Mobilní aplikace pro děti s poruchou učení

Bc. Michal Žampach

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal Žampach**
Osobní číslo:	**A15206**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**prezenční**
Téma práce:	**Mobilní aplikace pro děti s poruchou učení**
Téma anglicky:	**A Mobile Application for Children with Learning Disabilities**

Zásady pro vypracování:

1. **Prostudujte architekturu platformy iOS a její možnosti vývoje mobilních aplikací a stručně popište v teoretické části.**
2. **Shromážděte funkční a nefunkční požadavky na mobilní aplikaci pro děti s poruchou učení a prostudujte její grafické podklady.**
3. **Na základě požadavků vyberte knihovny vhodné pro praktickou implementaci, navrhněte základní obrazovky aplikace a způsob interakce uživatele s aplikací.**
4. **Naprogramujte aplikaci pro zařízení iPad.**
5. **Aplikaci otestujte na reálném zařízení a diskutujte také možnosti jejího případného dalšího rozšíření.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **NEUBURG, Matt. IOS 10 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781491970027.**
2. **MANNING, Jonathon a Paris BUTTFIELD-ADDISON. IOS Swift Game Development Cookbook, 2nd Edition: Simple Solutions for Game Development Problems. 2nd edition. Newton, Massachusetts, USA: O'Reilly Media, 2015. ISBN 9781491920800.**
3. **NEUBURG, Matt. Programming iOS 10: Dive Deep into Views, View Controllers, and Frameworks. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781491970119.**
4. **KNOTT, Matthew. Beginning Xcode: Swift 3 Edition. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781430250050.**
5. **The Swift Programming Language: Swift Programming Series [online]. 2014. Apple [cit. 2017-01-23]. Dostupné z: https://itunes.apple.com/cz/book/swift-programming-language/id881256329?mt=11**
6. **GOODWILL, James a Wesley MATLOCK. Beginning Swift Games Development for iOS: Updated for Swift 3. 2nd edition. New York City, USA: Apress, 2017. ISBN 9781484223093.**

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **17. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

L.S.

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Cílem této diplomové práce je navrhnout a vytvořit aplikaci pro děti s poruchou učení, která je určena pro zařízení iPad. Náplní je návrh a samotná realizace aplikace včetně vypracování vlastní grafiky a odladění. V rámci práce byl vytvořen wireframe aplikace, sepsány funkční a nefunkční požadavky, navržena databáze uživatelů a následně vytvořena i samotná aplikace na platformě iOS využívající herní framework SpriteKit.

Klíčová slova: aplikace, iOS, Swift, XCode, iPad, porucha učení

ABSTRACT

The goal of this diploma is to desing and develop mobile application for children with learning disability. Application is intended for iPad devices. Diploma contents design and realization of the application also with creating own graphic content and debugging. Wireframes, function and non-function requirements and user database were created as a part of this diploma. In the end whole application was created for iOS with SpriteKit framework.

Keywords: application, iOS, Swift, Xcode, iPad, learning disability

Rád bych poděkoval svému vedoucímu diplomové práce, panu Ing. Radku Valovi Ph.D., za skvělý námět a rady, které mi během vypracovávání poskytl. Mému kamarádovi Ing. Jakubu Hubáčkovi za jeho tipy spojené s vývojem her. Také své sestře za zapůjčení jejího zařízení iPad po celou dobu programování a testování aplikace. A také všem ostatním, kteří aplikaci pomáhali testovat.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

„Hra je jeden z nejefektivnějších způsobů, jak zjednodušit život.

Přesně to jsme dělali jako děti, ale v dospělosti jsme si hrát zapomněli.“

Albert Einstein

OBSAH

ÚVOD

Dnešní doba rychlého technologického pokroku je nezastavitelná. Rok za rokem se setkáváme s novými technologiemi, novými zařízeními a způsoby, jak ona zařízení a technologie skloubit dohromady k vytvoření něčeho užitečného, co bude člověku pomáhat.

Klasickým příkladem jsou chytrá zařízení, jako jsou chytré telefony a tablety. Například telefon nahradil uživateli pevnou linku, poskytl mu fotoaparát do kapsy, přístup k internetu téměř odkudkoliv a další nemalé změny. Dnes už si lidé nezapínají televizi brzy ráno, aby stihli ranní zprávy a předpověď počasí. Lidé si už ani nechodí tolik kupovat noviny do trafiky. Pohodlně si tyto informace vyčtou ze svého chytrého zařízení pomocí pár gest a doteků. Chytrá zařízení jsou dnes tak komplexní systémy, že technologie, jako čtečky otisků prstů nebo rozeznání řeči, jsou běžným standardem. Výkonný ředitel společnosti Apple, Tim Cook, nedávno během jednoho rozhovoru řekl o chytrém telefonu iPhone následující: „IPhone je natolik integrovanou a nedílnou součástí našeho života, že bychom z domu neměli odcházet bez něj.“

Cílem této diplomové práce je navrhnout a vytvořit aplikaci pro chytré zařízení iPad. Aplikace je určena pro děti předškolního a školního věku a klade za cíl pomoci dětem s poruchou učení, konkrétně dysgrafií, způsobem interaktivní hry. Hru tvoří dlouhý herní plán s cestičkou, kterou hráč musí projet co nejpřesněji prstem až do jejího cíle. Cestička v průběhu herního plánu nabírá tvarů písmen abecedy, které hráč musí svým prstem obkreslit, a tím si tak procvičovat psaní jednotlivých písmen.

Teoretická část diplomové práce popisuje co to je mobilní aplikace jako taková. Popisuje mobilní platformy, vývoj na platformě iOS, komponenty použité při vytváření aplikace a herní framework SpriteKit.

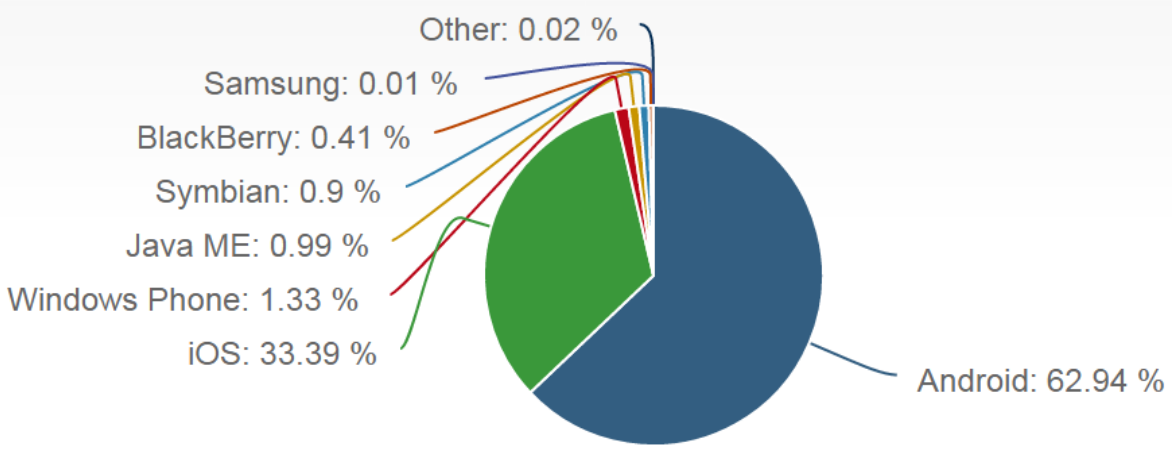
Praktická část dokumentuje postup tvorby mobilní aplikace, od návrhu po samotnou realizaci. Dokumentuje jednotlivé prvky využité při vytváření včetně grafických podkladů. V závěru práce je pojednáno o testování aplikace a možnostech jejího dalšího rozšíření.

I. TEORETICKÁ ČÁST

1 MOBILNÍ APLIKACE

Mobilní aplikace jako taková je softwarová aplikace určená pro mobilní zařízení, jako jsou chytré telefony, tablety a v poslední době i chytré hodinky. Dnes jsou chytrá zařízení tak využívána, že drtivá část lidí se s nějakou mobilní aplikací už setkala nebo je denně používá. Aplikace má za úkol uživateli usnadnit práci, zabavit dlouhou chvíli nebo poskytnout informace, které potřebuje. Výrobci distribuují mobilní zařízení s již předinstalovaným operačním systémem a balíčkem základních aplikací, které uživatel více či méně využije v závislosti na tom, co požaduje. Telefonní adresář, emailový klient, internetový prohlížeč, mapy - to jsou jedny z mobilních aplikací, které najdeme v každém chytrém telefonu nebo tabletu. V jistém momentu může uživatel začít hledat aplikace jiné, které si může stáhnout. K tomu slouží obchody s aplikacemi.

Obchod s aplikacemi si můžeme představit jako samostatnou aplikaci, kde si uživatelé mohou novou aplikaci vyhledat, přečíst k čemu aplikace slouží, zhlédnout její hodnocení a samozřejmě také stáhnout do svého zařízení. Některé aplikace jsou placené, spousta jich je však zadarmo. A v tento moment nastupují vývojáři mobilních aplikací, kteří nové aplikace vytvářejí a následně je umisťují do výše zmíněných obchodů. Každý operační systém má svůj obchod s aplikacemi. Například pro iOS je to AppStore, pro Android je to GooglePlay a ve Windows Phone najdeme Marketplace.



Obrázek 1 – Procentuální využití mobilních operačních systémů [7]

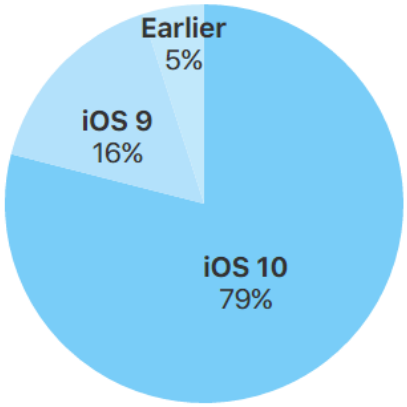
Obrázek 1 nám znázorňuje procentuální pokrytí operačních systémů v zařízeních uživatelů po celém světě. Data jsou aktuální, z března 2017. Jak je možné si na první pohled všimnout, tak drtivá většina zařízení obsahuje operační systém Android či iOS, přičemž Android zabírá bez mála 63% a iOS něco málo přes 33%. Dohromady je to tedy

96% trhu, z čehož vyplývá, že pokud chceme vyvíjet mobilní aplikaci, měli bychom se soustředit na tyto dva operační systémy pro pokrytí co nejvíce potencionálních uživatelů.

1.1 Platforma iOS

Operační systém iOS byl poprvé představen bok po boku prvního telefonu iPhone od firmy Apple v roce 2010. Od té doby si získal pověst stabilního, spolehlivého a intuitivního systému, což mohu jen potvrdit. Postupem času, jak Apple představoval nová mobilní zařízení, jej implementoval i do svých tabletů iPad a multimediálních přehrávačů iPod Touch.

Apple k systému přistupuje velice obezřetně, tak, aby byla zachována jeho hodnota a hodnota samotného zařízení. Firma totiž podporuje svá zařízení velice dlouho. Podporou jsou myšleny aktualizace systému, které Apple vydává pravidelně, několikrát ročně, pro zařízení až 5 roků stará. Ve srovnání s tím, jakou rychlostí se dnes vyvíjí mobilní zařízení je tento přístup velice vítaným prvkem. Ruku v ruce s podporou zařízení jdou i statistiky o aktuálním nainstalovaném operačním systému v zařízeních s iOS. Na obrázku 2 je jasně vidět většinový podíl systému iOS verze 10 se 79%. Obrázek je aktuální k datu 20. únor 2017, to znamená 5 měsíců od vydání samotného systému iOS 10.



Obrázek 2 – Procentuální podíl verzí systémů iOS [8]

Uživatelské prostředí je navrženo tak, aby bylo jednoduché a snadno ovladatelné a zvládl ho tak každý. Uživatel má sice minimální možnost si systém přizpůsobit, ale to vše za dosažením konzistentní jednoduchosti a celistvosti. S lehkou nadsázkou lze říci, že celý systém lze ovládat pouze dotyky a jedním tlačítkem. Výsledkem je tak neměnný uživatelský zážitek bez ohledu na to jaké zařízení s iOS zrovna uživatel používá. To samé lze říci i o samotných verzích iOS. Hlavní funkční prvky napříč verzemi už od samotného

počátku zůstávají stejné a přibývají akorát funkce, které systém obohacují. Uživatel, který zrovna odložil svůj telefon iPhone a přesunul se k tabletu iPad ho bude umět ovládat naprosto stejně jako svůj telefon. Apple s iOS myslel i na lidi s různými tělesnými disfunkcemi. Pro nevidomé je tu možnost, aby s nimi systém mluvil a byl ovladatelný pouze hlasem. Pro lidi s horším zrakem tu jsou možnosti nastavení zvětšení textu nebo lupa. Pro neslyšící iOS nabízí například funkci, kdy zabliká blesk fotoaparátu při příchozím hovoru. U jiných mobilních operačních systémů takové funkce nenajdeme.

1.2 Vývoj aplikací na platformě iOS

Vývoj mobilních aplikací pro iOS zařízení si dneska může vyzkoušet každý, kdo vlastní nějaký počítač značky Apple. Jedinou podmínkou je mít nainstalované vývojové prostředí Xcode, které nabízí Apple zdarma ke stažení. V Xcode vývojář vytvoří aplikaci pomocí programovacího jazyku Swift nebo Objective-C a přímo na počítači si ji může spustit v simulátoru. Právě v simulátoru mohou vývojáři testovat své aplikace na různých velikostech obrazovek a verzích operačních systémů. Oproti jiným platformám mají však svou práci snazší, protože drtivá většina iOS zařízení má nainstalovanou aktuální verzi operačního systému, viz obrázek 2. To pro vývojáře znamená, že mohou vyvíjet vždy pro poslední verzi systému a využít tak všech dostupných funkcí, které jsou aktuálně k dispozici. Jestliže chtějí vytvořit aplikaci pro maximální množství uživatelů, mohou vyvíjet i pro starší verze iOS. Aplikace vyvinuté pro starší verze iOS verze jsou kompatibilní s novými verzemi, vývojáři jen musí počítat s faktem, že nemusí mít k dispozici všechny funkce, které by mohli využít a budou muset vymýšlet alternativy.

Dalším krokem vývoje je si vyvíjenou aplikaci vyzkoušet na reálném zařízení, protože v simulátoru nemusejí být dostupné všechny prvky opravdového zařízení jako je například senzor otisku prstů – TouchID. Dalším důvodem je, že na reálném zařízení jsou občas některé grafické prvky zobrazeny jinak, než v simulátoru. Instalace vyvíjených aplikací do fyzického zařízení je podmíněna více faktory. Prvním z nich je mít založený Apple účet AppleID. Toto AppleID je nutné si zaregistrovat na Apple stránkách určených pro vývojáře. Tímto se z běžného Apple účtu stane zároveň účet vývojářský. Dalším krokem je si předplatit členství v Apple Developer Programu, které zpřístupní možnosti jako: přístup k beta verzím iOS, možnost nahrát aplikaci do reálných zařízení, nahrát aplikace ke schválení společnosti Apple, nechat externí testery otestovat vyvíjenou aplikaci

a samozřejmě nechat zveřejnit aplikaci do AppStore. Vývojáři mají na výběr ze dvou variant:

- **Vývojář jednotlivec**
  Tento Apple Developer Program je určený pro vývojáře, který vytváří aplikaci sám za účelem ji pak distribuovat v AppStore pod svým vlastním jménem. Je zpoplatněn částkou 99 Amerických dolarů na 1 rok. [9]
- **Organizace**
  Tato varianta se rozděluje do dvou podkategorií:
  1) Standardní Apple Developer Program pro organizace, ve které může vystupovat více vývojářů najednou s cílem vytvořit a vložit aplikaci do AppStore. Tento program je zpoplatněn 99 Americkými dolary na 1 rok. [9]
  2) Apple Developer Enterprise Program je určený pro organizace, které mají v záměru vyvíjet aplikace exkluzivně pro své zaměstnance. Enterprise program stojí 299 Amerických dolarů na rok. [9]

Apple si aplikace, které chtějí vývojáři umístit do AppStore, prověřuje svými zaměstnanci. Tímto se snaží předcházet aplikacím, které by mohly vést ke špatnému uživatelskému zážitku nebo dokonce aplikacím závadným, které by například sbíraly informace a data uživatelů.

1.2.1 Programovací jazyk Swift

Programovací jazyk Swift je poměrně nový programovací jazyk. 2. června 2014 společnost Apple na WWDC (Worldwide developer conference) oznámila tenkrát šokující novinku, že má pro vývojáře nový programovací jazyk. Bylo to velké překvapení pro komunitu vývojářů, protože do té doby se používal programovací jazyk Objective-C. [1]

Swift je od základů postavený na těchto charakteristických pilířích:

- **Objektově orientovaný**
  Swift je moderní, objektově orientovaný jazyk, kde vše existuje jako objekt. [1]
- **Přehledný**
  Ve Swiftu je jednoduché programovat a též i číst zdrojový kód. Způsob zápisu je jasný, konkrétní, s minimem skrytých funkcí. [1]

- **Bezpečný**

  Swift nutí vývojáře k silnému typování objektů, aby si byli jistí, že ví, co se v kterémkoliv momentě nachází v daném objektu. [1]

- **Ekonomický**

  Swift je dosti malý programovací jazyk poskytující základní typy a funkce a nic víc. Zbytek musí být naprogramován vývojářem nebo knihovnami, které vývojář používá. [1]

- **Správa paměti**

  Swift spravuje paměť automaticky. Vývojář se shledá se správou paměti jen velmi zřídka. [1]

- **Kompatibilní s Cocoa**

  Cocoa API jsou frameworky určené k fungování aplikací pro iOS. Cocoa API jsou psané v původním Objective-C a klasickém programovacím jazyku C. Swift je navržen tak, aby byl propojený s většinou těchto frameworků. [1]

Tyto vlastnosti dělají jazyk Swift excelentním jazykem pro naučení se a tvorbu mobilních iOS aplikací. [1]

Apr 2017	Apr 2016	Change	Programming Language	Ratings	Change
1	1		Java	15.568%	-5.28%
2	2		C	6.966%	-6.94%
3	3		C++	4.554%	-1.36%
4	4		C#	3.579%	-0.22%
5	5		Python	3.457%	+0.13%
6	6		PHP	3.376%	+0.38%
7	10	^	Visual Basic .NET	3.251%	+0.98%
8	7	v	JavaScript	2.851%	+0.28%
9	11	^	Delphi/Object Pascal	2.816%	+0.60%
10	8	v	Perl	2.413%	-0.11%
11	9	v	Ruby	2.310%	-0.04%
12	15	^	Swift	2.287%	+0.81%
13	12	v	Assembly language	2.168%	-0.03%
14	13	v	Objective-C	2.163%	+0.45%
15	18	^	R	2.138%	+0.87%
16	14	v	Visual Basic	2.058%	+0.45%
17	16	v	MATLAB	2.045%	+0.70%
18	44	^^	Go	1.974%	+1.73%
19	24	^^	Scratch	1.668%	+0.86%
20	17	v	PL/SQL	1.619%	+0.30%

Obrázek 3 – Popularita programovacích jazyků k dubnu 2017 [11]

Apple považuje Swift za úspěšného nástupce Obejctive-C a snaží se jej co nejvíce prosazovat a dále vyvíjet. Aktuální verze je Swift 3 s tím, že tento rok se plánuje představení verze 4. O jeho popularitě vypovídá i obrázek 3. Na obrázku je seřazeno 20 nejpoužívanějších programovacích jazyků na světě k dubnu 2017. Swift se umístil na 12. místě. Můžeme i vidět, že oproti dubnu 2016 si polepšil o 3 místa a předčil tak Objective-C.

Aby se Swift dostal i do povědomí mladých, čili potencionálně budoucích vývojářů, vytvořil aplikaci s názvem Swift Playgrounds určenou pro tablety iPad, dostupnou zdarma ke stažení z AppStoru. Ve Swift Playgrounds má uživatel možnost naučit se základům Swift programování pomocí interaktivních puzzle lekcí, kde má vždy splnit zadaný úkol tím, že napíše část kódu, který se následně provede.

Na následujícím obrázku číslo 4 je pro doplnění logo programovacího jazyku Swift v podobě rychle letícího ptáka.

Obrázek 4 – Logo pro jazyk Swift [14]

1.2.2 Vývoj herní mobilní aplikace

Hry a herní aplikace jsou jedny z nejpopulárnějších kategorií v AppStore. S představením programovacího jazyku Swift se stal vývoj her na iOS více lákavý a jednodušší pro stávající i nové vývojáře. [10]

Vývoj herní mobilní aplikace následuje stejný postup vývoje, jako vývoj klasické mobilní aplikace s tím rozdílem, že je při něm využit nějaký herní framework. Frameworků k vývoji herní aplikace je na výběr spousta. Známé jména jako Unity nebo Unreal Engine se dají využít k vývoji herní aplikace, nicméně, jako spousta dalších frameworků třetích stran, jsou zpoplatněny roční částkou. Apple sám poskytuje tři frameworky, které může vývojář použít při vývoji hry:

- **SpriteKit**

  SpriteKit je framework pro vytváření 2D her. Podporuje spoustu efektů, podporu videí, filtrů a masek. Zahrnuje vestavěnou knihovnu fyziky a spoustu dalšího. Je doporučený pro začínající herní vývojáře. [13]
- **Metal**

  V iOS 8 Apple představil nové API pro 3D grafiku nazvané Metal. Metal má takzvané nízkoúrovňové API pro interakci s 3D grafickým hardwarem. Je navržený pouze pro hardware Apple zařízení, díky tomu je velmi efektivní, poskytující nízkou výpočetní náročnost a nízké režijní náklady oproti jiným 3D frameworkům. [13]
- **SceneKit**

  Je další 3D framework od společnosti Apple pro tvorbu 3D her a jiných 3D scén. SceneKit obsahuje engine fyziky, generátor částic a skriptování animací a animace 3D objektů. [13]

2 STRUKTURA MOBILNÍ IOS APLIKACE

Struktura mobilní aplikace, tak jak ji vidí uživatel, se skládá z jednotlivých prvků, jako jsou texty, obrázky, tlačítka a další. Těmto prvkům se říká View. Všechna View, které uživatel na dané obrazovce vidí, jsou umístěna ve ViewController, což je kontrolér, který řídí zrovna zobrazené okno. Úlohou kontroléru je obsluhovat a zpracovávat všechna zobrazené View a uživatelské zásahy. Jak a co bude kontrolér dělat, už závisí na tom, jak bude naprogramován.

2.1 Komponenty pro zobrazení obsahu

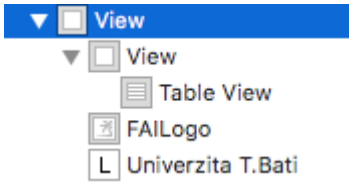
Primárním prvkem pro zobrazení obsahu je vestavěný framework UIKit. UIKit poskytuje stěžejní infrastrukturu potřebnou k vytvoření a řízení iOS aplikací. Do architektury spadají okna aplikace a jednotlivá View potřebná ke správě UI (uživatelského rozhraní), odezvy na interakce uživatele a události systému, práce s animacemi, textem a obrázky. [12]

UIKit obsahuje spoustu základních objektů. Vývojář tak nemusí žádné objekty vytvářet, spíš používat a upravovat ty, které mu iOS již nabízí. Níže jsou popsány View objekty používané v aplikaci.

2.1.1 UIView

UIView je základní objekt pro zobrazení obsahu a všechny ostatní objekty jako jsou UITableView, UIImageView, UIAlertView a další jsou podtřídami právě UIView. UIView jako objekt definuje obdélníkovou oblast na obrazovce a spravuje vše, co se v této oblasti nachází. Je možné například vytvořit UIView o daných rozměrech, určit jeho pozici, nastavit mu barvu pozadí a vložit do něj tlačítko. Tlačítku lze též nastavit atributy pro zobrazení a funkci, kterou má vykonávat. Po spuštění aplikace, se nám zobrazí vytvořené UIView s tlačítkem. Tímto způsobem aplikace může fungovat velice jednoduchým způsobem „Nastav a zapomeň“. Nicméně s View se dají dělat i složitější operace. View se dá upravit způsob zobrazení, nechat jej zmizet, posunout ho, změnit mu velikost, nastavit mu animaci a další. View je též zodpovědné za uživatelská gesta jako například dotyk jedním prstem, posunutí prstem nebo posunutí dvěma prsty a další. Takže jednotlivá View nejsou jen prvky pro zobrazení obsahu, ale též prvky se kterými může uživatel interagovat. [3]

Jak bylo zmíněno výše, UIView může obsahovat jiná View. Ono jiné View se pak stává Subview pro UIView a naopak, pro vložené View se stává UIView Superview. Tato stromová hierarchie dovoluje přeskládat a měnit pozice View dle potřeby. Taktéž dovoluje vytvořit UIView se všemi Subview a když se UIView zobrazí, zobrazí se se všemi Subview dohromady. Naopak, když bude UIView odebráno, skryto nebo posunuto, stane se totéž se všemi Subview. [3]



Obrázek 5 – Hierarchie jednotlivých View

2.1.2 UITableView

UITableView je vertikálně posuvné View obsahující řádky. Tyto řádky jsou obdélníkové buňky třídy UITableViewCell, jenž je podtřídou UIView. TableView je klíčovým prvkem strategie Applu, neboť TableView umí využit malého displeje telefonu pro následující účely: [3]

- **Prezentování informací**

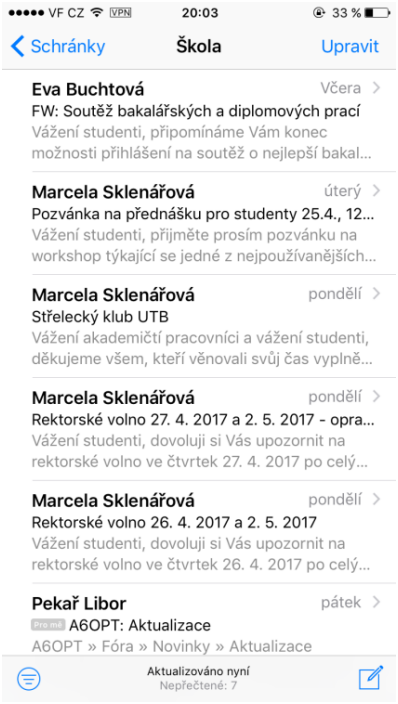
  Buňky typicky obsahují text, který si uživatel může přečíst. Buňky jsou většinou docela malé, aby se maximalizovalo kvantum informací zobrazených v jeden moment na ploše obrazovky. Text často bývá zkrácený, zjednodušený nebo nahusto napsaný. [3]

- **Výběr**

  TableView může poskytnout uživateli buňky s možnostmi. Uživatel pak vybírá možnosti dotykem na konkrétní buňku, což danou buňku označí a aplikace by měla vhodně zareagovat na výběr uživatele. Klasickým příkladem je nastavení systému iOS. [3]

- **Navigaci**

  Vhodná odezva na výběr buňky je navigace uživatele do jiného rozhraní. Například, když uživatel klikne na nepřečtený email a dostane se tak do náhledu emailu, kde si jej může přečíst. Původní buňka tak může nést název předmětu emailu a drobný text s obsahem emailu, viz obrázek 6.



Obrázek 6 – TableView

TableView disponuje i dalšími možnostmi. Buňky mohou být shlukovány do takzvaných sekcí. Každá sekce může být určena pro zobrazení jiných informací nebo nastavení. TableView může být též editovatelné uživatelem. Uživatel může mít svolení vkládat nové buňky, mazat buňky nebo měnit jejich pořadí. [3]

2.1.3 UIImageView

UIImageView je UIView, které je určené pro zobrazování obrázků. IOS umí pracovat s formáty obrázků jako TIFF, JPEF, GIF a PNG, ale nejvíce zaměřený je iOS právě na PNG a vývojář by se měl snažit prosazovat tento formát všude, kde je to jen možné. Obrázky mohou být umístěné v samotném projektu aplikace, nebo mohou být staženy, nejlépe ale, když jsou umístěny v takzvaném *Asset* katalogu, kde jsou k nalezení i defaultní ikony aplikace. Obrázky pak jsou lehce dostupné v celém projektu aplikace. [3]

UIImageView umí uchovávat dva obrázky zároveň. Jeden jako výchozí a druhý pro případ, že uživatel vybere daný ImageView. Příkladem může být obrázek žárovky. Ve výchozím stavu bude UIImageView zobrazovat obrázek zhasnuté žárovky. Po kliknutí na ImageView se změní obrázek na rozsvícenou žárovku. [3]

Poněvadž UIImageView je podtřídou UIView, lze mu nastavovat stejné atributy jako UIView. Důležité je nastavení barvy pozadí. ImageView může mít ve stejný moment nastavenou barvu pozadí a zobrazovat obrázek. Jelikož jsou preferované obrázky formátu

PNG, mohou mít obrázky toho formátu nastavenou hodnotu *alpha*, čili průhlednost. ImageViw tuto průhlednost obrázku respektuje a průhlednosti v obrázku zobrazí. UIImageView bez nastavené barvy pozadí je ve své podstatě neviditelný, až na zobrazovaný obrázek, takže se uživateli objeví čistě jen obrázek bez toho, aby si uživatel všimnul, že je umístěn v nějakém obdélníkovém View. Někdy je toto záměr, někdy je potřeba se tomuto vyvarovat. UIImageView bez nastaveného obrázku a barvy pozadí je kompletně neviditelný. V takovém stavu je možné jej vytvořit a později, až například uživatel zvolí profilovou fotku, jej zobrazit na pozici, kde je UIImageView umístěno. [3]

Dalším důležitým atributem pro UIImageView je *contentMode*, neboli způsob zobrazení obrázku. UIImageView má při zobrazení na zařízení nastavené své hodnoty pro šířku a délku v pixelech. Samotný obrázek vyobrazený pomocí ImageView však z pravidla má rozměry větší, nebo naopak menší. To může vést k tomu, že je obrázek nevhodně roztažený, nebo není zobrazený celý. Proto je potřeba nastavit *contentMode* na požadovanou hodnotu. Níže jsou popsány nejběžněji používané způsoby, jakými lze obrázek zobrazit:

- **scaleToFill**

  Tento způsob zobrazení upraví výšku i šířku obrázku přesně na velikost samotného UIImageView. Na obrázku 7 je vidět, že UIImageView bylo na obrázek loga Fakulty aplikované informatiky příliš úzké a o něco vyšší.

Obrázek 7 – Zobrazení scaleToFill

- **aspectFit**

  UIImageView zobrazený na obrázku 8 má nastavenou fialovou barvu pozadí a přizpůsobení obrázku *aspectFit*. Jak je vidět, obrázek loga Fakulty aplikované informatiky je zobrazen správně, ikdyž velikost samotného UIImageView je vyšší.

Obrázek 8 – Zobrazená obrázku aspectFit

- **aspectFill**

  Způsob zobrazení *aspectFill* vezme obrázek a roztáhne jej tak, aby vyplnil celé UIImageView. Na obrázku 9 si lze povšimnout, že ImageView je vyplněné jak na výšku, tak i na šířku. Nicméně logo je větší než samotné ImageView a přesahuje ve své šířce z ImageView ven. Někdy takovýto výsledek může být záměrný.

Obrázek 9 – Zobrazení obrázku aspectFill

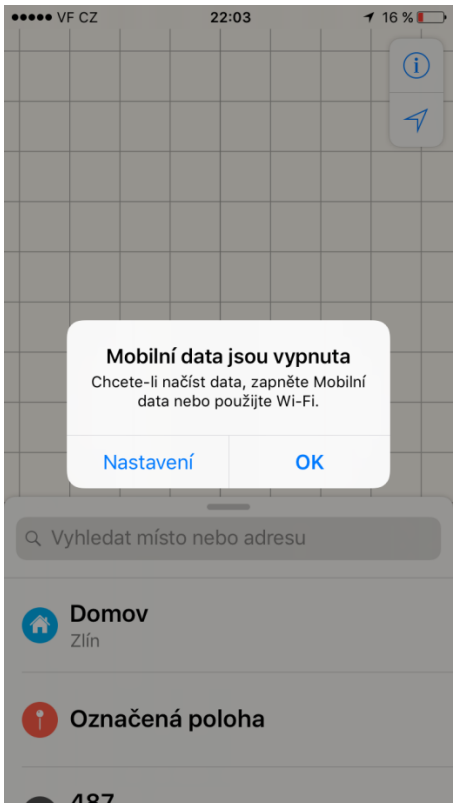
Jestliže nám obrázek přesahuje rozměry ImageView, můžeme ImageView nastavit atribut *clipToBounds* na hodnotu *true*. Tím zamezíme přesahování okrajů a obrázek zůstane ořezaný, jak je tomu na obrázku 10.

Obrázek 10 – Zobrazení obrázku aspectFill s clipToBounds

2.1.4 UIAlertView

AlertView se zobrazuje uprostřed obrazovky, obvykle nečekaně, s patřičnou animací, tak, aby upoutal pozornost uživatele. Obsahují nadpis, zprávu, která sděluje informaci uživateli a tlačítka, které dávají uživateli možnost nějak na AlertView zareagovat. Jedno z těchto tlačítek by mělo z pravidla být tlačítko zavírající AlertView. [3]

AlertView je záměrně minimalistické pro rychlé a jednoduché zobrazení důležité informace a následné interakce. Často obsahuje pouze uzavírací tlačítko. Dodatečná tlačítka už dávají uživateli možnost, tak jako je to na obrázku 11.



Obrázek 11 – Ukázka AlertView

2.1.5 UILabel

UILabel je objekt pro jednoduché zobrazení textu s atributy k nastavení jako: font, velikost, barva textu, zarovnání textu, stín textu a offset stínu, a spoustu dalších. Užitečným atributem je i zvýraznění textu, to například, když uživatel vybere nějaký prvek, ve kterém bude UILabel právě umístěn. [3]

Obrázek 12 – UILabel s nastaveným textem, pozadím a stínem

2.2 Kontroléry

Uživatelské rozhraní na iOS je dynamické. Na stolních počítačích mohou být aplikace velké a může jich být spuštěno hned několik zobrazujících se vedle sebe. Na iOS tolik místa není, takže na iOS je potřeba vytvořit nějakou hierarchii jednotlivých View, přičemž každé View má svůj účel a překrývají se jedno přes druhé v závislosti na tom, jak se uživatel pohybuje v dané aplikaci. [3]

V iOS je dynamičnost jednotlivých View spravována pomocí kontrolérů. Každé View má právě jeden ViewController, což je instance třídy UIViewController. Nebo naopak, každý ViewController musí mít své View, bez kterého by byl zbytečný. Existují i podtřídy jako UITableViewController, UITabBarController, UINavigationController a mnoho dalších. Ty se používají v závislosti na tom, jakým způsobem je View navržené a co v něm bude potřeba zobrazit. Nejdůležitější úkol ViewControlleru je starat se o své View. Aby View mělo nějaký přínos, musí se nějakým způsobem zobrazit na zařízení. Zobrazení View počíná přechodem z jiného View a z pravidla bývá doprovázeno animací. To samé platí, když nějaké View mizí z obrazovky a zobrazí se View předchozí. [3]

2.3 Přechody mezi kontroléry

Pro navigování mezi jednotlivými View se používají přechody. Přechod se nazývá Segue a má pevně dané destinace, které tvoří počáteční a cílový ViewController. Samotný Segue je instancí objektu UIStoryboardSegue a je vytvářen v moment, kdy se přechod má provést. Počáteční ViewController je ten, který už existuje a ze kterého Segue vychází. Zatímco cílový ViewController je vytvářen společně se Segue. [3]

Nejběžněji používané jsou následující Segue:

- **Show**

  Používá se v navigačním kontroléru. Ten má za úkol uchovávat zásobník jednotlivých View v pořadí, v jakém uživatel postupoval napříč aplikací. Počáteční

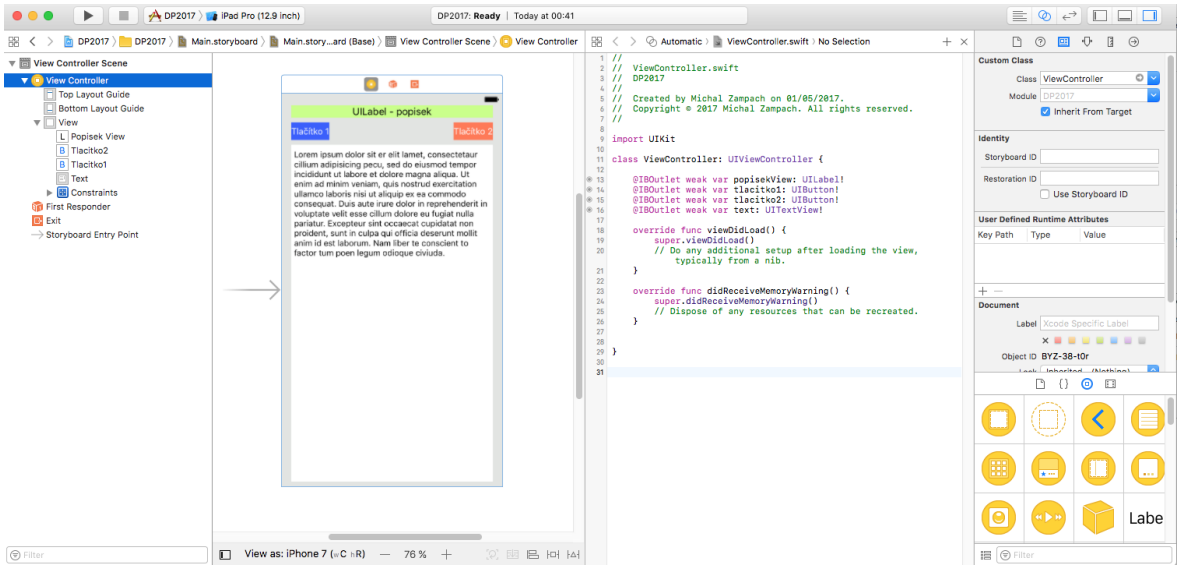
kontrolér je tedy aktuálně zobrazovaný a cílový kontrolér je ten, který se zobrazí před ním. [3]

- **Present modally**
  Present modally jednoduše zobrazí budoucí ViewController před aktuálním a z aktuálního udělá původní, pro případ zpětné navigace. Není zde potřeba používat navigační kontrolér. [3]

2.4 Storyboard

Storyboard je jedinečný fragment vývoje aplikací v iOS. Je možnost vytvářet všechna View a jejich kontroléry programově. Druhá, pohodlnější cesta, je vytvářet je pomocí vizuálního návrhu. Storyboard si lze představit jako takové plátno, kde se dá graficky navrhnout hierarchie jednotlivých zobrazovaných obrazovek. Co každá obrazovka bude zobrazovat a jaké budou mezi nimi přechody – segue.

Ve Storyboard návrhu je tak možnost si například navrhnout celý TableView kontrolér i se samotným UITableView. V TableView si nastavit počet buněk, nastavit si každou buňku zvlášť, přidat obrázky, tlačítka a další prvky uživatelského rozhraní. Toto ušetří spoustu práce při programování, jak při samotném vytváření, tak i umisťování a chování. Ostatní programová logika je však potřeba doplnit do příslušných tříd, které se navrženému kontroléru předávají v samotném Storyboardu. Ostatní View patřící do daného kontroléru se musí pomocí referencí přidat do vytvořené třídy, aby třída věděla, že nějaká další View v kontroléru existuje a vývojář s nimi mohl dále pracovat. Na následujícím obrázku číslo 13 je znázorněn Storyboard s vytvořeným kontrolérem a příslušící třídou.



Obrázek 13 – Storyboard s třídou patřící vybraném kontroléru

Je však potřeba dodržovat náležitosti, aby zůstal zdrojový kód přehledný. Jestliže se v aplikaci používá Storyboard, neměly by se pak nové kontroléry a jejich View vytvářet programově. Při dalších úpravách to může vést k tomu, že se musí složitě hledat, kde se jaký kontrolér vytváří.

Další náležitostí je vytvářet víc Storyboard návrhů. To platí v případě, že se aplikace skládá z většího počtu kontrolérů. Storyboard je vytvořen v momentě, kdy je potřeba ho využít a zobrazit nějaký z kontrolérů, který je v něm navržený. Vytvořen je však se všemi dalšími kontroléry a view. To znamená větší výpočetní a paměťovou náročnost. Lepší je tak rozsáhlejší aplikaci rozdělit do více návrhů.

2.5 Interakce uživatele s aplikací

Touch, neboli dotyk, je instance prstu dotýkající se obrazovky. Systém společně s dotykovou vrstvou pracují dohromady a ví, kdy se prst uživatele dotkl obrazovky a také, kde to bylo. I když je prst tlustý, je lokace dotyku obrazovky chytře zredukována na jeden bod. Adresát, kam přijdou všechny informace o dotyku je třída UIResponder. UIResponder má spoustu podtříd. Jednou z nich je UIView a ta je taky jako jediná viditelná pro uživatele. Dotyk je reprezentován instancí třídy UITouch, která je svázána do pomyslné obálky třídy UIEvent, kterou systém doručí do aplikace, kterou uživatel zrovna používá. Pak záleží na aplikaci, jak na event a na dotyk zareaguje. [3]

Výchozí stav, kdy uživatel na nic nekliká je takzvaný „finger-free“. Tento stav přetrvává do doby, než se uživatel dotkne jedním nebo více prsty. Znovu pak nastává, když se uživatel přestane dotýkat obrazovky. Veškeré dotyky a gesta Apple nazývá *Multitouch sequence*. [3]

Systém, během multitouch sekvence, posílá aplikaci hlášení o každé změně pohybu prstu, takže aplikace si může odvodit to, co uživatel právě dělá. Každé jednotlivé hlášení je objektu UIEvent. Ve skutečnosti tak přichází do aplikace série objektů UIEvent. V každém z nich je pak aktuální rozmístění prstů uživatele na obrazovce. [3]

Každý UIEvent přinášející změnu rozmístění prstů obsahuje jednu nebo více UITouch objektů. Každý UITouch objekt koresponduje s jedním prstem uživatele. Jinak řečeno, každý prst dotýkající se obrazovky je reprezentován UITouch objektem v UIEventu. Jakmile je UITouch instance vytvořena pro daný prst, který se dotýká obrazovky, je ta samá UITouch instance používána v multitouch sekvenci až do doby, dokud prst neopustí obrazovku. [3]

Systém však nezahlcuje aplikaci neustálým přísunem informací o pozicích prstů. Systém oznamuje pouze změny polohy prstů. Pro daný UITouch objekt, reprezentující jeden prst, mohou nastat čtyři stavy. Tyto stavy jsou nazývány *Touch phases*, neboli fáze dotyku: [3]

- **began**

  Prst se poprvé dotkl obrazovky a tím se právě vytvořila instance třídy UITouch. Toto je vždy první fáze a vykonává se vždy jednou. [3]
- **moved**

  Fáze, kdy se prst posouvá po obrazovce a mění svou polohu. [3]
- **stationary**

  V této fázi prst setrvává na jednom místě. O této fázi systém neinformuje jen tak bez příčiny, ale jak bylo výše zmíněno, instance UITouch existuje až do doby, dokud se prst nevzdálí od obrazovky. Proto, když nastane nějaká další událost, například uživatel se dotkne dalším prstem, je potřeba informovat o tom, že původní prst se nepohnul. [3]
- **ended**

  Moment, kdy se prst přestane dotýkat obrazovky je sdělen fázi *ended*. Stejně jako fáze *began* je tato fáze vykonávána pro objekt UITouch právě jednou. Konkrétní

UITouch instance během této fáze zanikne a nikdy více se již v multitouch sekvenci neobjeví. [3]

- **cancelled**

  Nastává, když systém přeruší multitouch sekvenci, protože ji něco přerušilo. Například, když uživatel zmáčknul tlačítko na vypnutí obrazovky zrovna v momentě, kdy probíhala multitouch sekvence. Jestliže se v aplikaci pracuje s gesty uživatele, je zapotřebí odchytávat i stav *cancelled*. Díky němu se můžou vrátit prvky do výchozích stavů. [3]

2.6 Animace

Animace je vlastnost nějakého objektu, která se mění v čase. Změna atributu může být poziční: objekt se posune nebo změní svou velikost, ne skokově, ale plynulou změnou. Ostatní atributy mohou být animované taktéž. Barva pozadí se může změnit ze zelené na červenou, ne náhlou změnou barev, ale vyblednutím jedné do druhé. View se může změnit z viditelného stavu do průhledného, ne náhlým zmizením, ale postupným vymizením. [3]

V rámci animací je celá řada komplikací, jako jsou výpočty, časování, obnova obrazovky a mnoho dalších. Naštěstí iOS poskytuje vývojáři v tomto ohledu pomoc. Vývojář nevytváří animaci celou sám, pouze zadá, jaké změny se mají stát, seřadí je a systém už se postará o jejich vykonání a vykreslení. Apple se snažil o co největší usnadnění animací. Animace jsou totiž stěžejním prvkem v uživatelském rozhraní iOS. Není to pouze na parádu, jak se říká, ale taky je to příznakem, že se něco děje a reaguje. [3]

Animace lze tedy vytvářet změnou atributů objektu a nastavením časové prodlevy, po jakou se mají vykonávat. Animaci lze nastavit i časové odložení, za jak dlouho se má animace vykonat a jestli se má po animaci stát něco dalšího.

Dalším typem animací využívaným v této diplomové práci jsou animace vlastní, neboli vytvořené z jednotlivých obrázků. Příkladem může být skákající králíček. Takovou animaci nám samotné iOS nevytvoří. Slouží k tomu *texture atlas*. To je skupina příbuzných textur, obrázků, použitých k vykreslení animace. Více je o této animaci popsáno v kapitole 2.6.4 SKTextureAtlas. [12]

2.7 Framework SpriteKit

SpriteKit je Framework přímo od společnosti Apple představený v září 2013 společně s iOS verze 7. Je to grafický 2D renderovací framework který poskytuje možnosti jednoduše animovat textury, vykreslovat obrázky, přehrát video, renderovat text a přidat částicové efekty. Též obsahuje vestavěnou knihovnu pro simulaci fyziky. Sprite Kit je první vytvořený herní engine vestavěný přímo do iOS SDK. [10]

2.7.1 SKNode

SKNode je podstatným stavebním prvkem většiny obsahu frameworku SpriteKit. Třída SKNode nevykresluje žádný vizuální obsah. Primární role třídy je poskytnout základní vlastnosti, které ostatní podtřídy použijí. Veškeré vizuální elementy ve hře založené na frameworku SpriteKit využívají předdefinované podtřídy SKNode. [12]

Objekty SKNode jsou hierarchicky organizované do stromové struktury, podobně, jako fungují Subview u UIView. Zcela běžně je stromová struktura definována rodičovskou instancí třídy SKScene. SKScene je tak kořenový Node a veškerý obsah v ní jsou potomci. SKScene provádí animační smyčku, která zpracovává akce jednotlivých Node tříd, simuluje fyziku a posléze vykresluje obsah celé stromové struktury na displej. [12]

Každý Node, ve stromové struktuře, poskytuje souřadnicový systém svým potomkům. Posléze, kdy je Node přidán do stromové struktury, je umístěn do souřadnicového systému svého rodiče. Jsou mu tak předány hodnoty os x a y, plus hodnota rotace. V momentě, kdy je souřadnicový systém pozměněn, tak se tato informace přenese na všechny potomky daného Node. [12]

Všechny Node třídy jsou objekty, které dokáží reagovat přímo na uživatelské interakce. Lze též převést souřadnicový systém a určit, jestli se uživatelův prst leží přímo na některém objektu SKNode. Taktéž lze zjistit průsečíky mezi jednotlivými Node ve stromové struktuře a zjistit, jestli se jejich fyzické oblasti překrývají. [12]

Kterýkoliv Node, může vykonávat nějakou akci, která je použita k provedení animace, přidání nebo odebrání dalšího Node, přehrání zvuku a dalších. Akce jsou srdcem animačního systému ve SpriteKit frameworku. Další vlastností SKNode je fyzické tělo, což je objekt, který simuluje fyzické vlastnosti objektu. Když má Node fyzické vlastnosti,

simulace fyziky automaticky vypočítává novou pozici pro fyzické tělo a následně ho posouvá a otáčí, než odpovídá vypočítané pozici. [12]

2.7.2 SKScene

SKScene reprezentuje scénu s obsahem. SKScene je kořenový Node všech Node ve stromové struktuře. Tyto Node, poskytují obsah, který scéna animuje a vykresluje na displej. Scéna je reprezentována pomocí View. Obsahuje vlastnosti, které definují, kde je počátek scény a její velikost. Pokud scéna neodpovídá velikosti View, může být přizpůsobena, tak by se do View vešla. [12]

2.7.3 SKAction

SKAction je objekt, který je volán pomocí třídy SKNode ke změně její struktury nebo obsahu. Většina akcí je animována pomocí samotného SpriteKit frameworku. Každá animace má následující vlastnosti, když je vykonávána: *duration* (délka trvání), *timingMode* (tempo jakým jsou animace vykonávány), *speed* (rychlost jakou animace provedeny, například hodnota 2.0 znamená, že animace proběhne 2x rychleji). [12]

Mnoho akcí může být přehráno pozpátku. To dovoluje vytvořit další objekt SKAction, kterým se objekt vrátí do původních hodnot. Například, když akce pohne s Node objektem o 20 bodů doprava, zpětná akce jej vrátí o 20 bodů doleva. Některé akce obsahují jiné akce jako potomky a jsou vykreslovány patřičným způsobem: [12]

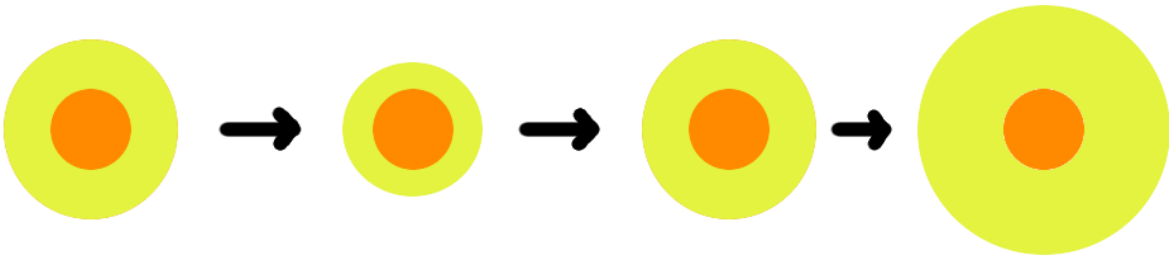
- Posloupnost akcí má vícenásobný počet potomků třídy SKAction. Každá akce v posloupnosti započne, až předcházející skončí.
- Skupina akcí, které mají vícenásobný počet potomků třídy SKAction. Všechny umístěné ve skupině se začnou vykonávat zároveň.
- Opakující se akce uchovávající pouze jednoho potomka SKAction. Když potomek dokončí svou akci, je restartován a začíná od znovu. [12]

Posloupnosti akcí, skupiny akcí a opakující se akce mohou být vnořené. Kombinací akcí je možno vytvořit velice sofistikované vlastnosti jednotlivých Node. [12]

2.7.4 SKTextureAtlas

Objekt SKTextureAtlas je kolekce textur vytvořených během kompilace z patřičných složek projektu nebo vytvořených programově z *Asset Catalog* během spuštěné aplikace. Atlas textur vylepšuje využití paměti a vykreslování. Je doporučeno vykreslované textury

umisťovat do atlasů. Ty jsou pak využity při vykreslování vlastních animací, například blikání hvězdy může být animace sestávající ze čtyř textur, obrázků jako na obrázku 14. [12]



Obrázek 14 – Ukázka animace hvězd ze 4 obrázků

Vytvoření atlasu z *Asset Catalog* má tu výhodu, že v případě, kdy jsou sady stejných textur, ale různých rozlišení umístěny do katalogu, tak se uživateli stáhnou pouze ty, které jsou ideální pro jeho zařízení. Aplikace tak ušetří paměť na úložišti v zařízení a zlepší se výkon vykreslování. Vykreslování velkých textur pro 12“ iPad Pro na 7,9“ iPad Mini by bylo neoptimální. [12]

Jakmile je vytvořen atlas, je pomocí SKAction zavolána metoda *animate*. Této metodě je předán samotný atlas a nastaven časový interval *timePerFrame*, který určuje, jak dlouho se jednotlivé textury budou zobrazovat, než přejde animace na další. Čas je udáván v sekundách.

2.8 Ukládání obsahu

V následující části jsou popsány způsoby ukládání dat v iOS aplikaci.

2.8.1 UserDefaults

UserDefaults je persistentní úložiště určené pro uživatelské nastavení a stavy aplikace. Nejsou primárně určena k ukládání velkého objemu dat. Fungují na principu slovníku, kdy pro daný výraz vrátí uloženou hodnotu, pokud takový výraz byl někdy uložen a jestli je v něm nějaká hodnota uložena.

UserDefaults není třeba vytvářet, existují již na úložišti v zařízení a jakou hodnotu si aplikace uloží, k takové má pak přístup. Stačí tedy jednoduše hodnoty ukládat a zpětně načíst. Důležité je zajistit, aby se hodnoty ukládaly předtím, než se aplikace stane neaktivní (uživatel opustí aplikace). [3]

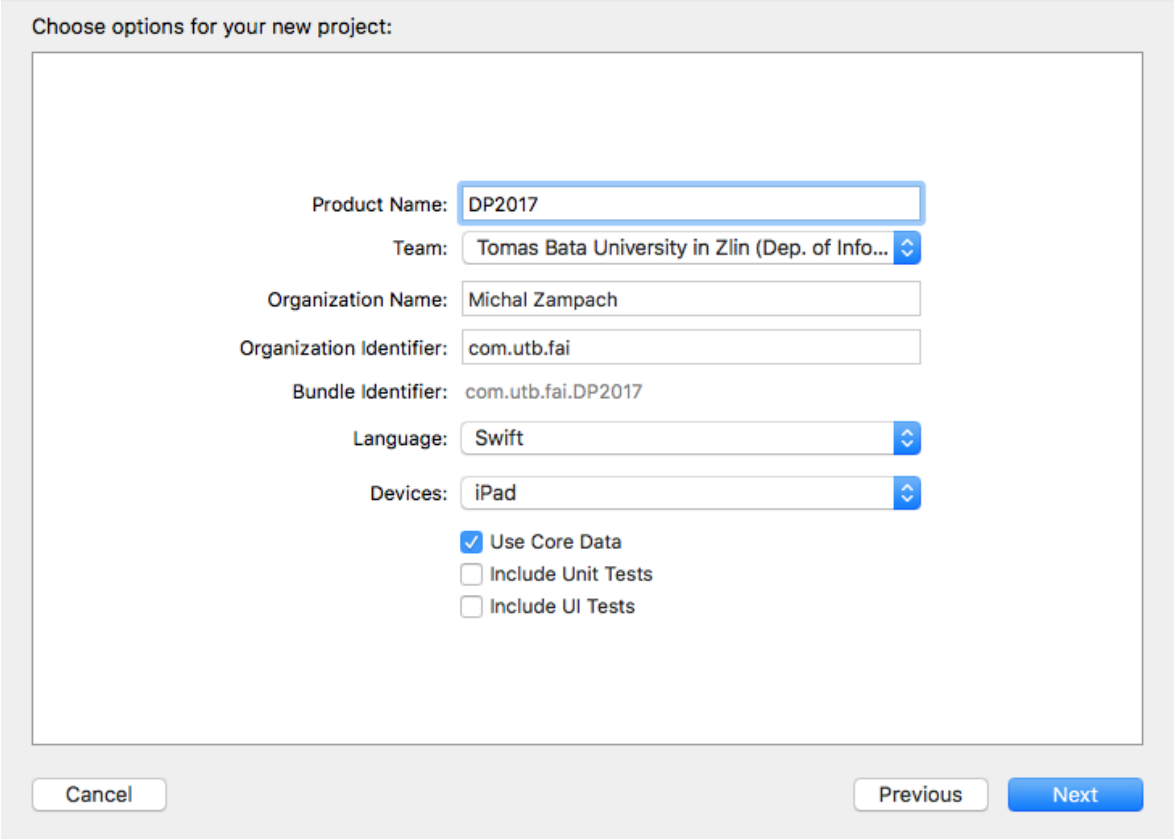
Užitečným použitím UserDefaults je si do nich ukládat stavy kontrolérů, které byly používané uživatelem. Uživatel může mít rozepsaný nějaký text, rozehranou hru a přesunutím aplikace na pozadí, nebo vypnutím aplikace by mohl o svůj rozpracovaný stav přijít. Takže poté, co uživatel aplikaci znovu spustí, se bude moci vrátit zpátky k tomu, kde skončil. [3]

2.8.2 CoreData

CoreData je framework, který poskytuje vyjádření objektů a vlastností objektů formou relačního grafu. Navíc dokáže perzistentně ukládat data objektů přímo do paměti zařízení, a když je potřeba, tak je i zpětně načítat. Coredata jsou užitečným nástrojem například v případech, kdy máme uživatele, který může mít více adres, ale také může mít více přátel, kteří jsou taky objektu uživatel a ti mají taky své adresy a přátele. Coredata umí pracovat s těmito vztahy, umí pracovat s objekty v paměti a v paměti zařízení, sledují a upravují objekty při změnách, kdy je například uživatel smazán. [3]

Nevýhodou frameworku CoreData je, že to není jednoduchá technologie. Je náročná na použití a extrémně náročná na odladění. Je reprezentovaná nepružnou a upovídanou cestou a neměla by být považována za náhražku skutečných relačních databází, CoreData fungují zcela odlišně. [3]

Nicméně jsou CoreData velice užitečným pomocníkem, navíc vestavěným přímo v samotném iOS SDK a připravené pro použití. Stačí je pouze importovat do našeho projektu, ať už ručně nebo z předvolby při vytváření projektu, viz obrázek číslo 15 níže.



Obrázek 15 – Vytváření projektu v programu XCode

Objekty v databázi jsou reprezentovány pomocí entit. Každá entita má své atributy a své vztahy k jiným entitám. Vztahy mohou být klasických typů, jako jedna ku jedné, jedna ku více nebo M:N. Entity se vytváří v modelu entit. Pro představu bude vytvořena entita „Pes“ s atributy jméno, barva, číslo známky. V programové části pak mohou existovat objekty entity Pes a při uložení do paměti se uloží entita Pes tolikrát, kolik bude vytvořených objektů. V paměti zařízení pak mohou být uloženy 3 entity typu Pes, každá s jiným jménem, barvou a číslem známky.

Model entit je vytvořen automaticky, v závislosti na tom, jestli byla při vytváření projektu zaškrtnuta volba jejich použití. Jestliže ne, je třeba je vytvořit ručně. Coredata automaticky všechny entity převádí do relačního grafu pro lepší přehlednost. Do grafu se dá přepnout v modelu entit a podívat se tak, jaké náležitosti mezi jednotlivými entitami existují, případně je v grafu změnit, přidat či odebrat. Modely mohou být verzovatelné nebo statické. U verzovatelných je výhodou, že při zamýšlené úpravě modelu se vytvoří nová verze, a když si pak uživatel aktualizuje aplikaci, bude aplikace pracovat s poslední aktuální verzí. To u statických modelů nejde. Jakákoliv změna statického modelu způsobí pád aplikace.

II. PRAKTICKÁ ČÁST

3 NÁVRH MOBILNÍ APLIKACE

Cílem této diplomové práce je navrhnout a vytvořit mobilní aplikaci pro platformu iOS, která bude určena pro zařízení iPad. Aplikace má záměr pomáhat dětem předškolního a školního věku se specifickou poruchou učení, konkrétně dysgrafií. Pomáhat má tím způsobem, že budou dětem dostupné herní plány, na kterých bude cestička. Cestička v průběhu herního plánu nabírá různých tvarů a písmen. Tuto cestičku budou děti muset projet svým prstem co nejpřesněji, aby se dostaly až na konec cesty. Díky tomu, že cestička nabírá tvarů písmen, se děti mohou naučit jejich psaní už v předškolním věku. Dysgrafii lze u dětí vypozorovat již v předškolním věku, ale skutečný problém pro ně začne přestavovat až po nástupu do školy, kde se učí psát. Tím, že si děti budou procvičovat psaní písmen během hraní, mohou být jejich problémy s dysgrafií potlačeny a nemusí jim dělat takové problémy.

Aplikace bude vyvíjena ve vývojovém prostředí XCode programovacím jazykem Swift verze 3. Aplikace se nebude přizpůsobovat otáčení displeje, bude takzvaně pouze „na výšku“ zařízení iPad. K vývoji bude využit Apple Developer program poskytnutý Univerzitou T. Bati.

3.1 Funkční a nefunkční požadavky

Návrh mobilní aplikace musí mít dané funkční a nefunkční požadavky, které aplikace musí splňovat. Mezi funkční požadavky aplikace patří následující body:

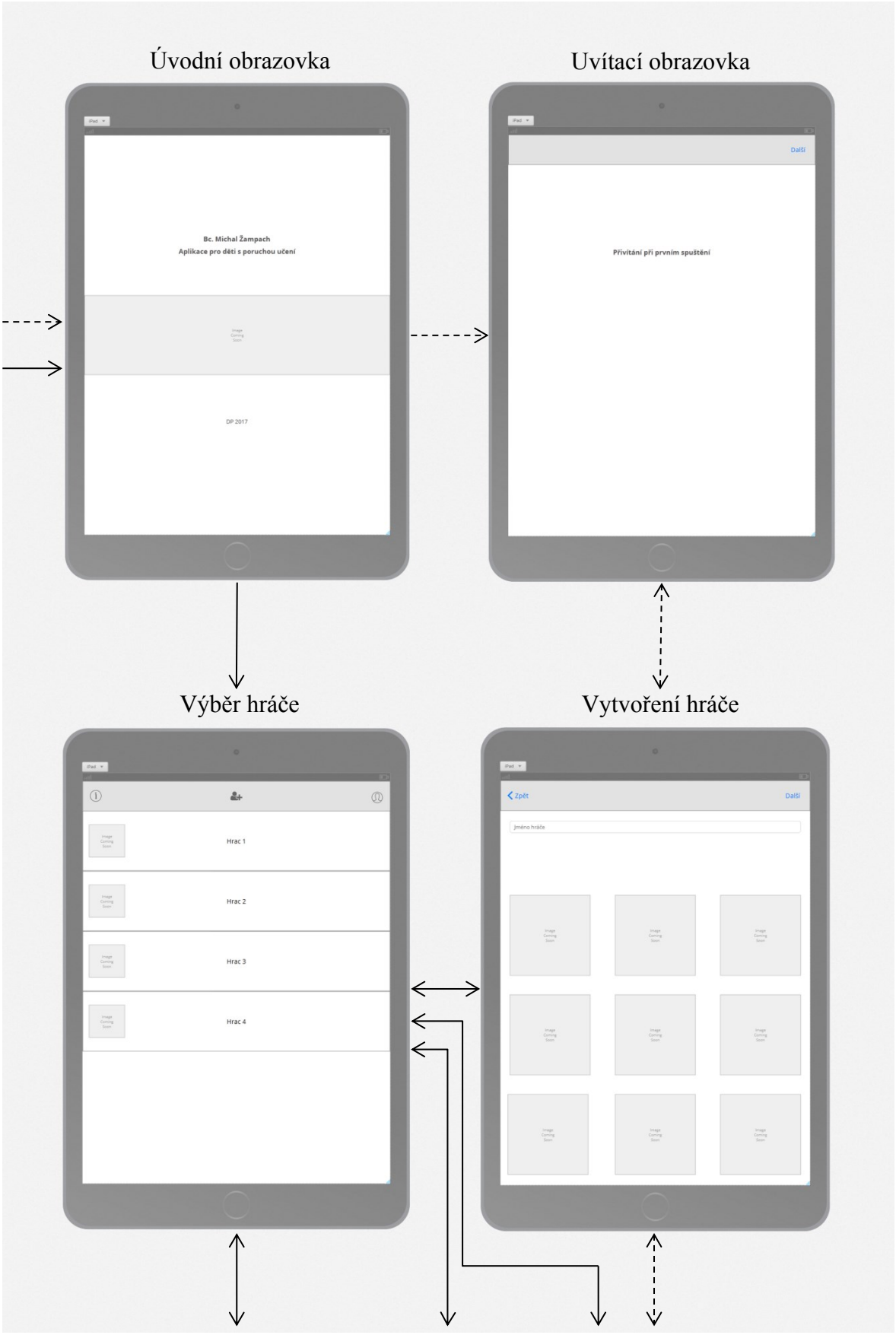
- nový hráč si může zvolit jméno a vybrat obrázek;
- každý hráč má přístup k herní části s kapitolami;
- hráče je možné smazat;
- může být nastaven rodič s číselným zámkem;
- rodič může nastavit dobu, jak dlouho může hráč hrát, než bude přerušen;
- rodič může herní dobu povolit nebo vypnout;
- rodič může uvést aplikaci do výchozího stavu – vymazat všechna data aplikace;
- rodič může smazat sám sebe;
- hráč může kapitolu hrát od znovu po jejím zdárném dokončení.

Do nefunkčních požadavků patří následující:

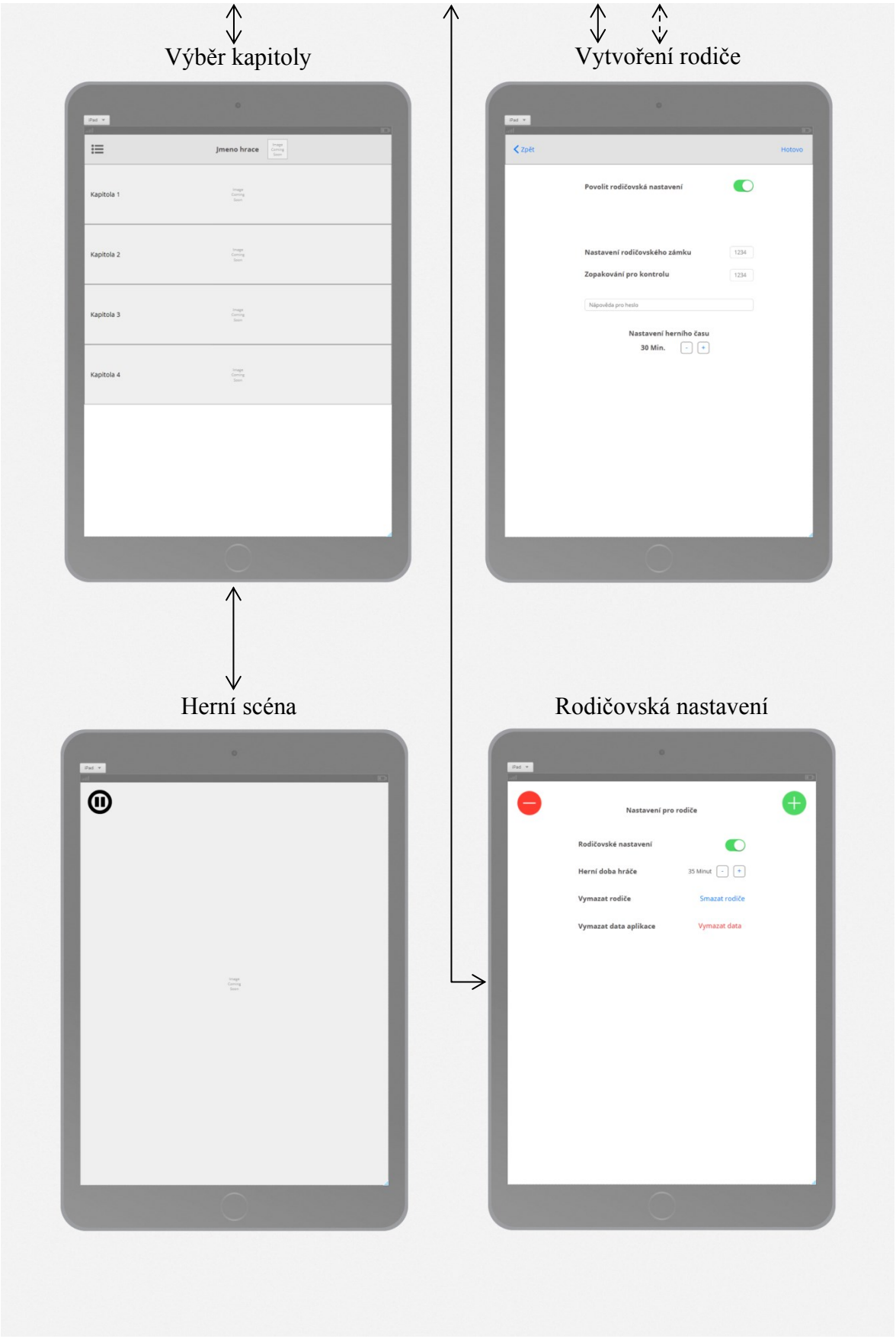
- kompatibilita se zařízeními iPad;
- rozšiřitelnost o další herní kapitoly;
- zpracování a uložení dat aplikace do úložiště zařízení;
- jednoduché a intuitivní uživatelské prostředí;
- rychlá odezva aplikace;
- spolehlivost – předcházení pádům aplikace;
- robustnost – zachování dat uživatele při náhlém ukončení aplikace.

3.2 Wireframe aplikace

Následující část této práce popisuje mobilní aplikaci reprezentovanou pomocí wireframe návrhů vytvořených v programu *Indigo Studio*. Tento návrh slouží jako předloha k samotnému vypracování včetně šipek znázorňující pohyb po jednotlivých obrazovkách. Šipky jsou dvojího druhu, přerušované šipky naznačují postup obrazovkami při prvním spuštění aplikace, kde je uživatel proveden vytvořením prvního hráče a rodiče a po dokončení je přenesen na obrazovku s výběrem hráče. Druhá šipka, nepřerušovaná, znázorňuje druhé a všechna další spuštění aplikace, kde je uživateli rovnou zobrazen výběr hráče. Z výběru hráče může vytvořit nového hráče, stejně tak vytvořit rodiče, pokud nebyl vytvořen. Jestli byl rodič vytvořen, může se přihlásit do rodičovského nastavení. Po výběru hráče se uživatel přenese na seznam kapitol a po kliknutí na kapitolu je načtena herní scéna, která se mu následně zobrazí a může začít hrát.



Obrázek 16 – Wireframe aplikace – první část



Obrázek 17 – Wireframe aplikace – druhá část

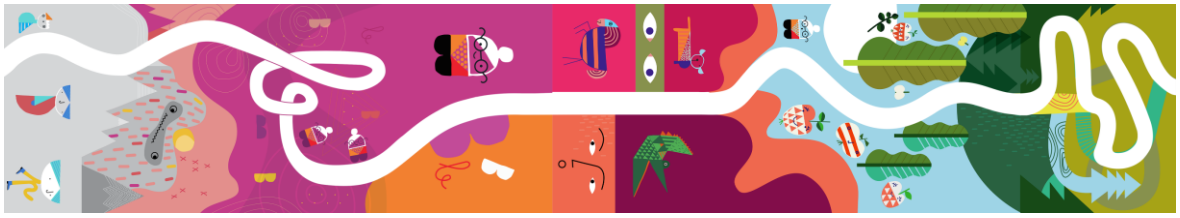
3.3 Grafické zpracování aplikace

Grafické zpracování aplikace se zakládá na diplomové práci Mgr. Alžběty Suré na téma „Mobilní aplikace pro děti se specifickou poruchou učení“, kterou vypracovávala k zakončení studií na Fakultě multimediálních komunikací. V její práci je teoreticky pojednáno o poruchách učení a jsou vytvořeny grafické podklady pro aplikaci, která by mohla dětem pomáhat. Po kontaktování Alžběty Suré byly získány grafické podklady a jsou v aplikaci plně využity. Ostatní grafické materiály, jako jsou ikony a profilové obrázky jsem vytvářel sám za pomocí programu *GIMP*.

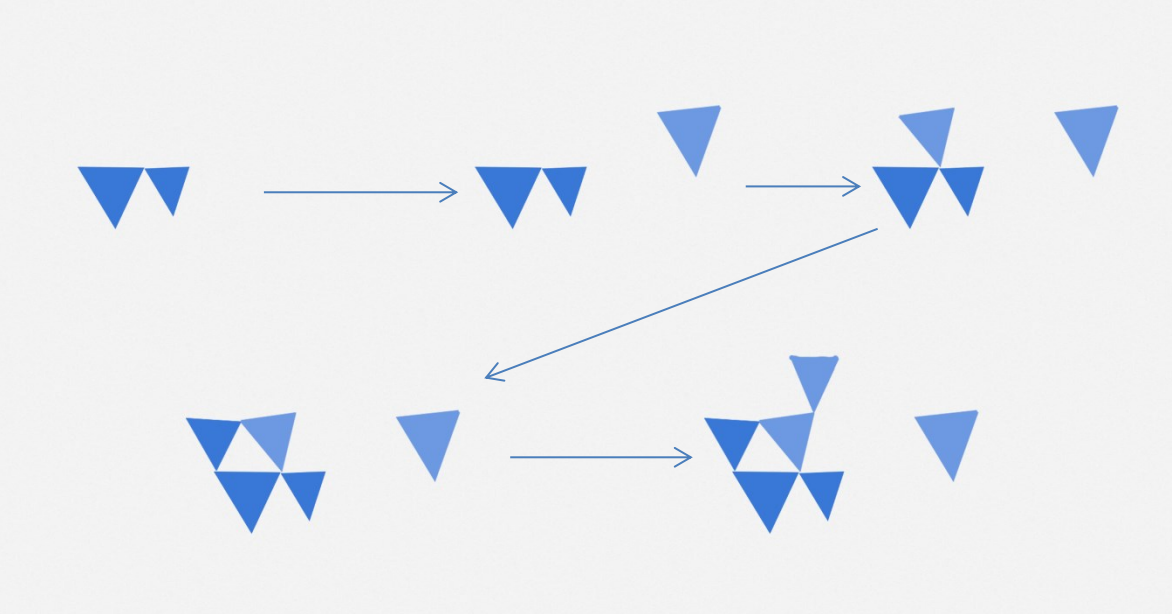
3.3.1 Grafika herního plánu

Herní plán je základem pro každou kapitolu hry. V současnosti existuje jen jedna kapitola s jedním herním plánem. Herní plán je jeden dlouhý obrázek s cestičkou doplněný o kreslené předměty a postavičky vyplňující volný prostor okolo cestičky. Předmětům a postavičkám jsou přidány animace, tak aby vypadaly, že se v průběhu hraní pohybují. Herní plán i příklad animací je ukázán níže. Jen herní plán musel být kvůli své velikosti rozdělen na dvě části a otočen o 90 stupňů doprava.

Obrázek 18 – Herní plán, 1. část



Obrázek 19 – Herní plán, 2. část

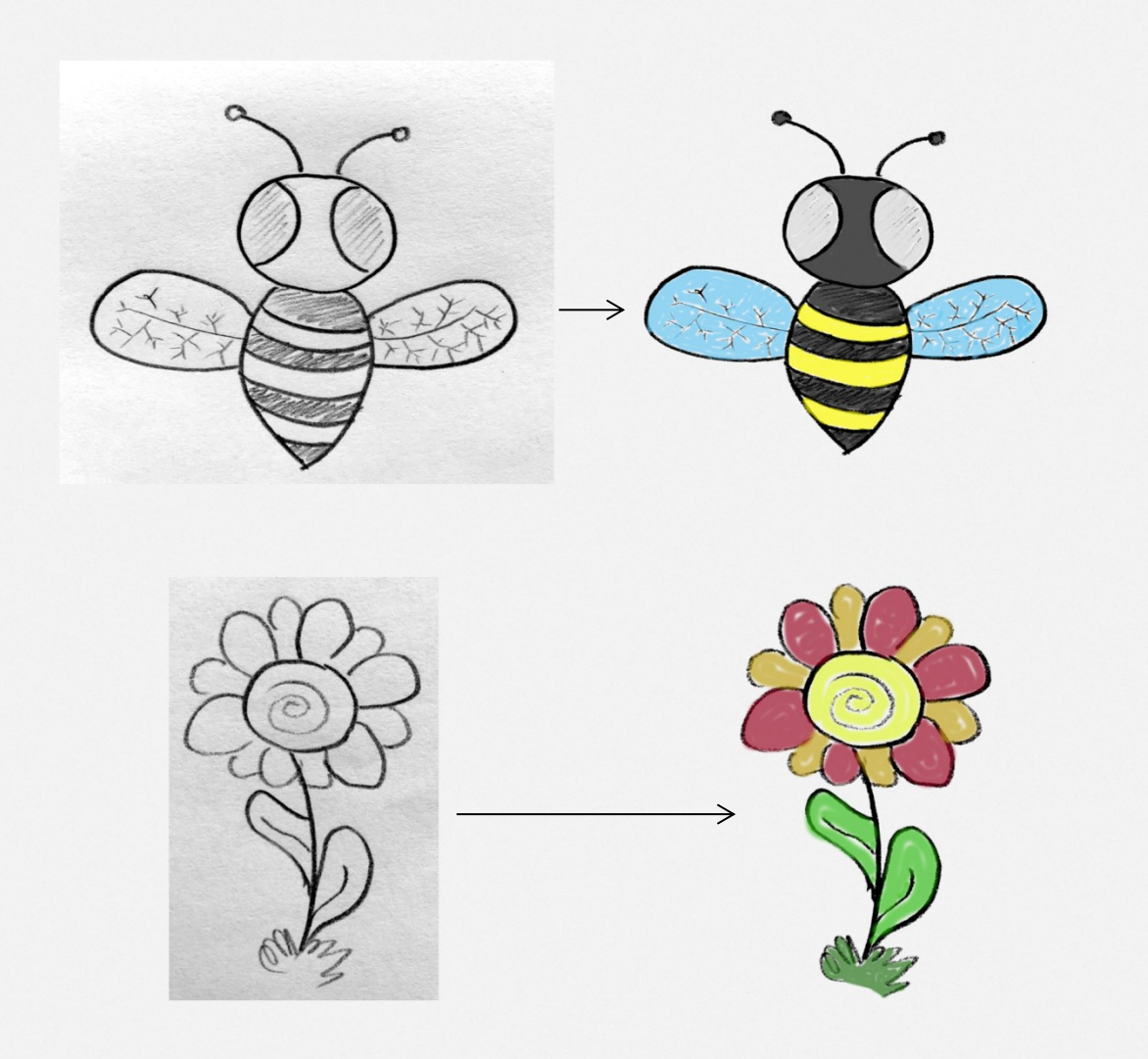


Obrázek 20 – Ukázka animace z herního plánu

3.3.2 Grafika uživatelského prostředí

Prostředí aplikace bylo doplněno o vlastní ikony, které jsem vytvořil v programu *GIMP*. Byla snaha dosáhnout co možná nejpřesnějšího vyjádření, co která ikona provede za akci po jejím spuštění, protože se předpokládá, že děti nebudou umět číst.

Další obrázky, které obohatily aplikaci, jsou profilové obrázky, které si hráč vybírá při vytváření uživatelského účtu. Chtěl jsem vytvořit obrázky, které by byly pro děti atraktivní a vypadaly, jakoby je někdo doopravdy namaloval. Nejprve jsem si tužkou namaloval na čistý poznámkový blok několik obrázků a vybral devět nejpovedenějších. Následně je převedl do elektronické podoby pomocí fotoaparátu svého telefonu a znovu aplikoval svou znalost programu *GIMP*. Po sladění kontrastu a jasu obrázku jsem získal čistou bílou a jasnou černou barvu. Po získání čisté bílé jsem ji převedl do průhlednosti a zbytek vymaloval stylem, který připomíná omalovánky. Výsledek páru profilových obrázků je znázorněn na obrázku 21.



Obrázek 21 – Ukázka tvorby profilových obrázků

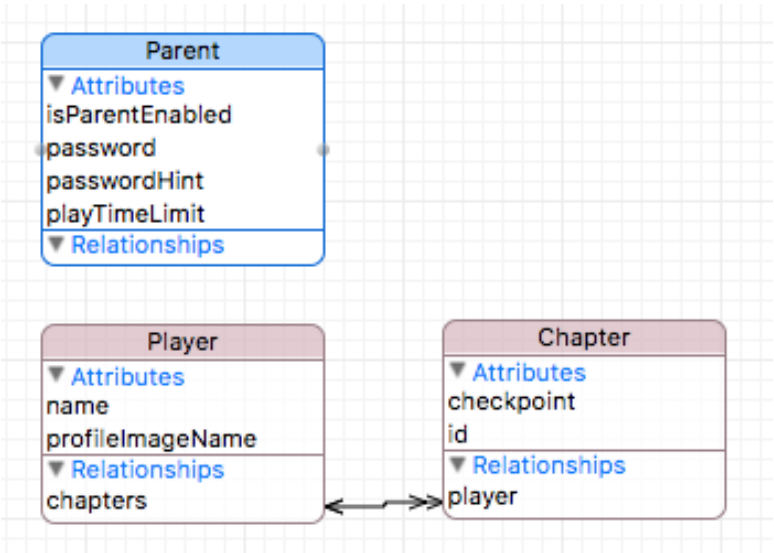
3.4 Databáze uživatelů

Jelikož aplikace nevyužívá klasických databázových návrhů, nýbrž frameworku CoreData, jsou jednotlivé záznamy databáze reprezentovány pomocí entit, kde každá entita má vlastní atributy. Celá databáze uživatelů, a potažmo i celé aplikace, se skládá ze tří následujících entit:

- **Parent** – entita pro rodiče. Rodič v aplikaci existuje pouze jednou, proto v databázi bude uchovávána pouze jedna entita tohoto objektu. Entita *Parent* má následující atributy:
  - **isParentEnabled** – binární hodnota udávající, jestli jsou rodičovská nastavení povolena či nikoliv;

- **password** – celočíselná hodnota uchovávající čtyřmístný kód, pro přihlášení do nastavení rodiče;
- **passwordHint** – uchovává nápovědu k heslu;
- **playTimeLimit** – celočíselná hodnota představující dobu minutách, po jakou může hrát hráč do upozornění.

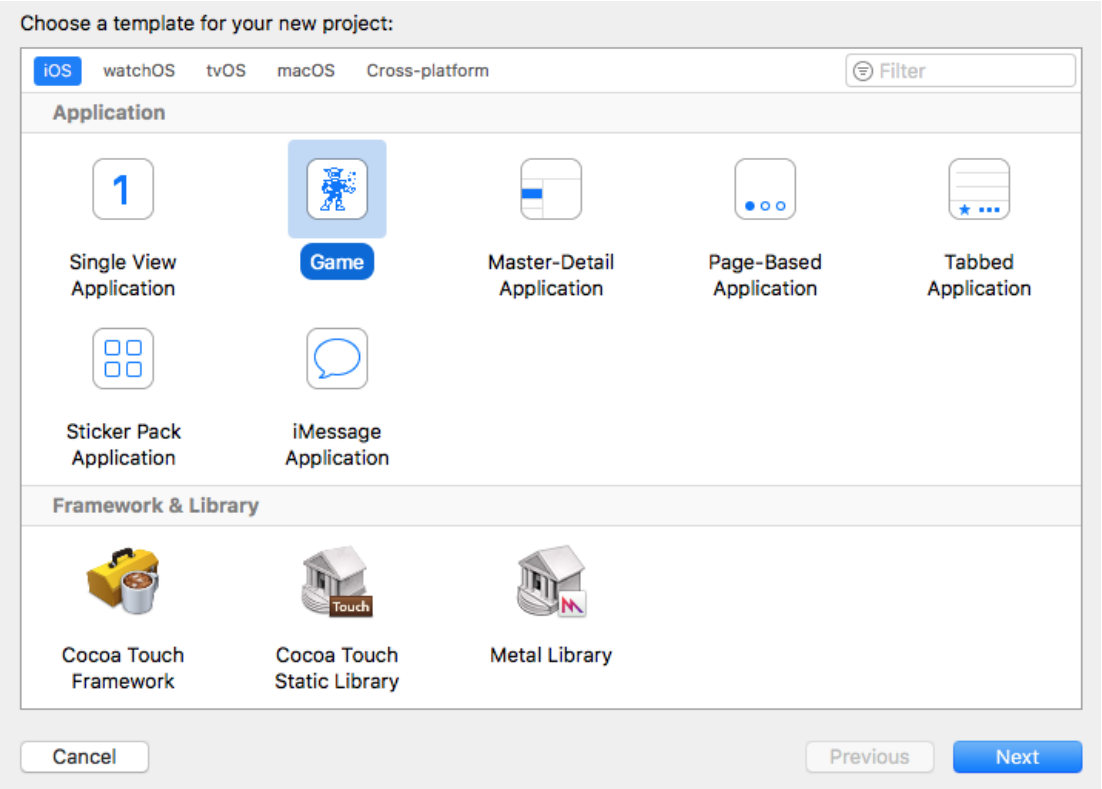
- **User** – entita pro hráče. Hráčů může být v aplikaci neomezené množství, proto může v databázi existovat neomezené množství entit reprezentující tyto hráče. Každý hráč má níže uvedené atributy:
  - **name** – atribut uchovávající jméno hráče;
  - **profileImageName** – představuje název profilového obrázku, který si uživatel vybral při vytváření.
- **Chapter** – *Chapter* je entita v databázi představující jednu kapitolu. Tato entita je svázána k entitě *User* – hráči vztahem 1 ku N, kde jeden hráč může mít přidružených několik entit *Chapter* v závislosti na tom, kolik kapitol již hrál. Entita *Chapter* uchovává dva atributy:
  - **id** – představuje ID kapitoly číslované od 1;
  - **checkpoint** – představuje ID posledního záchytného bodu odehraného hráčem, ke kterému je entita *Chapter* vázána.



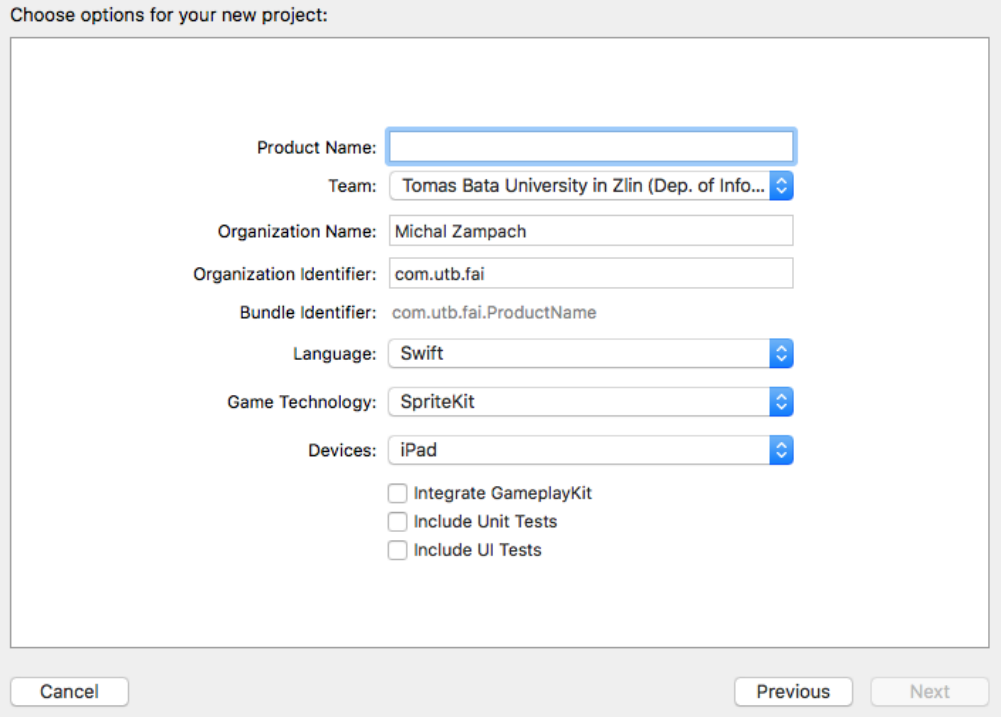
Obrázek 22 – grafické znázornění databáze CoreData

4 VYPRACOVÁNÍ MOBILNÍ APLIKACE

Vypracování aplikace začalo založením projektu v programu XCode. Během vytváření byla zvolena možnost, že je v plánu vývoj herní aplikace s použitím frameworku SpriteKit. Vývojové prostředí XCode vygenerovalo základní projekt obsahující jeden Storyboard a jeden kontrolér. Bohužel nebyla během vytváření projektu k dispozici volba na využití frameworku CoreData pro ukládání dat do paměti zařízení. Proto byl vytvořen nový projekt, obyčejná aplikace, v níž je během vytváření možné si zvolit využití frameworku CoreData. To vedlo k vygenerování potřebné funkce v projektu pro práci s frameworkem. Tyto funkce byly zkopírovány do prvního projektu a následně vytvořen model databáze CoreData, který v projektu též chyběl. Další, co následovalo v přípravě projektu, bylo promazání nepotřebných funkcí, které byly vygenerovány k frameworku SpriteKit. Šlo pouze o ukázkové funkce, které v této aplikaci nebyly použity, a tudíž byly zbytečné. Dalším krokem bylo vytvoření obrazovek, kontrolérů a View podle wireframe návrhů. Toto vytváření je popsáno v následující části diplomové práce.



Obrázek 23 – Zvolení typu projektu

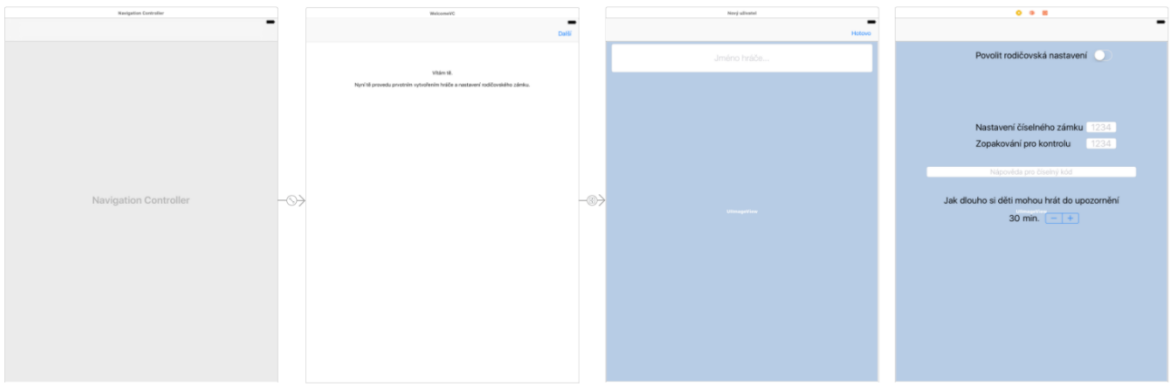


Obrázek 24 – Výběr technologie SpriteKit do projektu aplikace

4.1 Vytvoření Storyboard návrhů a používaných obrazovek

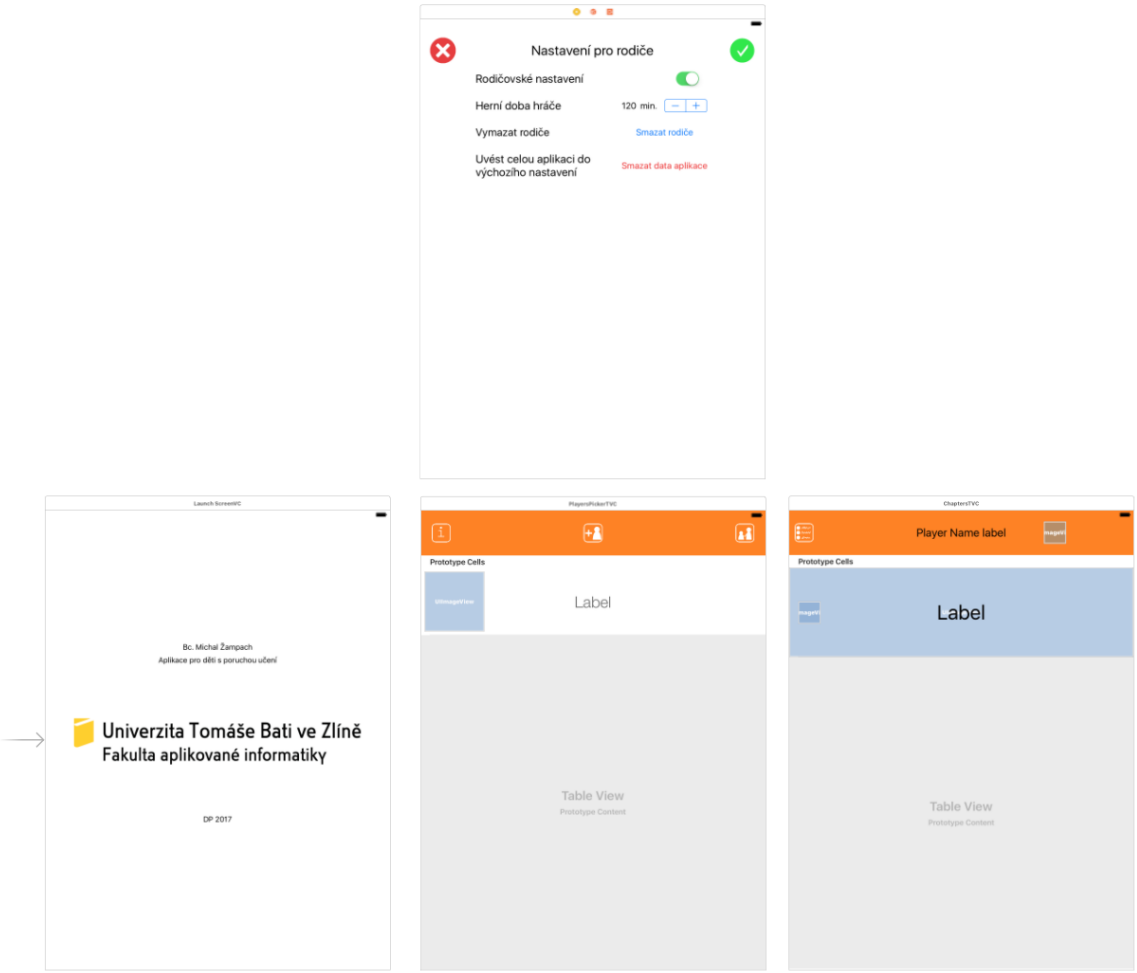
Jelikož se obrazovek v aplikaci vyskytuje větší množství, bylo vytvořeno více vizuálních návrhů Storyboard. Některé obrazovky jsou zobrazeny pouze při prvním spuštění, některé nemusejí být při běžném použití vůbec zobrazeny. Výsledkem jsou tři Storyboard návrhy a každý z nich uchovává jiný set obrazovek – View.

První Storyboard obsahuje uvítací obrazovku, prvotní vytvoření hráče a prvotní vytvoření rodiče. To vše zasazené do *navigation* kontroléru, který se stará o přecházení mezi těmito úvodními obrazovkami, dokud není dokončeno úvodní nastavení. Obrázek kontroléru je vidět na obrázku 25 níže.



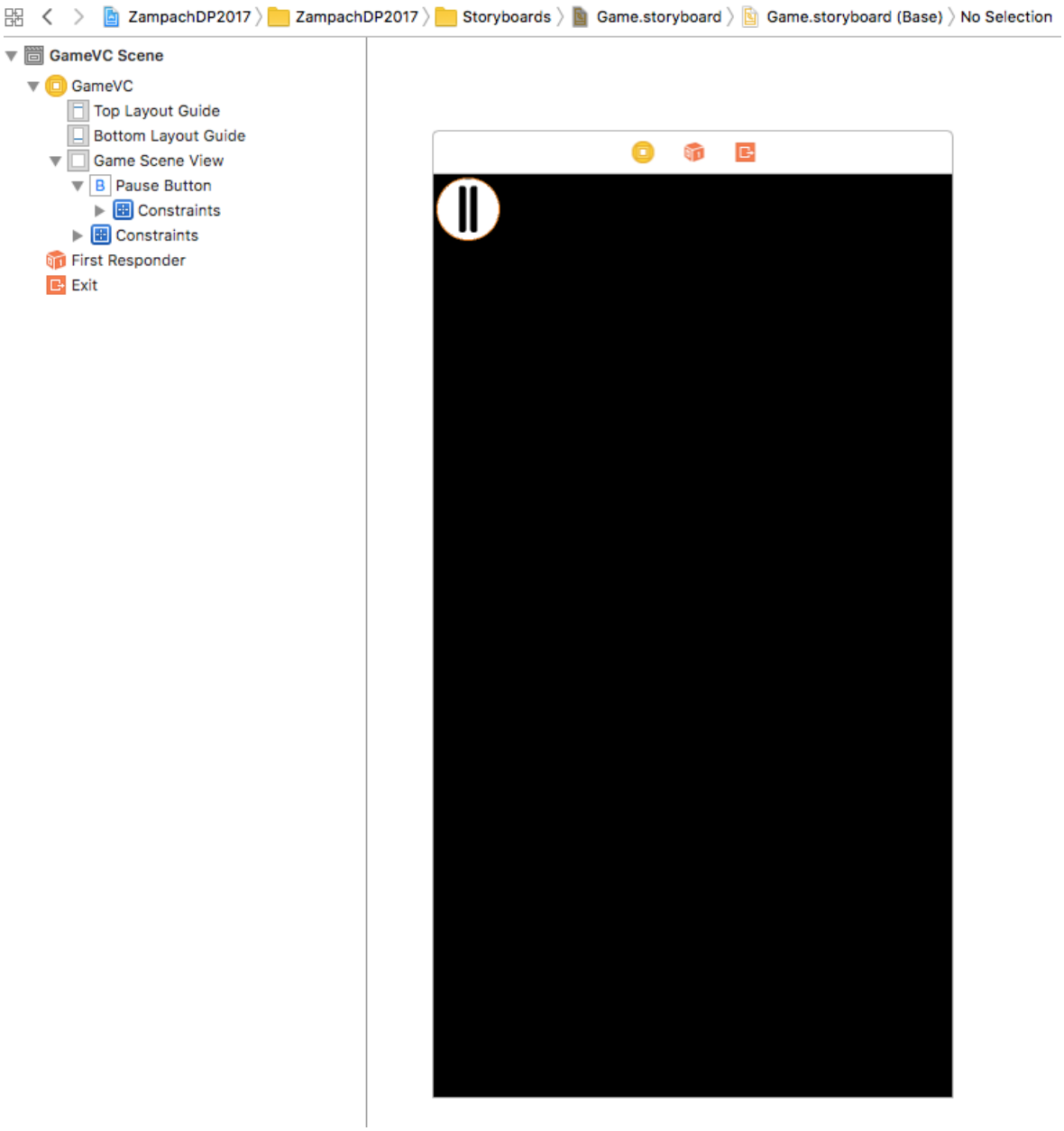
Obrázek 25 – Úvodní Storyboard

Následující Storyboard, který byl vytvořen, obsahuje takzvané *InitialView*, což je View, které se zobrazuje vždy při spuštění aplikace. Jedná se o vstupní obrazovku, stejně jako na wireframe návrhu. Kromě této jsou zde umístěny i obrazovky s výběrem hráče, výběrem kapitoly a rodičovským nastavením. Tento Storyboard je vytvořen tak, aby obsahoval nejdůležitější obrazovky a nemuselo se vytvářet nic navíc. Hráč tak zapne aplikaci, vybere svůj profil, vybere kapitolu a může hrát, to vše v rámci obrazovek v jednom Storyboard návrhu. Jeho zobrazení je vidět na obrázku 26 níže.



Obrázek 26 - Hlavní Storyboard aplikace

Třetím, a posledním Storyboard návrhem, je Storyboard obsahující jednu obrazovku. Tato obrazovka pojímá samotnou herní scénu. Záměrem tohoto návrhu je možnost budoucího rozšíření o další herní prvky, kde by bylo potřeba přidat například další scénu. Tento Storyboard oddělí herní část od zbytku aplikace, čímž udrží její přehlednost.



Obrázek 27 – Storyboard s herní scénou

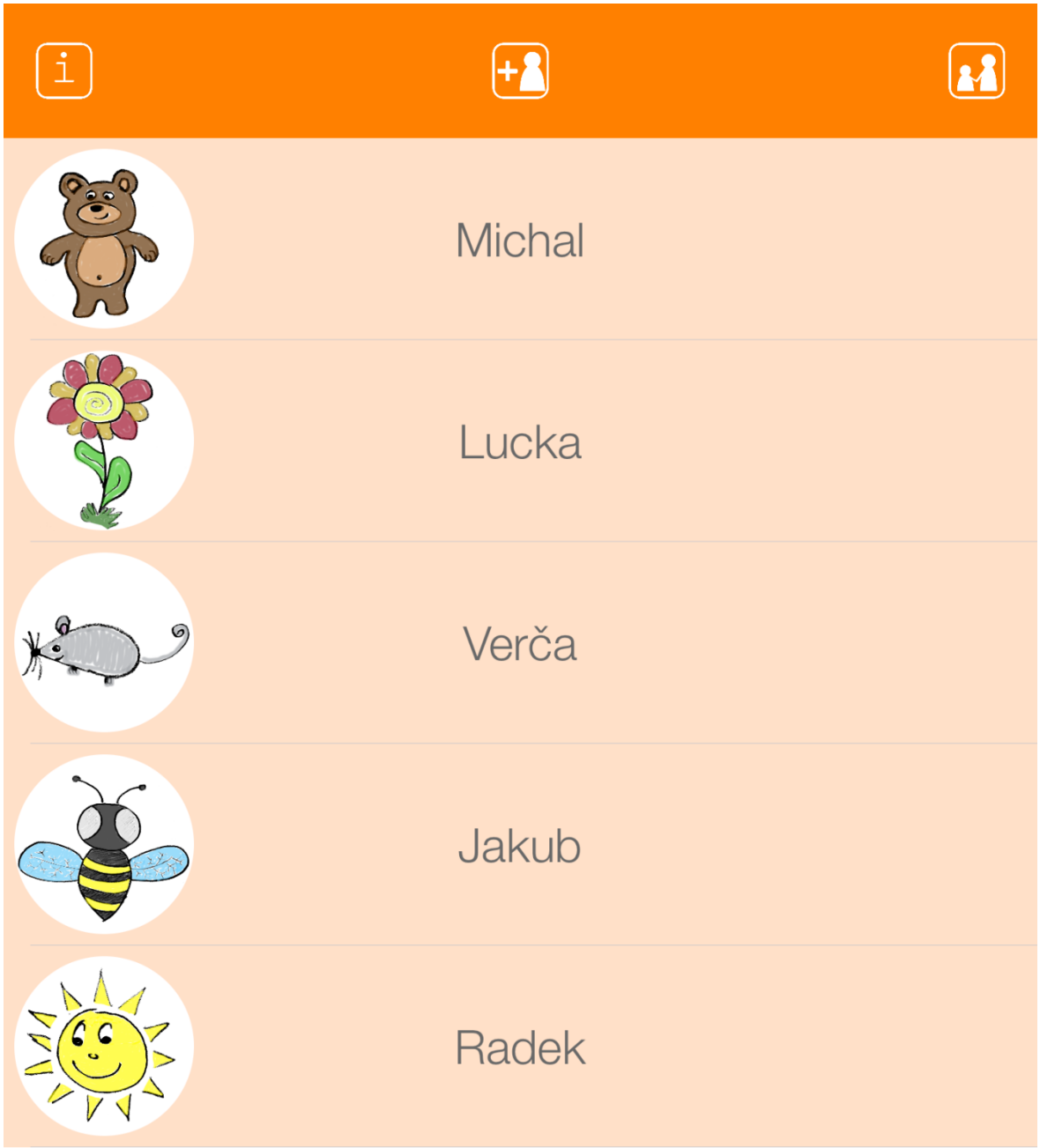
4.2 Vytvoření kontrolérů k jednotlivým View

Každá obrazovka v aplikaci má vlastní kontrolér. Kontrolér je třída obsahující programovou logiku k dané obrazovce. Bez kontroléru je samotná obrazovka víceméně jen statická. Její chování, je tak potřeba naprogramovat v kontroléru, který je obrazovce přiřazen. Níže jsou vypsány a vysvětleny vytvořené kontroléry použité v aplikaci:

- ***LaunchScreenVC*** – *ViewController*, který se stará o úvodní obrazovku. Úlohou tohoto kontroléru je zjistit, jestli je aplikace spuštěna poprvé či není. Rozlišení probíhá na základě uložené hodnoty pomocí *UserDefaults*. Jestliže je aplikace

spuštěna poprvé, je automaticky zobrazena další obrazovka – úvodní obrazovka. V opačném případě přejde aplikace k obrazovce s výběrem hráčů. Automatický přechod k další obrazovce má nastavenou časovou prodlevu, aby si uživatel stihl obrazovku prohlédnout a přečíst text na ní umístěný.

- ***WelcomeVC*** – *ViewController*, který se stará o obrazovku zobrazenou při prvotním spuštění aplikace. Její účel je pouze uvítací a nejsou na ní prováděny žádné operace.
- ***NewPlayerVC*** – *ViewController* starající se o vytvoření hráče. Při vytváření obrazovky kontrolér načítá profilové obrázky a umisťuje je do rovnoměrné mřížky, dle naprogramovaného algoritmu, aby byly připraveny pro hráče k výběru. Důvod, proč byl vytvořen algoritmus na načítání obrázků, a nebylo využito možnosti Storyboard návrhu, je ten, že do aplikace můžeme přidat více profilových obrázků a algoritmu pouze zadáme počet obrázků na jeden řádek. O vše ostatní, jako jsou velikosti a souřadnice umístění, se již postará sám a není potřeba nic měnit ve Storyboard návrhu. Po zobrazení obrazovky je úkolem kontroléru obstarat zdárné zadání uživatelského jména a výběr profilového obrázku, případně upozornit na nedostatky. Po úspěšném dokončení je objekt nového hráče uložen do databáze a načten v přehledu hráčů.
- ***NewParentVC*** – *ViewController* s podobným záměrem jako *NewPlayerVC*. Stará se o vytvoření rodiče. V kontroléru je posuzováno, zdali je vytvoření rodiče vyžadováno uživatelem na základě přepínače. Pokud ano, opět je ověřováno vyplnění všech potřebných informací, po jejichž zdárném vyplnění je rodič vytvořen a uložen do databáze.
- ***PlayersPickerTVC*** – *TableViewController* jehož primárním úkolem je obsluha seznamu hráčů umístěných v jeho obrazovce. Před samotným zobrazením obrazovky kontrolér načítá z databáze objekty hráčů, které pak seznamu poskytne a uživatel je tak uvidí a může si zvolit právě ten svůj. Mimo seznam hráčů jsou v horní části obrazovky ještě tři tlačítka. Tlačítko pro zobrazení informací o aplikaci, tlačítko pro přidání hráče a tlačítko pro přístup do rodičovského nastavení.



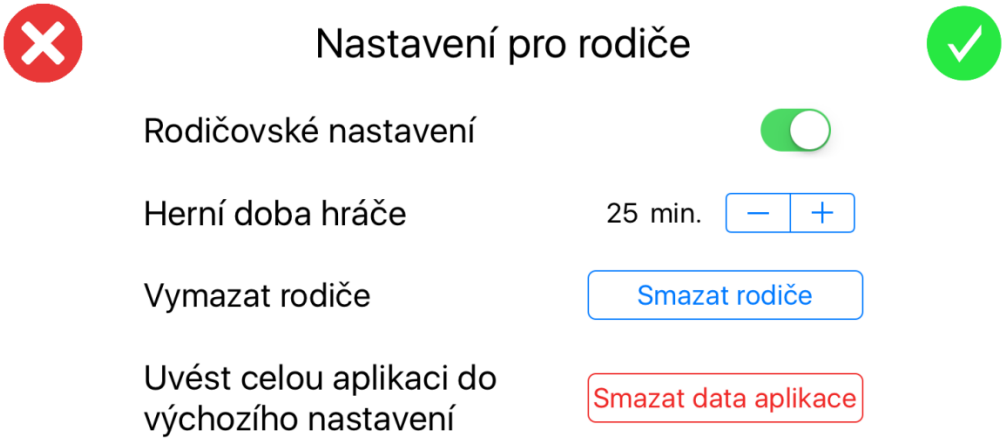
Obrázek 28 – Snímek obrazovky s výběrem hráčů

- ***ChaptersTVC*** – další *TableViewController* starající se o seznam na jeho obrazovce. Tentokrát seznam kapitol, na které se hráč dostane po výběru svého profilu. Tento kontrolér načítá druhy dostupných kapitol a předkládá je na výběr hráči.

Mimo seznam je na obrazovce ještě tlačítko pro navrácení zpět k výběru hráčů. Po výběru kapitoly je hráč přenesen do herní scény.

Obrázek 29 – Oříznutý snímek obrazovky s výběrem kapitoly

- ***ParentVC*** – *ViewController* na jehož obrazovku se uživatel – rodič dostane po zdárném zadání přístupového kódu z přehledu hráčů. Má zde možnosti nastavení, které jsou po jakékoliv změně uloženy do databáze.



Obrázek 30 – Oříznutý snímek obrazovky rodičovského nastavení

- ***GameVC*** – *ViewController* do kterého se hráč dostane po výběru kapitoly. Kontrolér vytvoří třídu konkrétní kapitoly na základě informace říkající, kterou kapitolu si uživatel zvolil. Následně vytvoří objekt třídy *GameScene*, což je vlastní třída dědící od třídy *SKScene* určená k vykreslení herní scény na obrazovku. Její popis je obsažen v kapitole 4.3. Této scéně je předán vytvořený objekt kapitoly a číslo záchytného bodu, na kterém uživatel skončil, jestliže už danou kapitolu někdy předtím hrál. Kromě vytváření objektu kapitoly a scény má kontrolér za úkol obstarávat všechny funkce, které nejsou přímo spojeny s fungováním herní části. Stará se tak o ukládání informací o posledním dosaženém záchytném bodu do databáze. Též o zobrazení hlášek po dokončení herního plánu a hlášky oznamující konec herního času. Také obstarává pozastavení hry v momentě, kdy hráč zmáčkne tlačítko pauzy v levém horním rohu obrazovky. Důležitou úlohou je rovněž odebrání všech vytvořených objektů z dočasné paměti RAM v zařízení. Především vytvořené scény, která si pro své fungování zabírá hodně místa a nemůže být v paměti uchovávána, pokud není zobrazena.

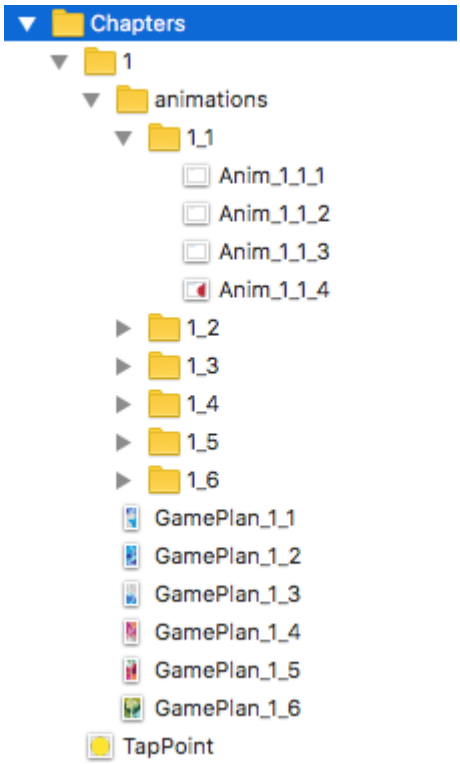
4.3 Vytvoření kapitol

Kapitoly jsou jedinečné svým obsahem, který předkládají uživateli do herní scény. Z toho důvodu byla vytvořena třída, která obsahuje všechny potřebné funkce a názvy proměnných. Po boku této hlavní třídy budou vytvářeny třídy dědící její proměnné a funkce. Taková další třída bude představovat již opravdovou kapitolu, protože bude naplněna potřebnými daty a informacemi jako jsou rozměrové parametry herního plánu, atlasy animací se souřadnicemi pro umístění do herního plánu a v neposlední řadě také data křivek, po kterých uživatel pojede svým prstem. Takto vytvořené kapitoly budou ve svém základu všechny stejné a herní scéna je bude moci zpracovávat stejným způsobem.

4.4 Vytvoření animací

K vytvoření animací v herním plánu je použito sérií jednotlivých obrázků, jako je tomu na obrázku číslo 14 a 20. Obrázky jsou vloženy do katalogu určeného na obrázky přímo v projektu aplikace, tzv. *Asset Catalog*. Obrázky jsou pak postupně vloženy do objektu *SKTextureAtlas*. Tento atlas si uchovává každá třída kapitoly zvlášť. Každá kapitola má totiž vlastní herní plán s vlastními animacemi. Kromě atlasů si kapitoly

uchovávají souřadnice pozic, kam mají být umístěny animace na herním plánu a pozice, na které když uživatel najede prstem, tak se animace spustí.



Obrázek 31 – Asset Catalog s obrázky ke kapitole č. 1

4.5 Vytvoření herní scény

Třída *GameScene*, dědící od třídy *SKScene*, je třída starající se o herní scénu. Scéna má na starost vše, co se má zobrazit a přehrát uživateli na displeji. Přitom musí dosahovat stálé plynulosti pro co nejlepší uživatelský zážitek během hraní. Právě *GameScene* je nejkomplexnějším fragmentem celé aplikace a během vývoje byl několikrát přepracován. Scéna si při svém vytváření přebírá informace o kapitole, kterou má vykreslit, informace o posledním záchytném bodu, na kterém uživatel skončil, a toto vše připraví a vykreslí hráči tak, aby mohl začít hrát. Nicméně, ještě před samotnou přípravou zobrazovaných prvků je vypočítán koeficient udávající, o jak moc je zobrazovaný herní plán větší, než je obrazovka zařízení. To je spočítáno jako poměr aktuální šířky displeje zařízení a šířky herního plánu. Tímto koeficientem jsou pak vynásobeny všechny zobrazované prvky. Herní plán i všechny ostatní grafické materiály jsou vytvářeny s rozměry pro současně největší iPad zařízení, které má rozlišení displeje 2733 pixelů na výšku a 2048 pixelů na šířku. Tím, že se vynásobí rozměry zobrazovaných objektů koeficientem, se zmenší na ideální velikost potřebnou pro zobrazení na konkrétním zařízení.

První, co scéna vytváří je objekt typu *SKNode*. Tento Node má jedno, nicméně o to důležitější poslání uchovávat všechny ostatní prvky vytvořené pomocí SpriteKit tříd. Důvod, proč vytvářet takový Node je ten, že ve hře dochází k pohybu herního plánu na základě pohybu prstu uživatele – v návaznosti na tento fakt se musí pohybovat i všechny ostatní objekty umístěné na scéně. Tím, že tyto prvky budou umístěny do objektu Node, stačí pohybovat pouze s Node objektem, čímž dojde k pohybu i všeho ostatního v něm obsaženého.

Dalším krokem je tak vytvořit zbývající objekty nutné k vytvoření celé kapitoly a umístit je do hlavního Node objektu. Prvním objektem přicházejícím na řadu je herní plán na pozadí. Během vývoje bylo zjištěno, že třída SKSpriteNode zodpovědná za vykreslování grafických objektů nezvládá vykreslovat obrázky větší než 4096 pixelů na šířku nebo na výšku. Herní plán první kapitoly má sice na šířku 2048 pixelů, ale na výšku přes 23000 pixelů. Herní plán tak musel být v grafickém editoru rozdělen na menší části. O přidání všech částí plánu do hlavního Node se stará algoritmus, který pracuje se všemi částmi herního plánu a postupně je umístí nad sebe v přesném pořadí. Tímto dojde k vytvoření jednolitého herního plánu a v plné délce.

Dalším na scéně vytvořeným objektem je posuvný bod, kterého se uživatel musí dotknout předtím, než může táhnout po cestičce. Tento bod má nastavenou animaci pomalého blikání. Když se ho uživatel dotkne, zmizí. Tento mechanizmus zajistí, aby hráč obtáhnul celou cestičku od začátku do konce, protože bod se znovu objeví tam, kde uživatel pustil prst z obrazovky nebo tam, kde vyjel z cestičky. Bod se objeví se svou animací a čeká, dokud se jej uživatel znova nedotkne, aby mohl v cestě pokračovat. Bod je umisťován buďto na začátek herního plánu nebo na některý z předešle dosažených záchytných bodů.

S umístěním posuvného bodu na své poslední místo je též spojeno posunutí herního plánu na ono místo, aby byl bod viditelný a plán nebyl posunutý někde „mimo“. O toto počáteční posunutí se stará jedna z funkcí posouvající hlavní Node po scéně.

Posledním prvkem, který chybí na herní scéně, je cestička, kterou hráč obtahuje. Původní myšlenka byla, že během obtahování cestičky prstem, bude zjišťována barva na pozici, kde se zrovna nachází prst hráče. Pokud by zjištěná barva odpovídala barvě cestičky, aplikace by věděla, že je hráč na cestičce, a ne mimo ni. Tento koncept byl zrealizován, nicméně byl i zavrhnut z důvodu vysoké paměťové náročnosti. Ověřování,

zda je uživatel svým prstem na cestičce, či nikoliv, je založeno na využití Bézierovy křivky. Respektive hned několika křivek, které dohromady tvoří celou cestičku skrze herní plán. Bézierova křivka je křivka, která má souřadnice udávající její konec a počátek. Kromě těchto souřadnic má i dva řídící body vázáné na její počáteční a koncový bod. Správnou změnou polohy řídících bodů lze dosáhnout požadovaného zakřivení křivky. Pro vytvoření všech křivek jsem znovu využil svých zkušeností s grafickým editorem *GIMP,* ve kterém je možné Bézierovy křivky vytvářet. Otevřel jsem si obrázek herního plánu, vytvářel jednu křivku za druhou tak, aby přesně odpovídaly herní cestičce v plánu. Souřadnice všech jednotlivých křivek jsem přenášel do aplikace. V aplikaci byl pak vytvořen algoritmus pro zpracování souřadnic všech křivek. Programovací jazyk Swift disponuje objektem určeným přímo pro Bézierovy křivky – *UIBezierPath.* Tomuto objektu lze předat hned několik Bézierových křivek, ze kterých vytvoří jednu souvislou. Tato souvislá křivka je poté vykreslena se zadanou tloušťkou a barvou. Ve skutečnosti jsou křivky vytvořeny hned dvakrát. Jedna křivka slouží pro naznačení cesty, kudy má hráč táhnout prstem. Druhá není viditelná, ale má tloušťku nastavenou tak, aby pokryla celou šířku cestičky viditelné na herním plánu. Právě s touto cestičkou je kontrolováno, zdali se hráč svým prstem drží cestičky, či nikoliv. Tento koncept se osvědčil a přidal jednu výhodu navíc. S menšími úseky cestiček lze dosáhnout správné navigace při psaní písmen. Příkladem je hned první písmeno *d* na herním plánu. První cestička dovede hráče do středu písmena, druhá cestička ho donutí nakreslit „bříško“ a nedovolí mu prstem jet nikam jinam. Pro lepší představu je písmeno *d* s cestičkami znázorněno na obrázku 33.

Po dokončení vytváření všech počátečních objektů potřebných ke spuštění hry je scéna přenesena na obrazovku zařízení a připravena až uživatel začne hrát.

4.6 Zpracování interakcí od uživatele

Během hraní jsou vyhodnocovány pohyby prstem po obrazovce. K tomu slouží funkce, které scéna obsahuje již ve výchozím stavu. Jde o funkce *touchesBegan*, *touchesMoved*, *touchesEnded* a *touchesCancelled*.

První funkce *touchesBegan* je provedena vždy, když se uživatel dotkne prstem nebo prsty obrazovky. Tato funkce vyhodnocuje, jestli některý z prstů uchopil posuvný bod. Pokud ano, scéna si tento dotyk zapamatuje pro další zpracování a posuvný bod animací vymizí.

Další funkcí je *touchesMoved*. Ta je provedena vždy, když se některý z prstů po obrazovce pohne. V ní je zjišťováno, jestli prst, který se pohnul po obrazovce je ten, který uchopil posuvný bod. Jestliže ano, je volána funkce na posun hlavního Node, ve které se vypočítává, jakým směrem a o jakou vzdálenost se prst posunul. Kromě toho se zjišťuje, jestli se uživatel neposunul na pozici záchytného bodu neboli konec křivky. V momentě, kdy se na takový bod dostane, je vytvořena nová křivka a stará křivka odebrána. Pakliže, když už není žádná další křivka k dispozici, scéna pozná, že hráč dojel prstem na konec herního plánu a předá tuto informaci svému kontroléru.

Během posouvání prstu po cestičce je také prováděna funkce pro zjišťování, jestli nemá být přehrána některá z animací. Animace se přehrávají podle toho, na jakých souřadnicích se uživatel zrovna vyskytuje svým prstem. Tato funkce však není přímo nutná pro fungování samotné hry, proto je její běh přesunut do vedlejšího výpočetního vlákna aplikace. Hlavní vlákno aplikace je určeno primárně pro zpracování všech podnětů od uživatele a pro vykreslování herní scény. Proto, když funkce zjistí, že má být vykreslena nějaká animace, je tato animace provedena zpět v hlavním vlákně.

Funkce *touchesEnded* je provedena v momentě, kdy se uživatel přestane dotýkat prstem obrazovky. Znova je testováno, jestli prst opouštějící displej je ten, který držel posuvný bod. V případě shody je posuvný bod nastaven na poslední pozici uživatelova prstu a spuštěna animace pro jeho zobrazení. Též je volána funkce pro přizpůsobení herního plánu. Jestliže uživatel skončil u horního okraje obrazovky, je hlavní Node posunut směrem dolů, tak aby uživatel znova viděl cestičku, která ho čeká a mohl pokračovat ve hraní. Obdobné je to i v případě, kdy uživatel skončí u dolního okraje displeje. V tomto případě je hlavní Node naopak posunut směrem nahoru, dokud se poloha posuvného bodu nenachází uprostřed výšky displeje.

Poslední funkce *touchesCancelled* je volána, když je dotek uživatele zrušen systémovým podnětem a provádí všechny procesy jako funkce *touchesEnded*, kromě funkce na přizpůsobení herního plánu.

4.7 Zpracování uživatelských účtů

V aplikaci jsou dva druhy účtů, účet hráče a účet rodiče. Účet hráče je více aktivním prvkem v celé aplikaci. Účet rodiče je spíše myšlen jako zámek nastavení, kde přistupujeme k možnostem, jež ovlivňují samotnou aplikaci. Vytvoření rodiče rozhodně

není podmínkou pro fungování aplikace, naopak vytvoření hráče či hráčů je. Chce-li si hráč zahrát, musí si nejdříve vytvořit svůj účet, napsat jméno a vybrat profilový obrázek. Po zdárném vytvoření je mu vytvořen profil, který uvidí na obrazovce s přehledem hráčů. Během vytváření je také vytvořen záznam v databázi a také otevřena první kapitola, kterou může uživatel hrát. K přehledu kapitol se uživatel dostane při kliknutí na svůj profil. Zde pak uvidí své kapitoly, které má již odemčené a ve kterých může pokračovat nebo hrát od začátku.

4.8 Ukládání dat pomocí CoreData

Pro práci s daty byl při vytváření projektu přidán Framework CoreData. Pomocí CoreData lze ukládat data aplikace do úložiště zařízení a po spuštění opět načíst.

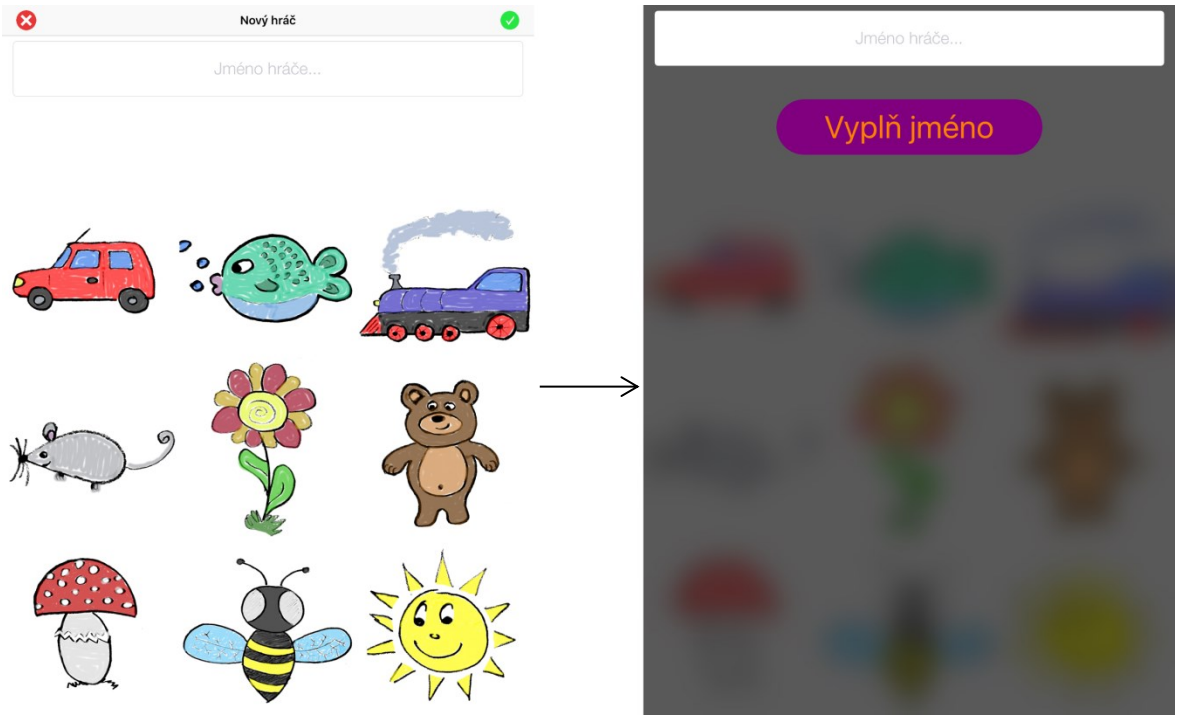
První, co bylo potřeba vytvořit, byl databázový model. V tomto modelu jsou navrženy entity, které se budou v databázi vyskytovat. Tyto entity je potřeba navrhnout přesně tak, jak s nimi bude pracováno, tj. určit přesně jejich název, vlastnosti a vztahy k ostatním entitám. V aplikaci existují tři entity: *Parent, User* a *Chapter*. Entity jsou blíže popsány v kapitole 3.4. Po vytvoření všech entit nám model nabízí vytvořit třídy, které budou reprezentovat entity databáze jako objekty. To velice usnadňuje práci s daty při všech operacích jako načítání, ukládání, úpravách a mazání. Přístup k objektům, potažmo entitám, je snazší a bezpečnější, než se snažit z databáze složitě získávat jednotlivé záznamy.

Následně byla vytvořena třída speciálně určená pro práci s daty, Tato třída obsahuje funkce pro načtení hráčů, kapitol a rodiče. Stejně tak i pro uložení nového hráče, nového rodiče nebo nového záznamu kapitoly pro konkrétního hráče. Kromě načítání a ukládání dat má třída za úkol data i mazat a upravovat. V aplikaci je možnost smazat konkrétního uživatele, smazat samotného rodiče z rodičovského nastavení, nebo rovnou veškerá data aplikace.

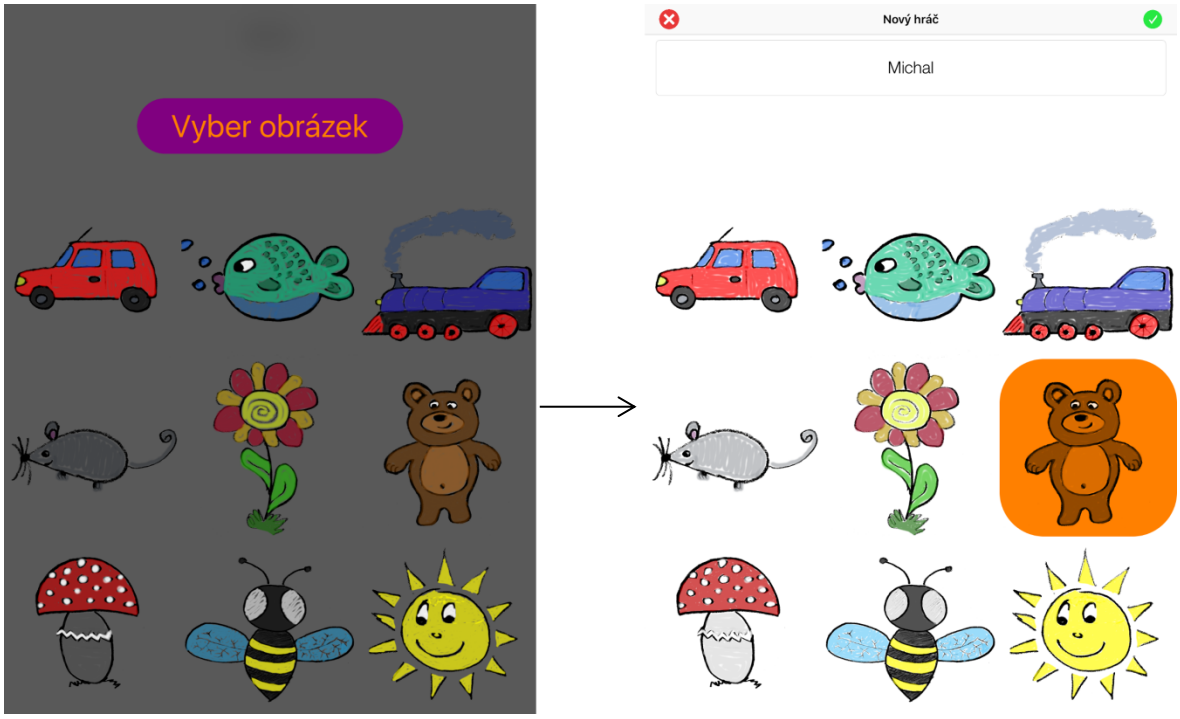
Funkce pracují na principu, kdy od aplikace je vyžádán tzv. kontext databáze. Tento kontext je upraven dle aktuálně plánovaných změn v databázi a po dokončení změn je na aplikaci zavolána funkce uložení kontextu. Aplikace si vezme pozměněný kontext a provede změny v databázi.

5 TESTOVÁNÍ MOBILNÍ APLIKACE

Testování mobilní aplikace probíhalo tak, jak se vyvíjela samotná aplikace. Což si lze představit jako dvě etapy. V první etapě bylo vyvíjeno uživatelské prostředí, kam spadá vytvoření hráče, vytvoření rodiče, přehled hráčů, seznam kapitol, rodičovská nastavení. Zde bylo zapotřebí testovat nejen správné přechody mezi kontroléry, ale hlavně uživatelské vstupy, které jsou klíčové. První, na co uživatel narazí při prvotním spuštění aplikace, je úvodní obrazovka a úvodní nastavení hráče. Právě zde je zapotřebí zajistit, aby si hráč napsal své jméno a vybral jeden z nabízených profilových obrázků. Na další obrazovce je posléze nastavení rodiče, kde si rodič, mimo jiného, zadá svůj číselný kód k přístupu do nastavení. Též se musí zajistit, aby zadávaný kód byl zopakovaný správně, a i ostatní položky nezůstaly bez povšimnutí. Většinou tyto náležitosti byly testovány v simulátoru a po dokončení jednotlivých větších celků pak v reálném zařízení iPad pro ujištění, že vše funguje, jak má a vše je korektně vykresleno. Grafický projev, co aplikace po uživateli žádá, byl brán, jako důležitý prvek. Proto i při pokusu o vytvoření uživatele bez vyplněného jména nebo nevybraného obrázku aplikace jasně znázorní, co má uživatel udělat, než může pokračovat dále. Na funkčnost takových znázornění byla při testování brána zvýšená pozornost, viz obrázky 32 a 33



Obrázek 32 – Zvýraznění nevyplněného jména

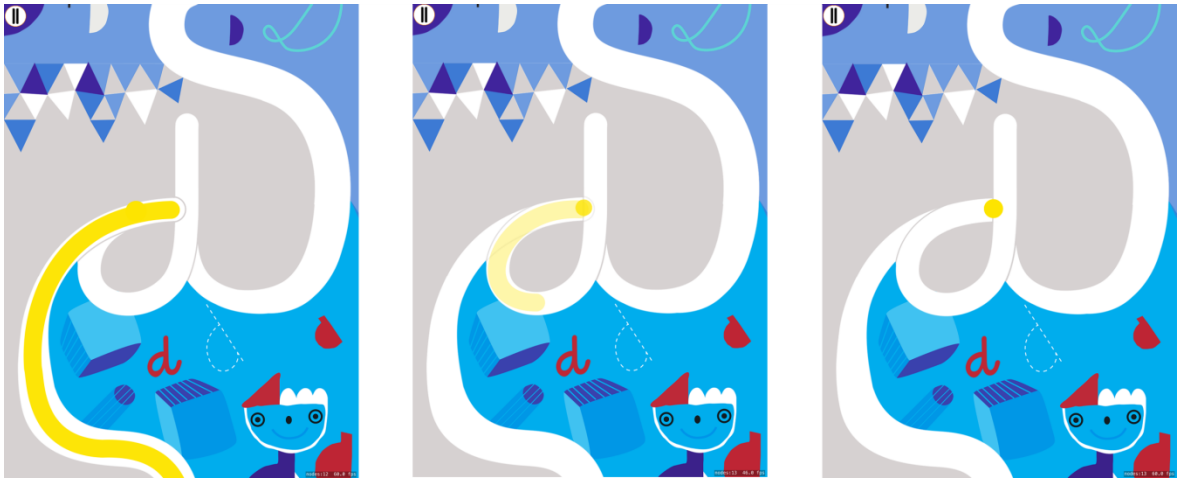


Obrázek 33 – Zvýraznění výběru obrázku

V rámci první etapy byla také vytvářena databáze, která se uchovává v úložišti zařízení. Muselo se ověřit, zda veškerá ukládaná data jsou ukládána správně se všemi vzájemnými vztahy a stejně tak se muselo ověřit korektní načtení po opětovném spuštění aplikace. To platilo i pro mazání dat z databáze. V této části vývoje bylo vše testováno v simulátoru se zdárným výsledkem bez zádrhelů.

Druhá etapa představuje již samotnou herní část, kde by měl hráč, v ideálním případě, trávit nejvíce času a z převážné většiny byla testována na reálném zařízení. Herní část obsahuje herní scénu, která mimo jiné vyhodnocuje interakce hráče pomocí doteků obrazovky, a na jejichž základě se mění obsah scény. Zde bylo testování o něco komplikovanější kvůli matematickým výpočtům ovlivňující pohyb všeho, co je na scéně viditelné. První k otestování bylo přesné umístění herního plánu a křivek znázorňujících aktivní část cesty, kterou má hráč za úkol projet prstem. Postupně se přidávaly další části aktivně vyznačené cesty. Následujícím krokem bylo umístění animací, které musí být též umístěny přesně na správných souřadnicích. Dalším, co se muselo u animací testovat, byly spouštěcí body animace. Animace nejsou v aplikaci spouštěny zároveň nebo nahodile, nýbrž na základě lokace prstu uživatele. Animace tak mají své poziční spouštěče. Nejdůležitějšími jsou ty, které spouští animace u cesty tvořící písmeno. Uživatel, jenž dojede prstem například do písmena *d* po první křivce, musí ve druhé křivce obkreslit bříško samotného písmena *d*. Právě při přechodu z jedné křivky do druhé musí být

přehrána animace otevírající cestičku dále po křivce, jako je tomu na obrázku 34. Postup testování spouštění animací, tak byl dán opakovaným hraním a nastavováním co největší přesnosti, protože křivky, po kterých uživatelé jezdí svým prstem, mají nastavenou určitou toleranci a mohlo by se tak stát, že uživatel animaci nikdy nespustí, kdyby neměla správně umístěný svůj spouštěč.



Obrázek 34 – Současná změna křivky a animace otevírající cestu

Druhá etapa pokračovala testováním posunování obsahu herní scény při pohybu prstem po herním plánu. Důležité bylo ověřit, že scéna posouvá herní plán správným směrem, o správnou vzdálenost, a že herní plán neposune za jeho hranice, kde nic není. Právě ošetření toho, že se plán neposune za své hranice, bylo důležité. Výpočet posunu byl během testování několikrát přepočítán.

V poslední části druhé etapy probíhalo testování herního plánu jako celku se zaměřením na projíždění cestičky prstem. Zde bylo doceněno poznatků testerů, kterým byl svěřen iPad s nainstalovanou hrou. Bylo zjištěno, že co člověk, to jiný způsob projíždění plánem. Někdo jede prstem pomalu a přesně, někdo naopak jede rychle, spěchá a čáru neobtahuje zcela dokonale. Na základě zpětné vazby jsem přizpůsoboval toleranční hodnoty jak pro samotné křivky, tak pro záchytné body, kde se jedna křivka přenáší na další. Tím bylo dosaženo toho, že i hráčům, kteří projíždí cestičku rychleji a méně přesně, se pohybový bod zastavuje jen v případech, kdy vyjedou úplně mimo hranice cesty. Na základě této tolerance se pak poupravily i toleranční hodnoty pro záchytné body, aby byly spuštěny v momentě, kdy hráč dojede na konec zrovna aktivní části cestičky.

5.1 Možnosti dalšího rozšíření

Během samotného vývoje a testování aplikace, přišlo nemalé množství myšlenek k vylepšení a rozšíření aplikace. V následujících bodech tyto možnosti rozšíření rozepsány:

- **Obsahová část** – obsahová část je zásadním prvkem. Aktuální verze obsahuje jedinou kapitolu s jedním herním plánem, který obsahuje písmeno *b* a *d* k procvičení. Rozšíření kapitol o další je přímo žádoucí. S dalšími kapitolami by přibylo i počet procvičovaných písmen a nebylo by od věci, kdyby se písmena vyskytovala i opakovaně. Kapitoly by se pak mohly řadit abecedně, a hráč by tak měl možnost si procvičit právě ta, která mu dělají problémy.
- **Mód nekonečného hraní** – rozšíření obsahové části o nové plány by se dalo skombinovat s novým herním módem, kde by hráč byl uzavřen v nekonečné smyčce herních plánů. Jak by hráč pokračoval po herním plánu svým prstem, byly by náhodně vybírány další herní plány, po kterých by pokračoval. Ve výsledku by se jednalo o jeden dlouhý, nikdy nekončící herní plán. Zajímavé by určitě bylo i zakomponovat algoritmus, který by upřednostňoval herní plány s písmeny, jenž si uživatel potřebuje nebo chce procvičit. Zde by byla nutná podmínka upravit herní plány tak, aby na sebe vzájemně navazovaly.
- **Obohacení grafické části** – samotné uživatelské prostředí je nyní tvořeno klasickými kontroléry s výchozími View, které jsou běžně vývojářům k dispozici. Většina uživatelského prostředí je obarveno kombinací oranžové a bílé barvy. Aplikace by byla hezčí, kdyby místo jednobarevných ploch obsahovala obrázky tematicky shodné s grafikou na herním plánu.
- **Přidání animací** – V herním plánu se nacházejí animace, jejichž přehrávání spouští uživatel po dosažení určité lokace svým prstem. Jsou momenty, kdy při nečinnosti uživatele se nepřehrává žádná z animací a na celé scéně bliká pouze bod, čekající na další dotyk uživatele. V tyto momenty se scéna jeví, jako velmi statická. Přidání několik náhodných animací, by statickou scénu oživilo (letící brouček, padající list, stékající kapka apod.).
- **Zvýšení zábavnosti** – stávající herní plány jsou ve své podstatě neměnné obrázky s cestičkou, kterou hráč musí projet a dokončit až do jejího cíle. Po určitě době si hráč již cestičky zapamatuje, okouká a stanou se pro něj méně zábavnými. Právě zábavnost by se mohla zvýšit přidáním malých podúkolů, které by hráč plnil

na cestě herním plánem. V určitých bodech daného herního plánu by se hra pozastavila a spustila se herní sekvence podúkolu. Podúkoly by mohly být tipu „najdi babičce brýle“ a hráč by musel vzít brýle a přetáhnout je na obrázek babičky. Další úkoly by mohly spočívat v najití dvou stejných obrázků, vyber největší hvězdičku; klikni, kde je schovaný pejsek a další. Typů podúkolů by bylo potřeba vymyslet větší množství, aby se neopakovaly příliš často.

- **Zvuky** – současná verze aplikace je bez zvuků. K pěknému vizuálnímu projevu patří i líbivý zvukový projev, jako je podmanivá hudba na pozadí, zvuková odezva po výběru aktivního prvku, zvuky přehrávaných animací a další. Zvukového obsahu by nakonec bylo potřeba udělat stejně tolik, jako je obsahu grafického. Pro každý plán by bylo vhodné zvolit jinou hudbu na pozadí, a i zvuky animací by byly pokaždé jiné.
- **Hlasová odezva aplikace** – kromě samotných zvuků by bylo vhodné aplikaci doplnit i o hlasovou odezvu aplikace. Jelikož je aplikace směřována i na předškoláky, není tak pravděpodobné, že by uměli číst. Aplikace obsahuje minimum textu a co nejvíce obrázků. Nicméně kdyby aplikace na hráče mluvila pomocí hlasových nahrávek, vdechlo by to do celého hraní nový rozměr. Aplikace by tak říkala hráči, co na aktuální obrazovce může nebo má udělat. Mohla by i oznamovat, kdy vyjel mimo cestičku na herním plánu a další. Pokud by bylo zamýšleno aplikaci lokalizovat i do angličtiny, bylo by vhodné se domluvit nejlépe s rodilým mluvčím o poskytnutí hlasových materiálů. To samé platí i u českého jazyku. Hlas by měl být spíše ženský, příjemný se srozumitelnou výslovností.
- **Použiti TouchID** – naopak funkčním rozšířením by mohlo být využití snímače otisku prstů nazvané TouchID. Mnoho zařízení iPad dnes TouchID disponuje a vývojáři jej mohou ve svých aplikacích využít. Rodič by si tak při vytváření rodičovského účtu nejen zadal svůj číselný kód, ale zároveň by mohl povolit, že místo zadávání číselného kódu chce použít svůj otisk prstu a číselný zámek by byl použit v případě, kdy nebude otisk prstu rozpoznán. Například při pokusu dítěte pokračovat dále ve hře po vypršení času.
- **Záloha na iCloud** – společnost Apple poskytuje svým uživatelům cloudové úložiště zvané iCloud na které si kromě svých souborů mohou ukládat i data aplikací. Tato záloha dat by se mohla využít i v této aplikaci pro uložení databáze

uživatelů a jejich postupů ve hře. Při opětovném nainstalování aplikace, by se data nechala stáhnout z úložiště a hráči by tak nepřišli o svůj herní postup.

- **Využití fotoaparátu** – dalším funkčním rozšířením je použití fotoaparátu při výběru profilového obrázku. Všechna zařízení iPad obsahují fotoaparát, jak na přední, tak i na zadní straně zařízení. V případě, kdy by se hráči nelíbil žádný z profilových obrázků, mohl by si vyfotit vlastní nebo například sám sebe a získal tak jedinečný profilový obrázek.

ZÁVĚR

Cílem této diplomové práce bylo navrhnout, vytvořit a následně otestovat mobilní aplikaci pomáhající dětem se specifickou poruchou učení, konkrétně dysgrafií.

V teoretické části se čtenář může dočíst o tom co je to mobilní aplikace, jaké mobilní platformy dnes existují pro fungování mobilních aplikací a blíže je rozepsána platforma iOS včetně způsobu, jakým se mobilní aplikace pro tuto platformu vyvíjejí. Tuto část následuje úsek diplomové práce charakterizující strukturu mobilní iOS aplikace. Jsou v ní popsány komponenty použité při tvorbě samotné aplikace od jednotlivých zobrazovacích prvků až po způsoby ukládání obsahu.

Praktická část se zabývá návrhem mobilní aplikace. Byly stanoveny funkční a nefunkční požadavky a vytvořeny wireframe návrhy pro bližší představu, jak bude aplikace ve výsledku vypadat. Stejně tak byla navržena databáze, se kterou aplikace pracuje a uchovává v ní svá data. Kromě návrhů je v praktické části pojednáno o samotném grafickém zpracování aplikace. Důležitou pasáží praktické části je část o vypracování samotné aplikace. Aplikace byla vytvářena ve vývojovém prostředí XCode s využitím programovacího jazyku Swift a herního frameworku SpriteKit. Vývoj se dělí na jednotlivá stádia od vytvoření jednotlivých obrazovek po naprogramování samotné programové logiky a vytvoření herní části aplikace. Právě v herní části se děti budou učit zvládat psaní jednotlivých písmen pomocí herních plánů. Herní plán představuje jeden dlouhý obrázek s cestičkou mající různé tvary včetně tvarů písmen. Onu cestičku budou děti opisovat prstem a nepřímo se tak učit technikám psaní.

Po vytvoření aplikace, ale částečně i během vývoje, byla aplikace testována. Testování probíhalo jak v simulátoru na počítači, tak především v samotném zařízení iPad, pro které je tato aplikace navržena. Testování se zúčastnilo několik dobrovolníků z kruhu rodiny. Bylo zjištěno, že každý hráč má ke hře jiný přístup a na základě poznatků byla hra doladěna, aby vyhovovala co nejlépe všem.

V samotném závěru práce je nakonec pojednáno o dalších možnostech rozšíření aplikace, kterých není málo. Více herních plánů, více grafických materiálů a zvukového doprovodu jsou hlavní náměty, kam by se aplikace měla dále ubírat.

Během vypracovávání jsem si prohloubil své znalosti s vývojem iOS aplikací. Osvojil jsem si i práci s herním frameworkem SpriteKit a přišel na kloub novým funkcím

programovacího jazyku Swift. Sám mám z aplikace pro děti radost a věřím, že se s pokračováním jejího vývoje může stát i velice užitečnou.

SEZNAM POUŽITÉ LITERATURY

[1] NEUBURG, Matt. IOS 10 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781491970027.

[2] MANNING, Jonathon a Paris BUTTFIELD-ADDISON. IOS Swift Game Development Cookbook, 2nd Edition: Simple Solutions for Game Development Problems. 2nd edition. Newton, Massachusetts, USA: O'Reilly Media, 2015. ISBN 9781491920800.

[3] NEUBURG, Matt. Programming iOS 10: Dive Deep into Views, View Controllers, and Frameworks. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781491970119.

[4] KNOTT, Matthew. Beginning Xcode: Swift 3 Edition. Newton, Massachusetts, USA: O'Reilly Media, 2016. ISBN 9781430250050.

[5] The Swift Programming Language: Swift Programming Series [online]. 2014. Apple [cit. 2017-01-23]. Dostupné z: https://itunes.apple.com/cz/book/swift-programming-language/id881256329?mt=11

[6] GOODWILL, James a Wesley MATLOCK. Beginning Swift Games Development for iOS: Updated for Swift 3. 2nd edition. New York City, USA: Apress, 2017. ISBN 9781484223093.

[7] Mobile/Tablet Operating System Market Share. In: *Netmarketshare* [online]. [cit. 2017-04-19]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1

[8] AppStore Support. In: *Developer Apple* [online]. 2017 [cit. 2017-04-20]. Dostupné z: https://developer.apple.com/support/app-store/

[9] Choosing a Membership. *Apple Developer* [online]. 2017 [cit. 2017-04-26]. Dostupné z: https://developer.apple.com/support/compare-memberships/

[10] GOODWILL, James. *Beginning Swift games development for iOS*. ISBN 9781484204016.

[11] *Tiobe: TIOBE Index for April 2017* [online]. [cit. 2017-04-26]. Dostupné z: https://www.tiobe.com/tiobe-index/

[12] *Apple Developer: API Reference* [online]. [cit. 2017-04-27]. Dostupné z: https://developer.apple.com/reference

[13] *Raywenderlich: Apple Game Frameworks* [online]. [cit. 2017-04-30]. Dostupné z: https://www.raywenderlich.com/category/apple-game-frameworks

[14] *Apple Developer: Swift* [online]. [cit. 2017-04-30]. Dostupné z: https://developer.apple.com/swift/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
SDK	Software Development Kit
UI	User Interface
WWDC	Worldwide Developer Conference

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

P1 CD s diplomovou prací a soubory obsahující zdrojové kódy