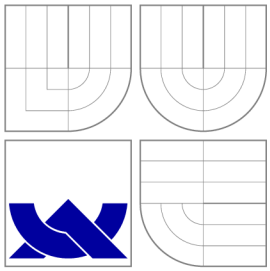
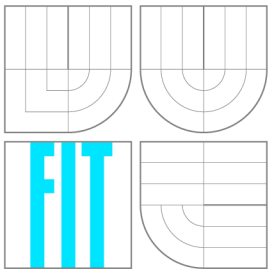


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DOTYKOVÉ UŽIVATELSKÉ ROZHRANÍ PRO EDITOR ZDROJOVÝCH KÓDŮ

TOUCH FRIENDLY USER INTERFACE FOR SOURCE CODE EDITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TEREZA HONZÁTKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL NAJMAN

BRNO 2014

Abstrakt

Tato bakalářská práce popisuje návrh, implementaci a testování alternativního uživatelského rozhraní editoru zdrojového kódu jazyka C na dotykové zařízení. Cílem práce je ulehčit a zpříjemnit uživatelům práci se zdrojovým kódem na dotykových zařízeních. V navrženém rozhraní je kód rozdělen do bloků, s nimiž lze operovat gestem drag&drop a nahrazují nepohodlnou softwarovou klávesnici.

Abstract

This paper describes a draft, an implementation and a testing mechanism of an alternative user interface of a C language's source code editor for touch devices. The aim of this interface is to facilitate users' work with the source code for touch devices. The resulting interface application is divided into blocks, which can be operated by the end user through drag&drop principle. The interface replaces uncomfortable software keyboard.

Klíčová slova

uživatelské rozhraní, Android, editor, drag&drop, GUI

Keywords

user interface, Android, editor, drag&drop, GUI

Citace

Tereza Honzátková: Dotykové uživatelské rozhraní pro editor zdrojových kódů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Dotykové uživatelské rozhraní pro editor zdrojových kódů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Pavla Najmana

.....
Tereza Honzátková
18. května 2014

Poděkování

Ráda bych poděkovala Ing. Pavlu Najmanovi za odborné vedení a konzultace.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V současné době existuje spousta editorů zdrojových kódů na dotyková zařízení. Tyto editory se liší v množství nejrůznějších funkcí poskytovaných uživateli, grafickém rozhraní, výčtu podporovaných programovacích jazyků, ale jedno mají společné - softwarovou klávesnici, která zabírá mnohdy větší část obrazovky než samotný kód. Psaní na těchto klávesnicích je pomalé, nepohodlné a dochází k častým překlepům [7], proto můžeme najít nejrůznější alternativní klávesnice např. Swype, SwiftKey, SlideIT slibující rychlejší psaní textu [3]. Tyto klávesnice jsou však navrženy pro běžné psaní textu a neřeší tak složité psaní speciálních znaků, které se v programování využívají v mnohem větší míře.

Cílem této práce je vytvořit a následně vyhodnotit alternativní uživatelské rozhraní, pomocí kterého se bude vytvářet, editovat a ukládat zdrojový kód s minimálním použitím softwarové klávesnice a tím usnadnit uživatelům práci s kódem na dotykových zařízeních. Toho je docíleno vytvořením bloků, které reprezentují určité sekvence kódu a lze s nimi manipulovat za pomocí gest drag&drop.

Aby bylo možné rozhraní otestovat a vyhodnotit, vytvořila jsem aplikaci s tímto rozhraním, pomocí které lze vytvořit jednoduchý zdrojový kód jazyka C. Jelikož jde v této práci primárně o uživatelské rozhraní, jsou zde implementovány pouze základní prvky jazyka.

V první kapitole se dočteme něco málo o uživatelských rozhraních, jejich typech a pravidlech použitelnosti. Podrobněji je rozebráno dotykové uživatelské rozhraní a na závěr kapitoly jsou uvedeny a popsány existující řešení aplikací pro tvorbu programů bez využití klávesnice. Druhá kapitola popisuje návrh uživatelského rozhraní pro editor. Kapitola třetí se zabývá implementací navrženého uživatelského rozhraní. Ve čtvrté kapitole je popsán způsob testování výsledné aplikace s vytvořeným rozhraním. Dále jsou zde uvedeny výsledky z testování a nápady na možnost budoucího vývoje a pokračování v bakalářské práci. V závěrečné kapitole je shrnuta práce na celém projektu.

Kapitola 2

Uživatelská rozhraní

Tato kapitola popisuje a vysvětluje pojem uživatelské rozhraní, zmiňuje jeho typy, pravidla použitelnosti uživatelských rozhraní a problematiku uživatelských rozhraní na dotykových zařízeních. Na konci kapitoly se dočteme o již existujících řešeních problému se softwarovou klávesnicí Scratch a TouchDevelop.

2.1 Typy uživatelských rozhraní

Uživatelské rozhraní je především komunikace s uživatelem. Uživatel přes rozhraní ovládá aplikaci, program či systém a je proto nutné, aby bylo logické, intuitivní a snadné na ovládání. V dnešní době existuje několik typů uživatelských rozhraní s rozlišnými vlastnostmi, náročností a způsobu zadávání vstupu a každé je tedy vhodné pro jiné zařízení. Jednotlivými typy uživatelských rozhraní jsou:

Rozhraní příkazové řádky (CLI)

Rozhraní příkazové řádky využívá ke komunikaci s uživatelem zadávání příkazů pomocí klávesnice ze strany uživatele, a tisk vstupu na monitor ze strany zařízení. Nevýhodou tohoto rozhraní je velká složitost a náročnost na naučení se příkazů daného jazyka, proto jej využívají především odborníci, administrátoři a technici.

Textové uživatelské rozhraní (TUI/CUI)

Textové uživatelské rozhraní pracuje v textovém režimu a je jistým mezistupněm rozhraní příkazové řádky a grafického uživatelského rozhraní. Obrazovka je rozdělena na sloupce a řádky, přičemž každá pozice umožňuje zobrazit jediný znak. Ze speciálních znaků jsou vytvořeny i jednotlivé ovládací prvky jako posuvníky, rámečky, okna, připomínající grafické uživatelské rozhraní.

Grafické uživatelské rozhraní (GUI)

Toto uživatelské rozhraní umožňuje ovládat zařízení pomocí interaktivních grafických prvků. Uživatel ke komunikaci s tímto rozhraním potřebuje určité hardwarové prostředky, jako je myš, klávesnice, tlačítka. Tyto prostředky slouží k zadávání vstupu a k manipulaci s kurzorem. Teprve pomocí kurzoru uživatel manipuluje s grafickými prvky. Toto rozhraní je snadné na ovládání a umožnilo tak práci s počítačem uživatelům, kteří nejsou odborníci.

Najdeme jej v dnešní době téměř všude – v počítačích, autech, domácích spotřebičích, přehrávačích atd.

Dotykové uživatelské rozhraní (TUI)

Tento typ rozhraní se na první pohled vizuálně příliš neliší od grafického uživatelského rozhraní. Nenachází se zde ale žádný kurzor k manipulaci s prvky, ke komunikaci tedy nepotřebujeme žádný další hardware. Veškeré ovládání lze provádět pomocí dotyků konečků prstů. Samozřejmě můžeme použít i hardwarovou klávesnici, myš či stylus, ale není to nezbytné. Rozhraní se vyskytuje v zařízeních se speciálními dotykovými obrazovkami, jako jsou např. monitory počítačů, chytré telefony, čtečky knih, tablety, informační stojany a mnoho dalších. Toto rozhraní pomalu nahrazuje grafické uživatelské rozhraní díky svému intuitivnímu ovládání.

2.2 Pravidla použitelnosti uživatelských rozhraní

Aby byla navrhována použitelná a kvalitní uživatelská rozhraní, vznikla pravidla, kterými by se každý návrhář rozhraní měl řídit. Uvedu zde **Osm zlatých pravidel designu rozhraní** [11], která shrnují nejdůležitější body návrhu, na které se zaměřit. Těmito pravidly jsou:

1. **Usilovat o konzistenci** - Konzistencí zde rozumíme například stejnou terminologii v menu, výzvách a nápovědách. Jestliže jeden prvek nazveme odlišně v nápovědě a v samotném rozhraní, uživatele tím zmateme a nebude si jist, o jaký prvek se jedná. Dále by měly být v celém rozhraní jednotné fonty, aby bylo rozhraní přehledné. Důležité je také stejné umístění hlavních prvků na jednotlivých layoutech, jako jsou například dialogová tlačítka. To vše je důležité pro snadnou orientaci uživatele v rozhraní.

2. **Zaměřit se na univerzálního uživatele** - Každé rozhraní využívá spousta uživatelů rozdílných věkových skupin s odlišnými znalostmi a zájmy. Rozhraní by tedy mělo být navrženo tak, aby jej mohl využívat odborník v dané oblasti, ale také běžný uživatel, student či dítě. To je však mnohdy velmi náročné, jelikož každý z těchto uživatelů má odlišné nároky a návrhář pak často musí zvolit, které skupiny jsou pro něj primární. Vždy je však vhodnější zaměřit se na uživatele s menšími zkušenostmi a podle toho rozhraní navrhovat, aby bylo snadné na použití a nebylo nutné k ovládání vytvářet manuál.

3. **Nabídnout zpětnou vazbu** - Aby měl uživatel důvěru v používané rozhraní, měla by mu být nabídnuta možnost zpětné vazby. Časté dotazy lze snadno vyřešit importováním nápovědy do systému s odpověďmi právě na tyto dotazy. Pro méně časté dotazy či větší problémy je nutné poskytnout podporu například pomocí fóra, kam uživatel může psát jaké má problémy s daným systémem/rozhraním, vznášet nejrůznější dotazy a kde mu bude co nejdříve odpovězeno.

4. **Navigovat uživatele v dialozích** - Má-li rozhraní více dialogů, které spolu souvisí, měl by mít uživatel přehled o tom, v jaké části se právě nachází. Jako příklad uvedu objednávku zboží na internetových stránkách, kde je dialog rozdělen do skupin – dodací adresa, doprava, platba. Uživatel je tedy proveden dialogem od začátku až do konce, má přehled, kde se nachází a nemůže se v dialozích ztratit.

5. **Předcházet/zabránit chybám** - Rozhraní musí být navrženo tak, aby uživatel nemohl udělat chybu nebo nevratnou akci. Omezení takových stavů lze dosáhnout například použitím selekce předdefinovaných vstupů místo jejich vlastního zadávání, při kterém mohou vzniknout překlepy a chyby. Dále je nutné implementovat nejrůznější kontroly, aby uživatel nemohl zadat nesmyslné kombinace. Důležité jsou také dialogové zprávy o chybách, aby uživateli bylo jasné, že dělá něco špatně a mohl na tuto situaci reagovat.

6. **Umožnit snadný návrat/zrušení akcí** - Jestliže uživateli nabídneme snadné vracení provedených akcí, nebude mít strach experimentovat a rozhraní tak naplno využívat, jelikož ví, že udělá-li něco jinak než chtěl, není problém vrátit se zpět do pozice, v které začal. Možnost vracet se v krocích zpět tak zmírňuje úzkost uživatelů z jejich možných chyb a dává jim možnost prozkoumávat rozhraní a jeho chování i v jeho méně využívaných funkcích.

7. **Učinit systém předvídatelným** - Aby bylo rozhraní použitelné, mělo by být navrženo tak, aby měl uživatel dojem, že má jeho chování zcela pod kontrolou a ne naopak, že je ovládán. Měli bychom tedy uživatele ušetřit zdlouhavého zadávání vstupu, složitého nastavení rozhraní a především překvapení z chování rozhraní. Rozhraní by mělo být předvídatelné, čehož můžeme docílit tak, že tlačítka budou vypadat jako tlačítka, odkazy budou podtrženy a zvýrazněny, tak jak je uživatel zvyklý z jiných systémů, jednotlivé skupiny menu budou mít značku označující rozbalení/sbalení položek atd.

8. **Snížit krátkodobé zatížení paměti** - Jelikož lidská paměť je limitována, jednotlivé layouty by měly být jednoduché a nemělo by po uživateli být vyžadováno, aby si pamatoval věci z jiné stránky, než na které se nachází. V rozhraní by nemělo být moc vrstev, do kterých by se uživatel mohl zanořit a ztratit tak pojem o místě, na kterém se právě nachází. Jestliže má rozhraní více oken, měla by být ukotvena, aby o nich měl uživatel přehled.

Zmíněná pravidla nezabraňují všem problémům s rozhraním, pouze shrnují nejdůležitější body, na které bychom se měli při návrhu rozhraní zaměřit. Slouží tak jako dobrý základ pro designéry všech rozhraní.

2.3 Problematika dotykových uživatelských rozhraní

Jak již bylo zmíněno ve výčtu typů uživatelských rozhraní, dotykové rozhraní je nástupcem grafického rozhraním a má své výhody i nevýhody, které je potřeba zohlednit při návrhu. V následujícím textu jsou zmíněny hlavní z těchto vlastností.

Dotyková zařízení mohou mít různé dotykové obrazovky založené na odlišných technologiích. Těmito technologiemi jsou:

- rezistivní
- kapacitní
- projekční kapacitní
- infračervené záření
- povrchová akustická vlna

U dotykových uživatelských rozhraní je důležité brát v potaz to, že dotyková zařízení mají omezené vlastnosti v zadávání uživatelského vstupu. K zadávání vstupu uživatelé nejčastěji používají dotyky konečků prstů a tomu musí být rozhraní přizpůsobeno. Ovládání prsty není, vzhledem k jejich velké ploše, tak přesné jako myš, a uživatel tak nedokáže přesně vybrat malé prvky. Nepomáhá tomu ani fakt, že při dotyku si právě prstem zakryjeme ovládací prvek, který chceme vybrat, pokud tento prvek není dostatečně velký. Ovládání prsty má však i své výhody – je pro uživatele více kontaktní a intuitivní, jelikož s objekty manipulují přímo pomocí gest, aniž by se museli potýkat s myší, touchpadem nebo jiným hardwarovým prostředkem. Další výhodou dotykového zařízení je podpora vícedotykového ovládání, které dále najdeme pouze u některých touchpadů a které dává uživateli nové možnosti ovládání, například při rotaci a přiblížení. Shrnutí rozdílů mezi vstupními zařízeními znázorňuje tabulka 2.1.

Dalším problémem dotykových zařízení je velikost a rozlišení dotykové obrazovky, kterých je velké množství. Při návrhu musíme brát v úvahu, pro jaký typ zařízení (chytrý telefon, tablet, dotykové monitory, informační panely atd.) rozhraní navrhujeme. Je důležité, aby se rozhraní zobrazovalo na jakémkoli rozměru obrazovky, které může daný typ zařízení mít, správně.

Podstatnou roli při tvorbě dotykových uživatelských rozhraních hraje také to, jak uživatelé dotykové zařízení drží a jak jsou rozmístěna tlačítka, aby stále nedocházelo k jejich náhodnému stlačení. Některá dotyková zařízení mají tyto tlačítka řešena hardwarově, jiná softwarově. Hardwarová tlačítka jsou na každém zařízení na jiném místě a bývají náhodně stlačována právě při držení zařízení. Proto je důležité poskytnout uživateli možnost volby v orientaci rozhraní. Softwarová tlačítka činí problém při ovládání, jsou-li ovládací prvky malé a příliš blízko panelu s tlačítky. [4]

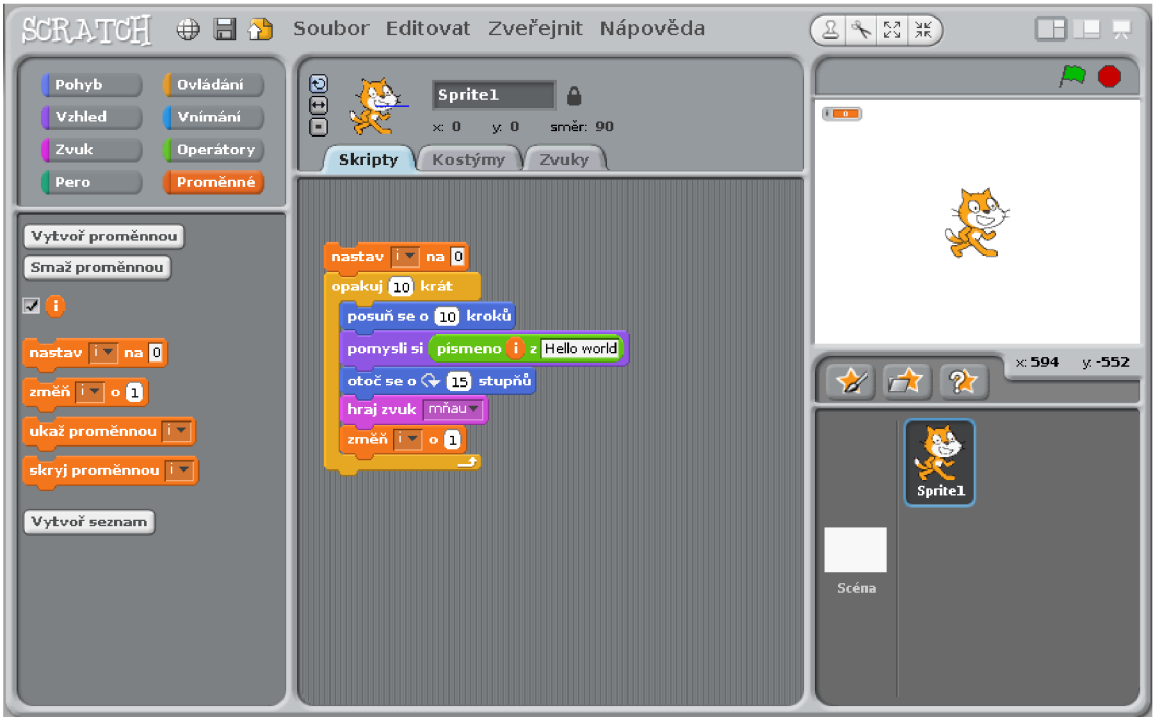
2.4 Existující řešení

Prozkoumáním existujících řešení problému, lze získat důležité informace pro následné navržení a vytvoření nového a lepšího uživatelského rozhraní. Důležité jsou především nedostatky existujících rozhraní, které by se v navrženém rozhraní již neměly vyskytovat. Vzhledem k tomu, že jsem nenašla mnoho aplikací na dotykové zařízení, které by nevyužívaly SW klávesnici, rozšířila jsem oblast hledání editoru také o desktopové verze. Uvedu zde dvě řešení problému – program Scratch [10] a aplikaci TouchDevelop [2].

Scratch

Na obrázku 2.1 je vidět uživatelské rozhraní programu Scratch, z kterého jsem při prvotním návrhu vycházela. Scratch slouží, díky jednoduchosti jeho užívání, převážně jako výukový program pro děti ve věku 8 až 16 let a pro uživatele bez předchozích zkušeností s programováním. Primárním cílem Scratch není vytvářet z lidí profesionální programátory, ale ukázat programování jako nástroj pro vyjádření myšlenek a vyvrátit tak domněnku, že programování je činnost určená pouze pro malou část společnosti. Z programu Scratch vychází spousta dalších programů jako např. BYOB (Build Your Own Blocks), který uživateli umožňuje vytvořit si své vlastní bloky s příkazy.

Princip Scratch je založen na grafických blocích, které lze různě skládat dohromady a tím vytvořit fungující program. Tyto bloky jsou umístěny v panelu na levé straně obrazovky a jsou rozděleny do osmi barevně odlišených skupin. Z těchto skupin lze bloky přetahovat a umisťovat do skriptovací části obrazovky. Ovládání Scratch je velice snadné a intuitivní, bloky kódu lze snadno nalézt díky logickému rozdělení do skupin. Kromě samotné tvorby kódu Scratch poskytuje spoustu dalších nástrojů jako například malování a nahrávání zvuků, které lze rovnou použít. Vytvořený kód lze přímo v programu spouštět a jeho průběh uživatel vidí na plátně, které je umístěno v pravém horním rohu. Dále lze kód snadno zveřejnit a sdílet s ostatními uživateli.

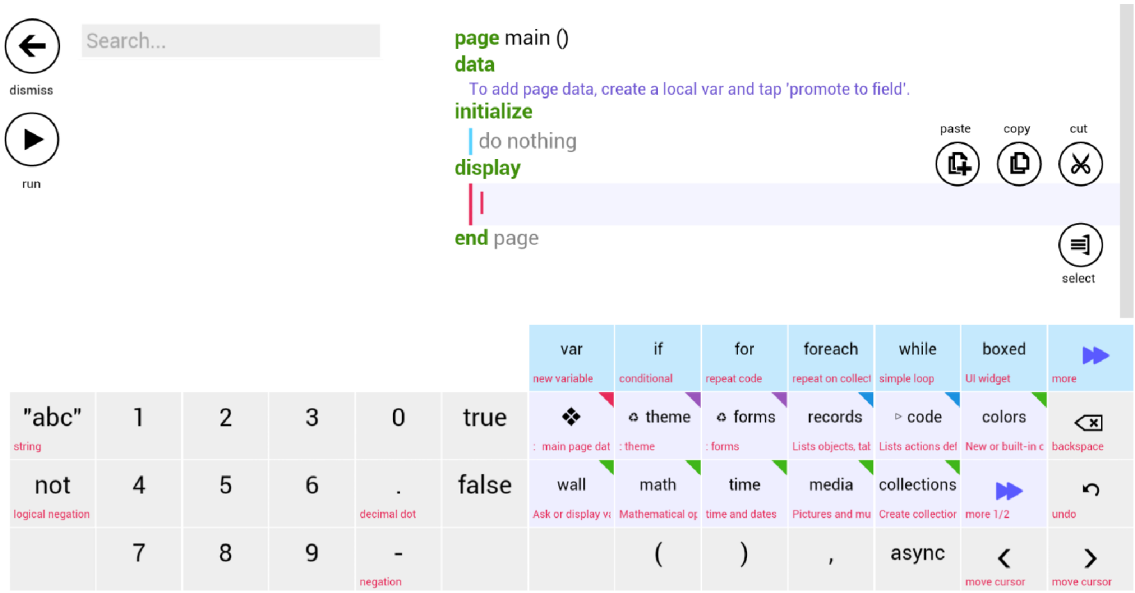


Obrázek 2.1: Program Scratch.

TouchDevelop

Na obrázku 2.2 je ukázka aplikace TouchDevelop, která je vyvíjená primárně na platformu Windows Phone, což se odráží i na jejím grafickém uživatelském rozhraní. Na platformách iOS, Android, PC, Mac a Linux aplikace pouze poskytuje notifikaci a spouští offline verzi aplikace ve webovém prohlížeči, kde probíhá samotná práce s kódem. Aplikace TouchDevelop vychází z programu Scratch a z vize zadávání celého kódu pomocí dotykového displeje, tedy bez klávesnice. Za aplikací TouchDevelop stojí také myšlenka vyvíjet software bez nutnosti použít PC a umí tedy vytvořený kód i překládat a spouštět. Vytvořené skripty se automaticky synchronizují a lze je tak snadno sdílet s jinými uživateli, hodnotit a komentovat.

Rozhraní aplikace TouchDevelop je založeno na opačném principu než Scratch. Nepřemisťujeme zde jednotlivé bloky z nabídky na určité místo, ale dotykem na místo se nám vytvoří nabídka, z které si můžeme dalším dotykem blok vybrat, a on se sám vloží na danou pozici. Toto ovládání je o poznání složitější než ovládání programu Scratch a to díky měnící se nabídce a tedy i poloze jednotlivých bloků. Jednotlivé bloky jsou rozlišeny podle funkce pouze barevným trojúhelníkem v rohu každého bloku, jsou rozděleny a umístěny do jednotlivých skupin. V některých nabídkách se však vyskytne příliš velké množství bloků, které nelze zobrazit najednou a uživatel musí nabídkou neustále listovat. To je nepřehledné a také tím dochází ke značnému zdržení při tvorbě kódu. Další nevýhodou jsou ikony *paste*, *copy*, *cut* a *select*, které jsou zobrazovány v místě, kam chceme vložit nějaký blok, a které celý editor činí nepřehledným.



Obrázek 2.2: Aplikace TouchDevelop.

Faktor	Interakce pomocí dotyků	Interakce myši, klávesnice, pera nebo stylusu
Přesnost	Kontaktní plocha prstů je větší než jedna souřadnice x-y, což zvyšuje riziko nechtěné aktivace příkazů.	Myš a pero nebo stylus zajišťují přesné souřadnice x-y.
	Tvar kontaktní plochy se při pohybu mění.	Pohyby myši a tahy pera nebo stylusu zajišťují přesné souřadnice x-y. Fokus klávesnice je explicitní.
	Není k dispozici žádný kurzor myši, který by pomáhal při cílení.	Kurzor myši, pera nebo stylusu a fokus klávesnice pomáhají při cílení.
Lidská anatomie	Pohyby prstů jsou nepřesné, protože je obtížné pohybovat se jedním i několika prsty po přímce. Důvodem je zakřivení kloubů ruky a počet kloubů zapojených do pohybu.	Pohyb po přímce je snazší provádět myší, perem nebo stylusem, protože ruka, která je ovládá, urazí kratší fyzickou vzdálenost než kurzor na obrazovce.
	Některé oblasti na dotykové ploše zobrazovacího zařízení jsou těžko dosažitelné z důvodu polohy prstu a úchopu, kterým uživatel zařízení drží.	S myší, perem nebo stylusem se dá dostat do libovolné části obrazovky a libovolný ovládací prvek je dostupný pomocí klávesnice v definovaném pořadí.
	Objekty můžou být zakryté jedním nebo víc prsty nebo rukou uživatele. Tomu se říká zakrytí.	Nepřímá vstupní zařízení zakrytí nezpůsobují.
Stav objektu	Dotykové ovládání využívá dvoustavový model: na dotykovém displeji byl buďto proveden dotyk (zapnuto) nebo ne (vypnuto). Neexistuje žádný stav umístění ukazatele.	Myš, pero nebo stylus a klávesnice všechny využívají třístavový model: nahoru (vypnuto), dolů (zapnuto) a umístění ukazatele (fokus).
Bohatá interakce	Podporuje vícedotykové ovládání: víc vstupních bodů (dotyků prsty) na dotykovém povrchu.	Podporuje jeden vstupní bod.
	Podporuje přímou manipulaci s objekty prostřednictvím gest, jako je klepnutí, tažení, posunutí, sevření nebo rozevření prstů a otáčení.	Žádná podpora pro přímou manipulaci, protože myš, pero nebo stylus a klávesnice jsou nepřímá vstupní zařízení.

Tabulka 2.1: Rozdíly mezi vstupními zařízeními.
Zdroj: http://msdn.microsoft.com

Kapitola 3

Návrh řešení

Při návrhu rozhraní jsem brala ohled na výše zmíněných osm zlatých pravidel tvorby uživatelských rozhraní, která jsem se snažila co nejlépe splnit. Ještě před návrhem jsem však provedla průzkum ohledně používání editačních nástrojů zdrojových kódů na dotykových zařízeních, abych získala přehled o názoru uživatelů a největších překážkách při programování na dotykových zařízeních.

3.1 Editace zdrojových kódu na dotykových zařízeních

Z průzkumu provedeného před samotným návrhem uživatelského rozhraní jsem získala informace ohledně používání editorů zdrojových kódů na dotykových zařízeních. Průzkumu se zúčastnilo 107 uživatelů, kteří programují a zároveň vlastní dotykové zařízení.

Z výsledků vyplývá, že pouze 8% programátorů vlastnících dotykové zařízení ho využívá k tvorbě či editaci kódu a to pouze zřídka. Nejčastěji uváděné důvody, proč uživatelé takové zařízení nevyužívají, byly následující:

- Notebook/počítač s hardwarovou klávesnicí je efektivnější.
- Nevidím důvod dotykové zařízení tímto způsobem využít.
- Psaní na SW klávesnici je nepohodlné.
- Psaní na SW klávesnici je příliš pomalé.
- SW klávesnice nemá žádnou zpětnou vazbu v podobě stisknuté klávesnice.
- Příliš malé obrazovky a příliš velké SW klávesnice.
- Vzniká spousta překlepů.
- Složité psaní speciálních znaků.
- Nelze využít klávesové zkratky.

Z odpovědí uživatelů, kteří využívají dotyková zařízení k programování, stojí za zmínku fakt, že pouze třetina těchto uživatelů využívá speciální editory navržené k programování. Přitom tyto editory mají spoustu funkcí usnadňující práci s kódem, jako například zvýraznění syntaxe a lepší umístění speciálních znaků. Další zajímavou informací je skutečnost, že funkci pro kopírování, vyjmutí a vkládání textu aktivně využívá pouze 44% těchto

uživatelů a to z důvodu složité práce s kurzorem v editorech. Tento kurzor se využívá také při editaci kódu, kdy se musíte trefit mezi jednotlivé znaky, což činí problém celým dvěma třetinám uživatelů. Přitom je editace kódu častou akcí, jelikož v editorech se softwarovou klávesnicí vzniká spousta překlepů. Jak vyplývá z výsledků průzkumu, časté nebo velmi časté překlepy má celých 78% uživatelů.

Všechny tyto informace jsou důležité z hlediska návrhu rozhraní. Ačkoli se pravděpodobně nepovede odstranit všechny výše zmíněné nedostatky a důvody k nevyužití dotykových zařízení, vím, jaké vlastnosti jsou pro uživatele nejdůležitější, a na ty se mohu zaměřit, a které funkce naopak nevyužívají.

3.2 Myšlenka navrženého uživatelského rozhraní

Myšlenkou uživatelských rozhraní je vytvořit takové prostředí, které bude intuitivní a bude se s ním lehce pracovat. Při návrhu rozhraní jsem vycházela z výše zmíněných existujících řešení, v nichž není nutné použít klávesnici pro tvorbu kódu - aplikace TouchDevelop primárně vytvořené pro dotyková zařízení s operačním systémem Windows Phone a program Scratch. Ačkoli je program Scratch určen pro desktopové zařízení a nebyl tedy vytvořen jako řešení problému se softwarovou klávesnicí, lze z něj také vycházet, jelikož funguje na principu snadného tvoření programů skládáním kusů kódu v podobě bloků do celku přetahováním pomocí myši. Dále jsem vycházela z výsledků průzkumu, které jsou uvedeny v kapitole 3.1.

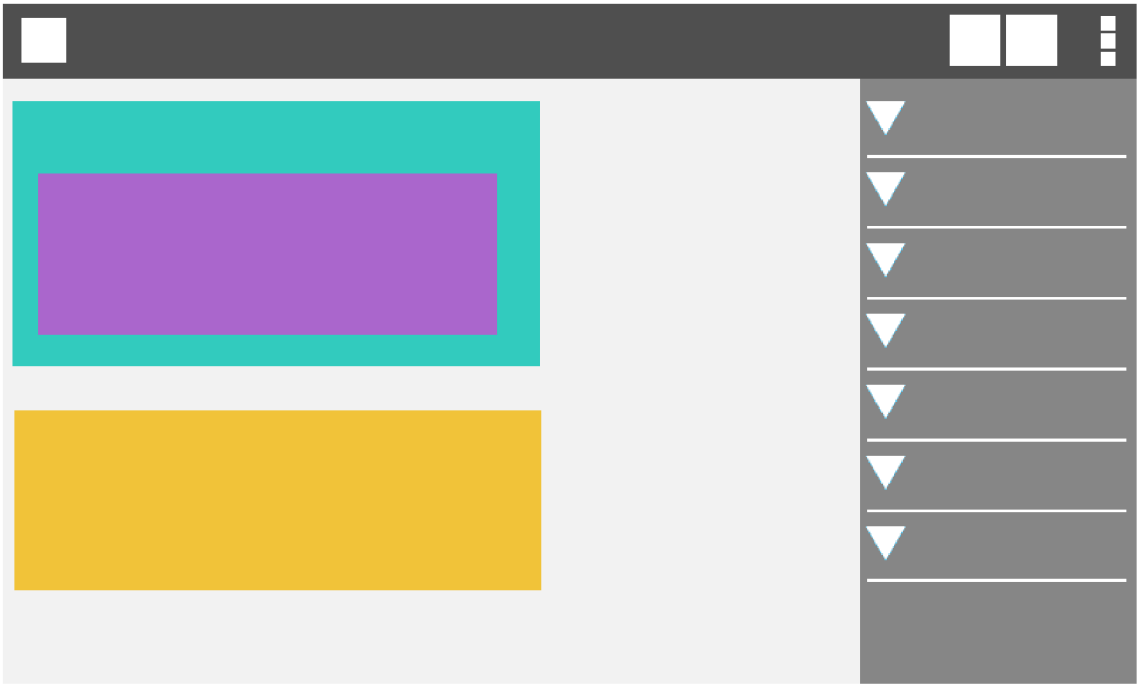
Celé rozhraní jsem se rozhodla vytvářet pro programovací jazyk C, jelikož má pevně danou strukturu kódu. Vzhledem k tomu, že ve zdrojových kódech používáme často stejné sekvence kódu, např. funkce, cykly, iterace, import knihoven, můžeme uživateli ulehčit práci tím, že tyto sekvence již budou v editoru nadefinované a nebude potřeba je neustále dokola psát. Tím také zabráníme spoustě překlepů, které uživatele odrazují od tvorby kódu na dotykových zařízeních. Dalším problémem při psaní kódu na malé klávesnici jsou pro uživatele speciální znaky, jako jsou například ;, <, >, {, }, které bývají mnohdy schované až ve třetí vrstvě klávesnice a proklikání se k nim zabírá uživateli cenný čas. Tyto znaky jsou přitom využívané ve velké míře, je tedy vhodné, aby také tyto znaky uživatel nemusel složitě zadávat. Navrhla jsem tedy řešení, kde jsou různé sekvence kódu již předem definované ve formě bloků a to včetně speciálních znaků. Uživatel tedy žádné speciální znaky nezadává a nemusí hlídat jejich správné umístění. Díky tomu se nemůže stát, že zapomeneme na nějaký středník či závorku.

Každý ze zmíněných bloků reprezentujících kód má také identifikátor, díky kterému je možné kontrolovat jeho umístění a hlídat tak nesmyslné kombinace, jako je například zadání cyklu do podmínky. S bloky je možné manipulovat pomocí gesta drag&drop a lze tak velice snadno složit v podstatě jakýkoliv kód téměř bez použití klávesnice. Tu je nutné použít pro první zadání názvů proměnných a hodnot konstant.

Nahrazením softwarové klávesnice bloky získáme značnou část obrazovky, kterou klávesnice zabírala, pro zobrazování napsaného zdrojového kódu. Zároveň také odstraníme nepohodlné a pomalé zadávání zdrojového kódu, na které si uživatelé dotykových zařízení stěžují.

3.3 Rozvržení editoru

Na obrázku 3.1 je vidět navržené uživatelské rozhraní. Rozhraní je orientováno na šířku zařízení, aby bylo možné zde umístit menu a zároveň zbyl dostatečný prostor pro samotný zdrojový kód. Celé menu se nachází napravo a v každé položce reprezentující skupinu má znak šipky vyzývající uživatele k dotyku. Vlevo se nachází prostor pro umístění jednotlivých bloků tvořících kód. Na horní liště jsou tlačítka pro často využívané akce, jako jsou načtení a ukládání souboru a tlačítko pro hlavní menu.



Obrázek 3.1: Návrh uživatelského rozhraní.

3.4 Menu

Přes menu jsou zpřístupněny jednotlivé bloky v logickém uspořádání do skupin:

- funkce,
- proměnné,
- konstanty,
- datové typy,
- iterace,
- selekce,
- logické operátory,
- unární operátory,
- binární operátory,
- přiřazovací operátory.

Do menu je zařazen import pár základních a nejpoužívanějších knihoven postačujících pro jednoduché programy. Vzhledem k pevně dané struktuře použití knihoven není potřeba pro ně vytvářet bloky, zde zcela postačuje zaškrtávací pole. Zároveň s vybráním jakékoli knihovny se zpřístupní v položce menu s funkcemi několik vybraných funkcí z dané knihovny.

Tím eliminujeme použití klávesnice v případech, kdy není potřeba. Vzhledem k tomu, že aplikace slouží pouze jako nástroj pro testování použitelnosti navrženého uživatelského rozhraní, nebude uživateli umožněno vkládat další knihovny s funkcemi.

Klávesnici však bude nutné použít pro první zadání názvu každé proměnné a hodnoty konstanty. Název proměnné/konstanta se vloží do seznamu proměnných/konstant a stává se z něj blok, jež je možné opakovaně použít. Je zde zavedena kontrola, kdy při zadání písmene do hodnoty konstanty se text vloží do seznamu proměnných.

Menu má v každé své položce, která reprezentuje skupinu, obrázek šipky. Tato šipka znázorňuje, zda je daná skupina rozbalená či sbalená a uživatele podvědomě vede k tomu, že se jedná o tlačítko, které lze stisknout.

3.5 Bloky a ovládání gestem Drag&Drop

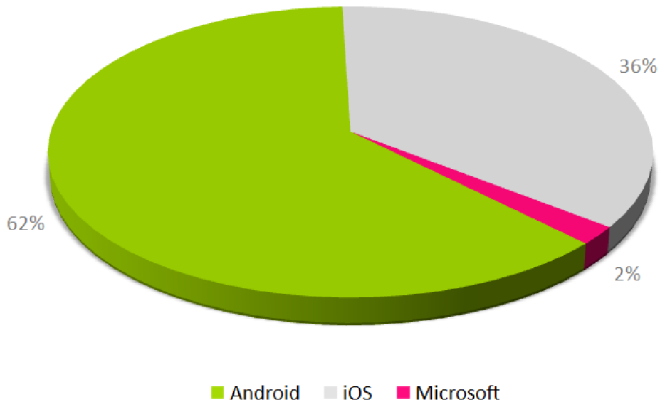
Bloky jsou, pro lepší orientaci uživatele v kódu, barevně rozlišeny podle jednotlivých skupin v menu. Veškerý text v bloku je nadefinovaný a uživatel jej nemůže nijak editovat.

Ovládání je založeno na principu gesta Drag&Drop a slouží pro manipulaci s bloky. Jednotlivé bloky můžeme dotykem uchopit, přetáhnout na požadované místo a zde blok upustit, tím se blok na dané místo vloží. Bloky tak můžeme přesunout z menu do pracovního prostoru a zde s nimi dále manipulovat - zanořovat, vynořovat, měnit jejich pořadí a odstraňovat.

Bloky lze vkládat na místa označená šipkou či nápovědou v podobě bílého textu. Tyto nápovědy při vložení bloku zmizí a jsou nahrazeny vkládaným blokem. Při snaze přesunout blok na místo, na kterém se již jiný blok nachází, se neprovede žádná akce a oba dva bloky zůstávají na svém původním umístění. Změnu pořadí bloku lze provést uchopením daného bloku a přetáhnutím na šipku v místě, na které chceme blok přemístit. Pro odstranění jakéhokoli bloku jej stačí přetáhnout zpět do menu. Na pozici, z které byl blok odstraněn, se opět objeví původní nápověda.

3.6 Platforma

Testovací aplikaci s rozhraním jsem se rozhodla vytvářet pro operační systém Android. K tomuto rozhodnutí mne vedla data ze statistických analýz celosvětového trhu s dotykovými zařízeními, konkrétně tablety [1]. Na obrázku 3.2 je graf znázorňující zastoupení jednotlivých operačních systémů na tabletech v roce 2013. Platformu Android mělo 61.9% všech prodaných tabletů za tento rok. Operační systém Android pro mne tedy byl jasnou volbou.



Obrázek 3.2: OS na tabletech prodaných v roce 2013.

Vzhledem k výsledkům statistických analýz zastoupení jednotlivých verzí OS Androidu na celosvětovém trhu z května roku 2014 [5], v kterých má 83% zastoupení Android verze 4.0 a vyšší, jsem se rozhodla rozhraní navrhnout právě na tyto verze:

- Ice Cream Sandwich - Android 4.0,
- Jelly Bean - Android 4.1, 4.2, 4.3,
- KitKat - Android 4.4.

Kapitola 4

Implementace

V této kapitole jsou popsány implementační detaily hlavních prvků uživatelského rozhraní a stručně představeno vývojové prostředí, ve kterém celá práce vznikala. Dále zde nalezneme zjednodušený diagram tříd se stručným popisem jednotlivých balíků a hlavní třídy.

4.1 Vývojové prostředí

Rozhraní jsem, vzhledem k programovacímu jazyku Java používanému pro programování na OS Android, implementovala v open source vývojovém prostředí **Eclipse** k tomu určenému. Samotný Eclipse ovšem potřebuje spoustu pluginů rozšiřujících možnosti prostředí. Doplňky, které jsem použila, jsou:

JDK (Java Development Kit) - JDK je základním nástrojem pro práci s Javou nezávisle na zařízení na které aplikaci vyvíjíme.

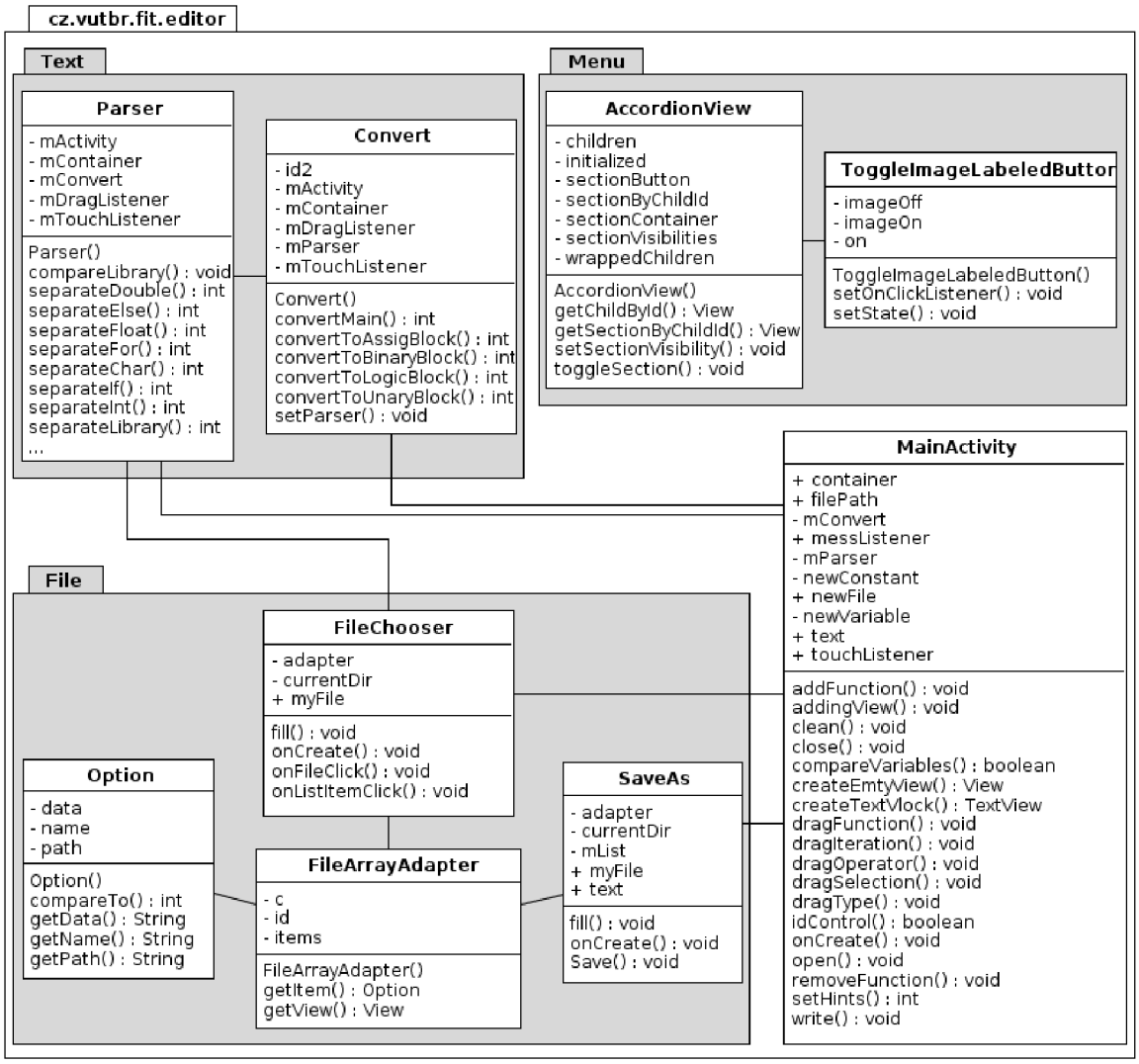
SDK Tools (Software Development Kit) - Jedná se o balíček s kompletní sadou nástrojů pro vývoj a ladění aplikací, včetně emulátoru, které Eclipse využívá.

ADT Plugin (Android Development Tools) - Tento plugin rozšiřuje prostředí Eclipse o nabídky týkající se Androidu. Poskytuje mimo jiné tvorbu uživatelských rozhraní aplikací, vkládání balíčků založené na Android Framework API a exportuje *.apk* soubory aby bylo možné aplikace distribuovat.

Emulátor - Emulátor nahrazuje fyzické zařízení při testování aplikace, testování tak lze provádět přímo na počítači, aniž bychom museli vlastnit jakékoli zařízení s Androidem. Ačkoli takové zařízení vlastním, emulátor jsem využila při testování zobrazení uživatelského rozhraní na zařízeních s jiným rozlišením obrazovky. To lze v emulátoru nastavit spolu s velikostí obrazovky zařízení a máme tak k dispozici jakékoli zařízení.

4.2 Diagram tříd

Na obrázku 4.1 můžeme vidět zjednodušený diagram tříd, který znázorňuje logické propojení jednotlivých tříd, jejich metody a atributy. Tento diagram je zde uveden pro lepší pochopení struktury celé práce a nezobrazuje tak, s ohledem na rozsah a prostor, veškeré implementované metody a jejich parametry. Práce se skládá také z grafické části, která není v diagramu znázorněna.



Obrázek 4.1: Diagram tříd.

MainActivity

Třída `MainActivity` zajišťuje zobrazení hlavního okna aplikace a následnou práci s bloky. Třída implementuje metody, které vytvářejí jednotlivé bloky, provádí kontrolu ID, nastavují správné nápovědy, přidávají/odebírají funkce na základě importovaných knihoven atd. Jsou zde implementovány také naslouchače na uchycení a upuštění bloků.

File

Balík `File` obsahuje kód, který jsem převzala ze zdroje [8] a obsahuje třídy `FileChooser`, `FileArrayAdapter`, `SaveAs` a `Option`. Třída `FileArrayAdapter` představuje adaptér a definuje umístění názvu, typu a velikosti jednotlivých položek — složek a souborů — v aktivitě. Třída `FileChooser` implementuje aktivitu pro načítání souborů, aktivita pro jejich ukládání je implementována ve třídě `SaveAs`. Obě tyto třídy mají definovanou metodu `fill()`, pomocí které vyplní zmíněný adaptér složkami a soubory umístěnými v dotykovém zařízení či na SD kartě.

Text

Balík `Text` obsahuje dvě třídy `Parser` a `Convert`. Třída `Parser` obsluhuje rozdělení textu při načítání na části. Jejich následný převod na odpovídající bloky zajišťuje třída `Convert`. Tato část práce je implementována pouze částečně a očekává pouze ideální vstupní text.

Menu

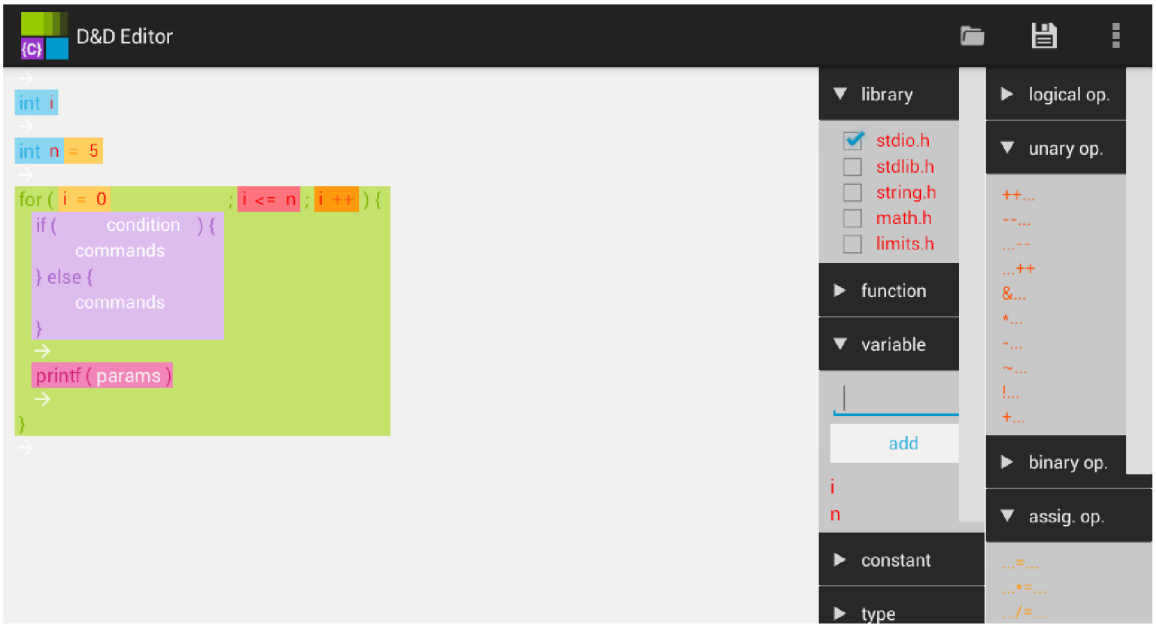
Tento balík obsahuje kód převzatý ze zdroje [6] a implementuje menu, které využívám pro rozdělení bloků do jednotlivých skupin. Třída `ToggleImageLabeledButton` nastavuje přepínač, který slouží pro změnu obrázku identifikujícího rozbalenou/sbalenou skupinu v menu. Třída `AccordionView` obstarává zobrazení grafických prvků menu a jednotlivých sekcí v závislosti na zmíněném přepínači. Obsah jednotlivých sekcí je implementován v grafické části aplikace v **XML** souborech.

4.3 Rozvržení editoru

Rozhraní je implementováno a přizpůsobeno na dotyková zařízení všech velikostí. Jednotlivá dotyková zařízení s operačním systémem Android se dělí podle jejich minimální velikosti obrazovky do následujících skupin.

- **Xlarge** – 960dp x 720dp
- **large** – 640dp x 480dp
- **normal** – 470dp x 320dp
- **small** – 426dp x 320dp

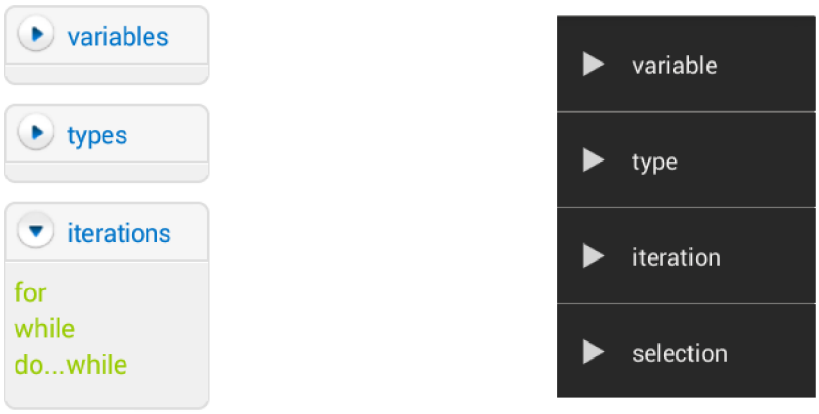
Každá skupina má vlastní zdrojový kód pro grafické zobrazení a je tak snadné vytvořit rozdílné zobrazení pro každou tuto skupinu. Toho je v implementovaném rozhraní využito především k rozdílnému zobrazení menu. Na obrázku 4.2 je zobrazeno rozhraní na zařízení s velikostí obrazovky odpovídající skupině large. Rozhraní se přizpůsobuje jednotlivým dotykovým zařízením v počtu sloupců menu právě v závislosti na jejich velikosti a rozlišení. Více sloupců v menu oceníme především při importu více knihoven, jejichž funkce zaberou při zobrazení značnou část prostoru obrazovky.



Obrázek 4.2: Rozhraní editoru na zařízení s obrazovkou velikosti large.

4.4 Menu

Zdrojový kód implementující menu jsem převzala ze zdroje [6] a upravila mu vzhled tak, aby odpovídal celému rozhraní. Na obrázku 4.3a můžeme vidět, jak vypadalo původní menu a na obrázku 4.3b je zobrazeno menu již upravené. Veškeré změny byly provedeny pouze v grafické části programu ve složce `res/layout-land`.



(a) Původní vzhled menu.

(b) Nový vzhled menu

Obrázek 4.3: Vzhled menu.

V menu jsou využity následující ovládací grafické prvky:

- Checkbox - usnadňující uživateli práci s knihovnami
- EditText - umožňující zadání názvu proměnných a konstant
- Button - sloužící pro vložení proměnných a konstant do seznamu
- TextView - reprezentující položky seznamů

Veškeré zmíněné prvky jsou v menu definovány napevno, včetně předdefinovaných položek operátorů, selekcí, iterací. Výjimkou jsou funkce, názvy proměnných a hodnoty konstant, které jsou do menu vkládány dynamicky.

4.5 Bloky

Jednotlivé bloky jsou vytvořeny pomocí **mřížkového rozvržení**, které obsahuje **popisky**, **lineární rozvržení** a **obrázky** [12]. Každý blok je také identifikován podle specifického **ID**. Ukázku bloku, který reprezentuje selekci if/else, můžeme vidět na obrázku 4.4.

Obrázek 4.4: Ukázka bloku.

Popisky

Popisků je využito pro jasně definované texty např.: *while*, *if/else*, *double*, které jsou znázorněny písmem v barvě daného bloku a nelze je uživatelem nijak měnit. Do této skupiny patří také speciální znaky, jako jsou například jednoduché a složené závorky.

Lineární rozvržení

Lineární rozvržení je použito na místech, kde se očekává nějaký vložený blok. Tato místa poznáme díky vnořenému prvku popisku s nápovědou v podobě bílého textu. Každý popisek má své ID, které je využito při kontrole vkládání bloků. Nápověda je vytvořena z Backus-Naurovy formy (BNF) a označuje, jaký blok má být na tomto místě vložen. V případě, že na dané místo je možné umístit více typů bloku, je nápověda tvořena slovem reprezentujícím tuto skupinu, například *condition*. Při vložení bloku na toto místo je popisek s nápovědou nahrazen daným blokem. V případě odstranění bloku je zde opět umístěna původní nápověda.

Obrázky

Obrázky slouží k zobrazení šipky a naslouchají, stejně jako lineární rozvržení, na upuštěné bloky. Tato šipka je vložena za každý blok, mimo bloky vložené do podmínek selekce *if* a cyklů *while*, *do* a *for*, kde lze umístit vždy pouze jeden jediný blok.

ID

Každý blok má přiřazené ID, jehož hodnota odpovídá skupině v menu, do které blok patří. Výjimkou je funkce main, která nemá ID shodné s ostatními funkcemi, ale má své vlastní. Toto ID je využíváno při vkládání jednotlivých bloků do prostoru. Na základě hodnoty ID bloku a hodnoty ID popisku s nápovědou se rozhodne, zda se na dané místo blok vloží, či nikoli. V případě, že blok nelze na dané místo umístit, se na obrazovce objeví oznámení s informací, že se uživatel snaží vložit nevyhovující blok.

4.6 Ovládání pomocí drag&drop

Ovládání je založeno na principu **drag&drop** což znamená, že je implementováno pomocí naslouchačů na uchycení a upuštění. Naslouchače na uchycení jsou použity na hlavním lineárním plátně, jednotlivých blocích na tomto plátně umístěných a položkách zobrazovaných v menu. Naslouchače na upuštění jsou umístěny na popiscích zobrazujících nápovědy v blocích, a jeden takový naslouchač je také nad prostorem menu, který slouží jako odpadkový koš pro již nepotřebné bloky.

4.7 Ukládání a načítání souborů

Aby bylo možné vyzkoušet a otestovat editaci kódu, implementovala jsem v mé práci také ukládání a načítání souborů. Aktivity, jejichž grafické zobrazení můžete najít v příloze, jsou výsledkem kódu, který jsem převzala ze zdroje [8] a mírně upravila tak, aby vyhovoval mým požadavkům.

V seznamu souborů zobrazovaných při otevírání souboru byly původně soubory všech typů. To je vzhledem k možnosti interpretace pouze zdrojových kódu v jazyce C zcela zbytečné, tudíž jsem kód upravila tak, aby se zobrazovaly pouze soubory s příponou *.c*.

Grafický prvek pro vkládání názvu souboru při ukládání jsem přizpůsobila tak, aby se na každé obrazovce zobrazoval po její celé šířce, nezávisle na velikosti a rozlišení, a aby název ukládaného souboru byl vždy implicitně nastaven na *newfile.c*.

Obě zmíněné aktivity jsou vyvolány při stisku odpovídajících tlačítek na hlavním panelu.

Kapitola 5

Testování a výsledky

Abych mohla shrnout splnění cíle práce, jímž je usnadnit uživatelům práci se zdrojovým kódem na dotykových zařízeních, bylo nutné vytvořené rozhraní otestovat. Tato kapitola popisuje způsob testování rozhraní v jeho jednotlivých fázích, dosažené výsledky z testování a výsledky získané z t-testu.

5.1 Testovací zařízení

Navržené uživatelské rozhraní jsem testovala primárně na zařízení s parametry uvedenými v tabulce 5.1. Přestože jsem se rozhodla pro zařízení s danými parametry, je nutné, aby rozhraní bylo použitelné i na zařízeních s rozdílným rozlišením a bylo tomu přizpůsobeno velikostí plochy pro kód a také počtem sloupců v menu tak, aby byl pro oba prvky vyhrazen dostatečný prostor.

Operační systém	Android
Verze OS	4.1.2
Typ dotykové obrazovky	kapacitní
Velikost displeje	10.1”
Rozlišení	1920x1200 px

Tabulka 5.1: Parametry testovacího zařízení.

5.2 Způsob testování

Testování probíhalo v několika fázích, vždy jinou metodou a s jinými cíli.

Alfa-testování

Během implementace jsem rozpracované rozhraní dala otestovat třem dobrovolníkům, kteří se s rozhraním seznámili a vyzkoušeli si vytvořit zkušební zdrojový kód. Během této fáze testování byl pozorován především pracovní proces dobrovolníků a v jakých místech a případech váhají. Testování proběhlo za účelem odhalit případné chyby a nedostatky navrženého rozhraní, tak aby se uživatelé v závěrečném testování nemuseli těmito nedostatky zabývat

a chyby rozhraní je nezdržovaly v samotné tvorbě kódu. Dalším krokem této fáze testování bylo vyplnění hodnotících dotazníku s otázkami vztahujícími se k průběhu tvorby kódu.

Závěrečné testování

Vytvořené rozhraní testovalo osm dobrovolníků, kteří mají zkušenosti s dotykovými zařízeními a kteří tyto zařízení aktivně využívají. Tyto dobrovolníky jsem rozdělila podle jejich zkušeností s programováním v jazyce C do dvou skupin - **pokročilí** a **začátečníci**. Všechny dobrovolníky jsem nechala přepsat stejný kód nejdříve v jejich oblíbeném editoru se softwarovou klávesnicí, na kterou jsou zvyklí, a poté v editoru s navrženým rozhraním. Před tím, než bylo provedeno samotné testování v editoru s navrženým rozhraním, měli uživatelé čas na to, aby se s aplikací seznámili a napsali zkušební kód. Po této zkoušce, která účastníkům testování pomohla zorientovat se v rozmístění jednotlivých položek menu a ovládání celého rozhraní, byl vyhrazen prostor pro dotazy k rozhraní a poté již účastníci byli připraveni k samotnému testu. Během psaní kódu byl každému účastníkovi testování měřen čas a zároveň byl účastník při práci pozorován. Pozorovány byly především:

- místa výskytu překlepů,
- četnost překlepů,
- dlouhé hledání bloků.

Následně byl každému účastníkovi testování předložen dotazník, v kterém měl celé testování a rozhraní ohodnotit. Na naměřené výsledky jsem poté aplikovala dva párové t-testy na porovnání středních hodnot časů tvorby kódu v editoru se SW klávesnicí a v editoru s vytvořeným uživatelským rozhraním - jeden test pro začátečníky, jeden pro pokročilé. Dále pak jeden dvouvýběrový t-test na střední hodnotu pro porovnání časů v editoru s vytvořeným rozhraním mezi oběma skupinami.

5.3 Ukázka zdrojového kódu

Zde je ukázka zdrojového kódu, který měl za úkol každý účastník testování přepsat ve svém oblíbeném editoru a v editoru s navrženým rozhraním. Tento zdrojový kód jsme vybrala tak, aby bylo možné ho v editoru vytvořit, obsahuje tedy pouze základní prvky jazyka C, které jsou v rozhraní implementovány. Jde o zdrojový kód Fibonacciho posloupnosti a jeho délka je vhodná pro testování z pohledu uživatele - kód není ani moc krátký, jsou zde využity prvky napříč téměř všemi skupinami v menu, a není ani moc dlouhý, čímž neodrazuje uživatele od testování. Délku kódu je nutné brát v potaz při vyhodnocování testování.

```
#include <stdio.h>

void main()
{
   int n, next, c;
   int first = 0;
   int second = 1;

   printf("rad:\n");
   scanf("%d",&n);

   printf("posloupnost: \n");

   for ( c = 0 ; c < n ; c++ )
   {
      if ( c <= 1 )
         next = c;
      else
      {
         next = first + second;
         first = second;
         second = next;
      }
      printf("%d\n",next);
   }
}
```

5.4 Výsledky testování

Alfa-testování

V této fázi testování účastníci dokázali odhalit spoustu chyb a nedostatků, které se při mém používání aplikace neprojevily, nebo se nejevily jako problém. Přehled odhalených nedostatků:

- Nelze přidat novou proměnnou, je-li už nějaká proměnná přiřazena.
- Některé operátory nelze, kvůli jejich malé velikosti, uchopit.
- Pro 80% účastníků testování je písmo v celém rozhraní moc malé.
- Nelze vložit některé bloky na místo, na které patří.
- Nefunguje tlačítko na vyčištění obrazovky.
- Je-li importováno více funkcí z knihoven, nelze pořádně rolovat v menu.

Veškeré tyto nedostatky jsem do další fáze testování buď zcela odstranila, nebo upravila tak, aby nadále uživatelům nezpůsobovaly problémy a nezdržovaly je tak v práci s kódem.

Závěrečné testování

Výsledky této fáze testování jsou zobrazeny ve dvou tabulkách. Jsou zde zapsány časy tvorby výše zmíněného zdrojového kódu v jednotlivých editorech.

Naměřené časy uživatelů s pokročilou znalostí jazyka C najdeme v tabulce 5.2, v které můžeme vidět, že všichni testovaní uživatelé s pokročilou znalostí jazyka C, byli s tvorbou kódu ve vytvořeném editoru rychlejší než v editoru se SW klávesnicí.

Tester	**Čas - SW klávesnice [min]**	**Čas - vytvořené UI [min]**
1	6:56	5:36
2	5:36	5:04
3	7:00	6:08
4	5:27	4:35

Tabulka 5.2: Časy uživatelů s pokročilou znalostí jazyka C.

V tabulce 5.3 jsou zaznamenány časy uživatelů začátečníků. Z této tabulky vyčteme, že zde bylo 50% začátečníků s tvorbou kódu ve vytvořeném editoru pomalejší než v editoru se SW klávesnicí a 50% rychlejší.

Tester	**Čas - SW klávesnice [min]**	**Čas - vytvořené UI [min]**
5	8:27	8:37
6	7:47	7:38
7	7:35	7:19
8	8:26	8:28

Tabulka 5.3: Časy uživatelů se znalostí jazyka C na úrovni začátečník.

Informace, které jsme vyčetli z tabulek, jsou pouze ilustrativní, nejsou nijak podložené a složí tedy pouze pro naši představu o výsledku. Pro reprezentativní výsledky je nutné provést t-test a získané informace tak ověřit.

5.5 Výsledky T-testů

Výsledky ze závěrečné fáze testování jsem podrobila statistickému testu **t-test** [9] v programu Excel. T-test se používá pro porovnání dvou středních hodnot pocházejících z jiných vstupních souborů. Celkem byly provedeny 3 t-testy, jejichž výsledky jsou vyneseny do tabulek 5.4, 5.5 a 5.6. Střední hodnoty a rozptyl časů tvorby kódu jsou pak pro lepší názornost zobrazeny také v grafech na obrázku 5.1.

Pro každý t-test musíme nejdříve stanovit typ testu, **nulovou hypotézu** H_0 a **alternativní hypotézu** H_1. Samotným t-testem poté ověřujeme platnost nulové hypotézy. Nulovou hypotézu zamítáme na základě porovnání hodnoty vypočtené pravděpodobnosti P a hladiny významnosti testu.

Hladina významnosti testu značí pravděpodobnost, že je zamítnuta nulová hypotéza, ačkoli je platná. Hladinu významnosti jsem nastavila na hodnotu 0,05 (resp. 0,01), čímž získáme 95% (resp. 99%) jistotu správného rozhodnutí o nulové hypotéze.

Jestliže má **pravděpodobnost** P hodnotu větší nebo rovnu hodnotě 0,05, nelze zamítnout ani jednu z hypotéz. V případě, že hodnota P je menší než hodnota 0,05, je rozdíl hodnot statisticky významný, nulová hypotéza je neplatná a platí hypotéza alternativní. Je-li přitom hodnota P menší než hodnota 0.01 je rozdíl hodnot statisticky velmi významný.

Test může být buď **jednostranný** (1), nebo **oboustranný** (2) a na základě toho se liší jeho hypotézy. Oba testy pracují s nulovou hypotézou, která tvrdí, že data z obou souborů se rovnají. Oboustranný test pak proti nulové hypotéze klade alternativní hypotézu tvrdící, že mezi daty obou souborů existuje jakýkoli rozdíl, tedy k hodnotám větším i menším. Naproti tomu jednostranný test klade proti nulové hypotéze alternativní hypotézu takovou, kde střední hodnota jednoho souboru je menší/větší než střední hodnota souboru druhého. Pro ověření správně zvolené možnosti menší/větší nám poslouží hodnoty *tstat*. Tato hodnota může být v závislosti na datech kladná nebo záporná. Vzhledem k tomu, že nás zajímá, zda jednotliví uživatelé jsou rychlejší **nebo** pomalejší, není zde použití oboustranného testu vhodné, jelikož připouští odchylky v obou směrech. Všechny testy jsem proto zvolila jako jednostranné.

Kritická hodnota *tkrit* udává hranici mezi přijetím nulové hypotézy a hypotézy alternativní. Tato hodnota je závislá na zvolené hladině významnosti.

Pro dvojice hodnot {čas - SW klávesnice; čas - vytvořené UI} byl proveden párový t-test zvlášť pro skupinu pokročilých (TEST 1) a zvlášť pro začátečníky (TEST 2). Pro porovnání časů v editoru s vytvořeným uživatelským rozhraním mezi oběma skupinami byl proveden dvouvýběrový t-test (TEST3). Abych u TESTU 3 zvolila správný dvouvýběrový t-test, provedla jsem nejdříve F-test, který zkoumá předpoklad pro t-testy a analýzu rozptylu, čímž je pro t-testy předcházející.

TEST 1 - Párový t-test na střední hodnotu - pokročilí uživatelé

Nulová hypotéza: Čas tvorby kódu každého uživatele je s použitím editoru se SW klávesnicí stejný jako čas tvorby stejného kódu v editoru s vytvořeným uživatelským rozhraním. **Alternativní hypotéza:** Tvorba kódu každého z uživatelů je rychlejší v editoru s vytvořeným uživatelským rozhraním.

	SW klávesnice	**vytvořené UI**
Střední hodnota	6:15	5:21
Rozptyl	3,36271E-07	2,16065E-07
t Stat	5,47347	
P(1)	0,00599	
t krit(1)	2,35336	

Tabulka 5.4: Párový t-test na střední hodnotu - pokročilí uživatelé.

Hodnota pravděpodobnosti $P(1) = 0,00599$ z tabulky 5.4 je menší než stanovená hranice 0,01, nulová hypotéza je tedy zamítnuta a platí hypotéza alternativní - **tvorba kódu každého z uživatelů je rychlejší v editoru s vytvořeným uživatelským rozhraním**. Mezi středními hodnotami obou souborů dat je velmi významný statistický rozdíl a rozhodnutí pro alternativní hypotézu je z 99% rozhodnutí správné. Z hodnoty *tstat*, která je v tomto případě kladná, lze vyčíst, že data první souboru jsou větší, než data druhého souboru. Můžeme tedy říci, že uživatelé s pokročilou znalostí jazyka C vytvořili zadaný zdrojový kód ve vytvořeném rozhraní rychleji než v editoru se SW klávesnicí.

TEST 2 - Párový t-test na střední hodnotu - začátečníci

Nulová hypotéza: Čas tvorby kódu každého uživatele je s použitím editoru se SW klávesnicí stejný jako čas tvorby stejného kódu v editoru s vytvořeným uživatelským rozhraním.
Alternativní hypotéza: Tvorba kódu každého z uživatelů je rychlejší v editoru s vytvořeným uživatelským rozhraním.

	SW klávesnice	**vytvořené UI**
Střední hodnota	8:04	8:00
Rozptyl	9,56804E-08	1,92767E-07
t Stat	0,56379	
P(1)	0,30614	
t krit(1)	2,35336	

Tabulka 5.5: Párový t-test na střední hodnotu - začátečníci.

V tomto případě je hodnota pravděpodobnosti $P(1) = 0,30614$ větší než stanovená hranice 0,05 a není tedy dostatek dokladů k tomu nulovou hypotézu zamítnout. Rozdíl mezi středními hodnotami dat souborů je statisticky nevýznamný. To znamená, že rozdíl mezi časem tvorby kódu v editoru se SW klávesnicí a ve vytvořeném editoru není u skupiny začátečníků natolik významný, abychom mohli jednoznačně říci, zda vytvořený editor těmto uživatelům práci urychlil nebo ne.

TEST 3 - dvouvýběrový t-test - pokročilí vs. začátečníci

V tomto testu jsem nejdříve provedla F-test, abych zjistila, jaký dvouvýběrový t-test na data použít. Hodnota výsledku provedeného testu, která je větší než hodnota 0,05, znamená, že rozdíl mezi rozptyly je statisticky nevýznamný. Na základě tohoto výsledku jsem zvolila **Dvouvýběrový t-test s rovností rozptylů**.

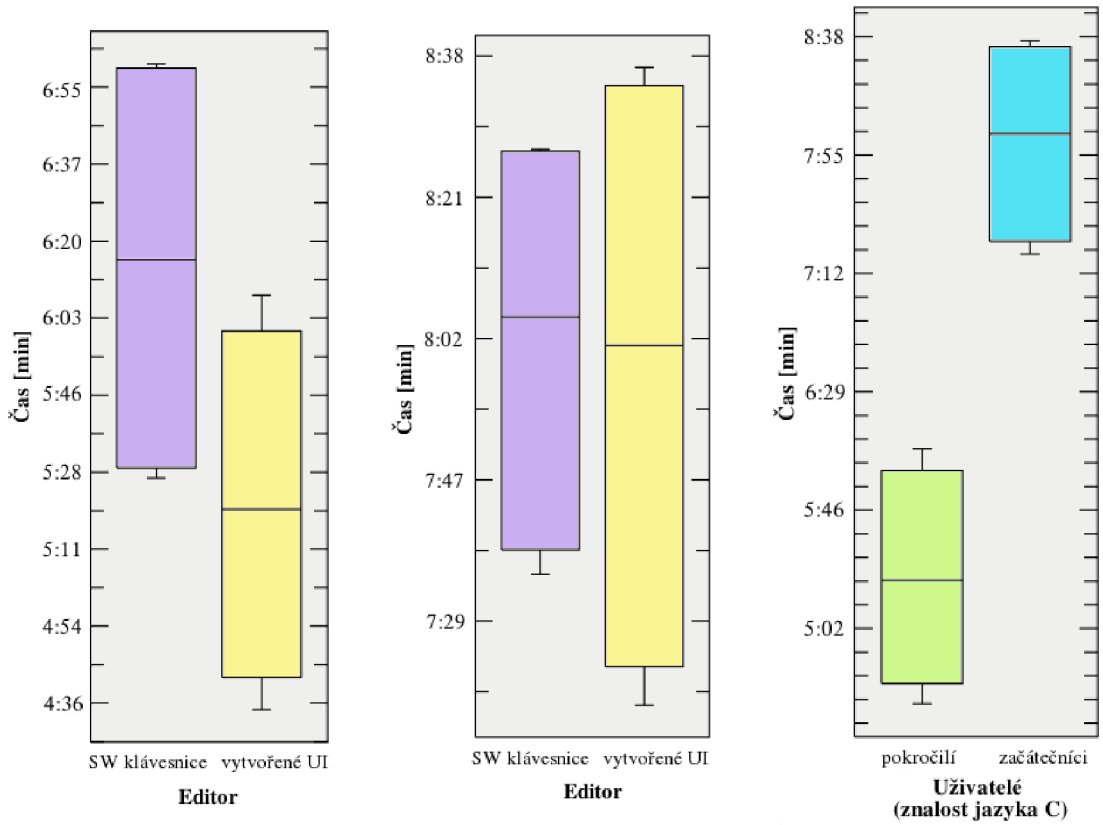
$$Ftest = 0,92748$$

Nulová hypotéza: Čas tvorby kódu začátečníka ve vytvořeného editoru je stejný jako čas pokročilého uživatele.
Alternativní hypotéza: Čas tvorby kódu začátečníka ve vytvořeného editoru je větší než čas pokročilého uživatele.

	začátečníci	**pokročilí**
Střední hodnota	8:00	5:21
Rozptyl	2,04416E-07	2,16065E-07
t Stat	5,78341	
P(1)	0,00058	
t krit(1)	1,94318	

Tabulka 5.6: Výsledky t-testu pokročilí uživatelé vs. začátečníci.

Ve výsledku posledního testu je hodnota pravděpodobnosti $P(1) = 0,00058$ menší než stanovená hranice 0,01, nulová hypotéza je zamítnuta. Platí hypotéza alternativní - **čas tvorby kódu začátečníka ve vytvořeného editoru je větší než čas pokročilého uživatele**, a to s velmi významným statistickým rozdílem středních hodnot. Kladná hodnota *tstat* potvrzuje, že data prvního souboru jsou větší než data souboru druhého. Rozhodnutí pro alternativní hypotézu je na 99% rozhodnutí správné a můžeme tedy prohlásit, že časy začátečníků a pokročilých programátorů v C jsou velmi rozdílné, skupina pokročilých uživatelů je výrazně rychlejší než skupina začátečníků.



(a) Pokročilí uživatelé. (b) Začátečníci (c) Pokročilí vs. začátečníci

Obrázek 5.1: Střední hodnoty a rozptyly naměřených časů při tvorbě kódu.

5.6 Hodnocení uživatelů

Z hlediska času stráveného tvorbou kódu v jednotlivých editorech byl u uživatelů zkušených v programování v jazyce C zjištěn pomocí t-testu markantní rozdíl. Tvorba kódu ve vytvořeném editoru však činila problém uživatelům méně zkušeným v programování, díky nízké míře znalosti jazyka C a jeho struktury, nevěděli tedy, kde jednotlivé bloky hledat. Z pozorování testovaných účastníků při práci však bylo jasně vidět, že jejich schopnost nalézt a vhodně umístit jednotlivé bloky se ke konci testování výrazně zlepšuje.

Ze subjektivního hodnocení zkušených uživatelů vyplývá spokojenost s tím, že v editoru s vytvořeným rozhraním uživatel nezadává středníky a závorky, tudíž na žádný nezapomene. Jako další výhodu uživatelé shledávají způsob importu knihoven, který výrazně šetří čas, a malý prostor pro případné překlepy. K dalšímu šetření času by došlo při častějším použití proměnných, které však s ohledem na malou délku testovaného zdrojového kódu není natolik výrazné.

S nutností vkládat nejdříve blok operátoru a teprve poté bloky hodnot je nespokojeno 38% uživatelů. Všichni testovaní uživatelé by si však dovedli představit práci na zdrojovém kódu ve vytvořené aplikaci, jelikož jim ovládání přijde velice intuitivní. Nikdo z uživatelů nenarazil na problém a 63% uživatelů vyhodnotilo tvorbu kódu ve vytvořeném editoru jako snazší oproti editoru se SW klávesnicí.

5.7 Možnosti budoucího vývoje

Vzhledem k vytvořené aplikaci, která zdaleka neobsahuje všechny prvky jazyka C, je zde velký prostor pro budoucí vývoj, především v oblasti implementační. Tato aplikace editoru by mohla být běžně používána při tvorbě kódů, k tomu jí však chybí některé prvky jako je podpora všech knihoven, možnost přidávat knihovny vlastní, podpora funkcí ve správném tvaru.

Kapitola 6

Závěr

Zabývala jsem se tvorbou dotykové uživatelské rozhraní pro editor zdrojových kódů se specifickou vlastností usnadnění práce uživatelům nahrazením nepohodlné softwarové klávesnice. Nastudovala jsem problematiku dotykových uživatelských rozhraní, vyhledala již existující řešení problému a provedla malý průzkum o využívání editorů na dotykových zařízeních. Poté jsem, s ohledem na získané informace, navrhla vlastní uživatelské rozhraní na dotyková zařízení s operačním systémem Android. V tomto návrhu byla softwarová klávesnice částečně nahrazena bloky, zastupující určité sekvence kódu jazyka C. Manipulace bloky probíhá pomocí gest drag&drop, a není tedy potřeba využívat softwarovou klávesnici nikde jinde, než při zadávání názvů proměnných a hodnot konstant, což má výrazně ulehčit práci s kódem a zabránit různým překlepům a chybám. Navržené rozhraní jsem implementovala v aplikaci na OS Android, aby bylo možné rozhraní zhodnotit a otestovat.

Testování proběhlo ve dvou fázích, v první byly odhaleny chyby, které jsem následně opravila, a v druhé fázi byl osmi testerům měřen čas tvorby kódu. Z výsledků t-testů provedených na základě naměřených časů vyplývá, že skupině pokročilých uživatelů zabrala tvorba kódu ve vytvořeném editoru výrazně méně času, zatímco u skupiny začátečníků není rozdíl prokazatelný. Podle předpokladu práce pokročilých uživatelů byla, ve srovnání s prací začátečníků, rychlejší. Shrneme-li tyto výsledky, lze o vytvořeném rozhraní prohlásit, že v žádném případě práci s kódem nezpomalilo. Uživatelům, kteří mají zkušenosti s programovacím jazykem C, rozhraní dokonce práci urychlilo. Z tohoto hlediska lze tedy považovat cíl práce za splněný.

Literatura

[1] Gartner [online]. 03.03.2014, [cit. 2014-05-05].
URL http://www.gartner.com/newsroom/id/2674215

[2] Microsoft Research [online]. 1.11.2012, [cit. 2014-03-08].
URL http://research.microsoft.com/en-us/projects/touchdevelop/

[3] Into Mobile [online]. 16.12.2013, [cit. 2014-03-02].
URL
http://www.intomobile.com/2013/12/16/top-5-best-keyboard-apps-android/

[4] Castledine, E.; Eftos, M.; Wheeler, M.: *Vytváříme mobilní web a aplikace pro chytré telefony a tablety*. Computer Press, 2013, ISBN 978-80-251-3763-5.

[5] Hildenbrand, J.: Andoid Central [online]. 02.05.2014, [cit. 2014-05-05].
URL http:
//www.androidcentral.com/latest-platform-numbers-are-kitkat-climbs-85

[6] Lopacinski, M.: Android accordion view [online]. 2011, [cit. 2014-04-30].
URL https://github.com/hamsterready/android-accordion-view

[7] Madej, T.: *Srovnání uživatelského rozhraní mobilních telefonu s dotykovým ovládáním*. Diplomová práce, Univerzita Palackého v Olomouci, Olomouc, 2013.

[8] Otero, T.: Simple file chooser [online]. 30.3.2007, [cit. 2014-04-30].
URL http://www.dreamincode.net/forums/topic/
190013-creating-simple-file-chooser/

[9] Otipka, P.; Šmajstrla, V.: Vysoká škola Báňská, technická univerzita Ostrava - Pravděpodobnost a statistika [online]. 14.11.2013, [cit. 2014-05-05].
URL http://homen.vsb.cz/~oti73/cdpast1/KAP11/KAP12.HTM

[10] Resnick, M.: Scratch: Programming for all. *Communications of the acm*, ročník 52, č. 11, Listopad 2009: s. 60–67.

[11] Schneiderman, B.; Plaisant, C.: *Designing the user interface: Strategies for Effective Human-Computer Interaction*. Prentice Hall, pátá vydání, 2009, ISBN 978-03-215-3735-5.

[12] Ujbányai, M.: *Programujeme pro Android*. Grada Publishing a.s., první vydání, 2012, ISBN 978-80-247-3995-3.

Příloha A

Obsah CD

Přiložené CD má následující strukturu

- `Editor`
  - `src` - složka se zdrojovými kódy aplikace
  - `res` - složka se zdrojovými kódy grafické části aplikace
  - `apk` - složka s výslednou aplikací ve formátu *.apk
- `Text`
  - `latex` - složka se zdrojovými kódy textu
  - text práce ve formátu *.pdf
- `Manuál` - složka s podrobným manuálem k instalaci aplikace

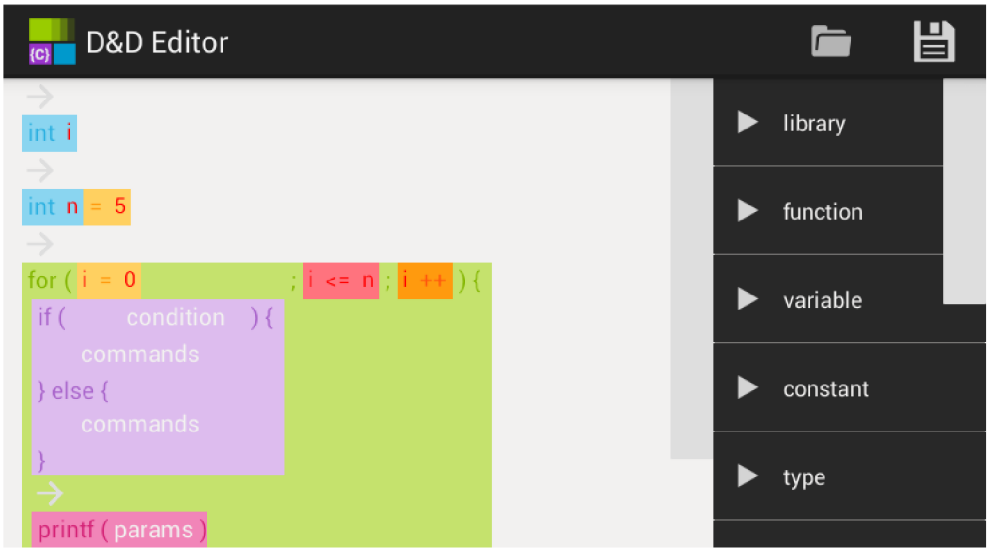
Příloha B

Manuál

Soubor **editor.apk**, který nalezneme na přiloženém CD ve složce `apk`, za pomocí kabelu nahrajeme na dotykové zařízení s operačním systémem Android. Dotykem na ikonu aplikace v zařízení vyvoláme akci instalace a zařízení nás vyzve k potvrzení této akce. Po nainstalování je aplikace plně spustitelná.

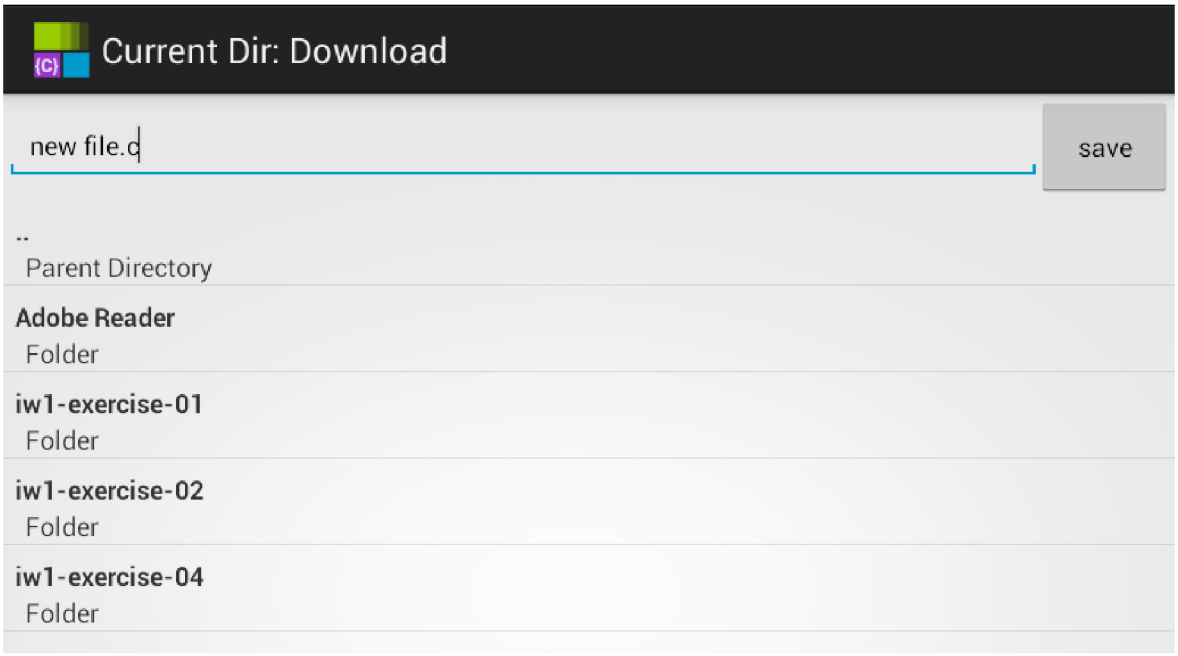
Příloha C

Ukázky UI



Obrázek C.1: Rozhraní editoru na zařízení s obrazovkou velikosti normal.

Obrázek C.2: Načítání souboru.



Obrázek C.3: Ukládání souboru.