Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj mobilní aplikace a její publikování v internetovém obchodě

Jakub Truhlář

Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2014

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jakub TRUHLÁŘ**
Osobní číslo:	**E11418**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Vývoj mobilní aplikace a její publikování v internetovém obchodě**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je vytvořit mobilní aplikaci a provést její distribuci pomocí elektronického obchodu s aplikacemi.

Metodický postup:

1. Studium odborné literatury.
2. Publikace výsledků rešerše.
3. Návrh a popis vývoje a implementace výsledné aplikace, umístění do veřejnosti přístupného e-shopu.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **ČADA, Ondřej.** *Cocoa: úvod do programování počítačů Apple.* **1. vyd. Praha: Grada Publishing, 2009, 199 s. ISBN 978-80-247-2778-3.**
2. **KOCHAN, Stephen G.** *Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone.* **Vyd. 1. Brno: Computer Press, 2010, 550 s. ISBN 978-80-251-2654-7.**
3. **MARK, Dave a Jeff LAMARCHE.** *IPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch.* **Vyd. 1. Brno: Computer Press, 2010, 480 s. ISBN 978-80-251-2820-6.**
4. **PHILIP KOTLER, Gary Armstrong.** *Principles of marketing.* **12th ed. Upper Saddle River, N.J: Pearson/Prentice Hall, 2008. ISBN 978-013-6132-370.**
5. **PILONE, Dan a Tracey PILONE.** *Head first iPhone development.* **1st ed. Cambridge [Mass.]: O'Reilly, c2010, xxxii, 517 p. Head first series. ISBN 05-968-0354-0.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **2. ledna 2013**

Termín odevzdání bakalářské práce: **15. dubna 2014**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 29. března 2013

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V ..... dne ..... Podpis autora

Poděkování

Touto cestou bych rád poděkoval mému vedoucímu práce Mgr. Radimu Remešovi za věcné připomínky, ochotu a věnovaný čas. Dále své rodině a partnerce za podporu během mého studia a v neposlední řadě pak patří můj dík všem, kdo se o projekt hry zajímali.

Obsah

Kapitola 1

Úvod a cíl bakalářské práce

1.1 Cíl práce

Téma bakalářské práce „Vývoj mobilní aplikace a její publikování v internetovém obchodě“ se snaží přiblížit fáze vývoje mobilní aplikace od konceptu, přes samotný vývoj aplikace, až po publikování v samotném internetovém obchodě.

Práce provede čtenáře výběrem frameworku a vývojem aplikace pro mobilní operační systém iOS, konkrétně pro iOS 7 od společnosti Apple. Vývoj je prováděn v IDE Xcode téže společnosti. K vývoji aplikace, konkrétně 2D hry je použit framework Sprite Kit, který slouží k tvoření 2D her pro operační systém iOS 7 a výše, a který společnost představila poprvé právě s vydáním systému iOS 7. Výsledná hra je publikována v App Store, internetovém obchodě s aplikacemi pro zařízení s operačním systémem iOS.

Součástí práce jsou zajímavé rady k řešení různých nedostatků při vývoji a ke zkvalitnění práce. Při tvorbě samotné hry byly zapotřebí i rozsáhlejší grafické práce, o kterých se čtenář také dočte. V celé práci se objevují příkazy objektově orientovaného programovacího jazyka Objective-C, který je téměř výhradně používán právě k vývoji pro operační systémy iOS a Mac OS.

1.2 Předmluva

Začátek práce se zabývá popisem zařízení, operačního systému a frameworků na vývoj 2D her pro iOS. Je zde popsána obecná terminologie a detailně je představen framework Sprite Kit.

Metodika práce se již věnuje všem fázím samotné tvorby 2D hry za použití Sprite Kitu i IDE Xcode. Ke konci je čtenář seznámen s přípravou hotové aplikace pro vystavení v App Storu, se schvalovacím procesem a následně je mu přiblíženo prostředí pro správu aplikací iTunes Connect se statistikami a verzemi vydané hry.

1.3 Úvod do iOS

Mobilní operační systém iOS (dříve iPhone OS) je operační systém vyvíjený a distribuovaný firmou Apple Inc., na všechna mobilní zařízení značky Apple. Původně byl systém představen a určen pouze pro chytré telefony iPhone, ale postupem času se systém vyvinul v plnohodnotný mobilní operační systém pro všechna mobilní zařízení značky Apple. Od září 2007 pro multimediální přehrávač iPod Touch, od ledna 2010 pro tablety iPad, od září 2010 pro digitální mediální přijímač Apple TV a od listopadu 2012 pro iPad Mini. S postupem času, jak se objevovala nová a aktualizovala stávající zařízení se aktualizoval i samotný iOS. Celou hlavní verzi aktualizuje Apple zpravidla při vydání nového chytrého telefonu iPhone jednou za rok.[1] V současné době je iOS 7 používaný na více než 82% všech způsobilých zařízení (iClarified, 2014).

Během psaní této práce je aktuální verze iOS 7.1, která byla vydána 10. března 2014. Z trojice nejvýznamnějších mobilních operačních systémů, mezi které patří ještě Android OS a Windows Phone OS je platforma iOS jako jediná uzavřená, společnost Apple ji tedy nenabízí ostatním výrobcům hardwaru. To umožňuje Applu maximálně využít potenciál celého operačního systému na zařízeních, které Apple sám produkuje. Systém iOS je díky tomu maximálně konzistentní a optimalizovaný právě pro konkrétní hardware. Díky své uzavřenosti je iOS považován za nejvíce kon-

[1] iOS 5 na iOS 6 v září 2012, iOS 6 na iOS 7 v září 2013.

zistentní mobilní operační systém na trhu. Navíc právě ve verzi iOS 7 došlo k velké grafické změně celého uživatelského prostředí, kdy systém upustil od klasického sofistikovanějšího designu a skeuomorphismu a přešel k takzvanému plochému designu a minimalismu.

Uživatelské rozhraní je přitom stále založeno na bázi přímé manipulace za použití multi-dotykových gest a je tvořeno posuvníky, tlačítky a přepínači. Jako gesta se nejčastěji používají poklepání, táhnutí nebo přiblížení/oddálení prstů v návaznosti na konkrétní kontext (Mark a LaMarche, 2010).

Aplikace vydané pro iOS jsou psané převážně v jazyce Objective-C nebo C a vývoj probíhá převážně ve vývojovém prostředí Xcode, který je dodáván přímo společností Apple a je dostupný pouze pro operační systém Mac OS, taktéž od Applu.

Hotovou aplikace lze legálně a oficiálně dostat na zařízení pouze s od Applu podepsaným certifikátem, který vám společnost udělí mimo jiné pouze pokud jste registrovaný vývojář, což obnáší poplatek 99 USD/rok. Pokud chce navíc vývojář publikovat svou aplikaci v obchodě App Store, musí ji napřed nechat projít schvalovacím procesem, na jehož konci ji Apple schválí nebo zamítne. Apple je zkrátka v tomto případě velmi striktní a snaží se tak chránit svoji platformu před problémovými aplikacemi se škodlivým kódem (Pilone a Pilone, 2010).

Kapitola 2

Přehled zařízení pro systém iOS

2.1 iPhone

Je tzv. chytrý telefon, který v sobě spojuje funkce mobilního telefonu a kapesního počítače. Uživatel iPhone ovládá pomocí multi-dotykového displeje, jehož součástí je i virtuální klávesnice. K internetu se lze připojit přes WI-FI nebo přes mobilní sítě pomocí GPRS, EDGE, 3G, 4G nebo třeba v současnosti nejrychlejší LTE technologií (Apple, 2014a). Telefon je vybaven operačním systémem iOS, který již nejnutnější aplikace jako jsou telefon, mail, prohlížeč, zprávy, iPod (hudební přehrávač), fotoaparát, kalendář, počasí, kalkulačka apod. obsahuje.

Obrázek 2.1: *Všechny telefony iPhone seřazené podle data vydání*

Krom toho Apple v březnu 2008 představil iPhone SDK, která umožnila třetím stranám začít vyvíjet nativní aplikace a publikovat je v App Storu, což je část internetového obchodu iTunes Store vyhrazená právě aplikacím.

2.1.1 Displej a rozlišení

Ve všech iPhonech, potažmo všech dotykových zařízeních od Applu najdeme displeje kapacitní, není tedy nutno na displej vyvíjet tlak a psaní je možné pouhými bříšky prstů (Wilson, 2007).

Displej měl od původního iPhonu až po iPhone 3GS 3,5 palcový displej s rozlišením 320x480 pixelů, při poměru stran 3:2 (viz obrázek 2.1). Po něm přišla velká revoluce v podobě tzv. retina displeje, což je displej u kterého uživatel není schopen rozeznat jednotlivé pixely. Toho Apple docílil zdvojnásobením počtu pixelů a nový iPhone 4 tak měl stejnou úhlopříčku 3,5 palce a stejný poměr stran tedy 3:2 jako jeho předchůdci, ale rozlišení se zvýšilo na 640x960 pixelů (viz tabulka 2.1).

Tabulka 2.1: *Přehled základních parametrů iPhone displejů*

Model	iPhone	iPhone 3G	iPhone 3GS	iPhone 4	iPhone 4S	iPhone 5	iPhone 5C	iPhone 5S
Rozlišení [pixelů]	320x480	320x480	320x480	640x960	640x960	640x1136	640x1136	640x1136
Poměr stran	3:2	3:2	3:2	3:2	3:2	16:9	16:9	16:9
Úhlopříčka	3,5"	3,5"	3,5"	3,5"	3,5"	4"	4"	4"

Další model iPhone 4S si zachoval stejné hodnoty a další výrazná změna nastala až s příchodem nového modelu. Tím byl iPhone 5, který si naopak ponechal hustotu pixelů, konkrétně 326 ppi, ale zvětšil svou úhlopříčku na 4 palce s poměrem stran 16:9 a s rozlišením 640x1136 pixelů. Zatím poslední iPhone 5C a iPhone 5S, vydané ke stejnému datu tyto proporce zachovávají.

2.1.2 Tělo zařízení

Telefon disponuje pouze 4 fyzickými tlačítky. Asi nejdůležitější a nejpoužívanější je tzv. home button, tedy tlačítko domů, kterým se uživatel vrací na hlavní obrazovku, minimalizuje aplikace, vyvolává otevřené aplikace nebo spouští Siri (inteligentní osobní asistent). Tlačítko v sobě navíc od iPhonu 5S skrývá čtečku otisků prstů využívanou především k odemykání iPhonu a k přihlašování k uživatelovo dalším účtům jako náhrada za vypisování hesla z virtuální klávesnice.

Dalšími tlačítky jsou dvě tlačítka na ovládání hlasitosti a tlačítko na uspání nebo

vypnutí iPhonu. Ještě se na iPhonu vyskytuje už od původní verze dvoupolohový přepínač hlasitého a tichého režimu.

2.1.3 Senzory

V iPhonu najdeme celou řadu senzorů. Akcelerometr k přepínání zobrazení na výšku nebo na šířku, magnetometr, gyroskop, proximity senzor nebo senzor dopadajícího světla, který reguluje jas displeje. Jak již bylo zmíněno výše, s příchodem iPhone 5S přibyl senzor otisku prstu.

2.1.4 Procesor a paměti

Co se výkonu týče, je procesor asi nejvíce obměňovanou částí celého přístroje. V iPhonu 4, který je stále považovaný za nejvíce revoluční ze všech modelů, byl poprvé použit Applem navržený procesor s označením Apple A4 taktovaným na 800 MHz. Následován procesory Apple A5 (800 MHz, dvoujádrový) v iPhone 4S, Apple A6 (1,3 GHz, dvoujádrový) v iPhone 5/5C a zatím poslední procesor použitý v iPhone 5S je Apple A7, který má oproti předchůdci jako první mobilní procesor na světě 64-bitovou architekturu (Souppouris, 2013).

Operační paměť se s postupem času zvětšuje od původních 128 MB, přes 256 MB u iPhonu 3GS, 512 MB u iPhonu 4 a 4S, po 1 GB u iPhonu 5, 5C a 5S.

2.1.5 Kapacita

Kapacita se od starého iPhonu 3G až po nejnovější iPhone 5S navyšovala zpravidla geometrickou řadou. Zatímco u starších modelů byly v nabídce 8 a 16 GB flash paměti, nejnovější modely již nenabízejí 8 GB variantu, ale nabízejí 16 GB, dále pak 32 GB a největší 64 GB varianty. Protože rozdíly ve velikostech jsou veliké, je stále nutné pamatovat na uživatele s menší pamětí a hledět na velikost publikované aplikace.

2.2 iPad

Je mobilní zařízení typu tablet, je tedy zpravidla vždy větší než iPhone. Uživatel tablet ovládá pomocí velkého multi-dotykového displeje, jehož součástí je i virtuální klávesnice.

Na rozdíl od iPhonu je iPad nabízen vždy ve dvou variantách. První varianta bez modulu pro připojení k mobilní datové síti, tedy pouze s WI-FI nebo varianta WI-FI + cellular, tedy k WI-FI modulu přibude navíc slot na SIM kartu v patřičné velikosti, díky které se uživatel připojí ke konkrétní mobilní datové síti, stejně jako iPhone.

Obrázek 2.2: *Všechny tablety iPad seřazené podle data vydání*

Takový iPad ale využívá SIM kartu skutečně pouze pro připojení k internetu, protože systém iOS na iPadu nikdy nezpřístupní aplikace telefon a zprávy jako u iPhonu, což je pochopitelné i ze samotného faktu, že se jedná o tablet a ne o telefon. Nicméně uživatel si může vybrat z velkého množství aplikací podle typu hovoru. Sám Apple nabízí službu a aplikaci FaceTime pro audiohovory i videohovory pomocí internetu, implementovanou přímo v systému, je tedy potřeba mít pouze přední kameru na svém zařízení. Pro textové zprávy existuje rovněž nepřeberné množství služeb a aplikací, samotný Apple opět do systému implementoval přímo službu a aplikaci iMessage pro textové zprávy přenášené pomocí internetu. Nutno dodat, že v případě obou zmíněných aplikací se jedná o distribuci pouze v rámci systému iOS.

2.2.1 Displej a rozlišení

Displej má od původního iPadu až po současný iPad Air úhlopříčku vždy 9,7 palců, výjimku tvoří pouze iPad Mini a iPad Mini 2, které mají 7,9 palců (viz tabulka 2.2).

Tabulka 2.2: *Přehled základních parametrů iPad displejů*

Model	iPad	iPad 2	The New iPad	iPad with Retina display	iPad Mini	iPad Air	iPad Mini 2
Rozlišení [pixelů]	1024x768	1024x768	2048x1536	2048x1536	1024x768	2048x1536	2048x1536
Poměr stran	4:3	4:3	4:3	4:3	4:3	4:3	4:3
Úhlopříčka	9,7"	9,7"	9,7"	9,7"	7,9"	9,7"	7,9"

Totožné displeje iPadů zasáhl až příchod retina displejů, proto první dva modely iPad a iPad 2 mají rozlišení 1024x768 pixelů a všechny ostatní počínaje tzv. Novým iPadem[1] mají rozlišení 2048x1536 pixelů, tedy dvojnásobné. U 9,7 palcových iPadů s retina displejem je hustota pixelů menší než 300 ppi, přestože bývalý ředitel firmy Apple, Steve Jobs, při představení prvního retina displeje pro iPhone prohlásil, že hranice hustoty pro retina displeje je 300 ppi a že až při dosažení této hodnoty uživatel přestane vnímat jednotlivé pixely (Apple, 2010). Ovšem důležité je zdůraznit, že právě hranice 300 ppi je chápána spíše jako hranice pro menší displeje telefonů, protože větší displej tabletu drží uživatel ve větší vzdálenosti od svých očí.

Poměr stran 4:3 je totožný pro všechny dosavadní iPady.

2.2.2 Tělo zařízení

Tělo iPadu obsahuje 5 fyzických tlačítek stejných jako iPhone, což nastiňuje záměr Applu, aby pro uživatele bylo jednoduché a přirozené ovládat po předešlé zkušenosti všechny ostatní Apple zařízení.

2.2.3 Senzory

V iPadu najdeme totožné senzory, jako u telefonu iPhone. Výjimku tvoří proximity senzoru, který reaguje na přiblížení přístroje k hlavě uživatele a zajistí, aby se displej vypnul. Protože však uživatel pomocí iPadu nevolá tímto stylem, není senzor v zařízení nutný. Druhou výjimku tvoří úplně první iPad, kterému ještě chyběl navíc gyroskop.

[1] The New iPad je oficiální označení iPadu 3. generace.

2.2.4 Procesor a paměti

V původním iPadu se nachází procesor Apple A4 taktovaný na 1 GHz. Následně se v iPadu 2 a iPadu Mini objevil procesor Apple A5 (1 GHz, dvoujádrový).

V Novém iPadu se objevil procesor Apple A5X, taktéž dvoujádrový procesor s taktem 1 GHz, rozdíl je ovšem v tom, že ve skutečnosti se nejedná pouze o CPU, ale všechny procesory od Applu jsou označovány jako SoC (Anthony, 2012), tedy v sobě snoubí jak CPU, tak jeho paměti a GPU. Procesor Apple A5X má tedy oproti předešlé verzi dvojnásobný grafický výkon.

V iPadu 4. generace označovaném jako iPad s retina displejem nebo „iPad with Retina display“, přestože tento displej měl už jeho předchůdce (Shanklin, 2012), najdeme dvoujádrový procesor Apple A6X taktovaný na 1,4 GHz. Poslední 9,7 palcový iPad Air a iPad Mini 2 obsahují stejně jako poslední iPhone 5S dvoujádrový procesor Apple A7 s taktem 1,3 GHz u iPadu Mini 2 nebo s taktem 1,4 GHz u iPadu Air.

Operační paměť se s postupem času zvětšuje od původních 256 MB u iPadu, přes 512 MB u iPadu 2 a iPadu Mini po 1 GB u všech ostatních iPadů.

2.2.5 Kapacita

Všechny vydané iPady byly a jsou nabízené s kapacitou 16, 32 a 64 GB. U iPadu s Retina displejem, iPadu Air a iPadu Mini 2 přibyla verze se 128 GB flash pamětí.

2.3 Shrnutí pro vývojáře

Před samotným vývojem aplikace je třeba zvážit, na jakých zařízeních aplikace půjde spustit. Pokud je aplikace spustitelná na více zařízeních, na jedné straně to znamená více potenciálních uživatelů, na straně druhé často více kódu, horší optimalizaci a větší velikost aplikace.

2.3.1 Retina displej

Jak bylo již zmíněno výše, retina displej dokáže zobrazit dvojnásobné množství pixelů při stejné úhlopříčce, uživatel není schopen pixely rozeznat a vnímá tak lépe každý detail. Vývojáři však s tímto faktem musejí počítat, a proto je třeba si uvědomit, že při psaní kódu se pozice, délka, souřadnice apod. neurčují v pixelech, ale v bodech (Waynik, 2014). Na obyčejném displeji je jeden bod právě jeden pixel, na retina displeji je bod roven dvěma pixelům jak horizontálně, tak vertikálně. Následující kód nastavuje pozici hráče v bodech.

Zdrojový kód 2.1: *Příklad nastavení pozice hráče*

```
_hrac.position = CGPointMake (10, 10);
```

Na obyčejném displeji se hráč objeví na pixelu o souřadnicích (10, 10) a na retina displeji na (20, 20). Stále však budou v rámci displeje na totožných místech.

Pokud při vývoji pracujeme s obrázky, určitě je žádoucí používat jejich správnou velikost. Pokud má tedy uživatel zařízení s retina displejem, musíme přinutit aplikaci, aby používala obrázky s dvojnásobným rozlišením. Apple proto implementoval do Xcode jednoduchý způsob, jak lze této funkce dosáhnout. Stačí pouze, aby obrázek, který se má zobrazovat v případě retina displeje měl stejný název jako obrázek pro obyčejný displej a hned za názvem uvedeno *@2x*. V celém kódu pak vývojář pracuje pouze s obyčejným názvem a Xcode sám vyhodnotí, kterou verzi má použít.

2.3.2 4-palcový iPhone

Čtyřpalcový displej je první změnou poměru stran od dob představení prvního telefonu iPhone. Ačkoliv šířka zůstala stejná, výška displeje se zvýšila, a proto vývojář v rámci konceptu nastavování bodů namísto pixelů pracuje s hodnotou 320x568 bodů.

2.3.3 iPhone, iPad a poměr stran

Pokud chceme, aby naše aplikace byla univerzální, tedy spustitelná správně jak na iPhonu, tak iPadu, musíme najít způsob, jak se vypořádat z rozdílným poměrem stran. Způsobů je velké množství, spousta vývojářů se s tím navíc vypořádává po svém. Dobrým způsobem je využití pokročilejšího typu definice (Čada, 2009).

Zdrojový kód 2.2: *Makro pro sestavení správného názvu souboru*

```
#define CHOOSE_BY_DEVICE(regular) ((self.size.height <= 480.0) ? regular : (self
   .size.height <= 568.0) ? [regular stringByAppendingString:@"-568h"] : [
   regular stringByAppendingString:@"-1024h"])
```

Definovaný název přebírá jeden argument, konkrétně textový řetězec. Samotné makro pak podle výšky displeje zadaný řetězec upraví, práci nám přitom usnadní služba *stringByAppendingString:* třídy *NSString* (Kochan, 2010). Protože se navíc jedná o definici (viz zdrojový kód 2.2), výstupem tohoto makra bude právě a pouze upravený textový řetězec.

Pokaždé když budeme v našem projektu zadávat název pozadí, využijeme toto makro k úpravě regulérního názvu, a tedy k navrácení správného názvu obrázkového souboru, se kterým bude aplikace dále pracovat.

Obrázek musí v tomto případě existovat v těchto pěti verzích:

- obyčejná verze pro iPhone
- retina verze (s názvem rozšířeným o *@2x*)
- retina verze pro čtyřpalcový iPhone (s názvem rozšířeným o *-568h@2x*)
- obyčejná verze pro iPad (s názvem rozšířeným o *-1024h*)
- retina verze pro iPad (s názvem rozšířeným o *-1024h@2x*)

Kapitola 3

Výběr frameworku

Při vývoji hry je třeba zvážit, jaké technologie při vývoji využijeme. Obecně budeme těmito technologiemi myslet systémové frameworky pro tvorbu her, popřípadě se lze u komplexnějších vývojových herních systémů setkat i s označením herní engine.

3.1 Přehled

Pro vývoj hry na platformu iOS existuje hned několik frameworků, které lze k realizaci projektu využít. Apple pro tvorbu aplikací nabízí vývojářům oficiální vývojové prostředí Xcode, do kterého se samozřejmě lehce implementují nejrůznější frameworky, ale až do vydání iOS 7 zde žádný oficiální framework pro tvorbu her nebyl. Také z toho důvodu vznikl Cocos2D-iPhone, odnož projektu Cocos2D. Dalším velkým a známým projektem je herní engine Unity, který nabízí verzi Unity 2D, zaměřenou právě na vývoj 2D her. S představením iOS 7 však Apple vydal svůj vlastní framework Sprite Kit pro vykreslování grafiky, práci s animacemi a 2D texturami, všeobecně tedy také pro tvorbu 2D her.

V zásadě dokáží všechny zmíněné frameworky vyprodukovat hodnotné hry, a tak se u různých úspěšných projektů můžeme setkat s každým ze zmíněných frameworků. Je proto třeba důkladně zvážit, na co bude při vývoji kladen důraz, o jaký koncept hry se bude jednat popř. pro jaké platformy plánujeme hru vydat.

3.2 Cocos2D-iPhone

Obrázek 3.1: *Cocos2D*

Jedná se o framework pro vývoj 2D her a interaktivních aplikací, který se implementuje přímo do vývojového prostředí Xcode. Vychází z konceptu svého rodičovského projektu Cocos2D, ale namísto programovacího jazyka Python využívá pro iOS vývoj typický programovací jazyk Objective-C (Cocos2D-iPhone, 2014).

Nabízí základní práci s animacemi, které je schopen pomocí akcí a časovačů dále propojovat a tvořit tak komplexnější animace. Umožňuje manipulovat s velikostí a pozicí spritu[1]. Umožňuje používat základní objekty jako jsou tlačítka, labely nebo třeba textová pole. Podporu fyziky zajišťuje Box2D a Chipmunk engine.

Novinkou je nástroj SpriteBuilder pro návrh grafického uživatelského rozhraní a jednoduchou tvorbu komplexních animací.

Výhody:

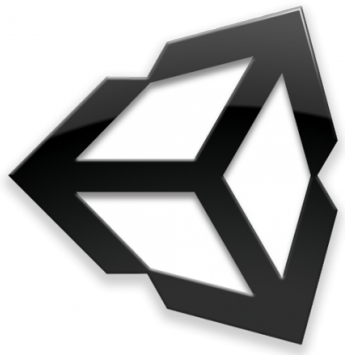
- Je multiplatformní, umožňuje tedy vyvíjenou hru exportovat jak na iOS, tak nejčastěji také na Android OS.
- Jedná se o open source distribuci.
- Disponuje velkou uživatelskou základnou.
- Často aktualizován.
- Existence velkého množství nástrojů (ParticleDesigner, SpriteBuilder).

Nevýhody:

- Nutná složitější implementace do Xcode.
- Většina užitečných nástrojů není součástí distribuce.

[1] Jde o menší dvourozměrný obrázek, který je vložen do větší textury.

3.3 Unity 2D



Obrázek 3.2: *Unity*

Jde o multiplatformní plnohodnotný herní engine s vlastním vývojovým prostředím, ve kterém má vývojář k dispozici většinu důležitých nástrojů. Může proto přímo ve vývojovém prostředí navrhovat GUI, nebo třeba pracovat s částicemi. Vývoj není zaměřen tolik na psaní kódu, ale je spojen i s nastavováním a prací s nástroji v programu. Při psaní kódu se používá programovací jazyk C#, JavaScript nebo Boo, pro které je třeba namísto Xcode využívat Mono develop IDE.

Jelikož se jedná o plnohodnotný a velmi komplexní herní engine, je v plné verzi zpoplatněn. Cena Unity Pro verze je 1 500 USD, popřípadě 75 USD/měsíc. Nabízí však svým uživatelům přístup do Unity Asset Storu, ve kterém se nachází balíčky s modely, texturami, hudba, zvuky, částice atd.

Výhody:

- Jedná se o plnohodnotný herní engine.
- Takřka nejlepší pro multiplatformní vývoj.
- Méně náročný na kód (jednodušší pro nového vývojáře).
- Unity Asset Store nabízí vývojářům balíčky textur, zvuků, modelů atp.
- Obsahuje vlastní IDE, ve kterém se nachází většina důležitých nástrojů.

Nevýhody:

- Je zpoplatněn, jinak je zdarma nabízen pouze v omezené verzi.
- Neimplementuje se do Xcode, má vlastní IDE.
- Pro psaní kódu se namísto Xcode a jazyka Objective-C využívá oddělené vývojové prostředí Mono develop a převážně jazyk C# nebo JavaScript.

3.4 Sprite Kit

Obrázek 3.3: *Sprite Kit*

Je nový Applem vydaný a oficiálně podporovaný framework pro vykreslování grafiky, práci s animacemi, 2D texturami, zvuky atp., jehož součástí je i fyzikální engine a editor emitoru částic (Apple, 2014b).

Byl představen při příležitosti vydání systému iOS 7 a v současné době funguje pouze na tomto systému. Z toho důvodu není možné využívat Sprite Kit pro vývoj na jiné platformy, ani spouštět aplikace v něm vytvořené na jiných verzích systému iOS.

Celý framework včetně nástrojů je přímou součástí vývojového prostředí Xcode, čímž je zajištěna maximální kompatibilita a konzistentnost celého vývojového procesu a jako programovací jazyk se přirozeně používá Objective-C.

Do značné míry je zde vidět obdobná syntaxe jako u jiných frameworků pro vývoj 2D her na iOS (Cocos2D-iPhone), což je pro framework i stávající vývojáře velmi vhodné. Jelikož se navíc jedná o mladý framework, je zde velký potenciál růstu možností frameworku i uživatelské komunity.

Výhody:

- Je součástí vývojového prostředí Xcode.
- Je zcela zdarma.
- Oficiálně vytvářen a podporován Applem.
- Důležité nástroje jsou součástí Xcode.
- Velký růstový potenciál.

Nevýhody:

- Stále není úplně plnohodnotným herním frameworkem.
- Aktualizován najednou, ale méně často.
- Menší uživatelská komunita.
- Uzavřenost, vývoj pouze pro iOS.

3.5 Výběr frameworku pro vývoj hry

Po představení nejběžnějších a nejúspěšnějších frameworků pro tvorbu 2D her na platformu iOS, je zapotřebí zvolit takovou technologii, která by splňovala stanovená kritéria. Záměrně je přitom vynecháno kritérium multiplatformní možnosti vývoje, protože je pro nás více podstatné vyvíjet pro systém iOS a dbát na maximální optimalizaci pro všechna jeho zařízení.

Kritériím je pomocí párového srovnání přiřazena váha. Následně je frameworkům v rámci každého kritéria přiřazeno pořadí, výsledné hodnoty variant jsou vypočteny jako vážené součty. Vítězí varianta s nejnižším váženým součtem.

Stanovená kritéria pro výběr frameworku:

1. **Minimální cenové zatížení** ($v_1 = 0,2$)
2. **Nenáročnost implementace a následná funkčnost** ($v_2 = 0,3$)
3. **Propojení s oficiálními nástroji** ($v_3 = 0,3$)
4. **Potenciál frameworku** ($v_4 = 0,2$)

Prvním kritériem je cenové zatížení, přičemž v tomto případě je ideální technologie, která je zdarma, což sebou nese i taková pozitiva jako větší uživatelská základna, její budoucí potenciál a podpora stran dokumentace.

Tabulka 3.1: *Hodnocení kritérií při výběru frameworku*

Framework	K1	K2	K3	K4	Výsledek	Umístění
Cocos2D-iPhone	**1.**	**3.**	**2.**	**2.**	**2,1**	**2.**
Unity 2D	**2.**	**2.**	**3.**	**2.**	**2,3**	**3.**
Sprite Kit	**1.**	**1.**	**1.**	**1.**	**1,0**	**1.**

Dále je velmi podstatným kritériem nenáročná implementace frameworku do vývojového prostředí, a protože je Xcode v současné době při vývoji aplikací velmi mocným nástrojem, je přínosné snažit se o maximální kompatibilitu a funkčnost tohoto IDE. Například Unity 2D má vývojové prostředí vlastní, ale pro psaní kódu

využívá jiné propůjčené IDE, takže propojení s oficiálními nástroji Applu je horší a vývoj se stává i komplikovanější.

Potenciál frameworku je do jisté míry odrazem jeho úspěšnosti, v případě nového Sprite Kitu je ale toto kritérium daleko zásadnější. Apple se s tímto frameworkem dostal do ideální situace, kdy má skvělé podmínky pro optimalizaci právě pro vlastní operační systém a vlastní zařízení. Tabulka 3.1 zachycuje pořadí variant v rámci jednotlivých kritérií a výsledné umístění frameworků.

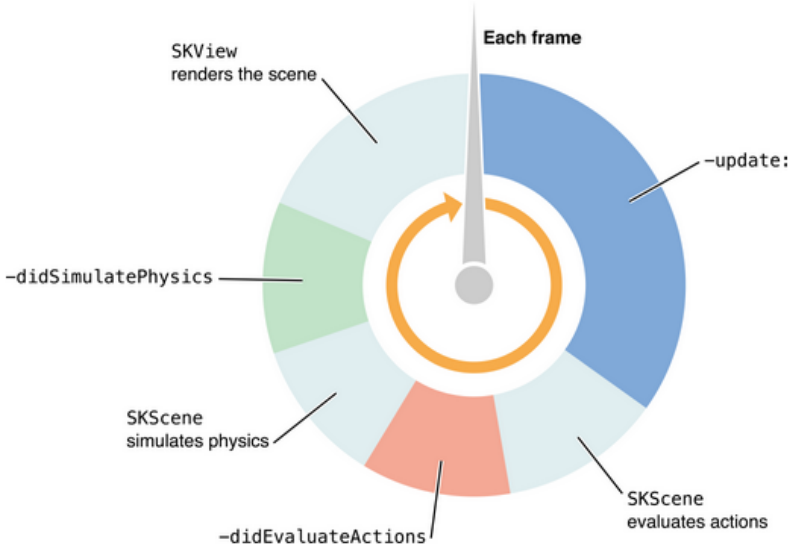
Vezmeme-li v úvahu i atraktivitu jednotlivých frameworků, nejlépe splňuje naše požadavky právě Sprite Kit a z toho důvodu bude využit pro vývoj právě tento framework.

Kapitola 4

Sprite Kit

Sprite Kit využívá pro vykreslování svého obsahu smyčku (viz obrázek 4.1), podle které je každý snímek vyhodnocen ještě předtím, než je vykreslen (Apple, 2014b). Mimo jiné tak framework zaručuje správné vyhodnocení akcí, fyziky a aktualizací kódu předtím, než dojde k samotnému vykreslení. Obnovovací frekvence displeje všech Apple zařízení je 60 FPS. Z toho důvodu je při ideálním stavu tato smyčka provedena 60krát za vteřinu.

Obrázek 4.1: *Vykreslovací smyčka*



Zdroj: Sprite Kit dokumentace

4.1 Scény

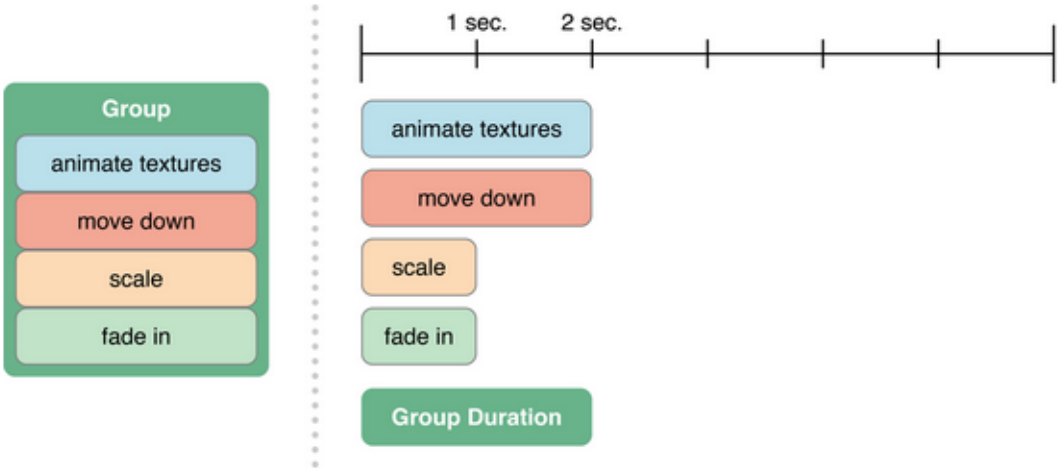
Každý obsah je vykreslován pomocí třídy *SKView* a takovému obsahu se říká jednoduše scéna, konkrétně jde o objekt třídy *SKScene* a jako takový může mít svá specifika. Například může mít scéna vypnutou gravitaci, protože není ve hře potřeba.

Avšak taková scéna je pořád prázdná. Je proto třeba vytvořit základní logický strom nodů, jakýchsi uzlů, které v sobě sdružují další nody. Pro příklad si představme letadlo, jedná se o node. Z motorů letadla jde kouř, tedy node (konkrétně jde o částice viz sekce 4.4), který je součástí svého rodiče, zde tedy letadla.

4.2 Akce

Součástí každé scény jsou i akce, které se po vytvoření určeným nodem spouštějí. Může jít o pohyb, rotaci, změnu velikosti nebo třeba přehrání zvuku tohoto nodu. V každém případě se akce provádí plynule, a proto lze ve většině případů nastavit právě dobu, po kterou se daná akce provádí.

Obrázek 4.2: *Provádění akcí ve skupině*

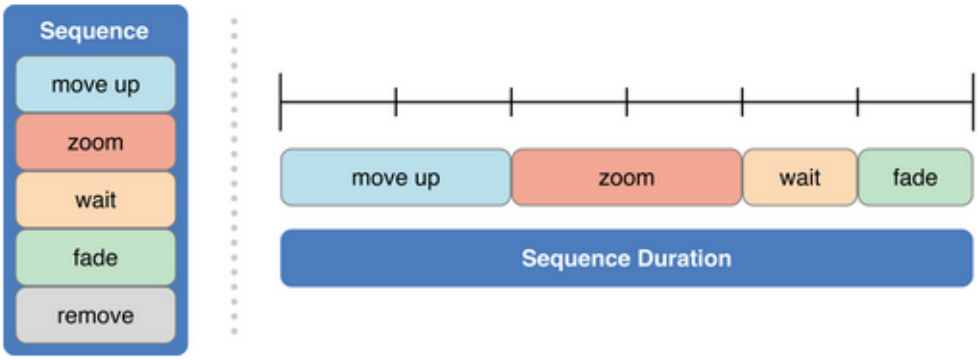


Zdroj: Sprite Kit dokumentace

Akce můžeme sdružovat do skupin, ve kterých se akce provádějí současně (viz obrázek 4.2) nebo do sekvencí, ve kterých se provádějí postupně (viz obrázek 4.3).

Můžeme je samozřejmě libovolně kombinovat mezi sebou. Dále je můžeme zastavovat, opakovat nebo je nechat spustit specifický blok kódu, můžeme tak tvořit celé komplexní akční řetězce.

Obrázek 4.3: *Provádění akcí v sekvenci*

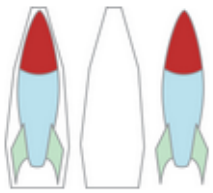


Zdroj: Sprite Kit dokumentace

4.3 Simulace fyziky

Při simulaci fyziky je třeba přidat nodům ve scéně fyzická těla, aby spolu mohly navzájem ve scéně kolidovat, nebo aby na ně působila gravitace. Fyzické tělo je opět objekt, který se váže na node a přejímá například jeho pozici ve scéně, ale sám má své specifické vlastnosti jako je třeba hmotnost. Při vytváření fyzického těla je nutné zvolit tvar masky, přičemž lze vybrat z tvaru kruhu, obdélníku, polygonu nebo nastavit vlastní tvar pomocí bodů (viz obrázek 4.4).

Obrázek 4.4: *Individuální fyzická maska aplikovaná na node.*



Zdroj: Sprite Kit dokumentace

Také je třeba ujasnit si pojmy kontakt a kolize. Zatímco kontakt je stav, při kterém se navzájem dvě těla dotýkají, kolize zajišťuje, že tyto dvě těla na sebe budou při kontaktu působit a neprojdou skrz sebe, aniž by nedošlo například k odrazu[1].

4.4 Editor emitoru částic

Je WYSIWYG editor implementovaný přímo do Xcode (viz obrázek 4.5), není tedy třeba přepínat z IDE do jiného programu a zpět. Samotný editor je velmi přehledný, flexibilní a dovoluje nastavit celou řadu parametrů daného emitoru částic. Emitor je jednoduše zdroj částic, který má své specifické parametry jako počet vytvořených částic, jejich životnost, směr nebo třeba velikost, které jsou navíc dále programovatelné. Pomocí emitoru lze simulovat např. kouř, déšť, sníh, oheň a mnoho dalších.

Obrázek 4.5: *Particle Emitter Editor - Editor emitoru částic v Xcode*

[1]Příklad: Postavička běží po zemi a sbírá mince. Koliduje se zemí, ale s mincemi přichází pouze do kontaktu.

4.5 Přechod mezi scénami

Většina her sestává z více scén. Jednotlivé úrovně hry mohou být samostatnými scénami, herní menu nebo nastavení hry také. Pro přechod mezi scénami nabízí Sprite Kit třídu *SKTransition*, která provádí animovaný přechod ze stávající scény do nově volané. Ve výchozím nastavení jsou obě scény během tohoto přechodu pozastaveny. Přechody můžou mít podobu otevírání dveří do nové scény, vyblednutí stávající nebo posun či vytlačení scény.

Kapitola 5

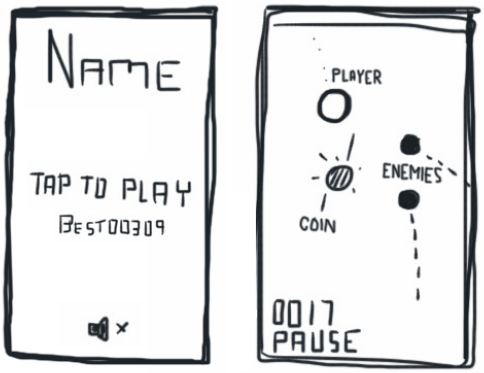
Hra SpaceOne

V této kapitole popíši mnou vytvořenou hru, která v průběhu vývoje dostala oficiální název SpaceOne, pod kterým je v současnosti i publikována v App Store, prezentována na sociálních sítích a jejíž název nese i oficiální webová stránka hry. Hra byla vyvíjena ve vývojovém prostředí Xcode 5 převážně v jazyce Objective-C, pomocí frameworku Sprite Kit. Je spustitelná na všech chytrých telefonech iPhone a všech tabletech iPad s verzí systému iOS 7.0 a vyšší. Minimální verze systému pro spuštění hry je iOS 7.0 z důvodů popsaných v kapitole 3.4. Pro přehlednost budou v této kapitole popsány pouze nejdůležitější části kódu, kompletní kód hry je potom součástí přiloženého CD.

5.1 Koncept hry

SpaceOne je 2D minimalistická indie hra, ve které se snažíte posbírat co nejvíce mincí. Během sbírání je nutné vyhýbat se všem nepřátelským objektům, které hráčovo těleso přitahuje. Pokud je posbíraných mincí dost a žádný nepřátelský objekt vás nezasáhl, hodnota sbíraných mincí se začne více a více násobit.

Obrázek 5.1: *Náčrt scén - vize*



Hru tvoří pouze jeden rámec, ve kterém se podle hráčovi interakce mění scény. Původní verzi hry tvořily pouze dvě scény (viz obrázek 5.1). První *menu*, kterou vidí hráč vždy při spuštění aplikace a do které se vrací. Druhá *game*, je scéna se samotnou hrou. V první aktualizaci ke stávajícím dvěma přibyla ještě třetí *stats*, scéna se statistikami.

Hra se skládá z 8 tříd (viz příloha G), z nichž první tři jsou třídy reprezentující každá jednu scénu a jsou si svojí počáteční konstrukcí podobné. Všechny tři obsahují například konstruktor *initWithSize:*, který je volán při vytvoření scény a konkrétně v naší hře vytváří ve scéně tlačítka, herní objekty, logo, nebo třeba pozadí hry. Další tři jsou pomocné třídy přidané později, především rozšiřující funkce hry. Naopak poslední dvě třídy jsou součástí každého Sprite Kit projektu, jsou tedy pro vývojáře připraveny.

Hra obsahuje tyto třídy:

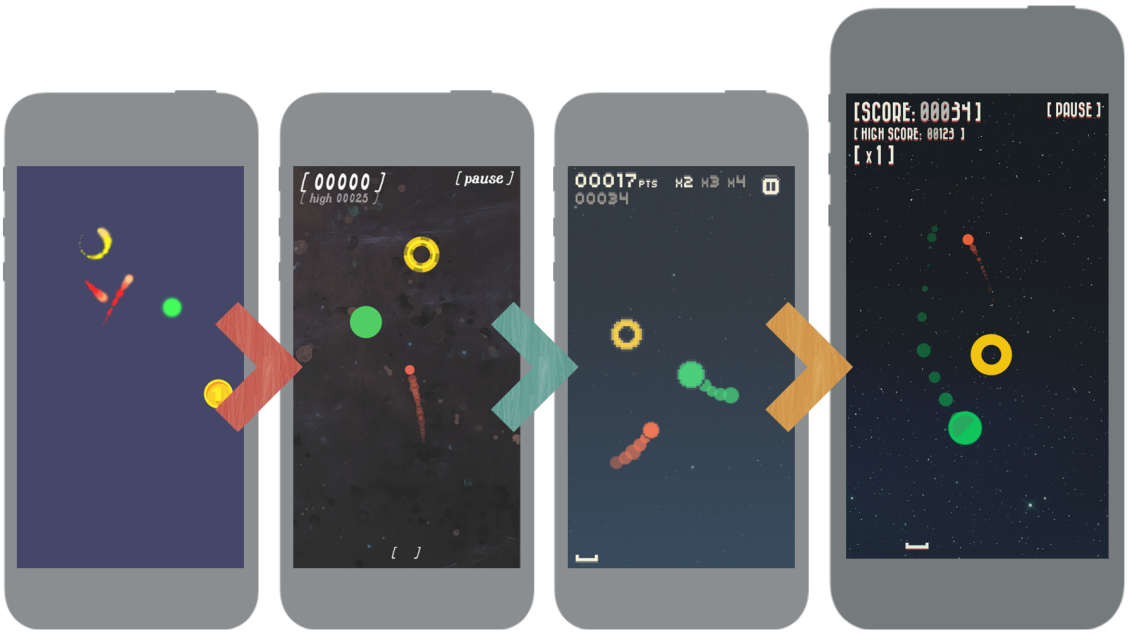
- *SOMenuScene* - Třída menu scény.
- *SOGameScene* - Třída scény samotné hry.
- *SOStatsScene* - Třída scény se statistikami.
- *SKEmitterNode* - Pomocná třída pro vkládání emitorů částic do scén. Jedná se o soubory s příponou *.sks*, které mají nadefinované různé emitování částic.
- *SOCloudSync* - Pomocná třída pro práci se službou iCloud jež je určena pro synchronizaci dat mezi Apple zařízeními. Obsahuje například metody *saveToCloud* a *loadFromCloud*.
- *Reachability* - Pomocná třída pro zjišťování dostupnosti internetového připojení v zařízení, jejíž součástí je například metoda *currentReachabilityStatus*, která vyhodnotí zda a případně přes co je zařízení připojeno k internetu a následně nám vrátí jednoduše číslo odpovídající tomuto stavu.
- *SOAppDelegate* - Delegát je odpovědný za provádění určitých operací pro jiné objekty. Tedy umožňuje provádět určité operace v určitých předefinovaných okamžicích. Pokud je hra například minimalizována, delegát volá metodu *applicationDidEnterBackground:*, jejíž součástí je kód na zveřejnění notifikace. Na několika místech v programu na tuto notifikaci hra reaguje, v tomto případě zastavením hry a vyvoláním tlačítek pauzy.

- *SOViewController* - Třída řízení, odpovědná za správu rámce hry. Spojuje model a vzhled z architektury MVC. Představuje logickou součást aplikace, která rozhoduje o tom, jak aplikace naloží s uživatelovým vstupem (Mark a LaMarche, 2010). Obsahuje například metodu *viewDidLoad*, která je volána vždy při spuštění hry. V našem případě spouští tato metoda mimo jiné i globální hudbu hry, tedy hudba je pro všechny scény společná.

5.2 Vzhled hry

Během vývoje prošel vzhled hry několika úpravami (viz obrázek 5.2, z něhož snímek ze hry úplně vpravo reprezentuje současný vzhled hry). Na počátku jsem vytvořil čistě dočasnou pracovní grafiku hry, současná grafika je tvořena částečně pastelovými barvami, retro vesmírem s retro vesmírným písmem a jednoduchými tvary objektů.

Obrázek 5.2: *Vývoj grafiky hry SpaceOne*



Z důvodu šetření operační paměti a velikosti výsledné aplikace jsou všechny obrázky hry ukládané ve formátu *PNG* ve 24-bitové barevné hloubce. Každý z obrázků

navíc prošel následnou úpravou v aplikaci imageAlpha, která pomocí ztrátové komprese snížila velikost obrázkového souboru a v aplikaci imageOptim, která pomocí několika dalších algoritmů opět tuto velikost dokázala snížit. Výsledkem bylo snížení velikosti souborů celé herní grafiky přibližně o 300%, aniž by došlo k viditelnému zhoršení kvality.

5.2.1 Menu scéna

Po spuštění hry se v rámci aplikace objeví jako první scéna s menu. To zajišťuje zdrojový kód 5.1 v již zmíněné metodě *viewDidLoad* třídy *SOViewController*.

Obrázek 5.3: *Menu hry na iPhonu a iPadu*

Scéna obsahuje (viz obrázek 5.3) logo hry, které je animované díky vytvoření

několika vrstev a naprogramovanému pohybu, nejedná se tedy o již animovaný obrázek. Nejvíce by hráče měl upoutat větší nápis umístěný ve středu displeje, ten totiž říká hráči, že pro začátek hry se stačí pouze dotknout displeje kdekoliv mimo ostatní tlačítka. Pod tímto nápisem je ještě jeden menší, který hráče informuje o jeho aktuální hodnosti. V první aktualizaci jsem přidal do hry systém hodností, podobný obdobným systémům sbírání zkušeností v jiných hrách, díky kterému může hráč ve hře získávat různé odměny, což by hráče mělo více motivovat. Ve spodní části scény se pak zleva nachází tlačítka nastavení, odměny, statistik a barevné tlačítko služby Game Center.

Zdrojový kód 5.1: *Počáteční scéna v rámci hry*

```
// Nastaveni ramce
SKView *skView = (SKView *)self.view;
// Vytvoreni a nastaveni sceny
SKScene *scene = [SOMenuScene sceneWithSize:skView.bounds.size];
scene.scaleMode = SKSceneScaleModeAspectFill;
[skView presentScene:scene];
```

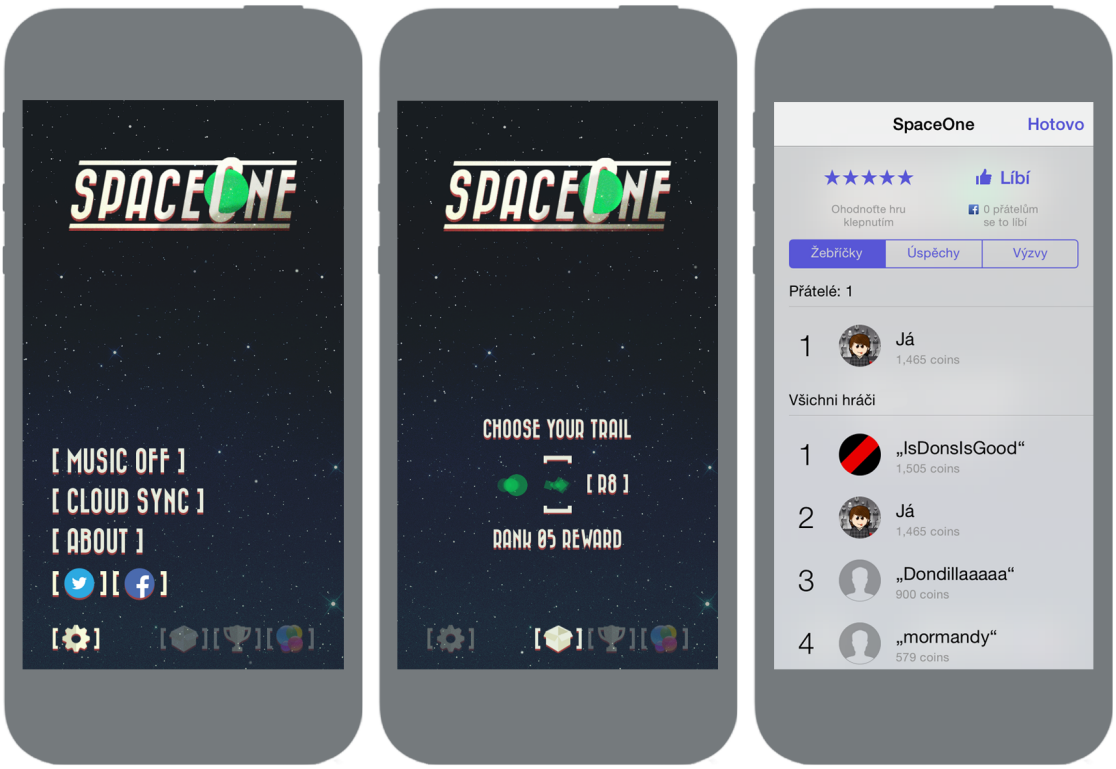
Z tohoto menu se tedy hráč může kromě samotné hry dostat také do nastavení (viz první snímek zleva obrázku 5.4), do výběru odměn (prostřední snímek obrázku) a do služby Game Center (poslední snímek obrázku).

Samotná tlačítka jsem ale jako tlačítka v kódu nedefinoval. Opět z důvodu šetření velikosti výsledné hry a jednoduššího kódu jsem pouze definoval v metodě *touchesEnded:*[1] bod *CGPoint location*, který přebírá vždy pozici doteku ve scéně. Následně pomocí podmínek pouze zjišťuji, jestli se na pozici tohoto bodu nachází nějaký node, pokud ano, podmínka je splněna a vyvolá určitou akci, jako právě zobrazení tlačítek nastavení po dotknutí se ozubeného kolečka a podobně. Pokud se pak hráč dotkne místa, které neobsahuje node žádný, hra se přesouvá pomocí přechodu do scény samotné hry (viz zdrojový kód 5.2)[2]. Ušetření místa spočívá v tom, že na dotek může reagovat prakticky jakýkoliv node, třeba i pouhý kus textu[3] a nebylo tak třeba tvořit tolik obrázků, které by nejen že navýšily velikost hry, ale zabíraly

[1]Metoda která se volá pokaždé když se hráč přestane dotýkat displeje zařízení.
[2]Celý program, stejně tak i podmínky jsou složitější, pro zjednodušení je zde pouze část metody.
[3]Tlačítko pause v herní scéně je pouze text.

by i větší místo v operační paměti.

Obrázek 5.4: *Snímky hry po stisknutí spodních tlačítek*



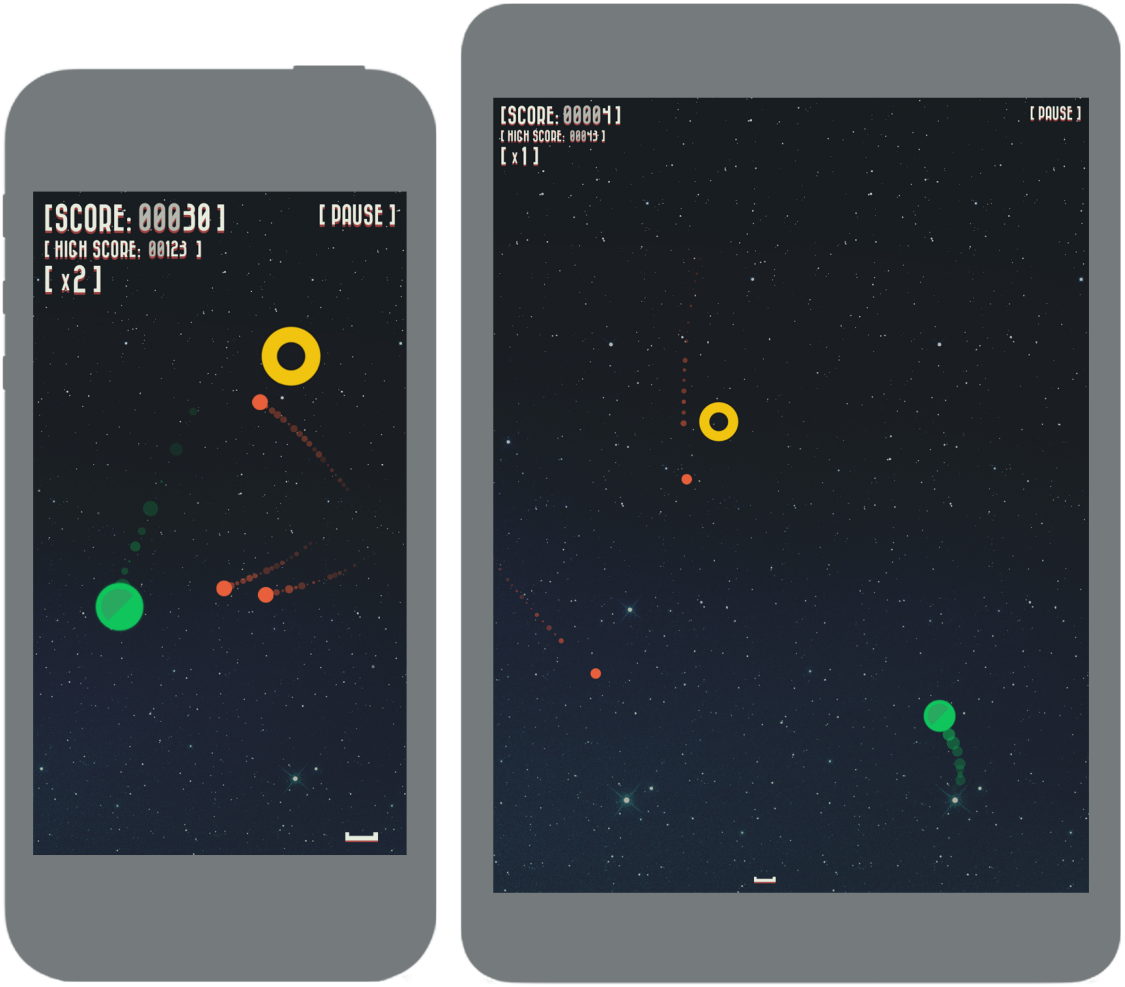
Zdrojový kód 5.2: *Část metody touchesEnded:*

```
// Ulozeni dotyku jako CGPoint, podle ktereho budu porovnavat
CGPoint location = [touch locationInNode:self];
if ([_gameCenter containsPoint:location] && !_twitter && !_facebook &&
    _gameCenter.alpha == 1.0f) {
    [self performSelector:@selector(gameCenterBtnPressed) withObject:nil];
else {
    if (!_twitter && !_facebook && !_saveCloudBtn && !_loadCloudBtn && !
    _multipleLineText && _settings.alpha == 1.0f && _rewards.alpha == 1.0f) {
        SOGameScene *game = [[SOGameScene alloc] initWithSize:self.size];
        [self.view presentScene:game transition:[SKTransition pushWithDirection:
    SKTransitionDirectionLeft duration:0.7]];
    }
}
```

5.2.2 Herní scéna

Nejdůležitější a nejobsáhlejší scénou je samotná herní scéna. V okamžiku, kdy hra začne, se ve scéně objeví několik nodů. Nejdůležitější z nich je samotný hráč, který je reprezentován zeleným kolečkem, které uživatel pomocí prstu posouvá po scéně (viz obrázek 5.5).

Obrázek 5.5: *Probíhající hra na iPhonu a iPadu*



Metoda *initWithSize:* vytvoří hráče uprostřed displeje a přidá ho do scény (viz zdrojový kód 5.3). Tento kód však neobsahuje klasické volání metody *basicPlayerMode*, ale pošle objektu zprávu s odpovídajícím selektorem. Pokud bychom například až po kompilaci kódu skládali název metody, byl by selektor nutností, v tomto případě na tom však nezáleží.

Zdrojový kód 5.3: *Část metody initWithSize: v herní scéně*

```
_player = [SKNode node];
// Pocatecni nastaveni nodu
[self performSelector:@selector(basicPlayerMode) withObject:nil];
// Pocatecni pozice hrace
_player.position = CGPointMake(self.size.width/2, self.size.height/2);
// Hrac do sceny
[self addChild:_player];
```

Zbytek hráče jsem definoval v metodě *basicPlayerMode* kvůli odlehčení metody *initWithSize:* a nutnosti volat tuto část kódu i v průběhu hry. Metoda *basicPlayerMode* tak dotváří samotného hráče tím, že pod něho například jako potomka vkládá obrázek zeleného kolečka a definuje jeho fyzické tělo (viz zdrojový kód 5.4).

Zdrojový kód 5.4: *Část metody basicPlayerMode*

```
SKSpriteNode *visage = [SKSpriteNode spriteNodeWithImageNamed:@"player.png"];
// Pozice v nodu
visage.position = CGPointZero;
// Odstraneni vsech potomku, pokud nejaci zbyli
[_player removeAllChildren];
// Pozice v prostoru, cili hloubka
visage.zPosition = characterVisageLevel;
[_player addChild:visage];
// Implementace fyziky na hrace
_player.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:22.f];
_player.physicsBody.mass = 9999;
// Kolizni masky
_player.physicsBody.categoryBitMask = playerCategory;
_player.physicsBody.contactTestBitMask = enemyCategory | coinCategory;
```

Pokud například dojde ve hře ke střetu hráče a nepřátelského objektu při kterém není hráč zničen, stává se na malou chvíli nesmrtelným. Nodu *_player* je poslána zpráva se selektorem *immortalPlayerMode*, jeho kolizní masky se proto změní a fyzické tělo tak na nepřátelské objekty nebude reagovat (viz zdrojový kód 5.5). Po krátké době se hráč stává opět zranitelný, čili se opět nodu *_player* posílá zpráva se selektorem *basicPlayerMode*.

Zdrojový kód 5.5: *Metoda immortalPlayerMode*

```
// Problikavani hrace jako symbol nesmrtelnosti + zmena koliznich masek
[_player runAction:FadeAlpha(0.5, 0.5, 2)];
_player.physicsBody.categoryBitMask = immortalCategory;
_player.physicsBody.contactTestBitMask = coinCategory;
```

To s jakými kategoriemi bude konkrétní node kolidovat určuje vlastnost *contactTestBitMask*. Naopak vlastnost *categoryBitMask* určí do jaké kategorie node patří.

Dalším nodem ve scéně je nepřítel, kterého reprezentuje červené kolečko. Tento objekt se ve scéně objeví na začátku hry a každých deset vteřin se vytvoří další v pravém nebo levém spodním rohu, to záleží na pozici pohybující se závorky.

Zdrojový kód 5.6: *Část metody update:*

```
for(SKNode *enemyNode in _enemies) {
    // Pozice konkretniho nepritele z pole nepratel
    CGPoint enemyPosition = enemyNode.position;
    // Rozdil mezi souradnicemi hrace a nepritele
    CGVector difference = VectorMinus(playerPosition, enemyPosition);
    // Normovany vektor
    CGVector normalized = VectorUnit(difference);
    // Nastaveni smeru a velikosti sily pusobici na teleso.
    // Obtiznost: iPad = 2 (vetsi displej = vetsi a rychlost), iPhone = 1)
    CGVector force = VectorMultiply(normalized, _difficulty);
    [enemyNode.physicsBody applyForce:force];
}
```

Úkolem nepřátel je zasáhnout hráče, tedy podstatnou částí těchto nodů je kód, který určuje jejich pohyb. Bylo třeba navrhnout algoritmus, který na základě pozice hráče a nepřítele vypočítá pozici nepřítele v dalším snímku a tedy jeho pohyb. Na fyzické tělo nepřátelského nodu je třeba působit silou, k tomu jsem využil metodu *applyForce:*. Tato síla se musí měnit v závislosti na pozicích obou nodů, kód na výpočet působící síly na těleso nepřítele jsem proto umístil do metody *update:*, ta je volána při každém novém vyhodnocování snímku aplikace.

Samotný kód na výpočet směru síly jsem vytvořil ze 4 základních funkcí, které jsem definoval v souboru *Functions.h*, který jsem založil převážně pro pomocné

funkce celé aplikace. Prací s vektory jsem pak určil výslednou sílu, která působí na těleso s určitou velikostí určitým směrem (viz zdrojový kód 5.6).

Výpočet směru síly se skládá z těchto funkcí

- *VectorMinus* - Pomocná funkce pro zjištění rozdílu mezi souřadnicemi obou nodů (viz zdrojový kód 5.7).
- *VectorLength* - Pomocná funkce pro zjištění délky neboli normy. Využil jsem Pythagorovu větu pro výpočet odvěsny (viz zdrojový kód 5.8).
- *VectorUnit* - Protože potřebujeme znát hlavně směr, vytvořil jsem normovaný vektor pomocí dělení původního vektoru jeho normou (viz zdrojový kód 5.9).
- *VectorMultiply* - Pomocná funkce pro násobení vektoru (viz zdrojový kód 5.10).

Zdrojový kód 5.7: *Funkce VectorMinus*

```
static inline CGVector VectorMinus(CGPoint p1, CGPoint p2)
{
    return CGVectorMake(p1.x - p2.x, p1.y - p2.y);
}
```

Zdrojový kód 5.8: *Funkce VectorLength*

```
static inline CGFloat VectorLength(CGVector vector)
{
    return sqrtf(vector.dx * vector.dx + vector.dy * vector.dy);
}
```

Zdrojový kód 5.9: *Funkce VectorUnit*

```
static inline CGVector VectorUnit(CGVector vector)
{
    CGFloat invertedLength = 1.0 / VectorLength(vector);
    return VectorMultiply(vector, invertedLength);
}
```

Zdrojový kód 5.10: *Funkce VectorMultiply*

```
static inline CGVector VectorMultiply(CGVector vector, CGFloat m)
```

```
{
    return CGVectorMake(vector.dx * m, vector.dy * m);
}
```

Všechny nepřátelská tělesa jsou nyní přitahována k tělesu hráče, který se jim v rámci hry snaží vyhýbat.

Posledním důležitým nodem je žlutá mince. Po kontaktu s hráčem je její hodnota přičtena do aktuálního hráčova skóre, které je v levém horním rohu scény. Její hodnota je vždy jedna a ta se následně násobí číslem jedna až čtyři, podle počtu nasbíraných mincí od poslední srážky s nepřítelem. Pokud je násobič větší než jedna, hráč po srážce s nepřítelem nekončí, pouze se mu násobení hodnoty mincí sníží na číslo jedna.

Obrázek 5.6: *Snímky určitých fází hry*

Ve scéně jsem ještě v pravém horním rohu vytvořil tlačítko pauzy, které dokáže hru přerušit a vyvolat tři další tlačítka (viz prostřední snímek obrázku 5.6), k pauze dochází například i při minimalizaci samotné aplikace. Hráč tak po pauze hry může

ve hře pokračovat, což před zrušením pauzy vyvolá krátký odpočet a hráč má čas se připravit, restartovat hru nebo se může přemístit do menu scény.

Pokud hráč nepřežije zásah od nepřátelského tělesa, je vyvolán konec hry s explozí obou střetnutých těles a přehled nahraných bodů. Z tohoto přehledu může hráč okamžitě začít hrát znovu nebo se opět přemístit do menu scény (viz první snímek zprava obrázku 5.6).

Důležitou funkcí je také krátké vysvětlení principu hry. Tento návod se spustí pouze pokud se jedná o první hráčovu hru a hra je při něm zastavována a zobrazuje stručné vysvětlivky se zvýrazněnými nody (viz první snímek zleva obrázku 5.6).

5.2.3 Scéna se statistikami

Z menu scény se může ještě hráč přesunout pomocí tlačítka se symbolem trofeje do scény se statistikami (viz obrázek 5.7).

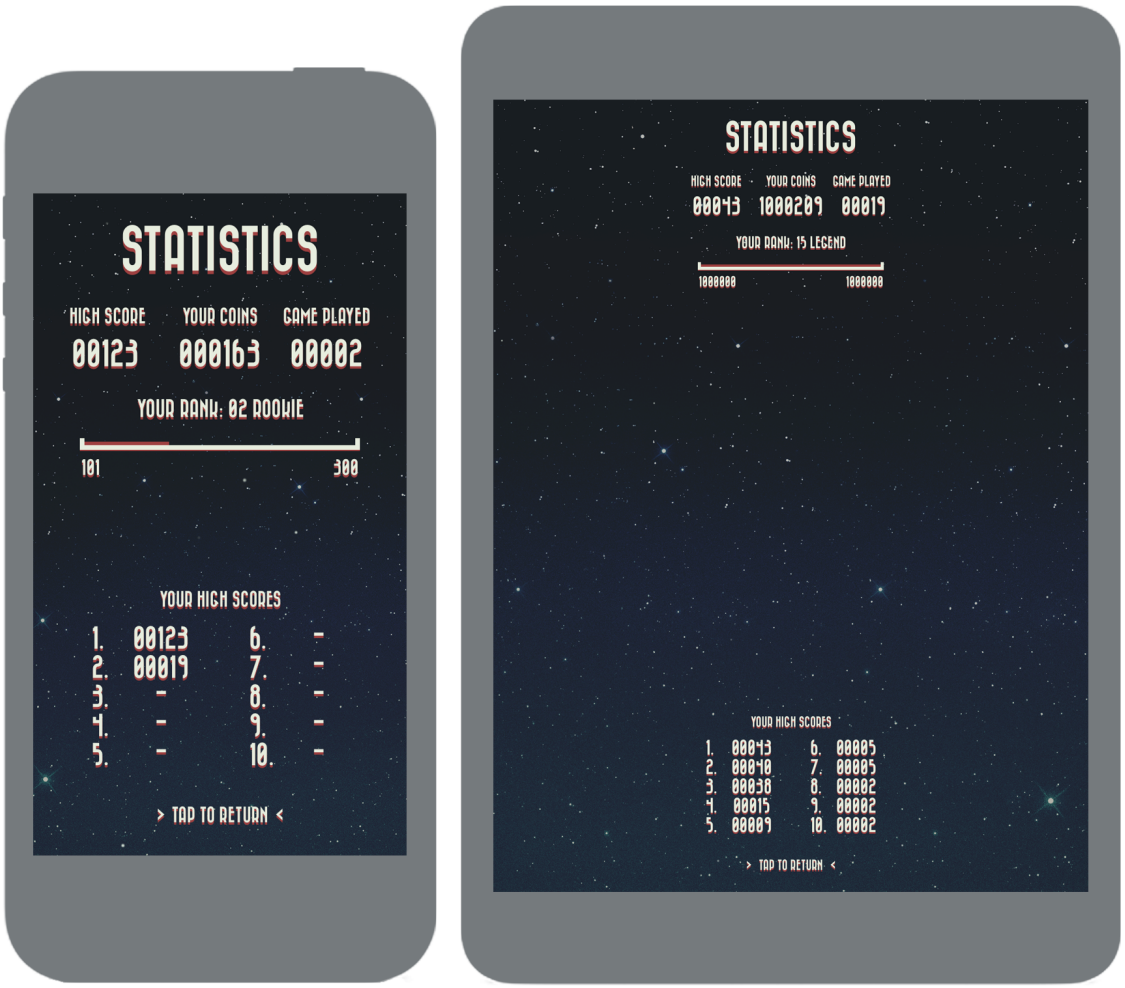
Zdrojový kód 5.11: *Část metody initWithSize: scény statsScene*

```
// DATA
int totalScore = (int)[[NSUserDefaults standardUserDefaults] integerForKey:@"
    TotalScore"];
// Panel hodnosti
switch (totalScore) {
    case 0 ... 100:
            _rank = MakeStringWithShadow(@"YOUR RANK: 01 NEWBIE", 16, 2, 0);
            _expMin = 0;
            _expMax = 100;
            break;
    case 101 ... 300:
            _rank = MakeStringWithShadow(@"YOUR RANK: 02 ROOKIE", 16, 2, 0);
            _expMin = 101;
            _expMax = 300;
            break;
}
```

V této scéně se nachází hodnoty nejvyššího dosaženého skóre ve hře, celkový počet nasbíraných mincí a počet odehraných her. Právě druhé zmíněné číslo má vliv

na systém hodností, kdy se hráčova hodnost zvyšuje s jeho celkovým počtem mincí.

Obrázek 5.7: *Statistiky na iPhonu a iPadu*



Pod těmito třemi hodnotami se nachází samotná hráčova hodnost a graficky znázorněný stav mincí v rozmezí, po jehož překonání se hráčova hodnost zvýší. Celkový počet hráčových mincí je po navýšení z odehrané hry vždy uložen pomocí třídy *NSUserDefaults* jako uživatelská předvolba, která má svůj klíč a hodnotu. Uživatelské předvolby mají tu výhodu, že k nim lze přistupovat kdekoliv v aplikaci, nemažou se po ukončení aplikace a lze je synchronizovat se službou iCloud. Po přechodu do scény se statistikami se tedy načte hodnota celkového počtu mincí a pomocí příkazu *switch* je vybrána správná hodnost (viz zdrojový kód 5.11).

Ve spodní části scény se nachází přehled deseti nejlepších hráčových skóre. Aby se všech deset hodnot vešlo na menší displeje, rozdělil jsem je do dvou sloupečků.

5.3 Data hry

Hráčův vstup v podobě hraní hry produkuje několik výstupů. Především se jedná o nahraný počet bodů, dále také o získané hodnoty popsané v podpodkapitole 5.2.3. Za data se dá považovat i současné nastavení hry.

Data nahraných bodů jsem propojil se službou Game Center a aby hráč žádná data a celý postup ve hře neztratil, přidal jsem do hry možnost synchronizovat data pomocí služby iCloud.

5.3.1 Game Center

Obrázek 5.8: *Game Center*

Hráč nyní mohl dosáhnout určitého skóre ve hře, ale zatím chyběl jakýkoliv žebříček, kde by své skóre mohl porovnat s ostatními. Proto jsem do hry implementoval službu Game Center, kterou nabízí Apple přímo za tímto účelem vývojářům. Jedná se o herní sociální síť, která sdružuje nahrané výsledky a získané úspěchy z her, které tuto službu implementují. Hráč si pak může u těchto her prohlédnout žebříčky a získá možnost přidávat si přátele, se kterými může nahrané skóre porovnávat nebo je vyzvat k jeho překonání.

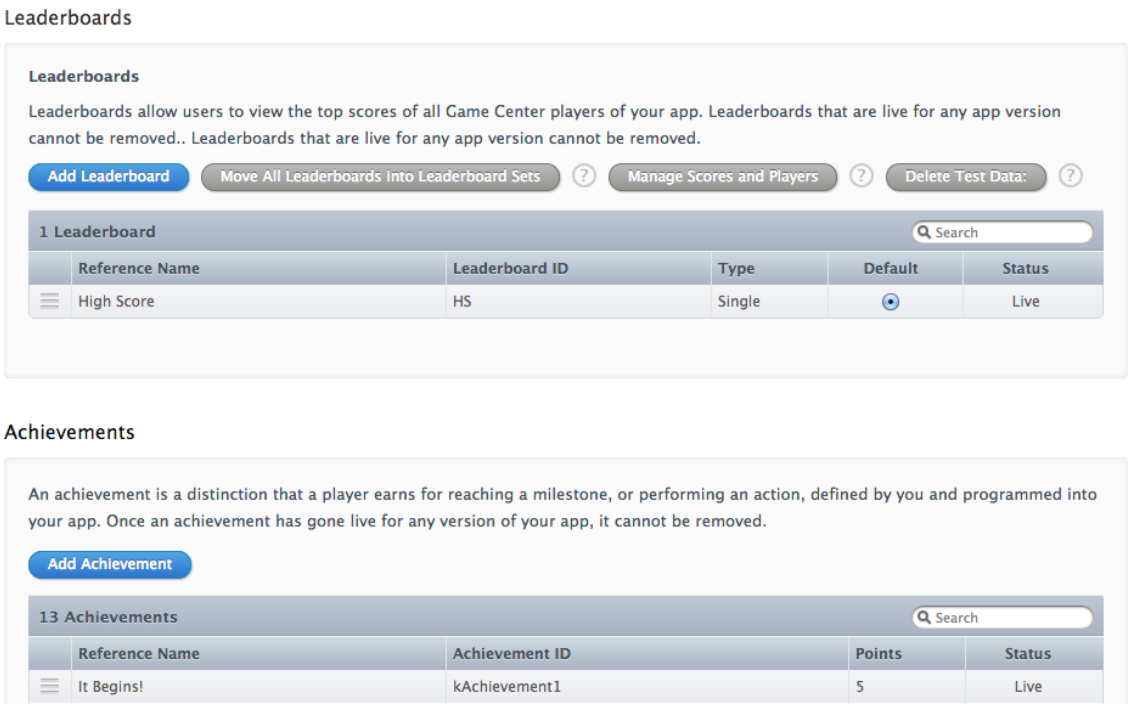
Začal jsem tedy tím, že jsem do projektu implementoval *GameKit* framework. Nyní jsem mohl plně využívat všechny potřebné třídy tohoto frameworku a začít tvořit své metody. Začal jsem metodou, která autentizovala samotného hráče. Metoda má název *authenticateLocalPlayer* a je volána při spuštění hry v metodě *viewDidLoad* třídy *SOViewController*.

Dále bylo třeba vytvořit pro aplikaci žebříček, do kterého se nahrané skóre bude odesílat. To však již nešlo řešit programově a bylo proto třeba přihlásit se do webového prostředí služby iTunes Connect, která je určena vývojářům a slouží hlavně ke správě vývojářových aplikací. Aby bylo navíc možné Game Center pro hru nastavit, musel jsem již hru ve službě iTunes Connect vytvořit. Nebylo však zatím potřeba nahrávat binární soubor hry, stačilo pouze vytvořit identifikátor aplikace a projekt

se základními informacemi jako je název hry, její popis, ikonka a podobně.

Když už byla hra ve službě iTunes Connect vytvořena, bylo možné pod ní nastavit i službu Game Center (viz obrázek 5.9). Nastavení probíhá stylem klikání a vývojáři stačí pouze vybírat z roletových menu a zaškrtávat požadované možnosti. Vytvořil jsem tedy základní žebříček *High Score*, který jednoduše vypisuje nejlepší dosažené skóre každého hráče, který tuto službu využívá. Rovněž jsem ve službě vytvořil několik úspěchů tzv. *Achievementů*, které hráč může odemknout a dostat za ně již zmíněné body do služby Game Center.

Obrázek 5.9: *Správa služby Game Center ve webovém prohlížeči*



Nyní stačilo pouze vytvořit metodu *reportScore:*, která nahrává celočíselnou hodnotu do konkrétního žebříčku (viz zdrojový kód 5.13). Jedná se o standardní postup, který Apple popisuje ve své dokumentaci.

Zdrojový kód 5.12: *Část metody gameOverPanel*

```
[self reportScore:_score forLeaderboardID:@"HS"];
```

Na konci každé hry je pak metoda *reportScore:* volána s dvěma argumenty. Prvním je celočíselná hodnota nahraného skóre *_score* a druhým je identifikátor žebříč-

ku *HS* (viz zdrojový kód 5.12). Pokud byl hráč v době odesílání hodnoty autentizován a hodnota *_score* byla největší dosažená, v žebříčku se tato hodnota s hráčovou přezdívkou objeví. Poslední mnou vytvořená metoda *checkAchievements* je volána při každém sebrání mince a kontroluje, zda došlo ke splnění některého z vytvořených úspěchů.

Zdrojový kód 5.13: *Metoda reportScore:*

```
- (void)reportScore: (int64_t)score forLeaderboardID: (NSString*)identifier
{
    GKScore *scoreReporter = [[GKScore alloc] initWithLeaderboardIdentifier:
    identifier];
    scoreReporter.value = score;
    scoreReporter.context = 0;
    NSArray *scores = @[scoreReporter];
    [GKScore reportScores:scores withCompletionHandler:^(NSError *error) {
        NSLog(@"Score sent to the Game Center");}];
}
```

5.3.2 iCloud

Obrázek 5.10: *iCloud*

Pro synchronizace dat jsem využil službu iCloud, která ukládá data na servery společnosti Apple. Jednoduše se implementuje a má k ní přístup každý uživatel systému iOS.

V nastavení hry jsem vytvořil tlačítko *CloudSync* po jehož stisknutí se zobrazí tlačítko *Save to Cloud*, *Load from Cloud*, *Reset game data* a datum poslední zálohy. Při každém zobrazení těchto tlačítek dochází ke kontrole dostupnosti internetového připojení pomocí metod třídy *Reachability* a načtení informací o poslední záloze. Pokud není zařízení připojeno k internetu, zálohu není možné nahrát a to ani v případě, že internetové připojení bylo ztraceno po načtení této nabídky, tedy kdykoliv v průběhu prohlížení. Předmětem zálohy jsou všechna data uložená v uživatelských předvolbách *NSUserDefaults*. Jedná se tedy o statistiky, hodnosti, nahraná skóre a nastavení hry.

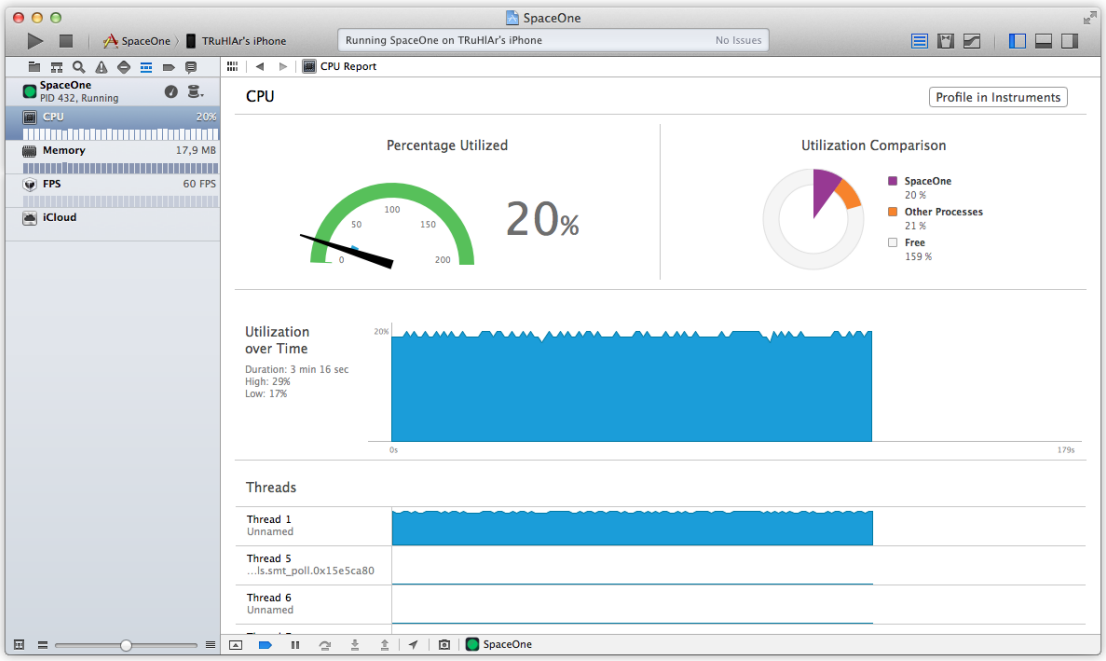
5.4 Ladění hry

Podstatnou částí dokončování hry, bylo její ladění. Během tohoto procesu jsem sledoval převážně zatížení procesoru, množství zabrané operační paměti, úniky paměti a snímkovou frekvenci. Samotné prostředí Xcode nabízí velké množství nástrojů a způsobů, jak se k těmto datům dostat a tím klasickým je přemístit se do ladícího okna, vybrat zařízení popřípadě simulátor na kterém má aplikace běžet a nechat aplikaci sestavit. Sestavená aplikace je následně na vybraném zařízení spuštěna.

V průběhu psaní kódu jsem postupně opravil všechny chyby, Xcode proto s kompilací neměl žádný problém. Hru jsem zpočátku ladil pouze na simulátorech, po zaplacení vývojářského programu pak převážně na iPhonu 5. Takové ladění bylo přínosnější, protože jsem se dozvěděl, jak hra zatěžuje skutečné zařízení.

Prvním zobrazeným měřidlem je vytíženost CPU. V průběhu hry se procentuální hodnota využití procesoru pohybovala vždy kolem 25% (viz obrázek 5.11). K žádným výkyvům zpravidla nedocházelo.

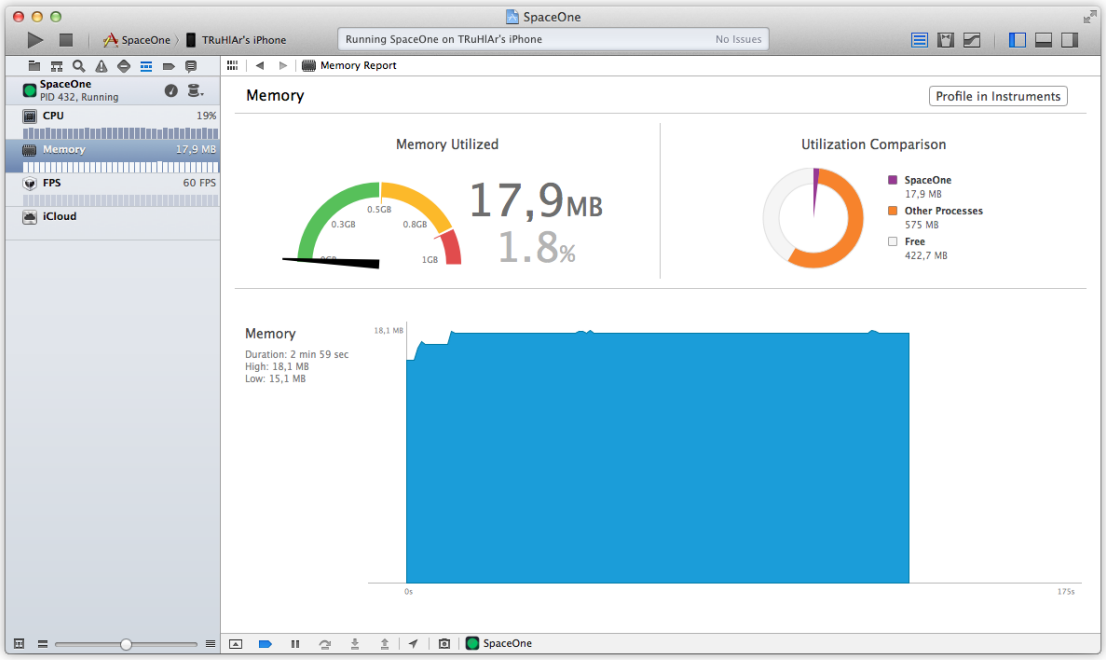
Obrázek 5.11: *Vytíženost procesoru po spuštění hry na iPhonu 5*



Druhým měřidlem je operační paměť. Po spuštění jí hra zabírá okolo 2% (viz obrázek 5.12), s načtením dalších dat, jako jsou textury, se zabraná paměť dále

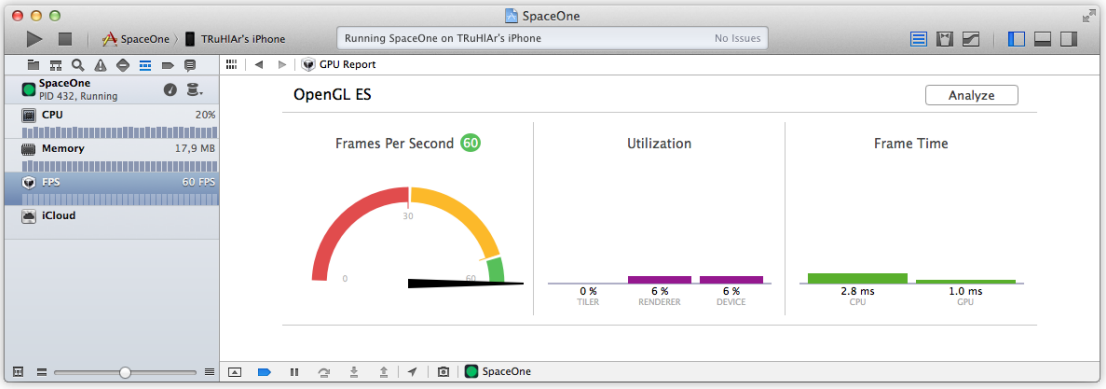
navýší. Průměrně tak dochází k obsazení přibližně 100MB operační paměti, což odpovídá 10% celkové kapacity iPhonu 5. Podrobnější data jako jsou úniky paměti lze najít pod tlačítkem *Profile in Instruments*, k žádným však nedocházelo.

Obrázek 5.12: *Zaplnění operační paměti po spuštění hry na iPhonu 5*



Třetím měřidlem je snímková frekvence. Jak je vidět na obrázku 5.13, spuštěná hra je vykreslována rychlostí 60 FPS, tedy rychlostí maximální možnou. Po čas hraní hry jsem navíc žádný pokles nezaznamenal.

Obrázek 5.13: *Rychlost vykreslování hry na iPhonu 5*



Kapitola 6

Publikování hry v App Store

Obrázek 6.1: *App Store*

Aby bylo vůbec možné nahrát první verzi hry do App Storu, bylo zapotřebí projít vcelku dlouhým procesem. V první řadě se dbá na bezpečnost, a proto se vývojář napříč celým procesem musí například prokazovat ověřenými a podepsanými certifikáty. Především je potřeba mít aktivní tedy zaplacený program vývojáře pro iOS.

Dalším krokem bylo získat vlastní certifikát vývojáře pro možnost sestavovat a testovat hru na zařízení s iOS, v mém případě na iPhonu 5. Certifikát si lze vyžádat ve webové aplikaci určené pro vývojáře na základě žádosti o podepsání certifikátu. Tu jsem si vytvořil na mém počítači, během procesu je generován a připojen můj privátní klíč, jméno a email, certifikační autoritou je samozřejmě Apple. Následně jsem již zmíněný iPhone přidal mezi zařízení, na kterých budu moci sestavovat aplikace.

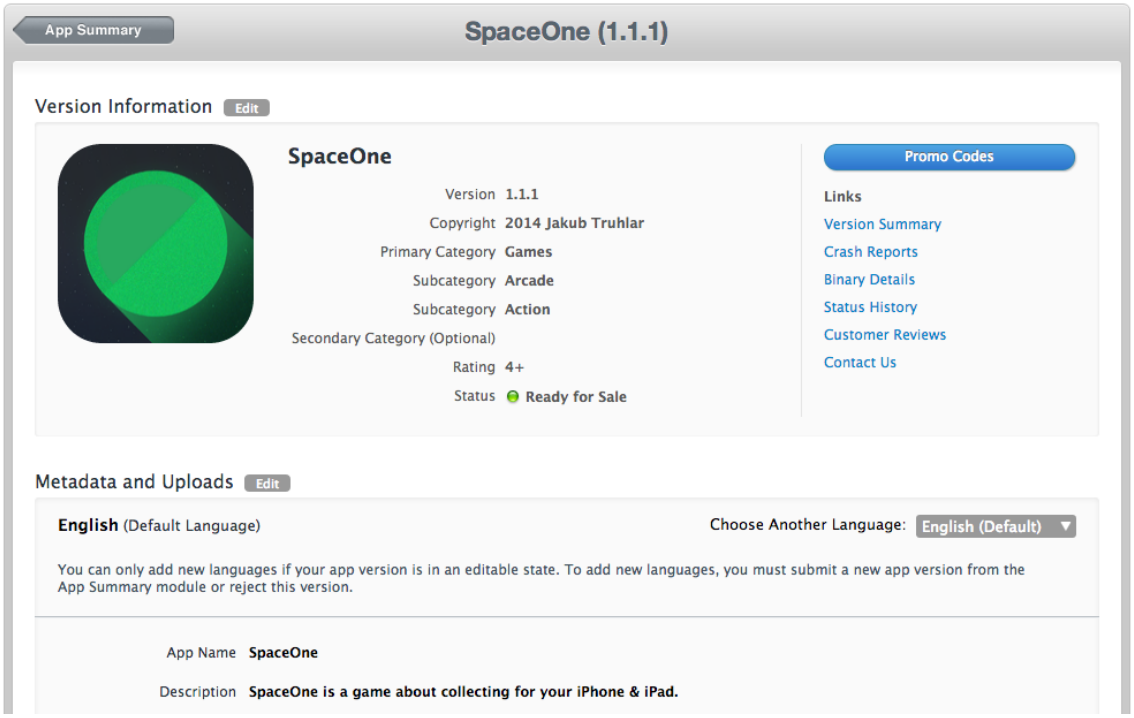
Na závěr jsem projekt v Xcode propojil s vytvořeným identifikátorem aplikace, který jsem si vytvořil již dříve při provizorní tvorbě hry ve službě iTunes Connect (viz podkapitola 5.3.1). Jedná se prakticky o textový řetězec sloužící k identifikaci aplikace, při využívání služeb jako jsou In-App purchase nebo Game Center.

V tuto chvíli již bylo možné začít aplikaci ladit a testovat na skutečném iOS zařízení.

6.1 Příprava aplikace

Po odladění a dokončení první verze hry jsem se rozhodl dokončit a doplnit všechny informace o hře v iTunes Connect (viz obrázek 6.2). Jedná se především o název aplikace, její popis, klíčová slova, kategorie do kterých tématicky spadá, informace o vývojáři, obrázky ze hry, ikonka a cena. Tyto informace budou později tvořit mnou publikovanou hru v App Storu. Vytvořil jsem další certifikát ve webové aplikaci pro vývojáře, tentokrát však distribuční pro App Store a nyní když bylo vše připraveno nastal čas na nahrání binárního souboru hry.

Obrázek 6.2: *Přehled informací o hře pro App Store*



To lze provést přímo z vývojového prostředí Xcode, stačí pouze sestavit projekt a pokud je vše v pořádku, vytvořit archiv s projektem. K archivu se lze dostat pomocí okna *Organizer*, ve kterém se u archivu objeví možnost *předložit pro distribuci v App Store*. Následně dojde ke kontrole archivu a odeslání hotového binárního souboru hry do Applu.

6.2 Proces schvalování

Nyní bylo nutné počkat, zda zaměstnanci Applu hru schválí. Průměrná doba, při které aplikace čeká ve frontě na kontrolu se pohybuje kolem pěti dní. Aktuální stav aplikace je pak možné sledovat v záložce *Status History* (viz obrázek 6.3), při každé změně stavu obdrží vývojář i upozornění emailem.

Obrázek 6.3: *Historie stavů aplikace v procesu schvalování*

Na konci schvalovacího procesu byla hra na první pokus schválena a o pár minut později se objevila ve všech Apple App Storech, ty jsou totiž rozděleny podle jednotlivých zemích na světě.

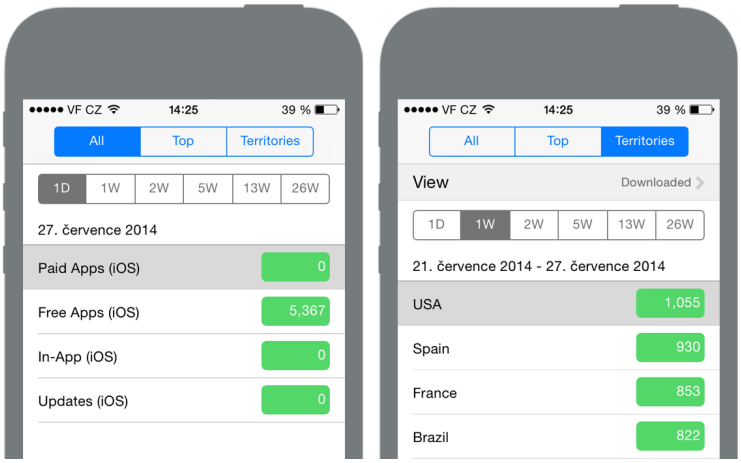
6.3 Statistiky

Prostředí iTunes Connect samozřejmě poskytuje podrobné statistiky stažení, lze je najít pod záložkou *Sales and trends* nebo v mobilní verzi pod záložkou *Trends*. Hodnoty lze následně filtrovat například podle území, typu transakce nebo typu zařízení, ze kterého byla aplikace stažena a výsledné hodnoty si lze nechat vyobrazit ve sloupcovém nebo spojnicovém grafu.

Na obrázku 6.4 je na prvním snímku celkový počet stažení mé hry za konkrétní den. Na druhém snímku je pak počet stažení hry v jednotlivých zemích za určitý

týden. Tyto hodnoty sdružují stažení všech aplikací daného účtu, ale protože pod tímto účtem je pouze hra SpaceOne, hodnoty se vztahují konkrétně k ní.

Obrázek 6.4: *Trendy a prodeje hry - mobilní verze iTunes Connect*



6.4 Verze hry

První vydaná verze v App Storu byla verze *1.0.0 build 531*. Hra již byla hratelná, ale to bylo prakticky vše. Proto jsem začal pracovat na první aktualizaci, která měla především rozšířit hru o motivační prvky (viz tabulka 6.1).

Tabulka 6.1: *Přehled verzí hry*

Verze	**Datum**	**Popis změn**
1.0.0	19. 5. 2014	· Přidána pauza hry.
1.1.0	17. 6. 2014	· Přidán systém hodností. · Přidána scéna se statistikami. · Přidáno nastavení. · Přidány odměny. · Opraveno pár drobných chyb.
1.1.1	24. 6. 2014	· Hráčův node se nyní lépe se ovládá a je animovaný. · Úspěchy se nyní v Game Centeru odemykají správně.

Kapitola 7

Závěr

V teoretické části jsem se zaměřil na zařízení, která budou hotovou hru spouštět a vytvořil jsem jejich přehled. Důraz byl přitom kladen na problematiku displejů těchto zařízení. Seznámil jsem se s rozšířenějšími frameworky pro tvorbu 2D her na systém iOS a provedl jejich srovnání v rámci vyvíjené hry. Pro vývoj jsem zvolil framework Sprite Kit a následně jsem mu věnoval celou jednu kapitolu.

Praktická část se již věnovala pouze vývoji hry SpaceOne a jejímu publikování v App Storu. První verzi hry jsem distribuoval až v pozdější fázi vývoje, stihl jsem však ve zbylém čase vytvořit a publikovat ještě další dvě verze. Nad rámec zadání jsem se totiž věnoval kromě programování i grafické, zvukové, licenční a marketingové stránce hry.

Rozhodl jsem se také celému projektu získat menší pozornost médií a vytvořit kolem hry komunitu fanoušků. Proto jsem ještě před samotným vydáním hry vytvořil tématické webové stránky a umístil je na mnou zakoupenou doménu. Stránky nastiňovaly koncept hry a sbíraly do databáze emailové adresy fanoušků, kteří chtěli být o vydání hry informováni. Díky jejich zpětné vazbě jsem navíc ještě před vydáním hry doladil některé především grafické nedostatky. Na sociálních sítích, především na Twitteru jsem pro fanoušky před i po vydání hry tvořil soutěže o kódy pro stažení hry a umisťoval jsem zde například bonusový materiál, nebo herní upoutávky.

Zhruba po třech týdnech od oznámení jsem nakonec hru publikoval v App Storu. Po dvou měsících od vydání a publikování má za sebou hra několik zahraničních

zrecenzování a do dnešního dne si jí stáhlo na 8000 uživatelů. Díky tomu se hra objevila v horních příčkách App Storu ve více než 40 zemích světa. Při současné konkurenci více než milionu dalších aplikací to je znatelný úspěch.

Co se týče samotného vývoje, kompletní kód hry se rozkládá zhruba na 4000 řádcích, projekt hry obsahuje na sto obrázkových a zvukových souborů. Přesto hra zabírá pouze 9,6 MB úložného prostoru. Součástí hry je také obsah jiných autorů, jedná se konkrétně o hudbu na pozadí hry a o použité písmo. Tento obsah jsem využil pouze v rozsahu, který schvalovala licence a jména obou autorů jsem v hotové hře uvedl.

V současné době se hra nachází ke stažení v obchodě App Store, odkud je každý den více či méně stahována, převážně díky zrecenzování na některém herním webu.

Summary and keywords

The beginning of the work describes the devices, the operating system and the frameworks for 2D iOS game development. Also the Sprite Kit framework and the common terminology are introduced.

The practical part is about development of the game itself with Sprite Kit framework in Xcode IDE. The reader is familiarized with submitting the app to the App Store and also with the approval process. At the end the thesis is aimed to parts of the web tool for developers called iTunes Connect with statistics and game versions.

KEYWORDS

Objective-C, Obj-C, Xcode, Apple, iPhone, iPad, iOS, App Store, Development, Sprite Kit, Game, App

Seznam literatury

Anthony, S. *SoC vs. CPU - The battle for the future of computing* [online]. Publikováno 2012-04-19. [cit. 2014-03-24]. Dostupné z: http://www.extremetech.com/computing/126235-soc-vs-cpu-the-battle-for-the-future-of-computing/.

Apple. *Apple WWDC 2010 Keynote Adress* [podcast]. Publikováno 2010-06-07. [cit. 2014-03-25]. Dostupné z: http://www.apple.com/apple-events/.

Apple. *Understanding cellular data networks* [online]. Publikováno 2014-07-16. [cit. 2014-08-24]. Dostupné z: http://support.apple.com/kb/ht1976.

Apple. *About Sprite Kit* [online]. Publikováno 2014-02-11. [cit. 2014-04-11]. Dostupné z: https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html.

Cocos2D-iPhone. About. In *Cocos2d-iphone.org* [online]. 2014. [cit. 2014-04-08]. Dostupné z: http://www.cocos2d-iphone.org/about/.

iClarified. *Apple's App Store Usage Numbers Place iOS 7 Adoption at 82%* [online]. Publikováno 2014-02-12. [cit. 2014-03-16]. Dostupné z: http://www.iclarified.com/38343/apples-app-store-usage-numbers-place-ios-7-adoption-at-82/.

Kochan, S. G. *Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone.* 1. vyd. Brno: Computer Press, 2010. 550 s. ISBN 978-80-251-2654-7.

Mark, D. a LaMarche, J. *iPhone SDK: průvodce vývojem aplikací pro iPhone a iPod touch.* 1. vyd. Brno: Computer Press, 2010. 480 s. ISBN 978-80-251-2820-6.

Pilone, D. a Pilone, T. *Head first iPhone development.* 1. vyd. Cambridge [Mass.]: O'Reilly, 2010. 517 s. Head first series. ISBN 05-968-0354-0.

Shanklin, W. *New iPad will have a 2048x1536 Retina Display* [online]. Publikováno 2012-03-07. [cit. 2014-03-24]. Dostupné z: http://www.geek.com/apple/new-ipad-will-have-a-2048x1536-retina-display-1473867/.

Souppouris, A. *Why Apple's 64-bit iPhone chip is a bigger deal than you think* [online]. Publikováno 2013-08-12. [cit. 2014-03-24]. Dostupné z: http://www.theverge.com/2013/9/12/4722470/iphone-5s-64-bit-processor-is-a-bigger-deal-than-you-think.

SpaceOneApp. In *Twitter* [online]. [cit. 2014-08-23]. Dostupné z: http://twitter.com/spaceoneapp.

Truhlář, J. *SpaceOne* [aplikace]. [Přístup 2014-08-23]. Dostupné z: https://itunes.apple.com/us/app/spaceone/id870179009?l=cs&ls=1&mt=8.

Truhlář, J. SpaceOne for iPhone & iPad. In *Spaceoneapp.com* [online]. 2014. [cit. 2014-08-23]. Dostupné z: http://spaceoneapp.com.

Waynik, N. *Sprite Kit Tutorial: Making a Universal App: Part 1* [online]. Publikováno 2013-10-14. [cit. 2014-03-28]. Dostupné z: http://www.raywenderlich.com/49695/sprite-kit-tutorial-making-a-universal-app-part-1/.

Wilson, T. V. *How the iPhone Works* [online]. Publikováno 2007-06-20. [cit. 2014-03-24]. Dostupné z: http://electronics.howstuffworks.com/iphone1.htm.

Čada, O. *Cocoa: úvod do programování počítačů Apple*. 1. vyd. Praha: Grada Publishing, 2009. 199 s. ISBN 978-80-247-2778-3.

Seznam tabulek

Seznam obrázků

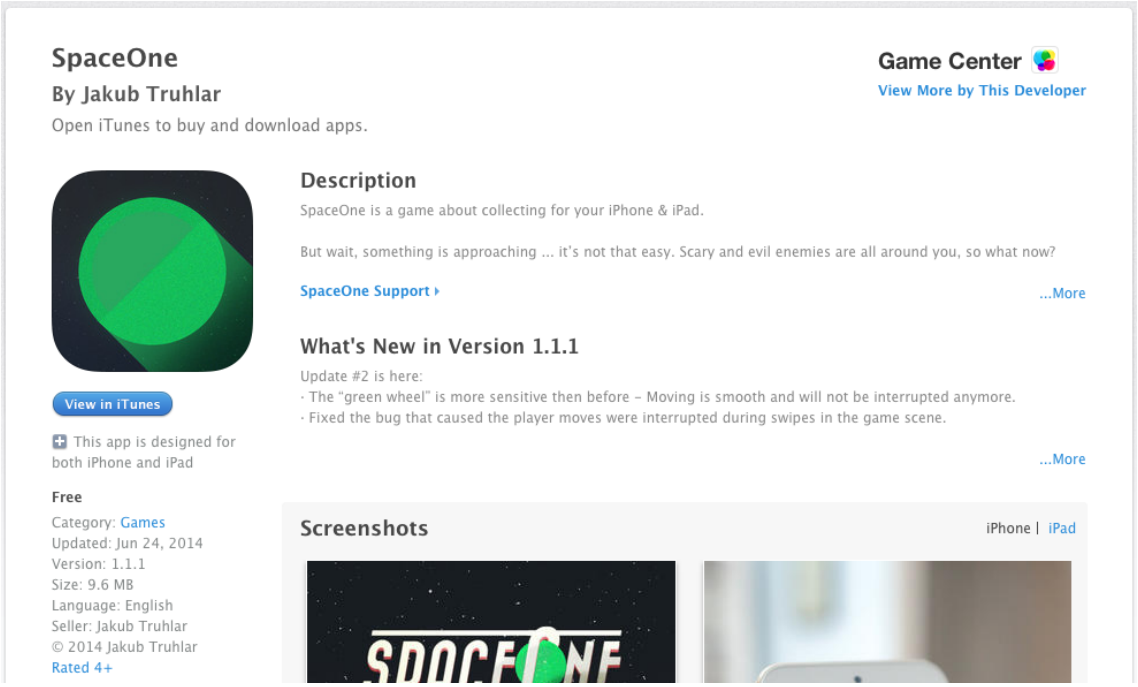
Seznam zdrojových kódů

Seznam použitých zkratek

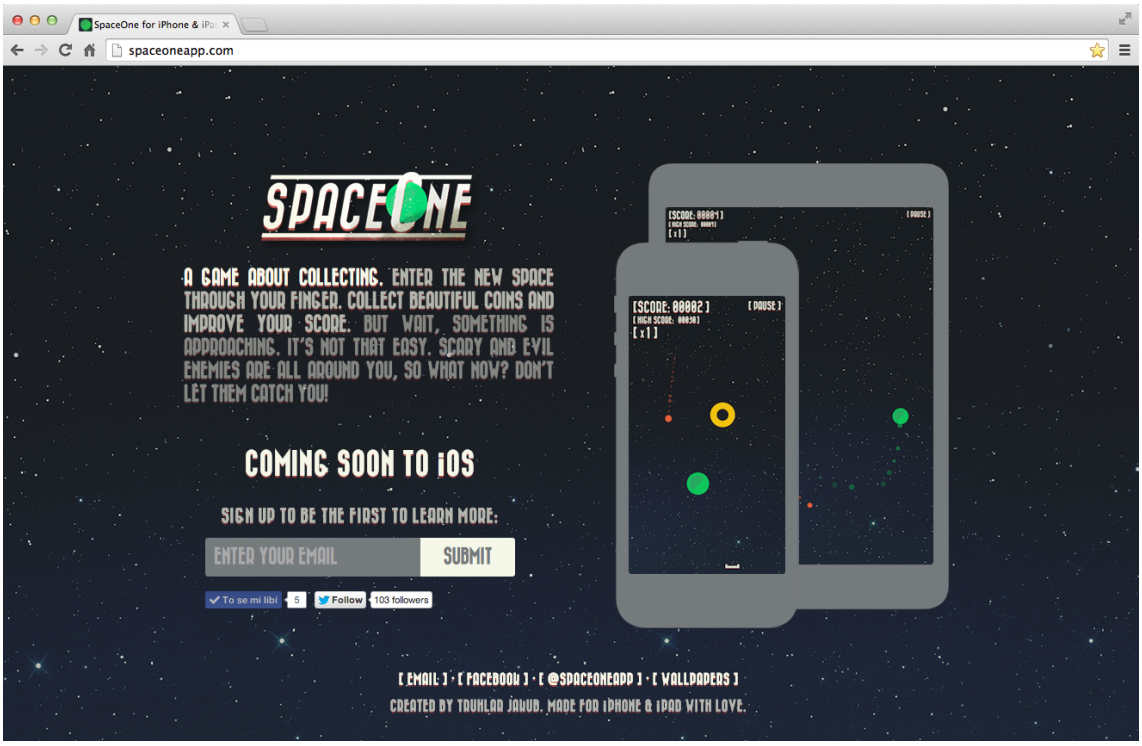
OS	Operating system - operační systém
ppi	Pixels per inch - pixelů na palec
SDK	Software development kit - sada vývojových nástrojů
SIM	Subscriber identity module - účastnická identifikační karta
CPU	Central Processing Unit - procesor
GPU	Graphic processing unit - grafický procesor
SoC	System on Chip
GUI	Graphical user interface - Grafické uživatelské rozhraní
IDE	Integrated Development Environment - Vývojové prostředí
2D	2-dimensional - dvourozměrný
WYSIWYG	What you see is what you get - Co vidíš, to dostaneš
Node	Uzel
FPS	Frames per second - Počet snímků za vteřinu
Indie	Nezávislá počítačová hra
MVC	Model-View-Controller - Model-Vzhled-Řízení

Seznam příloh

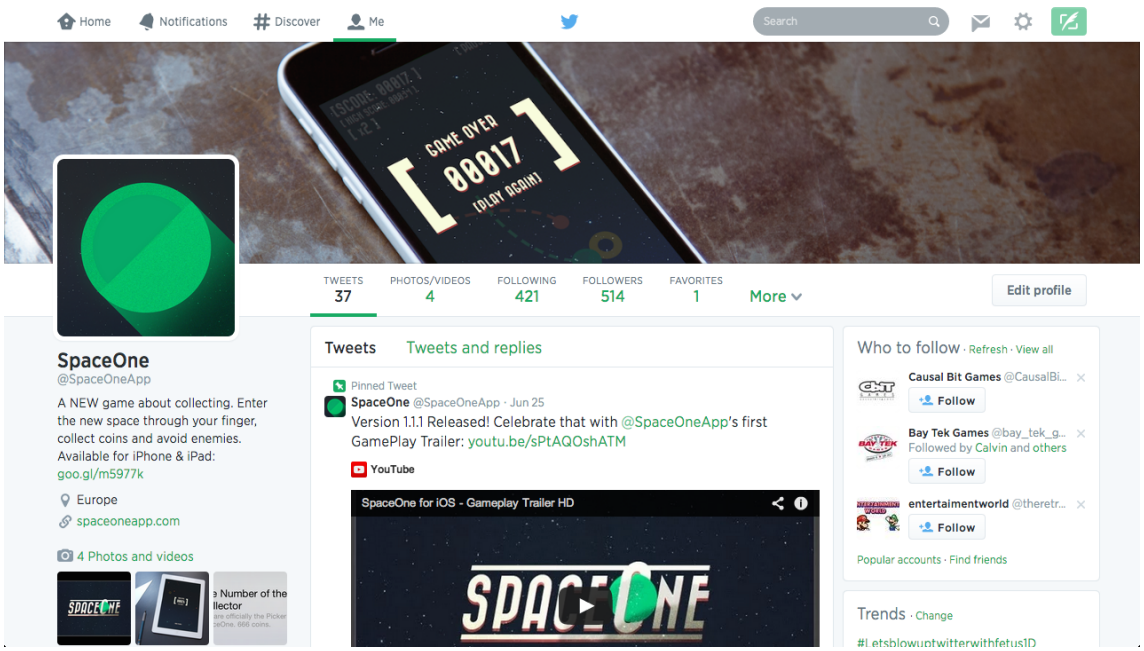
Příloha A - Hra v App Storu



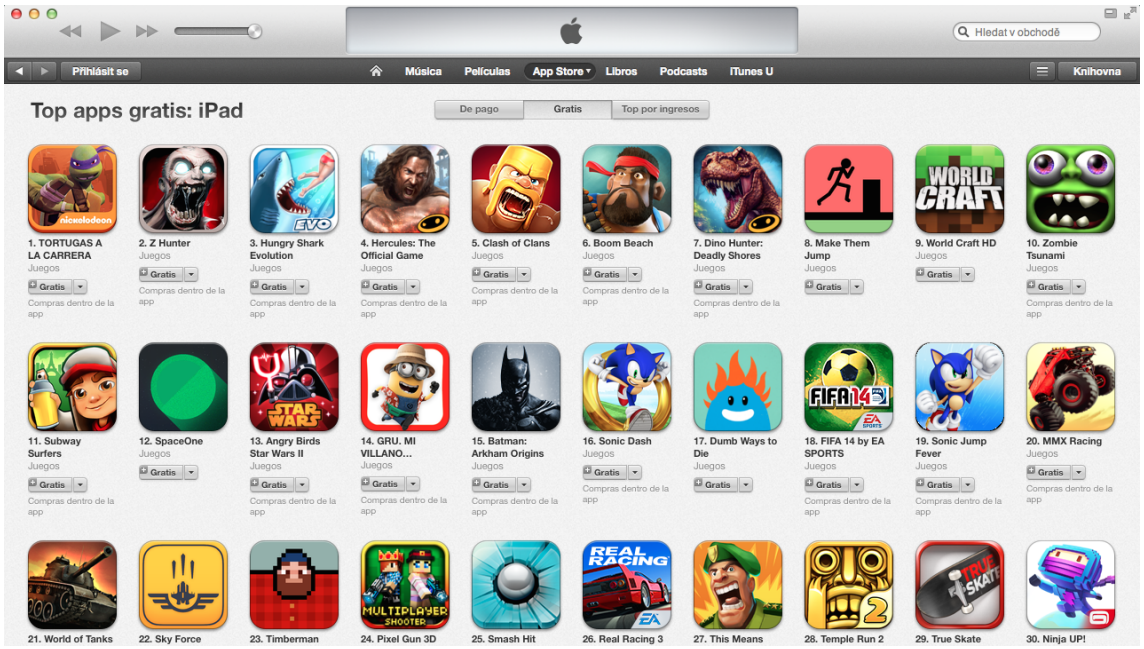
Příloha B - Oficiální webová stránka hry



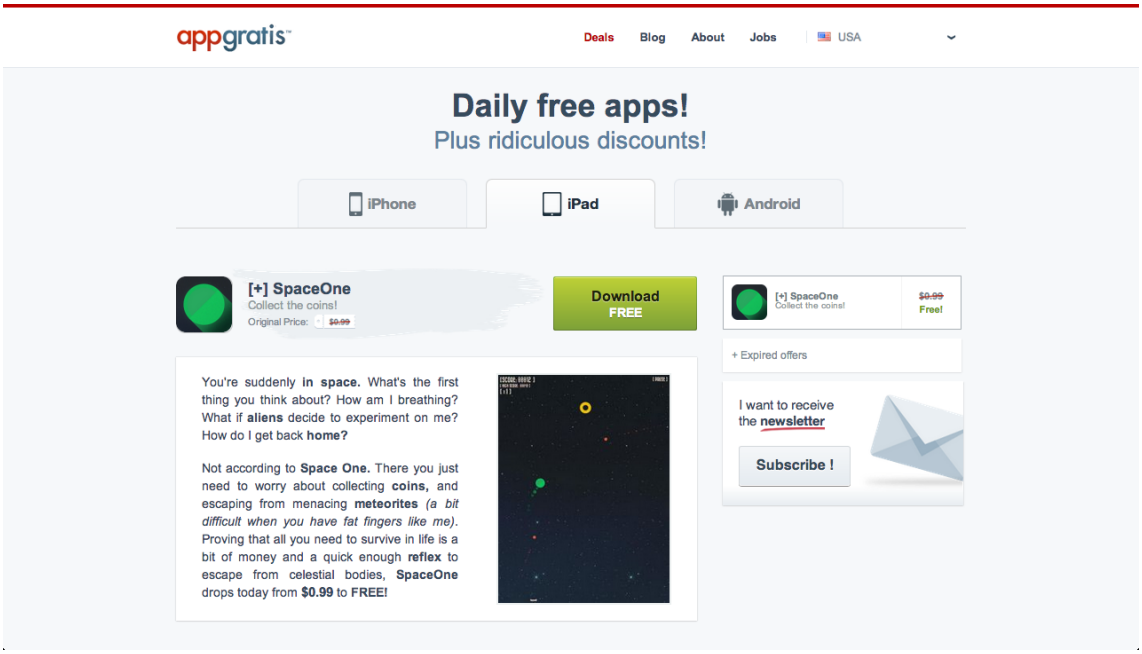
Příloha C - Oficiální Twitter hry



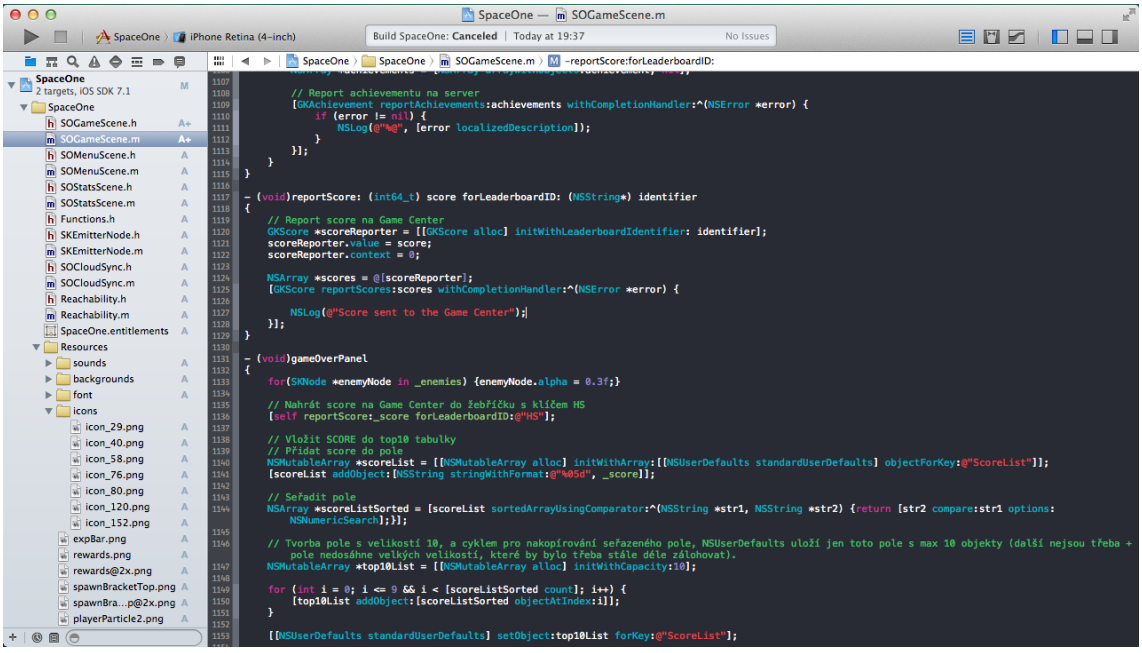
Příloha D - Hra na 12. místě iPad her ve Španělsku



Příloha E - Slevová akce na hru



Příloha F - Projekt hry otevřený v IDE Xcode



Příloha G - UML - Class diagram