráčů. Výchozí chování je nastaveno na inverzní, kde posun prstů ke spodní hraně obrazovky zvedá letadlo nahoru. Toto chování totiž více odpovídá chování kniplu ve skutečném letadle (případně hernímu joysticku).

Poslední část dotazníku se věnovala skladbě závodů a prostředí ve kterém se odehrávají. Letadlo podle většiny hráčů letí spíše pomalu, stejně jako obtížnost závodů hodnotí spíše jako nízkou. Je však vhodné dodat, že statistiky v období vyplňování dotazníku ukazovaly, že do pokročilejších závodů postoupilo jen malé množství hráčů. Toto tvrzení tak hodnotí spíše subjektivní obtížnost, která však může být zajímavým doplňkem objektivní obtížnosti, kterou lze získat ze statistik na základě četnosti úspěšných dokončení jednotlivých závodů. Drtivé většině odpovídajících hráčů se pak podařilo dokončit alespoň jeden závod a alespoň

jednou dokončit závod se ziskem tří hvězdiček. Hráči kladně ohodnotili také navigační šipku, která ukazuje směr k následujícímu kruhu či brance. Z dodatečných odpovědí (volitelný vzkaz na konci dotazníku) vyplývá, že by ocenili i informaci o vzdálenosti následujícího cíle, případně náhled na rozložení branek v celém závodě. V závěru bylo představeno několik tvrzení ohledně scenerie závodu. Z odpovědí lze vyčíst, že hráči prostředí spíše nevnímají, současně by však uvítali kdyby hra obsahovala více dekorativních objektů. Tyto reakce lze přičíst subjektivní prázdnotě prostředí závodu. Při vývoji se ukázalo jako velmi efektivní nechat závod probíhat nad mořem v teoreticky neomezeném prostoru. Hráči ale zřejmě vnímají jistý nedostatek ukotvení scenerie. Vhodným řešením by mohlo být omezení herního prostoru tak, že by závod probíhal například podél skalního útesu nebo města. Vzhledem k tématice by pak bylo možné doplnit divácké tribuny, reklamní plochy a další objekty. To může být tématem dalšího rozšiřování hry.

V rámci udržení hráčů jsem do veřejně testované verze zabudoval omezení, které hru dovolí hrát pouze dva týdny. Poté bude po spuštění nabízet pouze možnost přechodu do obchodu Google Play, kde bude možné stáhnout finální verzi hry.

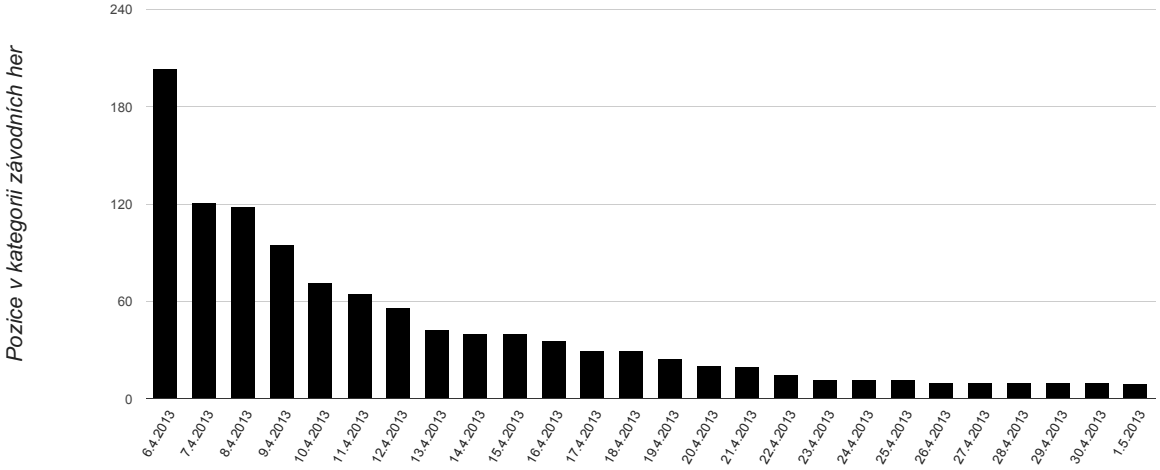
5.6.2 Zveřejnění na Google Play

Verze publikovaná na internetovém obchodě Google Play se od veřejně testované prakticky neliší. Opravil jsem několik drobných nedostatků, které se projevovaly pouze na úzkém okruhu zařízení a samozřejmě jsem také odstranil časové omezení. V souvislosti s publikováním bylo nutné vytvořit několik variant propagačních materiálů. Jelikož jsem veškeré grafické podklady vytvářel já sám, může být jejich kvalita sporná. Zpětně bych pravděpodobně zvážil spolupráci s profesionálním grafikem. Přesto se volba jednoduché ikony s vyobrazeným letadlem ukázala jako vhodná. Obecný vliv prezentačních materiálů na potenciální uživatele však nelze nijak měřit.

Jak již bylo popsáno v kapitole 2.7, pro publikování v obchodě Google Play je nutné vlastnit vývojářský účet. Vytvoření účtu, stejně jako publikování aplikace pak probíhá na počkání. Není tedy nutné počítat se zpožděním při vydávání aplikace tak, jak je tomu u jiných platforem. Publikovaná aplikace musí být podepsaná soukromým klíčem vývojáře a stejným klíčem pak musí být podepsány všechny další aktualizace. V případě ztráty tohoto klíče již nelze binární podobu aplikace nahrané v obchodě nijak upravovat.

Po vydání aplikace jsem ji opět propagoval na sociálních sítích, včetně několika komunit specializovaných na hry a platformu Android. Současně jsem žádal hráče, aby hru v obchodě ohodnotili (dle vlastní vůle), čímž ji učiní teoreticky lákavější pro další potenciální hráče. Hodnocení probíhá udělením jedné až pěti hvězdiček a přidání případného komentáře. Počet hodnotících hráčů je však velmi nízký, dle současných statistik vloží hodnocení méně než jeden hráč z 500, komentář pak vloží asi jeden z tisíce. Aktuálně hru hodnotilo 87 hráčů, přičemž 61 zvolilo nejvyšší hodnocení, 13 pak hodnocení nejnižší. Při takto nízkém počtu však nemá hodnocení ani většina komentářů téměř žádnou vypovídací hodnotu. Hodnocení hráčů jsem se snažil podpořit herním dialogem, který se jednorázově spustí po splnění všech závodů z první sady.

I přes nízkou účast hráčů v hodnocení se hra v obchodě prosadila a necelý měsíc po vydání se dostala na pozici deváté nejlepší závodní hry. Graf vývoje pozice v jednotlivých dnech je na obrázku 5.10. Celkový počet instalací přesahuje 70000, přičemž denní trend nových uživatelů v současnosti spíše stagnuje.



Obrázek 5.10: Vývoj pozice hry v Google Play, v kategorii závodních her. Po necelém měsíci od vydání je hra na 9. pozici.

Součástí vývojářského rozhraní obchodu Google Play je i sledování vzniklých výjimek. Těch se objevilo vzhledem k množství instalací jen minimum. Šlo o nepředvídatelné chyby, které vznikaly jen na některých jednotlivých modelech nebo konkrétních zařízeních. Přestože bývají zaznamenané výjimky běžně svázány s popisem zařízení na kterém vznikly, v některých případech tato informace chyběla. Lze předpokládat, že jde o upravená sestavení platformy Android. Mimo toto jsem narazil také na chybu vyskytující se v různém množství na celé modelové řadě. Jde o chybu přímo v platformě Android, která se však vyskytuje jen u konkrétního sestavení a verze. V tomto případě bylo možné přestat těmto typům zařízení hru nabízet a v obchodě ji označit jako nekompatibilní. V době před omezením podpory měla tato zařízení téměř 10% podíl na instalacích. Přestože jde o nezanedbatelný počet hráčů, jiné řešení dostupné nebylo. Chyba byla nahlášena a v dalších verzích platformy opravena, nicméně výrobce aktualizaci svých přístrojů nenabídl a jiné řešení než aktualizace platformy pro tento problém známé není.

5.7 Možnosti dalšího vývoje

Přestože jsem v dřívějších kapitolách často naznačil další možný vývoj, popíši jej nyní detailně v samostatné kapitole. Za položku s nejvyšší prioritou aktuálně považuji vizuální stránku hry. Tu by sice bylo možné zařadit v rámci her nabízených na Google Play do průměru, nicméně je to vlastnost, která produkt prodává. Úplně nejdříve potenciální zákazník vidí ikonu hry, následně propagační obrázky a teprve poté si může přečíst popis a hru případně stáhnout. Z reakcí na hru lze vyčíst, že s navrženým způsobem ovládání je většina hráčů spokojená, nicméně jde právě o tu skupinu hráčů, které neodradil žádný ze zmiňovaných kroků. Lze předpokládat, že existuje početná skupina návštěvníků internetového obchodu, které hra nezaujala právě kvůli grafickému zpracování. Konverzi těchto potenciálních hráčů na hráče skutečné by mělo pomoci právě další zlepšování propagačních materiálů. Jejich součástí je také popis hry v obchodě, který by bylo vhodné přeložit do více jazyků (aktuálně je pouze anglicky). Stejně tak by bylo možné přeložit všechny texty ve hře, jak také napovídaly některé reakce hráčů.

Součástí vizuální stránky hry je také herní prostředí. V tomto případě by úpravy cílily

převážně na již existující hráče. Ti se v dotazníku svojí většinou shodli, že by hře prospělo více dekorativních objektů. Spolu s tím, jak by v rámci rozšiřování přibývaly ve hře další závody, by mohly začít působit příliš jednotvárně. Herní objekty jako tribuny, reklamní plochy a další by mohly takovou jednotvárnost narušit a současně (ovšem ne nezbytně) zvýšit obtížnost závodů, například jejich vhodným umístěním do trasy letu. Ve větším měřítku pak může jít o úplnou změnu scenerie závodů. Mohly by se odehrávat například v poušti, ve městě mezi domy anebo téměř v libovolné krajině. Bylo by zde nutné dbát na výkonové nároky, které jsou v současnosti velmi nízké, a případná rozšíření zavádět tak, aby nebylo nutné omezovat podporu některých starších a méně výkonných zařízení. Z méně náročných úprav pak zmíním například efekty odlesku a rozptylu světla na objektivu kamery (tzv. efekt *lens flare*) nebo viditelné linky vzduchu či páry u konců křídel letadla, které by kromě vizuálního efektu mohly hráči pomoci lépe si představit, jakým směrem se letadlo aktuálně pohybuje.

Z hlediska ovládání zopakuji již zmíněnou možnost ovládání směrového kormidla. Jelikož by pravděpodobně většina hráčů tuto změnu nepřivítala, bylo by možné takové rozšíření nabízet volitelně. Bylo by však velmi zajímavé zjišťovat další možnosti navržené metody ovládání. Důležitým aspektem u úprav tohoto typu v již publikované hře by bylo zachování zpětné kompatibility herních výsledků. Změna ovládání, stejně jako změna parametrů letadla, může zpřístupnit nové a efektivnější letové trasy anebo naopak ztížit dokončení již existujících závodů. Případ, kdy hráč dostává možnost dokončit závod rychleji, je přirozeně lepší variantou. I ten by však mohl narušit možnost vzájemného srovnávání výkonů hráčů, například ve sledovaných statistikách.

Podle některých ohlasů by hráči uvítali také lepši přehled o konkrétních časech závodů, například ve smyslu možnosti zobrazení časů dřívějších letů. Pokud bych tuto myšlenku měl rozvést dál, pravděpodobně bych navrhl možnost zobrazení tzv. *ducha*. Šlo by o částečně průhledné letadlo, které by představovalo hráčův dřívější let. Díky tomu by bylo zřejmé, kde se hráč od dřívější trasy odchyluje a namísto přímého srovnání časů by hra nabízela srovnání vizuální.

S možností vzájemného srovnání souvisí i další možnost rozšíření – online žebříčky a závody pro více hráčů. Zmíněnou možnost zobrazení duchů by bylo možné rozšířit o sdílení těchto záznamů závodů mezi hráči. Ti by pak vedle předdefinovaných časů závodů mohli překonávat globální rekordy skutčných hráčů. Toto rozšíření by ovšem znamenalo řádově více práce, neboť by bylo nutné zajistit server ukládající tato data a jejich synchronizaci. Nelze také opomenout nutnost zabezpečení takových přenosů, aby nebylo možné výsledky závodů podvrhnout.

Jinou cestou rozšiřování hry pro více hráčů by mohlo být lokální hraní s využitím technologií jako Bluetooth nebo Wi-Fi. Zde by nároky na synchronizaci a bezpečnost spojení nebyly tak vysoké a současně by bylo možné nabídnout společné hraní v reálném čase. To by umožnilo také zavedení nových přístupů k závodům, které by vyžadovaly například spolupráci obou hráčů nebo naopak jejich vzájemné soupeření, v extrémním případě i formou vzdušných soubojů (které hráči v dotazníku také několikrát zmínili).

Pro úplnost zmíním také přidávání dalších závodů, přestože jde o rozšíření naprosto zřejmé. Tímto způsobem by bylo možné udržovat v aktivitě existující hráče. Opět zde ale hrozí jistá monotónnost a bylo by vhodné tato rozšíření kombinovat právě se zmíněnými grafickými doplňky. Zajímavým, byť opět realizačně náročným rozšířením, by byla možnost vytváření vlastních závodů samotnými hráči. Základní prvky pro tuto funkcionalitu už hra obsahuje, popsány byly v kapitole 5.5. Po nezbytných úpravách by se tato dosud vývojářská

obrazovka mohla stát plnohodnotnou součástí hry. Hráči by mohli vytvořené závody sdílet a soupeřit o nejlepší časy, ať už globálně či lokálně. Opět by ale platily dříve uvedené předpoklady o zabezpečení takovéto funkcionality proti zneužití.

Při úvahách o dalším rozšiřování hry je vhodné uvážit také ekonomickou stránku věci. Zatímco některé z navrhovaných úprav jsou jednoduché, jiné by vyžadovaly například provoz webového serveru. V takovou chvíli by nutně vznikaly náklady, které by vzhledem k počtu hráčů zřejmě nebyly nízké. Přestože náklady na dosavadní vývoj hry představovaly pouze strávený čas, i tento představuje omezené prostředky, které by měly vést k zisku (byť ne nezbytně materiálnímu). U her pro mobilní zařízení bývá obecným zvykem vydat hru ve dvou variantách – bezplatné a zpoplatněné. Bezplatná verze hry může sloužit jako demo a současně často obsahuje omezenou nabídku funkcí. V tomto případě by bylo možné vydat hru obohacenou o některé z výše uvedených rozšíření ve zpoplatněné verzi. Přestože by si placenou verzi zakoupil zřejmě jen zlomek stávajících hráčů, bylo by možné uvažovat o alespoň částečnému návratu investic do vývoje či provozu hry. Současně by ale vznikaly další problémy, kterými bezplatná verze netrpí, jako například pirátství. To může u žádaných her může dosahovat až k 90% nelegálně získaných kopií [15]. Další výzvou by tedy bylo nastavení vhodného obchodního modelu tak, aby se tyto ztráty minimalizovaly. Příkladem takového modelu může být nákup přímo v aplikaci (tzv. *in-app payments* nebo *in-app transactions*), kdy se samotná aplikace se základní funkcionalitou distribuuje bezplatně a dokupování dalšího obsahu a možností probíhá přímo v aplikaci.

Kapitola 6

Závěr

Tento text shrnul možnosti vývoje 3D her na platformě Android s využitím knihovny Libgdx. Dále se věnoval popisu letu skutečného letadla a z něj vycházející zjednodušené simulace letu používané ve hrách. Srovnává také existující metody ovládání leteckých her a navrhuje novou metodu, která závažnými nedostatky oproti ostatním netrpí. Navrhovaná metoda je založená na dotykovém vstupu za použití dvou prstů a nabízí velmi rychlé a přesné ovládání, čímž posiluje dojem, že má hráč letadlo plně pod kontrolou.

Dále tato práce popsala návrh a implementaci hry, která toto ovládání využívá. Jde o hru, kde hráč s akrobatickým letadlem prolétává brankami a snaží se doletět do cíle v co nejkratším čase. Hra je navržená tak, aby co nejvíce těžila z představené ovládací metody, podpořila tak její myšlenku a v ideálním případě i další rozšíření na poli mobilních leteckých her. V průběhu vývoje vznikl prototyp hry, který ověřil základní principy ovládání a umožnil srovnání s již používanými metodami. Následně vznikla plnohodnotná implementace hry s využitím právě knihovny Libgdx.

Hra byla průběžně konzultována s firmou CUKETA, s. r. o., která se vývojem her profesionálně zabývá. Před zveřejněním hry proběhl také dotazníkový průzkum, který ukázal, že hráči ovládací metodu i samotnou hru hodnotí velmi pozitivně.

Po získání a zpracování zpětné vazby byla hra publikována na internetový obchod Google Play pod názvem Air Racer[1]. Tam se v době dokončování této práce umístila na pozici 9. nejlepší bezplatné aplikace v kategorii závodních her a nainstalovalo si ji více než 70000 hráčů. Hra je kompatibilní s více než 94% mobilních zařízení platformy Android (dle dříve uvedených statistik).

[1] https://play.google.com/store/apps/details?id=cz.davidsabata.airracer

Literatura

[1] StatCounter. *Top 8 Mobile Operating Systems from Apr 2012 to Apr 2013* [online]. duben 2013 [cit. 17.4.2013]. Dostupné na: <http://gs.statcounter.com/?mobile_os-ww-monthly-201204-201304>.

[2] *Android Developers: Guides and reference* [online]. [cit. 28.12.2012]. Dostupné na: <http://developer.android.com/develop/>.

[3] Mario Zechner, R. G. *Beginning Android Games*. 2. vyd. [b.m.]: Apress, 2012. ISBN 978-1-4302-4677-0.

[4] *Libgdx: Desktop/Android/HTML5 Java game development framework* [online]. [cit. 17.4.2013]. Dostupné na: <http://libgdx.badlogicgames.com>.

[5] *Performance Tips: Use Native Methods Carefully* [online]. [cit. 28.12.2012]. Dostupné na: <http://developer.android.com/training/articles/perf-tips.html>.

[6] *Platform Versions* [online]. [cit. 2.5.2013]. Dostupné na: <http://developer.android.com/about/dashboards/index.html>.

[7] *Scene2d.ui: Official project documentation* [online]. [cit. 2.5.2013]. Dostupné na: <https://code.google.com/p/libgdx/wiki/scene2dui>.

[8] *Bullet Physics Library: Game Physics Simulation* [online]. [cit. 2.5.2013]. Dostupné na: <http://bulletphysics.org>.

[9] Langewiesche, W. *O umění létat*. [b.m.]: Baronet, 2010. ISBN 978-80-7384-307-6.

[10] NASA. *The Beginner's Guide to Aeronautics* [online]. [cit. 28.12.2012]. Dostupné na: <http://www.grc.nasa.gov/WWW/K-12/airplane/>.

[11] Banks, C. *A discussion of methods of real-time airplane flight simulation* [online]. [b.m.]: The Pennsylvania State University, 2000 [cit. 28.12.2012]. Dostupné na: <http://www.aerojockey.com/files/meng.pdf>.

[12] Jirkovský, J. *Game industry*. [b.m.]: D.A.M.O., 2011. ISBN 978-80-904387-1-2.

[13] Jirkovský, J. *Game industry 2*. [b.m.]: D.A.M.O., 2012. ISBN 978-80-904387-3-6.

[14] Likert, R. A technique for the measurement of attitudes. *Archives of Psychology*. 1932, roč. 22, č. 140. S. 5–55.

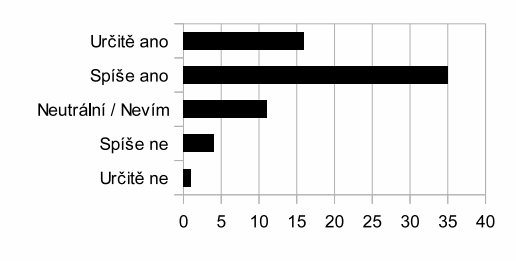
[15] Yin Poole, W. *Football Manager dev hopes to stick with Android despite 9:1 piracy rate* [online]. [cit. 2.5.2013]. Dostupné na: <http://www.eurogamer.net/articles/2012-04-24-football-manager-dev-hopes-to-stick-with-android-despite-9-1-piracy-rate>.

Příloha A

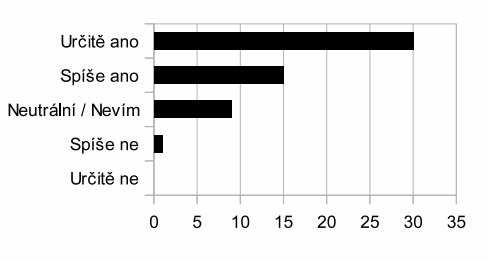
Úplné výsledky průzkumu

Tato příloha obsahuje úplný přepis uživatelských hodnocení, která byla získána v rámci průzkumu při prvotním zveřejnění hry, jak bylo popsáno v kapitole 5.6. V téže kapitole se také nachází zhodnocení výsledků. Průzkum se skládal z 32 otázek, kde hodnotící vyjadřovali, nakolik se shodují s daným tvrzením. Volitelně pak mohli doplnit libovolný komentář. Tyto komentáře jsou přejaty v původním znění a formátování.

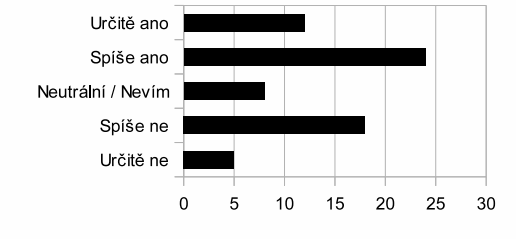
A.1 Míry souhlasu s tvrzením



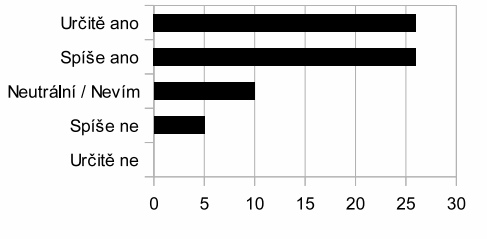
Obrázek A.1: Můj první dojem z aplikace byl pozitivní



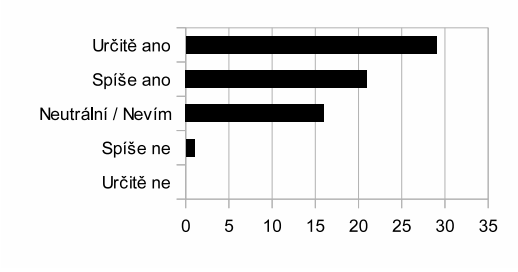
Obrázek A.2: Menu mi připadá přehledné



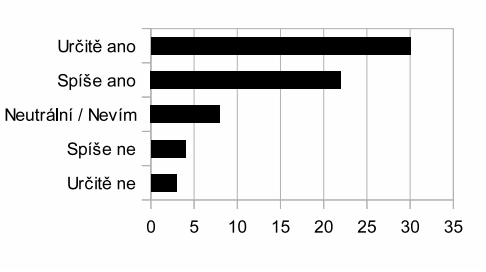
Obrázek A.3: Mám představu o tom, kolik je ve hře levelů



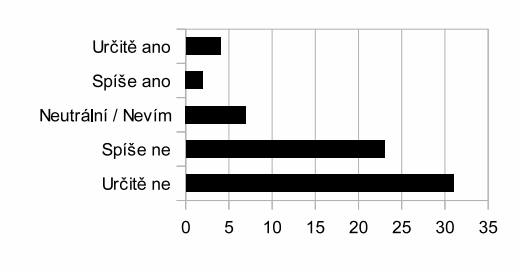
Obrázek A.4: Menu se chová tak, jak od něj očekávám



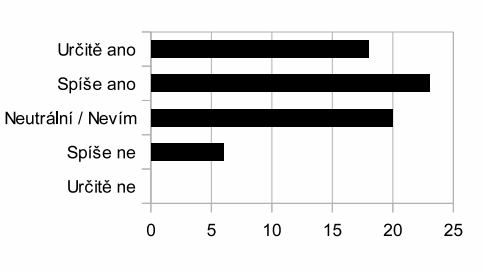
Obrázek A.5: Dovedl bych ve hře zapnout/vypnout zvuky



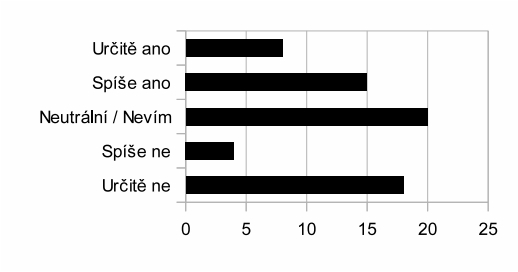
Obrázek A.6: Vím jak odemknout zamčený level



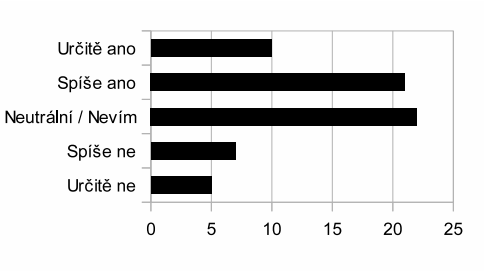
Obrázek A.7: Nedaří se mi vybrat položku kterou bych chtěl



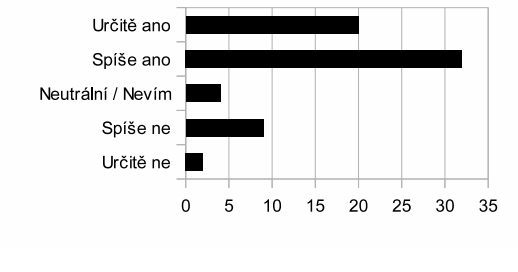
Obrázek A.8: Menu má hierarchickou strukturu



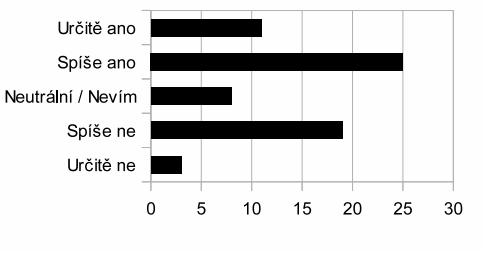
Obrázek A.9: Level lze spustit pouze tlačítkem Play



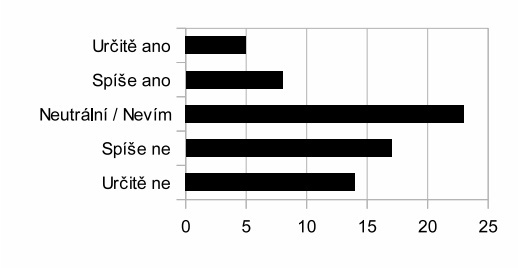
Obrázek A.10: Menu je na pohled pěkné



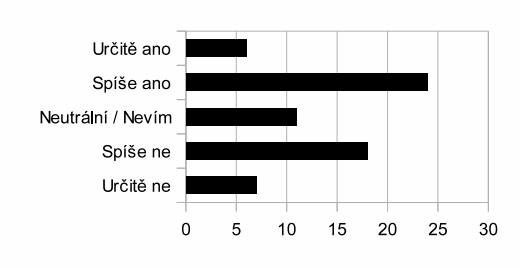
Obrázek A.11: Jsem schopen letadlo bez předchozího vysvětlení ovládat



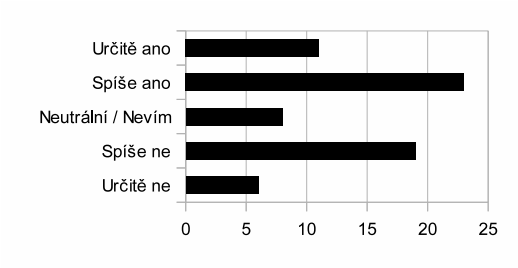
Obrázek A.12: Letadlo se chová předvídatelně



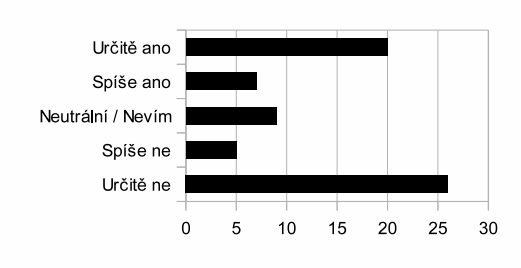
Obrázek A.13: Letadlo se chová stejně jako skutečná letadla



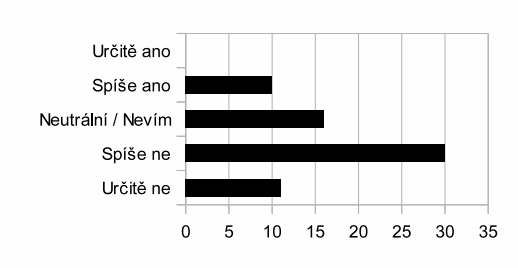
Obrázek A.14: Při letu mám letadlo pod kontrolou



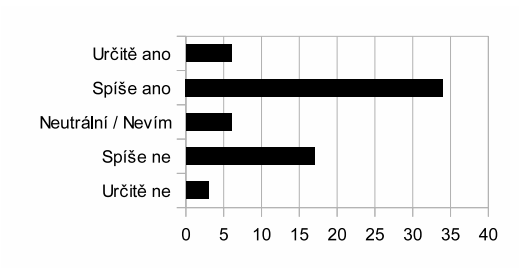
Obrázek A.15: Při letu mám dobrý odhad o výšce ve které letím



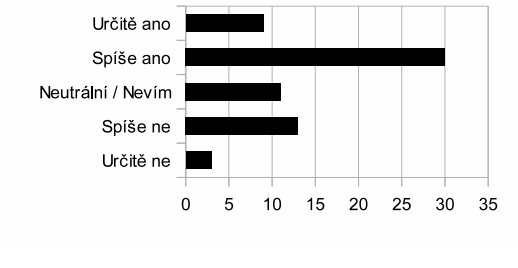
Obrázek A.16: Ovládání pro let nahoru/dolů je převrácené



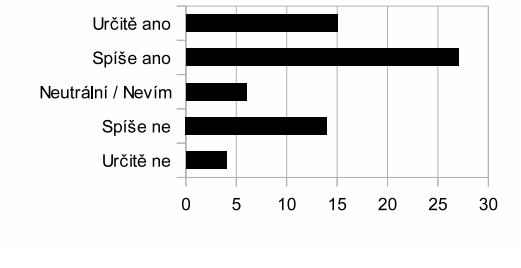
Obrázek A.17: Letadlo letí rychle



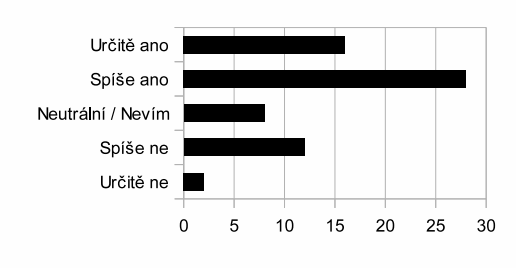
Obrázek A.18: Umím se trefit do kruhů



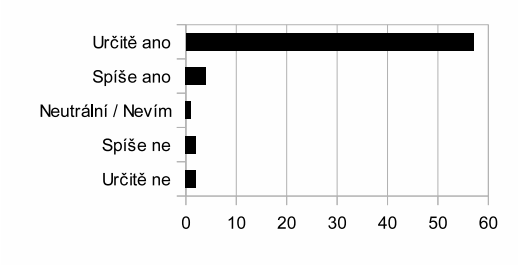
Obrázek A.19: Letadlo lze ovládat dostatečně jemně



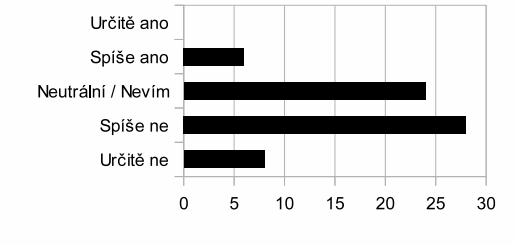
Obrázek A.20: S letadlem lze dělat ostré zatáčky



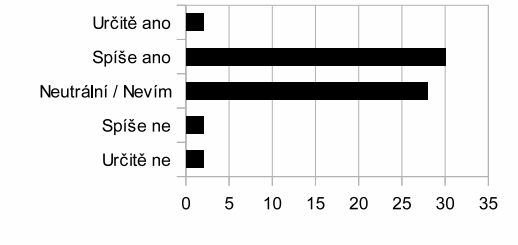
Obrázek A.21: Vím, jak můj pohyb prstů letadlo ovlivní



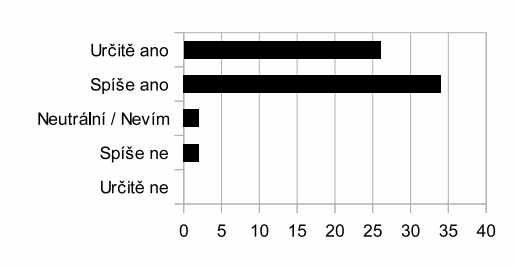
Obrázek A.22: Podařilo se mi dokončit alespoň jeden level



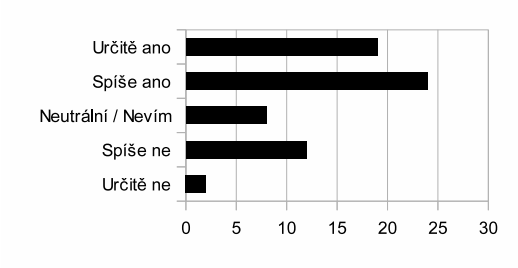
Obrázek A.23: Obtížnost levelů je příliš vysoká



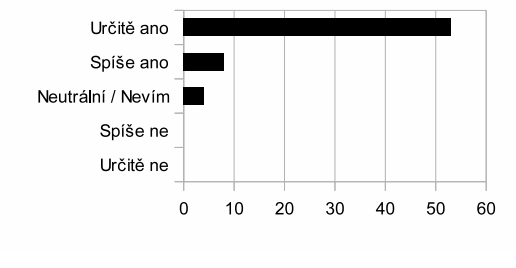
Obrázek A.24: Obtížnost levelů v jedné skupině je srovnatelná



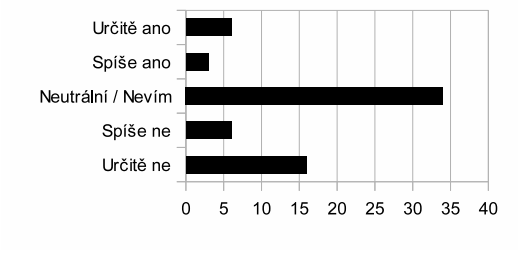
Obrázek A.25: Vždy vím, kam mám letět



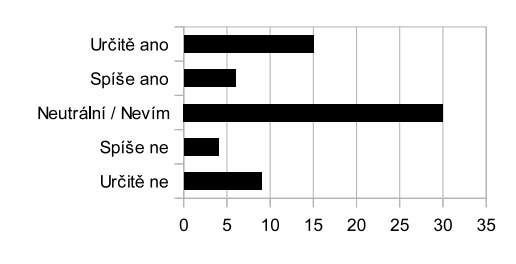
Obrázek A.26: Navigační šipka pro mě byla užitečná



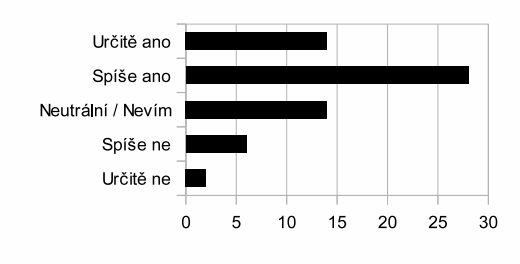
Obrázek A.27: V pozadí levelu bylo vidět město



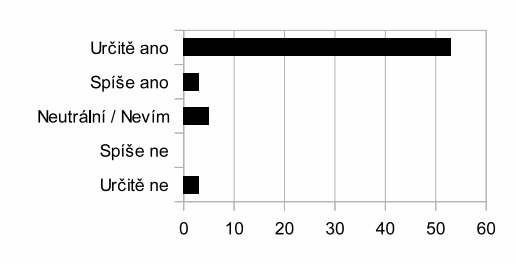
Obrázek A.28: V pozadí levelu byla vidět loď



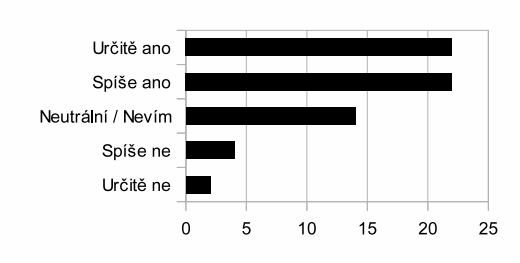
Obrázek A.29: V pozadí levelu byl vidět ostrov



Obrázek A.30: V levelu by mělo být více dekorativních objektů



Obrázek A.31: Alespoň jednou jsem získal za dokončení levelu tři hvězdičky



Obrázek A.32: Chtěl bych mít přehled o svých předchozích časech

A.2 Volitelné komentáře

Na SGNote(stock 4.1.2) přerušovaný zvuk. Pro mně náročné ovládání(ale já jsem levej a starej :D) Moře příliš jednotvárné. Ačkoliv obdivuji tvůrce, hra mi připadá spíše jako tutorial ke hře leteckých soubojů. Dlužno říci, že jsem nedokončil ani první level, a tento typ her není můj oblíbený. Přesto klobouk dolů a přeji hodně štěstí.

HTC One V - bez problémů

---

navigacni sipka je nepouzitelna, lepsi by byl "radar"+1 za ovladani bez akcelerometru

---

ZTE KIS PLUS 2.3.6 ok

Dobrá hra...jen ovládání bude chtít chvilku praxe. Jinak hra běží krásně plynule na Samsung Galaxy Nexus (i9250) Android 4.2.1.

V prvním levlu, když nedám prsty do kruhů, tak při zmáčknutí tlačítka zpět se level zastaví ale pořád je obraz překryt "prvnotním návodem"dále zvuk by se mohl plynule jevit, ne "zzzzzz (nic) zzzzzz (nic) ...."

Zlepši grafiku.

Výška by stačila třeba jen číselným údajem s případným varováním

---

UTC one x - jede bez problemu Herní zážitek je o něco horší než technická kvalita hry (ta se mi moc líbí)

Chybi mi tam moznost zatocit pomoci smerovky - tj. aby kridla zustatala vodorovne.

LG 4X HD P880 bez problemu.

Hral jsem 5 minut. Napad ovladani je supr. Vzhled je supr. Aplikace se mi libi. Ovladani letadla je ale v mym pripade docela katastrofa, pravdepodobne to ale jeste nemam uplne v ruce. Dokazu zatocit vlevo/vpravo, dokazu klesnout a stoupat, i kdyz to uz je horsi, jakmile mam ale tyhle dve veci kombinovat, tak je to katastrofa. Letadlo zataci hodne malo na to jak je nakloneny. Je prakticky nemozny udelat rychlou otocku o 180° pokud minu branku. Pokazdy, kdyz zatocim, prijde mi, ze letadlo zacne automaticky klesat, ale to je mozna moje chyba. Ja bych si to ovladani predstavoval nejak takhle: dam dva palce na obrazovku do neutralni polohy, pokud se levym palcem posunu dolu, zatocim vlevo a klesnu, pokud

pravym nahoru, zatocim vlevo a stoupam, levy nahoru =¿ vpravo a stoupam, pravy dolu =¿ vpravo a klesam (ted ale po zamysleni je to asi blbost, muselo by se hodne prehmatavat, takze asi nic :-). Celkove je na me ovladani malo citlive a letadylko pomalu prechazi mezi stavy vlevo¡-¿vpravo a nahoru¡-¿dolu, ale mozna je to takhle v realu, nevim jestli je to spis akrobaticky letadlo co maji na red bull soutezich nebo vyhlidkovy jednoplosnik. Suma sumarum je vsechno supr, ale nedokazu to ovladat, mozna pridat do menu hbitost letadla. Drzim palce na obhajobe. Pokud by to nekdy dospelo do stadia online leteckych souboju nebo bombardovani lodi/pevniny, byl by to narez. Mej se.

---

defaultně bych nastavil obrácenou y osu (původně se mi to nedalo hrát, až po přenastavení (za což chválím) to šlo)

Bylo by dobré ukázat časy potřebné k jednotlivým počtům hvězdiček. Když udělám poslední level v nějaké oblasti, hodí mě to poté v seznamu na předposlední level. Moc malý poloměr zatáčky při letu křídlem dolů. Tipy: Přidat ovládání směrovky a plynu. Přidat pohled z kabiny. Zvýšení rychlosti při klesání, snížení při stoupání. Možná udělat ukazatel rychlosti a výšky.

Co možnost ovládání gyroskopem a výběr letadel ?

Ahoj, hra vypadá opravdu pěkně! Líbí se mi jednoduchost a přímočarost hry. Jako námitky bych možná řekl toto: Uvítal bych, kdyby letadlo šlo ovládat i jedním prstem. Chybí ve hře nějaké menu, či pozastavení hry. Zvuk motoru je dost stereotypní. Šoupáním prstů vodorovně bych očekával zahnutí letadla. Zkus dát nějakou hudbu do pozadí.

Nexus 7, preji hodně štěstí s prací :-)

---

pěkné :)

Xpetia T, 4.1.2; postrádám ovládání zadních směrovek, ovládání multitouch bude chtít asi ještě pořádně vylepšit, asi je to zvyk z PC a joysticku, ale levely, kde jsem musel při klesání / stoupání zatáčet mi drnkaly na nervy.

Menu by se mohlo snaze posouvat... Letadlo se ovlada trochu ztuha

---

Acer B1 Textura moře. Asi by to chtělo nějakou, která víc navazuje. Pohled z vrchu byl slabší. Kostičky. Přijde mi, že letadlo reagovalo dost prudce. Aplikace nesla napoprvé odinstalovat. Napodruhé zmizela ze seznamu, ale systém napsal, že se odinstalace nepovedla. Jinak vše plynule.

Ani nevis jak ti zavidim mit mit diplomku na androidi hru. No kazdopadne obcas letadlo

reaguje ztezka a pomalu. Na to jakej to je prcek se to chova jak kdyz chces otocit Tupoleva nebo air force one o 360 stupnu. Trosku by to chtelo aby rychleji reagovalo. Ale jinak bylo vse plynule a bez problemu.

Samsung Galaxy W, Cyanogenmod 10.1 - běží perfektně :))

Osobně bych tam přidal i to ovládání pomocí senzoru v telefonu, protože přeci jen může některým lidem více vyhovovat a také by mohlo přilákat další hráče ;)

Na to, že je to od jednoho člověka, skvělá práce. Ve vyšších levelech je to trochu obtížnější, myslím, že by pomohlo mírné zpomalení letadla nebo oddálení překážek. U zvuku je znát, že se nahrávka opakuje,což může být rušivé. Jinak ovládání bezproblémové, funguje, jak má. Galaxy Nexus 4.2.2

Pozadie by sa mohlo meniť, aby človek nelietal vždy v rovnakom prostredí. Uvítal by som nejaké značky na display, ktoré by indikovali neutrálnu polohu. Uvítal by som umelý horizont, na ktorom by som videl, v akej polohe je lietadlo a prípadne aj výškomer. Tiež možnosť nejako meniť rýchlosť by nebolo odveci. Pylony by mohli byť vyššie. Robiť prudké zákruty v takej nízkej výške je trochu dosť náročné. Viac stupňov obtiažnosti (napríklad menšie/väčšie kruhy)

Možná by bylo fajn, kdyby šlo pomocí ukazatelu pozice v menu (tečky nad jednotlivými levely) posunout pozici rychleji než postupnými slajdy přes jednotlivé levely.

Šlape v pohodě SGN2 4.1.2 Omega v15 Pri nakloneni letadla na zvolenou stranu bych ocekaval mirny pohyb letadla do dane strany.

Pokud se netrefim do kruhu, podle sipky sice vim kde je kruh, jen nevim v jake vzdalenosti. Pridal bych nejakou vizuelni mapku kde jsem a kde je kruh popr. jak je daleko.