PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

Aplikace pro iOS a Mac OS X

2013 Jakub Mejtský

**Anotace**

*Mac OS X a iOS jsou jedny z nejznámějších a nejpoužívanějších platforem. Předmětem práce je najít možné způsoby pro psaní co nejvíce platformově nezávislého kódu, implementace aplikací pro iOS a Mac OS X s důrazem na multiplaformnost. Práce obsahuje seznámení s platformami, popis použitého postupu, uživatelskou a programátorskou příručku, analýzu možných rozšíření projektu a popis alternativních způsobů vytváření multiplatformních aplikací.*

Rád bych poděkoval Doc. RNDr. Michalu Krupkovi, Ph.D za trpělivost, rady a poskytnutí školní vývojářské licence pro potřeby projektu.

# Obsah

**1. Úvod** **7**
1.1. Cíle projektu . . . . . 7

**2. Seznámení s platformami** **7**
2.1. Platforma Mac OS X . . . . . 7
2.2. Platforma iOS . . . . . 10

**3. Společné a rozdílné prostředky** **12**
3.1. Framework Foundation . . . . . 12
3.2. Framework Core Data . . . . . 12
3.3. iCloud . . . . . 13
3.4. AppKit a UIKit . . . . . 14
3.5. Game Center . . . . . 14
3.6. Důležité pojmy . . . . . 15

**4. Použitý postup, architektura a prostředky** **15**

**5. Uživatelská příručka** **17**
5.1. iOS aplikace . . . . . 17
5.2. Mac OS X aplikace . . . . . 21

**6. Programátorská příručka** **25**
6.1. Struktura aplikace . . . . . 25
6.2. iOS aplikace . . . . . 25
6.3. Mac OS X aplikace . . . . . 29

**7. Možná rozšíření projektu** **31**

**8. Alternativní postupy, architektury a prostředky** **32**
8.1. Využití řídících tříd a protokolů . . . . . 33
8.2. Použití OpenGL . . . . . 33
8.3. Použití HTML5 a JavaScriptu . . . . . 34
8.4. Použití jiných programovacích jazyků . . . . . 34
8.5. Xamarin MonoTouch Project . . . . . 35
8.6. Embarcadero RAD Studio . . . . . 36

**Závěr** **40**

**Conclusions** **41**

**Reference** **42**

**A. Obsah přiloženého CD** **43**

# Seznam obrázků


1. Ukázka jednoduchého modelu Core Data . . . . . 13
2. Ilustrace služby iCloud . . . . . 14
3. Zobrazené hlavní menu hry . . . . . 18
4. Mezifáze zadávání tahu. . . . . 18
5. Rámec vzhledu pro ukládání v režimu editace . . . . . 19
6. Rámec vzhledu pro ukládání v režimu mazání . . . . . 19
7. Rámec vzhledu pro načítání . . . . . 20
8. Rámec vzhledu pro statistiku . . . . . 20
9. Rozhraní zařízení iPad v režimu landscape . . . . . 20
10. Rozhraní zařízení iPad v režimu portrait . . . . . 21
11. Zobrazení rámce vzhledu pro načítání zařízení iPad . . . . . 21
12. Zobrazení herní statistiky na zařízení iPad se zobrazeným menu aplikace . . . . . 22
13. Vytvoření nové hry . . . . . 22
14. Mezifáze zadávání tahu . . . . . 23
15. Okno načtení hry z úložiště aplikace . . . . . 24
16. Ukázka HTML5 aplikace v prohlížeči Safari . . . . . 34
17. Ukázka HTML5 aplikace pomocí nativní aplikace . . . . . 34
18. Editace zdrojového kodu potomka třídy `UIViewController` v prostředí Xamarin Studio . . . . . 37
19. Návrh UI třídy v prostředí Xcode z obrázku 18. . . . . 37
20. Návrh UI v prostředí RAD Studio . . . . . 38
21. Exportovaný projekt z obrázku 20. do prostředí Xcode . . . . . 39
22. Aplikace z obrázku 21. spuštěná na iPhone simulátoru . . . . . 39

# Seznam tabulek

1. Frameworky vrstvy Core Services platformy Mac OS X . . . . . 8
2. Frameworky Core Services platformy iOS . . . . . 11

# 1. Úvod

Práce se zabývá platformami Mac OS X a iOS. První část se zabývá seznámením s platformami z hlediska programátora, dále technologiemi a ostatními důležitými záležitostmi pro projekt. Další část je věnována samotné implementaci aplikací dle požadavků. Práce popisuje použitý postup, který je uplatnitelný při jiných projektech. Pozornost je věnována i alternativním možnostem architektury a dostupným možnostem, jak lze aplikace tvořit.

## 1.1. Cíle projektu

Cílem projektu bylo vytvořit dvě nezávislé aplikace pro platformu iOS a Mac OS X. Při vytváření stuktury aplikací měl být kladen důraz zejména na víceúčelnost kódu. Cílová architektura předem stanovena nebyla. Předlohou měla být jednoduchá desková hra. Platformy mají společné prvky. Bylo vhodné používat společné prostředky a co nejvíce se vyhnout používání prostředků, které nejsou k dispozici na druhé platformě.

# 2. Seznámení s platformami

## 2.1. Platforma Mac OS X

Platforma Mac OS X [4] je operační systém společnosti Apple Inc. Je určen především pro počítače Macintosh. Jeho největší výhoda tkví v odladěnosti software na hardware. Jeho historie sahá až do starého Mac OS, který byl provozován na počítačích Macintosh před nástupem Mac OS X. Přejímá velkou část z operačního systému NeXTSTEP [4] společnosti NeXT. X v názvu Mac OS X označuje určitý druh příslušnosti k Unixu stejně jako je tomu u systému NeXT. X je možné považovat i za návaznost na poslední verzi Mac OS 9. Systém obsahuje hybridní jádro XNU (X is Not Unix) založené na mikrojádře Mach a kompatibilním obalu s FreeBSD [4]. Jádro se souhrně nazývá Darwin. Celá historie systému Mac OS X je dostupná v [4] na straně 3 - 9.

Systém Mac OS X se dá rozdělit do 4 základních vrstev, na které může programátor narazit při tvorbě aplikací pro tento systém. Postupně bude rozebrána každá vrstva. Podrobnější informace na [2]. Jejich výčet je Core OS, Core Services, Media, Cocoa (Application).

### Vrstva Core OS

Vrstva Core OS poskytuje nízkoúrovňové služby, týkající se počítačového hardware a sítí. Obsahuje následující frameworky[1]:

- Accelerate poskytující hardwarovou akceleraci některých složitých operací, jako například matematické operace video operace, vektorové operace a jiné,
- OpenCL poskytující výkon GPU pro nejuniverzálnější výpočetní techniky,
- Open Directory umožňující přístup k informacím o uživatelech, skupinách, počítačích, tiskárnách a spoustě dalšímu v existujícím síťovém prostředí,
- System Configuration.

### Vrstva Core Services

Vrstva Core Services obsahuje základní služby pro aplikace. Této vrstvy většinou využívají vyšší vrstvy. Do této vrstvy se řadí služba iCloud, která bude rozebrána později. Dále se zde nachází funkce Automatic Referention Counting poskytující automatickou správu paměti, služba Bonjour pro komunikaci mezi zařízeními, služba Time Machine pro zálohování dat a spousta dalších. Tabulka 1. vypisuje některé frameworky poskytované vrstvou Core Services. Nejdůležitější budou podrobněji rozebrány později.

| **Název frameworku** | **Použití** |
|---|---|
| Accounts | informace o uživatelských účtech |
| Address Book | využívání kontaktů uživatele |
| Core Data | ukládání dat pomocí databáze sqlite |
| Core Foundation | práce s řetězci, kalendářními daty, URL, vlákny, porty, ... |
| Event Kit | přístup k událostem v kalendáři |
| Foundation | objetive-C wrappery pro funkce z Core Foundation |
| Store Kit | přístup k Mac App Store a možnost nákupů |

Tabulka 1. Frameworky vrstvy Core Services platformy Mac OS X

### Vrstva Media

Vrstva Media layer obsahuje frameworky pro práci s grafikou, zvukem a videem. Všechny jednotlivé části byly navrženy pro snadné vytváření aplikací se skvělým vzhledem. Mezi grafické technologie se dá zařadit:

[1]Framework je dle [5] kolekce tříd, metod, funkcí, podpůrných zdrojů a dokumentace logicky seskupených tak, aby byl vývoj programů snazší.

- Core Graphics (Quartz 2D) sloužící k vykreslování 2D objektů a obrázků,
- Core Animation sloužící k vykreslování sofistikovaných animací,
- OpenGL sloužící k hardwarové akceleraci grafiky,
- Core Text sloužící k vykreslování textových řetězců,
- Image I/O sloužící k používání známých grafických formátů.

Mezi zvukové technologie patří frameworky:

- The Media Player Framework sloužící k přístupu a přehrávání z iTunes knihovny uživatele,
- AV Foundation pro záznam a přehrávání,
- Core Audio nabízející jednoduché a sofistikované rozhraní ro práci se zvukem.

Mezi videotechnologie lze zařadity tyto frameworky:

- Media Player pro přehrávání a přístup ke knihovně videí uživatele,
- AV Foundation pro záznam a přehrávání,
- Core Media poskytující nízkoúrovňovou práci s videi.

### Vrstva Cocoa (Application)

Vrstva Cocoa je primárně zodpovědná za komunikaci s uživatelem, reakce na akce. Obsahuje práci s celoobrazovkovým režimem aplikací, funkcí Auto Save, notifikačním centrem a mnohým dalším. Frameworky vrstvy Cocoa jsou:

- AppKit obsahující implementaci oken, dialogů, nabídek a všech ovládacích prvků,
- Game Kit umožňující aplikacím používat služby Game Center a komunikace pomocí protokolu Bonjour,
- Preference Panes umožňující správu nastavení systému,
- Security Interface umožňující používání prvků uživatelského rozhraní provádějících bezpečnostní funkce.

## 2.2. Platforma iOS

Platforma iOS je do jisté míry platformou Mac OS X, ale zásadně se liší od platformy Mac OS X. Největším rozdílem je využívání ARM procesorů namísto procesorů Intel. Platforma iOS je mnohem více uzavřená. Je to zejména z důvodu bezpečnosti. Původně byla určena pouze pro mobilní telefony iPhone. Později se dostala i do další mobilních zařízení, jako je iPod Touch, iPad a nejnověji AppleTV. Původní oficiální název byl iPhone OS. Ten byl následně zkrácen na iOS. Základem systému je jádro Darwin. Uživatelské rozhraní systému iOS je orientováno na dotyková zařízení. Neposkytuje plnou funkcionalitu systému Mac OS X. Naopak jsou zde přidány některé funkce oproti systému Mac OS X. Systém iOS se dá rozdělit do 4 základních vrstev využívaných programátory. Jejich výčet je Core OS, Core Services, Media Layer, Cocoa Touch.

Nyní rozebereme každou vrstvu zvlášť. Hlavním částem vrstev bude věnována větší pozornost později. Podrobnější informace o frameworcích poskytovaných pro systém iOS je k dispozici na [2].

### Vrstva Core OS

S vrstvou Core OS se běžný programátor nesetká. Využívají ji všechny vyšší vrstvy systému. Obsahuje 3 nejdůležitější frameworky:

- Accelerate poskytující funkce pro práci s matematickými funkcemi optimalizovanými na daném hardware,
- External Accessory sloužící k komunikaci s externími zařízeními pomocí BlueTooth, třicetipinového konektoru a nově také Lighting konektoru,
- Security sloužící k používání nízkoúrovňových bezpečnostních záležitostí, jako používání svazku klíčů či uživatelských certifikátů a jiné.

### Vrstva Core Services

Vrstva Core Services obsahuje základní systémové služby, které využívají všechny aplikace. Pokud ne přímo, tak je využívají vyšší vrstvy systému iOS. Do této vrstvy se řadí i služba iCloud, která bude podrobněji rozebrána dále. Frameworky poskytované vrstvou Core Services se dají shrnout do tabulky 2. Nejsou zde uvedeny všechny frameworky. Nejdůležitější budou podrobněji zmíněny dále.

### Vrstva Media layer

Vrstva media pro platformu iOS je prakticky totožná s jejím protějškem z platformy Mac OS X. Jedním z rozdílů je, že na platformě iOS je využívána grafická knihovna OpenGL ES místo klasické knihovny OpenGL.

| **Název frameworku** | **Použití** |
|---|---|
| Address Book | využívání kontaktů uživatele |
| Core Data | ukládání dat pomocí databáze sqlite |
| Core Foundation | práce s řetězci, datumy, URL, vlákny, porty, ... |
| Core Location | hledání aktuální geografické polohy uživatele |
| Core Media | nízkoúrovňový přístup k audio/video |
| Core Telephony | přístup k informacím o mobilní síti |
| Event Kit | přístup k událostem v kalendáři |
| Foundation | objetive-C wrappery pro funkce z Core Foundation |
| Store Kit | přístup k iTunes Storu a možnost nákupů |

Tabulka 2. Frameworky Core Services platformy iOS

Do vrstvy Media layer se řadí i služba AirPlay pro streamování zvuku a obrazu do kompatibilních zařízení. Zdrcadlit obraz i zvuk je možné ze zařízení se systémem iOS 5 a vyžší případně Mac OS X 10.8 do zařízení AppleTV 2 generace a novější, případně do počítače PC nebo Mac s nainstalovaným programem AirServer. Služba využívá frameworků pro práci se zvukem a obrazem.

### Vrstva Cocoa Touch

Vrstva Cocoa Touch obsahuje klíčové frameworky pro tvorbu iOS aplikací. Vrstva definuje základní aplikační infrastruktury, podporu klíčových funkcí, jako je Multitasking, dotykové ovládání, Push Notifikace, standartní systémové rámce, sdílení souborů, ochrana dat a mnoho dalších vysokoúrovňových služeb. Vrstva obsahuje následující frameworky:

- The Address Book UI framework obsahující rozhraní pro přístup ke kontaktům uživatele a jejich úpravu,
- Event Kit UI Framework obsahující rozhraní pro správu událostí a jejich úpravu v kalendáři uživatele,
- iAd Framework umožňující zobrazení reklamy v aplikacích,
- Game Kit Framework umožňující komunikaci mezi zařízeními pomocí protokolu Bonjour a využívání služby Game Center,
- Map Kit Framework umožňující zobrazit mapové podklady a umístění,
- Message UI Framework umožňující vytvářet a odesílat e-maily a sms zprávy.

# 3. Společné a rozdílné prostředky

Platforma iOS a Mac OS X sdílí mnoho společných prostředků. Některé jsou jen pozměněné pro chod na jiných typech zařízení a obsahují prvky uzpůsobené typu ovládání pro konkrétní zařízení. Následující kapitola popíše ty prvky, které jsou důležité pro projekt. V poslední části kapitoly jsou uvedeny pojmy, vyskytující se v práci.

## 3.1. Framework Foundation

Framework Foundation je malý ale nejdůležitější framework, který chtě nechtě musí využít každý programátor pro platformu iOS a Mac OS X při použití programovacího jazyka Objective-C. Foundation je k dispozici na obou platformách. Foundation v doslovném překladu znamená základ. Jedná se o balík tříd, metod a funkcí zajišťující základní funkcionalitu. Obsahuje třídy pro práci s poli, řetězci, kalendářními daty, čísly a mnoho dalšího. Třídy tohoto frameworku jsou platformově nezávislé. Jejich použití na obou platformách není limitováno a kódy jsou plně přenositelné.

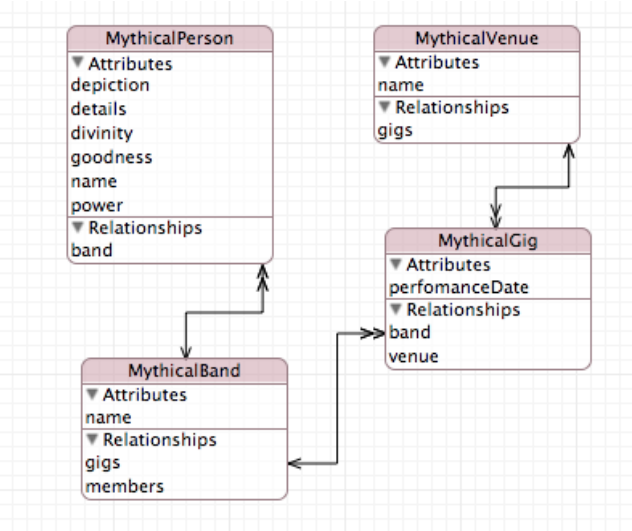
## 3.2. Framework Core Data

Core Data je framework poskytující výkonou a rychlou objektovou nádstavbu nad relační databází typu SQLite. Je k dispozici jak pro platformu iOS, tak pro platformu Mac OS X. Core Data je založen na 3 konceptech:

- entity popisující samostatný objekt, který může existovat sám o sobě
- atributy udávající vlastnosti entit
- vztahy sloužící k určení vztahů mezi entitami

Společně tyto položky dávají dohromady model aplikace. Ukázka jednoduchého modelu je na obrázku 1. Atributy mohou nabývat typu z předdefinovaných možností. Vztahy mezi entitami mohou být jednosměrné nebo obousměrné. Vztahy mohou být typu jeden k mnoha, mnoho k jednomu, mnoho k mnoha. Core data integruje některé funkce jako Zpět a Vpřed pro zjednodušení vývoje aplikací. Na platformě Mac OS X je přidána integrace s technologií Cocoa Bindings zajišťující mechanismus propojení rámců vzhledu s modelem, umožňující takzvané vizuální programování[2].

[2]Vizuální programování je druh programování aplikací, při kterém se eliminuje množství napsaného kódu na minimum.




Obrázek 1. Ukázka jednoduchého modelu Core Data

## 3.3. iCloud

iCloud je služba společnosti Apple Inc. poskytující úložiště dat uživatele. Služba byla spuštěna v roce 2011. iCloud je v základu bezplatná služba. Funguje v rámci AppleID. V bezplatné verzi nabízí úložiště pro 5 GB dat. Je možné si zaplatit rozšíření a službu tak lze rozšířit až o 50 GB. Je tedy možné mít k dispozici úložiště o cekové velikosti 55 GB. Tato velikost vyžaduje zaplacení nemalého ročního poplateku 80 euro (k 15. 5. 2013). Hlavní výhodou služby iCloud je jednoduchost a bezpečnost. K datům uživatele má přístup pouze jejich vlastník. iCloud je podporován jak pro platformu iOS, tak pro platformu Mac OS X. Vestavěné aplikace využívají zvláštního API [2] pro přístup k úložišti. Jedná se například o aplikace iCal, Kontakty, Mail, služba Fotostream, iBooks a další. Do kapacity úložiště se některé záležitosti nepočítají. Příkladem je služba Fotostream. Pro aplikace třetích stran distribuovaných pomocí iTunes pro iOS, Mac App Store pro Mac OS X je k dispozici speciální API pro přístup k úložišti iCloud. Existují 3 možné způsoby používání úložiště:

- Key-value úložiště pro diskrétní hodnoty jako jsou předvolby, nastavení nebo jednoduchá data,
- Document úložiště pro ukládání dokumentově orientovaných dat,
- Core Data úložiště pro synchronizaci Core Data úložiště vrámci aplikace mezi zařízeními.

Funkcionalita je velice jednoduchá. Aplikace provede změny a data uloží do iCloudu. Ostatním zařízením je oznámeno, že data byla změněna a automaticky se stáhnou, jak ilusturuje obrázek 2. Obrázek 2. je přejat z [2].

Obrázek 2. Ilustrace služby iCloud

## 3.4. AppKit a UIKit

Framework AppKit je balík objektů určených k vytváření rozmanitých uživatelských rozhraní, jako jsou například okna, panely, tlačítka, menu a mnoho dalšího. Jeho počátky sahají až do systému NeXT. Slouží pro platformu Mac OS X. Třídy z tohoto frameworku se využívají povětšinou společně s nástrojem Interface Builder, který slouží k tvorbě uživatelského rozhraní. V nástroji Interface Builder se provádí grafické změny, napojení prvků na outlety a akce zatímco programováním se komponentám rozhraní dodává potřebná funkcionalita.

Framework UIKit je upravená verze frameworku AppKit s orientací na dotyková zařízení a jejich rozhraní. Je určen pro platformu iOS. Neobsahuje všechny prvky frameworku AppKit, naopak některé nové stežejní prvky byly přidány. Velká část objektů z frameworku UIKit má svého sourozence ve frameworku AppKit. Většinou je rozdíl zejména v názvu třídy. Název třídy pro obyčejné tlačítko je UIButton ve frameworku UIKit a NSButton ve frameworku AppKit. Mají podobné atributy a vlastnosti. Stejné nejsou. Podrobný soupis tříd, které obsahují tyto frameworky a jejich atributy, je dostupný na [2].

## 3.5. Game Center

Game Center je online herní sociální síť společnosti Apple Inc. Umožňuje jednoduše spojovat hráče z celého světa. Služba spravuje herní žebříčky pro porovnávání skóré ve hrách. Služba Game Center je k dispozici pro platformu iOS od verze 4.2 a novější. Je k dispozici i pro Mac OS X verze 10.8. Největším problémem této služby je fragmentace na jednotlivé platformy. Platformy jsou prozatím tedy odděleny (k 15. 5. 2013). K přihlášení do služby Game Center je využito AppleID. Lze tak dohledat přátele z kontaktů. Pro spojení platforem je prozatím nutno využít jiných dostupných možností.

## 3.6. Důležité pojmy

### IBOutlet a IBAction

IBOutlet a IBAction jsou speciální makra jazyka Objective-C. Slouží k propojení nástroje Interface Builder s kódem aplikace. Každá třída má své sloty. Použití klíčových slov IBOutlet v hlavičce třídy může vypadat například následovně:

```
@property (weak, nonatomic) IBOutlet NSToolbar *toolbar;
@property (weak, nonatomic) IBOutlet NSMenuItem *undoMenuItem;
```

Následně dojde k tomu, že sloty pro toolbar a undoMenuItem jsou viditelné nástrojem Interface Builder a lze je přiřadit objektům v návrhu grafického rozhraní. Klíčová slova v závorkách nejsou podstatná, souvisí se správou paměti. Při použití klíčového slova IBAction u definice hlavičky metody dojde k jejímu zpřístupnění v nástroji Interface Builder. Prvku grafického rozhraní lze pak přiřadit akci, která se má provést jako reakce na určitou akci. Například stistk stlačítka. Syntax těchto metod je předem dán. Lze jej ilustrovat na tomto příkladě:

```
- (IBAction)openFile:(id)sender;
```

Klíčové slovo IBAction je návratová hodnota funkce. Ve skutečnosti je návratová hodnota typu void. Následuje název. Jediný možný argument je ukazatel na objekt, který událost vyvolal.

### Storyboard

Storyboard je označení nového souboru používaného k návrhu uživatelského rozhraní na platformě iOS. Nahrazuje původní xib soubory, které se nadále používají na platformě Mac OS X. V souborech tohoto typu se ukládá sled obrazovek. Vývoj uživatelského rozhraní v souborech tohoto typu je optimalizován na platformu iOS. Každá obrazovka vyžaduje svůj vlastní řídící objekt. Soubory lze editovat v nástroji Interface Builder. Data jsou ukládána jako XML. Návrh uživatelského rozhraní je velice jednoduchý, rychlý a nabízí prakticky neomezené možnosti s dostupnými komponentami. Společnost Apple Inc. nutí programátory dodržovat architekturu aplikací model-view-controller (MVC) [2].

# 4. Použitý postup, architektura a prostředky

Při programování aplikací bylo použito nejjednodušší možné architektury pro takto malý projekt. Základem projektu bylo napsat správně odladěnou a funkční podobu iOS aplikace s dobře promyšlenou strukturou tak, aby byla co nejméně platformově závislá. Následně byla provedena manuální úprava kódu pro druhou platformu. Bylo vhodné se první věnovat iOS aplikaci. Mac OS X platforma

nabízí víceméně všechny možné prostředky, jako nabízí iOS platforma. Vyjímkou jsou záležitosti týkající se mobilních zařízení. Mezi tuto skupinu lze považovat akcelerometr, polohové služby a další. Při dodržení použité architektury trval vývoj iOS aplikace řádově měsíce, naopak vytvoření plně funkční Mac OS X aplikace trvalo 2 týdny při stejné intenzitě práce. Postup lze shrnout do několika kroků:

- důkladně promyslet požadovanou funkci programu,
- vytvořit strukturu aplikace pro iOS zařízení,
- ujistit se, zda použité prvky mají oponenta na platformě Mac OS X,
- omezit používání platformově závislých tříd na minimum,
- zajistit přítomnost chyb v programu na minimum,
- vytvořit plně funkční iOS aplikaci,
- promyslet strukturu Mac OS X aplikace, aby se co nejvíce podobala iOS aplikaci ale splňovala standarty HIG,
- zkopírovat do projektu Mac OS X aplikace platformově nezávislé třídy,
- upravit platformově závislé třídy,
- implementovat neupravitelné třídy,
- vyhnout se následným úpravám.

Použitá architektura má spoustu nevýhod. Vyžaduje znalost obou platforem a prvotní promyšlení postupu. Největší nevýhoda spočívá v reflektování změn jedné aplikace do druhé v podobě manuálního přepisu na všech místech, kde je to zapotřebí. Pro takto malý projekt je to přijatelné. Je důležité se snažit tomuto kroku co nejvíce vyhnout. Při velkých projektech v rozsahu několik desítek tisíc řádků a stovek tříd je tento postup nepoužitelný. Je třeba se poohlédnout po alternativních možnostech. O těch budeme hovořit dále.

Architektura má i své výhody. Odpadá nutnost použití řídících tříd a protokolů. Je jednoduchá na pochopení a lze ji použít celkem snadno pro jakýkoliv projekt v rozsahu tohoto projektu. Použití platformově nezávislých tříd je možné i jako pouhé reference na fyzické soubory z druhého projektu. Jejich editace se pak značně usnadní, jelikož zde není třeba požadované soubory kopírovat.

K vývoji aplikací bylo použito vývojové prostředí Xcode a jazyk Objective-C. Aplikace pro iOS platformu využívá k vykreslování Quartz 2D knihovnu. Aplikace využívá zejména systémová tlačítka a navigační prvky z frameworku UIKit tak, aby splňovala HIG společnosti Apple Inc. K ukládání dat je použito technologie

Core Data. K přenosu dat mezi iOS zařízeními se využívá technologie iCloud. Aplikace pro Mac OS X platformu kopíruje používání prostředků z platformy iOS. Nevyužívá technologie iCloud. Rozhraní je optimalizováno. Na platformě Mac OS X je umožněno ukládání do souboru a tak přenášet data.

# 5. Uživatelská příručka

Aplikace Gotická dáma je desková hra inspirována variací velice známé deskové hry Dáma. Pravidla popisuje [1]. Samotnou hrou se již dále nebudeme zabývat.

## 5.1. iOS aplikace

Při vytváření projektu nebyla k dispozici klasická placená vývojářská licence společnosti Apple Inc. Bylo využito univerzitního programu, který neumožňuje distribuce pomocí Apple Store. Aplikace je možné spustit z vývojového prostředí Xcode 3.2 a novější pomocí iOS simulátoru jak ve verzi pro iPhone, tak ve verzi pro iPad simulující iOS 5.x a novější. Pro spuštění aplikace na reálném zařízení je nutné mít speciální povolovací profil aplikace Provisioning Profile, který obsahuje:

- UDID[3] zařízení, na kterých je možné aplikaci spustit a testovat
- jedinečný identifikátor aplikace zvaný Bundle Identifier
- informace o uživatelských certifikátech, pomocí kterých lze aplikaci signovat a následně spustit

Aplikace je kompatibilní pro zařízení iPhone verze 3GS a novější, iPod touch 3G a novější, iPad 1G a novější. Rámce vzhledu jsou optimalizované pro iPhone 5 se systémem iOS verze 5.x a vyšší. Kompatibilita s předchozími verzemi zachována. Uživatelské rozhraní pro zařízení typu iPad se maličko liší od rozhraní zařízení typu iPhone. Nejdříve bude popsáno rozhraní pro iPhone, následně rozdílné záležitosti zařízení iPad. Uživatelská rozhraní jsou optimalizováná pro každé zařízení. Rozhraní se nastaví automaticky. Jazykem aplikace je angličtina.

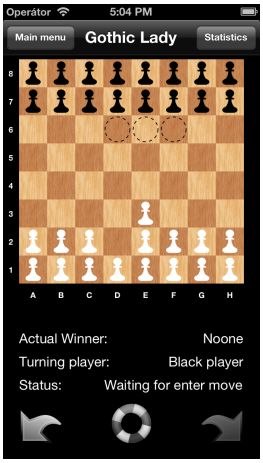
### iPhone rozhraní

Aplikace pro zařízení typu iPhone podporuje pouze režim orientace na výšku. Aplikace automaticky ukládá poslední rozehrátou hru. Pokud je aplikace spuštěna poprvé, nenačte se žádná předchozí hra. Při každém dalším spuštění aplikace se načte poslední hru. Po kliknutí na tlačítko Main Menu v Navigační liště vyjede

[3]UDID je jedinečný identifikátor zařízení Apple. Jedná se o 40-ti místný alfanumerický řetězec obsahující znaky abecedy a čísla.

zleva hlavní menu aplikace. Odtud je možné se navigovat dále. Tuto akci ilustruje obrázek 3. Po vybrání nové hry se otevře rámec vzhledu pro nastavení parametrů hry. Po stistknutí tlačítka Play navigační lišty se aplikace vrátí do hlavního rozhraní a nastaví se nová hra. Pro nastavení parametrů hry, již vytvořené, slouží tlačítko Settings v hlavním menu. Otevře se obdobný rámec jako u nové hry rozdílný tak, že místo tlačítka Play je zde tlačítko Save a změní se popisky.


Obrázek 3. Zobrazené hlavní menu hry



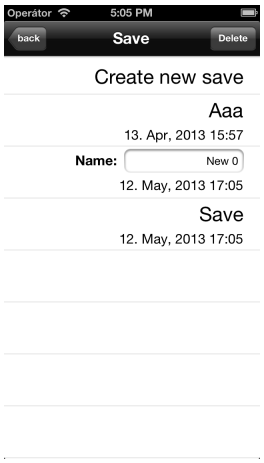

Obrázek 4. Mezifáze zadávání tahu.

Pokud je na řadě hráč, je možné zadat tah třemi způsoby. První je nápověda nejlepšího tahu. Ta se provede automaticky po kliknutí na tlačítko záchrany ve formě záchraného kruhu. Celý tah se animuje najednou. Druhou možností je zadat tah manuálné pomocí vybrání výchozího a následně cílového políčka. Aplikace po vybrání výchozího políčka zobrazí nápovědu, jak ukazuje obrázek 4. Po provedení tahu se celý tah animuje. Aplikace dovolí pouze korektní a přípustné tahy. Třetí možností je vybrání výchozího políčka a jeho tažení na cílové políčko. Aplikace opět dovolí pouze korektní tahy. Při použití této metody neproběhne animace tahu. Pokud se jedná o vícenásobný tah, je proces opakován několikrát. Tah zpět a tah vpřed je možné realizovat stiskem tlačítka zpět případně vpřed po stranách tlačítka pro výpočet nejlepšího tahu.

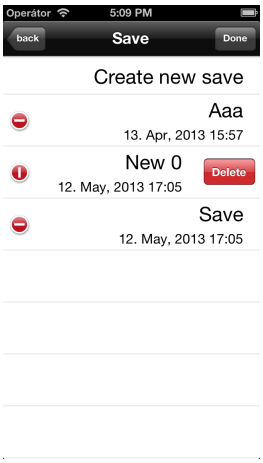
Pokud je na řadě počítačový hráč, jeho tah se provede automaticky. Výpočet tahu je signalizován indikátorem. Pokud je na řadě počítačový hráč po provedení tahů zpět či vpřed, po nahrátí uložené hry, případně po návratu do hlavního rámce z ostatních rámců, je výpočet tahu pozastaven. Pro pokračování je nutné stisknout tlačítko pro pokračování mezi tlačítky pro tah zpět a vpřed, které se zobrazí místo tlačítka pro výpočet nejlepšího tahu.

Ukládání a načítání je řešeno skrze úložiště aplikace. Úložiště je automaticky synchronizováno s ostatními zařízeními pomocí iCloudu v rámci AppleID. Při každém spuštění aplikace jsou zkontrolovány a případně staženy změny. Změny

se stahují i během fungování aplikace. Pokud iCloud není k dispozici, data se ukládají normálně bez synchronizace. Pokud aplikace zaznamená uložení hlavní hry na jiném zařízení, umožní její načtení na tomto zařízení. Pokud je již rozehrátá neukončená předešlá hra, aplikace se dotáže, zda má hru načíst. Pro uložení hry slouží rámec vzhledu, který je dostupný pomocí tlačítka Save z hlavního menu aplikace. Pro vytvoření nového záznamu slouží první řádek s názvem Create new save. Nový řádek je přidán na příslušné místo v seznamu a je rovnou přepnut do režimu editace viz obrázek 5. Přejmenovávat a mazat uložené hry je možné pomocí výběru řádku. Následně se zobrazí dialog pro výběr akcí. Jako první je možnost smazat uloženou hru, druhou možností je uložit hru (předchozí uložená hra je přepsána), třetí možností je přejmenování uložené hry. Přejmenování přepne řádek do režimu editace jako na obrázku 7. Mazat uložené hry je také možné vybráním tlačítka Edit v navigační liště, po kliknutí na levé tlačítko se zobrazí tlačítko pro smazání. Řádek je tímto vymazán. Toto ilustruje obrázek 6.




Obrázek 5. Rámec vzhledu pro ukládání v režimu editace

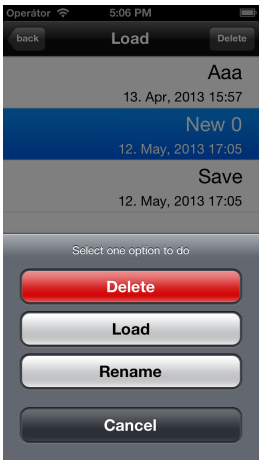



Obrázek 6. Rámec vzhledu pro ukládání v režimu mazání

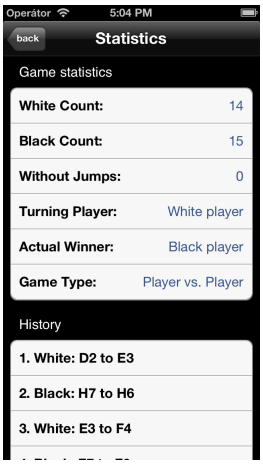
Pro načtení hry slouží rámec vzhleu dostupný pomocí tlačítka Load z hlavního menu aplikace. Po vybrání příslušného řádku je zobrazen dialog. Podoba dialogu pr rámec načítání ukazuje obrázek 7. Statistiku hry lze zobrazit pomocí tlačítka Statistics v navigační liště. Následně je zobrazen rámec vzhledu dle obrázku 8. Dole v hlavním menu aplikace se nachází tlačítko About. Toto tlačítko zobrazí rámec, ve kterém jsou vypsány pravidla hry.

### iPad rozhraní

Rozhraní aplikace pro zařízení iPad se maličko liší od rozhraní pro iPhone. Největší a nejzásadnější rozdíl je v podpoře všech orientací zařízení. Po otočení zařízení se otočí i rámec vzhledu celé aplikace a změní se poloha některých prvků. Ovládání celé aplikace je až na malé vyjímky totožné s iPhone aplikací. Obrázek




Obrázek 7. Rámec vzhledu pro načítání




Obrázek 8. Rámec vzhledu pro statistiku

9. ilustruje vzhled aplikace se zařízením v režimu landscape. Obrázek 10. ilustruje vzhled aplikace pro režim portrait se zobrazeným hlavním menu aplikace. Samozřejmostí je zobrazení menu v obou režimech. Podobnost s iPhone verzí je více než zřetelná.


Obrázek 9. Rozhraní zařízení iPad v režimu landscape

Vytvoření nové hry, upravení parametrů stávající hry, načtení a uložení rozehrátých her se zobrazují skrze speciální komponentu. Obrázek 11. ilustruje zobrazení rámce vzhledu pro načítání. Ukládání je zobrazeno obdobně. Obrázek 12. ilustruje zobrazení rámce vzhledu statistiky hry. Rámce vytváření nové hry či úprava nastavení stávající hry je obdobná. Funkcionalita je shodná s iPhone aplikací. Vše ostatní je totožné s iPhone aplikací.


Obrázek 10. Rozhraní zařízení iPad v režimu portrait


Obrázek 11. Zobrazení rámce vzhledu pro načítání zařízení iPad

## 5.2. Mac OS X aplikace

Aplikaci Gotická dáma není třeba instalovat, je spustitelná ze souboru Gothic Lady.app pod operačním systémem Mac OS X 10.6 a novější (projekt testován na verzi 10.8). Jazyk rozhraní aplikace je angličtina.

Aplikace ukládá stav poslední hry automaticky při ukončení aplikace. Při prvotním spuštění aplikace nebyla v minulosti vytvořena žádná hra. V tomto případě není k dispozici žádná předešlá hra. Po dalších spuštěních, kdy již byla hra vytvořena, se načte poslední známá pozice. Obnoví se hra i po smazání souboru aplikace a následném zkopírování se stejným názvem aplikace. Data se ukládají


Obrázek 12. Zobrazení herní statistiky na zařízení iPad se zobrazeným menu aplikace

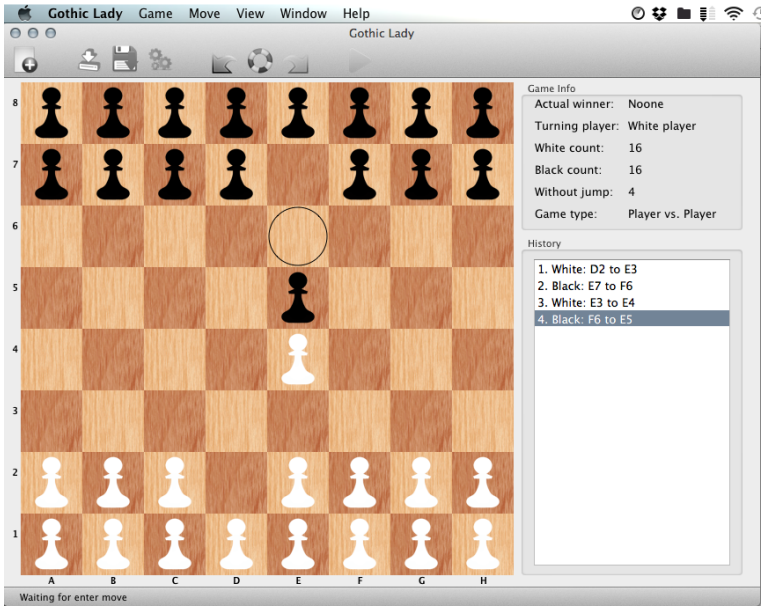
v systému do speciální složky přidělené aplikace. Vytvořit novou hru je možné z menu Game a nabídky New dle obrázku 13. nebo pomocí klávesové zkratky „Cmd+N“ , případně první nabídky panelu nástrojů zleva při výchozím nastavení (pořadí ikon je možné měnit pomocí menu View a nabídky Customize Toolbar). Jedná se o ikonku signalizující nový soubor. Následně se otevře dialog pro nastavení parametrů nové hry. Po kliknutí na tlačítko play se hra vytvoří.


Obrázek 13. Vytvoření nové hry

Hlavní okno umožňuje změnu velikosti. Lze ji změnit klasicky jako jakékoliv jiné okno v systému. Aplikace si pamatuje poslední známou pozici okna a při

následném spoštění je okno umístěno na tutéž pozici. Změnit parametry bílého a černého hráče je možné z menu Game a nabídky Game Settings nebo pomocí klávesové zkratky „Alt+Cmd+,“, případně ikony panelu nástrojů signalizující nastavení. V pravé straně jsou vyobrazeny statistiky hry a herní historie, ve které je možno se pohybovat kliknutím na příslušné místo v historii.

Pokud je na řadě hráč, může zadat tah třemi způsoby. Jako první je možnost využít nápovědy z menu Move a nabídky Best move nebo pomocí klávesové zkratky „Cmd+B“, případně ikony panelu nástrojů signalizující záchranu v podobě záchraného kruhu. Druhou možností je zadat tah manuálně pomocí kliknutí na výchozí políčko a následně cílové. Aplikace dovolí kliknout pouze na validní výchozí políčka. Po kliknutí na validní výchozí políčko jsou zobrazeny možnoti, kam s kamenem táhnout. Při vícenásobném tahu se proces opakuje. Třetí možností je tažení kamenem z výchozího políčka na cílové. Aplikace opět nedovolí nepřípustné tahy. Mezifáze zadávání tahu po zadání výchozí pozice je zobrazeno na obrázku 14. Tahy zpět a v před je možné provádět z menu Move z nabídky Undo a Redo, nebo pomocí klávesových zkratek „Cmd+Z“ pro Undo a „Cmd+Shift+Z“ pro Redo, případně příslušnými ikonkami toolbaru.

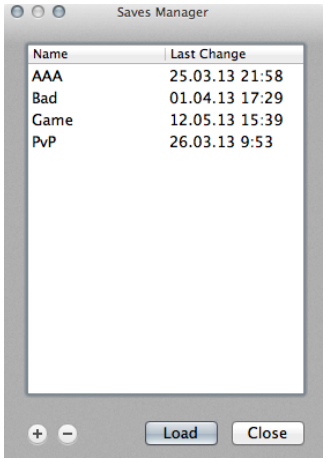



Obrázek 14. Mezifáze zadávání tahu

Pokud je na řadě počítačový hráč, jeho tah se provede automaticky. Výpočet je signalizován v levém dolním rohu. Pokud byl proveden tah zpět či vpřed, případně načtena uložená hra, aplikace se chová jako iOS aplikace.

Aplikace umožňuje uložit hru dvěma způsoby. Jako první je způsob ukládání uložených her do úložiště aplikace. Při použití iCloudu by se synchronizovaly tyto data. Pro načtení předešlé hry slouží menu Game a nabídka Load nebo klávesová zkratka „Cmd+O“, případně ikona panelu nástrojů pro načtení souboru.

Následně se zobrazí dialogové okno dle obrázku 15. V okně se nachází tabulka uložených her. Zobrazuje vždy název uložené hry a datum poslední změny souboru. Tlačítko Close slouží k zavření okna bez načtení hry. Tlačítko Load načte vybranou uloženou hru. Pokud není žádná hra vybrána, tlačítko Load neprovede nic. Tlačítko mínus slouží ke smazání vybrané uložené hry ze seznamu. Přejmenovat název uložené hry je možné poklepáním na název souboru. Tabulka se přepne do režimu editace políčka a umožní tak název změnit. Provedené změny se uloží po stisku klávesy enter, či klepnutí mimo editované políčko. Okno pro uložené hry se nijak neliší od okna načtení hry. Lze jej vyvolat z meny Game z nabídky Save nebo klávesové zkratky „Cmd+S“ případně příslušné ikony panelu nástrojů. Vzhled je totožný. Změněno bylo tlačíko Load na tlačítko Save, které umožní přepsání vybrané uložené hry ze seznamu. Tlačítko plus vytvoří nový záznam v tabulce. Název pak lze libovolně měnit zmíněným způsobem.




Obrázek 15. Okno načtení hry z úložiště aplikace

Druhým způsobem je ukládání do souboru. Aplikace používá koncovku gls. Tento typ souboru byl vytvořen pro potřeby projektu. Pro uložení hry do souboru je možno využít menu Game a nabídky Save File, nebo klávesové zkratky „Shift+Cmd+S“. Po vyvolání se objeví systémové dialogové okno pro uložení souboru. Koncovka se doplní k názvu automaticky. Načtení hry ze souboru je možné dvěma způsoby. První možností je užití menu Game a nabídky Load File. Alternativně pomocí klávesové zkratky „Shift+Cmd+O“. Otevře se systémové dialogové okno pro načtení souboru. Jsou přípustné pouze soubory typu gls. Druhou možností je poklepat na příslušný uložený soubor v systému. Pokud je se souborem asociována aplikace, otevře se a soubor se načte. Při obou způsobech se načtou pouze validně uložené hry. Aplikace případně oznámí poškození souboru a nemožnost jeho načtení. Pokud je v aplikaci uložena předchozí neukončená hra, aplikace se dotáže, zda má pokračovat načítání.

# 6. Programátorská příručka

Základní myšlenkou celého projektu bylo vymyšlení architektury aplikace, při které se za co nejmenšího úsilí podaří vytvořit aplikaci jak pro platformu Mac OS X, tak pro platformu iOS, přičemž každá aplikace bude dodržovat standarty HIG[4] společnosti Apple Inc. a bude pro kažou platformu optimalizovaná. Obě aplikace jsou napsány výhradně v jazyce Objective-C. Existují i alternativní možnosti jazyků, v nichž lze vytvářet zdrojový kód a používat jej ve zmíněných platformách. Těmi se budeme zabývat v další části práce. Aplikace využívají pouze frameworků dodávaných společností Apple Inc. Při vývoji aplikací byl striktně dodržován model aplikace MVC [2]. Jelikož jsou obě aplikace psány v jazyce Objective-C a obě dodržují model MVC, je jasné, že některé části budou společné.

## 6.1. Struktura aplikace

Celá aplikace se dá rozdělit na 4 části:

- vrstva Core,
- vrstva Board,
- vrstva Application delegate,
- vrstva Game.

Každou část detailně rozebereme pro obě platformy zvlášť. Nejdříve začneme platformou iOS. Následně budeme pokračovat platformou Mac OS X.

## 6.2. iOS aplikace

### Vrstva Core

Vrstva Core se stará o základní herní logiku, pravidla, tahy a generování nejlepšího tahu počítačového hráče. Obsahuje třídy `Change`, `Move` a třídu `Game`. Třída `Change` je elementární datový kontejner popisující změnu na desce. Třída `Move` představuje datový kontejner detailně popisující tah. Třída `Move` se skládá z elementárních součástí složených z jednotlivých instancí třídy `Change`. Třída `Game` se stará o interní reprezentaci desky, generování možných tahů (generování pole tříd `Move`), generování nejlepšího tahu, tahy zpět a vpřed a dotazy na stav hry jako například zda je konec hry, či aktuální vítěz atd. Třídy `Change`, `Move`, `Game` podléhají protokolu `NSCoding` dle [2], pomocí nějž lze implementovat převod

[4]HIG, anglicky Human interface guidelines, je soubor pravidel, které mají za cíl zlepšit komfort uživatele tím, že aplikační rozhraní je více intuitivní, učenlivé a konzistentní.

třídy na binární data a následné vytvoření třídy z binárních dat. Použití tohoto rozhraní je nezbytné pro správné fungování ukládání dat pomocí frameworku Core Data.

Třídy `StoneInfoForView` a třída `TempBoardForView` až tak dalece nepatří do vrstvy Core. Tyto třídy slouží k ukládání mezistavů herní desky během fáze zadávání tahu od uživatele. Jelikož je to ovšem pouze jakýsi datový kontejner, podle které so vykresluje herní deska, lze tyto třídy zařadit také do vrstvy Core. Všechny tyto třídy jsou absolutně platformově nezávislé. Jejich použití není nijak omezeno.

### Vrstva Board

Vrstva Board se stará o herní smyčku a zadávání vstupního tahu od uživatele. O implementaci této vrstvy se stará třída `BoardView`, jež je potomkem třídy `UIView` dle [2]. Třída `UIView` je součástí frameworku UIKit. Třída je dostupná pouze pro platformu iOS. Jedná se o základní vykreslovací prvek. Zde dochází k prvotním problémům jak pokračovat ve vývoji této třídy. K ukládání mezipozic herní desky slouží třída `StoneInfoForView`, jež reprezentuje jedno dočasné políčko a třída `TempBoarForView`, jež reprezentuje celou desku jednotlivých políček. Celá herní smyčka je implementována do této třídy. Je implementována prakticky zcela nezávisle na platformě. Vyjímkou je požadavek na překreslení desky. Jelikož se jedná o zásah pouze v cca 5ti místech v kódu, v takto malém projektu bude nejlepším řešením to ponechat platformově závislé a při portu manuálně přepsat.

Signalizace kliknutí je na první pohled platformově závislé. O to se starají metody :

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
```

Syntax těchto metod je dán dle [2]. Nemůže být změněn. Požadovaná funkčnost by se ztratila. Na první pohled se může zdát, že tyto metody jsou platformově závislé. Při bližším prozkoumání kódu však zjistíme, že metody potřebují získat souřadnici $[x, y]$ v desce a následně metody pracují nezávisle na platformě. Je podporována jak metoda drag and drop, tak i metoda kliknutí na výchozí políčko a následné kliknutí na cílové políčko.

Nejdůležitější součástí této vrstvy je ale samotné vykreslení desky. O to se stará metoda dle [2]

```
- (void)drawRect:(CGRect)rect;
```

Vždy chceme stejný výsledek. Kód bude vždy pro každou platformu jiný. Největší problém je, že platforma iOS má počáteční bod obrazovky $[0, 0]$ v levém dolním rohu display, než jak jsme zvyklí na desktopech. Je nutno využívat jiné prostředky

pro vykreslení požadovaného zobrazení. Používané metody nemají shodné názvy, ani parametry. Tato metoda je zcela platformově závislá a pro port na platformu Mac OS X ji bude nutno implementovat znovu.

### Vrstva Application delegate

Vrstva Application Delegate obsahuje třídu `AppDelegate`, což je delegát aplikace. Vedle funkcí, které třída implementovat musí dle protokolu[5] `UIApplicationDelegate` dle [2]. Třída implementuje perzistentní ukládání dat do datového úložiště aplikace pomocí technologie Core data, metody zprostředkující spojení s úložištěm a v případě iOS platformy i synchronizaci úložiště vrámci ostatních iOS zařízení pomocí technologie iCloud.

Pro aplikaci byl zvolen jednoduchý model technologie Core data s jedinou entitou Game, která má pouze 3 atributy. Atribut name, udávající název uložené hry, atribut lastChange, udávají kdy naposledy byla uložená hra pozměněna a atribut gameData, který obsahuje uložení samotné hry pomocí binárních dat. Jelikož třída game splňuje protokol `NSCoding` dle [2] a k datům aplikace nemá žádná jiná aplikace přístup bez pomoci jailbreaku[6]. To má za důsledek, že není třeba řešit bezpečnost a integritu uložených binárních dat. Uživatel k nim nemá přístup, framework Core data obstarává vše za nás.

Do iOS aplikace byla přidána podpora synchronizace mezi ostatními zařízeními vrámci AppleID pomocí technologie iCloud. Toto bylo umožněno díky možnosti využít iOS Developer University Program. Tento program umožňuje používat všechny dostupné prostředky platformy iOS jako s placenou vývojářskou licencí s jediným rozdílem, a to nemožností využít distribuce aplikace pomocí Apple Store. Distribuce aplikace není předmětem této práce. Tato licence plně postačuje potřebám projektu.

Po malé úpravě metod třídy `AppDelegate`, dodání synchronizačních metod a notifikací k upozornění ostatních objektů na změny perzistentního úložiště bylo dosaženo pozitivního výsledku. Synchronizace úložiště Core data má ale jednu nepříjemnou vlastnost. Během synchronizace, kdy fyzické úložiště aplikace neobsahuje žádná data, k němu nelze přistupovat ani z něj data číst. V extrémním případě se může jednat i o desítky sekund. Toto je známý problém a snad jej vývojáři firmy Apple Inc. vyřeší s další verzí operačního systému iOS. Proto k obnovení automaticky uložené hry při spuštění aplikace bylo zvoleno alternativní možnosti. Třída `NSUbiquitousKeyValueStore` poskytuje možnost uložení hodnot typu `<klíč, hodnota>` . Běžně se využívá k ukládání uživatelských nastavení mezi zařízeními. Má mnoho omezení. Jedním z nejzásadnějších je, že aplikace může využít pouze 1 MB sama pro sebe. Ovšem rychlost je ná-

[5]Pojem protokol může být zavádějící, je specifický pro jazyk Objective-C, v jiných programovacích jazycích se užívá pojmu rozhraní.

[6]Jailbreak je softwarová úprava iOS zařízení, které není schváleno společností Apple Inc. a umožňuje používat některé zakázané funkce. Při použití jailbreaku přichází uživatel o záruku.

sobně vyžší a pro automatické obnovení předešlé hry je plně dostačující. Dalším omezením je, že pokud uživatel nemá zapnutou podporu iCloud, nemůže aplikace toto úložiště používat a data budou ztracena. V tomto případě bylo zvoleno uložení pomocí třídy `NSUserDefaults`, pomocí které se standartně ukládají lokální uživatelská nastavení. Použití má obdobné omezené jako použití třídy `NSUbiquitousKeyValueStore`, data se však nesynchronizují přes iCloud do ostatních zařízení a jsou přístupná i bez podpory iCloud.

### Vrstva Game

Do vrstvy Game patří veškeré třídy, sloužící k řízení jednotlivých obrazovek. Samotné rámce vzhledu a jejich návaznost jsou uloženy ve souboru typu `storyboard`. Zařízení iPhone/iPod touch má svůj vlastní soubor a zařízení iPad má taktéž svůj vlastní soubor. Aplikace pak na základě toho, na jakém běží zařízení, použije příslušný soubor.

Nejdůležitější je třída `SinglePlayerScrollViewController`. Slouží k řízení rámce vzhledu, ve kterém je obsažena samotná hra. Obsahuje nejdůležitější outlet na rámec vzhledu `BoarView` a ostatní outlety, sloužící k interkci s uživatelem a zobrazováním stavu aplikace. Celá tato třída se na první pohled může zdát platformově závislou. Většina použitých metod akcí `IBAction` jsou buďto přímo zkopírovatelných na druhou platformu nebo je lze za malého úsilí převést na použití pro druhou platformu. U outletů `IBOutlet` je situace odlišná. Nedají se přímo zkopírovat, je nutno použít oponenty jednotlivých tříd. Například třída `UIButton` má svého oponenta v třídě NSButton. Má podobné atributy, podobné názvy metod (některé se dokonce shodují), ale kompatibilní nejsou. Stejně tak je to s dalšími outlety. Tato třída slouží jak pro zařízení typu iPad, tak pro zařízení typu iPhone. Vždy na základě toho, na jakém přístroji aktuálně běží, použijí příslušné metody a outlety. Většina je však pro obě zařízení stejná a jedná se pouze o jiné napojení rámců vzhledu ze souboru typu `storyboard` na příslušné outlety a akce.

Dále se zde nachází třída `SettingsViewController` sloužící k řízení rámce vzhledu pro nastavení parametrů hry a vytvoření nové hry. Následně třída `StatsViewController`, což je řídící třída rámce vzhledu, zobrazující aktuální herní statistiky a celou historii hry. Pro dosažení potřebné funkcionality i na zařízení iPad nebylo potřeba přidávat další rámce do storyboard souboru zařízení iPad ale stačí použít již vytvořené rámce ze storyboard souboru zařízení iPhone a následně je zobrazit pomocí speciální třídy `UIPopoverController` dle [2], která je součástí frameworku UIKit. `LoadTableViewController` a `SaveTableViewController` jsou řídící třídy rámců vzhledu pro načítání a ukládání uložených her. `StatsViewController`, `LoadTableViewController`, `SaveTableViewController` jsou potomci třídy `UITableViewController` dle [2], jež představuje řídící objekt pro rámce vzhledu obsahující tabulku. Všechny tyto třídy jsou platformově závislé. Pro jejich port bude nutná reimplementace. Při

bližším zkoumání zjistíme, že část kódu je znovupoužitelná a přenositelná.

V projektu se nachází několik dalších malých pomocných tříd, jež nemají valný význam pro projekt. Jsou však nezbytné pro fungování celého projektu. Příkladem může být třída `CellWithEditBox`, jež slouží jako buňka tabulky pro načítání a ukládání ve třídách `LoadTableViewController` a `LoadTableViewController`. Těmito třídami se není nutno příliš zabývat.

## 6.3. Mac OS X aplikace

### Vrstva Core

Vrstva Core obsahuje tytéž Třídy jako vrstva Core iOS aplikace. Jejich port na platformu Mac OS X znamenal pouhopouhé zkopírování zdrojových kódů do příslušného projektu. Kódy nebyly nijak upravovány a byly použitelné okamžitě.

### Vrstva Board

Vrstva Board obsahuje opět třídu `BoardView`. Narozdíl od svého protějšku je potomkem třídy `NSView`, dle [2], což je oponent třídy `UIView` ,dle [2], ve frameworku AppKit. Většina metod byla zkopírována ze svého protějšku z platformy iOS. Vyjímkou jsou místa, kde aplikace vyžaduje překreslení okna. Pro tento převod stačilo přepsat několik míst v kódu. Metody pro rozpoznání kliknutí a metodu pro překreslení okna bylo nutné uzpůsobit tvaru, požadovaným třídou `NSView`.

```
- (void)mouseDown:(NSEvent *)theEvent;
- (void)mouseDragged:(NSEvent *)theEvent;
- (void)mouseUp:(NSEvent *)theEvent;
- (void)drawRect:(NSRect)dirtyRect;
```

Jak je vidět z názvů metod, metody jsou velice podobné metodám protějškům třídy z platformy iOS. První tři metody se liší pouze v převodu souřadnice bodu kliknutí v rámci vzhledu do souřadnic $[x, y]$ v desce. Po té jsou metody identické. Poslední metoda vyžadovala úplnou reimplementaci. Zejména bylo nutné věnovat pozornost na změnu rozlišení rámce vzhledu. Na platformě iOS je běžné konstatního rozlišení. Na platformě Mac OS X se však rozlišení často mění a musí být přizpůsobitelné. Vzhledem k délce tvorby původní třídy pro platformu iOS to byl velice malý zlomek času na úpravu a třída byla připravena k použití.

### Vrstva Application delegate

Platforma Mac OS X nezanechává strukturu řízení aplikace jako platforma iOS. Tedy: jedno okno = jeden řídící objekt. Většinu práce zde řeší delegát apli-

kace. Část práce a odpovědnosti z vrstvy Game byla přenesena do vrstvy Application delegate. Jedná se zejména o interakci s uživatelem, outlety a akce z a do grafického rozhraní.

Model z iOS aplikace by pouhopouhým zkopírováním přenesen na platformu Mac OS X. Zůstává nepozměněn. Základní rozhraní k modelu nám vytvořilo prostředí Xcode stejně jako na platformě iOS. Metody pro práci s úložištěm core data:

```
- (void)saveGame:(NSString*)newName;
- (BOOL)loadGame:(NSString*)nameToLoad;
- (NSArray*)allSavesNames;
- (BOOL)removeSave:(NSString*)nameToDelete;
- (BOOL)renameSave:(NSString*)oldName newName:(NSString*)newName;
```

byly vytvořeny pouhým zkopírováním z platformy iOS. Nebyly nijak upravovány a byly připravené k použití ihned. Do aplikace pro platformu Mac OS X byla přidáno jedno malé vylepšení. Aplikace umí ukládat uložené hry nejen do svého úložiště, umí taktéž ukládat do souboru samotná binární data uložené hry. Za tímto účelem byl vytvořen speciální typ souboru `.gls`, aplikace se tak stala zároveň dokumentově orientovanou. Nastal problém s bezpečností a integritou uložených dat. Naproti platformě iOS, kde se data ukládaly pouze do úložiště aplikace a žádná jiná aplikace k nim neměla přístup, zde lze data otevřít a editovat v jakémkoliv textovém editoru. Problém byl vyřešen velice jednoduše. Výsledná data byla zakódovana pomocí symetrické blokové šifry AES. Využívá se stále jednoho klíče, který je použit jako konstanta v programu. Bylo by možné dodat podporu zakodování každého souboru jiným klíčem, který by si uživatel zadal. Data zde jsou ochráněna proti neumyslnému změnění. Není nutné zadávat kód k uložení souboru vždy manuálně. Při pokusu o změnu data nebudou integritní a nebude možné je znovu sestavit v původní obraz.

Co se týče outletů a akcí z třídy `SinglePlayerScrollViewController`, ty bylo možné buďto zkopírovat přímo nebo je nahradit jejich oponenty. Zde nenastal téměř žádný problém. Outlety byly napojeny na objekty v hlavním okně aplikace.

Pro používání technologie iCloud na platformě Mac OS X je nutno mít placený vývojářský účet zaregistrovaný u společnosti Apple Inc. Tento účet k dispozici nebyl. Do Mac OS X aplikace tedy nebyla přidána podpora iCloud synchronizace. Úprava aplikace, aby iCloud podporovala by byla velice jednoduchá. Stačilo by upravit několik metod, kopírovat synchronizační a notifikační metody a aplikace by pak iCloud podporovala. Úložiště Core data na aplikace pro Mac OS X je kompatibilní s platformou iOS. Toto není předmětem práce. Podpoře iCloud úložiště nebyla věnována velká pozornost na platformě Mac OS X.

### Vrstva Game

Jak již bylo řečeno, většinu odpovědnosti vrstvy Game z platformy iOS pře-

jímá vrstva Application Delegate. Zůstalo zde spousta nevyřešených věcí. Bylo nutné implementovat podporu vytvoření nové hry, úprava parametrů stávající hry a práce s uloženými hrami. Ke všemu byla použita podtřída třídy `NSPanel`, což je jednoduché dialogové okno. Zde není dodrženo postupu, kdy okno má svého vlastního delegáta. Vše si obstarává samotná třída. Bylo nutno najít objekty, které plní stejnou funkci, jako plní objekty v požadovaných rámcích na platformě iOS. Po krátkém hledání byly objeveny podobné objekty. Pro příklad pro nahrazení použité třídy `UITableView` bylo použito třídy `NSTableView`. Na první pohled se může zdát, že třídy budou stejné. Zde je bohužél opak pravdou. Práce s nimi je naprosto odlišná. Navzdory předchozím pochybnostem až na malé vyjímky měly podobné metody a atributy. Bylo tak možno využít část původního kódu z platformy iOS aplikace a upravit jej pro potřeby platformy Mac OS X. Po vyřešení všech problémů bylo dosaženo požadovaných výsledků opět v relativně malém zlomku délky tvorby původních rámců na platformě iOS.

# 7. Možná rozšíření projektu

Ihned po získání vývojářské licence pro platformu Mac OS X bude do aplikace dodána podpora iCloud. Úprava vyžaduje minimální zásah do zdrojových kódů aplikace a čas strávený nad tímto problém by se dal shrnout do několika desítek minut.

Nejvhodnější vylepšení spočívá v přidání podpory hry více hráčů v rámci více zařízení. Nejjednodušší bude napojit aplikace na službu Game Center společnosti Apple Inc. Tento zásah by vyžadoval velké úsilí. Největším limitem použití služby Game Center je oddělenost platforem. Důsledek tohoto omezení je takový, že uživatel s iOS zařízením nemůže hrát proti uživateli, jež má Mac OS X aplikaci. Aplikace mají společné jádro, proč by tedy nebyl multiplayer umožněn? Toto otázku nebylo možné vyřešit. Ve skutečnosti se tato meziplatformní komunikace ve hrách nachází velice zřídkakdy. Nejjednodušší metodou, jak dodat podporu hry více hráču bez použití nízkoúrovňové TCP komunikace, je využít protokolu Bonjour dle [2]. Komunikace by probíhala na základě jednoduchých zpráv. K zakódování zprávy je možno použít například značkovacího jazyka XML. Výsledná zpráva pro odeslání výběru tahu hráče by mohla vypadat například takto:

```
<move>
<from x = "1" y = "1">
<jumps></jumps>
<to x = "2" y = "2">
</move>
```

Pro ošetření chybových stavů by bylo třeba přidat potvrzovací zprávy podobně jako je tomu u protokolu TCP. Toto řešení by umožnilo hru více hráču vrámci sítě. Toto řešení z časových důvodů nebylo možné provést.

Aplikace je odzačátku psána pro podporu více jazyků. Nejvhodnější je aplikaci vyvinout v angličtině. Je to nejuniverzálnější jazyk. Při dodání podpory více jazyků do aplikace si pak aplikace sama rozhodne na základě systémového nastavení, jaký jazyk použije. Primárně se snaží o systémový jazyk. Pokud není jazyk k dispozici, nastaví aplikace jazyk, v jakém byla aplikace vytvářena. Postup je shodný na obou platformách. Kdyby aplikace byly od začátku psány pouze pro český jazyk, aplikace by pak defaultně nastavovala jazyk český.

Při použití maker `NSLocalizedString(@"string", @"comment")` lze aplikaci jednoduše lokalizovat. Pomocí speciální utility bude ze všech zdrojových kódů vytvořen speciální soubor v následující podobě:

```
/* The number 5 */
"Five" = "Pět";

/* The number 4 */
"Four" = "Čtyři";

/* The number 1 */
"One" = "Jedna";

/* The number 3 */
"Tree" = "Tři";

/* The number 2 */
"Two" = "Dvě";
```

Jednotlivé použité textové řetězce jsou seřazeny abecedně, nalevo je originální název ve vývojovém jazyce. Napravo jeho lokalizovaná verze. Mezi `/* */` se pak nachází komentář textového řetězce. Pro každý jazyk je nutno vytvořit speciální soubor. Nevýhodou je, že je nutné opakovat celý postup pro každou platformu zvlášť. Překládají se i soubory typu storyboard. U nich se ale žádný soubor negeneruje, vytvoří se dva podsoubory. Textové řetězce se pak manuálně upraví přímo v návrhu uživatelského rozhraní. Lokalizace aplikací nebyla předmětem práce. Lokalizaci nebyla věnována přílišná pozornost.

# 8. Alternativní postupy, architektury a prostředky

Použitý postup má své výhody i nevýhody. Jak již bylo zmíněno, při použití ve větším projektu je nepoužitelný. Nyní se zaměříme na alternativní postupy a prostředky, které umožňují vyvíjet iOS a Mac OS X aplikace.

## 8.1. Využití řídících tříd a protokolů

Tato metoda by v tomto projektu znamenala přesunutí odpovědnosti a řízení aplikace do speciálních tříd, které by byly platformově nezávislé. Nastává zde problém; UI obou platforem není kompatibilní. V těchto třídách není možné se přímo odkazovat na outlety a akce v rozhraní. Třída by se tím stala platformově závislou. Elementární problém, jako nastavení popisku textu, zde může být zásadní. Problém lze vyřešit vytvořením protokolu, který musí cílová třída splňovat.. Implementace jednoduchého protokolu pro nastavení dvou popisků může vypadat následovně:

```
@protocol MyProtocol <NSObject>
- (void)setLabel1Caption:(NSString*)s label2:(NSString*)s2;
@end
```

u požadovaného objektu pak použijeme známe syntaxe:

```
@interface ObjectWithProtocol : NSObject <MyProtocol>
...
```

V řídící třídě přidáme odkaz na objekt typu rozhraní kterým můžeme volat metody požadovaného protokolu, ale o konkrétní implementaci se již starat nemusíme a vyřešíme ji na každé platformě zvlášť.

Toto řešení je výhodnější u velký projektů a zejména u projektů, na kterých spolupracuje více lidí najednou. Při tomto řešení odpadá problém s manuální změnou kódu. Změny v řídících třídách lze snadno přenášet. Není potřeba kopírovat změny do druhého projektu.

Největší nevýhodou tohoto řešení, je nutnost mít pro nastavení každé proměné v naší třídě, jenž má splňovat protokol, zvláštní metodu pro nastavení každého požadovaného atributu. Pokud bude dopředu známo, zda se bude vždy nastavovat několik parametrů najednou, lze je sloučit do jedné metody. Třídy se stanou univerzálnějšími na úkor délky prvotní implementace. Místo původního přímého nastavení popisku objektu, je zapotřebí metoda v rozhraní, která zavolá metodu rozhraní na objekt typu rozhraní a tento objekt pak musí požadavek zpracovat v další metodě. Vzniká tak místy až zbytečná režie aplikace. Problém se ještě více komplikuje při nastavování několika různých atributů jednoho objektu v různých místech kódu.

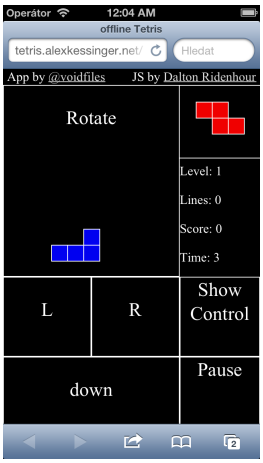
## 8.2. Použití OpenGL

Obě platformy nabízejí možnost použití grafické knihovny OpenGL. Jak již bylo řečeno v předchozí kapitole, knihovny nabízejí podobné prostředky. Stejné nejsou. Podrobnosti a detaily se lze dočíst na [2]. Na [2] je k dispozici ukázkový kód pro použití OpenGL na platformu Mac OS X a OpenGL ES na platformu iOS. Použití této grafické knihovny má řadu výhod. Největší výhodou je přenositelnost

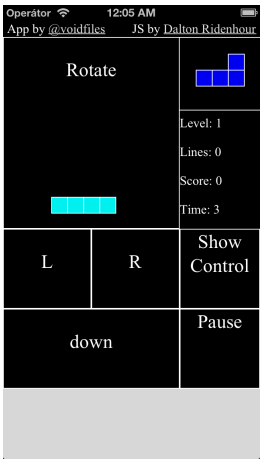
kódu i na jiné platformy jako například Windows či Linux při použití knihovny OpenGL a mobilní platformy, jako Android, Symbian, Windows Mobile, při použití knihovny OpenGL ES. Použití knihoven je specifické a nelze jej uplatnit na všechny typy aplikací. Je tedy pouze na programátorovi, aby se rozhodnul, zda je použití knihovny OpenGL namístě. Převést kód z knihovny OpenGL na kód knihovny OpenGL ES je nelehký úkol. Toto řešení není doporučeno.

## 8.3. Použití HTML5 a JavaScriptu

Nejuniverzálnější metodou pro vytvoření aplikace pro platformu iOS a Mac OS X je využití frameworku WebKit a postavit celou aplikaci na standartu HTML5 a JavaScriptu [3]. Je možné vytvořit aplikaci formou webové aplikace a zobrazovat její obsah skrze internetový prohlížeč nebo vytvořit nativní iOS aplikaci a zobrazit aplikaci pomocí speciální třídy `UIWebView`. V druhém případě lze využít standartu HTML5 pouze k některým klíčovým funkcím. Ostatní je možné vytvořit nativními prostředky. První řešení nabízí mnohem více. Aplikace se stane zcela platformově nezávislou. Není potřeba ji nijak upravovat. Lze ji následně zobrazit a spustit na jakémkoliv internetovém prohlížeči na jakémkoliv zařízení podporující HTML5 a JavaScript. Ukázka jednoduché HTML5 aplikace na iOS zařízení nám ukazuje obrázek 16. pro aplikaci zobrazenou v prohlížeči Safari a zobrazní pomocí nativní aplikace ukazuje obrázek 17.




Obrázek 16. Ukázka HTML5 aplikace v prohlížeči Safari




Obrázek 17. Ukázka HTML5 aplikace pomocí nativní aplikace

## 8.4. Použití jiných programovacích jazyků

Prozatimní metody se zabývaly zejména převodem UI z platformy iOS na platformu Mac OS X. Objective-C se dá považovat za objektovou nadstavbu

jazyka C. Není překvapením, že platforma iOS podporuje i vývojový jazyk C. Podporuje i jazyk C++. Požadovaný kód je převáděn přímo do strojového kódu. Lze tedy mísit kód C/C++ s kódem jazyka Objective-C. Je známo, že v jazyce C/C++ je napsána řada knihoven pro nejrůznější účely. Mohou tak značně usnadnit samotné programování. Ukázka kódu zde:

```
static float obvod(float a, float b, float c)
{
    return a+b+c;
}

- (float)heron:(float)a b:(float)b c:(float)c
{
    float s = obvod(a, b, c)/2.0;
    return sqrtf(s*(s - a)*(s - b)*(s - c));
}
```

Z ukázky je patrné, že v metodě heron, jenž je syntakticky zapsána v jazyce Objective-C, užíváme statické funkce jazyka C obvod. Použití jazyka C/C++ je limitováno dostupnými systémovými knihovnami. Jejich detailní rozpis lze nalézt na [2].

## 8.5. Xamarin MonoTouch Project

Společnost Xamarin[6] se zabývá přenosem platformy .NET a jazyka C# na platformu iOS a ostatní mobilní platformy, jako je Android či Windows Phone. Největší přínos tohoto řešení je pro programátory zběhlé v jazyce C# , čili pokud je pro ně jazyk C# přívětivější a nechtějí se učit novou syntaxi. Celé řešení je založené na vytvořením kodu v jazyce C# a jeho následném převedení do nativního kodu. Programátor se tak nemůže vyhnout jistému druhu poznání jazyka platformy iOS a frameworků jí nabízejích. Ke správnému fungování tohoto řešení je nutné mít nainstalované prostředí Xcode. Samotný vývoj aplikace pak probíhá tak, že zdrojové kódy aplikací jsou editovány pomocí programu Xamarin Studio a návrh uživatelského rozhraní je vytvářen v prostředí Xcode. Malou ukázkou je metoda v jazyce C# třídy `UIViewController` dle [2], jež se zavolá po nahrátí rámce vzhledu:

```
public override void ViewDidLoad ()
{
base.ViewDidLoad ();
base.View.BackgroundColor = UIColor.Black;
}
```

a nyní její ekvivalent v jazyce Objective-C:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor blackColor]];
}
```

Ekvivalence kódu lze jistě vidět naprvní pohled. Projekt MonoTouch umožňuje vytvářet nativní aplikace i pro platformu Mac OS X. K dispozici jsou dva způsoby:

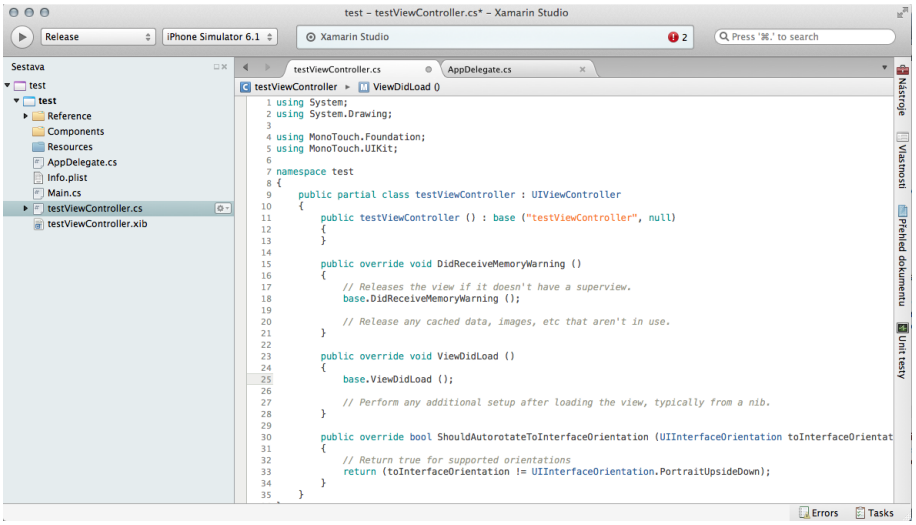
- využití klasického open source projektu Mono,
- využití licencované verze projektu Mono od společnosti Xamarin.

Obě možnosti nabízejí vývoj aplikace v nativní formě. Rozdíl mezi možnostmi je zásadní. První možnost se zakládá na využití virtuálního stroje a následném spouštění aplikace pomocí tohoto virtuálního stroje. Je narozdíl od druhé možnosti zcela zdarma. Druhá možnost se zaměřuje na převod kódu do nativního kóu jazyka Objective-C a umožňuje používat iCloud, pravděpodobně i distribuci pomocí Apple Store a ostatní dostupné služby společnosti Apple. Programování zdrojového kódu se provádí v prostředí Xamarin Studio, jak ukazuje obrázek 18. Návrh uživatelského rozhraní se vytváří v prostředí Xcode, viz obrázek 19. Z obrázků 18. a 19. je patrné, že prostředí Xamarin Studio není zaměření na vývoj UI. Jedná se pouze o editor zdrojového kodu, který má za snahu se napodobit prostředí Xcode a zároveň částečně i prostředí Microsoft Visual Studio. Aplikace se dají tím pádem nazvat nativní. K využívání této služby je zapotřebí mít zakoupenou licenci od společnosti Xamarin. Základní a nejlevnější licence stojí 299 dolarů na rok. Podrobný soupis zde [6].

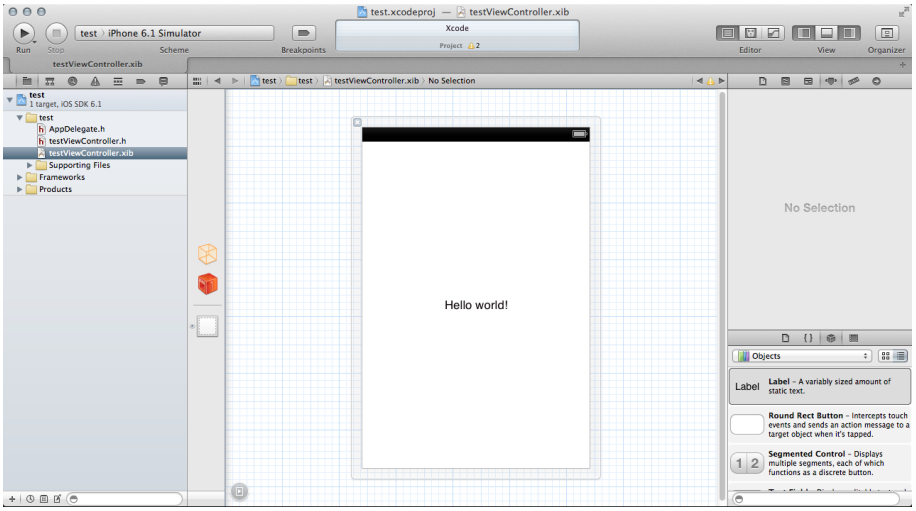
Projekt MonoTouch nabízí možnost programovat nejen skrze platformu iOS a Mac OS X, ale i pro platformu Android a Windows Phone. Pomocí klasického projektu Mono lze používat vytvořený kód i na platformách Windows, Linux. Jedná se však pouze o programování částí netýkajících se grafického rozhraní. Platí tak předešlé problémy a jejich řešení, jelikož pro platformu iOS a pro platformu Mac OS X je nutno navrhnout uživatelské rozhraní samostatně.

## 8.6. Embarcadero RAD Studio

Zajímavou alternativou, jenž nabízí společnost Embarcadero [7], je nejnovější verze vývojového prostředí RAD Studio XE2 a novější s balíčkem FireMonkey iOS Tools. Základním vývojovým jazykem je programovací jazyk Delphi. Umožňuje vytvářet nativní aplikace vedle Mac OS X platformy i pro platformu iOS. Prostředí nabízí i jazyk C++ ale ten nelze využít pro potřebný efekt. Všechny předešlé možnosti a alternativy vyžadovaly ke spuštění a následném programování operační systém Mac OS X. Tento problém neodpadá. K vývoji aplikací pro



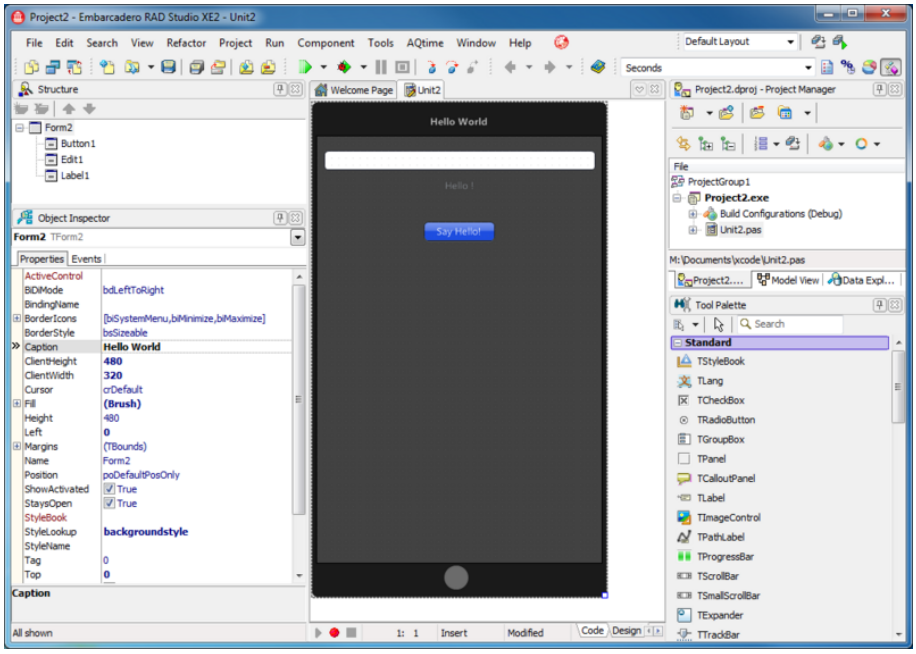

Obrázek 18. Editace zdrojového kodu potomka třídy `UIViewController` v prostředí Xamarin Studio




Obrázek 19. Návrh UI třídy v prostředí Xcode z obrázku 18.

platformu iOS je zapotřebí mít jednak nainstalované prostředí RAD Studio, ale i samotnou vývojářskou licenci společnosti Apple Inc. Předchozí možnosti využívaly buď přímo jazyka Objective-C nebo se do něj převáděly. Společnost Embarcadero si vytvořila svůj vlastní a optimalizovaný ARM kompilátor pro tuto platformu, který z části využívá standartního kompilátoru jazyka Pascal. K možnosti vytvářet uživatelské rozhraní pomocí nástroje RAD Studio si společnost Embarcadero vytvořila svoje vlastní grafické knihovny, kopírující vzhled a možnosti standartních knihoven, dostupných ve frameworku UIKit. Ve výsledku vše funguje tak, že se v prostředí RAD Studio vytvoří kompletní projekt na platformě

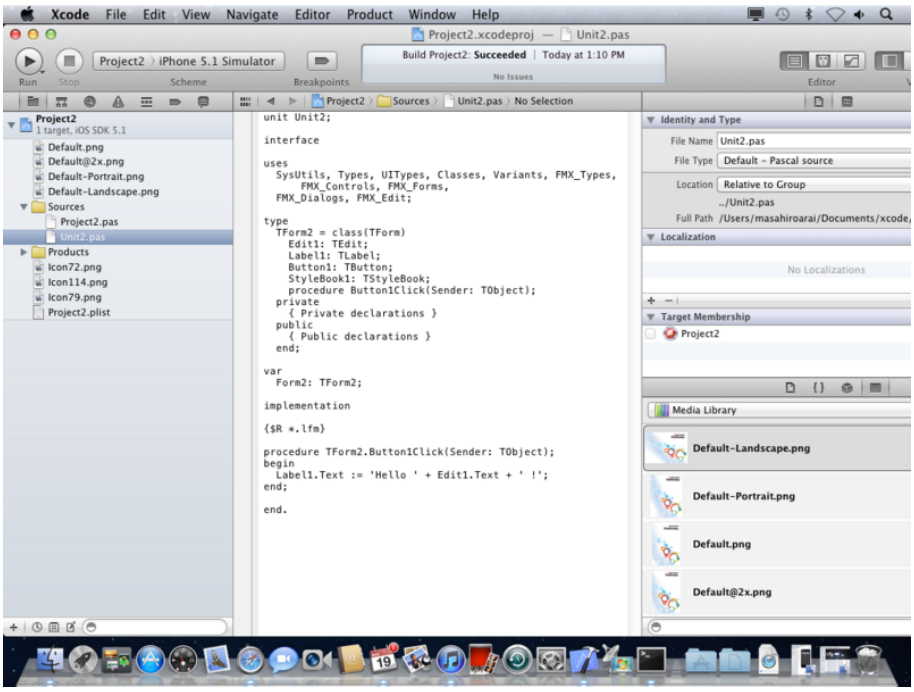
Windows. Toto popisuje obrázek 20.. Následně je nutné provést export na Xcode projekt, který obsahuje kód v jazyce Pascal, jak ukazuje obrázek 21.. V projektových nastaveních je předem nastaveno využití pascal kompilátoru. Projekt pak lze zkompilovat a dostat do zařízení standartní cestou jako je tomu u klasického použití jazyka Objective-C. Obrázek 22. ukazuje spuštěnou aplikaci z obrázku 20. pomocí iPhone simulátoru. Na internetu[8] lze nalézt způsob automatického převodu a následného spuštění projektu pomocí Xcode. Obrázky 20., 21., 22. byly přejaty z [8].




Obrázek 20. Návrh UI v prostředí RAD Studio

Řešení založené na využití nástroje RAD Studio od společnosti Embarcadero najde uplatnění u vývojářu, kteří jsou zvyklí na programovací jazyk Pascal, knihovnu FireMonkey a mají již hotové kódy a aplikace, které by rádi uplatnili i na platformě iOS. Za pomoci tohoto řešení je možno vyvíjet aplikace i na platformě Windows. Předchozí možnosti nutily programátory využívat operační systém Mac OS X. Zde je nutná přítomnost systému Mac OS X pouze na samotné testování programu na reálném zařízení. Návrh aplikace a základní testování, pro nějž programátoři Objective-C využívají služeb iOS simulátoru, je možné provádět přímo v systému Windows ve speciálním typu okna. Simulátor platformy iOS nenabízí používání prostedků, které nabízí pouze reálná zařízení. Jedná se o polohové služby, kameru, akcelerometr a další. Výsledný efekt je stejný.

Největší nevýhodou tohoto řešení je cena. Vedle klasické vývojářské licence pro platformu iOS je nutné si zakoupit samotné vývojové prostředí s cenou převyšující desítky tisíc korun. Nejlevnější verze RAD Studia stojí v přepočtu přes 30 tisíc




Obrázek 21. Exportovaný projekt z obrázku 20. do prostředí Xcode


Obrázek 22. Aplikace z obrázku 21. spuštěná na iPhone simulátoru

korun. Nejdražší pak přes 100 tisíc a jsou to programy určené výhradně pro profesionály. Na [7] lze stáhnout zkušební verze RAD Studia na 30 dní. Aplikace je možno distribuovat pomocí Apple Store.

# Závěr

Práce seznamuje čtenáře s platformami iOS a Mac OS X. Ukázala jaké prostředky jsou jimi dostupné, které prostředky jsou sdílené a které naopak sdílené nejsou. Práce také vysvětlila důležité pojmy z obou platforem. Postup použitý při vývoji softwaru je znovupoužitelný pro mnoho tohoto projektů v rozsahu projektu. Je snadný, rychlý, přímočarý a naopak často neefektivní a při určitých situacích problémový. Při dodržení postupu se doba vývoje aplikace pro druhou platformu značně zkrátila. Znovupoužitelnost kódu by se dala vyčíslit v procentech. Je to cca 75%. Práce ukazuje alternativní možnost postupu při použití stejného jazyka. Ukazuje také možnosti, jež nejsou založeny na jazyce Objective-C a v některých případech nebyly založeny ani na použití vývojového prostředí Xcode. Všechny možnosti do jisté míry vyžadovaly přítomnost vývojového prostředí Xcode. Je to tedy neodělitelná součást vývoje pro obě platformy.

# Conclusions

This thesis introduces reader the iOS and Mac OS X platform. It shows what resources are available to them, which means which are shared, and which are not shared. Work also explains important concepts from both platforms. The code used in the development of software is reusable for many projects in the scope of the project. It's easy, quick, and straightforward on the other hand often inefficient and problematic in certain situations. Time needed for develop the second application is reduced with used process. Achieved through the development time for the second application platform greatly reduced. Reusability of code could be quantified as a percentage. It is about 75 %. The work shows the possibility of alternative procedure using the same language. It shows options that are not based on the language, Objective-C, and in some cases were not based on the use or development environment Xcode. All options to some extent require the presence of Xcode development environment. It is therefore a part of development for both platforms.

# Reference


[1] Pravidla hry *Gotická dáma.* Elektronická publikace, 2009.

[2] Apple Inc. *Developer Library* Elektronická publikace, 2013.

[3] Scott Preston, *Learn HTML5 and JavaScript for iOS* Distributed to the book trade worldwide by Springer Science Business Media, c2012.

[4] Jonathan Levin, *Mac OS X and iOS Internals* Wiley, 2013

[5] Stephen G Kochan *Objective-C 2.0: výukový kurz programování pro Mac OS X a iPhone* Brno: Computer Press, 2010

[6] Xamarin *MonoTouch Project* Elektronická publikace, 2013.

[7] Embarcadero *RAD Studio* Elektronická publikace, 2013.

[8] Embarcadero *Wiki* Elektronická publikace, 2013.

# A. Obsah přiloženého CD

V samotném závěru práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. závazné adresářové struktury, důležitých souborů apod.

`bin/`
: program Gothic Lady.app spustitelný přímo z CD/DVD pro platformu Mac OS X.

`doc/`
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu), tj. zdrojový text dokumentace, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty programu Gothic Lady pro Mac OS X a programu Checkers pro iOS.

`readme.txt`
: Instrukce pro spuštění programů Gothic Lady pro Mac OS X a Checkers pro iOS, včetně požadavků pro jeho provoz.

U veškerých odjinud převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru `readme.txt`.