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

TUTORIÁL TVORBY APLIKACÍ PRO APPLE IPHONE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. DANIEL ONDRŮJ
AUTHOR

BRNO 2010

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

TUTORIÁL TVORBY APLIKACÍ PRO APPLE IPHONE

CREATION OF APPLE IPHONE APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. DANIEL ONDRŮJ
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL ŽÁK
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Ondrůj Daniel, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Tutoriál tvorby aplikací pro Apple iPhone**
Creation of Apple iPhone Applications
Kategorie: Počítačová grafika

Pokyny:

1. Seznamte se s problematikou tvorby aplikací pro Apple iPhone.
2. Po konzultaci s vedoucím navrhněte vhodnou aplikaci pro Apple iPhone
3. Proveďte implementaci řešení a dokumentujte postup tvorby dané aplikace. Soustřeďte se na výklad celého postupu řešení tak, aby byl materiál vhodný i pro neznalé uživatele
4. Zhodnoťte vytvořené dílo a proveďte srovnání s podobnými aplikacemi.

Literatura:

- na základě domluvy s vedoucím

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Žák Pavel, Ing.**, UPGM FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce si klade za cíl vytvořit tutoriál, který by pomohl vývojářům, majícím zájem o tvorbu aplikací pro Apple iPhone, překonat počáteční neznalost iPhonu, vývojového prostředí, jazyka a možností využití prvků telefonu. Tutoriál by měl být nápomocný čtenářům, kteří mají alespoň nějaké předešlé zkušenosti s vývojem aplikací. Pro ty čtenáře, kteří nemají žádné povědomí o tvorbě programů, je v místech, kde není daný problém řešen do hloubky, uveden odkaz na patřičné zdroje. V těch si mohou problematiku prostudovat detailněji. Aby měl tutoriál své opodstatnění, cílem není jen popis přístroje a nástrojů potřebných pro vývoj. V další části práce je popis návrhu a tvorby funkční aplikace. Protože telefon přišel na trh se zcela novými a do jisté míry i revolučními prvky (akcelerometr, velký dotykový displej a mnoho dalších), jedná se o aplikaci, která obsahuje základní a nejvíce známé funkce. Aby bylo možné tyto prvky rozumně spojit, aplikace je na motivy známé ruské hry Tetris. V tomto dokumentu je rozebrán funkční a grafický návrh aplikace. Konkrétní implementace a zdrojový kód projektu je součástí příloh.

Abstract

The purpose of this thesis is to create a tutorial that could help developers interested in creating new applications for Apple iPhone to pass their initial unfamiliarity with the phone, its development environment, the programming language and the use of other components of the phone. The tutorial should be helpful for readers who have at least basic knowledge of the application development. There are also some useful links for readers who have no previous knowledge of programs' development. These materials describe some basic features and then clarify the common issues in more depth. Not only does this tutorial describe the phone's features and development tools, it also explains development procedures of concrete application. The phone has been introduced with a lot of new features such as accelerometer, large touch screen, etc. Therefore the application is trying to take advantage of most of these characteristics. The motive of this application stems from the old Russian game Tetris. This document describes only the functional and graphical design, concrete implementation and the source code are included in attachments.

Klíčová slova

Tutoriál, tvorba aplikací, Apple, iPhone, Xcode, Instruments, Interface Builder, iPhone OS, jazyk Objective-C, App Store.

Keywords

Tutorial, creation of applications, Apple, iPhone, Xcode, Instruments, Interface Builder, iPhone OS, Objective-C language, App Store.

Citace

Daniel Ondrůj: Tutoriál tvorby aplikací pro Apple iPhone, diplomová práce, Brno, FIT VUT v Brně, 2010

Tutoriál tvorby aplikací pro Apple iPhone

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Pavla Žáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Daniel Ondrůj
25. května 2010

Poděkování

Tímto bych rád poděkoval svému vedoucímu za dostatečný prostor a volnost při tvorbě této práce. Zároveň také za jeho ochotu a čas, který mi věnoval. Dále za cenné rady a podněty, které mi pomohly práci dokončit.

Obsah

Kapitola 1

Úvod

Cílem této práce je vytvořit tutoriál, který by měl pomoci začínajícím vývojářům překlenout neznalost vývoje aplikací pro telefon Apple iPhone. V dnešní době je velmi populární a zájem o něj jak uživatelů, tak vývojářů neustále roste. Protože se ale jedná o zařízení, které svým příchodem na trh zároveň vytvořilo nový trend v oblasti mobilních telefonů, je zde i mnoho novinek, se kterými je třeba se seznámit. Ty ovlivnily způsob ovládání přístroje i vývoje aplikací. Proces vývoje bude hlavní součástí tohoto dokumentu.

Popis samotného vývoje aplikace nebude omezen jen na rozbor programovacího jazyka, prostředí nebo výpis technické specifikace telefonu. Aby bylo možné proniknout do problematiky, je dobré vědět od každého alespoň základní informace. Nyní si projdeme všechny části vývoje aplikace. Jak popis samotného zařízení, jeho vlastnosti a omezení, tak programovací jazyk, ve kterém se aplikace píší (*Objektive-C*). Dále pak nástroje nezbytné, a také doporučené, pro vývoj samotných aplikací.

Aby měl tutoriál své opodstatnění, cílem nebude jen popis přístroje a nástrojů potřebných pro vývoj. V další části práce bude popis návrhu a tvorby aplikace. Výstupem je tedy funkční aplikace. Protože telefon přišel na trh se zcela novými a do jisté míry i revolučními prvky (akcelerometr, velký dotykový displej a mnoho dalších), bude se jednat o aplikaci, která se pokusí základní a nejvíce známé funkce obsáhnout. Pochopitelně se nedají popsat a do aplikace zakomponovat veškeré prvky a možnosti telefonu. Nicméně hodně prvků, zvláště uživatelského rozhraní, pracuje na podobném principu. Pokud tedy čtenář pochopí základní principy práce s těmito prvky, nebude mu činit problém použít jiný, ale funkčně podobný prvek.

Naše aplikace, na které budeme demonstrovat vývoj, by měla mít smysl. Nejedná se tedy o chaotické spojení komponent a vlastností do nepoužitelné aplikace. Aby uživatel pochopil význam jednotlivých prvků a zároveň si osvojil řešení problémů v implementaci na reálných situacích, byla pro tento účel zvolena hra. Tento žánr je do jisté míry schopný postihnout i pro nejjednodušší typ hry široké pole působnosti vlastností, funkcí a schopností přístroje. Pokud budou navíc rozumě a logicky umístěny, dokáží i jednoduchý nápad rozvinout.

Naše ukázková aplikace je na motivy známé ruské hry Tetris. V tomto dokumentu si rozebereme funkční a grafický návrh této aplikace. Probereme tedy popis funkce jednotlivých prvků a návrh jejich implementace. Protože by nebylo efektivní zahrnout vše do tohoto dokumentu, je tutoriál rozdělen na několik částí. Tento dokument slouží jako teoretický základ a návrh aplikace. V dalších částech je věnován prostor konkrétní implementaci (příloha *A* - tutoriál), dále pak příloha *B* (softwarová dokumentace) a příloha *C* (hotový projekt aplikace).

Posledním krokem po dokončení aplikace je její uvedení na trh. Jak je popsáno v ka-

pitole 3.5, je k tomu nutná placená registrace do Apple Developer programu. Distribuce aplikací probíhá přes webové stránky App Store. Tento způsob je trochu inovativní oproti dnes běžným způsobům distribuce aplikací pro mobilní zařízení a byl ze strany Applu rozhodně drobným riskem. Protože je potřeba následovat určitý postup pro správné nahrání a vyžádání si tak schválení programu, je tomuto kroku věnována kapitola 3.5.2.Je v ní popsán stručný popis, co vše je potřeba splnit. Vzhledem k tomu, že jde o širší téma, není možné ho zde popsat tak, jak by se hodilo. Proto je uveden odkaz na postup přímo na stránkách Applu, který uživatele provede krok za krokem.

Tento poslední krok je ovšem volitelný. Je zde popsán jen pro úplnost pro ty uživatele, které třeba vývoj aplikací pro iPhone bude dál zajímat a měli by zájem o budoucí distribuci vlastních aplikací. Běžní uživatelé, které zajímá problematika vývoje pro telefon iPhone pouze z informativního hlediska, tento krok mohou vynechat.

Obrázek 1.1: Apple iPhone

Kapitola 2

Apple iPhone, základní fakta

Když 23. října 2001 uvedla společnost Apple na trh MP3 přehrávač iPod, byl ze začátku pro svoji netradiční konstrukci odsuzován jako naprosto neperspektivní. Nicméně v následujících asi 6-ti letech se prodalo přes 100 milionů kusů a co víc, Apple částečně ovlivnil a vytvořil novou éru designu, funkčnosti a ovládání osobních přehrávačů.

Podobným vývojem si prošel i iPhone. Apple se dlouho držel stranou trhu s mobilními telefony. Ovšem když přišel na trh nový iPod Touch (obrázek 2.1), cesta k iPhonu byla už krátká. Stačilo přidat GSM modul, pár nezbytných funkcí a nový fenomén mohl začít.

Obrázek 2.1: Apple iPod první generace, iPod Touch, iPhone

Telefon opět následoval cestu svého prapředka iPodu. Musel zdolat několik větších či menších překážek. Mezi nejvýznamnější patři:

- *Nezvyk uživatelů na rozměry.* Pravda, v USA byla cesta snazší. Američané jsou zvyklí na větší rozměry mobilů díky komunikátorům Blueberry, atd. (viz obrázek 2.2). Naopak v Evropě byl trend co největší miniaturizace.

- *Chybějící HW klávesnice.* To už byl pro hodně uživatelů velký nezvyk. V USA velmi rozšířené Blueberry měli už od počátku plnohodnotnou QWERTY klávesnici. A v dnešní době, kdy i američtí uživatelé si čím dál víc oblibují psaní textových zpráv nebo emailů, mohla být chybějící klávesnice problém. Když se nad tím zpětně zamyslíme, problém to nebyl. Sice celodotykové telefony už dříve zkoušely renomované firmy (např. Ericsson). Ale vždy byl doplněn alespoň numerickou klávesnicí, která je u telefonů běžná. iPhone byl proto v tomto segmentu celodotykových displejů opravdu

Obrázek 2.2: Komunikátor BlueBerry

první.

- *Neschopnost MMS zpráv první verze iPhonu.* Tento problém si ale Apple brzy uvědomil a nedostatky brzy odstranil vydáním aktualizací. Ty, mimo jiné, řešily i absenci funkcí pro kopírování a vkládání textu.

- *Absence slotu pro paměťové karty.* Ano, dnes, v době rostoucích nároků na paměť a neustálého rozšiřování kapacity razí Apple cestu, kdy telefon je už od výroby vybaven určitou kapacitou vnitřní paměti, kterou nelze dále rozšiřovat. Ale má to svoji logiku. Apple se tak snaží omezit možnost dostat do telefonu škodlivý software, který by tak uživatele obtěžoval a kvalitu telefonu subjektivně snižoval. Data je tak do telefonu možné dostat pouze přes iTunes. Dále také může prodávat shodné přístroje za odlišné ceny jen díky rozdílné paměti.

- *Chybějící multitasking.* Jedna z nejvíce vyčítaných vad iPhonu. Apple se snažil tuto „vadu“ omlouvat tím, že má zájem na tom, aby každá aplikace byla naprosto stabilní, měla maximální možné dostupné prostředky a její chod nemohl být ničím ovlivněn. Později ovšem lehce musel poodstoupit tlaku a uvedl tzv. Push Notifikace (viz kapitola 3.4). A s uvedením iPhone OS 4.0 toto padlo zcela. Ten totiž obsahuje plnohodnotný multitasking. V současné době je pouze v beta verzi pro vývojáře a jeho masivní nástup se předpokládá až s nástupem nové verze telefonu. V tomto dokumentu se tedy předpokládá práce s iPhone OS řady 3.

Mezi velmi oblíbenou funkci, která si okamžitě získala obrovskou popularitu je akcelerometr. Dokonce nápad je tak povedený, že ostatní výrobci telefonů si dnes u svých produktů vyšších kategorií nedovolí akcelerometr, nebo tuto součástku jinak řešenou, nepřidat. Sice podobná pohybová čidla používal už dříve SonyEricsson ve svých walkmanových telefonech. Ale funkčnost (například pro přepínání písniček poklepáním, ...) nebyla kvalitně dotažena do konce a tak si u uživatelů nezískala velkou oblibu. Zde ovšem umožňuje detekci rotace telefonu ve třech osách a to poměrně s přesnými hodnotami. Toho lze využít samozřejmě v aplikacích mnoha způsoby.

iPhone přišel na trh s novými prvky (design, funkce, ovládání, ...), které si získaly velký zájem zákazníků. Jejich kombinace a načasování zajistily, že si okamžitě získal velikou popularitu. Tomu odpovídají i milionová čísla prodejů (jen od června 2007, kdy se iPhone začal prodávat, do konce roku 2007 se ve světě prodalo 4 miliony kusů). Oproti

smartphonům se ještě určitě najde mnoho věcí, které by se daly vylepšit, aby se mohl stát plnohodnotnou managerskou pomůckou. Mnoho uživatelů jej vnímá proto spíše jako velmi propracovanou hračku, než jako „dospělý“ komunikátor.

Kapitola 3

iPhone z pohledu vývoje aplikací

3.1 iPhoneOS

iPhone OS se skládá z operačního systému a dalších technologií, které jsou použity pro běh aplikací jak na iPhonu, tak na přehrávači iPod touch. iPhone OS má mnoho vlastností a technologií společných s Mac OS X. Vývojáři pro Mac OS X v něm proto najdou množství známých technologií. Pro potřeby mobilního telefonu byl ale systém mírně upraven vzhledem k rozdílnosti použití zařízení oproti osobnímu počítači/laptopu. Mezi rozdílné vlastnosti, které nalezneme pouze u iPhone OS můžeme zmínit například akcelerometr, prostředí Multi-Touch pro ovládání telefonu nebo neschopnost multitaskingových aplikací (o multitaskingu se dočteme více v kapitole 3.4). Rozdílů by bylo možné najít samozřejmě více, ale jejich vyjmenovávání není teď podstatné. Je ale důležité si uvědomit rozdíl mezi běžnou kancelářskou aplikací a aplikací pro mobilní telefon a cíl jejich využití. To také ve svých materiálech zdůrazňuje Apple. Snaží se tak působit na vývojáře, aby vytvářeli aplikace co nejvíce intuitivní a pro uživatele pohodlné. Existuje dokument (viz [7]), ve kterém se velmi podrobně popisují zásady pro tvorbu uživatelského rozhraní. Podle tohoto dokumentu také Apple kontroluje hotovou aplikaci před jejím uveřejněním.

3.1.1 Vrstvy iPhone OS

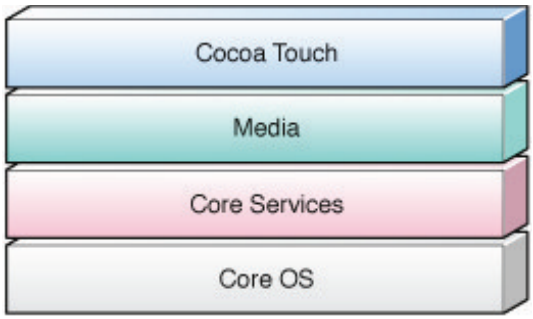
Jak již bylo napsáno výše, iPhone OS je svoji strukturou velmi podobný architektuře použité u Mac OS X. Skládá se ze čtyř základních vrstev, které si teď popíšeme.

Nejspodnější vrstva hierarchie je *Core OS*. Je založena na stejném jádře jako Mac OS X. Toto jádro je založeno na kódu pro UNIX BSD. Společně s vyšší vrstvou *Core Services* obsahuje základní rozhraní iPhone OS. Včetně rozhraní pro přístup k souborům, nízkoúrovňovým datovým typům a dalším. Napsána jsou většinou v jazyce C a obsahují technologie jako je Core Foundation, CFnetwork, SQLite a přístup k POSIX vláknům a další.

Vyšší vrstvy obsahují pokročilejší technologie napsané v jazyce C nebo Objective-C. Vrstva *Media* obsahuje technologie pro podporu vykreslování ve 2D a 3D, dále pak audio a video. Obsahuje v C napsané OpenGL ES, Quartz a Core Audio. V Objective-C také pokročilý animační Core Animation.

V nejvyšší vrstvě *Cocoa Touch* je většina technologií v Objective-C. V této vrstvě je základní infrastruktura pro vývojáře. Jsou zde knihovny pro práci se soubory, síťové operace, vizuální struktura aplikace (tvorba oken, pohledů, ...). Dále rozhraní pro práci s hardwarovými moduly, jako je například akcelerometr.

Při vytváření nového projektu je počáteční bod vrstva Cocoa, dále jen API (*Application*



Obrázek 3.1: Systémové vrstvy

Programming Interface - označuje v informatice rozhraní pro programování aplikací). Celkově se doporučuje držet se vyšších vrstev. API je velmi obsáhlé a umožňuje pohodlnou práci s celým systémem. Proto je vhodný důvod využít služeb nižších vrstev pouze pokud chceme implementovat vlastní chování, které není dostupné na vyšší úrovni. Toto je ovšem nutné konzultovat přímo s Applem na stránkách podpory.

3.2 Vlastnosti telefonu podstatné pro vývoj aplikací

Vlastnosti telefonu je zbytečné v této práci popisovat prostým výčtem vlastností nepodstatných pro vývoj. Jako například počet řádků při psaní sms nebo výdrž baterie. Mezi podstatné vlastnosti pro vývojáře patří prvky, které mají vliv na výsledné vlastnosti a vzhled aplikace. Proto si zde ve stručnosti probereme

- multimedia a vibrace
- displej (rozměry a vlastnosti)

3.2.1 Multimedia a vibrace

Mezi multimediální prvky telefonu patří možnost přehrávat zvuky, hudbu (knihovna `AudioToolbox.h`, resp. `AudioServices.h`) a videa (knihovna `MPMoviePlayerController`). Od verze iPhone OS 2.0 je pro přehrávání objekt AVAudioPlayer. Ten umožňuje práci s mnoha formáty souborů (mp3, aiff, aac a další) a s prakticky neomezenou velikostí souboru. Dále umožňuje nastavení počtu opakování skladby, popřípadě nekonečnou smyčku nebo přehrávání více souborů najednou. Starší objekt AudioServices měl omezené možnosti a mezi nejzásadnější rozdíl patří neschopnost přehrávat soubory delší než 30s. Používá se tedy pro svoji jednoduchost pro přehrávání krátkých zvuků a upozornění.

Mezi hlavní podporované formáty pro přehrávání zvuků a hudby patří například *.aif, .wav, .aac, .mp3, . . .* Video lze přehrávat i na celé obrazovce, a to ve formátech *.mov, .mp4, .mpv a .3gp.*

Další část, kterou zde zmíníme jsou vibrace. Nepatří sice zcela do multimédií, ale z hlediska programátora je práce s vibracemi v rámci objektu AudioServices. Bohužel pro vibrace není širší základna použitelnosti. Programátor nemůže jakkoli ovlivnit délku nebo intenzitu vibrace. Proto se spuštění vibrací omezuje pouze na zavolání funkce ve tvaru `AudioServicesPlaySystemSound(kSystemSoundID_Vibrate)`. To zajistí zavibrování zhruba délky jedné sekundy. Pokud bychom chtěli delší vibrace, jako jediná možnost se nabízí opakované volání funkce, což je ovšem poněkud krkolomné.

3.2.2 Displej (rozměry a vlastnosti)

Velmi podstatnou částí telefonu je displej. Vzhledem k tomu, že se jedná u celodotykového přístroje o jediný komunikační kanál mezi programem a uživatelem, je mu ze strany Applu věnována velká pozornost. Už dříve jsme zmínili, že umístění prvků, logice uspořádání uživatelského rozhraní a nebo nepříklad logice ovládání pomocí dotyků se věnuje mnoho dokumentů.

Jen v krátkosti si projdeme technické parametry. Technologie displeje je tzv. kapacitní. Z toho plyne, že dotyk je vyhodnocen na základě odporu lidského těla, a proto ovládání pomocí jiných nástrojů (stylusy, tužka nebo i nehet, ...) není možné. Další vlastností z hlediska programátora je, že není možné vyhodnocovat tlak na displej. Dotyk je tedy zjednodušeně jen hodnota ANO, NE.

Rozměry displeje a jednotlivých prvků jsou velmi dobře zobrazeny na obrázku 3.2. Vidíme na něm jak hlavní rozměr displeje (320x480px), tak i rozměry ostatních ovládacích prvků. To je velmi podstatné, protože ty ovlivňují zbývající plochu displeje, na kterých můžeme zobrazovat další prvky.

Obrázek 3.2: Rozměry prvků uživatelského rozhraní

3.3 Jazyk Objective-C

Jazyk Objective-C je objektově orientovaný jazyk, který vznikl rozšířením standardního jazyku ANSI C o syntaxi pro objektové prvky. Umožňuje tak využít základní rysy objektového programování, jako jsou zapouzdření, dědičnost nebo polymorfismus. Konstrukce a syntaxe tříd je podobná jako v jazyku Smalltalk.

Je nemožné zde probrat celý jazyk Objective-C. A to také není účelem tohoto dokumentu. Nicméně v následujících několika bodech si popíšeme pár základních prvků jazyka nutných pro vytvoření nebo pochopení nejjednodušší aplikace. Dostaneme tak představu o tom, jaká je syntaxe jazyka a jakou logikou se aplikace vytvářejí. Neměl by tedy být problém po přečtení této kapitoly porozumět nebo alespoň se umět zorientovat v dokumentu *Your First Application* (viz [6]), kde je popsána tvorba jednoduché aplikace. To je mimo-

chodem dobrý začátek pro seznámení se s vývojem aplikací pro iPhone a s jeho logickou skladbou.

Pokud je ovšem čtenář pokročilý programátor nebo mu nečiní rychle si osvojit nový jazyk, může přeskočit až na kapitolu 3.4.

3.3.1 Typy souborů

Díky tomu, že je tento jazyk nadstavbou nad jazykem C, podporuje tak v mnoha ohledech stejnou syntaxi. Například shodná je definice hlavičkových a zdrojových souborů.

Soubory Objective-C používají tyto druhy přípon:

- **.h** hlavičkový soubor. Klasické hlavičkové soubory obsahující dekralarace tříd, typů, ...
- **.m** zdrojový soubor. Zdrojové soubory obsahující Objective-C tak i C kód.
- **.mm** zdrojový soubor. Zdrojový soubor může oproti předešlému obsahovat i C++ kód. Ovšem užití této přípony by mělo být jen v případě, kdy se odkazujeme na C++ třídy nebo na vlastnosti Objective-C kódu.

Pokud chceme vložit soubory do zdrojového kódu, je možné použít standardní direktivu `#include`. Jazyk ale nabízí i lepší způsob. Direktiva `#import` je stejná jako `#include`, ale navíc zajistí, že se soubor vloží pouze jednou. Tento postup se tedy samozřejmě preferuje.

3.3.2 Řetězce

Objective-C podporuje stejné konvence pro tvorbu řetězců jako v C. Ale doporučuje se používat pro řetězce třídu `NSString`. Je analogická k třídě `String` v C++. Obaluje řetězce, obsahuje správu paměti, podporuje Unicode, způsob formátování pomocí printf a další. Protože se s řetězci pracuje velmi často, existuje zkrácený zápis pro vytváření řetězců pomocí `NSString` objektů z konstant. K použití zkráceného zápisu stačí před normální, v dvojitých uvozovkách uzavřený řetězec, přidat symbol `@`. Příklad vidíme na 3.3.

```
NSString *myString = @"My String\n";
```

Obrázek 3.3: Ukázka zkráceného zápisu řetězce

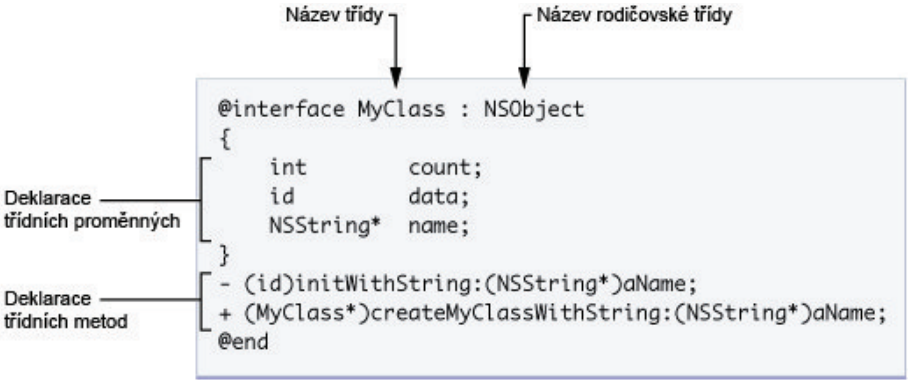
3.3.3 Třídy

Specifikace třídy v Objective-C vyžaduje dvě věci: rozhraní a implementaci. V rozhraní deklarujeme třídu, definujeme proměnné a metody této třídy. V implementaci je konkrétní kód pro tyto metody.

Na obrázku 3.4 je ukázána deklarace třídy MyClass poděděné od třídy *NSObject*. Deklarace třídy vždy začíná direktivou `@interface` a končí direktivou `@end`. Dále seznam proměnných uzavřených blokem { } závorek. Následuje deklarace metod třídy.

Implementace deklarované třídy je podobně jako deklarace vyznačena dvěmi direktivami `@implementation` a `@end`.

Objekty uložené v proměnných jsou vždy přes ukazatel. Objective-C podporuje slabě i silně typované proměnné s objekty. Silně typované ukazatele obsahují jméno třídy v deklaraci typu. Slabě typované ukazatele používají typ `id` pro konkrétní objekt. Slabě typované



Obrázek 3.4: Deklarace třídy

ukazatele se často používají pro třídy kolekcí, kde může být přesný typ objektu neznámý. Oba typy vidíme na 3.5.

```
MyClass* myObject1; // Silně typované

id myObject2; // Slabě typované
```

Obrázek 3.5: Ukázka dvou druhů typu objektu

3.3.4 Metody

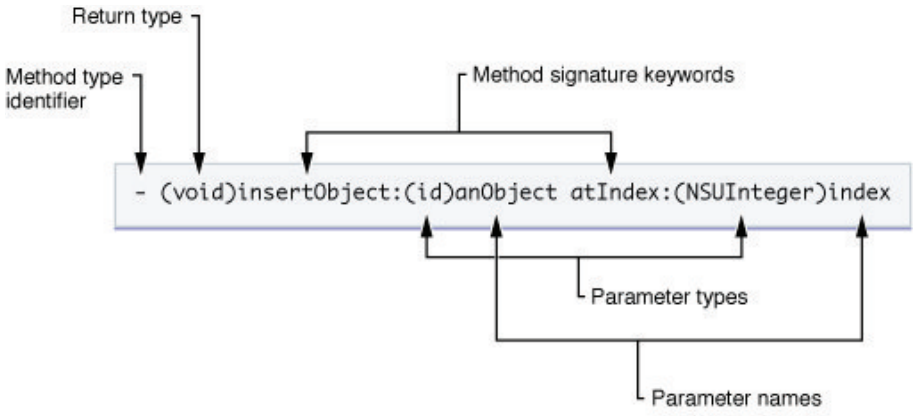
Podobně jako například jazyk Java, třída v Objective-C může deklarovat dva typy metod: metody instance a metody třídy. Metodu instance je možno vykonat pouze z instance třídy. Naopak metoda třídy instanci třídy nevyžaduje.

Deklarace metody se sestává z identifikátoru typu metody, návratového typu, jednoho nebo více klíčových slov podpisu metody (signatury), typů parametrů a jmen parametrů. Tento popis zní sice dosti komplikovaně, a těžko si pod ním po prvním přečtení lze něco představit. Ale pokud se podíváme na obrázek 3.6, bude nám určitě systém tvorby názvů metod jasnější. Je zde ukázána deklarace metody instance `insertObject:atIndex:`. Deklaraci předchází znaménko mínus `-`, které značí, že se jedná o metodu instance (znaménko `+` by bylo v případě metody třídy). Jméno metody (`insertObject:atIndex:`) je výsledkem spojení všech klíčových slov v signatuře, včetně dvojteček. Dvojtečka deklaruje přítomnost parametru. Skládání názvů metod vychází z jazyka Smalltalk, kdy každá metoda je v podstatě věta. Tedy metoda z obrázku 3.6 by se dala přeložit jako „vlož tento objekt na tento index“. Z počátku je nutné si na tento způsob psaní metod zvyknout, nicméně pokud se držíme jednotného stylu názvosloví, kód je potom dobře čitelný.

Když chceme zavolat metodu, učiníme tak „zasláním“ zprávy odpovídajícímu objektu (podobně jako ve Smalltalku). Zprávou je v tomto případě signatura metody spolu s informací o parametrech, které metoda potřebuje.

Zprávy jsou uzavřeny v hranatých závorkách. Uvnitř závorek je objekt, který přijímá zprávu uveden na levé straně a zpráva samotná (spolu s jakýmikoli parametry, které zpráva vyžaduje) na pravé straně. Například k zaslání zprávy `insertObject:atIndex:` objektu v proměnné `myArray` bychom použili syntaxi jako vidíme na 3.7.

Abychom se vyhnuli množství lokálních proměnných ukládajících mezivýsledky, je možné



Obrázek 3.6: Deklarace metody

```
[myArray insertObject:anObj atIndex:0];
```

Obrázek 3.7: Ukázka volání metody

zprávy vnořovat a zřetězit tak několik zpráv do jednoho příkazu.

3.3.5 Vlastnosti

Pomocí vlastností nahrazujeme přístupové metody (anglicky accessors) objektu. Jsou to zkratky pro definování metod, které přistupují k již existujícím proměnným instance. Jedná se o podobnost syntaxe s *gettery* a *settery* (viz odkaz [9]). V praxi to tedy znamená, že nemusíme vytvářet *gettery* a *settery* pro každou proměnnou instance. Místo toho se jen určí vlastnosti proměnných a poté se konkrétní *gettery* a *settery* vytvoří v době překladu.

Deklarace vlastností se vkládají s deklaracemi metod do rozhraní třídy. Základní definice používá direktivu `@property`, následovanou typem informace (atributem) a jménem vlastnosti. Vlastnosti je také možné upravovat a definovat, jak se budou přístupové metody chovat. Direktiva `@Synthesize` v implementačním souboru zařídí vytvoření vlastností atributů. Syntaxi je možno vidět na 3.8.

```
@property [(attribute [, attribute2, ...])] type name;
```

Obrázek 3.8: Ukázka syntaxe vlastnosti

Probereme si zde nejdůležitější atributy vlastností:

- *assign* - prosté přiřazení. Atributem říkáme, že do privátní proměnné se má přiřadit adresa objektu, který předáváme „setteru“.
- *retain* - tento typ říká, že se do privátní proměnné přiřadí adresa předávaného objektu a současně se zvýší reference na objekt. Tedy se stáváme vlastníkem a objekt existuje dokud nezavoláme release. Důležité ovšem je, že objekt má stále stejnou adresu, tedy pokud někde jinde změníme hodnotu, projeví se změna i zde.
- *copy* - tento typ je podobný jako retain. Jen s tím rozdílem, že vytvoří nové paměťové

místo pro předávaný objekt. Proto pokud tento objekt někde změníme, na privátní proměnné se to neprojeví.

- *readonly* - zcela samovysvětlující. Nemůžeme hodnotu měnit, pouze získat.
- *readwrite* - plný přístup k proměnné.

Ještě jsou zde další dva typy vlastností. `atomic` a `nonatomic`. Pokud explicitně nespecifikujeme, všechny vlastnosti jsou atomic. V principu se to týká jen programů pracujících s vlákny. Je potřeba totiž řešit situace, kdy dvě vlákna přistupují (zápis/čtení) do atributu objektu ve stejný čas. Řízení přístupů tedy řeší typ *atomic*. *nonatomic* je jednodušší a toto neřeší, proto jej použijeme pokud víme, že vlákna nemohou s proměnnou pracovat současně (typicky v nevláknových aplikacích).

3.3.6 Správa paměti

Práce s pamětí v jazyce Objective-C je velmi podobná jako v jazyce C. Objective-C sice umožňuje správu pomocí *Garbage collectoru* (viz [10]), ale ne, pokud programujeme pro iPhone. V tom případě se o paměť musíme starat sami.

Práce s pamětí je založena ne na přímém uvolňování paměťového místa objektů, ale na počítání referencí. Pokud vytvoříme objekt nebo se staneme jeho vlastníkem (upřesněno dále), zvýší se jeho počet referencí o jedna.

Vlastníkem objektu se stáváme, pokud ho vytvoříme (metody `alloc` nebo `copy`). To je zřejmé. Ovšem Objective-C nabízí ještě třetí možnost a tou je zisk vlastnictví přes property objektu. Přesněji, pokud je property daného objektu typu *retain*.

Jen pro doplnění, můžeme také pracovat s objekty, které nevlastníme. Tedy objekty (typicky řetězce), které vytvoříme bez použití `alloc` nebo `copy` metody. Tento objekt v paměti existuje, my s ním pracujeme, protože nejsme vlastníci, nejsme zodpovědni za jeho uvolnění.

Dvojí druh vlastnictví můžeme vidět na ukázce 3.9.

```
NSString *myString1 = [NSString stringWithString:@"Řetězec cizí."]; // Tento řetězec nevlastníme

NSString *myString2 = [[NSString alloc] initWithString: @"Řetězec můj."]; // Tento řetězec vlastníme
[myString2 release]; // A také jsme zodpovědni za uvolnění jeho paměť
```

Obrázek 3.9: Ukázka dvou typů vlastnictví

Jakmile už s objektem v daném místě nepotřebujeme pracovat, je nutné počítadlo referencí opět snížit (metoda `release`). Každý objekt, který vytvoříme, je děděný od výchozího objektu *NSObject*. Ten obsahuje tuto metodu `release`, která okamžitě neuvolní objekt z paměti, ale pouze sníží referenci na objekt o 1. Pokud počítadlo dosáhne 0, systém takový objekt uvolní z paměti sám. Pokud tak neučiníme, dochází k tzv. „memory leaku“, tedy k úniku paměti.

Ještě je zde jiná možnost práce s pamětí. A tou je objekt `NSAutoreleasePool`. Toto je zvláštní objekt vytvářený v `main` metodě (viz obrázek 4.2). Pokud vytvoříme objekt (pomocí `alloc` nebo `copy`) nebo jsme vlastníkem (pomocí *retain*), můžeme tomuto objektu poslat zprávu `autorelease`. Tedy se o jeho uvolnění nemusíme starat. Objekt je přidán do `NSAutoreleasePool` a ten odstraní objekty na konci aplikace. Vzhledem k hospodárné práci s pamětí se doporučuje tento způsob používat pouze v případech, kdy objekt v aplikaci potřebujeme a není zřejmé, kdy jej uvolnit. Pokud tedy víme, že objekt už nepotřebujeme, použijeme metodu `release` místo `autorelease` při vytváření objektu.

3.4 Tvorba aplikace

Aby bylo možné vyvíjet aplikace pro iPhone, je zapotřebí vývojářský balíček *Xcode* (viz kapitola 3.4.1). Kromě něj se na vývoj aplikací pro iPhone využívají ještě další nástroje. Jsou jimi *Interface Builder* (viz kapitola 3.4.2) a *Instruments* (viz kapitola 3.4.3).

Tyto nástroje běží pouze na systému Mac OS X. Sice po uvedení několika softwarů Applem pod platformu Windows, například Safari, Quick Time Player a další, se spekuluje, jestli Apple neuvede i *Xcode* pod Windows. Bylo by to logické vzhledem k sílícímu zájmu o vývoj aplikací pro tyto telefony.

Ještě je možné najít zmínky o emulaci celého systému Mac OS X pod Windows. Ovšem tato možnost nebyla v rámci tohoto dokumentu zkoušena. Ale vzhledem k výkonu VirtualPC by asi práce nebyla příliš příjemná.

Pokud chceme vytvořit jednoduchou aplikaci, je to poměrně snadné díky velkému množství šablon dostupných v *Xcode*. Ovšem náročnější aplikace, které mají širší uplatnění vyžadují širší studium knihoven a dokumentace. Na stránkách *Apple Developer* [3] v sekci pro platformu, pro kterou chceme aplikace vyvíjet, jsou odkazy na dokumentaci, kde jsou popsány nástroje a proces vytváření aplikací poměrně kvalitně.

Důležitý dokument pro prvotní hlubší seznámení s vývojem a technologiemi, které poskytuje iPhone najdeme v *iPhone OS Technology Overview* [5]. Je zde popis klíčových konceptů, specifických informací o vývoji aplikací nebo příklady a popis použití *Xcode*. Velmi podstatná část je na konci, kde je přehledná tabulka, ve které je seznam knihoven a názvů jejich dokumentací s popisem, co má daná knihovna za funkci. Dále základní informace potřebné pro vývoj aplikací je obsažen v *Cocoa Fundamentals Guide* [4]. Zde je popsán základ jazyka Objective-C a programovací zvyklosti a návrhové vzory použité pro UIKit a mnoho jiných systémových knihoven.

Výše uvedené dokumenty jsou volně dostupné bez jakékoli registrace. Po registraci do developers programu má vývojář přístup k iPhone SDK, na fórum vývojářů a další.

Aplikace, které vytvoříme, jsou v telefonu umístěny na hlavní obrazovce spolu s ostatními systémovými aplikacemi, jako jsou Fotky, Hodiny a další. Pokud aplikaci spustíme, je to kromě jádra a několika systémových démonů, jediná běžící aplikace. Za běhu zabírá aplikace celou obrazovku. Pokud uživatel zmáčkne tlačítko Home, aplikace se ukončí a systém zobrazí opět domovskou nabídku s programy (tlačítko Home je jediný způsob, jak korektně ukončit aplikace). Neexistuje tak možnost souběžně otevřených oken nebo více spuštěných aplikací. Všechna data jsou zobrazena v jediném okně. Z toho důvodu byly vytvořeny nové pohledy a ovládací prvky umožňující rozumné prezentování dat. Výhodou tohoto řešení je, že umožňuje čerpat veškeré systémové prostředky.

Jak již bylo dříve zmíněno, iPhone OS nepodporuje multitasking. Není tedy možné mít více spuštěných aplikací v jeden okamžik. Systémové aplikace, jako budím nebo příchozí hovor samozřejmě řeší tuto událost podle očekávání. Ale pokud budeme mít například aplikaci pro komunikaci na Facebooku nebo přes ICQ, o nové zprávě by se nebylo možné dozvědět do té doby, než bychom aplikaci spustili. Apple proto řeší tento problém takzvanými *Push notifikacemi*. V praxi fungují tak, že aplikace sice neběží na pozadí, nicméně pokud dojde k události, na kterou má být uživatel upozorněn, aplikace na tuto událost zareaguje zvukem, textovým oknem nebo malým červeným kolečkem u ikony aplikace. Pokud uživatel chce zjistit, o jakou událost se jedná, musí aplikaci spustit a ta potom zobrazí změny (například příchozí zprávy) standardním způsobem. Na obrázku 3.10 je ukázán příklad, jak toto upozornění může vypadat. Typickým příkladem *Push Notifikace* je SMS zpráva. Přehraje zvuk, zobrazí se textové okno s kouskem zprávy a u ikony *Zprávy* v menu telefonu je malý

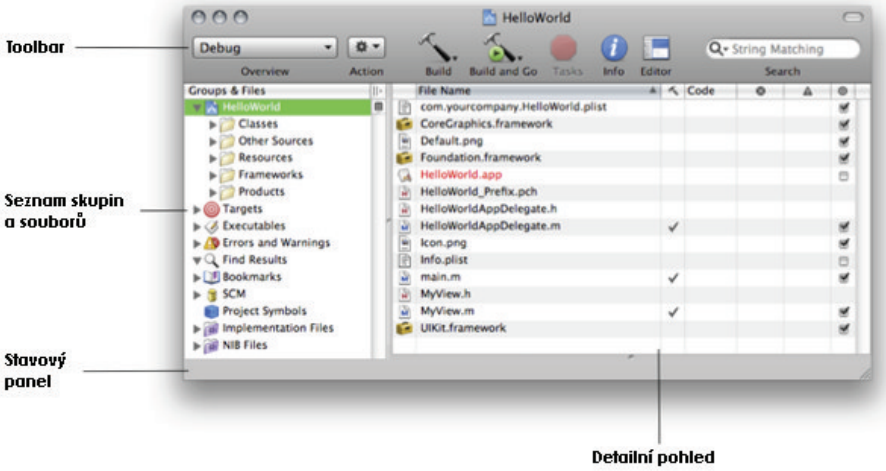
červený bod.

Obrázek 3.10: Ukázka Push notifikace

Oproti běžným aplikacím na počítačích je i systém ovládání odlišný. Dotykový displej umožňuje jak jednoduché vybírání jedním stisknutím, tak i známé Multi-Touch ovládání. Tedy kombinace několika stisknutí a pohybů. Pokud jsou tyto prvky rozumně použity, dá se celkově ovládání velmi zefektivnit a zpříjemnit uživateli.

3.4.1 Xcode

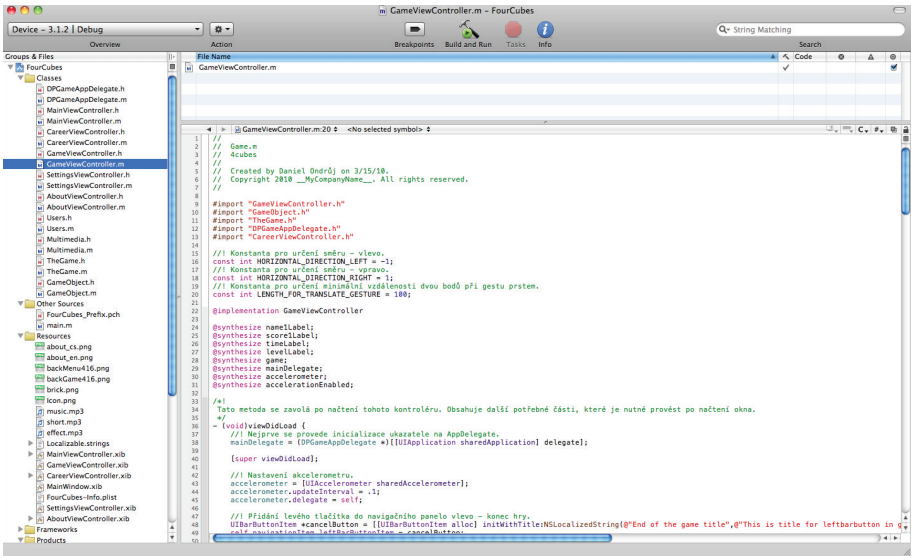
Xcode lze chápat jako vývojářský balíček. Obsahuje základní prostředí pro vývoj zdrojového kódu a balíčky pro kompletní správu projektů.



Obrázek 3.11: Schéma prostředí Xcode

S *Xcode* se pracuje podobně jako s ostatními vývojovými prostředími, jako je například Visual Studio od společnosti Microsoft. Založený projekt spravuje všechny informace spojené s aplikací. Tedy zdrojové soubory, pravidla potřebná k složení všech částí dohromady a další. Okno s otevřeným projektem pak může vypadat například jako na obázku 3.11.

Můžeme zde vidět několik klíčových částí. *Toolbar* pro změnu nastavení kompilace aplikace; *Seznam skupin a souborů* pro správu zdrojových souborů a cílů, které z nich budou sestaveny;*Stavový panel* pro zobrazení informací například o proběhlé kompilace a *Detailní pohled* volitelně zobrazující pracovní prostor. Samozřejmostí je pokročilý textový editor umožňující zvýraznění syntaxe, doplňování kódu, skrývání bloků a mnoho dalšího. Schéma s popisky vidíme na obrázku 3.11. Okno *Xcode* s naším projektem na obrázku 3.12.



Obrázek 3.12: Okno projektu v Xcode

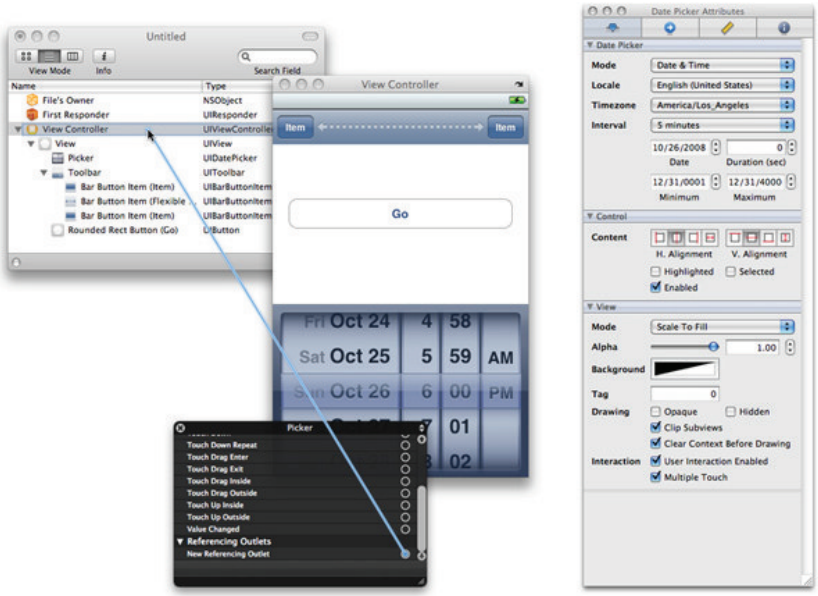
Sestavit aplikaci v *Xcode* je možné pro iPhone simulátor i přímo pro fyzické zařízení. Simulátor obsahuje prostředí pro testování aplikací a je možné tak zjistit, jestli se aplikace chovají jak mají. Poté co je vývojář spokojen se základním chováním aplikace, *Xcode* umožňuje sestavení a spuštění na iPhone nebo iPod Touch připojenému k počítači. Na připojeném telefonu je pak možné dále aplikaci ladit. Abychom mohli testovat na fyzickém zařízení, je nutné mít zaplacen poplatek $99 (99 amerických dolarů). Dále pak vlastnit certifikát *Developer* nebo *Distribution*. Toto je blíže popsáno v kapitole 3.5.1.

3.4.2 Interface Builder

Nástroj Interface Builder slouží k vizuální tvorbě uživatelského rozhraní. S jeho pomocí se sestavuje aplikace jen díky přetahování hotových komponent na její okno. Komponenty obsahují standardní systémové prvky jako jsou přepínače, textová pole, tlačítka a další.

Poté co jsou komponenty umístěny, je možné dále s nimi manipulovat, nastavovat různé atributy a napojovat je na objekty ve zdrojovém kódu.

Hotové rozhraní je možné uložit do souboru *nib* (historicky jsou soubory vedeny jako nib, dnes je ovšem nahradil vylepšený formát *xib*). Jedná se o speciální formát pro uložení zdrojů (anglicky resources). *xib* soubory obsahují všechny informace, které UIKit potřebuje ke znovuvytvoření stejných objektů v běžící aplikaci. Načtením *xib* souboru se vytvoří objekty uložené v souboru a nastaví se tak, jak byly v *Interface Builderu* při tvorbě. V *xib* souboru jsou pomocí ukazatelů uloženy informace, díky kterým je možné propojit nově vytvořené objekty s již existujícími a také jak mají objekty předávat informace od uživatele.



Obrázek 3.13: Ukázka práce s Interface Builder

Tvorba uživatelského rozhraní

Jak již bylo napsáno výše, *Interface Builder* slouží pro vizuální tvorbu uživatelského rozhraní. Obrazovku a její prvky si umístíme jak se nám líbí (a jak to program dovolí) a poté je musíme napojit na kód. Metody, jako reakce ovládacích prvků na dotek nebo prvky napojit na objekty, se kterými chceme dále pracovat (skrývání tlačítek a podobně). To je poměrně příjemné a dosti názorné, proto se tento postup doporučuje pro začátečníky pro lepší představu o vzhledu a funkci jednotlivých prvků.

Uživatelské rozhraní je možné ale vytvářet i jinak. A to přímo v kódu. Je tedy možné celý projekt vytvořit bez jediného souboru xib. Všechny prvky si vytvoříme v kódu. Má to svou nespornou výhodu, že máme plnou kontrolu nad ovládáním prvků. Nevýhodou ale je někdy složitá tvorba uživatelského rozhraní jen pomocí umísťování prvků pomocí souřadnic a konfigurace dalších nezbytných objektů.

Ukázka funkce `applicationDidFinishLaunching` (viz 4.2.1) pro případ použití *Interface Builderu* je na obrázku 3.14a. Čistě programový způsob je na 3.14b. V obojím případě byl vytvořen *viewController* (objekt pomocí kterého pracujeme s daným xib souborem) s jedním tlačítkem. Výsledek můžeme vidět na obrázku 3.15.

3.4.3 Instruments

Tento nástroj umožňuje analyzovat výkon iPhone aplikace jak při běhu na simulátoru, tak na samotném zařízení.

Nástroj *Instruments* sbírá data z běžící aplikace a prezentuje tato data na grafickém panelu. Je možné sbírat data, která nás o aplikaci zajímají. Například spotřeba paměti, aktivita disku, síťová aktivita nebo výkon grafického systému. Na časové ose je možné zobrazit všechny typy nejrůznějších informací srovnané u sebe a umožňuje tak nahlédnout na celkové chování aplikace, ne jen na její určité části. Detailnější informace jsou uchovávány

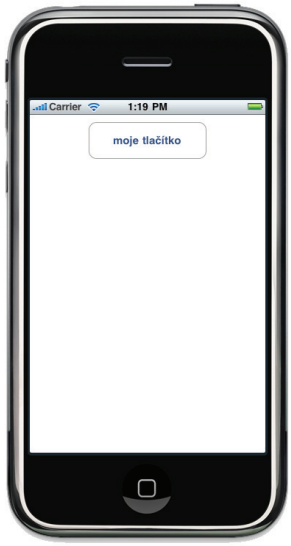
```
- (void)applicationDidFinishLaunching:(UIApplication *)application {

    MyView *viewController = [[MyView alloc] initWithNibName:@"MyView" bundle:[NSBundle mainBundle]];

    [window addSubview:[viewController view]];
    [window makeKeyAndVisible];

}
```

(a) Tvorba uživatelského rozhraní pomocí Interface Builderu

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {

    MyTableViewController *controller = [[MyTableViewController alloc] initWithStyle:UITableViewStylePlain];

    UIButton *myButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    myButton.frame = CGRectMake(80, 30, 160, 50);
    [myButton setTitle:@"moje tlačítko" forState:UIControlStateNormal];
    [controller.view addSubview:myButton];

    [window addSubview:[controller view]];
    [window makeKeyAndVisible];

}
```

(b) Tvorba uživatelského rozhraní programově

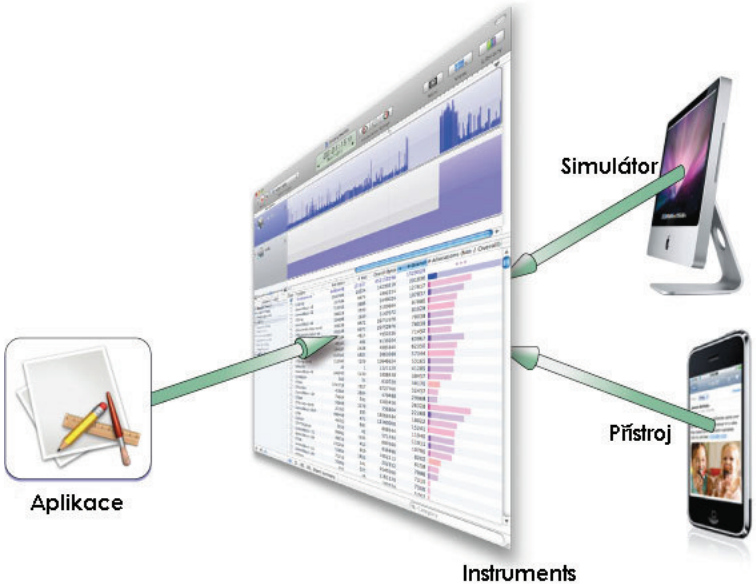
Obrázek 3.14: Ukázka obou přístupů tvorby uživatelského rozhraní



Obrázek 3.15: Výsledné uživatelské rozhraní

v podrobných záznamech.

Dále tento nástroj umožňuje uchovávat data z jednotlivých měření a ty pak vzájemně srovnávat. Což může být vhodné během procesu ladění, kdy můžeme vidět pokrok nebo právě neúspěch ladění aplikace.



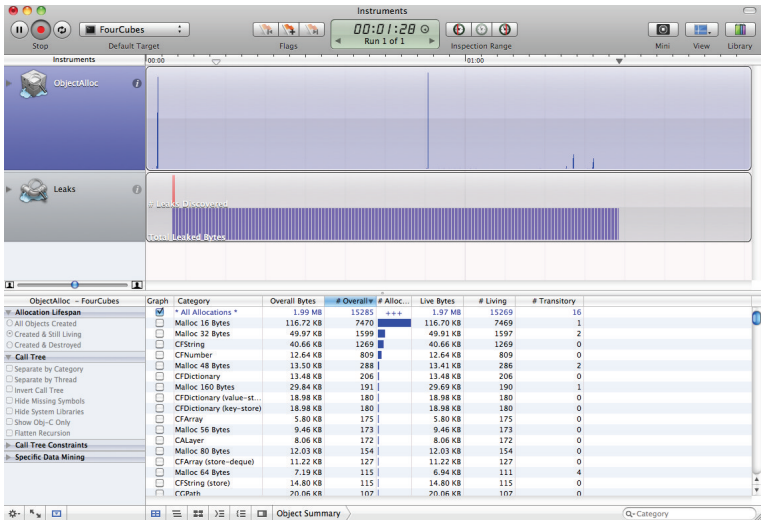
Obrázek 3.16: Schéma práce prostředí Instruments

Detekce úniků paměti

Nástroj je velmi propracovaný a například umožňuje v debug verzi přeložené aplikace krokovat pomocí brakepointů (jak v simulátoru, tak na fyzickém zařízení). Chod aplikace je poté sice velmi zpomalený, nicméně umožňuje tak získat velmi důležité informace. Mezi jednu z nejdůležitějších patří detekce tzv. *memory leaků*, neboli úniků paměti. Správa paměti je popsána v kapitole 3.3.6. Úniky paměti umí *Instruments* detekovat a dokonce na časové ose vidíme, kdy se tak stalo a jaká metoda tento únik způsobila. To je patrné z obrázku 3.17. Pro odladění aplikace je to velmi důležitá schopnost.

Instruments ovšem obsahují i několik chyb. Jedna z nich je, že pokud spustíme *Instruments* pod simulátorem, jsou detekovány úniky paměti i falešné. Tedy že interpretuje *leaky* i tam, kde nejsou. Zda-li se jedná o falešný poplach nebo ne, zjistíme buď až na telefonu nebo se můžeme podívat, jaká metoda tento konkrétní únik paměti vytvořila. Pokud je to metoda, se kterou nepracujeme, nemusíme si dělat starosti.

Mezi zajímavější část patří úniky paměti samotného systému. Pokud například spustíme hudbu, po chvíli nám vyskočí malý únik paměti (v řádu kilobytů). Ten vyvolá vrstva *media*. Jsou to ovšem úniky systémové a ty nijak neovlivníme. Naší starostí je pouze paměť, se kterou pracujeme my. A pokud s ní pracujeme hospodárně a aplikace je stabilní, je vše v pořádku.



Obrázek 3.17: Okno Instruments - detekce úniků paměti

3.5 Aplikace na prodej

3.5.1 Developer program - zdarma a placený

Aby mohl vývojář vůbec začít s vývojem aplikací, musí se registrovat do Apple Developer programu (viz [3]). Registrace probíhá přímo pro platformu, pro kterou chce vyvíjet aplikace (iPhone, iPad, Safari a MacOSX). Tato bezplatná registrace umožní přístup do developerské části stránek Applu. Zde si, mimo jiné, může stáhnout balík *SDK*, který obsahuje vývojové prostředí *Xcode*, nástroje *Instruments* (viz 3.4.3), *Interface Builder* (viz 3.4.2) a další. Dále je zde obrovské množství dokumentace, tutoriálů a návodů. Snad pro každou část API je možné najít nějaký materiál. Ovšem pro nového uživatele může být právě toto nepřeberné množství materiálů matoucí. Proto ve vývojářské sekci je i fórum (v angličtině), kde ostatní uživatelé a i vývojáři přímo z Applu radí při jakýchkoli problémech. Neregistrovaný uživatel má pouze přístup k dokumentaci tříd jednotlivých prvků (tzv. Class Reference).

Díky Apple Developer programu tedy vývojář může vyzkoušet tvorbu aplikací pro iPhone (samozřejmě se to týká i iPod Touch a iPad). Pokud z jakéhokoli důvodu nemá zájem pokračovat, není jakkoli vázán k dalším povinnostem (vyjma například zákazu šíření softwaru a informací poskytnutých ve vývojářské sekci a dalším povinnostem plynoucím z podmínek, se kterými uživatel při vstupu do vývojářského programu musí souhlasit). Developer program umožňuje vývoj v *Xcode* a obsahuje i simulátor zařízení, na kterém je možné si aplikaci zkoušet a ladit. Ovšem jedná se spíše pro orientační ladění. Simulátor se nechová vždy stejně a některé funkce podporuje jen omezeně nebo vůbec (například akcelerometr).

Pokud vývoj aplikací myslí vývojář vážně, zaplatí členský poplatek $99. Pro zaplacení částky je nutné projít několika stránkami s instrukcemi, kde na konci je samotné placení. Pro většinu zemí je placení částky pomocí kreditní karty přímo na stránkách. Česká republika bohužel nepatří do této skupiny. Proto musí zájemce o developer program vyplnit formulář (ke stažení podle instrukcí na webu), kde vyplní potřebné údaje a odfaxuje(!) tento formulář do Applu. Tato možnost se jeví jako poněkud nepříjemná, protože je nutné uvést číslo kreditní karty včetně CC2 kódu. Tedy všechny údaje, které jsou potřeba ke zneužití karty. Bohužel jiná možnost než tato není (snad zatím) a nezbývá tedy, než důvěřovat

v nezneužití. Většinou do 24 hodin získá vývojář odpověď emailem, že platba proběhla v pořádku a od této doby běží 1 rok platnosti developer programu. V uživatelské části na stránkách Apple Developer programu (viz [3]) si vygeneruje *Developer* certifikát. Ten umožní další funkce při vývoji. Jedna ze dvou nejdůležitějších je, že je umožněno nahrání a testování aplikace přímo na fyzickém přístroji. Tato možnost je naprosto nezbytná a velmi doporučená. Protože až na skutečném telefonu můžete vyzkoušet chování aplikace tak, jak se bude chovat u uživatele a jak již bylo řečeno dříve, otestovat i některé funkce, které na simulátoru zkoušet možné nebylo.

Dále pak placený developer má možnost ad-hoc distribuce až na 100 dalších telefonů. Tato možnost je například pro testování aplikace více uživateli a druhy přístrojů. Dále pak testování síťové komunikace a další. Je potřeba zaslat dané osobě tzv. *provisioning profile* a *app* soubor s aplikcí. Uživatel si potom oba soubory přes program iTunes nahraje do telefonu.

Placená registrace je celkově mírnou nevýhodou a hodně začínajících vývojářů zcela jistě odradí. Je nutná i v případě, pokud chceme vyvíjet a distribuovat programy zcela zdarma. Samotný vývoj aplikací se bez placené registrace neobejde. Během vyvoje programu je sice možné testovat na simulátoru. Ale vzhledem k omezeným možnostem slouží jen pro orientační náhled, jak se aplikace chová. Pokud bychom chtěli aplikaci distribuovat na App Store, je zcela nutná placená registrace. Možná se zdá celkový systém registrací, certifikátů a placení poněkud komplikovaný. Apple se tak snaží chránit záruku kvality a spolehlivosti aplikací pro běžné uživatele. Proto je pochopitelné, že na vývojáře se kladou vyšší nároky.

3.5.2 App Store - zveřejnění aplikace

Pokud už máme aplikaci hotovou a řádně odladěnou, zbývá ji jen představit světu. Aby ji ale bylo možné dále distribuovat, musíme ji umístit na stránky App Store [1]. Jiný způsob masového šíření neexistuje. Apple tak chrání uživatele, protože než je aplikace zveřejněna, odborníci v Applu ji prozkoumají, otestují a zjistí, zda-li není závadná. Pro vývojáře je to poměrně nepohodlné. Zároveň ale pokud Apple hru schválí, uživatel má alespoň malou kontrolu, že je jeho program rámcově v pořádku. Popíšeme si tedy základní části nutné k odeslání aplikace k posouzení.

A zde je tedy druhá výhoda placeného developer programu. A tou je možnost zveřejnění aplikace na App Store. Zveřejněním není myšleno přímo, ale pouze možnost odeslat aplikaci na schválení. Až po kladném vyřízení je aplikace zveřejněna. K odeslání aplikace ke schválení je připraven v uživatelské části na stránkách (developer sekce) speciální, několika stránkový, formulář, kde musíme vyplnit několik položek. Na závěr pak nahrát samotnou aplikaci. Mezi základní podmínky pro nahrání a položky formuláře patří:

- Doposud jsme měli vygenerován pouze certifikát *Developer*. Aby bylo možné odeslat aplikaci ke zveřejnění, je nutné mít certifikát *Distribute*. Bez toho se nepodaří nahrát aplikaci do webového formuláře. Systém pak bude hlásit chybu *The binary you uploaded was invalid. The signature was invalid, or it was not signed with an Apple submission certificate.*

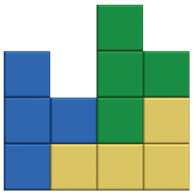
- Jedna z hlavních a podstatných věcí. Musíme uvést název a dále kategorii a podkategorii aplikace.

- Dále je nutné nastavit App ID a ikonu program. Toto je nastavení je dobře popsáno na [8] (pouze pro registrované vývojáře).

- Aplikace musí mít webové stránky podpory, které musí být dostupné během celé doby existence aplikace na App Store. V opačném případě to může vést až ke stažení aplikace. Pro podporu musí být vedena i emailová adresa, na kterou se mohou uživatelé obracet, pokud mají dotazy nebo problémy.

- Musí být vyplněn dotazník o škodlivosti aplikace pro různé věkové skupiny. Například jak moc aplikace obsahuje násilí, hazardní hry, nevhodné obrázky a další. Podle toho je aplikaci přiřazen status závadnosti například 12+ (tedy nevhodné pro děti do dvanácti let).

- Aplikace musí mít svoje logo. Jednak logo aplikace umístěné přímo v projektu, a tedy i finální verzi programu, nese název Icon + přípona (například Icon.png). Toto malé logo musí mít rozměry 57x57px a je umístěno v menu telefonu. Dále pak logo pro webové stránky. To je velikosti 512x512px. Název by neměl začínat číslicí, protože se může stát, že webový portál pro příjem aplikací ke schválení tento název nepřijme. Toto logo je pak uvedeno v App Store u aplikace. Ukázka loga naší aplikace je na obrázku 3.18.



Obrázek 3.18: Ukázka loga naší aplikace

- Mezi jednu z posledních věcí, které je nutné vyřešit, patří cena. Výběrem z možností *Free, Tier 1 - 0.99, Tier 2 - 1.99, Tier 3 - 2.99, ...* Dále můžeme vybrat měnu, ve které aplikaci chceme účtovat. Apple si bere z placené aplikace 30% a stará se o distribuci na App Store a dále o rozšíření aktualizací naší aplikace (stačí, když nahrajeme novou verzi programu). Daně platí uživatel podle daného regionu. Je potřeba podepsat dokument, že budeme platit daně sami, aby nedocházelo ke dvojímu zdanění (daně by se platily v Americe a my bychom museli platit daň v ČR). Apple potom pošlecelých zbývajících 70% z ceny.

- Musíme vygenerovat release verzi aplikace (pod certifikátem *Distribute*). Vznikne tak soubor *název.app*, který zabalíme do *zip* nebo *ipa* (obdoba zip) formátu. Tento kompresovaný soubor nahrajeme do formuláře.

- Dále je nutné nahrát velké logo a několik obrazovek (tzv. screen shotů) naší aplikace v podporovaných rozměrech 320x480, 320x460px.

Pokud se vše podařilo, aplikace je odeslána. Nyní zbývá už jen čekat a doufat, že aplikace bude schválena a uveřejněna na App Store.

Kapitola 4

Vlastní aplikace

Nyní si probereme konkrétní návrh naší aplikace. Jak by aplikace pro tento telefon měla vypadat, co by měla umět a s tím související, jaké vlastnosti iPhonu jsme se rozhodli implementovat.

Na začátek si musíme říct, co vlastně touto aplikací sledujeme. Naším cílem je představit schopnosti a prezentovat základní (nejpopulárnější) vlastnosti iPhonu. Jak již bylo napsáno výše, abychom srozumitelně dokázali tyto věci spojit do jedné aplikace, budeme se tedy snažit vytvořit jednoduchou hru. Nebudeme zbytečně vymýšlet komplikovaný scénář a charaktery postav. Pro naše účely bude naprosto stačit, pokud si vezmeme jako vzor jednoduchou, a o to slavnější, starou ruskou hru Tetris.

Co by tedy hra měla umět? A pro koho je vůbec určena? Toto jsou asi dvě základní otázky, které si musíme položit před samotným vývojem. Neplatí to samozřejmě jen u tohoto programu. V oblasti vývoje software je velmi často vidět i přes promyšlený a dobrý nápad zároveň zmatené a nepřipravené ztvárnění. U iPhone aplikací platí důkladný návrh dvojnásob. Apple všechny aplikace před zveřejněním kontroluje a pokud nevyhovují zaběhlým standardům nebo nemají logickou strukturu, může je dostat tvůrce zpět na přepracování.

4.1 Návrh hry

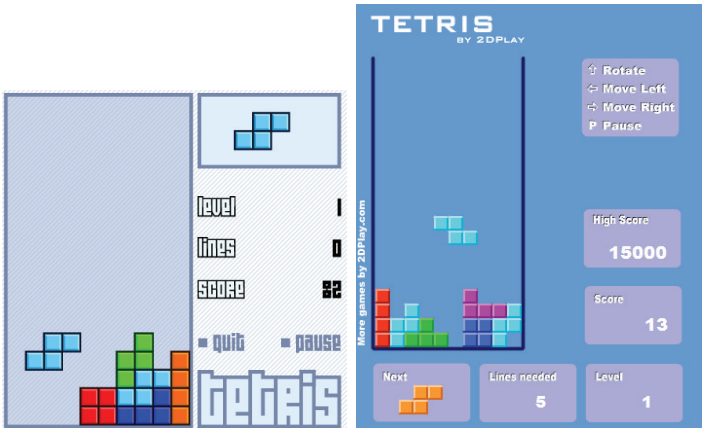
Návrh a tedy i vzhled a celkové chování aplikace je velmi důležitou částí ve vývoji. Pokud se podíváme na obyčejnou hru Tetris, ta je svým prostým, ale zároveň propracovaným, nápadem velmi variabilní.

Pro přiblížení, o co ve hře jde. Po hracím poli padá vždy zhora v jeden okamžik blok. Tento blok má tvar jednoho ze sedmi základních tvarů. Tyto bloky můžeme skládat pomocí posunu nebo rotace. Cílem je složit horizontální řadu. Pokud se tak stane, řada mizí a my jsme oceněni body. Hra trvá tak dlouho, dokud je kam bloky skládat.

Hra za svou historii už dostala mnoho vzhledů a přepracování. Pro naše účely však bude stačit, když si vybereme jeden ze základních typů. Pro inspiraci mohou sloužit obrázky 4.1a a 4.1b.

Co všechno by tedy měla aplikace umět a které prvky jsme se rozhodli implementovat:

- padající bloky
- hrací pole
- přepínání obrazovek a navigace pomocí navigačního panelu



(a) Ukázka hry Neave Tetris (b) Ukázka hry Tetris (od 2DPlay.com

Obrázek 4.1: 2 ukázky hry Tetris

- hudba ve hře a dále zvuky a vibrace jako reakce na určitou událost (veškeré funkce je možné individuálně nastavit)
- multitouch, simpletouch a poklepání
- akcelerometr
- ukládání hráčů a jejich herních výsledků
- reakce na neočekávané události
- lokalizace

4.2 Implementace

V části implementace nebudeme probírat detaily implementace. Ty jsou probrány v příloze *A - tutoriál* a příloze *B - softwarová dokumentace*. Zde si jen nastíníme logickou stavbu programů pro iPhone a objektový návrh naší aplikace. Vzhledem k poměrně podobné struktuře všech aplikací pro iPhone (vždy práce s obrazovkami, kde každá plní nějakou funkci - nastavení, vlastní funkce aplikace, ...) bude objektový návrh všech aplikací podobný.

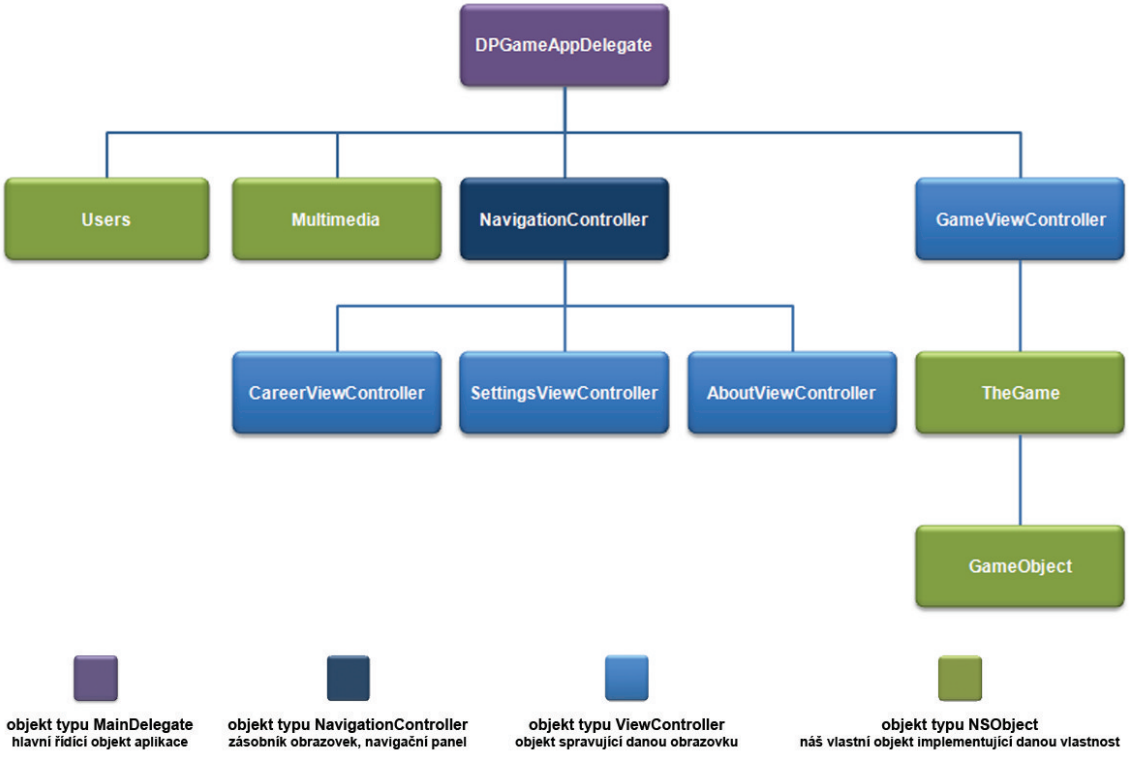
4.2.1 Struktura aplikace

Nejprve si řekněme něco ke struktuře každé aplikace. Jak již bylo zmíněno dříve, jazyk Objective-C vychází z jazyka C. Tedy začátek je vždy funkce `main`. Ovšem ta je v případě Objective-C velmi krátká (viz obrázek 4.2).

Je zde pouze vytvoření objektu `NSAutoreleasePool`, což je něco jako hloupý *garbage collector* (viz kapitola 3.3.6). Dále se zde volá funkce `UIApplicationMain`. Tato funkce tvoří hlavní přístupový bod, kde se vytváří objekt aplikace (načtení potřebných knihoven do paměti, ...), objekt MainDelegate (kořen stromu - viz obrázek 4.3) a nastaví další potřebné parametry.

```
int main(int argc, char *argv[]) {

    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];

    return retVal;
}
```

Obrázek 4.2: Funkce main



Obrázek 4.3: Stromová struktura aplikace

Pro nás je nejdůležitější objekt *MainDelegate* (na obrázku 4.3 je to kořen fialovou barvou). Obsahuje metodu `applicationDidFinishLaunching`, která se spustí jakmile je vše potřebné připraveno k běhu aplikace (veškeré knihovny a potřebná nastavení jsou nahrána v paměti). Zde je nutné načíst nezbytná data (typicky uložená nastavení aplikace). Dále pak velmi důležitá část. V této funkci totiž vytvoříme hlavní okno (`window`). `window` je něco jako softwarový displej. Jednotlivé obrazovky jsou pak reprezentovány objekty *viewController*, jejichž atribut `view` reprezentuje vždy danou obrazovku. `view` daného *viewControlleru* můžeme buď přiřadit v této funkci přímo do `window` (jako výchozí obrazovka po spuštění aplikace) nebo, jako v našem případě, použijeme objekt *UINavigationController*. Ten funguje něco jako zásobník oken, kde vždy na vrcholu je aktuálně zobrazené okno. Pomocí metod `push` a `pop` další okna přidáváme nebo odebíráme. Tím vzniká hierarchická struktura oken aplikace, kdy se můžeme buď vrátit na předchozí obrazovku nebo se zanořit dál. V našem případě tedy `window` nepřiřadíme jen jedno aktivní okno, ale celý tento zásobník. Pomocí změn oken v zásobníku se objekt postará i o zobrazení aktivní obrazovky na displej telefonu.

Na kořen jsou navázány další dva objekty. Jsou to *Multimedia* (pro přehrávání zvuků a spouštění vibrací) a *Users* (uživatelé a jejich data).

Dalším důležitým objektem je *MainViewController*. Ve stromové struktuře jej vidíme pod *UINavigationController*. Jedná se o pomyslný první prvek zásobníku *UINavigationController* (toto se nastaví v InterfaceBuilderu - výchozí controller pro navigationController - viz příloha A a B). V programu je to hlavní nabídka menu (viz obrázek 4.4). Odtud je přístup ke všem částem aplikace - hra, kariéra, nastavení a nápověda. Tedy pokud uživatel stistkne například tlačítko nastavení, tento objekt provede `push` s konkrétním controllerem a uživateli se tak zobrazí obrazovka nastavení. Analogicky s ostatními objekty. Tyto objekty jsou tedy navázány na tento objekt.

Obrázek 4.4: Hlavní menu aplikace

GameViewController je umístěn mimo *MainViewController*, což se může zdát podivné. Nicméně má to své opodstatnění, protože je nutné ukládat stav hry, která během hraní náhle skončí. Tento stav se uloží a při opětovném spuštění se pokračuje v rozehrané hře. Tedy musíme být schopni ihned po načtení aplikace zobrazit panel s hrou. Nejprve se tedy vloží

do objektu typu *UINavigationController* objekt typu *MainViewController*, ovšem ihned na něj se vloží *GameViewController*, jako aktivní okno. Pokud hru ukončíme, ze zásobníku tento kontroller odstraníme a zůstane tam tak nabídka menu, což očekáváme.

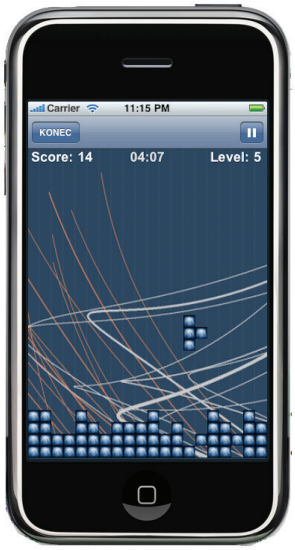
GameViewController má na sebe navázán objekt *TheGame*. Ten reprezentuje samotnou hru (správa skóre, herní pole, čas, ...) Na něj je připojen objekt *GameObject*, který reprezentuje jeden herní blok (jeho příslušné instance reprezentující tvar si aplikace vytváří v průběhu). *TheGame* obsahuje objekt reprezentující herní pole a další metody pro práci s ním. Podrobnější popis je v kapitole 4.4.2. *GameObject*, neboli herní blok, je rozebrán v kapitole 4.4.1.

4.3 Grafický návrh

Po spuštění na uživatele čeká hlavní menu. Nabidka menu je:

- *Rychlá hra.* Uživatel hraje bez nustnosti ukládat jméno a je proto možné tento scénář zvolit ihned po startu aplikace. Ovšem jeho herní výsledky nejsou ukládány.
- *Kariéra.* V tomto režimu je nejprve nutné vložit jméno (nebo zvolit z již vytvořených), ke kterému se potom ukládá nejlepší dosažený výsledek (skóre a čas).
- *Nastavení.* V nastavení si uživatel může měnit volby, zda-li chce mít během hry zapnutou hudbu, zvukové efekty, vibrace nebo ovládání pomocí akcelerometru.
- *Nápověda.* V nápovědě čeká na uživatele jednoduchý průvodce, co je cílem hry a jak hro ovládat pomocí jednotlivých prvků (dotyky, akcelerometr).

Pokud uživatel spustí hru (ať už rychlou nebo kariéru - rozdíl bude pouze v uvedení/neuvedení jména), uvidí obrazovku podobnou na obrázku 4.5.



Obrázek 4.5: Ukázka obrazovky hry

Pod navigačním panelem (viz 4.4.3) je v levé části hodnota skóre, pod ním v případě hry v kariéře jméno aktuálního hráče. Uprostřed je uplynulý čas a v pravé části aktuální úroveň. Zbytek obrazovky je určen samotné hře, ve které se skládají padající bloky.

4.4 Prvky aplikace

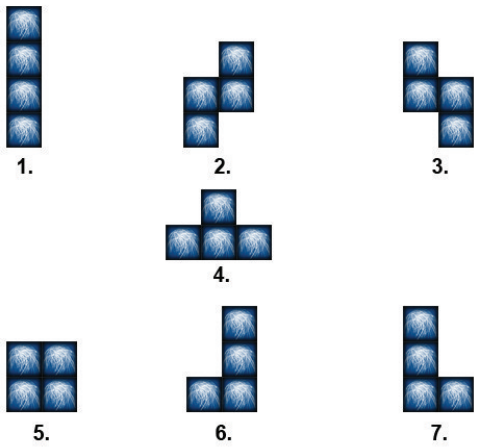
Vlastnosti hry jsou v tomto případě svým významem trochu syntetické. Během běžného návrhu aplikace nebudeme vymýšlet, jak vhodně zakomponovat schopnosti zařízení do nějakého programu. Je to jako bychom se snažili do textového editoru zapojit vibrace jen proto, že telefon umí vibrovat. Nicméně tento dokument slouží jako právě popis a ukázková implementace vlastností určitého zařízení. Proto je tato část nutná.

Nyní si tedy postupně popíšeme vlastnosti/funkce, které chceme do hry zapojit. Dále také, jak vhodně funkce spojit.

4.4.1 Padající bloky

Popis

Základní verze hry Tetris obsahuje sedm základních bloků (viz obrázek 4.6). Každý blok je tvořen čtyřmi malými čtverci. Tyto bloky padají dolů, dokud nenarazí na překážku. Každý blok je ještě možno otáčet kolem středu. Díky tvaru bloků a rotaci je zde velká variabilita uspořádání kostek na hracím poli.



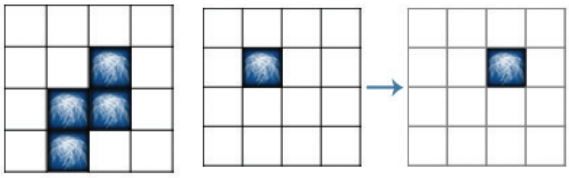
Obrázek 4.6: 7 základních bloků hry

Implementace

Aby bylo možné pracovat s celým blokem při pohybu dolů a zároveň při odstraňování kompletních řad s jednotlivými částmi, jeden blok je tvořen ze čtyř malých čtverců (nazvěme je například elementy). Tyto elementy jsou ve vizualizaci tvořeny obrázkem.

Každý blok má svoji matici o velikosti 4x4 prvky (viz obrázek 4.7a. Tato matice je tvořena `bool` hodnotami, kde `TRUE` nám říká, že v tomto poli má být umístěn element. `FALSE` nikoli. Výsledný tvar objektu je potom dán uspořádáním hodnot v této matici a tím i uspořádáním elementů v prostoru.

Jak je popsáno v kapitole 4.4.2, jeden prvek matice má pomyslné rozměry 16x16px. Z toho plyne, že i jeden element má rozměry 16x16px.



(a) Blok s maticí (b) Rotace bloku doprava

Obrázek 4.7: Padající blok

Rotace objektu

Pokud rotujeme s objektem, přeskládáme prvky v matíci podle směru dané rotace. Podle nového uspořádání projdeme čtyři jednotlivé elementy bloku a pomocí animace je přesuneme na nové pozice. Animace je v API určitá forma interpolace souřadnic mezi dvěma body. Nastavíme délku pohybu a koncové souřadnice, na které chceme daný objekt přesunout a tento objekt se plynule na tyto souřadnice přesune. Vznikne tak plynulé přeuspořádání prvku a zároveň poměrně zajímavý efekt. Logika rotace je zobrazena na obrázku 4.7b

Posun objektu

Pokud bychom měli matici bloku jako na obrázku, kde je na souřadnicích [1, 1] umístěn element, při rotaci doprava je umístěn na souřadnice [2, 2]. Souřadný systém bereme s počátkem [0, 0] v levém horním rohu. Změna rotované matice způsobí, že se musí prvek posunout na souřadnice [2*16, 2*16]. Obdobně se pracuje při posunu celého bloku dolů po obrazovce.

Objekt bloku obsahuje čítač aktuálního indexu X a Y. Nazvěme je například actualIndexX a actualIndexY. Tento index znamená umístění horního levého rohu matice v herní matici - viz 4.4.2. Pokud blok posuneme dolů, index Y se zvýší o jedna. Pokud jej posuneme do stran, index se zvýší o jedna při posunu doprava, naopak sníží při posunu doleva.

V tomto okamžiku víme umístění matice v poli. Zbývá tedy určit přesnou polohu jednotlivých elementů bloku, abychom je mohli postupně přesunout na nové souřadnice. Stejně jako každý blok, i každý element má svůj index, který určuje polohu v matici bloku. Nazvěme tyto indexy například indexElemX, indexElemY. Výpočet nové polohy každého ze čtyř elementů bloku (a tím i celkové změna polohy celého bloku na obrazovce) je dán následovně:

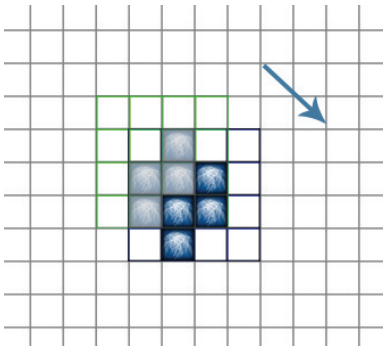
Příklad:

1. `[(actualIndexX + elemIndexX) * 16, ((actualIndexY + elemIndexY) * 16]`

Tento výpočet se musí provést při každém požadavku na změnu polohy. Mimo to se musí před každým pohybem řešit kolize (viz 4.4.2). Na obrázku 4.8 je vidět logika posunu bloku v herním poli. I přesto, že pracujeme s jednotlivými elementy zvlášť, díky přesným pozicím se jeví čtyři elementy jako jeden blok. Že se jedná o malé nezávislé části poznáme až při přeuspořádávání bloku při rotaci.

4.4.2 Hrací pole

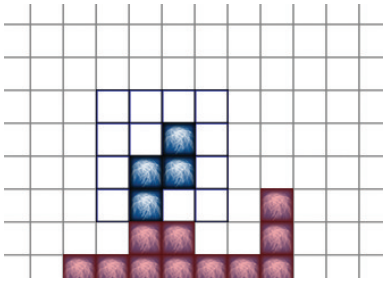
Naše aplikace má v horní části displeje zobrazeny 2 panely („status bar“ a navigační panel - viz 3.2.2). Z toho důvodu se oblast pro hru změnší ze 480px na pouhých 416. Herní pole má tedy rozměry 320 na šířku a 416 na výšku.



Obrázek 4.8: Posun bloku dolů a doprava v herním poli

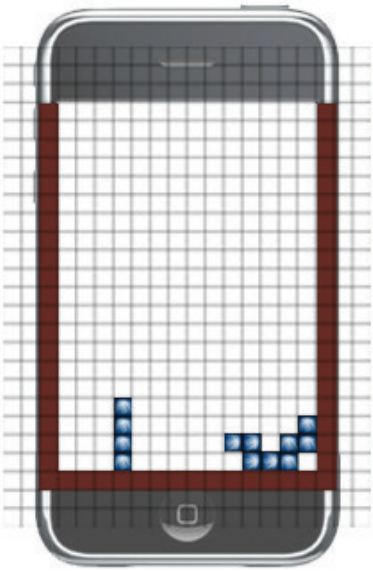
Herní pole je reprezentováno maticí, aby bylo možné řešit kolize a umístění prvků (viz dále). Aby byly prvky matice čtvercové, z výše uvedených rozměrů jsme zvolili, že jeden prvek matice má pomyslné rozměry 16x16px. Z toho plyne, že herní matice je velikosti 20x26 prvků.

V matici jsou opět hodnoty `TRUE` nebo `FALSE`. Respektive, pokud na aktuálních souřadnicích není umístěn žádný blok, je zde hodnota `FALSE`. Pokud je zde čtverec, je zde ukazatel na tento čtverec a tedy hodnota není `FALSE`. Pokud během hry posouváme blok po poli, vždy testujeme matici bloku s aktuální částí matice hry, ovšem musíme testovat o jedna dopředu v daném směru pohybu. Logickou operací `OR` nám potom výjde, jestli se na dané souřadnice může blok posunout nebo ne. K upřesnění. Pokud budou všechny prvky herní matice `FALSE`, potom celkový výsledek operace bude `FALSE` a kolize tedy není. Pokud bude jeden prvek herní matice `TRUE` ve stejném místě jako prvek bloku, operace bude `TRUE` a tedy nastala kolize. V tomto případě blok nemůžeme na tyto souřadnice posunout. Na obrázku 4.9 je tato situace pro lepší představu zobrazena.



Obrázek 4.9: Ukázka kolize

Pole musíme ovšem mít vzhledem k rozměrům matice bloku o něco větší. Pro představu, pokud bychom testovali pozici úplně v pravo a blok měl prvky umístěny pouze v levé části matice, dostaneme se tedy o 3 prvky matice bloku mimo obrazovku. Abychom nemuseli ošetřovat tuto událost (a tím i zpomalit výpočetní výkon aplikace), herní matice je tedy o 3 prvky větší než je potřeba. Nemůže se tedy stát, že se maticí bloku dostaneme mimo matici hry. Aby nedošlo k posunutí mimo obrazovku, herní matice má při vytvoření umístěny `TRUE` hodnoty prvků po okrajích a na spodní části. Tedy pokud bychom chtěli blok posunout mimo obrazovku, v místě konce obrazovky nastane kolize a program tak nedovolí posunout blok mimo. Logika je zobrazena na obrázku 4.10.



Obrázek 4.10: Okraje herního pole

(a) Navigační panel hry

(b) Navigační panel menu nastavení

Obrázek 4.11: Ukázka navigačních panelů

4.4.3 Navigační panel

Popis

Tento prvek uživatelského rozhraní je v telefonu prakticky na každé obrazovce menu. Pomocí elegantního způsobu umožňuje navigaci mezi jednotlivými obrazovkami. Uprostřed je vždy název aktuálního menu/obrazovky. Uživatel tedy ví, že zrovna je například v nastavení displeje. Dále v levé části je většinou tlačítko pro návrat na předchozí obrazovku. Ovšem nemusí to být pravidlem. V pravé části může být další tlačítko pro libovolnou akci. Například pro smazání vybraného prvku v seznamu na obrazovce, vlastnosti prvku a další.

Implementace

V naší aplikaci je tedy zobrazen na tomto panelu název aktuálního menu (kromě hry, kde by to postrádalo smysl). Vlevo je potom tlačítko pro návrat do předchozího menu. Ve hře je toto tlačítko zároven pro ukončení hry. V pravé části tlačítko není. Pouze ve hře, kde slouží pro pauzu/pokračování. Na obrázku 4.11a je vidět navigační panel pro obrazovku hry, obrázek 4.11b je pro menu nastavení.

4.4.4 Hudba, zvuky, vibrace

Aby bylo možné hru trochu oživit, umožňuje 4 různé efekty:

- *Přehrávání hudby.* Pro jednoduchost byl zvolen pouze jeden zvukový soubor pro hudbu, který se neustále opakuje.
- *Krátký zvukový efekt při dopadu na překážku.* Délka efektu je asi 1 sec.
- *Krátký zvukový efekt při zkompletování řady.* Dálka efektu opět asi 1 sec.
- *Krátký efekt vibrace.* Pokud blok narazí na překážku nebo je celá řada zkompletována, je spuštěna vibrace. Vibrace mají v API jednotnou délku (opět asi 1 sec.). Programátor tedy nemůže ovlivnit délku vibrace, protože pouze zavolá knihovní funkci, která vibraci spustí. Jediná, a zároveň trochu krkolomná, metoda, jak ovlivnit délku vibrace je opakovat vícekrát volání této funkce. Možná ale úpravu přinese nová verze OS.

Aktivitu všech výše popsaných efektů lze nastavit na sobě nezávisle v menu nastavení (viz obrázek 4.12).

Obrázek 4.12: Menu nastavení

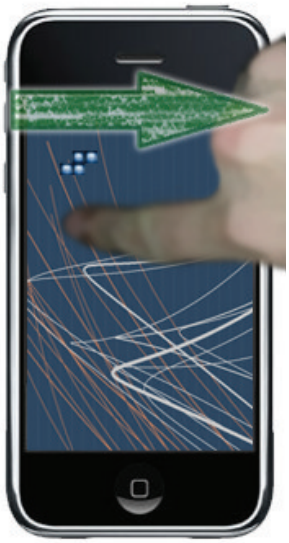
4.4.5 Multi-Touch, „Simple-Touch“ a poklepání

Popis

Mezi výsadu zařízení s dotykovým displejem je možnost ovládat aplikaci ne pomocí tlačítek napevno umístěných, ale pomocí gest prstů. iPhone v tomto směru umožňuje poměrně širokou paletu, jak s tímto prostředkem naložit. Počet gest pro ovládání umožňuje v rozmezí 1 až 5 dotyků. Tedy můžeme definovat chování aplikace až pro pět prstů. Je to samozřejmě jen „možnost“. Implementace a ergonomie tohoto chování by byla druhá věc. Běžně se tedy používají jeden nebo dva dotyky. Pro dva dotyky je velmi známé například využití pro zoom fotek oddalováním nebo přibližováním prstů. Dále umožňuje API detekci poklepů na displej. Tedy jeden nebo více.

„Simple-Touch“

V naší aplikaci definujeme chování pro gesto jedním prstem a pro gesto dvěma prsty. Pokud jedním prstem přejedeme po obrazovce například zleva doprava, posune se aktuální blok o jednu část hracího pole doprava (16px). Analogicky na druhou stranu. Blok samozřejmě stále padá, proto je nutné při každém pohybu řešit kolize ve směru dolů a v případě pohybu do strany, šikmo dolů na danou stranu. Pokud se blok nemůže daným směrem posunout, zkusí se test na kolize pouze ve směru dolů. Pokud je volno, posune se pouze dolů. Pokud je kolize, blok se na tomto místě zastaví. Princip tétu funkce je na obrázku 4.13. pozn. *„Simple-Touch“* není oficiální název této technologie na rozdíl od *Multi-Touch*.



Obrázek 4.13: Posun pomocí gesta jedním prstem

Multi-Touch

Gesto, kdy oběma prsty přejedeme zleva doprava, rotuje objekt doprava a naopak. Jako v předešlém případě je nutné opět řešit kolize, které mohou nastat v případě rotace. Rotace tedy proběhne tak, že se provede rotace nad maticí bloku. Poté se provede test s herní maticí, jestli nenastala kolize. Pokud nenastala, jednotlivé elementy bloku se přeskupí podle nového uspořádání matice bloku a pokračuje se v pohybu bloku dolů. Pokud ovšem kolize nastala, je provedena rotace nad maticí bloku zpět. V tomto případě se prvky nepřeskupují, protože uspořádání je stejné jako před rotací.

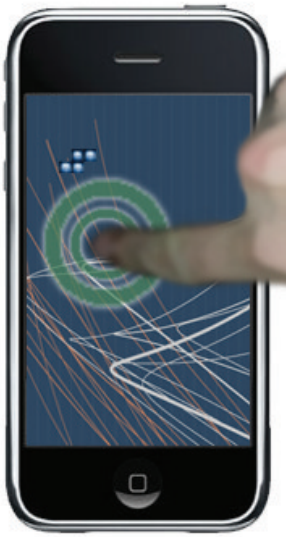
Poklepání

Poklepáním se v naší aplikaci detekuje pokyn pro rychlý pád padajícího objektu dokud nenarazí na překážku. Na obrázku 4.15 je zobrazen princip tohoto gesta.

Implementace

Implementace a vyhodnocení, jaká funkce se má provést na základě gesta, je poněkud složitější. Každý *Multi-Touch* začíná a končí ve většině případů jako *„Simple-Touch“*. A to

Obrázek 4.14: Rotace bloku pomocí gesta dvou prstů



Obrázek 4.15: Rychlý posun bloku dolů pomocí poklepání

z prostého důvodu. Není možné přesně ve stejný okamžik přiložit na displej oba prsty a současně je také pustit. iPhon OS při začátku a konci dotyku zavolá funkci `touchesBegan` resp. `touchesEnded`. V těchto funkcích je možné vyhodnotit kolik dotyků v aktuální moment je aktivích a získat jejich souřadnice. Abychom byli schopni korektně vyhodnotit *„Simple-Touch“*, *Multi-Touch* a *poklepání*, postupujeme podle schématu na obrázku **??**.

Příklady vyhodnocení gest

Mějme situaci, kdy se dotkneme pouze jedním prstem. Nejprve se tedy zavolá funkce `touchesBegan`. Zde si uložíme souřadnice bodu dotyku. Uživatel nyní posouvá prst po displeji obrazovky až k místu, kde dotyk ukončí. V tomto místě se zavolá funkce `touchesEnded`. Zde vyhodnotíme, že nastal pouze jeden dotyk, odečteme souřadnice posledního bodu od souřadnic prvního bodu a pokud je rozdíl X-ových souřadnic menší jak námi určený práh (délka pohybu, která má být vyhodnocena jako gesto), voláme funkce pro posun bloku doprava. Pokud je rozdíl větší, posuneme blok doleva.

Schéma vyhodnocení *„Simple-Touch“* vidíme na obrázku 4.16.

Obrázek 4.16: Schéma vyhodnocení „Simple-Touch“

Komplikovanější situaci je při gestu *Multi-Touchi*. Nejprve se zavolá funkce `touchesBegan` pro jeden dotyk. Zde vyhodnotíme, že se jedná o první dotyk a že před ním zatím žádný nebyl. Také si uložíme souřadnice tohoto bodu. Pokud se nyní zavolá metoda `touchesEnded`, víme, že se jedná o *„Simple-Touch“*. Pokud ovšem se místo toho zavolá metoda `touchesBegan`, víme, že už před chvílí byla zavolána metoda `touchesBegan` a proto se musí jednat o *Multi-Touch* (jde o druhý dotyk bez ukončení prvního). V tomto případě už čekáme na `touchesEnded`, kde vyhodnotíme rozdíl bodů a provedeme akci rotace. Vyhodnocení je naprosto stejné jako u *„Simple-Touch“*. Jen díky tomu, že víme, že jde o dotyk dvou prstů, provedeme místo pohybu doprava rotaci doprava.

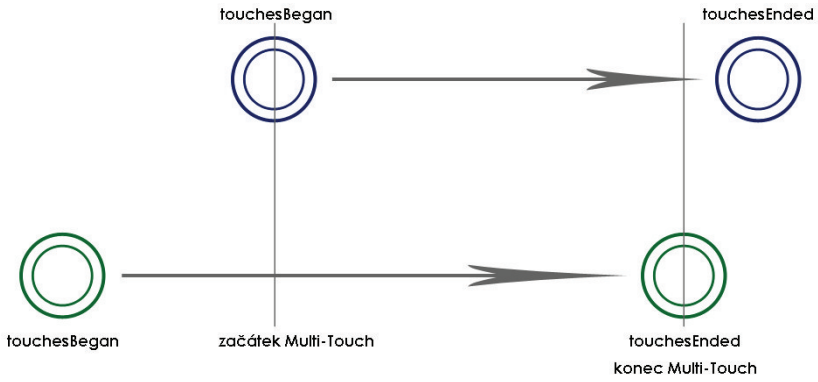
Schéma vyhodnocení *Multi-Touch* vidíme na obrázku 4.17.

Poklepání je naopak velmi jednoduché. Zde nabízí API detekci poklepání. Tedy můžeme určit, kolikrát uživatel poklepal na displej. Aby bylo možné odlišit náhodné dotyky od gesta *poklepání*, je zde ještě časový interval (0.4s), během kterého musí oba dotyky proběhnout.

4.4.6 Akcelerometr

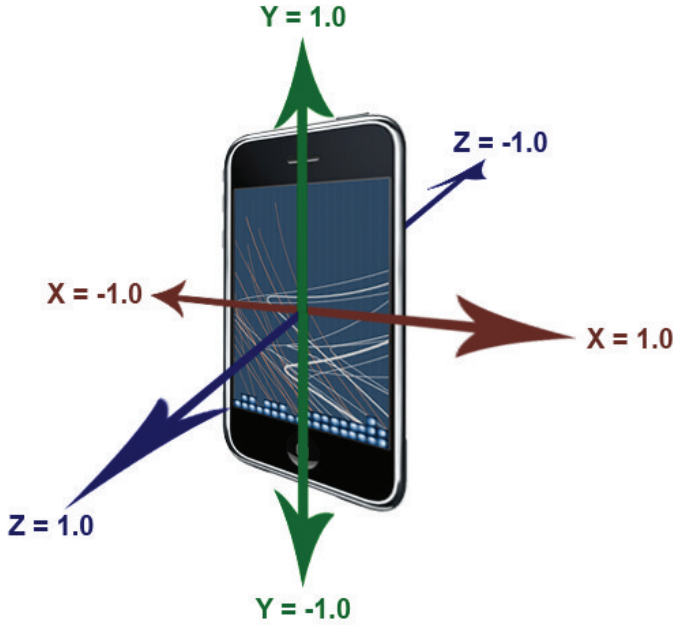
Popis

Akcelerometr patří rozhodně mezi nejznámější vlastnost iPhonu. Tento hardwarový prvek umožňuje telefonu zjišťovat svoji orientaci v 3D prostoru. Toho se dá využít například pro rotaci internetového prohlížeče při otočení telefonu o 90°. Jiné aplikace využívají tuto součástku sofistikovanějším způsobem. Například detekují zatřepání telefonem, dokonce i intenzitu třepání, a dokáží tak přiřadit gestům různé akce.



Obrázek 4.17: Schéma vyhodnocení Multi-Touch

Výchylka dané osy, jak už bylo napsáno, je brána jako odchylka osy od země. Pokud si vezmeme jako příklad osu Z, hodnoty výchylky telefonu ve svislé poloze bude kolem nuly. Pokud telefon položíme displejem nahoru, výchylka bude 1. Jednotlivé osy akcelerometru včetně kladných a záporných poloos vidíme na obrázku 4.18.



Obrázek 4.18: Osy akcelerometru

Pro gesta akcelerometru přináší nový iPhone OS 4.0 (květen 2010) lepší API. Umožňuje například detekci zatřepání, kterou musíme teď vyhodnocovat například podle odchylky jednotlivých os od námi zvoleného prahu v určitém časovém okamžiku.

Implementace

V naší aplikace používáme akcelerometr pro detekci náklonu od svislé osy. Tedy pokud telefon nakloníme doleva, padající blok se posune doleva a analogicky na druhou stranu. Toto gesto je alternativou gesta pro *„Simple-Touch“* (viz 4.4.5). Princip funkce můžeme vidět na obrázku 4.19.

Obrázek 4.19: Posun bloku do strany pomocí akcelerometru

Dále pak akci pro rychlé padnutí bloku dolů až po náraz na překážku vyvoláme, pokud telefon krátce vychýlíme displejem nahoru nebo dolů. Toto je opět alternativou na gesto *poklepání* (viz 4.4.5). Toto gesto můžeme vidět na obrázku 4.20.

Obrázek 4.20: Rychlý posun bloku dolů pomocí akcelerometru

Nutné ovšem poznamenat, že telefon musí být ve svislé poloze, protože vyhodnocujeme pouze pro boční náklon výchylku osy Y a pro náklon dopředu nebo do zadu výchylku osy Z (viz obrázek 4.20). Abychom tedy mohli hrát bezproblémů s jinou prostorovou orientací

telefonu, v nastavení je možnost vypnout detekci gest akcelerometrem. V tomto případě je ovládání pouze pomocí dotyků prstů.

4.4.7 Ukládání dat

Popis

Ukládání dat není v iPhone OS příliš specifikováno. Lépe řečeno ukládat data je možné mnoha způsoby. Nejkomplexnější je pomocí *SQLite*. V aplikaci si vytvoříme celou databázi (tabulky, vztahy, ...) a poté už jen pomocí příkazů data vkládáme nebo odebíráme. Tento způsob je samozřejmě doporučený pro větší oběm dat nebo pro složitější databázové operace.

Pokud potřebujeme uložit jen pár údajů (například nastavení), API obsahuje jednoduchý objekt `NSUserDefaults`. Jedná se v podstatě o malou databázi uživatelských dat. Zde se ukládají nastavení uživatele (myšleno profil v iPhonu, ne v naší hře), kde se uloží nastavení. Do této části má programátor také přístup a je zde možné uložit potřebné informace. Ovšem Apple nedoporučuje příliš velký objem dat. To by mohlo vést k tomu, že se některá data neuloží.

Mezi další možnosti ukládání patří například `Core Data, Object archivers, ...`

Implementace

Naše aplikace neukládá velké množství dat. Proto byl pro jednoduchost zvolen způsob pomocí `NSUserDefaults`. U uživatelů se ukládají jména, nejvyšší skóre, čas a index aktuálně zvoleného hráče. Právě z výše uvedeného důvodu velikosti dat pro `NSUserDefaults` je počet uživatelů omezen číslem deset. Tento limit byl zvolen jako kompromis mezi co nejmenším objemem dat a zároveň dostatečným počtem hráčů ve hře. Dále se uloží pouze v případě náhlého ukončení hry stav herního pole a aktuální skóre, čas a level.

4.4.8 Reakce na neočekávané události

Popis

Předešlé prvky, které jsme si probrali jsou prvky rámcově základní, které se přímo dotýkají uživatele. Ovšem, jak již bylo řečeno dříve, aplikace pro mobilní zařízení musí umět i reagovat na události neočekávané. A to je například příchozí hovor během hraní nebo hru uživatel sám náhle ukončí. Pokud se toto stane, uživatel by měl mít možnost pokračovat v hraní po opětovném spuštění aplikace. Proto je nutné při „pádu“ aplikace si uložit aktuální stav (jestli je hra v kariéře, v jakém stavu je herní pole, herní čas, skóre, ...) a při opětovném startu aplikace tyto data obnovit.

Implementace

Implementace není kupodivu nikterak složitá. Jak jsme si řekli v kapitole 4.2.1, při startu aplikace je jedna z prvních funkcí zavolána funkce `applicationDidFinishLaunching`. V té si mimo vytvoření okna a dalších nastavení nutných pro chod aplikace můžeme i načíst data (stav herního pole, skóre a další údaje), která jsme si uložili, pokud byla aplikace při minulém spuštění přerušena (během hraní tlačítkem HOME nebo například hovorem. Tato data musíme na začátku aplikace obnovit. Inicializujeme tedy hru s počátečními hodnotami

ne nula, ale uloženými a zobrazíme místo hlavní obrazovky menu obrozovku hry. Uživateli se tak jeví, že pokračuje kde skončil.

Obdobná situace, ale poněkud jednodušší, je při konci aplikace. Jak se volá na začátku aplikace funkce `applicationDidFinishLaunching`, těsně před skončením aplikace se volá podobná funkce `applicationWillTerminate`. Zde je tedy vhodné místo k uložení dat (například uživatelů) aplikace, ale také právě stavu hry, pokud byla náhle přerušena.

4.4.9 Lokalizace

Popis

Mezi další prvek, který je v aplikace implementován patří lokalizace. Jazyková lokalizace aplikací pro iPhone je takřka nutností zvláště pro neanglicky mluvící země. Protože aplikaci zvěřejňujeme na App Store, je povinností mít aplikaci lokalizovanou do angličtiny. Další jazyky jsou pro nás volitelné. Naše konkrétní aplikace je tedy lokalizována do angličtiny a češtiny. Existenci lokalizace si uživatel v první chvíli nemusí vůbec uvědomit. Aplikace se chová tak, jak očekává. Pokud máme jako jazyk telefonu zvolenu angličtinu, aplikace je komplentě v angličtině. Analogicky pro češtinu. Pokud má uživatel zvolen jazyk například japonštinu, aplikace bude ve svém výchozím jazyce, kterým je angličtina. Je to velmi pohodlné, protože uživatel nemusí nikde nic hledat a nastavovat. Programy se chovají podle celého systému.

Implementace

Jak již bylo popsáno v kapitole 3.4.2, programy můžeme vyvíjet buď čistě pomocí *Xcode* (funkčí kód + uživatelské rozhraní) nebo kombinací *Xcode* (kód) a *Interface Builder* (uživatelské rozhraní). Podle toho, jaký přístup zvolíme, je odvozen i způsob lokalizace naší aplikace.

Apple nabízí vývojářům dva konzolové programy obsažené v balíku SDK - *genstrings* a *ibtool*. Oba si stručně probereme

Genstrings

Genstrings slouží pro vytváření lokalizovaných řetězců, které dokáže najít v kódu. Řetězce musí mít syntaxi, jak je vidět na 4.21.

```
NSString *navBarTitle = NSLocalizedString(@"Settings title",@"Title for settings page");
```

Obrázek 4.21: Ukázka syntaxe lokalizovaného řetězce

Pokud spustíme *genstrings* v následujícím tvaru:
`genstrings ./Classes/*.m` (musíme mít aktuální adresář zvolen adresář projektu), vznikne ve složce projektu soubor *Localizable.strings*, který přiřadíme do *Resources* projektu. Po jeho lokalizaci (tím je myšleno vytvoření dvou jazykových verzí, viz příloha *A* - tutoriál), vzniknou složky s jednotlivými jazykovými verzemi, ve kterých jsou dva stejné textové soubory. Tyto soubory už jen přeložíme a aplikace potom bude brát dané přeložené řetězce podle aktuálního jazyka telefonu (viz 4.22a a 4.22b.

```
/* Title for about page */
"About" = "Nápověda";

/* MainLabel for accelerometer setting */
"Accelerometer title" = "Akcelerometr";

/* Add, remove and choose name for career */
"Career title" = "Kariéra";

/* Sublabel for accelerometer setting */
"Detail accelerometer settings" = "Gesta akcelerometrem.";
```

(a) Soubor pro českou lokalizaci

```
/* Title for about page */
"About" = "Help";

/* MainLabel for accelerometer setting */
"Accelerometer title" = "Accelerometer";

/* Add, remove and choose name for career */
"Career title" = "Career";

/* Sublabel for accelerometer setting */
"Detail accelerometer settings" = "Accelerometer gestures";
```

(b) Soubor pro anglickou lokalizaci

Obrázek 4.22: Ukázka lokalizace řetězců

IBtool

V případě nástroje *ibtool* je situace obdobná. Opět je to program konzolový a slouží pro pohodlnou práci při lokalizaci *xib* souborů (soubory uživatelského rozhraní). Nejprve je nutné vytvořit ty obrazovky, které chceme lokalizovat, lokalizované (viz příloha *A* - tutoriál). Poté vzniknou ve složce *Classes* (zde jsou zdrojové kódy jednotlivých modulů) opět složky podle jazyků, které jsme zvolili. V každé složce je nutné nyní spustit program *ibtool* v následujícím tvaru:
`ibtool --generate-strings-file cs.lproj/NázevSouboruProRetezce.strings cs.lproj/NazevZdrojovehoXibSouboru.xib`.
To způsopbí, že *ibtool* extrahuje všechny řetězce z jednotlivých xib souborů a vytvoří soubor *NazevZdrojovehoXibSouboru.strings*. V tom je pro každý prvek uživatelského rozhraní řetězec, který je nutno přeložit podle dané lokalizace. Ukázka obou souborů (pro českou a anglickou lokalizaci) vidíme na 4.23a a 4.23b.

Oproti *genstrings* se zde postup ale liší. Po přeložení souborů musíme opět nahrát textové popisky zpět do xib souboru. To uděláme pomocí spištění *ibtool* ve formátu:
`ibtool --strings-file cs.lproj/ZdrojovypPrelozenySouborSRetezci.strings --write cs.lproj/CilovyXibSoubor.xib cs.lproj/ZdrojovyXibSoubor.xib`

Pokud bychom měli rozsáhlejší projekt, je samozřejmě i tento způsob poměrně něšikovný. Musíme vždy pro každý sobor přepisovat cesty a názvy. Proto, vzhledem k poměrně monotonní funkci, by bylo dobré postup extrakce a zpětného vložení řetězců automatizovat (například pomocí skriptu).

Samozřejmě práce s ibtool se může zdát poněkud krkolomná. Pokud vytvoříme lokalizace xib souborů, rovnou je můžeme otevřít v *Interface Builderu* a přepisovat popisky na prvcích uživatelského rozhraní. Ale pokud bychom měli rozsáhlejší aplikaci s mnoha prvky, bylo by to velmi pracné a náročné. Navíc, pokud nechceme překládat prvky sami (nebo nemůžeme například z důvodu mnoha jazykových verzí), je velmi pohodlné vzít textové soubory a

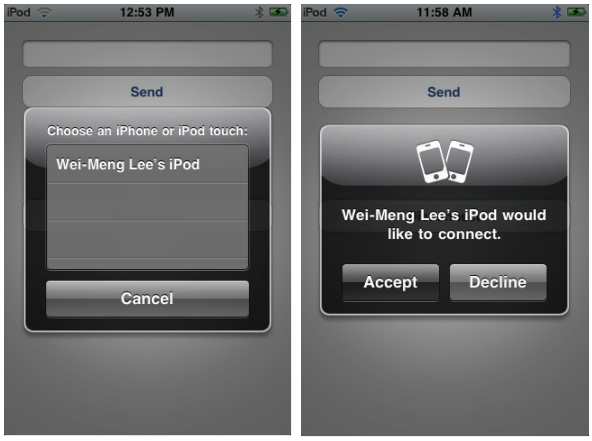
```
/* Class = "IBUIButton"; normalTitle = "Add player"; ObjectID = "114"; */
"114.normalTitle" = "Přidat hráče";

/* Class = "IBUIButton"; normalTitle = "Change player"; ObjectID = "115"; */
"115.normalTitle" = "Změnit hráčer";

/* Class = "IBUIButton"; normalTitle = "Start game"; ObjectID = "116"; */
"116.normalTitle" = "Spustit hru";

/* Class = "IBUILabel"; text = "Score:"; ObjectID = "125"; */
"125.text" = "Score:";
```

(a) Soubor pro českou lokalizaci xib souboru

```
/* Class = "IBUIButton"; normalTitle = "Add player"; ObjectID = "114"; */
"114.normalTitle" = "Add player";

/* Class = "IBUIButton"; normalTitle = "Change player"; ObjectID = "115"; */
"115.normalTitle" = "Change player";

/* Class = "IBUIButton"; normalTitle = "Start game"; ObjectID = "116"; */
"116.normalTitle" = "Start game";

/* Class = "IBUILabel"; text = "Score:"; ObjectID = "125"; */
"125.text" = "Score:";
```

(b) Soubor pro anglickou lokalizaci xib souboru

Obrázek 4.23: Ukázka lokalizace xib souborů



(a) Okno výběru zařízení ke spojení

(b) Potvrzení nabídnutého spojení

Obrázek 4.24: Ukázka oknen při spojování bluetooth

poslat je na překlad. S xib soubory by to bylo komplikovanější.

4.4.10 Bluetooth

Bluetooth je v dnešní době velmi populární v mobilních zařízeních. Umožňuje tak díky propracovanému standardu a široké podpoře od výrobců párovat mezi sebou mnoho zařízení. Mezi nejzákladnější příklad využití patří spárování bezdrátové handsfree sady s telefonem, počítače, jiného telefonu nebo třeba tiskárny.

API systému nabízí velmi propracovanou podporu práce s Bluetooth. Programátor tak jen vyvolá okno (například jako událost na stisk tlačítka). Uživatel si v něm vybere dostupné zařízení v dosahu. Druhému zařízení se objeví podobné okno, které nabízí potvrzení spojení. Tyto akce jsou zobrazeny na obrázcích 4.24a a 4.24b. Pokud tak uživatel učiní, vytvoří se spojení mezi těmito dvěma zařízeními (obecně jich ale může být více) a poté je již možné posílat a získávat zprávy.

Obdobné spojení by fungovalo, i kdybychom vytvářeli spojení mezi telefony přes wi-fi nebo dokonce přes internet. Takže není problém udělat hru, ve které proti sobě budou hrát hráči ze dvou zemí.

Samozřejmě ale je nutné řešit více situací, které během spojení mohou nastat. Například při vytvoření spojení ho jeden z hráčů ukončí (ať už záměrně nebo ne) nebo spojení hráč odmítne nebo například během vytváření spojení nás někdo jiný bude žádat o spojení. Veškeré stavy musíme korektně ošetřit a to také bude bráno v potaz při schvalovacím procesu Applu. Obecně aplikace pracující s komunikací (internet, bluetooth, ...) mají schvalovací dobu u Applu delší právě z důvodu nutnosti testovat různé stavy během omunikace.

Pro podrobnější prostudování je vhodný materiál viz [2].

V naší aplikaci nebyla bezdrátová komunikace implementována.

Kapitola 5

Závěr

V této práci bylo provedeno seznámení s vývojem aplikací pro telefon Apple iPhone. Protože se jedná o tutoriál, je tedy cílený na široké spektrum čtenářů, kteří mají ale různé úrovně povědomí jak o přístroji, tak o vývoji aplikací.

Aby bylo možné snížit rozdíly v úrovni znalostí čtenářů, je úvodní část věnována obecnému popisu přístroje, jeho architektuře, vývojářským prostředkům a distribuci výsledných programů. V místech, kde není možné tyto části v rámci rozsahu práce detailněji probírat, jsou uvedeny zdroje na materiály, kde si může čtenář danou problematiku prostudovat.

Dále je proveden návrh aplikace, která má za cíl demonstrovat funkční prvky telefonu. Aby bylo možné jednotlivé prvky zapojit do smysluplného celku, byla pro tento účel zvolena hra. Pro lepší představu o jejich smyslu a funkčnosti je čtenář s jednotlivými částmi nejprve seznámen obecně a poté následuje návrh implementace daného prvku. Detaily implementace včetně zdrojových kódů a detailních popisů nejsou v této práci rozebírány. K tomuto účelu slouží příloha *A* - tutoriál a příloha *B* - softwarová dokumentace. V příloze *A* najde čtenář podrobnější popis implementace a zejména příloha *B* obsahuje velmi podrobně dokumentovaný projekt. V příloze *C* - projekt, najde čtenář samotný funkční projekt pro vývojovéí prostředí *Xcode*. Ten stačí otevřít a přeložit. Díky tutoriálu, projektu a softwarové dokumentaci získá čtenář maximální možné zdroje, ze kterých lze při studiu programování aplikací pro iPhone vycházet.

Pro tento telefon existuje velké množství různých programů. Zvláště hry jsou velmi populární. Na App Store (viz [1]) je možné stáhnout mnoho programů podobných našemu vzorovému. Ale byly vytvořeny za jiným účelem (placené programy pro zisk nebo zdarma pro sebeprezentaci a reklamu). Naším cílem bylo prezentovat vlastnosti telefonu a naučit se s nimi pracovat. Proto srovnání nebo hledání konkurenčních aplikací nemá v této situaci příliš smysl.

Protože iPhone Apple se svojí politikou drží co nejvíce stranou nelegálního zacházení (odblokování, nahrávání aplikací z jiných zdrojů než je App Store a další), tato práce postupuje čistě podle pravidel Applu. Není proto nutný žádný zásah do telefonu, který by měl za následek porušení podmínek Apple Inc. Protože po registraci do vývojářského programu má uživatel možnost si stáhnout SDK potřebné pro vývoj aplikací, není nutné nelegální řešení ani v tomto směru.

Aplikace může sloužit mnoha uživatelům jako typový projekt, neboli základní prvek vytvářené aplikace. Potom už stačí jen změnit vzhled nebo chování aplikace, přidat vlastní prvky nebo odebrat zakomponované a tím je možné vývoj velmi zrychlit.

Literatura

[1] Apple: AppStore [online]. http://www.appstore.com/, 2008-07 [cit. 2010-1-1].

[2] Apple: Game Kit Programming Guide [online]. http://developer.apple.com/iphone/library/documentation/GameKit/Reference/GameKit_Collection/GameKit_Collection.pdf, 2009-05-28 [cit. 2010-05-16].

[3] Apple: Apple Developer [online]. http://developer.apple.com/, 2009-10-19 [cit. 2010-05-16].

[4] Apple: Cocoa Fundamentals Guide [online]. http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaFundamentals.pdf, 2009-10-19 [cit. 2010-1-1].

[5] Apple: iPhone OS Technology Overview [online]. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf, 2009-10-19 [cit. 2010-1-1].

[6] Apple: Your First Application [online]. http://developer.apple.com/iphone/library/documentation/iPhone/Conceptual/iPhone101/iPhone101.pdf, 2009-10-19 [cit. 2010-1-1].

[7] Apple: iPhone Human Interface Guidelines [online]. http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf, 2009-11-20 [cit. 2010-1-1].

[8] Apple: Preparation - iPhone Provisioning Portal - Apple Developer [online]. https://developer.apple.com/iphone/manage/distribution/index.action, 2010 [cit. 2010-05-17].

[9] Microsoft: Accessors [online]. http://msdn.microsoft.com/en-us/library/aa287786%28VS.71%29.aspx, 2009-10-19 [cit. 2010-1-1].

[10] Wikipedia: Garbage collector [online]. http://cs.wikipedia.org/wiki/Garbage_collector, 2010-05-14 [cit. 2010-05-17].

Seznam obrázků

Dodatek A

Obsah CD

- Technická zpráva (pdf dokument a zdrojové soubory)
- Tutoriál (pdf dokument a zdrojové soubory)
- Softwarová dokumentace (html a pdf dokument)
- Zdrojové kódy ukázkové aplikace
- info.txt (popis obsahu CD)