VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PROGRAM PRO VÝUKU CIZÍCH JAZYKŮ PRO MOBILNÍ ZAŘÍZENÍ

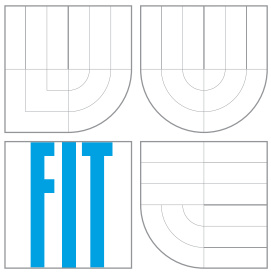
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID JANČA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PROGRAM PRO VÝUKU CIZÍCH JAZYKŮ PRO MOBILNÍ ZAŘÍZENÍ

TEACHING OF FOREIGN LANGUAGES ON MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID JANČA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2015

Abstrakt

Bakalářská práce se zabývá vývojem výukové aplikace cizích jazyků pro mobilní zařízení, která vyučuje fráze a slova pomocí účinného adaptivního algoritmu. Výukový algoritmus se přizpůsobuje uživateli využitím informací z průběhu učení za účelem maximalizace efektivity výuky. V práci jsou představeny výukové algoritmy a programování v XCode pro iOS. Tvorba výukové aplikace je složena z mnoha disciplín, a proto se práce zabývá i tématy jako teorie výuky, výběr cílové mobilní platformy, přehled populárních výukových aplikací a uživatelské testování.

Abstract

This thesis deals with the development of foreign language teaching applications for mobile devices. The applications teach phrases and words using effective adaptive algorithms. Educational algorithms adapt to the user by using information from the course of learning in order to maximize teaching effectiveness. The thesis presents educational algorithms and programming in XCode for iOS. The creation of an educational application is composed of several disciplines and so the thesis deals with topics such as the theory of teaching, choice of target mobile platforms, an overview of popular educational applications and user testing.

Klíčová slova

mobilní, aplikace, iOS, Android, adaptivní výuka, cizí jazyk

Keywords

mobile, application, iOS, Android, adaptive learning, foreign language

Citace

David Janča: Program pro výuku cizích jazyků
pro mobilní zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2015

Program pro výuku cizích jazyků pro mobilní zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Janča
20. května 2015

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce Jaroslavu Rozmanovi za jeho cenné rady ohledně řešení technických problémů i ohledně psaní textu bakalářské práce.

Obsah

Úvod

Mobilní aplikace získaly v posledním desetiletí významný podíl na trhu s aplikacemi. Uživatelé vítají možnost využívat aplikaci kdy a kde potřebují.

Využití informačních technologií ve výuce má velký potenciál. Počítače sice zatím nedokáží uspokojivě zastupovat roli učitele, který by aktivně vedl studenta, odpovídal na otázky. Na procvičování látky, které často zabere více času, jsou však počítače plně využitelné. Přitom role učitele není v některých oblastech nezbytná, jako při procvičování cizích slov.

Zatímco lidstvo pokročilo s výzkumy paměti, výukové aplikace na trhu tento pokrok často neodrážejí. Nabídka aplikací pro opakování slovní zásoby je na systémech *Android* a *iOS* velmi rozsáhlá. Naprostá většina těchto aplikací však postrádá efektivní výukový algoritmus.

Cílem této práce je navrhnout a implementovat výukovou aplikaci na procvičování cizího jazyka s efektivním přizpůsobivým algoritmem, který by umožnil rychlejší učení, či delší udržení naučené látky v paměti.

Teoretická část práce se věnuje teorii učení v kontextu výukové aplikace, psychologickým jevům, díky kterým jsou dnešní výukové aplikace tak účinné, dále průzkumu současné situace na poli mobilních výukových aplikací a výběru cílové platformy. Teorie je zakončena přehledem výukových algoritmů. Praktická část práce dokumentuje návrh, implementaci a testování výukové aplikace pro *iOS*.

Kapitola 1

Teorie výuky

V této kapitole si představíme samotnou základní teorii výuky a paměti z hlediska výukové aplikace spolu s vybranými psychologickými jevy. Ze zjištěných informací se pokusíme vyvodit závěry pro vývoj vlastní výukové aplikace. Na konci kapitoly je zmíněna vybraná metoda pro výuku frází a slovíček, metoda karet a jsou představeny způsoby, jakými využívá nebo může využívat výše zmíněných psychologických jevů pro zvýšení efektivity výuky.

Učení [21] je psychický proces sloužící k přizpůsobování organismu k prostředí. Lidský organismus se historicky vyvíjel v mnohem méně komplexním prostředí. Proto se s rychlostí technologického pokroku zvětšuje i propast mezi tím, na co je organismus člověka od narození připraven a s čím se musí vyrovnat v průběhu života. Tím se zvyšuje význam učení člověka, které je získáváním předpokladů pro aktivní vyrovnávání se životním prostředím. Znamená získávání zkušeností ve formě vědomostí, dovedností a návyků. Významně jeho efektivitu zvyšuje povzbuzování kladným hodnocením a plánovité procvičování. Výchozí bází této aktivity je paměť.

1.1 Paměť

Paměť je jednou z nejdůležitějších vlastností živých organismů. Je součástí všech rovin duševního dění. Definujeme ji jako vlastnost a soubor procesů, které umožňují osvojení informací, jejich uchování a vybavení.

Forma

Z hlediska formy se dělí na procedurální a deklarativní. Procedurální je vývojově starší, z velké části neuvědomělá. Zahrnuje podmiňování a senzomotorické dovednosti. Deklarativní paměť [20] zajišťuje vědomé vybavování zkušeností a osvojování vědomostí. Četným opakováním lze deklarativní paměťovou stopu přeměnit v procedurální.

Trvání

Z hlediska trvání se paměť dělí na krátkodobou, pracovní a dlouhodobou. Krátkodobá paměť je velmi malá a pojme pouze několik sekund nebo 7+-2 položky. Její rozsah postačuje ke splnění mnoha dílčích úkolů. Pokud po splnění úkolu už tyto informace nepotřebujeme, dojde poměrně rychle k jejich zapomenutí. Pracovní paměť je delší než krátkodobá, ale kratší než dlouhodobá. Trvání pracovní paměti je možno vyměřit tím, jak dlouho řešíme problém, k jehož zvládnutí jsou příslušné informace potřeba. Dlouhodobá paměť slouží k

zapamatování toho, co budeme potřebovat po delší dobu. Její trvání může být až celý život. Významným faktorem pro délku uchování je frekvence opakování.

1.1.1 Paměť z hlediska výukové aplikace

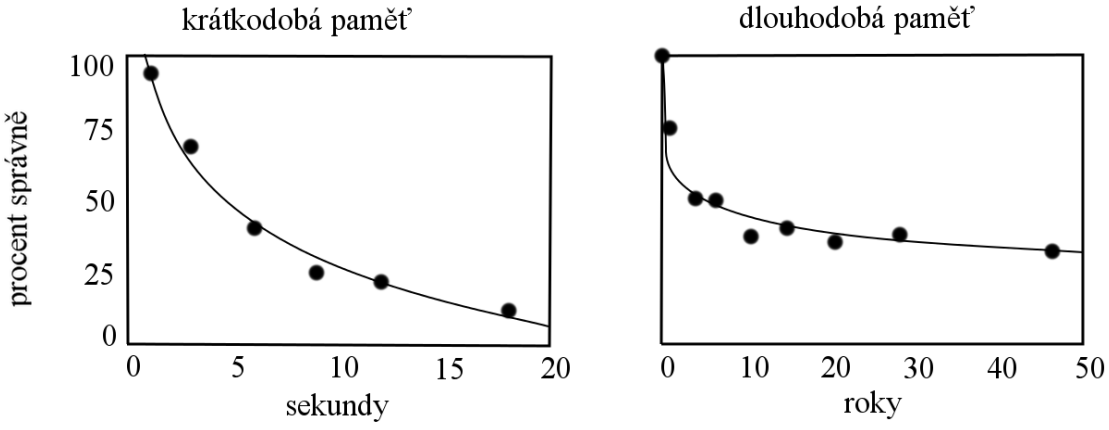
Cílem výukové aplikace je z hlediska paměti nejprve přenést vyučované informace uživateli do deklarativní, krátkodobé paměti. To může být provedeno prezentací vyučované fráze, či slova včetně překladu. Následně tyto paměťové stopy posílit, zajistit přechod do dlouhodobé paměti, případně proměnit paměťovou stopu v procedurální.

Uložit něco do paměti krátkodobé vyžaduje pouze vůli. Informace však chceme přenést do paměti dlouhodobé. K tomuto přenosu je již nezbytná práce ve formě opakování. Efektivitu opakování lze zvýšit motivací. Jinými slovy člověk si lépe pamatuje to, co ho baví [21]. Proto je vhodné aplikaci realizovat tak, aby ji uživatelé považovali za zábavnou a průběh výuky za motivující. Motivaci by mohlo zvýšit přehledné zobrazení průběhu výuky. Zábavnost vhodně zvolené způsoby výuky, líbivé uživatelské rozhraní.

1.2 Zapomínání

Kapacita lidského mozku je omezená, a proto dochází k zapomínání. Zapomínání [10] je vyhasínáním paměťových stop, ke kterému dochází v průběhu času, toto vyhasínání se projevuje změnami v uchování a vybavování. Jestliže něco zapomeneme, nemuselo to naši paměť zcela opustit. Pravděpodobnější je, že paměťová stopa pouze zeslábla a je ještě možné ji opět posílit.

Rozsáhlý výzkum paměti [11] realizoval roku 1885 Německý psycholog Hermann Ebbinghaus. Testoval učení na 2300 slabikách, které nemají smysl. Objevil křivku zapomínání, která ukazuje exponenciální průběh zapomínání v čase. Nejrychleji probíhá zapomínání přímo po naučení. Zjistil, že při jednorázovém učení lze snadno dosáhnout přeučení. Přeučení nastane po určitém množství opakování, po kterém je pokračování asi 7 krát méně efektivní. Tento jev nastává přibližně po dvojnásobku opakování potřebnému k naučení.



Obrázek 1.1: Křivka zapomínání převzata z [19].

O nalezení co nejpřesnější funkce popisující křivku zapomínání se pokusili Rubin a Wenzel v roce 1996 [19]. Použili 210 datových sad, které přetvořili ve formu měřeného

průběhu zapomínání. Na nich ověřili odchylku 105 funkcí dvou proměnných od reálného průběhu zapomínání. Nejmenší odchylku vykazuje logaritmická funkce y = b - m* ln(t).

Z těchto zjištění lze odvodit požadavek, aby aplikace neučila slovíčka, která již dobře umíme. Zároveň je však potřeba respektovat zapomínání. Slovíčko které uživatel dobře umí, může o několik slovíček později odpovědět špatně. O splnění těchto požadavků se musí postarat správně navržený výukový algoritmus.

1.3 Učení s prodlevami

Psycholog Hermann Ebbinghaus objevil *spacing effect* [11], jev který lze přeložit jako učení s prodlevami. Učenou látku si osvojil za 3 dny. Aby si stejně náročnou látku osvojil za 1 den, potřeboval téměř dvojnásobek celkového počtu opakování, čímž ukázal výraznou efektivitu kratšího učení rozloženého do více dnů.

Význam tohoto efektu pro tvorbu výukové aplikace spočívá v tom, že aplikace je používána více dnů, a tak je možnost, aby efektu využívala pro delší pamatování procvičované látky.

Práce [9] se zabývá otázkou trvalosti vzpomínek ve vztahu k prodlevám mezi učením. Skupina 1350 se naučila 32 faktů, na které byla jednoslovná odpověď. Po prodlevách od žádného po 105 dnů došlo k zopakování těchto informací. Po dalších 7 až 350 dnech byla testována znalost faktů. Důležitým závěrem je, že délka optimálního rozložení učení je závislá na tom, jak dlouho si chce člověk naučené pamatovat. Pro zapamatování na 350 dnů byl optimální interval 23 dnů, zatímco pro zapamatování na 7 dnů byla nejlepší prodleva 4 dny.

Viktor de Boer se v diplomové práci [8] zabýval optimalizací učení slovíček. Použil metodu výuky s kartičkami. Slovíčko bylo zobrazeno v rodném jazyce, účastník experimentu pak odpověděl slovíčko v cizím jazyce a byla mu zobrazena správná odpověď. Srovnával 2 skupiny účastníků, obě dostaly testované slovíčko 5 krát. První skupina měla mezi každým testovaným slovem vloženo jiných 10 náhodně vybraných slov, druhá skupina měla vložené slovo pouze 1 jiné slovo. Obě skupiny měly po fázi učení před testem projít další náhodná 2 až 32 slova. Po kratší pauze mezi učením a závěrečnou zkouškou uměly obě skupiny sledované slovo s 94% pravděpodobností. Po dalších 32 slovech klesla šance na vybavení skupiny s 1 vloženým slovíčkem na 56%, zatímco u skupiny s delšími prodlevami mezi učením klesla pouze na 83%.

Tímto dokazuje, že při výuce cizích slov má *spacing effect* vliv nejen z dlouhodobého hlediska v řádu dní, ale i z velmi krátkodobého hlediska během samotného procvičování. Pro výukovou aplikaci jazyků z toho vyplývá, že by měla dělat určité prodlevy mezi opětovným opakováním konkrétního slova. Prodlevy ve formě výuky jiné látky. Vzhledem k tomu, že pořadí prezentace látky je v režii výukového algoritmu, je potřeba dbát na to, aby použitý algoritmus efektu využíval.

1.4 Výuka s kartami

Výuka s kartami, v originále *flashcard*, je metoda výuky cizích slov a frází. Vyučovaný materiál ve formě slov, obrázků nebo čísel je umístěn na dvou stranách karty. Na jedné straně je otázka, na opačné odpověď. Takto se lze učit věty, data nebo jakýkoli materiál, který lze učit formou otázky a odpovědi. Použití karet probíhá zobrazením strany s otázkou,

pokusem o odpověď a zobrazením správné odpovědi. Tato metoda výuky začala jako fyzické karty, nyní převládá využití softwarových implementací.

Softwarové implementace mohou mít libovolnou formu karty i způsobu interakce s uživatelem. Díky této obecnosti můžeme kartovou metodu vypozorovat ve většině úspěšných aplikací pro výuku cizích jazyků.

Tato metoda účinně využívá *testing effect* [18] díky zkoušení vyžadováním správné odpovědi. I jednodušší implementace kartové metody také využívají výše popsaný efekt učení s prodlevami. Tohoto efektu využívají zvětšováním nebo snižováním intervalů (v počtu jiných karet) před opětovným učením karty na základě předpokládané znalosti uživatele, tedy jeho schopnosti správně odpovědět. V základě pro alespoň minimální využití výhod učení s prodlevami stačí prostá predikce podle poslední odpovědi na danou kartu. Když uživatel odpověděl správně, otázku dostane znova později a pokud špatně tak dříve. Složitější algoritmy tuto predikci zlepšují mnoha způsoby. Jiné komerčně používané algoritmy naopak vůbec nepredikují, případně uživatel musí více či méně sám určovat průběh výuky, což efektivitě takového nástroje nepřidá. V této práci se budeme věnovat algoritmům z první skupiny kromě sekce s populárními existujícími aplikacemi, kde se horší variantě zcela vyhnout nelze, neboť nelze ignorovat zájem uživatelů a úspěch aplikace.

Efektivně tento problém řeší mnoho algoritmů. Některé jsou jednodušší, jako Leitner systém, který řadí karty do 5 skupin podle znalostí. Jiné jsou komplexní a sledují individuálně každou kartu jako je algoritmus *ARTS* popsaný dále.

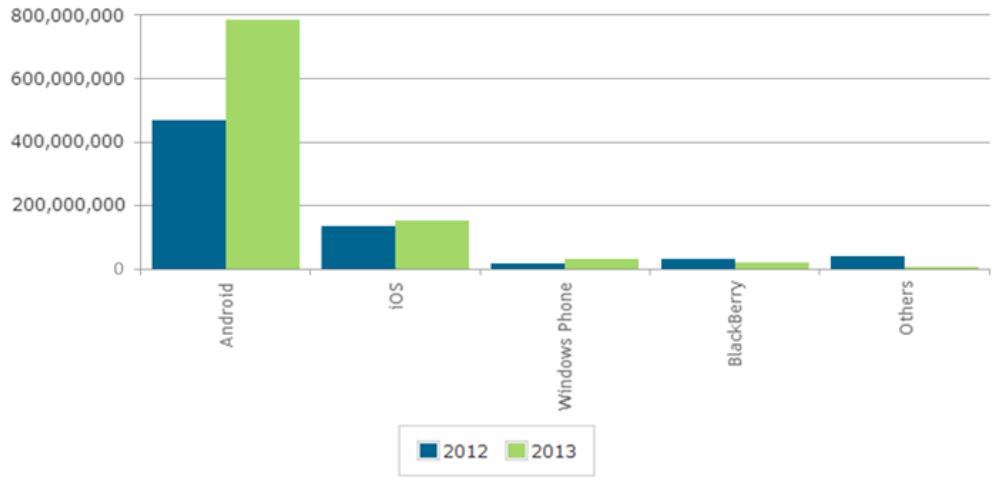
Kapitola 2

Výběr platformy

Trh mobilních zařízení je dnes velmi roztříštěn. O místo na trhu se dělí mnoho platforem. Podíváme se tedy na populární mobilní platformy a vybereme pro jaký systém bude aplikace *Výuka Jazyků* vyvíjena.

2.1 Mobilní OS

Nejrozšířenější operační systémy mobilních zařízení [15] jsou dnes *Android*, který vykazuje největší meziroční růst. Druhým v pořadí je *iOS*, který má celkem stabilní pozici na trhu. Pak následuje *Windows Phone* jehož roční prodej se za 2013 zdvojnásobil. Na ústupu je dnes platforma *BlackBerry*, která naopak skoro o polovinu propadla. Jen nepatrný zlomek trhu zbyl na *OS Bada* a *Symbian*.



Obrázek 2.1: Počet prodaných mobilních zařízení podle OS v letech 2012, 2013.

Společnost Apple distribuuje aplikace pro *iOS* prostřednictvím služby *App Store*, aplikace pro *Android* služba *Google Play* a samostatné jsou i distribuční služby aplikací pro systémy *Blackberry*, *Windows Phone* a *WebOS*. Pro každou podporovanou platformu je pak potřeba aplikaci dělat téměř od začátku a zcela ji přeprogramovat. V ideálním případě

bychom tvořili nativní aplikaci pro každou platformu pokaždé v jiném prostředí a v jiném programovacím jazyce.

2.1.1 Android

Platforma *Android* byla vytvořena společností Google roku 2007 a předána do rukou aliance Open Handset Alliance, které je Google členem. První zařízení, využívající této platformy se objevila na scéně roku 2008.

Psaní aplikací pro *Android* [13] vyžaduje znalosti *XML* a jazyka *Java*. Využívá se syntaxe tohoto jazyka a knihovny tříd, která připomíná podmnožinu knihovny Java SE a obsahuje specifická rozšíření. Nejpoužívanějším vývojovým prostředím je *Eclipse* s přídavným modulem *Android Developer Tools (ADT)* a vývojovým balíčkem *Android Software Development Kit (SDK)*.

2.1.2 iOS

V roce 2007 vypustila firma Apple na trh *iPhone*. Operační systém tohoto mobilního telefonu *iPhone OS* je upravenou verzí *Mac OS X*, což umožňuje rychlé a pohodlné testování v simulátoru na počítačích *Mac*. V roce 2010 vzniklo další zařízení s tímto operačním systémem, *iPad*. Od té doby se označuje jako *iOS*.

Zpočátku byl vývoj aplikací umožněn pouze pomocí webu. Tyto aplikace se spouštěly v zabudovaném prohlížeči Safari a vyžadovaly připojení k internetu. To se nelíbilo uživatelům a omezení daná webovým vývojem znepříjemňovala život vývojářům. Zanedlouho firma Apple umožnila vývoj nativních aplikací, poskytla sadu vývojových nástrojů *Software Development Kit (SDK)*.

Vývoj pro *iOS* [17] probíhá nejčastější v *iOS SDK* společnosti Apple, které obsahuje integrované vývojové prostředí *Xcode*. Využívá programovací jazyk *Objective-C* [12], což je objektová nástavba jazyka *C*. Na rozdíl od *Android*u, na který lze vyvíjet v systémech *Linux*, *Windows* i *OS X*, je sada *iOS SDK* k dispozici pouze pro systém *OS X* a vývojář je tedy nucen pracovat na počítači *Mac*.

2.1.3 Windows Phone

Windows Phone [16] je od roku 2010 následovníkem předchozího mobilního operačního systému firmy *Microsoft*, *Windows Mobile*. *Windows Phone* byl však vyvinut od základu, takže neexistuje zpětná kompatibilita.

2.1.4 Multiplatformní vývoj

Při vývoji pro více platforem zároveň je nejlepším řešením použít jeden nástroj, který vytvoří aplikaci pro všechny cílové platformy. Nabízí se nám tři způsoby kterými je možno se při multiplatformním vývoji vydat.

HTML5

Výkonný kód je interpretován internetovým prohlížečem, ten jako mezivrstva konzumuje značnou část výkonu telefonu. Možnosti čistého *HTML 5* se stále rozšiřují, i přesto jej považuji za vhodný jen pro jednodušší aplikace, kde je kladen důraz na rychlou distribuci.

Hybridní technologie

Samotný vývoj pomocí těchto technologií je stejně snadný. Avšak dle mých zkušeností programy vytvořené tímto způsobem narážejí na problém uživatelského rozhraní, často trpí nedostatkem plynulosti a neestetickým vzhledem.

Nativní vývoj

Podle mého názoru nejpokročilejší varianta multiplatformního vývoje. Využívá po potenciál zařízení na plno a poskytuje tak nástroj pro profesionální použití. Na druhou stranu je ale používání těchto nástrojů nejsložitější a je potřeba naučit se pracovat s novými nástroji a případně i s novým jazykem.

2.2 Zvolená platforma

Mobilní OS se liší v mnoha aspektech. Vývoj pro Android je zdarma, zatímco při vyvíjení pro iOS je potřeba zaplatit nemalý poplatek. Při výběru přihlížím zejména k maximalizaci ziskového potenciálu.

Dělat projekt pro více platforem zároveň zní sice lákavě, ale omezení vyplývající z tohoto způsobu vývoje by mohli ztížit vývoj nebo zhoršit kvalitu výsledné aplikace. Prot

Zbývá rozhodnout, který systém má největší potenciál. Vzhledem k malému podílu na trhu vyřazuji *Windows Phone* a *BlackBerry*. Mezi zbylými kandidáty má *Android* výhodu největšího počtu uživatelů, navzdory tomu má však *iOS* větší prodeje na aplikaci. Finanční výhoda výrazně většího počtu uživatelů je především v rychlém rozšiřování aplikace zdarma vybavené reklamou. Vytvářená aplikace je však koncipována jako placená, a proto volím raději *iOS*. Další výhodou této volby je bohatší potenciální uživatel.

Kapitola 3

Existující výukové aplikace

Před vznikem aplikace bylo potřeba zjistit, jaká je situace na trhu. V obchodech s aplikacemi *Google Play* i *iTunes* jsou desítky výukových aplikací. Pro lepší orientaci v tomto množství aplikací byly vybrány ty, které jsou zajímavé svou vysokou popularitou nebo způsobem výuky. Z výběru jsem naopak vyřadil hry pro děti, neboť ty nejsou cílovou skupinou.

3.1 Mobilní aplikace

Přehled vybraných aplikací pro *Android* a *iOS*.

Angličtina - Mobilní učitel

Jedná se o poměrně neinteraktivní aplikaci. Obsahuje výuková videa, texty. Umožňuje procházet slovíčka, věty a poslouchat vše namluvené rodilým mluvčím. Výuka kartovou metodou má v této aplikaci hned dvě realizace.

Obrázek 3.1: Snímky obrazovek Angličtina - Mobilní učitel

První realizací je výukový režim, jenž zobrazuje slovíčko či frázi spolu s překladem a uživatel volí zda slovíčko zná nebo nikoli. Podle toho aplikace volí následující položky k výuce.

Druhou variací na metodu karet je režim, kdy kartou je slovo v cizím nebo rodném jazyce, mluvené nebo psané. Uživatel má po zobrazení či přečtení slova na výběr ze 3 možností překladu. Podle správnosti volby systém usuzuje, zda uživatel slovo zná nebo ne, což má vliv na plánování, i když jen velmi základní. Co umíme, je dáno na konec fronty vyučovaných slov. Co neumíme je umístěno jinam do fronty.

Angličtina slovíčka

Výuka kartičkami *flashcard*, většina slovíček s výslovností. Po zobrazení slovíčka je překlad skryt a uživatel má na výběr jedno ze tří vyjádření znalosti slovíčka od umím po neumím, navíc možnost vyřadit příliš známé položky z výuky. Uživatel však nemusí odpovídat ihned. Může si nejprve nechat zobrazit překlad či přečíst výslovnost. Během výuky aplikace střídá tři druhy karet: S anglickým slovíčkem, českým nebo pouze výslovnost. Karta zobrazí až tři překlady slovíčka a další jsou snadno k dispozici. Je zobrazen také slovní druh.

Slovíčka jsou rozdělená do několika úrovní podle analýzy jejich používání v angličtině. Program nejprve učí nejfrekventovanější slova, a pak pokračuje méně používanými.

Program umožňuje cvičení formou her šibenice, výběr z možností, psaní slov na klávesnici podle poslechu výslovnosti. Uživatelé si mohou vytvářet vlastní lekce.

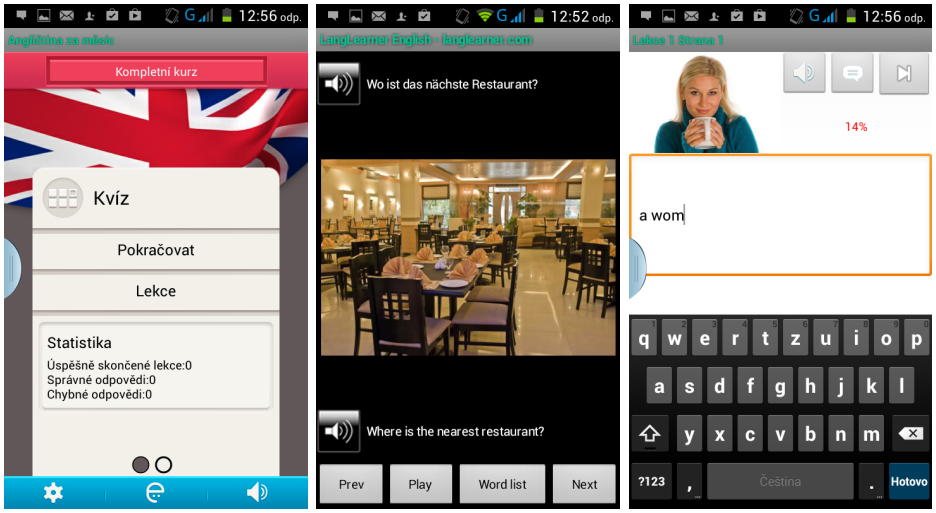
Podle tvrzení autorů je použitý výukový algoritmus netriviální, adaptivní a využívá efekt vyučování s prodlevami *spaced learning*.

Slovíčka

Umožňuje prohlížení slovíček nebo jejich výuku metodou karet. Při výuce je zobrazena karta s překladem a znalost uživatele je testována pouze otázkou zda umí, nebo neumí slovíčko. Možnost tvorby vlastních lekcí.

Angličtina za měsíc

Aplikace má jedinou výukovou metodu. Uživatel přiřazuje cizí slovo nebo větu s výslovností k jedné z 6 fotek. Také výukový algoritmus je zcela triviální. Věty jdou prostě po sobě tak, jak jsou definovány v aplikaci.



Obrázek 3.2: Snímky obrazovek Angličtina za měsíc

Přesto má aplikace mnoho stažení a je umístěna mezi prvními nálezy při vyhledávání výukových programů v obchodě *Google Play*.

Drill - angličtina efektivně

Karty ve formě slovíček s překladem a výslovností. Uživatel hodnotí od 1 do 5 vlastní znalost. Slovíčka označená stupněm 5 jako nejznámější už nejsou znova zobrazena, naopak nejméně známá slovíčka jsou znova zkoušena velmi brzy.

Učit se a hrát Angličtina

Výuka prohlížením ilustrací se slovíčky a jejich poslechem. Program nemá komplexní výukový algoritmus ani žádný přehled o postupu uživatele, ten se při použití hlavní výukové metody prohlížení pouze proklikává k dalším a dalším slovům.

Další způsoby výuky tvoří tři různé kvízy: Přiřazování slov k ilustracím, test pravopisu a psaní slov. Přiřazování slov k ilustracím je zajímavé díky využití obrazového přenosu informace. Přenos informace obrazem je kognitivně méně náročný a uživatelsky příjemnější než přenos informací v podobě textu, který je navíc přenosem sériovým oproti možnosti náhodného přístupu u obrazových dat. I toto přiřazování by však bylo mnohem užitečnější, kdyby bylo vybaveno výukovým algoritmem. Takto je často příliš jednoduché nebo nabízí již známé informace, přestože by program mohl jejich znalost vypozorovat.

Test pravopisu a psaní slov vypadají na první pohled jako použitelné metody. Obě však využívají pro vstup od uživatele standardní klávesnici, z čehož plyne relativně nepohodlné a pomalé psaní díky malým písmenům a celkově malému prostoru ve spodní části displaye na celou klávesnici. Navíc uživatelské rozhraní těchto kvízů není příliš atraktivní, je totiž postavené pouze z neupravených systémových komponent a působí tak spíše dojmem testovacího prototypu, než jako hotová součást aplikace. Výsledkem je, že tyto kvízy nabízejí neefektivní učení v neefektivním prostředí. Bez dořešení jejich nedostatků a bez adaptivního výukového algoritmu tyto kvízy nepovažuji za přínos pro výukovou aplikaci.

Angličtina Ear hry

Výuka jedinou metodou. Touto metodou je poslech mluveného anglického slova a následný výběr ze dvou podobných možností například mezi *pray* a *play* . Nebýt podobnosti voleb, tak uživateli stačí k rozhodnutí první písmeno či délka slova, a tak by se student neučil jazyk, ale spíše rozpoznat od sebe dvě počáteční písmena. Využití podobnosti slov funguje výborně. Uživatel je nucen vybírat na základě drobných rozdílů ve výslovnosti obou slov, a tak si musí těchto rozdílů nejen všímat, ale také se je naučit vnímat. U příkladu *pray* a *play* jde o rozhodnutí mezi anglickou výslovností r a l, ta je podobná, a tak neznalý jazyka snadno rozdíl mezi slovy přeslechne. Aplikace pro výuku angličtiny je celá v angličtině, a tak mohou aplikaci používat i rodilí mluvčí na tréning spisovného psaní slov. Pro česky mluvícího člověka nemá tréning hláskování takový význam. Anglicky mluvící lidé však mohou mít špatný pravopis i těch slov, která denně používají, neboť správný pravopis anglických slov není příliš důležitý pro gramaticky správný mluvený projev.

Výuka anglického jazyka

Výuka pouhým prohlížením slovíček v seznamu je hlavním způsobem prezentace nových znalostí. K zopakování znalostí slouží hra, ve které se z několika písmen skládá překlad

slova. K otestování znalostí je zde primitivní test, kdy uživatel píše pomocí systémové klávesnice.

Skládání slov výběrem z několika málo písmen, z nichž některá slovu patří a jiná nikoliv, se příjemně používá. Těch několik písmen, ze kterých se vybírá, tvoří a učí správný pravopis slov, je reprezentováno dostatečně velkými a snadno stisknutelnými tlačítky, která tvoří praktickou redukovanou verzi klávesnice, které chybí nepotřebná písmena. Díky tomu uživatel, který slovo umí, bez větší námahy slovo rychle napíše.

Navzdory tomu, že by mohl být vstup od uživatele řešen u testového režimu podobně, využívá tento režim pro vstup systémové klávesnice. Na nejmenších přístrojích je díky tomu testový režim náchylný na překlepy a psaní je nepohodlné a pomalé.

Aplikace má dobře zpracované UI výběru lekce, které je tvořeno dobře vypadajícím a přehledným seznamem. Hlavní nevýhoda jinak nadějného výukového nástroje tkví v absenci výukového algoritmu, který by sledoval postup uživatele a uživateli se přizpůsoboval. Program je schopen učení i testování, ale nedokáže využít získaných informací o znalostech uživatele pro jeho další učení. A tak volba toho co a kdy se bude uživatel učit zůstává na něm a na rozložení lekcí v programu. To vede k učení známého učiva na jedné straně a k nedostatečné konfrontaci s učivem neznámým na straně druhé. Nutnost manuálního výběru učiva znamená pro uživatele další práci navíc k samotnému učení a jistě by bylo lepší, kdyby se o výběr učiva staral algoritmus.

3.2 Desktopové aplikace

EuroWord

Program učí slovíčka předkládáním anglických slovíček spolu s českým překladem. Uživatel má možnost zvolit, zda slovíčko umí či neumí. Výukový algoritmus spočívá v tom, že slovíčko, které neumí je zařazeno na konec seznamu se slovíčky k zobrazení.

Word manager

Slova, ve kterých chybujete, bude opakovat častěji. A naopak ta, která již znáte, se budou ve zkoušení objevovat méně. Metody zkoušení: Překlad z jednoho do druhého jazyka, křížovka, hláskování.

LANGMaster Angličtina ELEMENTS, RE-WISE

RE-WISE realizuje kartičkovou metodu učení. Odpověď formou psaní na klávesnici nebo výběrem jednoho z pěti stupňů znalosti od nevím po vím dobře. ELEMENTS je interaktivní učebnicí a cvičebnicí angličtiny. Cvičení jsou založena na dopisování textu do kolonek např. dokončování vět.

3.3 Význam pro vlastní aplikaci

Zkoumané aplikace jsou velmi různorodé a každá má své kladné i stinné stránky. Co si lze odnést do vývoje vlastního výukového programu? Které aspekty by bylo vhodné napodobit a kterým se raději vyhnout? Na tyto i jiné otázky se pokusím najít odpovědi zde.

3.3.1 Využitelné aspekty

Vlastnosti aplikací, které mají pozitivní vliv. Ať už na atraktivitu pro uživatele, účinnost výuky nebo na jiné žádané atributy.

Minimalismus

Některá mobilní zařízení mají velmi malé displaye, a proto může být uživatelské rozhraní přeplněné prvky, tlačítky a nápisy velmi nepřehledné a matoucí. Minimalismus v rámci obrazovky, která bude vykreslena na display, spočívá v malém množství naráz zobrazovaných interaktivních a zobrazovacích prvků. To umožňuje rychlou a kognitivně nenáročnou orientaci uživatele, která je zlepšením v předání informací uživateli. Zároveň sníží náročnost na přesnost kliknutí a pravděpodobnost překliknutí, což je zlepšením v interakci od uživatele. Obě výhody vedou k příjemnějšímu používání aplikace.

Mluvená výslovnost

Aplikace používající krom psaného textu i mluvenou výslovnost jsou při použití zábavnější a zároveň tak využívají komunikačního kanálu mnohem propustnějšího než čtení. Další nespornou výhodou je předání informací o intonaci při výslovnosti i o přesném znění správné výslovnosti.

Dalším případným využitím mluvené výslovnosti je možnost používat aplikaci bez nutnosti interakce nebo bez nutnosti pohledu na display, podle využití.

Dělení materiálu

Vyučovaný materiál by mohl být v jednom dále nerozděleném celku nebo rozdělen zcela náhodně. V některých aplikacích je dělení látky jemnější a jednotlivé části mají logickou návaznost a vnitřní souvislost. Příkladem je rozdělení konverzace na jednotlivá témata s přiměřeným rozsahem, navíc seřazená podle očekávané atraktivity pro studenta.

Dobré rozdělení materiálu usnadňuje uživateli výběr vyučované látky. Zároveň ulehčuje zapamatování díky snadnějšímu zařazení informace. Velikost jednotlivých části by neměla být příliš malá ani velká. Příliš malá by nedala dostatečný prostor výukovým algoritmům. Naopak příliš mnoho naráz vyučovaných položek by unavilo uživatele nebo by vedlo k učení po částech s menší efektivitou.

Uživatelské nastavení

Možnost ovlivnit chování aplikace pomocí trvalého nastavení je atraktivní formou přizpůsobení potřebám uživatele. Také je nastavením možno měnit například jazyk otázky a odpovědi, což je mnohem praktičtější než vytvářet další výukové režimy se zaměněným jazykem otázky a odpovědi.

Podobnost slov

Některá vyučovaná slova se liší jen velmi málo, třeba o jeden znak a i u frází lze najít mnoho podobně vypadajících a podobně dlouhých frází. Tato podobnost je využita aplikací *Angličtina ear hry*, kde zvyšuje obtížnost snadného úkolu volby ze dvou možností.

Využití spočívá tedy ve zvýšení obtížnosti rozlišení několika možných odpovědí pomocí výhodné volby nesprávných odpovědí.

3.3.2 Čemu se vyhnout

Inspirace negativním vzorem. Objevené prohřešky proti uživatelské přívětivosti, estetičnosti, použitelnosti a další. To, čemu se při vývoji aplikace pokusím vyvarovat.

Systémová klávesnice

Některé programy se uchylují k využití systémové klávesnice s velmi malými tlačítky, která svým použitím zdržuje uživatele a zároveň narušuje integritu grafického stylu aplikace. Pokud už by bylo potřeba zadávat písmena, považuji za lepší řešení vlastní formu klávesnice ve stylu aplikace a s menším počtem větších tlačítek resp. písmen.

Absence vlastní grafiky

Uživatelská rozhraní některých aplikací jsou složena pouze ze systémových komponent bez vlastních modifikací či přidaných obrázků. Tento vzhled je sice v souladu s vzhledem systému, na který je uživatel dostatečně zvyklý. Je však vnímán jako velmi nemoderní a primitivní.

Výukové hry

Výuka formou her typu pexeso, sestřelování slov nebo třeba šibenice je zábavnou variantou učení. Efektivita této varianty výuky je však výrazně nižší než u konvenčních metod, neboť herní prvky připravují výuku o čas i o pozornost uživatele. Také působí ve výukové aplikaci jako něco na hraní pro děti, což narušuje vnímání aplikace jako efektivního výukového nástroje pro studenty.

Ruční odstraňování slov

Pokud vyučovaná položka je uživateli známá a nemá zájem o její další opakování, musí ji ručně vyřadit z vyučovaných slov. Na první pohled to nevypadá jako špatné řešení. Při hlubším zkoumání však zjistíme, že uživatel často vyřadí něco předčasně, neboť se přecenil a slovíčko brzy zapomene. Navíc nutnost tohoto úkonu zdržuje výuku.

Dobrý výukový algoritmus přitom tyto neduhy uspokojivě eliminuje. Algoritmus pohlídá, zda uživatel zná slovíčko příliš dobře a dále mu jej nezobrazí. Navíc se o to postará bez nároků na uživatele. Jedinou nevýhodou jsou znalosti, které zná uživatel dostatečně dobře už na začátku výuky. Takovéto známé položky uživatel uvidí o něco vícekrát, než kdyby je při prvním zobrazení ručně vyřadil. Od uživatele se však očekává spíše menšinová znalost látky. Takže lze předpokládat, že výhody řešení, kdy si algoritmus vše hlídá sám, dostatečně převyšují zastaralejší řešení.

Kapitola 4

Návrh vlastního programu

Tato kapitola se zabývá návrhem programu a výběrem efektivního výukového algoritmu.

4.1 Způsob výuky

Nejpoužívanější metodou je výuka pomocí karet. Tuto metodu volím i proto, že využívá dvou důležitých jevů *spacing effect* a *testing effect*, které jsou popsány v teoretické části práce. Realizací této metody je na trhu obrovské množství. Proto se pokusím o netradiční přístup.

4.2 Algoritmus

Při výuce je nezbytné zvolit, která karta s otázkou bude zobrazena jako následující. Volba algoritmu předkládání karet je klíčovou z hlediska optimalizace učení, neboť lze využít učení s prodlevami a počítat tak s křivkou zapomínání.

4.2.1 Triviální algoritmy

Pro nastínění výukových algoritmů si představíme i nejzákladnější možné použitelné algoritmy. I když se to zdá na první pohled nepravděpodobné, i tyto mají své využití v nejmodernějších výukových algoritmech, neboť náhodné předkládání otázek využívají pokročilé algoritmy ve sporných případech, kdy algoritmus považuje několik položek za stejně vhodné k výuce.

Sekvenční

Nejjednodušší metodou zobrazovaní otázek je zobrazování sekvenční. Začneme s první otázkou a po poslední pokračujeme opět od začátku. Tento způsob má několik nevýhod. Otázky, které jsou dobře známé jsou předkládány stále znova a to stejně často jako ostatní.

Náhodný

Náhodná volba další otázky nemá žádné zvláštní výhody oproti sekvenčnímu předkládání otázek. Často se používá k porovnání efektivity adaptivních algoritmů.

Obrázek 4.1: Posloupnost indexů slovíček generovaná sekvenčním algoritmem

Obrázek 4.2: Posloupnost indexů slovíček generovaná náhodným algoritmem

4.2.2 Atkinson system

Tento výukový systém byl popsán v práci Richarda Atkinsona z roku 1972. Cílem práce bylo optimalizovat výuku slovíček na párech anglických a německých slovíček.

Vlastnosti

Jedná se o adaptivní výukový systém, neboť se rozhoduje na základě historie odpovědí.

Prezentace samotného algoritmu je doplněna experimentálním porovnáním účinnosti. První porovnání je s náhodným přidělováním slovíček. Zde byl výsledek v testech lepší o 108%. Při dalším porovnání si uživatelé vybírali další slovíčko sami, zde byl *atkinson system* lepší o 53%.

Vlastní algoritmus

Vlastní algoritmus je založen na rozdělení slov do 3 kategorií P, T a U. Kde P znamená permanentní znalost slovíčka v tom smyslu, že učení dalších slovíček neovlivní pravděpodobnost vybavení správného překladu. T je kategorie slov známých, ale ne natolik, aby další učení neovlivnilo znalost tohoto slova. Kategorie U je pro neznámá slovíčka. Podle kategorií jsou předpokládané pravděpodobnosti vybavení překladu: p = 1 pro P, T p = 0 pro U

Každé slovo má 3 parametry, které řídí průběh učení: x - Pravděpodobnost přesunu ze stavu T (skoro umím) do stavu P (umím) y - Pravděpodobnost přesunu ze stavu U (neumím) do stavu P (umím) z - Pravděpodobnost přesunu ze stavu U (neumím) do stavu T (skoro umím)

1. Všechna slova jsou umístěna do kategorie U, parametry x y z jsou nastaveny na výchozí hodnotu.

2. Jako následující vyučovaná položka se zvolí položka v jiném stavu než P, tedy T nebo U. Volí se ta položka, která má největší pravděpodobnost přesunu do stavu P, porovnává se tedy mezi parametry x a y podle stavu položky. Mezi shodnými parametry vybírá náhodně.

3. Otestuje se slovo, podle správnosti odpovědi se upraví parametry slovíčka a případně se přesune do vyšší či nižší kategorie. Dále se pokračuje 2. krokem.

4.2.3 Langsoft

Firma Langsoft používá ve svých výukových programech *Language Teacher* vlastní adaptivní výukový algoritmus pro výuku cizích jazyků.

Obrázek 4.3: Posloupnost indexů slovíček generovaná algoritmem Langsoftu

Vlastnosti

Tento algoritmus realizuje přizpůsobivou výuku. Je rozhodně účinnější, než algoritmy neadaptivní, neboť neobtěžuje se slovy, která uživatel dobře umí. Na druhou stranu je zde riziko, že vyučovaná fráze bude příliš rychle vyřazena z vyučovaných. Stačí dvakrát odpovědět na frázi dobře a při 40 větách ve výuce se k uživateli fráze nedostane nejméně 117 jiných frází (při 50% pravděpodobnosti na správnou odpověď při prvním setkání se slovíčkem). Dvakrát odpovědět správně se nezdá málo, ale pokud uvážíme například výuku výběrem odpovědi ze, 3 možností tak vychází vyřazení na 1 z 9 neznámých slov jen vlivem náhody.

Výhodou algoritmu je nízká složitost a z toho plynoucí jednoduchost implementace.

Vlastní algoritmus

1. Každá vyučovaná položka má přidělenu hodnotu znalosti, která reprezentuje celkový součet dobrých a špatných odpovědí. Výchozí znalost každé položky je 0 a k její změně dojde po zadání odpovědi.

2. Za další vyučovanou položku zvolí algoritmus tu, která má nejmenší znalost a nebyla zobrazena alespoň N tahů. Při 40 otázkách se fráze zobrazí nejdříve po 5 jiných frází. Při větším počtu frází se stejnou znalostí z nich zvolí algoritmus náhodně.

3. Po odpovědi se upraví hodnota znalosti položky a pokračuje se 1. krokem.

4.2.4 SuperMemo

Jeden z často používaných algoritmů je *SuperMemo* [6]. Tento algoritmus je dobře zdokumentován a má 11 verzí lišících se složitostí implementace, účinností a spolehlivostí. Autorem je Piotr Wozniak. Roku 1987 implementoval program na výuku slovní zásoby. Jednalo se tehdy o jeden z prvních nástrojů prokazatelně zvyšujících účinnost učení. Tento program používal algoritmus *SuperMemo 2*, jeho předchůdce *SM-0* byl používán v papírové verzi. Dále hovoříme o algoritmu *SuperMemo 2* [4], který je dobře zdokumentován.

Vlastnosti

SuperMemo je algoritmus zaměřený na opakování v řádu dní, nikoli v rámci jednotlivého užití aplikace. Prezentací fráze je zde rozuměno použití aplikace, během kterého byla fráze procvičována.

Vlastní algoritmus

1. Rozdělit znalosti do co nejmenších jednotek, například fráze nebo slovíčko. Všem položkám přidělíme $EF = 2{,}5$.

2. Procvičíme několik položek. Po každém procvičení je potřeba upravit *EF* této položky podle úspěšnosti uživatele.

   K tomuto slouží stupnice od 0 do 5:
   5 - perfektní znalost
   4 - správná odpověď po zaváhání
   3 - správná odpověď vybavena s obtížemi
   2 - nesprávné odpovědi, ale správná častěji
   1 - správná vybavena jednou
   0 - kompletní výpadek.

   Po každém procvičení se *EF* upravuje rovnicí
   EF'=EF+(0,1-(5-q)*(0,08+(5-q)*0,02))
   EF'=min(EF', 1,3),
   kde: EF' - nová hodnota *E-Factor*, EF - stará hodnota *E-Factor*, q - kvalita odpovědi na stupnici 0 -5.

   Pokud kvalita odpovědi nedosáhne alespoň 3, vynuluje se počítadlo opakování n.

3. Procvičování položek opakujeme v intervalech I jejichž jednotkou je den.
   I(1):=1
   I(2):=6
   pro n>2: I(n):=I(n-1)*EF,
   kde: n - počet opakování položky I(n) - interval mezi jednotlivými opakováními s parametrem n, jednotkou je den, intervaly se zaokrouhlují. EF - Faktor složitosti položky vyjadřuje náročnost položky na zapamatování, jeho hodnoty se pohybují od 1,3 do 2,5.

4.2.5 ARTS

Algoritmus označován v originále jako *ARTS*, což je zkratka *Adaptive Response Time Based Sequencing*. To by šlo přeložit jako adaptivní na čase odpovědi závislé plánování dalších vyučovaných položek. *ARTS* zkouší vylepšit koncept učení s prodlevami využitím času, který uživatel potřeboval k zodpovězení otázky. Jedná se o informaci, která je u většiny algoritmů naprosto přehlížena. Zatímco většina adaptivních výukových algoritmů využívá při výběru dalších frází historii odpovědí a přesnost uživatele. Čas potřebný k odpovědi je interpretován jako síla paměťové stopy.

Vlastnosti

Tvůrci na malém počtu uživatelů dokazují výrazné zlepšení při srovnání algoritmu *ARTS* s výše popsaným algoritmem *Atkinson system*. Nárust efektivity učení se projevil ve dvou

Obrázek 4.4: Posloupnost indexů slovíček generovaná algoritmem algoritmem *ARTS*

oblastech. V oblasti počtu naučené látky na čas byl nárust 76% a v testech znalostí zlepšení o 53%.

Vlastní algoritmus

Každá vyučovaná položka, tedy slovíčko či fráze má několik atributů. Nejdůležitějším atributem je priorita. Priorita vyjadřuje relativní důležitost výuky dané položky při dalším pokusu o naučení položky.

1. Na začátku je všem vyučovaným položkám přiřazena počáteční priorita. Počáteční prioritu je potřeba nastavit tak, aby byla vyvážená vůči měnícím se prioritám již prezentovaných položek. Je možné nastavit počáteční prioritu tak, aby byly preferovány nové položky, položky již prezentované nebo kombinace obou.

2. K výuce je vybrána položka s největší prioritou. Pokud má více položek stejnou nejvyšší prioritu, vybere se z nich náhodně. Priorita položky s indexem i se vypočítá pomocí rovnice
$$P_i = a(N_i - D)[b(1 - \alpha_i)\log(RT_i/r) + \alpha_i W] \tag{4.1}$$
Kde: RT_i - čas odpovědi, $\alpha_i = 0$ při správné odpovědi, $\alpha_i = 1$ při nesprávné, N_i - počet jiných položek vyzkoušených od poslední konfrontace s položkou, D - nejmenší prodleva před dalším zopakováním položky v počtu položek, a, b, r - konstanty, W - nárust priority při chybě.

3. Začne se počítat čas, který uživateli trvalo zodpovídání položky.

4. Po zodpovězení položky s indexem i se k této položce uloží čas odpovědi RT_i a informace, jestli byla správně zodpovězena $\alpha_i = 0$, při správné odpovědi, $\alpha_i = 1$ při nesprávné. Dále se vynuluje počítadlo stáří N_i, které znamená počet jiných položek vyzkoušených od poslední konfrontace s danou položkou.

5. U všech ostatních položek zvětšíme počítadlo stáří N_i o 1 a pokračujeme 2. krokem.

4.2.6 Volba algoritmu

Při volbě algoritmu je dobré stanovit kritéria výběru. U výukové aplikace je hlavním kritériem to, jak aplikace vyučuje. Formálněji by šlo požadavek formulovat jako efektivitu výuky, průměrný poměr naučených frází za čas.

Z popsaných algoritmů byl vybrán *ARTS*. Využívá největší množství informací, které uživatel při učení vytváří, historii odpovědí a čas odpovědi.

4.2.7 Vyučovaný materiál

Data k výuce dodala firma Langsoft s.r.o. ve formě databází a zvukových souborů wav. Formát databáze je neznámý a firma se vyjádřila, že tuto informaci neposkytne. K databázím bylo ještě dodáno, že obsahují 30 názvů lekcí a 1200 frází či slov. To dělá 40 frází na lekci. Další dodatečnou informací je, že kódování cizích znaků je vlastní, a že tabulky těchto kódování nejsou a nebudou k dispozici.

Formát zvukových souborů je mono wav 8bit. Kvalita nahrávek je velmi nízká, pokud je reprodukční zařízení alespoň na úrovni osobního počítače s kvalitními sluchátky. Nepříjemně posluchač vnímá i absenci dvoukanálového stereo zvuku. Tento problém však na mobilních zařízeních téměř zaniká, neboť mají často výstup mono. Ojedinělá je situace s použitím kvalitních sluchátek. Pro použití na mobilních zařízení je kvalita zvuků dostatečná. Formát zvuku je nepřijatelný z hlediska velikosti, kterou soubory zabírají. Je tedy potřeba převést zvuky do přijatelnějšího formátu dostupného na cílové platformě. Jako tento formát byl zvolen mp3.

Kapitola 5

Implementace

Aplikaci *Výuka Jazyků* jsem se rozhodl implementovat ve vývojovém prostředí *XCode* na systému *Mac OS*. Cílovými zařízeními jsou *iPhone* i *iPad*, tedy telefon i tablet. Podporovaný systém je *iOS* od verze *iOS 6* po aktuální *iOS 8*. Aplikaci implementuji architekturou *MVC* [1], třídy jsou podle účelu rozděleny na *Model* - datové třídy jako třeba přístup k databázi, *View* - třídy reprezentují zobrazitelné prvky a *Controller* - zde je logika aplikace.

5.1 Vývojové nástroje

V této sekci si představíme nejdůležitější nástroje použité při vývoji aplikace, včetně programovacího jazyka, ve kterém je aplikace tvořena. Stručně se seznámíme s *IDE XCode*, které je velmi propracovanou aplikací a díky tomu jsou všechny další zmíněné nástroje jeho součástí.

5.1.1 Objective-C

Nativním jazykem pro programování na *Os X* nebo *iOS* je *Objective-C* [12]. Jedná se o objektově orientovaný jazyk. Objektový přístup je přístupem zdola nahoru, kdy nejprve definujeme jednotlivé základní stavební prvky, objekty. Z nich potom vybudujeme celý program. Díky tomu je možné tyto objekty znovu použít, stačí jen zajištění jejich společné existence a komunikace s ostatními.

Jedná se v podstatě o modelování části reálné skutečnosti. Objekty jsou potom obrazem objektů skutečného světa. Mají svoje vlastnosti a mohou vykonávat určité činnosti. Každý objekt je instancí nějaké třídy objektů. Ta definuje množinu činností, které mají všechny objekty dané třídy společné. Liší se jen vlastnostmi. Tyto vlastnosti se nazývají atributy objektů. Činnosti, které mohou objekty vykonávat jsou metody. Metoda je nejvíce podobná funkci z funkcionálního programování.

Objective-C vytvořili Brad Cox a Tom Love jako objektově orientované rozšíření populárního programovacího jazyka C. Základem návrhu je úplná kompatibilita s jazykem C. Syntaxi převzali z jazyka *Smalltalk*.

5.1.2 XCode

Xcode [7] je integrované vývojové prostředí společnosti Apple, které obsahuje balíček profesionálních vývojářských nástrojů pro vývoj softwarových aplikací na platformy *iOS* a OS X. Jedná se o sofistikovanou aplikaci, která umožňuje jednoduše psát, překládat, ladit

a vykonávat daný program. Apple ho nabízí volně ke stažení z *App Store*, ale pouze pro operační systémy OS X.

5.1.3 iOS Simulator

Aplikace *iOS Simulator* [12] umožňuje rychlou tvorbu prototypů a testovacích verzí při vývoji vaší aplikace. Jedná se o součást prostředí *XCode* spolu s *iOS SDK*. *iOS Simulator* umožňuje simulovat několik *iOS* zařízení a několik verzí operačního systému. Jedná se o předběžný testovací nástroj, před testováním na skutečném zařízení. Simulátor pomáhá vývojářům s testováním, ukazuje, jak se bude aplikace chovat na různých zařízeních.

5.1.4 Interface Builder

Nástroj na tvorbu uživatelského rozhraní integrovaný v prostředí *XCode*. Umožňuje navrhovat uživatelské téměř bez nutnosti psaní zdrojového kódu. *Interface Builder* [7] ušetří čas a úsilí, pokud jde o tvorbu uživatelského rozhraní aplikace. Nemusíme kódovat každý objekt a co víc, protože *Interface Builder* je vizuální editor, vidíme při tvorbě to, jak uživatelské rozhraní aplikace bude vypadat za běhu.

Oproti klasickému operačním systému stolního počítače, který umožňuje mnoho spuštěných programů a má možnost vytvářet a ovládat několik oken, *iOS* umožňuje vaší aplikaci používat pouze jedno okno využívající celou obrazovku zařízení. V tomto jediném okně pak probíhá veškerá interakce s uživatelem.

Každý prvek uživatelského rozhraní v *iOS* dědí *UIViewController* a nazývá se *view*. Celá *iOS* aplikace je složená z několika oken tvořených třídou *UIViewController* mezi kterými uživatel přechází. Okno buď naplníme vlastními prvky nebo použijeme předdefinované dědice *UIViewController* sloužící k zobrazení různých typů dat. Definice použitých oken a vztahy mezi těmito okny jsou v souboru typu *storyboard*, který obsahuje kompletní obraz uživatelského rozhraní. Vztahy mezi okny jsou nejčastěji přechody *segue*, které zajišťují přechod ze stávajícího okna a zobrazení jiného.

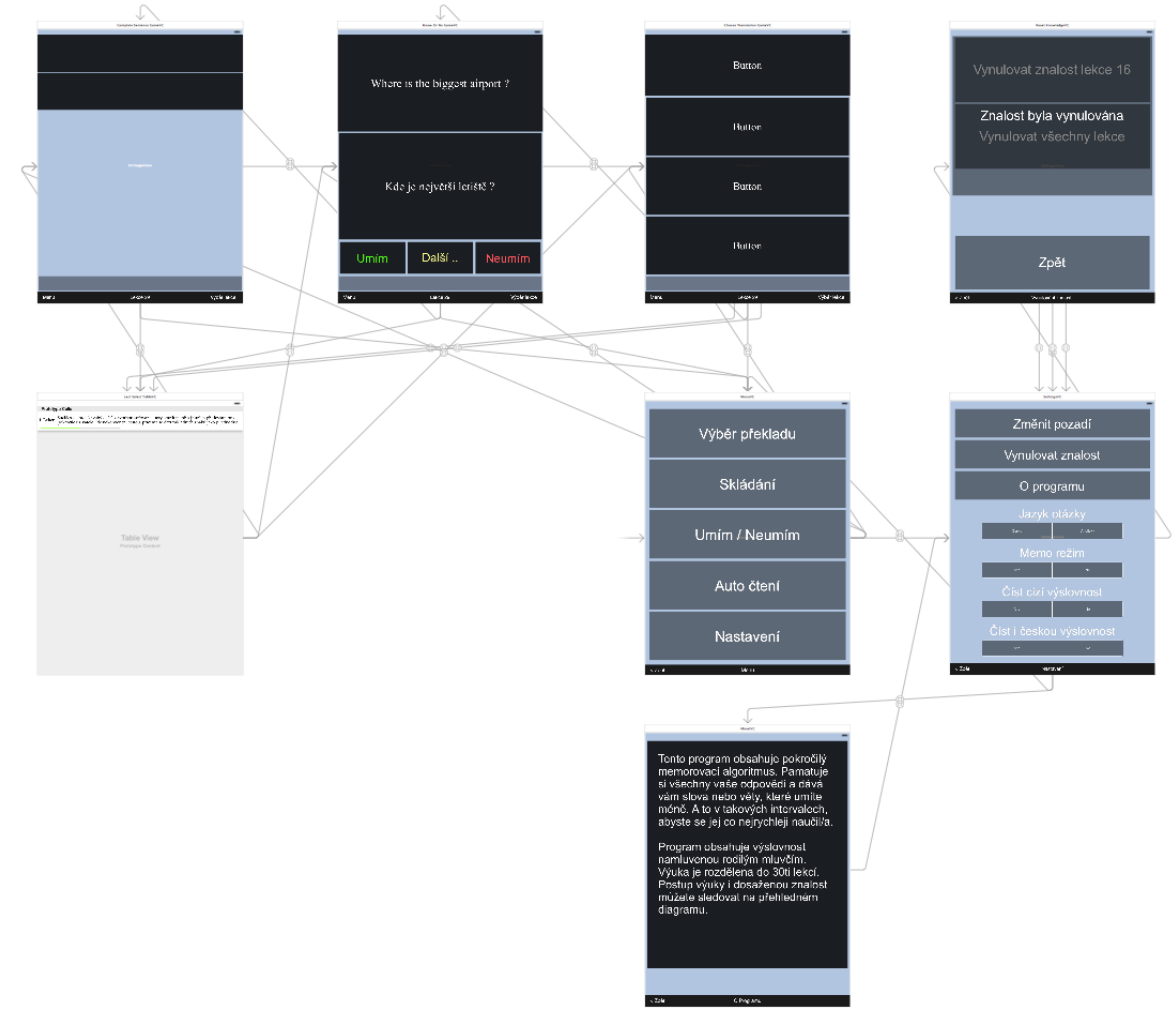
Interface Builder obsahuje nástroj na zobrazení, návrh a úpravy uživatelského rozhraní v podobě souboru *storyboard*, umožňuje přidávat prvky, konfigurovat jejich vlastnosti a vytvořit spojení nejen mezi prvky uživatelského rozhraní, ale i mezi prvky uživatelského rozhraní a kódu.

Návrh probíhá v grafickém prostředí přetahováním komponent do okna zobrazujícího celý *storyboard*. Komponentami jsou předem nastavené objekty nalezené v knihovně. Jsou to okna, ovládací prvky (jako jsou přepínače, textová pole a tlačítka) a pohledy, které budete používat, například zobrazení obrazu nebo tabulka.

5.2 Průběh vývoje

Základní verze potřebuje menu, výuku, výukový algoritmus a databázi s vyučovanými frázemi. Další vhodná rozšíření: vlastní prvky uživatelského rozhraní, nastavení, o programu, více výukových režimů, více výukových algoritmů. V duchu dekompozice jsem implementaci aplikace rozdělil na sadu úkolů, jejichž vyřešení bude znamenat úspěšnou implementaci.

Prvním úkolem je vývoj základní výukové aplikace, která je samostatně použitelná a potenciálně rozšiřitelná na finální verzi. Tato základní výuka musí mít implementován všechny 3 složky *model*, *view* a *controller*. Modelem jsou zde obslužné třídy databází a data pro výuku frází případně i zvuky a databáze výukového algoritmu. Grafické uživatelské rozhraní



Obrázek 5.1: *Storyboard* pro tablety *(iPad, iPad 2, ..)* s přechody mezi obrazovkami zobrazený v nástroji *Interface Builder*

zastupuje *view* a musí být schopné fráze prezentovat a zkoušet. Mezi těmito vrstvami je vrstva řídící v podobě výukového algoritmu, který využívá databáze a řeší požadavky GUI.

Základní verze je sice funkční, ale uživatel si nemůže nic vybrat či nastavit. Je potřeba vyřešit problém výběru učební látky pomocí obrazovky výběru lekce. Vytvořit nastavení aplikace, trvalé uložení uživatelského nastavení. Pro navigaci mezi jednotlivými obrazovkami slouží menu.

Vzhled a působení aplikace jsou stejně důležité či dokonce důležitější, než funkčnost. Neupravené prvky uživatelského rozhraní použité tak, jak je systém nabízí, působí neprofesionálně. Navíc dobře prodávané aplikace málokdy nemají vlastní grafiku. Proto je potřeba nahradit nepřizpůsobené prvky uživatelského rozhraní vlastními třídami s líbivějším vzhledem. Také by bylo dobré vytvořit diagram, který přehledně zobrazí průběh výukového algoritmu.

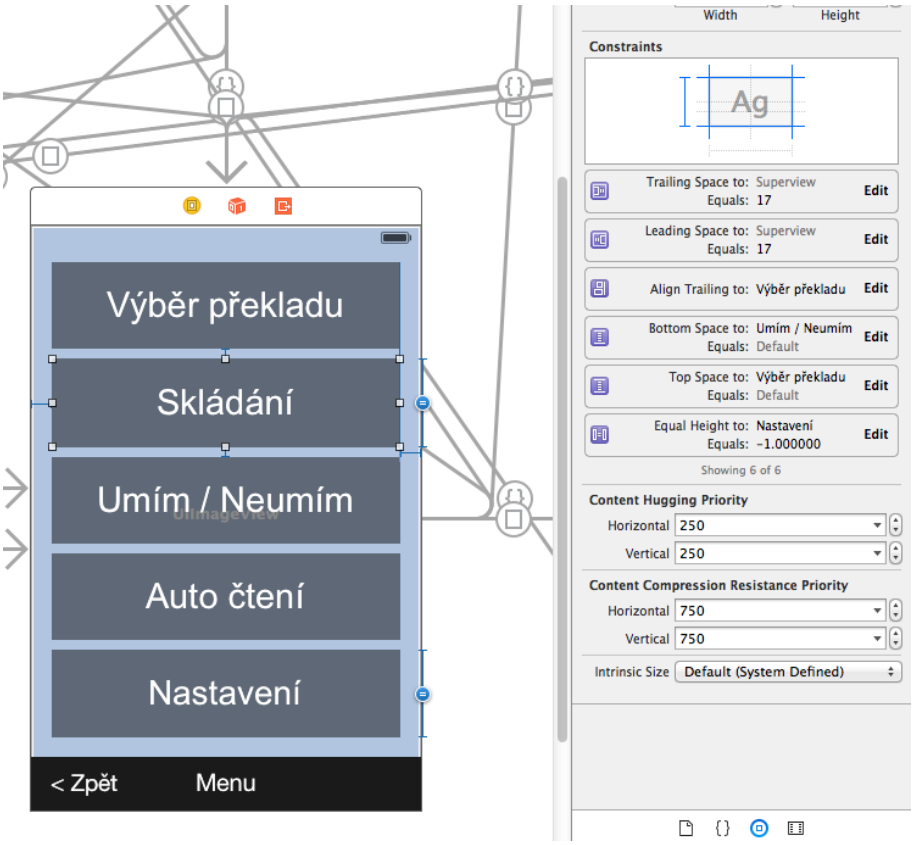
S hotovou základní verzí je další na řadě její rozšíření o další výukové režimy prezentující a zkoušející jiným způsobem.

5.3 Uživatelské rozhraní

Při návrhu uživatelského rozhraní bylo cílem jednoduché a přehledné uživatelské rozhraní. Neposledním cílem UI je nezaostávat ani v atraktivitě pro potenciálního uživatele.

Uživatelské prostředí je vytvořeno pomocí nástroje *Inteface Builder* a má formu 2 souborů *storyboard*. Uživatelské rozhraní se skládá z 10 obrazovek. Mezi těmito obrazovkami se přechází kliknutím na tlačítko či na vybranou lekci. Každá obrazovka má svůj zdrojový kód ve formě dědice třídy *UIViewController*.

Rozmístění prvků na obrazovce lze realizovat několika způsoby. Nejjednodušší je rozmístit je ručně pro každé rozlišení. Počet různých rozlišení při vývoji na *iPad* a *iPhone* je dán počtem různých fyzických rozlišení displayů a je násobem počtem rozdílů rozlišení zobrazovací plochy v podporovaných verzích *iOS*. Toto řešení postrádá róbustnost a i po malé úpravě UI, například spodní lišty, by bylo potřeba znova ručně udělat většinu oken.



Obrázek 5.2: Tvorba *auto layout* v *Interface Builder* - Nastavení *constraints* tlačítka skládání v obrazovce menu pro telefony *iPhone*. Zvýrazněno nastavení shodné výšky u tlačítek skládání a nastavení.

Proto jsem se rozhodl využít jiný způsob rozmístění prvků v obrazovkách zvaný *auto layout*. Tento systém funguje tak, že nastavíme prvkům v okně vzájemné vztahy, takzvané *constraints*. Samotné rozmístění a velikosti prvků se pak vypočítají přímo při použití okna za běhu aplikace a dokonale se přizpůsobí displayi, ať už má jakoukoliv velikost či poměr stran. Těmito vztahy mezi prvky jsou většinou vzdálenosti mezi prvky a shodnost rozměrů 2 a více prvků či třeba určení poměru těchto rozměrů nebo zarovnání některé souřadnice.

Samotná tvorba takového rozhraní vyžaduje trochu zkušeností, neboť automatický sys-

tém často funguje poměrně svérázně, a tak je potřeba chvíli testovat a některé vlastnosti implementovat alternativním způsobem. V aplikaci *Výuka Jazyků* je takto netradičně řešena horní černá průhledná status lišta, neboť v *iOS 6* je lišta přítomna sama od sebe a definujeme tedy zbytek obrazovky, zatímco v *iOS 7, 8* status lišta není a celé uživatelské rozhraní se posune o velikost lišty nahoru, případně se jiným způsobem roztáhne. Vždy to však vedlo k deformaci obrazovky. Řešením je vlastní lišta imitující systémovou, která je napevno umístěná do UI spolu s programovou obsluhou která lištu buď odstraní nebo ponechá, podle verze systému.

5.3.1 Grafické efekty

Prvky ze kterých je rozhraní složeno, jako *UIButton*, poskytnuté prostředím jsou dobrým základem pro vlastní úpravy, nicméně bez těchto úprav nesplňují estetické požadavky moderní aplikace. Proto je potřeba použít prvky cizí, udělat vlastní či upravit stávající. Rozhodl jsem se pro poslední možnost a to upravit systémové komponenty. K vylepšení prvku tlačítko, které používám i pro zobrazení textu, stačilo rozšířit standardní implementaci o jednu metodu *makePretty*. Tato metoda vykreslí částečně průhledné pozadí, černou linku se zaoblenými rohy kolem prvku a stín kolem prvku. Je nastavena jiná barva při stisku, jako forma zvýraznění, standardně modrá, speciální význam mají zvýraznění červené a zelené.

Obrázek 5.3: Ukázka grafických efektů na tlačítku výuky. Zleva tlačítko před použitím třídy *UIButtonPretty*, po použití třídy a zvýrazněné tlačítko při stisku.

5.3.2 Reprezentace znalostí

Je potřeba zobrazit průběh výuky pro lepší orientaci uživatele i pro jeho motivaci. Informacemi, které chceme reprezentovat jsou stavy znalosti jednotlivých položek v lekci. Také je potřeba vyjádřit, která slova z lekce jsou aktivně vyučována, a která čekají až se uživatel zlepší a v neposlední řadě zvýraznit aktuálně vyučovanou položku. O přehledné zobrazení těchto informací se stará diagram znalostí. Diagram reprezentuje každou položku z lekce stupnicí začínající na 0. V případě změny znalosti dojde k růstu či klesání ukazatele, ten má z důvodů názornosti zelenou barvu pro kladné hodnoty a červenou pro záporné. Neaktivní část diagramu je zvýrazněna průhledností zobrazení položek. Položka, která je zrovna učena, je zvýrazněna bílou barvou počátku stupnice, místo barvy šedé.

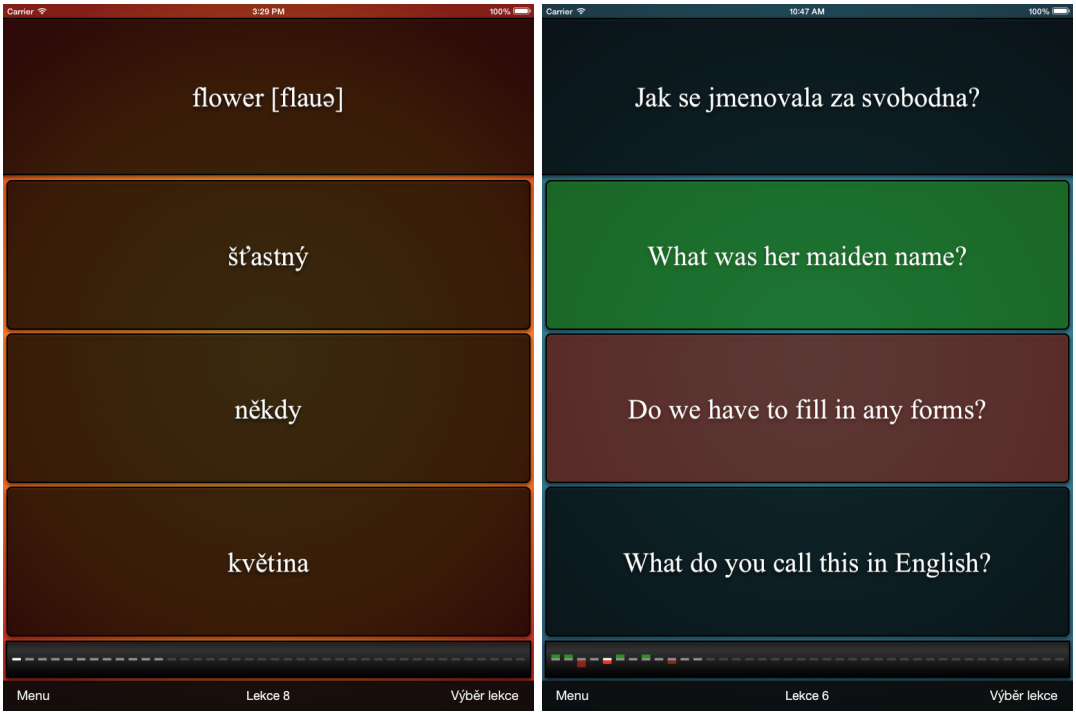
Obrázek 5.4: Prvek uživatelského rozhraní diagram znalostí slouží k zobrazení informací o průběhu výuky a o předpokládaných znalostech uživatele.

5.3.3 Metody výuky

Jako nástroj výuky jsem zvolil několik metod výuky, které různým způsobem realizující metodu karet. Jaké metody to jsou a jak fungují je obsahem této podkapitoly.

Výběr překladu

Výuka formou výběru překladu zobrazí uživateli větu či slovíčko v českém nebo cizím jazyce spolu se třemi možnostmi překladu. Alternativně lze kliknutím na zadanou větu přejít na jinou. Uživatel zvolí překlad, který považuje za správný. Výuka zvýrazní chybnou odpověď červeným pozadím stisknutého prvku, správnou odpověď zvýrazní zeleně. Efekt odpovědi je okamžitě patrný také na diagramu znalostí, kde lze pozorovat změnu znalosti nebo i zvětšení vyučované části lekce. Po zodpovězení přečte výuka výslovnost. Čtení opakuje do té doby, než uživatel uvolní stisk odpovědi.



Obrázek 5.5: Snímky obrazovky Výběr překladu řízené třídou *ChooseTranslationGameVC* ze simulátoru *iPad 2*.

Skládání vět

V tomto režimu je uživateli zobrazena věta v českém nebo cizím jazyce a upravená verze překladu, která má místo písmen tečky. Pod tímto zadáním je několik slov v cizím nebo českém jazyce. Jsou zde všechna slova z přeložené věty a několik slov, která do ní nepatří. Uživatel se pokusí vybrat ze kterých slov je překlad složen. Je možné splést několik slov, po té je oznámen neúspěch a pokračuje se další větou. Ať už uživatel složí správnou větu nebo ne, jeho pokus je zakončen zobrazením správného překladu a přečtením výslovnosti. Opět platí, že čtení výslovnosti se opakuje do té doby, než uživatel uvolní stisk odpovědi.

Z hlediska implementace bylo největší problém programové rozmístění proměnného počtu tlačítek proměnné velikosti do opět proměnného prostoru mezi textem odpovědi a diagramem znalostí.

Obrázek 5.6: Snímky obrazovky Skládání vět řízené na tabletu třídou *CompleteSentenceGameVC* ze simulátoru *iPad 2*.

Umím, neumím

Tento výukový režim opět začíná zobrazením věty nebo slovíčka v českém nebo cizím jazyce. Její překlad však zůstává skryt po dobu několik sekund. Během této doby je místo textu zobrazeno odpočítávání od 5. Uživatel může překlad zobrazit předčasně kliknutím. Odpovědět lze však i před před zobrazením překladu. Jsou nabídnuty tři odpovědi: *Umím*, *Neumím* a *Další ..*, tyto odpovědi se liší pouze vlivem na průběh učení a diagram znalostí. Po odpovědi je zobrazen správný překlad a následně se přejde na další otázku.

5.3.4 Výukový algoritmus

Požadavkem zadavatele bylo implementovat algoritmus firmy Langsoft. Tento algoritmus je vyzkoušen léty praxe v PC aplikaci *Language Teacher* avšak na základě teorie se domnívám, že se jedná o jeden z nejméně efektivních adaptivních algoritmů.

Pro větší použitelnost samotné aplikace jsem se rozhodl k implementaci 2 výukových algoritmů. Největší a tedy nejpozorovatelnější kontrast by k nejhoršímu algoritmu mohl tvořit ten nejlepší. Ve výše uvedeném odstavci jsem jako nejlepší algoritmus zvolil *ARTS* pro jeho unikátní vlastnosti.

Použití dvou výukových algoritmů intuitivně implementují 2 různé třídy *AlgorithmARTS* a *AlgorithmLangsoft*, které společně dědí ze třídy *GameBase*.

Obrázek 5.7: Snímky obrazovky Umím neumím řízené na tabletu třídou *KnowOrNoGameVC* ze simulátoru *iPad 2*.

Přepínání použitých algoritmů

Třídy implementující výuku používají třídu Game, která nemá žádné vlastní metody, pouze dědí jednu ze tříd s algoritmy. Aplikace samotná je navržena, aby pracovala pouze s jedním z algoritmů. Volba algoritmu probíhá tedy již při překladu. Nastavením předka třídy Game se určí využívaný výukový algoritmus.

Implementace algoritmu *ARTS* v jeho původním znění musí při každém výběru vyučované položky znova vypočítat prioritu všech položek v dané lekci a zároveň všem kromě 1 je potřeba zvýšit počítadlo. To znamená velké množství přístupů do databáze. Při implementaci jsem se rozhodl realizaci algoritmu optimalizovat při zachování funkcionality.

Optimalizovaná implementace ARTS

Když je zodpovězena otázka, původní implementace ukládá čas odpovědi a její správnost. Nyní ukládáme raději výsledek samotné prioritní rovnice, ze které byla vytknuta variabilní část $N_i - D$. Touto úpravou zmizela nutnost přepočítávat priority všech položek před každým hledáním nejvyšší priority. Dále jsem pro přehlednost rozdělil zbylou rovnici na 2 případy podle správnosti odpovědi.

Veškerý výpočet se nyní realizuje při odpovědi otázky a omezuje se na jedno spočítání rovnice a na inkrementaci priorit všech ostatních položek o jejich prioritní přírustky. Toto vede k téměř polovičním nárokům na zápisy do databáze a až n-krát menší četnosti počítání hlavní rovnice *ARTS*, kde n je počet položek ve vyučované lekci.

Vlastní algoritmus

1. Položkám se nastaví počáteční priorita P_i a vynuluje přírustek priority Pp_i.
2. K výuce je vybrána položka s největší prioritou. Pokud má více položek stejnou nejvyšší prioritu, vybere se z nich náhodně.
3. Začne se počítat čas RT_i, který uživateli trvalo zodpovídání položky.
4. Po zodpovězení položky s indexem i se k této položce uloží přirustek priority Pp_i, který se vypočítá při správné odpovědi pomocí rovnice

$$Pp_i = ab \log(RT_i/r) \tag{5.1}$$

,kde: RT_i - čas odpovědi a, b, r - konstanty

Při nesprávné odpovědi se přírustek priority Pp_i vypočítá

$$Pp_i = aW \tag{5.2}$$

,kde: a - konstanta, W - nárust priority při chybě

5. U všech ostatních položek aktualizujeme prioritu P_i jejím zvětšením o jejich přírustek priority, znázorněno rovnicí

$$P_i = P_i + Pp_i \tag{5.3}$$

5.4 Popis implementace

Zdrojový kód projektu se skládá z 31 tříd rozdělených do 3 složek *Controller*, *View* a *Model*. Vzhledem k počtu tříd vynecháme popis velmi jednoduchých tříd, jejichž význam je zřejmý již z názvu.

5.4.1 Controller

Zde můžeme nalézt 13 tříd tvořících logiku aplikace. V podsložkách jsou umístěny výukové algoritmy a různé způsoby výuky.

AppDelegate

Tato třída rozšiřuje *UIApplicationDelegate* [2], což je povinná systémová třída sloužící k interakci se systémem a jinými aplikacemi. Tento vyslanec aplikace běží současně se samotným objektem aplikace, čímž zaručuje správnou reakci na důležité změny. Mezi metodami jsou například změna stavu aplikace při přechodu do popředí.

Mp3Player

Tato třída slouží k přehrávání zvukových souborů ve formátu mp3. K přehrávání je požita systémová třída *AVAudioPlayer*, která komfortním způsobem umožňuje práci se zvukovými soubory. Po spuštění přehráváni vrací trvání zvukové nahrávky, aby volající třída věděla, kdy přehrávání skončilo. Přehrávání lze také přerušit v průběhu.

DismissSegue

Třída rozšiřující *UIStoryboardSegue* slouží jako přechod v souboru *storyboard*. Tento přechod místo přechodu na jiný *view* po použití uzavře *view*, kterým byl volán, což využívají výukové režimy ke svému uzavření a návratu do hlavní nabídky aplikace.

/Game Algorithm/GameBase

Tato třída je společným základem implementace výukových metod a algoritmů. Obsahuje metody související s průběhem výuky.

/Game Algorithm/

Zde jsou třídy s výukovými algoritmy, rozšiřující třídu *GameBase*.

/Game Algorithm/Game

Tato třída slouží jako prostředník mezi výukovým algoritmem a zbytkem programu. Díky ní lze programově zvolit algoritmus použitý v aplikaci změnou rodičovské třídy, může dědit ze tříd *AlgorithmARTS* a *AlgorithmLangsoft*.

/Game Types/

Tyto tři třídy rozšiřují třídu *Game* a tím implementují výukové metody.

/InApp Purchase/StoreObserver

Obsluha nákupů v aplikaci, výuková aplikace nabízí ke koupi českou výslovnost. Základ třídy je poskytnutý firmou Apple.

5.4.2 View

O uživatelské rozhraní se stará celkem 14 tříd. Z toho 10 je řadičů pohledů, tříd rozšiřujících *UIViewController*, které ovládají každá jednu obrazovku aplikace. Zbylé 4 třídy jsou komponenty uživatelského rozhraní, které dědí z *UIControl*.

Podle návrhového vzoru *MVC* se *view controller* řadí do řídící části. Je totiž řídícím kódem celého okna. Dokáže však mnohem víc, stará se například o rotace uživatelského rozhraní. Spíše do oblasti uživatelského rozhraní jej řadí i to, že poskytuje zobrazitelný interaktivní prvek *view* pokrývající celé okno, který je kořenem celé hierarchie dalších prvků dohromady tvořících celé uživatelské rozhraní. *View controller* [5] je potom správcem těchto prvků zajišťujícím komunikaci mezi prvky. Proto, že spolu se souborem *storyboard* tvoří celé uživatelské rozhraní, jsou řadiče pohledu z hlediska vzoru *MVC* v této práci zařazeny jako *view*.

/View Control/MyUIViewController

Třída je prostředníkem mezi systémovým *UIViewController* (ze kterého dědí) a třídami ovládajícími jednotlivé obrazovky, které rozšiřují *MyUIViewController*. V této třídě lze implementovat globální změny nad všemi obrazovkami zároveň.

Je zde řešení horní černé poloprůhledné status lišty, neboť v *iOS 6* je lišta přítomna sama od sebe a definujeme tedy zbytek obrazovky, zatímco v *iOS 7, 8* status lišta není

a celé uživatelské rozhraní se posune o velikost lišty nahoru, případně se jiným způsobem roztáhne. Vždy to však vedlo k deformaci obrazovky. Řešením je vlastní lišta imitující systémovou.

Tato třída při načtení obrazovky, zjistí jestli je verze systému větší než *iOS 6.1* . Pokud ano, vytvoří prvek uživatelského rozhraní imitující lištu, nápis bez textu s průhledným pozadím černé barvy a vloží jej na horní roh obrazovky. Po této úpravě se chová aplikace na všech podporovaných verzích operačního systému stejně.

/Components/UIKnowledgeDiagram

Třída slouží k definici vykreslování diagramu znalostí. Třída rozšiřuje prvek uživatelského rozhraní *UIControl* pomocí vlastní definice metody *drawRect* volané uživatelským rozhraním při vytvoření prvku. Vykreslí se gradient tvořící pozadí, zaoblené okraje diagramu, stínování, nakonec se do diagramu dokreslí sady různě barevných obdélníků reprezentujících jednotlivé věty.

/View Control/MenuVC

Implementace menu aplikace se zaměřením na jednoduchost a přehlednost.

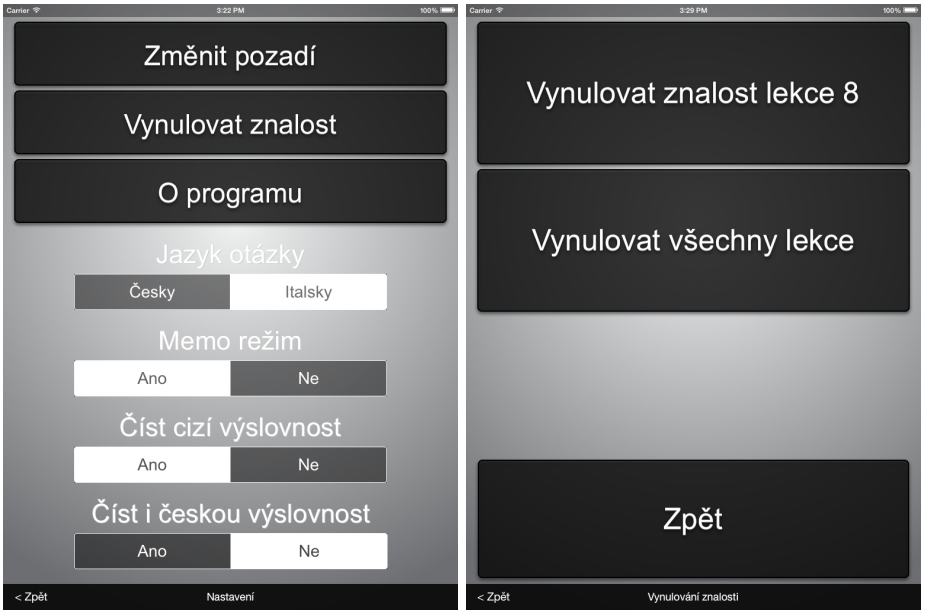
Obrázek 5.8: Spouštěcí obrazovka aplikace a po ní následující Menu řízené třídou *MenuVC* na tabletu *iPad 2*

/View Control/SettingsVC

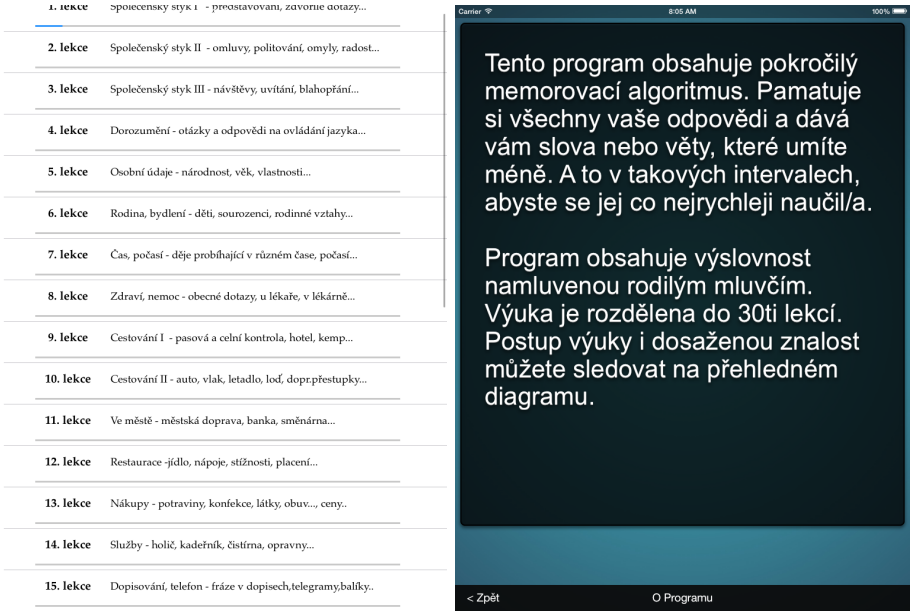
V duchu jednoduchosti a přehlednosti je vytvořeno i nastavení aplikace.

/View Control/LectSelectTableVC

Výběr lekce realizuje seznam lekcí.



Obrázek 5.9: Obrazovky Nastavení, Vynulování postupu řízené třídami*SettingsVC*, *ResetKnowledgeVC* na tabletu *iPad 2*



Obrázek 5.10: Obrazovky Výběr lekce, O programu řízené třídami *LectSelectTableVC*, *AboutVC* na tabletu *iPad 2*

5.4.3 Model

Třídy zajišťující trvalé uložení dat. Jsou zde 3 třídy pro práci s perzistentními daty, s 2 databázemi a nastavením aplikace.

PersistentSettings

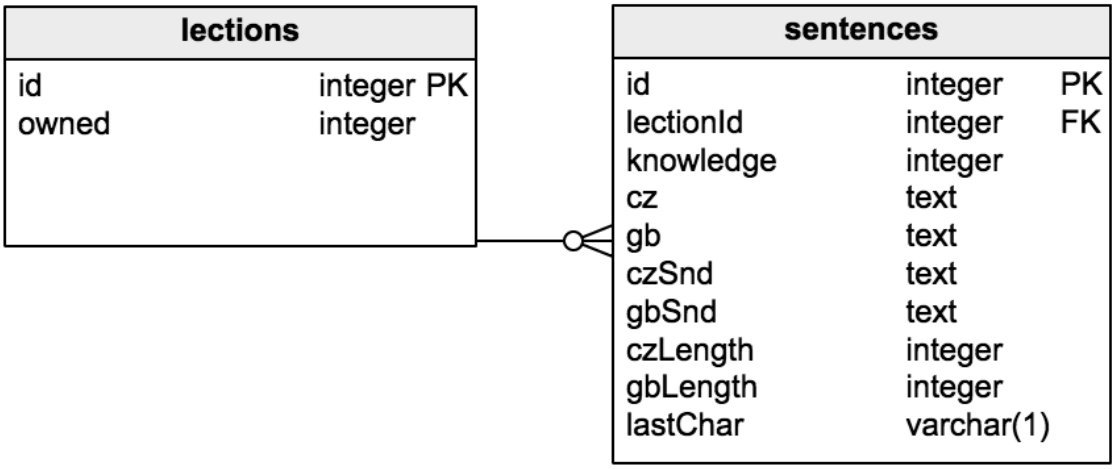
Sdílená třída typu *singleton* přístupná z celé aplikace slouží pro udržování a trvalé uložení nastavení a stavu aplikace. Umožňuje číst své hodnoty, uložit nastavení a obnovit nastavení.

MySQLiteDb

Třída pro obsluhu hlavní databáze nabízí jednoduché metody pro interakci s databází, aby ostatní třídy nemusely pracovat s SQL příkazy.

Hlavní databáze aplikace obsahuje data pro výuku a postup výuky včetně předpokládané znalosti každé vyučované položky. Jednodušší algoritmy včetně implementovaného algoritmu *Langsoft* si vystačí pro svůj běh pouze s touto databází. Složitější výukové algoritmy si musejí dodatečná data ukládat do vlastní databáze, a také se postarat o korektní spolupráci s hlavní databází a validitu svých dat.

Databáze má dvě tabulky: lekce *lections* a věty či slovíčka v nich obsažená *sentences*. Lekcí je 30, každá má svůj název a obsahuje 40 vět či slovíček s dodatečnými informacemi. Atributy tabulky *sentences* obsahují číslo lekce, dosaženou znalost položky, text ve dvou jazycích, jména souborů s výslovností. Zbývající atributy jsou délky textů a interpunkce z konce věty. Ty slouží pro vyhledávání podobných položek, což aplikace využívá při nabízení nesprávných odpovědí, které se mají lišit od správné odpovědi co nejméně.



Obrázek 5.11: Snímky obrazovky Skládání vět řízené na tabletu třídou *CompleteSentenceGameVC* ze simulátoru *iPad 2*.

ARTSDbManager

Databáze algoritmu *ARTS* je mnohem jednodušší než hlavní databáze s daty, a to proto, že je pouze doplňkem databáze s výukovým materiálem, abychom se vyhnuli zbytečné redundanci. Propojení obou databází není na úrovni *SQLite*, pouze na programové úrovni. Atribut *id* tabulky *items* z *ARTS* funguje v tomto vztahu jako cizí klíč odkazující na stejnojmenný atribut *id* tabulky *sentences* v hlavní databázi.

items	
id	integer PK
priority	real
priorityAddition	real
lastAnswerTorrent	integer

Obrázek 5.12: Snímky obrazovky Skládání vět řízené na tabletu třídou *CompleteSentenceGameVC* ze simulátoru *iPad 2*.

Kapitola 6

Testování

Jako metodu pro testování výukové aplikace jsem zvolil případovou studii. Případová studie je jednou z metod kvalitativního výzkumu, bývá charakterizována jako detailní studium konkrétního případu. Jedná se o výzkumnou metodou běžně užívanou v disciplínách psychologie, sociologie a jiných. Jelikož se při testování snažíme zjistit, jak aplikace působí na uživatele, lze případová studie použít i v tomto případě.

Testování aplikace je usnadněno možností distribuovat hotovou aplikaci pomocí internetu. Testovací skupina 19 uživatelů byla zajištěna hlavně pomocí sociálních sítí *Twitter* a *Facebook*. Sběr dat proběhl formou dotazníku. Uživatelé dostali dotazník a 2 balíčky s aplikací *Výuka Jazyků* lišící se výukovým algoritmem, instrukce k instalaci a spuštění avšak bez dalšího návodu k použití aplikace.

Cílem testování je ověřit platnost několika tvrzení o aplikaci pomocí vhodné metody pro testování hypotéz.

6.1 Postup testování

Nejprve je potřeba stanovit testovaná tvrzení, zajistit testovací skupinu uživatelů, zvolit a realizovat metodu sběru dat. Pomocí nasbíraných dat pak provedeme výpočty vedoucí k potvrzení nebo zamítnutí testovaných tvrzení.

6.1.1 Testovaná tvrzení

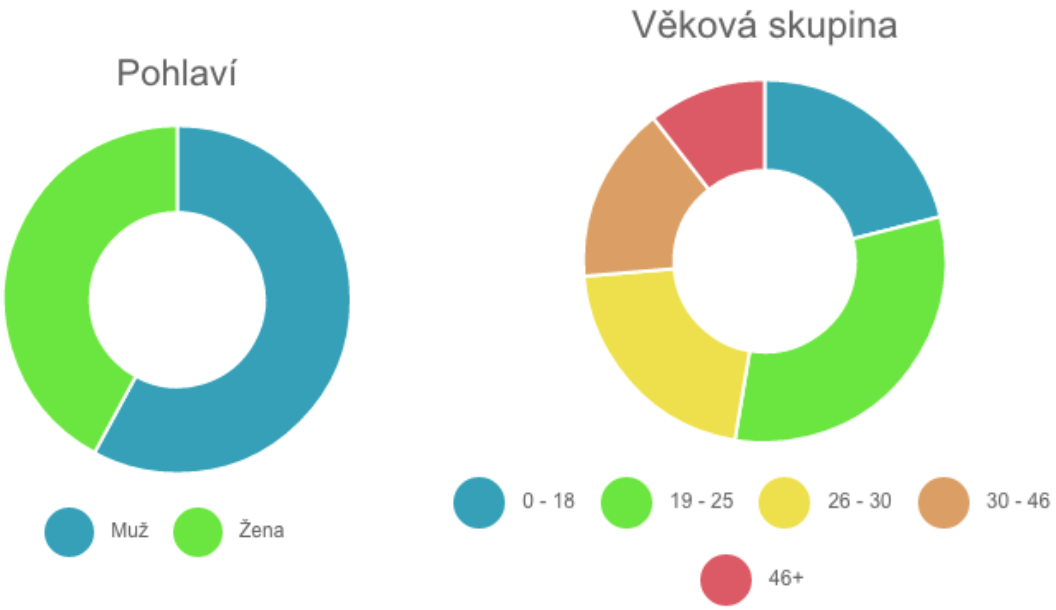
Seznam tvrzení, jejichž dokazováním či vyvracením se budeme zabývat.

- Aplikace se snadno používá, je intuitivní.
- Uživatelské rozhraní je pro uživatele atraktivní.
- Algoritmus *ARTS* předčí jednodušší algoritmus firmy *Langsoft*.
- Aplikace je uživateli přijímána pozitivně.

6.1.2 Sběr dat - dotazník

Při výběru metody sběru dat jsem se rozhodl pro dotazník kvůli možnosti využít k testování většího počtu uživatelů, než by umožnilo například pozorování jednotlivých uživatelů. K aplikacím dostali testující uživatelé přiložený dotazník, jehož otázky by měli umožnit testování stanovených tvrzení. Tento dotazník se skládá z několika otázek a začíná 6 instrukcemi,

jsou jimi některé případy užití aplikace představující možnosti *Výuky Jazyků*, aby měl uživatel prostor otestovat aplikaci. Testující se musí zorientovat v neznámém uživatelském rozhraní nové aplikace čímž zároveň prověří její intuitivnost.



Obrázek 6.1: Grafy zobrazující rozložení pohlaví a věkových skupin testujících uživatelů získané z dotazníků.

6.1.3 Metoda zpracování dat

Potřebujeme ověřit platnost několika tvrzení, která budeme nazývat statistické hypotézy. K tomuto použijeme matematický postup vedoucí k zamítnutí či nezamítnutí dané hypotézy, tento postup testování hypotéz označujeme jako statistický test (test významnosti).

6.2 Testování hypotéz

Při testování statistických hypotéz [14] vždy porovnáváme dvě skupiny případů, tedy dvě hypotézy. Hypotéza, kterou se testováním pokoušíme zamítnout se nazývá nulová, značí se obvykle H_0. Druhá hypotéza se nazývá alternativní hypotéza, značí se obvykle H_1. Nejprve se tedy zformulují 2 doplňující se hypotézy: nulová a alternativní, kde alternativní vymezuje situaci, když nulová hypotéza neplatí.

V případě testování aplikace *Výuky Jazyků* je alternativní hypotézou očekávaný efekt implementace, tedy například to, že UI je intuitivní. Pak se pokusíme o zamítnutí nulové hypotézy, která je opakem, tedy UI není intuitivní.

Nemůžeme absolutně dokázat platnost tvrzení. Můžeme pouze rozhodnout, zda danou hypotézu zamítáme a dopustíme se chyby s pravděpodobností menší než zvolené ? Nebo hypotézu nezamítáme, ale nevíme zda hypotéza platí či jenom nemáme dostatek informací (například počet měření) k zamítnutí hypotézy.

6.2.1 Testovací metoda

Vybranou metodou pro otestování hypotéz je metoda *Testování hypotéz o průměru pro jeden výběr*. Je vhodná v případě, že se u náhodného výběru n lidí měří nějaká spojitá veličina a pokoušíme se určit, že průměr populace je roven očekávané hodnotě.

6.2.2 Testovací kritérium

K otestování nulové hypotézy H_0 proti alternativní hypotéze H_1 je používána statistika Z, která se označuje jako testovací kritérium. Testovací kritérium Z má obecný tvar uveden v rovnici 6.1

$$Z = \frac{P - O}{chyba} \tag{6.1}$$

kde P je pozorovaná hodnota, O očekávaná hodnota, *chyba* značí směrodatnou chybu pozorované hodnoty.

Směrodatná chyba je neznámá. V takovém případě se místo směrodatné chyby používá pouze její odhad $s_{\bar{x}}$, který vypočítáme z našeho výběru. Nyní se již nejedná o testovou statistiku Z, ale o testovou statistiku T, kterou popisuje rovnice 6.2, ve které je očekávaná hodnota značena μ a pozorovanou hodnotou je průměr z naměřených hodnot $\bar{x}$.

$$T = \frac{\bar{x} - \mu}{s_{\bar{x}}} \tag{6.2}$$

Odhad směrodatné chyby $s_{\bar{x}}$ vypočítáme pomocí rovnice ,kde n je počet testujících uživatelů, s je směrodatná odchylka.

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \tag{6.3}$$

Tím, že jsme nahradili směrodatnou chybu jejím odhadem $s_{\bar{x}}$, jsme do výpočtu vnesli ještě další nejistotu, kterou je třeba brát v úvahu. Díky tomu již testová statistika T nemá normální rozdělení, ale rozdělení zvané *Studentovo t rozdělení* o $n - 1$ stupních volnosti.

V případě testování aplikace *Výuka Jazyků* je využita jednostranná alternativa H_1 znázorněná v rovnici 6.3, kde μ_0 je konstanta.

$$H_0\text{: } \mu < \mu_0 \qquad H_1\text{: } \mu > \mu_0 \tag{6.4}$$

Nulovou hypotézu zamítáme, pokud je hodnota testové statistiky větší nebo rovna kritické hodnotě podle rovnice 6.4, kde $t_\alpha(df)$ je kritická hodnota, kterou snadno určíme z tabulky *studentova rozdělení* [3], α je hladina významnosti, df je počet stupňů volnosti $df = n - 1$.

$$T \geq t_\alpha(df) \tag{6.5}$$

Při tvorbě této kapitoly jsem vycházel z [14].

6.2.3 Statistický test

Tato podsekce se věnuje samotnému testování pomocí výpočtů.

Testování tvrzení, že aplikace se snadno používá

Nulová hypotéza H_0 v tomto případě je, že aplikace není intuitivní. Alternativní hypotéza H_1 říká, že *Výuka Jazyků* je snadno použitelná i bez předchozí instruktáže. Zamítnutí H_0 znamená potvrzení dokazovaného tvrzení.

K ověření této hypotézy byl uživatel instruován, aby se pokusil o splnění několika úkonů ve výukové aplikaci, následně v testovacím dotazníku ohodnotil otázky: „Jak úspěšný jste byl v plnění zadaných instrukci?“ a „Myslíte si, že se aplikace snadno používá?“

Hranice úspěšného hodnocení byla stanovena na $\frac{3}{5}$ z nejvyššího hodnocení 10 pro první otázku, na 7 pro otázku druhou. Naměřené hodnoty spolu s hraničními jsou zobrazeny v tabulce.

Význam	**Hodnoty 1. otázka**	**Hodnoty 2. otázka**
Průměrné hodnocení μ	8,16	9
Směrodatná odchylka s	2,3	3,14
Hranice úspěšného hodnocení μ	6	7

Výpočet směrodatné chyby:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \quad s_{\bar{x}1} = \frac{2,3}{\sqrt{19}} = 0,53 \quad s_{\bar{x}2} = \frac{3,14}{\sqrt{19}} = 0,72 \tag{6.6}$$

Vypočítáme statistiky $T1$, $T2$:

$$T = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad T_1 = \frac{8,16 - 6}{0,53} = 4,08 \quad T_2 = \frac{9 - 7}{0,72} = 2,78 \tag{6.7}$$

Kritická hodnota je podle tabulky [3] pro počet respondentů 19 a $\alpha = 0,05$ rovna 1,734. Hodnoty obou statistik $T_1 = 4,23$ a $T_2 = 2,78$ překračují kritickou hodnotu, můžeme tedy zamítnout nulovou hypotézu. Pozitivní hodnocení intuitivnosti aplikace je významné na hladině $p < 0,05$.

Testování hypotézy o atraktivitě UI

Nulová hypotéza H_0 je, že aplikace není pro uživatele atraktivní. Alternativní hypotéza H_1 říká, že uživatelsky atraktivní je.

Ověření této hypotézy jsou v testovacím dotazníku věnovány otázky: „Jak hodnotíte uživatelské rozhraní (vzhled) aplikace?“ a „Je podle vás aplikace přehledná?“

Naměřené hodnoty spolu s hraničními jsou zobrazeny v tabulce.

Význam	**Hodnoty 1. otázka**	**Hodnoty 2. otázka**
Průměrné hodnocení μ	7,16	8,95
Směrodatná odchylka s	2,3	2,74
Hranice úspěšného hodnocení μ	7	7

Výpočet směrodatné chyby:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \quad s_{\bar{x}1} = \frac{2,3}{\sqrt{19}} = 0,53 \quad s_{\bar{x}2} = \frac{2,74}{\sqrt{19}} = 0,63 \tag{6.8}$$

Vypočítáme statistiky $T1$, $T2$:

$$T = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad T_1 = \frac{7,16 - 7}{0,53} = 0,30 \quad T_2 = \frac{8,95 - 7}{0,63} = 3,1 \tag{6.9}$$

Kritická hodnota je podle tabulky [3] pro počet respondentů 19 a $\alpha = 0,05$ rovna 1,734. Hodnota statistiky $T_1 = 0,30$ nedosahuje kritické hodnoty, nemůžeme tedy zamítnout nulovou hypotézu o vzhledu aplikace. Na základě výběru nelze dokázat tvrzení, že je UI atraktivní. Hodnota statistiky $T_2 = 3,1$ přesahuje kritickou hodnotu, můžeme zamítnout alespoň nulovou hypotézu, že aplikace není přehledná. Pozitivní hodnocení přehlednosti aplikace je významné na hladině $p < 0,05$.

Testování hypotézy o výukovém algoritmu

Nulová hypotéza H_0 je, že algoritmus *ARTS* nepředčí algoritmus firmy Langsoft.

K ověření této hypotézy jsou určeny otázky: „Považujete výuku algoritmem *ARTS* za účinnější?“ a „Preferujete *ARTS* pro vlastní výuku?“ Spolu s vysvětlením, že hodnocení 5 znamená nerozhodnost a vyšší hodnocení se přiklání k *ARTS*.

Hranice úspěšného hodnocení byla stanovena na $\frac{1}{2}$ z nejvyššího hodnocení 10 pro obě otázky. Naměřené hodnoty spolu s hraničními jsou zobrazeny v tabulce.

Význam	**Hodnoty 1. otázka**	**Hodnoty 2. otázka**
Průměrné hodnocení μ	6,95	7,84
Směrodatná odchylka s	2,43	1,81
Hranice úspěšného hodnocení μ	5	5

Výpočet směrodatné chyby:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \quad s_{\bar{x}1} = \frac{2,43}{\sqrt{19}} = 0,53 \quad s_{\bar{x}2} = \frac{1,81}{\sqrt{19}} = 0,42 \tag{6.10}$$

Vypočítáme statistiky $T1$, $T2$:

$$T = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad T_1 = \frac{6,95 - 5}{0,53} = 3,68 \quad T_2 = \frac{7,84 - 5}{0,42} = 6,76 \tag{6.11}$$

Kritická hodnota je podle tabulky [3] pro počet respondentů 19 a $\alpha = 0,05$ rovna 1,734. Hodnoty obou statistik $T_1 = 4,23$ a $T_2 = 2,78$ překračují kritickou hodnotu, můžeme tedy zamítnout nulovou hypotézu. Pozitivní hodnocení intuitivnosti aplikace je významné na hladině $p < 0,05$.

Testování hypotézy o kladném přijetí aplikace

Nulová hypotéza H_0 v tomto případě je, že aplikace není uživateli přijímána pozitivně. Alternativní hypotéza H_1 říká, že *Výuka Jazyků* je přijímána uživateli pozitivně. Zamítnutí H_0 znamená potvrzení dokazovaného tvrzení.

K ověření této hypotézy uživatel ohodnotil otázky: „Myslíte si, že má aplikace na to stát se užitečným nástrojem pro učení frází či vět?“ a „Jak by jste ohodnotil aplikaci jako celek?“

Naměřené hodnoty spolu s hraničními jsou zobrazeny v tabulce.

Význam	**Hodnoty 1. otázka**	**Hodnoty 2. otázka**
Průměrné hodnocení μ	7,47	7,11
Směrodatná odchylka s	2,26	2,17
Hranice úspěšného hodnocení μ	6	6

Výpočet směrodatné chyby:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \quad s_{\bar{x}1} = \frac{2,26}{\sqrt{19}} = 0,52 \quad s_{\bar{x}2} = \frac{2,17}{\sqrt{19}} = 0,5 \tag{6.12}$$

Vypočítáme statistiky $T1$, $T2$:

$$T = \frac{\bar{x} - \mu}{s_{\bar{x}}} \quad T_1 = \frac{7,47 - 6}{0,52} = 2,83 \quad T_2 = \frac{7,11 - 6}{0,5} = 2,22 \tag{6.13}$$

Kritická hodnota je podle tabulky [3] pro počet respondentů 19 a $\alpha = 0,05$ rovna 1,734. Hodnoty obou statistik $T_1 = 2,83$ a $T_2 = 2,22$ překračují kritickou hodnotu, můžeme tedy zamítnout nulovou hypotézu. Pozitivní hodnocení kladného přijetí aplikace je významné na hladině $p < 0,05$.

6.2.4 Výsledky testování

U 7 statistických testů se nám podařilo zamítnout nulovou hypotézu, čímž byla dokázána alternativní hypotéza významná na hladině $p < 0,05$. Tímto byla dokázána 3 ověřovaná tvrzení. Nepodařilo se dokázat, že UI aplikace je pro uživatele atraktivní, pouze to, že je přehledné. Výsledky jsou uvedeny v tabulce.

Tvrzení	**Potvrzení**
Aplikace se snadno používá, je intuitivní.	Ano
Uživatelské rozhraní je pro uživatele atraktivní.	Ne
Algoritmus *ARTS* předčí jednodušší algoritmus firmy *Langsoft*.	Ano
Aplikace je uživateli přijímána pozitivně.	Ano

Závěr

Hlavním cílem této práce bylo podle zadání navrhnout a vytvořit výukovou aplikaci na procvičování cizího jazyka s efektivním výukovým algoritmem. Popsal jsem způsoby, kterými lze efektivitu výuky zvýšit a dosáhnout tak lepšího pamatování učené látky při menších nárocích na čas studenta.

V cestě za tímto cílem bylo nejprve nutné nastudovat dosud objevené poznatky z psychologie učení, díky kterým jsou dnešní výukové aplikace tak účinné. Analyzoval jsem existující populární výukové nástroje.

Zachytil a sumarizoval jsem požadavky na výukový systém. Těmto požadavkům jsem se pokusil vyhovět při návrhu a implementaci vlastního řešení. Výsledkem práce je aplikace na platformě iOS.

Po implementaci byla aplikace otestována 19 uživateli a podařilo se odstranit několik drobných chyb. Pro testování aplikace byla použita metoda případová studie. Byla stanovena čtyři tvrzení o aplikaci, která byla následně testována metodou *Testování hypotéz o průměru pro jeden výběr*. Tři tvrzení byla dokázána jako pravdivá. Celkově byla aplikace hodnocena uživateli pozitivně. Aplikace byla publikována do obchodu *App Store* firmy Apple pod názvem *Angličtina - Konverzace*, kde se setkala s kladnou odezvou uživatelů.

Pokračování projektu může vést několika směry:

- Přidání podpory pro kartičky s multimediálním obsahem (obrázky, video, apod.).
- Přidání podpory pro vytváření uživatelských lekcí, aby uživatelé mohli sami vytvářet vyučovaný materiál.
- Zvýšení atraktivity procesu učení.

Literatura

[1] Concepts in Objective-C Programming. [online]. 2012 [cit. 2014-10-19].
URL https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html

[2] Concepts in Objective-C Programming. [online]. 2012 [cit. 2014-10-12].
URL https://developer.apple.com/library/mac/documentation/General/Conceptual/CocoaEncyclopedia/DelegatesandDataSources/DelegatesandDataSources.html#//apple_ref/doc/uid/TP40010810-CH11-SW1

[3] Statistické tabulky. [online]. 2012 [cit. 2015-2-13].
URL http://cit.vfu.cz/statpotr/POTR/Teorie/tabulky.htm#ttest

[4] Super Memory: Forget about forgetting. [online]. 1997 [cit. 2014-11-1].
URL http://www.supermemo.com/english/algsm11.htm

[5] View Controller Programming Guide for iOS. [online]. 2012 [cit. 2014-10-12].
URL https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/Introduction/Introduction.html

[6] Wikipedia. [online]. 2001 [cit. 2014-11-14].
URL http://en.wikipedia.org/wiki/Flashcard

[7] XCode. [online]. 2015 [cit. 2015-1-18].
URL https://developer.apple.com/xcode/interface-builder/

[8] de Boer, V.: Optimal Learning and the Spacing Effect. 2003, [Diplomová práce].

[9] Cepeda, N. J.; Vul, E.; Rohrer, D.; aj.: Spacing Effects in Learning. *Psychological Science*, ročník vol. 19, č. issue 11, 2008: s. 65–78.

[10] Clegg, B.: *Procvičování paměti a kreativity*. Brno: CP Books, vyd. 1. vydání, 2005, ISBN 80-251-0548-2.

[11] H., E.: *Memory: A Contribution to Experimental Psychology*. Columbia University. Teachers College. Educational reprints. no. 3, Teachers College, Columbia University, 1913, [online]. 2006 [cit. 2014-10-15].
URL http://books.google.cz/books?id=oRSMDF6y3l8C

[12] Kochan, S. G.: *Programming in Objective-C*. Indianapolis, IN: Addison-Wesley Professional, Čtvrté vydání, 2012, ISBN ISBN 978-0-321-81190-5.

[13] Meier, R.: *Professional Android application development*. Indianapolis, IN: Wiley, 2009, ISBN 978-047-0344-712.

[14] Meloun, M.; Militký, J.: *Interaktivní statistická analýza dat*. Praha: Karolinum, vyd. 3., v nakl. karolinum 1. vydání, 2012, ISBN 978-80-246-2173-9.

[15] Murphy, M. L.: *Android 2*. Brno: Computer Press, vyd. 1. vydání, 2011, ISBN 978-80-251-3194-7.

[16] N, L. G.: *Windows Phone 7.5 application development with F?* Birmingham, UK: Packt, new edition. vydání, 2013, ISBN 978-184-9687-843.

[17] Napier, R.; and, M. K.: *Pushing the limits with iOS 5 programming*. Hoboken, N.J.: Wiley, druhé vydání, 2012, ISBN ISBN 978-1-119-96158-1.

[18] Roediger, H. L.; Karpicke, J. D.: Test-Enhanced Learning. *Psychological Science*, ročník vol. 17, č. issue 3, 2006-03-01: s. 249–255.
URL http://pss.sagepub.com/lookup/doi/10.1111/j.1467-9280.2006.01693.x

[19] Rubin, D. C.; Wenzel, A. E.: One hundred years of forgetting. *Psychological Review*, ročník vol. 103, č. issue 4, 1996: s. 734–760.

[20] Čačka, O.: *Psychologie vrstev duševního dění osobnosti a jejich autodiagnostika*. Brno: Doplněk, vyd. 1. vydání, 1997, ISBN 80-857-6570-5.

[21] Čáp, J.: *Psychologie pro učitele*. Praha: Portál, vyd. 1. vydání, 2001, ISBN 80-717-8463-X.

Příloha A

Obsah CD

- sources - Zdrojové soubory aplikace.
- thesis - Text práce a jeho zdrojové soubory.
- readme.pdf - Návod ke spuštění.

Příloha B

Testovací dotazník

Před Vámi je dotazník, jenž je součástí bakalářské práce realizující výukovou aplikaci cizích frází a slov pro mobilní zařízení se systémem iOS. Dotazník je jedním z nástrojů určených pro zjištění úspěšnosti výsledku projektu. Děkuji Vám za jeho vyplnění. Prohlašuji, že tento dotazník je anonymní a bude využit pouze ke zpracování výsledku mé bakalářské práce.

Nejprve spusťte verzi pojmenovanou podle výukového algoritmu ARTS a pokuste se splnit těchto 6 instrukcí:

1. Spusťte výuku výběrem překladu a zvolte překlad věty, který považujete za správný.
2. Vyberte k výuce poslední lekci (30. lekce).
3. Obraťte směr výuky.
4. Přimějte aplikaci aby opakovaně přečetla výslovnost.
5. Přejděte do výuky skládáním vět a složte větu.
6. Pokuste se vynulovat znalost současné lekce.

po té prosím vyplňte dotazník děkuji.

Pohlaví

Muž
Žena

Věková skupina

0 - 18
19 - 25
26 - 30
30 - 46
46+

Používáte výukové aplikace ? (libovolné množství odpovědí)

Mám s výukovými aplikacemi zkušenosti
Používám
Nepoužívám
Používal bych kdyby aplikace učení zrychlila
Na mobilním zařízení
Na stolním pc
Na notebooku

Snadnost použití aplikace

Jako odpověď uživatel zadá k otázkám 1 až 10 hvězd.
Jak úspěšný jste byl v plnění zadaných instrukci?
Myslíte si, že se aplikace snadno používá?

Uživatelské rozhraní (menu, nastavení,..)

Jako odpověď uživatel zadá k otázkám 1 až 10 hvězd.
Jak hodnotíte uživatelské rozhraní (vzhled) aplikace?
Je podle vás aplikace přehledná?

Výukový algoritmus

Jako odpověď uživatel zadá k otázkám 1 až 10 hvězd.
Tato otázka se zabývá rozdíly v učení obou verzí aplikace, proto prosím zkuste před zodpovězením dostatečně porovnat příjemnost a účinnost výuky obou algoritmů. (5 hvězdiček znamená nerozhodnost mezi verzemi)
Považujete výuku algoritmem ARTS za účinnější?
Preferujete ARTS pro vlastní výuku?

Celkové hodnocení

Jako odpověď uživatel zadá k otázkám 1 až 10 hvězd.
Myslíte si, že má aplikace na to stát se užitečným nástrojem pro učení frází či vět?
Jak by jste ohodnotil aplikaci jako celek?

Otázky k vyučovanému materiálu v podobě 30 lekcí

Jako odpověď uživatel zadá k otázkám 1 až 10 hvězd.
Jak hodnotíte vyučovaný materiál?
Uvítal by jste možnost tvorby vlastních lekcí?