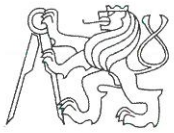
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Zásuvné moduly aplikace Dráček III - výuka matematiky
Student:	Jaroslav Ryba
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Dráček je dotyková vzdělávací aplikace pro OS Android pro žáky prvního stupně základní školy. Zásuvné moduly rozšiřují funkcionalitu klientské aplikace Dráček, která byla předmětem bakalářské práce z minulého roku.

1. Jeden z existujících modulů podrobte uživatelskému testování.
2. Analyzujte výsledky testování a navrhněte úpravy testovaného modulu.
3. Analyzujte učební osnovy pedagogů ze základních škol se zaměřením na výuku matematiky žáků prvního stupně.
4. Navrhněte alespoň 5 nových modulů, které budou podporovat výuku matematiky.
5. Tam, kde je to vhodné, navrhněte editor pro vytváření cvičení pro moduly.
6. Implementujte alespoň 5 zásuvných modulů pro vybrané typy cvičení včetně editoru, vycházejte přitom z existujících modulů a v maximální míře využijte odladěný kód.
7. Hotové moduly podrobte vhodným testům.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Zásuvné moduly aplikace Dráček III – výuka matematiky

Jaroslav Ryba

Vedoucí práce: Ing. Jiří Chludil

15. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Jiřímu Chludilovi za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych rád poděkoval svým kolegům Ondřeji Slabému a Jaroslavu Štěpánovi za velmi dobrou spolupráci s nimi.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Ryba, Jaroslav. *Zásuvné moduly aplikace Dráček III – výuka matematiky*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá vylepšením použitelnosti aplikace pro děti, která se jmenuje Dráček a pracuje pod systémem Android. Cílem první části této práce je odstranění chyb a implementace lepšího uživatelského rozhraní vhodného pro děti pro vybraný existující modul, včetně integrace nových grafických prvků, a testováním použitelnosti s nimi.

Cílem druhé části práce je vytvoření nových matematických naučných her ve formě modulů do této aplikace. V práci je podrobně rozebrána každá fáze vývoje modulů, tedy analýza, implementace i testování. Praktická část práce byla prováděna v nástroji Android Studio za využití programovacího jazyku Java. V implementační části je také popsán postup pro vytváření nových zásuvných modulů pro aplikaci Dráček.

Na přiloženém paměťovém zařízení lze nalézt zdrojové kódy upravovaných a nově vytvořených částí aplikace a nově vytvořené moduly ve formátu apk.

Klíčová slova výuková aplikace, aplikace pro děti, matematika, zásuvné moduly, GUI, Android, Java, mobilní zařízení

Abstract

This bachelor thesis deals with improving the usability of an application for children called Dráček, which works under the Android system. The goal of the first part of this work is to chose one existing plugin, fix it's bugs and implement for it user interface better suited for children. This includes integration of new graphical elements and usability testing with children.

The goal of the second part of this work is to creat new games for teaching mathematics in the form of plugins for the aforementioned application. The thesis contains every development stage of the modules, namely analysis, design, implementation and testing, in detail. The practical part of the work was being done in Android Studio using the Java programming language. In implementation part there is also described how to create new modules for the Dráček application.

You can find the source codes of both the modified and the newly created parts of the application, as well as the new plugins in the apk format, on the included storage medium.

Keywords educational application, apllication for children, mathematics, plugins, GUI, Android, Java, mobile devices

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V posledních letech se díky projektům Evropské unie a různých dalších organizací dostalo do škol velké množství mobilních elektronických zařízení, zejména tabletů. Tím vznikla také poptávka po aplikacích specificky určených pro potřeby učitelů a žáků základních škol. A právě rozvojem jedné z takovýchto aplikací se tato práce zabývá.

Toto téma jsem si zvolil, protože jsem se jím již dříve zabýval a chtěl jsem přispět k rozvoji výukových aplikací (pro Android), jejichž počet je v současné době, zvláště pak v českém jazyce, značně omezený. Jako vhodnou aplikaci pro rozšíření jsem si vybral aplikaci Dráček, která se na této fakultě rozvíjí už několik let.

Výsledky této práce by měly být prospěšné pro žáky prvního stupně základních škol a jejich pedagogy. Práce si taktéž klade za nepřímý cíl zvýšit zájem žáků o přírodovědné obory, zvláště pak obory mající přímou spojitost s matematikou, a jejich schopnosti v těchto oborech.

Tato bakalářská práce navazuje na práce studentů FIT ČVUT Dráček I a II. Dráček I měl za cíl vytvořit desktopovou aplikaci pro pomoc dětem s poruchami učení a Dráček II přesunout tuto aplikaci z PC na mobilní zařízení. Od předchozích iterací projektu Dráček se tato práce liší především rozšířením cílové skupiny z dětí s problémy s učením na všechny děti prvního stupně základních škol. Současně s touto prací vznikají další dvě bakalářské práce zabývající se jinými oblastmi výuky. Konkrétně se jedná o:

- Výuka základů algoritmizace - Ondřej Slabý[7]
- Výuka fyziky - Jaroslav Štěpán [6]

Kapitola 1

Cíl práce

Cílem rešeršní části práce je získat přehled o rámcovém vzdělávacím programu, zejména o učivu matematiky na prvním stupni základních škol, za účelem tvorby nových modulů pro výuku a procvičování matematiky. Dále si klade za cíl analyzovat stav současných výukových aplikací pro děti na prvním stupni základních škol a aplikace Dráček. V neposlední řadě pak vytipovat významné dobré a špatné praktiky sledované v těchto aplikacích pro účely využití ve vlastní tvorbě.

Cílem první části praktické složky práce je upravit jeden vybraný existující modul z prototypu na v praxi použitelný modul, podrobit jej testování a vyhodnocení takto získaných výsledků.

Cílem druhé části praktické složky práce je analýza, návrh a implementace pěti modulů pro aplikaci Dráček, zaměřených na výuku matematiky, včetně editorů sloužících pro přidávání, případně úpravu nových cvičení a propojení těchto modulů s uživatelským klientem aplikace Dráček. V poslední fázi by mělo proběhnout jejich otestování ve spolupráci s pedagogy a dětmi ze základní školy.

Kapitola 2

Analýza

2.1 Učivo matematiky prvního stupně

Dosud se veškeré moduly pro tuto aplikaci zaměřovaly na žáky prvního stupně. Vzhledem k jejich v současnosti nedostatečnému počtu je předčasné vytvářet moduly pro jiné věkové kategorie, proto budou i zde vytvořené moduly zaměřené na první stupeň základní školy. Dle rámcového vzdělávacího programu základního vzdělávání (dále jen RVP) je učivo prvního stupně toto:

- Číslo a početní operace
  - obor přirozených čísel do 1000
  - násobilka
  - zápis a rozklad čísla v desítkové soustavě, číselná osa
  - porovnávání čísel
  - sčítání, odčítání, násobení, dělení
  - práce s kalkulátorem
- Závislosti, vztahy a práce s daty
  - úlohy na orientaci v prostoru a čase
  - manipulační činnosti s konkrétními předměty
  - jednotky hmotnosti, délky a času
  - peníze
  - tabulky
- Geometrie v rovině a v prostoru
  - základní útvary v rovině – bod, čára, přímka, polopřímka, vzájemná poloha dvou přímek v rovině, úsečka, délka úsečky, trojúhelník, čtverec, obdélník, čtyřúhelník, kružnice, kruh
  - základní útvary v prostoru – kvádr, krychle, koule, válec
  - osová souměrnost

- Aplikační úlohy
  - číselné a obrázkové řady
  - doplňovačky [8]

Celé učivo je značně rozsáhlé a pojmutí jej v plném rozsahu dalece přesahuje rozsah této bakalářské práce. Z toho důvodu je nezbytné vybrat si pouze omezenou oblast, kterou se budu zabývat. Rozhodl jsem se, že pro tvorbu modulů se budu převážně zabývat obory Číslo a početní operace a Aplikační úlohy.

2.2 Současný stav výukových aplikací matematiky

Když jsem se začínal zabývat problematikou naučných her pro děti (2015), byla pod systémem Android na téma matematiky, zvláště pak v češtině, jen hrstka aplikací. V současné době (2017) jich však začíná značně přibývat, což je zřejmé při prohlédnutí si nabídky aplikací na Google Play. Je tedy nezbytné vytvářet další? V této části se na to pokusím odpovědět zhodnocením současných aplikací.

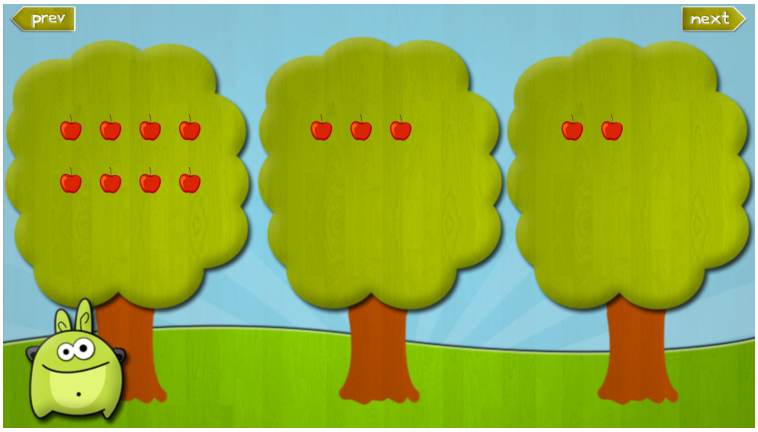
Aplikace budu vybírat dle několika kritérií. Za prvé musí být pro systém Android. Za druhé musí být (alespoň v základní verzi) zdarma. Za třetí se musí zabývat problematikou matematiky. Pro nalezení aplikací budu využívat platformy Google Play, protože má největší sbírku aplikací pro Android a je dostatečně rozmanitá, takže pro výběr vzorku na otestování postačí. Vyhledávat budu pomocí klíčových slov "matematika"a "výuka matematiky".

Dle výše uvedených kriterií byly jako vzorek vybrány aplikace Matematika příklady [4], Learn Math [2], Kids Learning Math Lite [1] a Child Math [3].

Hodnotit na těchto aplikacích budu (5b velmi dobré, 0b nejhorší):

- jazykovou podporu - 5b čeština, 2b alespoň angličtina
- snadnost používání a chybovost - 4b drobné, na první pohled nepatrné problémy, 2b zřejmé problémy, ale stále použitelné
- nápaditost, tzn. neobsahuje tutéž problematiku co mnoho ostatních - bonusové body za novátorské zpracování, neobvyklou část učiva atd.
- gamifikaci (herní prvky, animace, názornost) - 5-3b zajímavější herní prvky, 2b názorné grafické ztvárnění, 0b strohé
- rozsáhlost aplikace (počet cvičení, úrovně obtížnosti ...) - 1b základní obsah alespoň na 30 min hraní, +2b více oblastí z RVP, +2b větší počet cvičení (10+)
- podporu pedagogů - +3b možnost přidávat, editovat, mazat cvičení/zadání, +2b možnost kontroly výsledků žáků

2.2.1 Kids Learning Math Lite



Obrázek 2.1: Kids Learning Math [1]

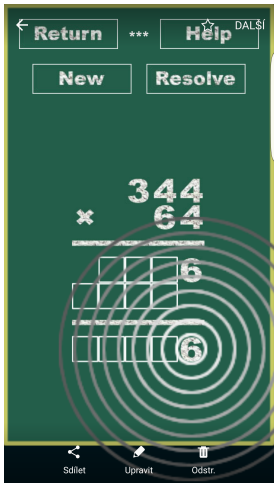
Tato aplikace si zřejmě klade za cíl za využití jednoduchého grafického ztvárnění pomoci s výukou základních matematických operací. Ve verzi zdarma jsou však k dispozici pouze cvičení na sčítání, odčítání a číslice, navíc aplikaci v některých případech reakce na kliknutí zabere příliš mnoho času (řády sekund) a někdy kliknutí přímo ignoruje. Největší limitací pro použití v českém prostředí je však nepřítomnost podpory českého jazyka, podporován je pouze jazyk anglický.

Tabulka 2.1: Kids Learning Math Lite

jazyk	angličtina	(2b)
chyby, problémy	pomalá odezva	(4b)
nápaditost	ne	(0b)
velikost	malá, jedna oblast	(1b)
podpora pedagogů	ne	(0b)
gamifikace	grafické ztvárnění příkladů pro pochopení	(2b)
celkové hodnocení	9/30	

2.2.2 Learn Math

Tato aplikace je na první pohled velmi zajímavá. Zaměřuje se na výuku počítání složitějších výpočtů způsobem „pod sebe“, což ji činí ojedinělou výjimkou mezi mnoha aplikacemi, které v zásadě dělají to samé (grafické ztvárnění jednoduchého sčítání, odčítání apod.). Je zde možnost volby obtížnosti a čtyři různá cvičení.



Obrázek 2.2: Learn Math [2]

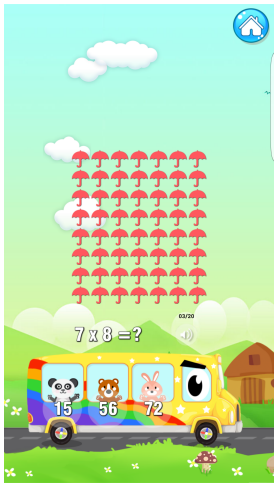
Learn Math podporuje hned 5 jazyků, čeština mezi nimi však naneštěstí chybí. V některých místech je navíc zřejmé, že i do angličtiny byla hra pouze přeložena a například spouštěcí ikonka se španělským popiskem (přesto, že oficiální název je v angličtině) dojem z aplikace, spolu s některými zřejmými nedodělky, značně kazí.

Tabulka 2.2: Learn Math

jazyk	překlad do angličtiny s částmi Španělsky	(1b)
chyby, problémy	chyby, nedodělky	(2b)
nápaditost	střední, unikátní oblast	(3b)
velikost	malá, jedna oblast	(1b)
podpora pedagogů	ne	(0b)
gamifikace	ne	(0b)
celkové hodnocení	6/30	

2.2.3 Child Math

Child Math je velmi pěkná aplikace na učení se základních početních operací. Ačkoliv neobsahuje žádné herní elementy jako takové (sbírání bodů, odměny za úspěchy a poprvé provedené akce apod.), příklady jsou podpořeny alespoň jejich animovaným ztvárněním, což zvyšuje užitečnost a zábavnost. Jako ostatní zmíněné aplikace je ovšem pouze v angličtině.



Obrázek 2.3: Child Math [3]

Tabulka 2.3: Child Math zhodnocení

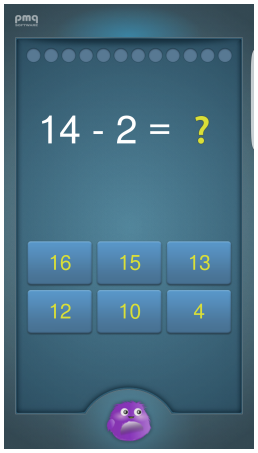
jazyk	angličtina	(2b)
chyby, problémy	dobré	(5b)
nápaditost	ne	(0b)
velikost	střední, jedna oblast	(3b)
podpora pedagogů	ne	(0b)
gamifikace	grafické ztvárnění příkladů pro pochopení	(2b)
celkové hodnocení	12/30	

2.2.4 Matematika příklady

Ojedinělá aplikace v češtině. Obsahuje 20 různých (ale malých) cvičení na základní početní operace. Na rozdíl od předešlých aplikací, nejeví však ani náznak originality. Jedná se pouze o strohé příklady a veškeré pokročilejší grafické zpracování se omezuje pouze na „smajlíky“ v menu. Největší předností této aplikace je však přítomnost jednoduché nápovědy.

Tabulka 2.4: Matematika příklady zhodnocení

jazyk	čeština	(5b)
chyby, problémy	dobré	(5b)
nápaditost	ne	(0b)
velikost	střední, jedna oblast	(3b)
podpora pedagogů	ne	(0b)
gamifikace	ne	(0b)
celkové hodnocení	13/30	



Obrázek 2.4: Matematika příklady [4]

2.2.5 Zhodnocení výsledků

Tabulka 2.5: Celkový výsledek aplikací

Jméno aplikaci	Bodů
Kids Learning Math Lite	9/30
Learn Math	6/30
Child Math	12/30
Matematika příklady	13/30

Většina těchto aplikací se zabývala učivem 1.-3. třídy, s výjimkou Learn Math, jejíž obsah sahal přes celé učivo 1. stupně, avšak byla (dle mého subjektivního názoru) značně nudná a především obsahovala zřejmé programové chyby. Žádná z aplikací nenabízela větší herní zpracování učiva nebo rozmanitost. Většina z aplikací obsahovala pouze propojení jednoduchých výpočtů s obrázky ($3 + 3 = ?$ 3 jablíčka + 3 jablíčka = ? jablíček) bez většího důrazu na zábavnost samotných her a zajímavost příkladů. Všechny aplikace se zaměřovaly pouze na učivo z oblasti „číslo a početní operace“. Žádná z aplikací neumožňovala přístup pedagogů k výsledkům žáků ani přidávání či úpravu příkladů. Nejlépe si vedoucí aplikace byly všechny v anglickém jazyce, zatímco výběr aplikací pro český jazyk je stále velmi omezený.

Z tabulky 2.5 lze nahlédnout, že žádná z testovaných aplikací nesplňuje hodnotící kritéria z více než poloviny, a proto by bylo vhodné vytvořit novou aplikaci, která si povede lépe. Právě o vytvoření takovéto aplikace, se také budu dále v této práci snažit (přesněji rozšiřováním o nové hry aplikace mající potenciál tohoto dosáhnout). Na závěr se pokusím své výsledky zhodnotit podle týchž kritérií, jako byly použity zde a mým cílem bude dosáhnout alespoň částečně lepšího výsledku.

Pro tvorbu nových modulů do aplikace Dráček (aplikace vybrané pro rozšiřování) ze současného stavu plyne, že je v českém prostředí více než dost místa pro nové výukové aplikace a tudíž tato tvorba má smysl. Dále je vhodné zaměřit se na jiný způsob výuky než pouhé spojení obrázků (jablíčka) a příkladů. V blízké budoucnosti se dá totiž očekávat vznik překladů, případně napodobenin anglických aplikací, kterých je na toto téma nejvíce (přesněji převážná většina je pouze na toto téma). Jakékoliv herní elementy jsou jistě také vítané, protože používání neherních aplikací velmi rychle omrzí, což je obzvláště pravdou pro dětské uživatele.

Z výše uvedeného si lze jako ponaučení (z kladného příkladu) pro tvorbu modulů vzít tyto věci:

- Nápověda pro žáky
- Možnost více úrovní obtížnosti
- Grafické zpracování příkladů nebo uživatelského prostředí
- Gamifikace

2.3 Současný stav aplikace Dráček

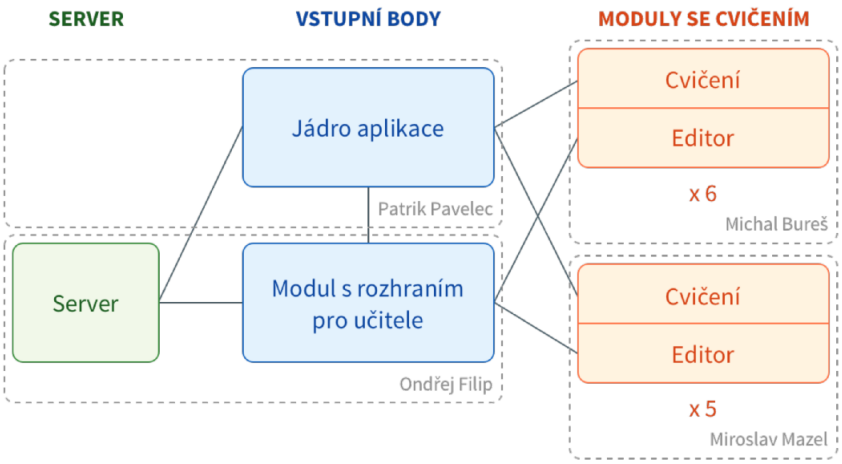
Dráček je dotyková vzdělávací aplikace pro OS Android zaměřená na žáky prvního stupně základních škol. Dle [9] vznikl projekt Dráček I pro vývoj této aplikace v desktopové verzi (verze pro PC) v roce 2012 s cílem pomoci dětem s poruchami učení. V roce 2015 poté vznikl projekt Dráček II, který si kladl za cíl přenést tuto aplikaci do mobilních zařízení, konkrétně pod systém Android.

Dle [5] byla původní desktopová aplikace i současná aplikace běžící pod Androidem rozdělena na několik části:

- **Jádro aplikace** starající se o správu přihlášených uživatelů, provázání a spouštění jednotlivých zásuvných modulů.
- **Zásuvné moduly**, kterými jsou převážně jednotlivá cvičení.
- **Serverová část**, jejímž hlavním úkolem je uchování a distribuce dat

Schéma architektury můžete nalézt na obrázku 2.5.

V současné době aplikace obsahuje prototypy modulů: Hodiny, Délka slabik, Doplnění prázdného místa, Meze čísel, Slovní úlohy [10], Vybarvování, Ukazování, Skládání, Obrácení, Rozlišování, Figury a pozadí. Všechny tyto moduly jsou zaměřené na pomoc dětem z prvního stupně základních škol s poruchami učení. [9]



Obrázek 2.5: Architektura aplikace dráček, převzato z [5]

2.4 Úprava a testování existujícího modulu

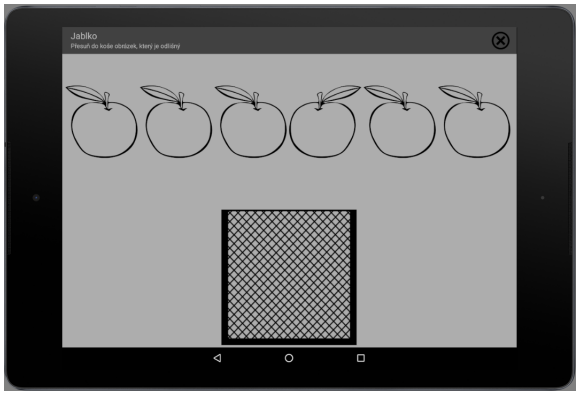
Před vytvářením nových modulů pro aplikaci Dráček bylo nejdříve nutné zanalyzovat současný obsah. Cílem takovéto analýzy bylo získání nových nápadů a zkušeností pro tvorbu modulů a také poučení se z chyb mých předchůdců. Jako nástroj jsem si zvolil uživatelské testování modulů, spojené se zkoumáním kódu a jeho úpravou na základě výsledků uživatelského testování a předchozí analýzy možných problémů. Tímto se navíc zvyšuje užitečnost a připravenost k nasazení vybraných modulů, zvláště přihlédnu-li k faktu, že při vytváření těchto modulů nebylo uživatelské testování provedeno.

Pro uživatelské testování jsem si zvolil modul Obrácení, protože je psán v jazyce Java, který znám mnohem lépe než jazyk Kotlin použitý pro některé další moduly. Tento modul je značně podobný s několika dalšími, proto tak může sloužit jako dobrý vzor pro jejich případnou pozdější úpravu.

Dle [9] se tento modul zabývá testováním schopnosti rozlišování reverzních figur. V modulu je zobrazena skupina obrázků a úkolem žáka je vybrat obrázek, který je stranově převrácený nebo pootočený a přesunout jej do koše.

V první fázi úprav byly zanalyzovány a odstraněny potencionální problémy, aby mohlo být smysluplně provedeno testování s žáky (původní modul byl pouze prototyp, a proto ne zcela připraven k nasazení). Zejména se zde jednalo o:

- **příliš malou citlivost koše na vložení obrázku** – obrázek se při nepřesném vložení do koše chápal jako nevložený. Toto bylo nutno upravit změnou algoritmu kontroly pozice obrázku. Místo původního algoritmu kontrolujícího všechny rohy obrázku, zda se nachází v koši, byl implementován nový algoritmus, který kontroluje, zda se v koši nachází střed obrázku, čímž dává mnohem větší volnost pro nepřesnost vložení.



Obrázek 2.6: Původní grafické rozhraní modulu Otáčení

Obrázek 2.7: Nové grafické rozhraní modulu Otáčení

- **nezobrazování výsledku po každém úkolu** – uživatel nebyl nijak informován, že úspěšně splnil jeden úkol a došlo k přímému přesunu na úkol další, takže mohl být z výsledku zmatený. Toto platí obzvláště když se jedná o aplikaci pro děti. Tento problém byl opraven vložením malého vyskakovacího okénka (komponenta Toast) s grafickým zhodnocením správnosti.

- **špatné zobrazování výsledků po splnění všech úkolů** – po splnění všech úkolů se zobrazil anglicky psaný řetězec v XML formátu. To nebylo chybou samotného cvičení, ale absencí jádra aplikace v době testování, jehož činnost byla tak pouze simulována uvnitř modulu. Pro testování s dětmi by však zobrazování se jakéhosi „nepochopitelného textu“ místo výsledků představovalo značný problém, a proto byla simulace jádra o tuto funkcionalitu rozšířena.

Dalším nedostatkem modulu Obrácení bylo příliš strohé grafické prostředí, které nebylo vhodné pro zaujetí pozornosti děti (viz Obr. 2.6). Proto bylo pro tento modul ve spolupráci se studenty oboru Počítačové grafiky navrženo a implementováno nové uživatelské prostředí (viz Obr 2.7). Toto nové grafické rozhraní je nejen barevnější a živější (což lépe odpovídá jeho účelu), ale také je stylově sjednoceno s dalšími moduly upravovanými jinými studenty, čímž se usnadňuje orientace uživatelů v modulech a vytváří se lepší celkový dojem.

Během uživatelského testování jsem se rozhodl zaměřit zejména na snadnost použití, jasnost výsledků a celkové pocity uživatelů z aplikace.

Testování použitelnosti proběhlo s žáky 3. třídy základní školy Smečno v laboratoři pro testování použitelnosti a laboratoři SAGE. V oblasti snadnosti použití si tato aplikace vedla velmi dobře. Uživatelé byli schopni orientovat se v ní bez problémů a dlouhého přemýšlení. Výsledky byly taktéž zobrazovány dostatečně zřetelně a pochopitelně. Celkový dojem z aplikace byl u uživatelů dobrý, dle jejich neverbálního vyjadřování však lze hodnotit, že úlohy byly až příliš lehké, a proto by dlouhodobější řešení mohlo vést ke znuděnosti a z toho vyplývající ztrátě pozornosti a efektu učení.

Vůči referenční aplikaci (Jdu do školy, vyvinutá taktéž na FIT ČVUT) tento modul zaostal absencí hlasových pokynů a animací. Tyto dva nedostatky bylo však rozhodnuto ponechat tak jak jsou, protože pro hlasové pokyny je vhodné používat ve všech modulech stejný hlas (což v současné době nelze zajistit) a animace jsou pro současné účely příliš pracné, případně drahé. V aplikaci Jdu do školy byla navíc nápověda pomáhající žákům s pochopením úkolů, které mají provádět a úkoly tak nemusely být vysvětlovány moderátorem. Na druhou stranu tato nápověda blokovala pokračování v řešení, a tak obzvlášť při řešení více úkolů po sobě působila značně rušivě.

Průběh testování ostatních modulů si můžete přečíst v pracích Ondřeje Slabého [7] a Jaroslava Štěpána [6] (vznikajících současně s touto prací). Pro účely této práce z ostatního testování zvláště vyplývá, že předem provedené úpravy byly účinné a ponechání těchto nedostatků může snižovat použitelnost aplikací. Navíc se při tomto testování také objevila nutnost „rozumně rychlé“ odezvy na uživatelskou akci (v řádu desetin sekundy). Pomalá odezva při stisknutí tlačítka vyhodnocení (a přejití na další úkol) totiž často vedla žáky k několikanásobnému klikání, tudíž snížení spokojenosti s hrou i k neúmyslnému přeskočení některých úkolů.

Pro účely tvorby nových modulů z testování tedy především vyplynuly tyto požadavky na moduly:

- Zobrazovat výsledky po splnění každého dílčího úkolu
- Grafické prostředí vhodné pro děti (barevnost, snadnost obsluhy)
- Rychlost odezvy na uživatelské akce pod 1s
- Integrace s jádrem

2.5 Moduly Dráček III

Nyní se tedy už mohu pustit do vytváření nových modulů. V této kapitole se budu věnovat sepisování jejich základního obsahu a požadavků na ně.

2.5.1 Modul Balónky

Bude se jednat o jednoduchou hru. Žák musí dle krátkého textového (či číselného) popisu vybrat balónek se správným číslem, což celé probíhá v časovém limitu určeném pohybem balónků směrem vzhůru.

Pokud žák odpoví špatně balónek praskne a žáku se odečtou body. Pokud vyprší čas (balónek se správnou odpovědí se dostane mimo obrazovku) odečtou se body a postoupí se k dalšímu příkladu. Pokud žák odpoví správně, přičtou se body a postoupí se k dalšímu příkladu. Po dokončení všech příkladů se zašle výsledek (čas a procentuální úspěšnost) jádru aplikace Dráček.

Modul Balónky je zaměřen na procvičování jednoduchých výpočtů. Využití tohoto modulu je velmi široké, protože základem modulu je výběr číselné odpovědi na otázku, která může být jak číselná tak i textová. Zejména může být (dle specifického zadání) využit pro procvičení tohoto učiva z oboru Číslo a početní operace:

- obor přirozených čísel do 1000
- násobilka
- sčítání, odčítání, násobení, dělení

2.5.2 Modul Pyramidy

Na obrazovce se objeví pyramida částečně vyplněná čísly. Každá dlaždice (kámen) pyramidy, kromě první řady, má tu vlastnost, že její hodnota je součtem hodnot dvou dlaždic pod ní. Úkolem žáka bude správně vyplnit všechny dlaždice pyramidy.

Modul Pyramidy je zaměřen na procvičování tohoto učiva:

- sčítání
- odčítání
- doplňovačky

2.5.3 Modul Hadi

Na obrazovce se objeví „had“ znázorňující řadu čísel a operaci, jak se z jednoho čísla dostat na jiné. Had bude tvořený obdélníky pospojovanými šipkami. Některé z těchto čtverců budou obsahovat čísla, zatímco jiné budou prázdné. U čar bude uvedena operace (+4, -1...) a může být uvedeno i obrazové znázornění této operace. Úkolem žáka bude doplnit správně všechny čtverce (po směru operací, případně i proti směru).

Předlohou pro vytvoření tohoto modulu byly úlohy z knihy Matematika hrou 1 [11].

Tento modul je zaměřen na výuku a procvičování tohoto učiva:

- sčítání, odčítání, násobení, dělení
- doplňovačky

2.5.4 Modul Magické trojúhelníky

Na obrazovce se objeví trojúhelník složený z 9 polí pospojovaných čarami a jednoho pole uprostřed. Některá z polí budou prázdná a některá budou obsahovat čísla určující zadání. Úkolem žáka bude vyplnit všechna pole tak, aby každá ze stran trojúhelníku měla stejný součet a každá číslice byla využita právě jednou. Toto cvičení také umožňuje nápovědu, která prozradí, kolik tento součet má být.

Předlohou pro vytvoření tohoto modulu byly úlohy z knihy Matematika hrou 2 [12].

Tento modul je zaměřen na logické myšlení a z rámového vzdělávacího programu zejména procvičování tohoto učiva:

- sčítání, odčítání
- doplňovačky

2.5.5 Modul Řady

Před žákem se objeví řada polí, z toho několik prvních vyplněných. Úkolem žáka bude zjistit, podle jakého principu (vzorce) jsou pole vyplněna a pokračovat v řadě vyplněním zbylých polí. Po dokončení úlohy se žákovi odkryje, podle jakého principu byla řada tvořena.

Tento modul slouží pro procvičování logického myšlení, z rámcového vzdělávacího plánu se jedná zejména o procvičování tohoto učiva:

- sčítání, odčítání, násobení, dělení
- číselné a obrázkové řady

2.6 Funkční a nefunkční požadavky

Pro přehlednost jsou funkční požadavky označené FR, nefunkční požadavky NR.

Společné požadavky

- **CFR1** Moduly dokáží načítat zadání od jádra
- **CFR2** Moduly zobrazí výsledek (správně/špatně) po každém dílčím úkolu
- **CFR3** Moduly pošlou jádru spotřebovaný čas a procentuální úspěšnost řešení po dokončení cvičení
- **CFR4** Cvičení bude možné ukončit před vyřešením, bez uložení rozpracovaného zadání
- **CFR5** Editory dokáží načítat a ukládat zadání z jádra
- **CFR6** V editorech lze vytvářet nová zadání
- **CFR7** V editorech lze upravovat stávající zadání
- **CFR8** V editorech lze mazat existující zadání
- **CFR9** Moduly přijímají zadání složené z více dílčích úkolů
- **CFR10** V modulech je nápověda
- **CNR1** Uživatelské rozhraní bude v českém jazyce
- **CNR2** Moduly i editory budou fungovat pod platformou Android 5.0
- **CNR3** Uživatelské rozhraní modulů bude vhodné pro žáky prvního stupně

Modul Balónky

- **FR1.1** Modul zobrazí zadání v textové podobě
- **FR1.2** Možná řešení se zobrazí na balóncích
- **FR1.3** Výběr řešení kliknutím na balónek
- **FR1.4** Balónky se pohybují vzhůru a po zmizení správné odpovědi mimo obrazovku je úkol brán jako nesplněný
- **FR1.5** Procentuální hodnocení se průběžně zobrazuje

Modul Pyramidy

- **FR2.1** Modul zobrazí pyramidu z dlaždic se zadáním
- **FR2.2** Do pyramidy lze doplňovat řešení
- **NR2.1** Předvyplněná pole jsou barevně odlišena od polí na řešení

Modul Hadi

- **FR3.1** Modul zobrazí posloupnost obdélníků se zadáním pospojovaných čarami
- **FR3.2** U spojnic bude operace a její grafické znázornění
- **FR3.3** Do prázdných obdélníků lze vyplnit řešení
- **FR3.4** Vyhodnocuje se správnost řešení - každé číslo je tvořeno jako předchozí číslo, na němž je provedena operace
- **NR3.1** Předvyplněná pole jsou barevně odlišena od polí na řešení

Modul Magické trojúhelníky

- **FR4.1** Modul zobrazí 10 polí se zadáním, 9 do trojúhelníku, 1 uprostřed
- **FR4.2** Do polí lze vyplnit vždy každou číslici jen jednou
- **FR4.3** Stisknutím tlačítka se vyhodnotí shodnost součtů na všech stranách trojúhelníku
- **FR4.4** Po stisknutí tlačítka nápovědy se zobrazí správný součet strany trojúhelníku

Modul Řady

- **FR5.1** Modul zobrazí řadu polí s několika prvními vyplněnými
- **FR5.2** Do nevyplněných polí lze vyplnit řešení (dle odvozeného vzorce)
- **FR5.3** Po dokončení cvičení se zobrazí správný vzorec
- **FR5.4** Vyhodnocuje se správnost řešení
- **FR5.5** V editoru lze zadávat řady pomocí počtu vyplněných polí, celkového počtu polí a jejich obsahu
- **NR5.1** Předvyplněná pole jsou barevně odlišena od polí na řešení

Kapitola 3

Návrh

3.1 Zvolené řešení, technologie

Programovat pod systémem Android lze ve více programovacích jazycích. Za zmínku především stojí programovací jazyky Java a Kotlin, které byly také použity v předchozí iteraci projektu Dráček [9],[10]. Pro řešení této práce byl nakonec zvolen jazyk Java a to především kvůli jeho vysoké podpoře pod systémem Android, a pak také kvůli jeho předchozí znalosti autorem.

Z možných vývojových prostředí bylo vybráno Android Studio pro jeho větší specializaci na systém Android než ostatní IDE, například Eclipse.

3.2 Společná architektura modulů a společná knihovna

Při návrhu třídní struktury jsem se inspiroval prací Michala Bureše [9] a to hned z několika důvodů:

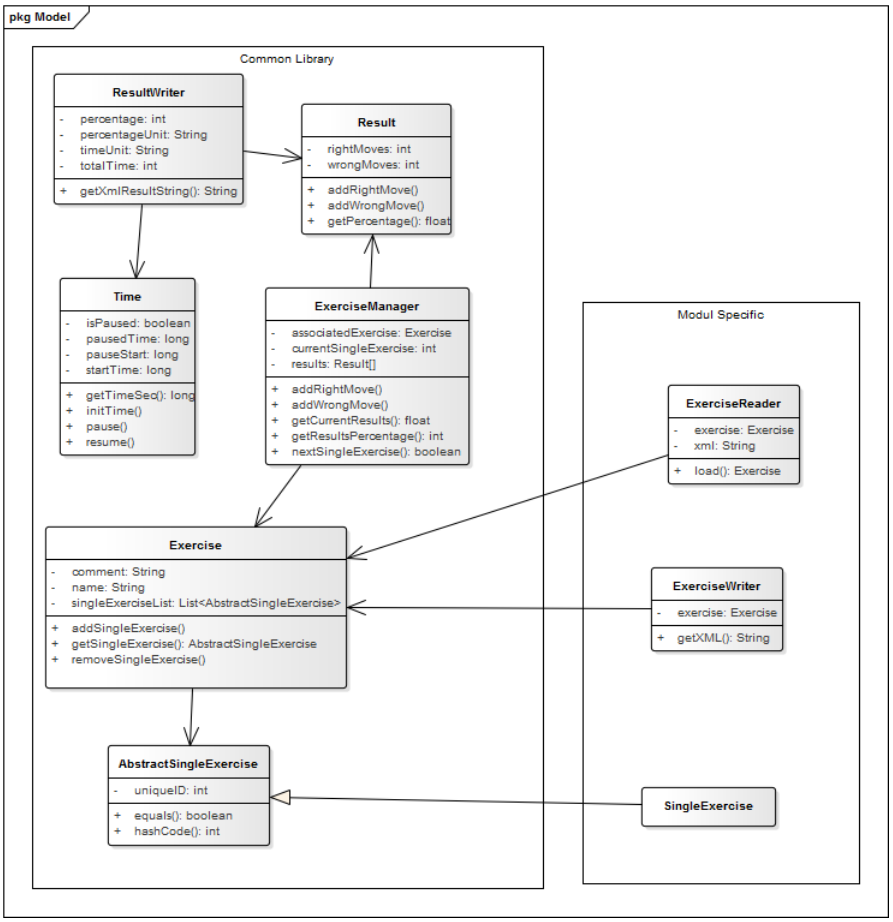
1. Je zbytečnou ztrátou času zcela nově navrhovat něco, co už někdo vymyslel a funguje.
2. Pro budoucí udržovatelnost modulů je vhodné, když se jejich struktura navzájem podobá.
3. Moduly jsou poměrně nekomplexní a struktura jejich dat i částí je značně podobná, a tak by konečný návrh byl velmi podobný i kdybych jej vytvářel nezávisle na návrzích předchozích iterací projektu Dráček.

Strukturní model mých modulů je však i přesto značně odlišný. Za prvé byl původní návrh poměrně složitý na používání a za druhé obsahoval velké množství kódu, který se doslovně, či téměř doslovně, opakoval mezi jednotlivými moduly. Aby se této duplikaci kódu předešlo, rozhodl jsem se ve spolupráci s Ondřejem Slabým a Jaroslavem Štěpánem (kteří pracují taktéž na projektu

Dráček III a jejich bakalářská práce vzniká souběžně s touto) značně rozšířit knihovnu k modulům (jejíž základ vytvořil taktéž Michal Bureš) o novou funkcionalitu a především pak implementaci některých základních kamenů společných pro všechny moduly.

Struktura modulů bude dvouvrstvá. Důvodem pro to je snadnost údržby takovéto struktury. Důvodem pro nevyužití kompletní třívrstvé varianty je přílišná složitost takového řešení kontrastující s příliš malou odhadovanou velikostí vznikajících modulů, menší než aby třívrstvá struktura přinesla viditelné výhody.

3.2.1 Aplikační vrstva



Obrázek 3.1: Diagram struktury tříd datové vrstvy

V této vrstvě se bude odehrávat samotné skladování dat a operace s daty. Vzhledem k podobnosti struktur dat mezi jednotlivými modely je zde taktéž možné velkou část kódu přesunout do společné knihovny a tím zredukovat

duplicitu kódu (i riziko vzniku chyb). Tím se také sníží pracnost vytváření nových modulů.

Základem datové struktury bude třída Exercise, reprezentující celé zadání, se kterým bude modul či editor spuštěn. Tato třída bude obsahovat seznam jednotlivých cvičení. Jelikož budou však mít jednotlivé moduly různé struktury samotných cvičení, je v knihovně přítomna pouze základní abstraktní implementace cvičení AbstractSingleExercise, od které budou konkrétní cvičení dědit své vlastnosti.

Celé zadání je předáváno v konstruktoru třídě ExerciseManager, která se stará o jeho propojení s výsledky. Taktéž má za úkol postupné předávání nových cvičení prezenční vrstvě.

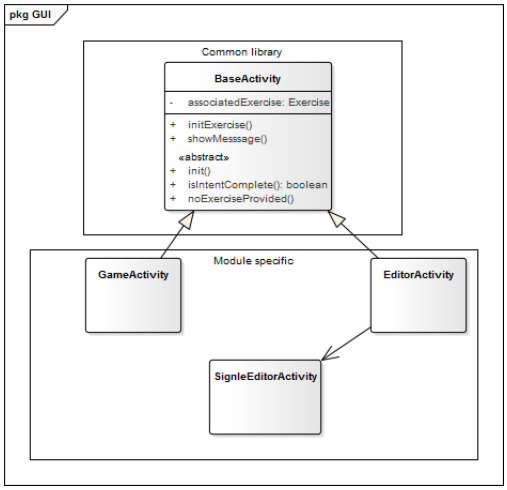
Dále se v knihovně ještě nachází třída Time, sloužící pro zaznamenávání času potřebného k vyřešení zadání a již dříve zmíněná třída Result, která se stará o zaznamenávání úspěšnosti žáka v úkolech.

Poslední nedílnou součástí knihovny je třída ResultWriter, která zapisuje výsledek žáka do XML formátu dle specifikace jádra.

V každém modulu je pak nezbytné zvlášť vytvořit třídy ExerciseReader pro načtení zadání z XML, ExerciseWriter pro zapsání do XML a SingleExercise pro uchovávání a práci s jednotlivými zadáními.

Podrobnější diagram tříd si můžete prohlédnout na obrázku 3.1.

3.2.2 Prezentační vrstva



Obrázek 3.2: Diagram struktury tříd GUI

Na rozdíl od aplikační vrstvy, jen velmi málo z prezentační vrstvy je možno přesunout do společné knihovny. Jedinou třídou z prezentační vrstvy, která se v knihovně nachází, je třída BaseActivity. Jedná se o abstraktní šablonu pro

třídy EditorActivity/GameActivity jednotlivých modulů. Je v ní navíc implementováno jednoduché testování konzistence dat předaných od modulu jádra a pomocné funkce. Veškeré ostatní části této vrstvy jsou natolik specifické pro jednotlivé moduly, že jejich implementace ve společné knihovně je buď přímo nemožná, nebo by bylo výsledné řešení mnohem pracnější a pro použití složitější než implementovat je zvlášť.

V jednotlivých modulech pak budou zvlášť implementovány minimálně tři třídy a to již zmíněné EditorActivity, tedy editor celého zadání, SingleEditorActivity, tedy editor jednotlivých úkolů ze zadání, a GameActivity, hlavní část modulu, se kterou žáci interagují.

Diagram tříd prezentační vrstvy se nachází na obrázku 3.2.

3.3 Procentuální hodnocení

Jednou věcí, kterou bylo u modulů nutno rozhodnout, je jakým způsobem se bude vypočítávat hodnocení žáka.

Nejjednodušší variantou je hodnocení pomocí poměru správně vyřešených cvičení vůči špatně vyřešeným. Při užití této varianty se však nezapočítá do výsledku, zda žák při řešení jedné úlohy udělal pouze jednu chybu, nebo jednoduše vyzkoušel veškeré možnosti.

Druhou variantou je pak přepočítávání přes celkový počet možných odpovědí. K tomu je ovšem nezbytné vypočítat před začátkem řešení počet všech možností, čímž se činí implementace zbytečně složitou. Navíc u některých modulů v této práci nemá takovéto řešení logický smysl.

Jako nejlepší variantu jsem proto vybral výpočet výsledku pomocí průměru procentuální úspěšnosti v jednotlivých dílčích cvičeních.

3.4 Rozložení polí ve cvičeních

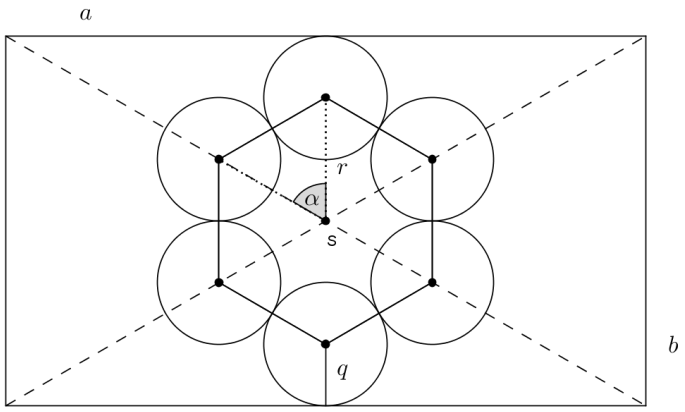
Specifikem aplikací navrhovaných pro systém Android je nutnost podporovat velké množství rozlišení a poměrů stran obrazovek.

Pro podporování rozdílných rozlišení je Java pro Android poměrně dobře vybavená. Má automatický převod fyzických pixelů na „pixely nezávislé na hustotě“ (density independent pixels - dp), podporu obrázků ve více různých rozlišeních (pro vykreslení je vždy vybrán nejvhodnější) a dalšími funkcemi, takže se programátor o tento aspekt nemusí příliš starat.

Druhý problém je však poněkud složitější. Absolutní pozicování grafických komponent je ze zřejmých důvodů nevhodné. Na druhou stranu i relativní (procentuální) rozmístění je při různosti poměrů stran poměrně neobratné. Navíc je i velmi málo podporováno, tudíž i poměrně složité. Tento problém je proto nezbytné řešit pomocí takzvaných „správců rozložení“ (Layout Manager).

Pro programování těchto modulů se nejvíce hodí lineární a relativní rozložení. Lineární rozložení umožňuje naskládat komponenty vedle sebe (případně pod sebe) a nastavit jim relativní „váhy“ a přesná velikost a umístění komponent je automaticky dopočítána pro dané zařízení. Relativní rozložení pak umožňuje libovolné umístění prvků na obrazovku relativně vzhledem k ostatním prvkům, nebo pomocí „okrajů“ (margin) vzhledem k pozici vrchního levého rohu rozložení.

3.4.1 Rozložení do kruhu



Obrázek 3.3: Náčrtek rozložení tlačítek do kruhu

Při navrhování jsem se setkal s jedním problémem. Při výběru z několika možností by bylo graficky pěkné, aby se možnosti objevily okolo stisknutého pole v kruhu (samy možnosti jako kruhová tlačítka). Pro zobrazení v kruhu však neexistuje ve standardních knihovnách (SDK 25, nejnovější v době psaní této práce) žádné vhodné rozložení, proto je nutné jej navrhnout.

Mějme tedy umístěnou obdélníkovou oblast okolo pole, kolem níž se mají možnosti zobrazit, vhodně posunutou, aby nezasahovala mimo obrazovku (umístění lze provést pomocí relativního rozložení a funkcí pro zjištění přesných souřadnic komponent), s velikostmi vodorovné strany a a svislé strany b a středem S. Chtěl bych rozmístit $n \geq 2$ ($n = 1$ je specifický, ale jednoduchý případ, a proto je řešen zvlášť) kruhových tlačítek tak, aby jejich středy tvořily pravidelný n-úhelník a tento n-úhelník měl střed ve středu obdélníkové oblasti. Z estetických důvodů by také měl první střed tlačítka ležet přímo nad středem S, tedy by přímka procházející spojnicí tohoto středu a S měla být kolmá na a. Pro znázornění případu šestiúhelníku si můžete prohlédnout náčrtek na obrázku 3.3.

Pro snadné vykreslování je nutné zjistit souřadnice středu S, vzdálenost r středů tlačítek od S, poloměr tlačítek q a středový úhel n-úhelníku α.

Středový úhel je dle základních vzorců pro pravidelné n-úhelníky $\alpha = \frac{2\pi}{n}$. Střed zřejmě leží v bodu $S = [\frac{b}{2}, \frac{a}{2}]$. Dále vybereme z obdélníku soustředný vepsaný čtverec (velikost strany $x = \min(a, b)$), v něm budou ležet všechna kruhová tlačítka. Je zřejmé, že aby byl takto vepsaný n-úhelník maximální, musí se q rovnat polovině strany n-úhelníku (aby se kruhy dotýkaly, ale nepřekrývaly) a zároveň i vzdálenosti prvního vrcholu n-úhelníku od vrchní vodorovné strany dříve vybraného čtverce. Z toho tedy můžeme vyvodit, že:

$$\begin{aligned} q &= q \\ \frac{x}{2} - r &= \sin\frac{\alpha}{2} + r \\ r &= \frac{\frac{x}{2}}{\sin\frac{\alpha}{2} + 1} \end{aligned}$$

Poloměr tlačítka je dle již výše použitého vzorce $\frac{x}{2} - r$.

Pro úplnost je tedy levý horní roh čtverce opsaného tlačítku (údaj používaný pro většinu vykreslovacích a umisťujících funkcí) s indexem i indexovaným od 0:

$$P_i = S + r * [\cos(\frac{\pi}{2} + i * \alpha), \sin(\frac{\pi}{2} + i * \alpha)]$$

a šířka i výška tlačítka jsou $x - 2r$.

3.5 Nápověda v modulech

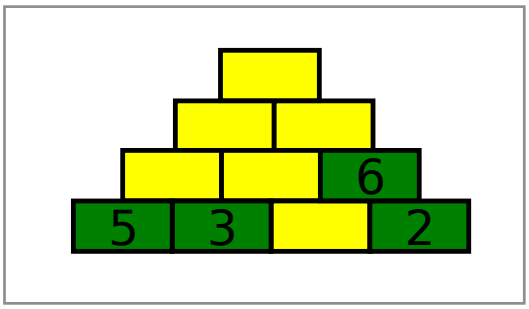
Existuje mnoho způsobů jak vytvořit nápovědu s obsluhou modulů. Základní možností je vytvořit jednotný dokument obsahující veškeré informace. To je však značně nevhodné pro děti a reálně hrozí, že tento dokument nikdo nebude číst.

Druhou možnou variantou je integrování jednoduché bodové nápovědy přímo do modulu. Tato nápověda by se zobrazila po kliknutí na tlačítko (kupříkladu se symbolem otazníku). Toto je již vhodnější, ale i toto neřeší problém, že takováto nápověda obsahuje množství textu. Navíc by buď mohlo docházet k podvádění s množstvím času spotřebovaným na vyřešení úkolu (v případě zastavování časoměru při prohlížení nápovědy), nebo naopak prodlužování času, během zjišťování žáka jak hru hrát.

Jako nejvhodnější možnost jsem vybral zobrazení nápovědy před započetím řešení prvního úkolu. Navíc by však vzhledem k zacílení aplikace na děti bylo vhodné, aby takováto nápověda obsahovala co nejméně textového popisu a naopak co nejvíce znázornění (například grafického). Z testování jsem se poučil, že dlouhé vysvětlování před započetím řešení je při opakovaném hraní naopak rušivé a kazí dojem z hry. Proto jsem se nakonec rozhodl pro zobrazení jednoho obrázku s převážně grafickým ztvárněním nápovědy, doplněným

textem kde to bude nezbytné či vhodné pro snazší pochopení. Tento obrázek zmizí při kliknutí na něj a od té chvíle může žák začít s řešením prvního úkolu.

3.6 Modul Pyramidy



Obrázek 3.4: Wireframe modul Pyramidy

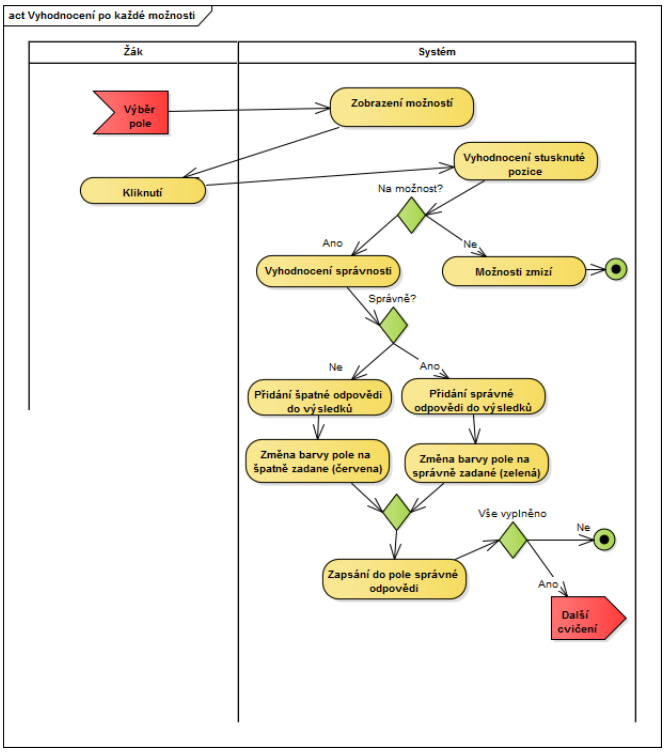
Nejdříve je třeba navrhnout, jakým způsobem se budou výsledky do pyramidy doplňovat. První možnost je přímý vstup z numerické klávesnice, která by se objevila po kliknutí na ještě nevyplněné pole. Druhou možností je výběr z N nabídnutých výsledků. Druhá varianta dává žákům nápovědu při vyplňování, čímž značně úkol zjednodušuje. Na druhou stranu první varianta je příliš statická a tak i nudnější. Z tohoto důvodu jsem se rozhodl pro druhou možnost.

Dále je nutné rozhodnout, zda bude správnost hodnocena po doplnění celé pyramidy, či zda bude hodnocena po doplnění každého pole zvlášť. Zřejmým favoritem je však druhá metoda. Za prvé proto, že je mnohem snazší ukázat žákovi, kde udělal chybu, za druhé proto, že to zabraňuje situaci, kde žák udělá v jednom místě pyramidy drobnou chybu a zbylé řešení je už celé špatně.Za třetí kvůli zkušenostem s testováním aplikací z Dráčka II, kde se vyhodnocování po celém úkolu ukázalo jako snadný zdroj problémů. Podrobnější rozpracování zvolené varianty naleznete v diagramu aktivit na obrázku 3.5.

Jednoduchý náčrtek vzhledu modulu si můžete prohlédnout na obrázku 3.4.

3.6.1 Chování modulu

- Spuštění jádrem zobrazí první úkol
- Kliknutí na prázdné pole → zobrazí možnosti výběru
- Kliknutí na možnost → vyhodnocení, zápis správného výsledku do pole, barevné označení dle správnosti



Obrázek 3.5: Diagram aktivit, zadávání a vyhodnocení řešení

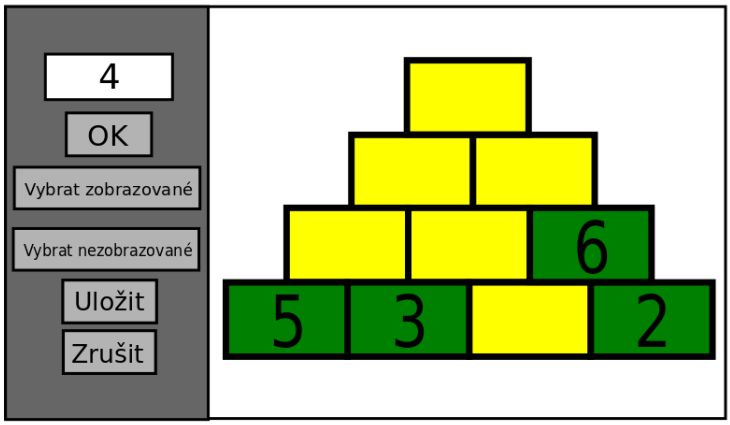
- Kliknutí na volnou plochu nebo již vyplněné pole → zmizení tlačítek možností
- Vyplnění všech polí → Jedna vteřina pauza, a poté přejití na další úkol

3.6.2 Původní návrh editoru

Editor tohoto modulu má dva návrhy. První návrh (zvláště jeho uživatelské rozhraní) se ukázal být značně nepraktický a neintuitivní z důvodů, které jsou uvedené v příští kapitole. Za účelem názornosti a vyjasnění některých mých rozhodnutí zde však napíši i návrh původní.

Po otevření editoru se zobrazí seznam úkolů v zadání, s nímž byl spuštěn. Úkoly bude možno mazat, přidávat a měnit (tak jak je již ve funkčních požadavcích). Při přidání nového úkolu do zadání, či úpravě existujícího, se otevře editor úkolů.

Cílem je vytvořit editor, který umožní co nejflexibilnější vytváření úkolů s co nejmenší pracností. Základní parametr pro vytváření je počet řad. Po jeho zadání se objeví nevyplněná pyramida. Kliknutím na pole (dlaždici) v pyramidě se zobrazí numerická klávesnice, pomocí které je možné zapsat do pole čísla.



Obrázek 3.6: Wireframe editoru úkolu pro modul Pyramidy

Pro jednoduchost vyplnění jsem se také rozhodl, že editor bude dopočítávat pole, která lze z již zadaných hodnot doplnit. Tedy stačí například zadat pouze spodní řadu a editor již doplní zbytek.

Dále by bylo vhodné, aby učitelé mohli zadat pole, která se zobrazí v zadání. Obměnou pozic zadaných polí lze totiž dosáhnout různé obtížnosti úloh. Navíc zvyšuje pestrost cvičení a tedy i prodlužuje dobu, než začne děti nudit. K tomu budou v editoru sloužit 2 tlačítka pro zobrazování a nezobrazování v zadání. Po kliknutí na tlačítko a následném kliknutí na pole pyramidy se nastaví tento parametr, přičemž v náhledu se to projeví nastavením barvy pole.

Samozřejmě by v editoru neměla chybět ani možnost uložení a zrušení změn. Při uložení se navíc provede kontrola, zda lze vytvořené cvičení vytvořit a zda jsou zadané hodnoty správné a úplné. V případě nějakého nedostatku se objeví okénko s chybovou hláškou, úkol se neuloží a je jej nutné opravit.

Grafický návrh editoru jednotlivých cvičení lze nahlédnout na obrázku 3.6.

3.6.3 Nový návrh editoru

Původně navržený boční panel s tlačítky byl značně nepraktický pro různé velikosti obrazovky. Při některých velikostech ubíral příliš mnoho prostoru a samotný náhled zadání byl kvůli němu příliš malý. Neformální testování také ukázalo, že uživatelé mají tendenci nejdříve klikat na pole, která chtějí upravit a až poté případně hledat nějaké další kontrolní mechanismy pro úpravu. To ukazovalo na neintuitivnost původního návrhu. Zadávání počtu řad po-

mocí klávesnice bylo poměrně pomalé a nepraktické a některé funkcionality a možnosti zřejmě v návrhu chyběly, proto bylo nutné jej upravit.

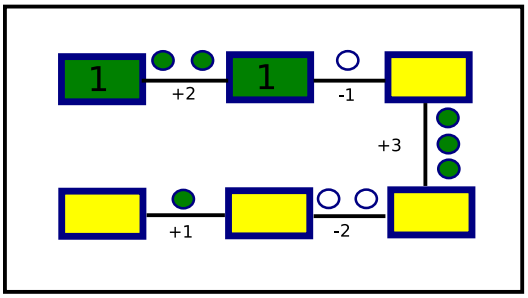
Rozhodl jsem se proto provést tyto změny:

- Boční panel z návrhu zcela odebrat a jeho prvky částečně rozmístit nad a pod náhled pyramidy (protože na mobilních zařízeních bývá mnohem více místa ve vertikálním směru a scrollování v tomto směru je také obvyklejší).
- Přidat pole pro zadání názvu úlohy, aby bylo možné úlohy rozeznat v seznamu úloh v jednotlivých zadáních.
- K poli pro zadání počtu řad přidat posuvník a poli ponechat pouze informativní účel.
- Zadání dopočítávat na stisknutí tlačítka a ne až při ukládání (pro snadnost ověření výsledku a možnost změn).
- Při dopočítání označit všechna nově vyplněná pole jako nezobrazovaná v zadání.
- Po vybrání pole zobrazit možnosti změn: Editovat číslo, Zobrazovat v zadání, Nezobrazovat v zadání.
- Při manuálním zapsání výsledku do pole jej defaultně označit za zobrazované v zadání, aby při kombinaci manuálního zadávání a automatického dopočtu nebylo většinou třeba měnit stav (zobrazování, nezobrazování) polí a vzniklo smysluplné zadání
- Pro všechna pole nezobrazovaná v zadání doplnit automatické vytváření špatných odpovědí (protože zadávání špatných odpovědí bylo zdlouhavé a nepříjemné provádět manuálně).

Tento nový návrh se zdál být značně použitelnějším, a proto u něj již bylo setrváno.

3.7 Modul Hadi

U tohoto modulu bylo nejdříve třeba určit, jakým způsobem budou jednotlivá pole rozmístěna. V úvahu přicházelo libovolné rozmístění zadané uživatelem a rozmístění takzvaným tabulkovým způsobem, tedy pravidelně do řad a sloupců. Výhodou první varianty je větší volnost při tvorbě cvičení. Nevýhodami této varianty jsou pak vyšší pracnost (jak při implementaci, tak i při tvorbě nových zadání) a zvláště pak problémy s rozdílnou velikostí obrazovky zařízení pro Android. Zatímco na některých zařízeních může být zadání pěkné a přehledné, na jiných může být nezobrazitelné, nebo přinejmenším



Obrázek 3.7: Wireframe editoru úkolu pro modul Hadi

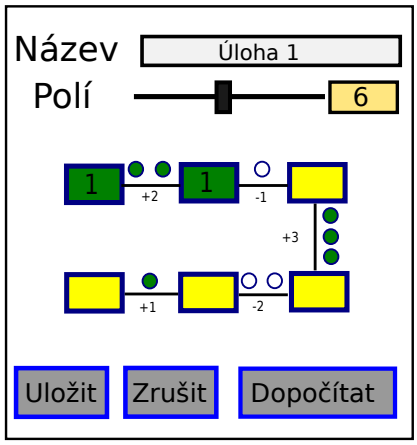
nepřehledné atd. Z těchto důvodů jsem se rozhodl pro řešení s tabulkovým rozložením.

Další problém, který musel být rozhodnut, byla metoda vkládání odpovědí a kontrola řešení. Jedná se o přibližně stejnou problematiku, která byla řešena již v předchozím modulu. Ze stejných důvodů, jako u modulu Pyramidy, jsem se rozhodl pro tutéž metodu, tedy výběr z navržených možností a kontrola po vyplnění každého pole. Zvolením stejného řešení je navíc zachována konzistence mezi (z grafického hlediska) podobnými moduly, čímž je pro žáky snazší orientovat se v nich a při implementaci je možné se vyhnout provádění zbytečné práce.

Grafický návrh se nachází na obrázku 3.7.

Základní chování modulu je stejně jako u modulu pyramidy.

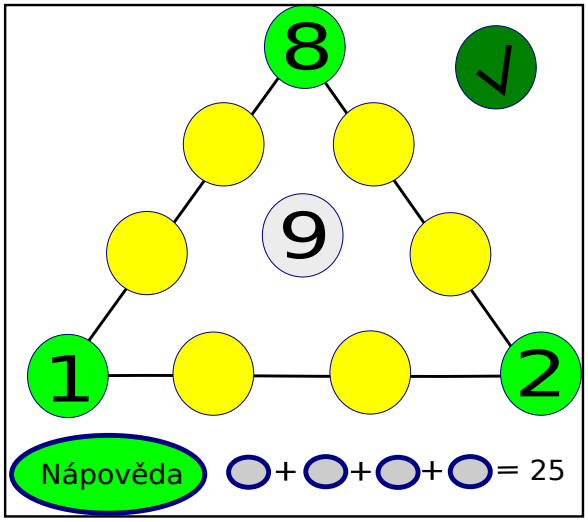
3.7.1 Editor



Obrázek 3.8: Wireframe modul Hadi

Při návrhu editoru jsem využil veškeré zkušenosti z předchozího modulu. Jedinou větší změnou v samotném návrhu je přidání volitelné operace nad spojnici polí. Bude zde možno zadat přičítání a odčítání. Rozsah přičtení/odečtení musí být také limitován, aby bylo možno smysluplně a jednoduše graficky tuto operaci vyjádřit. Jako pro dotyková zařízení nejvhodnější způsob zadávání jsem se proto rozhodl pro zadávání pomocí posuvníku (slideru), čímž se lze i vyvarovat mnoha možných chyb se špatným ošetřením rozmezí operací.

3.8 Modul Magické trojúhelníky



Obrázek 3.9: Wireframe modul Magické trojúhelníky

Kvůli rozdílnosti tohoto modulu oproti dříve zmíněným modulům bylo nezbytné pozměnit některé základní prvky jeho fungování.

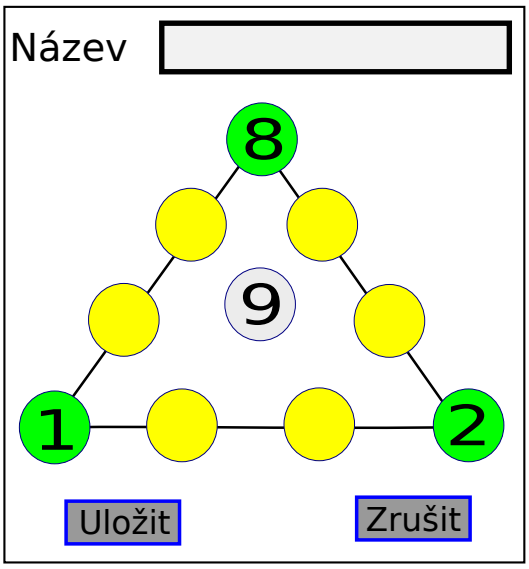
Vyhodnocování správnosti řešení musí být prováděno až po doplnění celého zadání, protože zadání mohou umožňovat několik různých správných řešení.

Počet polí jsem zvolil jako konstantních 10. Tím se zaručí, že bude možné vytvořit z polí trojúhelník a bude i možné vytvořit smysluplné zadání. Navíc při tomto počtu bude každá číslice 0-9 zadávaná právě jednou.

Tento modul navíc nabízí možnost nápovědy. I zde je nutné rozhodnout, jakým způsobem se její zobrazení projeví do výsledného hodnocení. Krajní varianty jsou nulové projevení se a označení celého úkolu za špatně vyřešený. Samozřejmě ani jedna z nich není příliš vhodná. Jako vhodný kompromis jsem se proto rozhodl pro snížení dosažitelného hodnocení u daného úkolu na polovinu, pokud byla nápověda zobrazena.

Grafický návrh se nachází na obrázku 3.9.

3.8.1 Editor



Obrázek 3.10: Wireframe editor modulu Magické trojúhelníky

Editor tohoto modulu je o něco jednoduší než v předchozích případech, protože z návrhu zmizí možnost zadávání počtu polí. Vyplňování polí jsem se rozhodl zde provádět pomocí výběru z nabídky možností (stejně jako je v herní části), protože je to mnohem rychlejší a praktičtější než zadávání pomocí klávesnice. Na rozdíl od ostatních cvičení zde není nutné vyplnit všechna pole, je však nutné zadat dostatečný počet polí, aby existoval právě jeden součet stran. Konkrétně tedy, aby zadání bylo řešitelné a zároveň v každém možném řešení měly všechny strany tentýž součet.

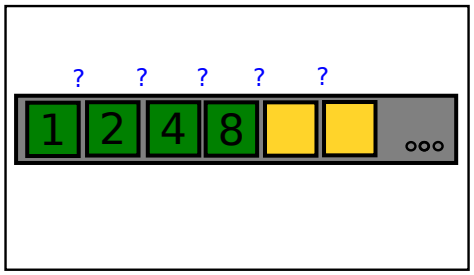
Tlačítko na automatické doplňování v tomto modulu nedává přílišný smysl, a proto bylo také odebráno.

Jednoduchý grafický návrh je na obrázku 3.10.

3.9 Modul Řady

Herní část tohoto modulu je z velké části stejná jako u modulu Hadi. Liší se především grafickým zpracování a množstvím podporovaných operací mezi vedlejšími poli.

Zbývá jen rozhodnout, jakým způsobem bude správnost odpovědi kontrolována. První možností je kontrola dle vzorce, na jehož základě je řada vytvořena. Druhou možností je kontrola podle předpřipravených správných výsledků, které by byly součástí zadání. První varianta je samozřejmě méně datově náročná, tato výhoda je však při současné rychlosti připojení a velikosti paměti zanedbatelná (rozdíl maximálně v řádu kB). Tato varianta však



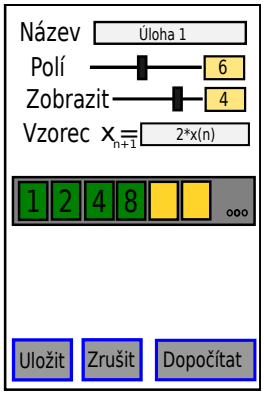
Obrázek 3.11: Wireframe modul Řady

nutně potřebuje mít specifikovány všechny použitelné operace přímo v herním modulu. Toto omezení by se mohlo ukázat být značně nepříjemné v případě budoucího rozšíření o další operace, kdy by muselo dojít k aktualizování nejen editoru, ale i všech herních částí pro žáky. Z tohoto důvodu byla zvolena druhá možnost.

Po dokončení celého cvičení (ať již špatně nebo správně) se žákovi zobrazí nad poli správný vzorec, podle kterého byla řada vytvořena.

Grafické znázornění je na obrázku 3.11.

3.9.1 Editor



Obrázek 3.12: Wireframe editoru úkolů pro modulu Řady

Jak už bylo zmíněno dříve, v editoru bude krom názvu cvičení možno zadat i celkový počet polí a počet polí zobrazených v zadání.

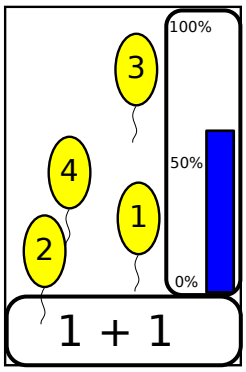
Co se týče zadávání samotného vzorce, existují dvě základní možnosti. Jednou možností je doplnění do předpřipraveného schématu, například $x_{n+1} = A * x_n + B$. Tento postup by umožnil automatické doplňování a snadnější vytváření zadání. Na druhou stranu by značně omezil množství vytvořitelných zadání (v tomto případě by například nešla zadat ani Fibonacciho posloup-

nost). Další možností je nechat uživatele (učitele) zapsat libovolný text pro zobrazení žákovi po dokončení cvičení, a vyplnit všechna pole ručně.

Jako ideální řešení byl zvolen kompromis. V editoru bude možno vybrat z několika předpřipravených schémat, případně i zapsat vzorec ručně, avšak automaticky doplnit pole bude možné pouze v prvním případě.

Grafický návrh editoru cvičení si můžete prohlédnout na obrázku 3.12.

3.10 Modul Balónky



Obrázek 3.13: Wireframe modul Balónky

Tento modul je nejrozdílnější od všech ostatních modulů, a proto je při jeho návrhu nezbytné zavést do architektury některé zcela nové prvky.

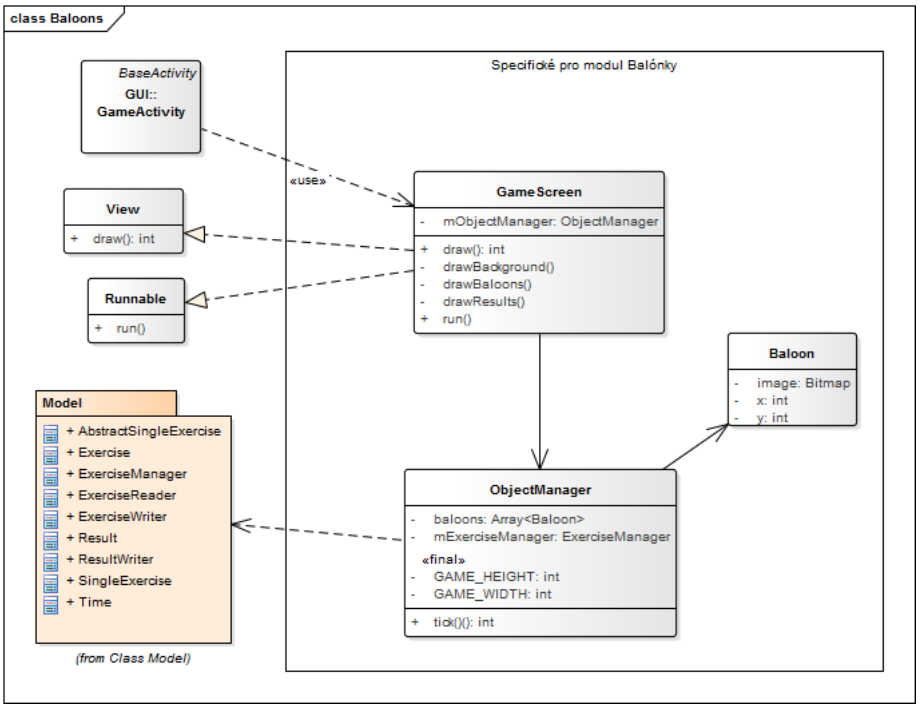
Základní ukládání dat pomocí:

$$Zadání \rightarrow Cvičení \rightarrow Špatné\ odpovědi + Správné\ odpovědi$$

může být zachován. Na druhou stranu pro chod hry je nutné mít uložené i pozice jednotlivých balónků, jejich obrázky a text na nich. Dále je nutné mít uložený stav hry a základní parametry. Pro uchování těchto informací byla proto vytvořena jakási „mezivrstva“ mezi původní prezentační a aplikační vrstvou (návrh se tak velmi přiblížil trojvrstvému návrhovému vzoru).

Konkrétně se jedná o třídu Baloons, která, jak už název napovídá, obsahuje informace o stavu balónku a třídu ObjectManager, která obsahuje veškeré informace o současném stavu hry (veškeré objekty ve hře). Navíc je zodpovědná za veškeré změny stavu hry pomocí metod Tick (volána periodicky, posouvá čas ve hře a provádí s ním spojené úkony jako je pohyb balónků), hit (kliknutí na plochu s případným zasažením balónku).

Dále je nutné rozhodnout, která třída bude zodpovědná za vykreslování hry a periodické posouvání herního času. Jako nejsnazší se jeví umístit tuto funkcionalitu do třídy ObjectManager. To by ale znamenalo vytvoření značně



Obrázek 3.14: Diagram tříd modul Balónky

obrovské třídy zodpovědné za mnoho různých věcí, navíc z dvou různých vrstev, což z návrhového hlediska není nejvhodnější. Druhou variantou je vložení této funkcionality přímo do třídy GameActiviy. Toto řešení již nerozbíjí vrstvy aplikace (vše z prezentační vrstvy), ale také tvoří jednu obří třídu. Navíc by bylo poměrně obtížné v budoucnosti jakkoliv změnit rozložení uživatelského rozhraní (například přidáním dalších tlačítek). Třetí možností je vytvoření nové třídy dědící od View (komponenta grafického návrhu), a celou tuto třídu pouze zasadit do grafického návrhu třídy GameActivity. Poslední varianta nemá krom lehce vyšší programovací náročnosti (a delšího kódu) žádnou velkou nevýhodu, a proto jsem se rozhodl ji použít.

Podrobnější diagram nově přidaných tříd si můžete prohlédnout na obrázku 3.14. Grafický návrh si můžete prohlédnout na obrázku 3.13.

3.10.1 Průběžné hodnocení

Vzhledem k herní povaze tohoto modulu by bylo pěkné, aby žáci měli možnost kontrolovat své výsledky průběžně. K tomuto účelu by měl sloužit pruh (jakýsi teploměr) na pravém okraji obrazovky. Prozatím jsem se však zabýval pouze vyhodnocováním výsledků po dokončení celého zadání, takže je nezbytné nově navrhnout i vzorec pro určování výsledku v průběhu hry.

První možností je výpočet výsledku pouze z dosud dokončených cvičení se

zanedbáním celkového počtu. Tato varianta ovšem produkuje značně nestabilní výsledky, kde se hodnocení zvláště v prvních několika cvičeních pohybuje nahoru a dolů v řádech desítek procent. Zvláště vysoké poklesy hodnocení při jediné chybě by mohly u malých dětí vést ke zbytečnému stresu a zklamání. Navíc tato varianta neposkytuje žádné vodítko co se týče pokroku v celém zadání. Z těchto důvodů jsem se rozhodl tuto variantu zavrhnout.

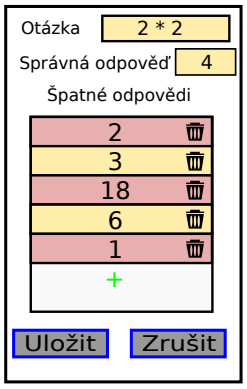
Druhou možností je použití přímo celkového výsledku, tedy se všemi i ještě nevypracovanými cvičeními, které by se do jejich vyplnění hodnotily jako špatně odpovězené. Tato varianta je již mnohem lepší. Naznačuje pokrok žáka v řešení zadání a navíc se ani příliš nesnižuje hodnocení při jednotlivých chybách. Na druhou stranu chyby provedené při řešení se ve výsledku cvičení neprojevují až do správné odpovědi, takže z pohledu žáka se může hodnocení chovat neočekávaně (někdy za správnou odpověď přibude více procent, někdy kvůli chybám méně) a hodnocení chyb není vidět.

Třetí varianta je vylepšením druhé. Pro hodnocení je stále využíván celkový výsledek, ale aby se okamžitě zohledňovaly chyby, je v současnosti řešený úkol hodnocen jako „napůl správně“ a výsledek je vypočten jako:

$$konečný\_výsledek + \frac{1}{2} * \frac{1}{celkový\_počet\_cvičení} * \frac{1}{počet_chyb + 1}$$

Tato varianta již okamžitě započítává chyby do výsledného hodnocení, zároveň si ale v převážné míře zachovává výhody druhé varianty. Z těchto důvodů jsem ji vyhodnotil jako nejlepší a použil.

3.10.2 Editor



Obrázek 3.15: Wireframe editoru pro modul Balónky

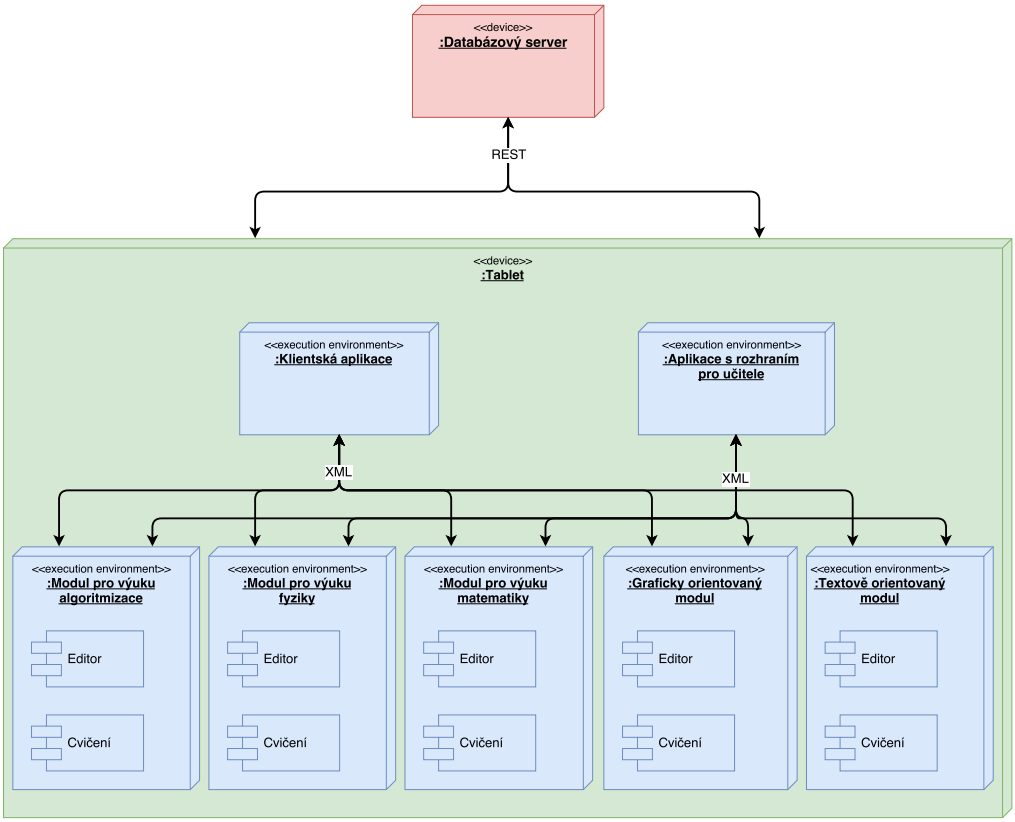
U tohoto modulu není u jednotlivých cvičení nutný název, protože pro jasnou identifikaci cvičení v editoru zadání jasně postačí samotná otázka (vzorec nebo cokoliv jiného). Pro každé cvičení je pak nutné zadat správnou odpověď

na zadanou otázku a špatné odpovědi. Ty se budou zobrazovat v seznamu a bude je možné libovolně přidávat, měnit a odebírat.

V ostatních aspektech se tento editor příliš neliší od ostatních editorů, jak si také můžete prohlédnout na obrázku 3.15.

Kapitola 4

Implementace



Obrázek 4.1: Schéma nasazení aplikace Dráček převzato z [6]

V této kapitole se budu zabývat tím, jak vytvořené moduly nasadit, používat, upravovat a jak vytvářet nové. Nebudu zde detailně popisovat samotný zdrojový kód, protože bych tak zabíhal do přílišných detailů a práce by tak zbytečně nabrala desítky stránek. Pokud chcete zjistit více o implementaci

konkrétních prvků, veškeré zdrojové kódy se nachází na přiloženém paměťovém médiu.

4.1 Schéma nasazení

Celý systém aplikace Dráček se skládá ze serverové části, dvou různých klientských aplikací (učitelská, žákovská), které mohou běžet na prakticky neomezeném počtu zařízení paralelně, a zásuvných modulů (tedy i modulů vytvářených v této práci), které běží na stejném zařízení jako klientské aplikace a zajišťují konkrétní naučný obsah. Podrobněji rozkreslený diagram nasazení se nachází na obrázku 4.1.

Pro nasazení modulů bylo nejdříve nutné zprovoznit server a následně upravit klientské aplikace, protože obsahovaly chyby znemožňující smysluplnou práci s nimi.

4.2 Příručka pro vývojáře

4.2.1 Instalace

V této kapitole budu popisovat základní kroky k vytváření nových modulů a editaci současných.

Nejdříve je třeba nainstalovat základní programové vybavení. Zejména se jedná o:

- **Android IDE** - integrované vývojářské prostředí pro vytváření nových aplikací. Veškeré moduly popsané v této práci byly vytvořeny pomocí Android Studia, nejvíce používaného IDE pro vytváření aplikací pro systém Android. Je vyvíjeno společností Google, která taktéž vyvíjí systém Android, a lze ho zdarma stáhnout přímo z oficiálních stránek developer.android.com, případně pro uživatele systému linux i v oficiálních repozitářích. Alternativně lze použít například Eclipse.

- **Android IDK** - balíček nástrojů pro kompilaci kódu, debugování a mnoho jiného. Je taktéž dostupné zdarma z internetu. Veškeré moduly v této práci byly vytvářeny pomocí SDK verze 25.

- **Emulátor mobilního zařízení** - pro pohodlný vývoj je taktéž vhodné nainstalovat nějaký emulátor mobilních zařízení. Android Studio má takovýto emulátor již zabudovaný, i zde je ale nezbytné jej nastavit a nainstalovat do něj systém vhodné verze (pro testování modulů z této práce byl převážně využit systém Android 6.0 Marshmallow, verze API 23).

- **Společná knihovna modulů** - obsahuje základní třídy pro usnadnění vytváření nových modulů. Je dostupná přes systém GitLab FIT ČVUT.

Tento repozitář musí být naklonován do složky nadřazené složce s modulem (defaultně AndroidStudioProjects). Přesný postup přidání této knihovny k modulu se nachází na wiki tohoto repozitáře.

4.2.2 Změna grafického provedení modulů, nápovědy

Grafické provedení a s ním související nápovědu lze snadno změnit pomocí nahrazení souborů s obrázky v adresáři:

```
app/src/main/res/drawable
```

Pro větší zásahy do grafického provedení je nutné změnit soubory rozložení android aktivit v adresáři

```
app/src/main/res/layout
```

Pro nejdetailnější úpravy (bez většího přepisování zdrojového kódu) existují v některých aktivitách konstanty určující drobné detaily zobrazení. Názvy těchto aktivit končí koncovkou Activity a naleznete je v podadresářích adresáře:

```
app/src/main/java
```

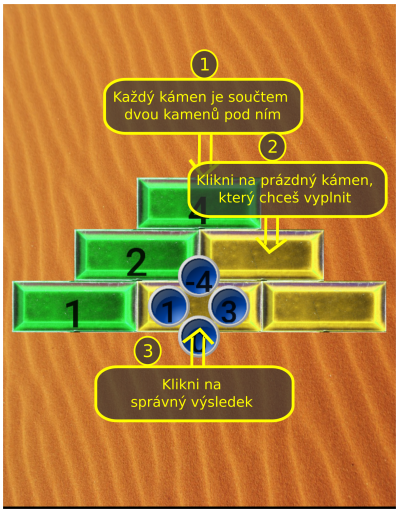
4.3 Uživatelská příručka

4.3.1 Instalace modulů

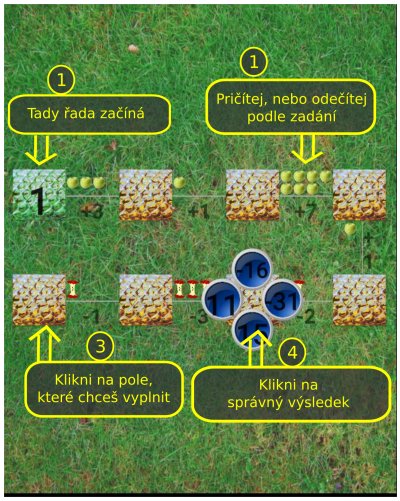
Nejdříve je nutné stáhnout a nainstalovat uživatelského (případně učitelského) klienta. Prozatím je toto možné udělat pouze manuálně (viz kapitolu 4.3.2), v budoucnu by mělo být možné stáhnout a nainstalovat aplikaci přímo přes Google Play.

Dalším krokem je instalace jednotlivých modulů (her). Toto lze provést pomocí uživatelského klienta, který by měl již veškeré zbylé kroky provést automaticky. Při současném stavu klientské aplikace je však nutné manuálně umístit instalační soubor modulu <ID>.apk do složky Download/dragon na interní uložiště zařízení. Například u modulu s ID 11 bude cesta k instalačnímu souboru Download/dragon/11.apk. Přesnější návod i důvody tohoto stavu můžete nalézt v práci zabývající se tvorbou uživatelského klienta. V případě problémů s touto variantou, lze moduly instalovat manuálně (dle návodu v kapitole 4.3.2).

Pro nově přidávané moduly (poprvé přidávané na jakékoliv zařízení) je také nutné manuálně doplnit řádek s názvem a cestou k editoru a herní části modulu do databáze na serveru (stejným způsobem, jako jsou již přidané ostatní moduly).



Obrázek 4.2: Nápověda pro modul Pyramidy



Obrázek 4.3: Nápověda pro modul Hadi

4.3.2 Manuální instalace aplikace

K manuální instalaci aplikace do zařízení se systémem Android je třeba provést tyto kroky:

1. Zkontrolovat verzi operačního systému (pro moduly vytvořené v rámci této bakalářské práce jsou podporovány verze Android 5.0 Lolipop a vyšší)
2. V nastavení systému povolit instalaci aplikací z neznámých zdrojů
3. Stáhnout instalační soubor (s koncovkou .apk) do zařízení
4. Ve správci souborů otevřít složku s instalačním souborem a otevřít jej (obvykle kliknutí)
5. Schválit požadovaná oprávnění a zahájit instalaci

4.3.3 Základní princip a řešení úkolů

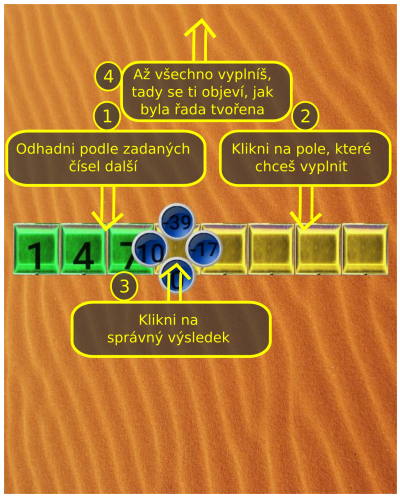
Pyramidy

Toto je modul na takzvané sčítací pyramidy.

Úkolem žáka je vyplnit všechna prázdná pole čísly tak, aby hodnota každého pole byla součtem dvou polí přímo pod ním.

Vyplňování se provádí kliknutím na pole, které chce žák zadat, a výběrem správné možnosti. Na vyplnění každého pole má žák právě jeden pokus. Po zadání výsledku se vždy vyplní pouze správné řešení a barvou (grafickým vyjádřením) se vyjádří správnost nebo nesprávnost zvoleného řešení.

Obrázek 4.4: Nápověda pro modul Magické trojúhelníky



Obrázek 4.5: Nápověda pro modul Pyramidy

Obrázek 4.6: Nápověda pro modul Hadi

Hadi

Toto je modul na takzvané číselné řetězce.

Úkolem žáka je vyplnit všechna prázdná pole čísly tak, aby hodnota každého pole odpovídala hodnotě pole předchozího po provedení operace napsané mezi nimi.

Vyplňování a vyhodnocování se provádí stejně jako u modulu Pyramidy.

Magické trojúhelníky

Toto je modul na takzvané magické trojúhelníky.

Úkolem žáka je vyplnit všechna prázdná pole tak, aby součet hodnot polí na všech třech stranách trojúhelníku byl shodný.

Vyplňování se provádí kliknutím na pole, které chce žák zadat, a výběrem správné možnosti.

Pro dokončení a vyhodnocení úkolu musí žák kliknout na tlačítko pro vyhodnocení (označeno graficky „fajfkou“).

Řady

Toto je modul na takzvané logické řady.

Úkolem žáka je odhalit vzorec, podle kterého je vyplněno několik prvních čísel řady a doplnit dle tohoto vzorce všechna prázdná pole.

Vyplňování a vyhodnocování se provádí stejně jako u modulu Pyramidy.

Balónky

Toto je modul na obecné procvičování matematických znalostí, převážně vhodný pro jednoduché výpočty a základní matematické znalosti.

Úkolem žáka je vyřešit otázku v dolní části obrazovky a kliknout na balónek se správnou odpovědí dříve, než tento balónek odletí mimo obrazovku.

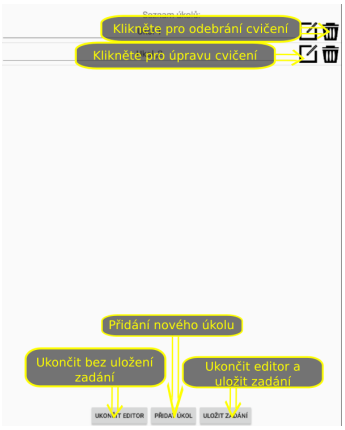
V pravé části obrazovky se průběžně zobrazuje dosažené skóre (procentuální hodnocení).

4.3.4 Společná správa úkolů

Přidávání a úprava nových zadání se provádí pomocí Učitelské aplikace.

Po otevření zadání (nového i upravovaného) se otevře editor zadání (stejný pro všech 5 modulů vytvořených v této práci). Pomocí tlačítka „přidat úkol“ lze přidat nové cvičení. Po zavření editoru cvičení se uživatel (pedagog) vrátí zpět do tohoto editoru. Jsou-li již nějaké úkoly přítomny, je možno pomocí tlačítek vpravo od názvů úkolů cvičení mazat a upravovat.

Při upravování a vytváření nových úkolů se otevře editor specifický pro konkrétní modul. V tomto editoru jsou tlačítka „uložit“ pro uložení úkolu a „zrušit“ pro zrušení všech provedených změn. Pokud bylo stisknuto tlačítko „zrušit“ při vytváření nového úkolu, žádný nový úkol se nevytvoří, pokud při úpravě existujícího úkolu, úkol zůstane v původní podobě. Pro urychlení



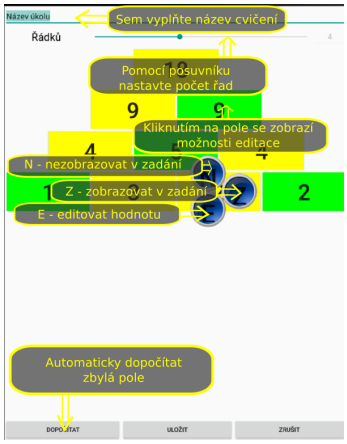
Obrázek 4.7: Nápověda pro společnou správu úkolů

práce je v některých editorech možné stisknout tlačítko dopočítat, čímž se provede automatické doplnění všech zjistitelných hodnot (jsou označeny jako nezobrazované).

Po dokončení práce se nové zadání odešle pomocí tlačítka „uložit zadání“. Pro vymazání veškerých provedených úprav a návrat do učitelského klienta je nutné kliknout na tlačítko „ukončit editor“.

4.3.5 Vytváření a upravování nových úkolů

Pyramidy



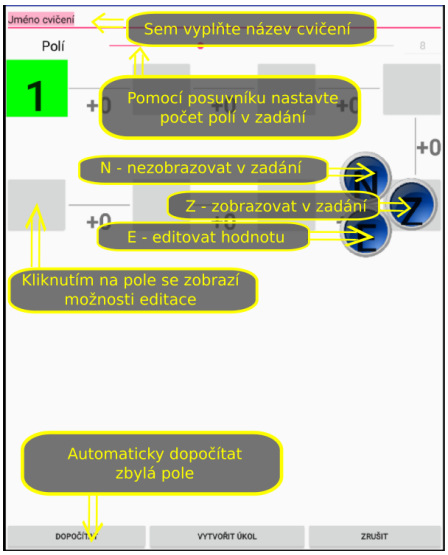
Obrázek 4.8: Nápověda editoru modulu Pyramidy

Do pole na vrcholu editoru je nutné vyplnit název úkolu.

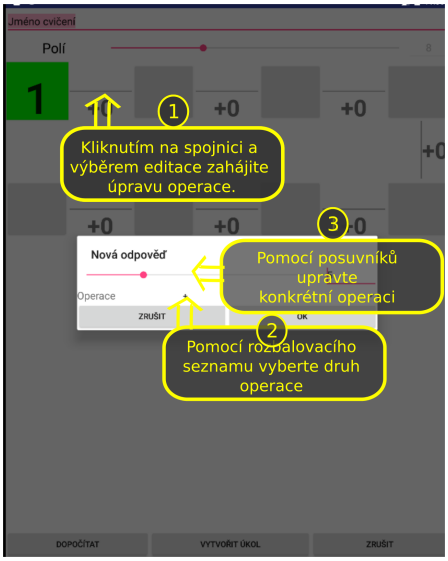
Pomocí posuvníku níže se nastavuje počet řad pyramidy, který lze také kontrolovat v poli napravo od posuvníku.

Aby bylo možné zadání uložit, musí být ze zobrazovaných hodnot jednoznačně určitelné správné řešení a takovéto řešení musí existovat.

Hadi



Obrázek 4.9: Nápověda editoru modulu Hadi



Obrázek 4.10: Nápověda pro editaci operací modulu Hadi

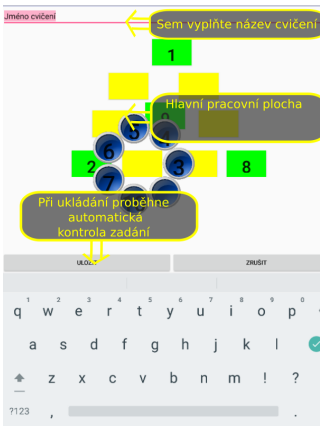
Do pole na vrcholu editoru je nutné vyplnit název úkolu.

Pomocí posuvníku níže se nastavuje počet polí v zadání.

Upravovat se dají nejen hodnoty polí, ale i operace mezi nimi.

Po výběru editování operace se pomocí posuvníku nastavuje velikost změny a pomocí rozbalovacího seznamu o jakou operaci se jedná (sčítání/odčítání).

Magické trojúhelníky



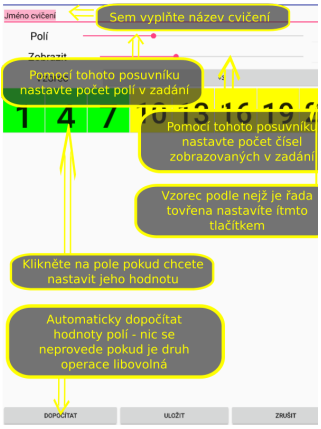
Obrázek 4.11: Nápověda editoru modulu Magické trojúhelníky

Do pole na vrcholu editoru je nutné vyplnit název úkolu.

Do polí lze vložit zadání, každé číslo pouze jednou, tak aby ve všech směrech byl stejný součet.

Stisknutím tlačítka Doplnit se doplní veškerá dopočitatelná pole.

Řady



Obrázek 4.12: Nápověda editoru modulu Řady

Do pole na vrcholu editoru je nutné vyplnit název úkolu.

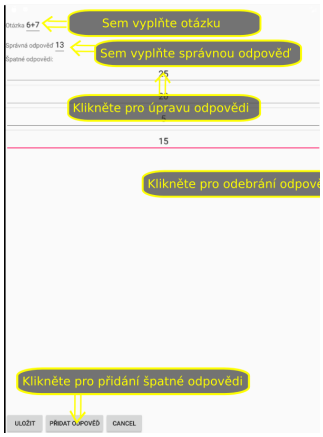
Pomocí posuvníku níže (Polí) se nastavuje počet polí v zadání.

Pomocí dalšího posuvníku (Zobrazit) se nastavuje počet polí zobrazených žákům v zadání (žáci uvidí tento počet vyplněných čísel).

Kliknutím na tlačítko níže (Vzorec) se otevře okno pro zadávání vzorce, podle kterého bude řada tvořena.

Konkrétní šablonu vzorce je možné vybrat pomocí rozbalovacího seznamu (Operace) v dolní části okna. Šablona „libovolná“ umožňuje přímé textové zadání jakéhokoliv vzorce, ale v takovémto případě není možné provést automatický dopočet polí a veškerá pole musí být vyplněna ručně.

Balónky



Obrázek 4.13: Nápověda editoru modulu Balónky

Do pole na vrcholu editoru (otázka) je nutné napsat otázku, která se zobrazí jako zadání úkolu a bude sloužit i pro odlišení jednotlivých úkolů v editoru.

Do pole níže (správná odpověď) se zadává správná odpověď na zadanou otázku.

Níže se nachází seznam špatných odpovědí. Špatné odpovědi lze přidat pomocí tlačítka „přidat odpověď“. Lze je odebrat kliknutím na ikonku koše a upravit kliknutím na odpověď.

Výběr možnosti editace v editoru úkolů Po kliknutí na pole zadání editoru se objeví výběr ze tří možností. Tyto možnosti jsou:

- Editovat (E) - zadání nového správného výsledku pole, po stisknutí se objeví pole pro zápis čísla, po zadání čísla se pole automaticky označí za zobrazované v zadání
- Nezobrazovat pole v zadání (N) - označí pole jako nezobrazované v zadání, tedy jako jedno z polí pro vyplnění žáky
- Zobrazit pole v zadání (Z) - označí pole jako zobrazované v zadání, tedy hodnotu tohoto pole vidí žáci při řešení úkolu.

4.4 Omezení vyvinutých prototypů

Veškeré základní navržené funkcionality modulů byly implementovány. Hlavním nedostatkem modulů je uživatelské rozhraní, u kterého je pro nasazení nezbytné změnit grafické provedení. Moduly taktéž nebyly testovány s uživateli, a proto samozřejmě nebyly provedeny ani úpravy nezbytné pro snadnost používání, které by z takovéhoto testování vyplynuly. Více o testování naleznete v sekci 5.

Prohlédnutím ke zdrojovým kódům lze také odhalit různé návrhové nedostatky. Jedná se o některé zbytečné duplicity kódů a další drobné prohřešky, proti „pěknému“ kódu. Většina z nich vznikla z důvodu mé nedostatečné zkušenosti s programováním pro Android a nebyly odstraněny převážně z časových důvodů.

Kapitola 5

Testování

Vzhledem k velké navázanosti kódu na grafické prostředí by bylo poměrně složité a především nadměrně pracné (zvláště s přihlédnutím k dalším očekávaným změnám grafického návrhu) vytvářet plně automatické testy. Proto jsem se rozhodl provádět testy především manuální.

Testování probíhalo po téměř celou dobu vývoje. V první fázi byly testy hlavně prováděny na emulátoru zabudovaném v Android Studiu, převážně z důvodu snadnosti provádění a kontroly výsledků u tohoto způsobu. Později jsem pro větší přesnost prováděl testy i na zařízení Nexus 10 (tablet) zapůjčeném od FIT ČVUT a pro kontrolu fungování na zařízeních s jinými parametry i na smartphonu Samsung Galaxy 6.

5.1 Unit testy

Unit testy jsou testy, při kterých jsou testovány jednotlivé komponenty s cílem ověřit, že se každá část chová tak, jak byla navržena. Tyto testy obvykle provádí sám programátor [13].

Toto testování probíhalo pomocí takzvaných „whiteboxových testů“, což bylo více věcí nutnosti, než rozhodnutí (je velmi složité dělat sám testy bez znalosti kódu, který píši také já).

Vzhledem k malé komplexitě jednotlivých modulů a velkému zaměření na uživatelské rozhraní jsem se rozhodl nevyužívat žádný framework. Toto se ovšem ukázalo být problematické, protože navržené testy byly často připravené pro manuální ověření při současném stavu aplikace, a tak je nebylo možné opakovat v budoucnosti a ani jich většinu smysluplně zachovat (testy byly implementované jako součást samotného kódu aplikací). Tento koncept jsem později již neměnil (protože bych musel znovu implementovat již jednou úspěšně provedené testy, což by stálo spoustu času s minimálními výsledky), ale pro vytváření dalších modulů by bylo vhodné nějaký vhodný framework využít.

5.2 Integrační testy

Integrační testování je fází testování softwaru, při kterém jsou jednotlivé části spojeny a testovány jako celek. Cílem tohoto testování je odhalení chyb v interakci mezi jednotkami [13].

Integrační testy jsem prováděl ve dvou fázích a zaměřovaly se na propojení a komunikaci modulů s žákovskou a učitelskou aplikací. Konkrétně se jednalo o testování správného předávání zadání do modulů a výsledků, případně nových zadání zpět do žákovské/učitelské aplikace.

První fází bylo testování s „dummy“ objekty, které interagovaly s moduly místo jádra a zasílaly jim požadavky dle specifikace jádra. V této fázi se nevyskytly žádné závažné problémy.

Druhou fází bylo přímé propojení s jádrem. Zde se však vyskytlo problémů hned několik. Tyto problémy byly zapříčiněny převážně nepřipraveností jádra na provoz. Pro jeho uvedení do provozu bylo nejen nutno místy upravit kód, ale i doprogramovat některé jeho části. Ze strany modulů se zde vyskytly pouze drobné nedostatky, které byly okamžitě opraveny.

5.3 Testování použitelnosti

Testování použitelnosti je způsob zjištění snadnosti používání softwaru otestováním jej se zkutečnými uživateli. Uživatelé jsou požádáni, aby provedli úkoly (obsluhu aplikace). Obvykle jsou při tom sledováni, aby se zjistilo, kde narazí na problémy a co je pro ně matoucí [14]. Pro účely této práce se toto testování značně kryje se systémovým testováním.

5.3.1 Vlastní testování

Před tím, než mohly být moduly předány k testování na žácích, bylo nejdříve třeba alespoň částečně je otestovat vlastními silami. Rozhodl jsem se proto pro manuální systémové testování. V tomto testování jsem prováděl kontrolu funkčnosti dle funkčních požadavků. Hlavním cílem bylo nalézt nejzřejmější nedostatky a chyby v softwaru, aby bylo možné získat z dalšího testování co nejvíce a ne pouze tyto velké, ale na první pohled zřejmé problémy.

5.3.2 Uživatelské testování

Na doporučení vedoucího se z organizačních důvodů bude uživatelské testování konat až po odevzdání této práce. Nenastanou-li další organizační problémy, měly by výsledky být uvedeny při obhajobě této práce.

Testovací scénáře, vstupní a výstupní dotazníky a minimální požadavky na znalosti žáků pro testování jsou uložené na této stránce:
`https://drive.google.com/drive/folders/0B7jDcvN8Qu01Ui1jM2xlRnROVTg`.

Závěr

V práci jsem se zabýval analýzou současného stavu výukových aplikací pro děti i konkrétně aplikace Dráček, která vznikla jako součást bakalářských prací studentů FIT ČVUT Dráček I a Dráček II. Dále jsem se zabýval úpravou zásuvného modulu vzniklého v projektu Dráček II a jeho otestováním s žáky základní školy.

V pozdější fázi práce jsem se zabýval analýzou a vytvářením požadavků na 5 nových zásuvných modulů. Ve fázi návrhu jsem se zabýval převážně návrhem herní i editorové části všech těchto modulů a hodnocením úspěšnosti žáků. Velký důraz jsem kladl na položení obecného základu pro vytváření nových zásuvných modulů s cílem usnadnit budoucí vývoj. Toto se projevilo především navržením společné knihovny.

V práci jsem také popsal způsob nasazení aplikace Dráček, ovládání vytvořených modulů a základní kroky pro vývojáře, nutné k vytvoření a integraci nových modulů, případně k úpravě modulů v rámci této práce vzniklých.

V neposlední řadě jsem se také věnoval testování modulů.

Rád bych zde také uvedl, jak si vytvořené moduly vedly při zhodnocení pomocí stejné metriky, jakou byly hodnoceny aplikace v analytické části. Z tabulky 5.1 vyplývá, že celkově dosáhly moduly hodnocení 25/30, což je ve srovnání s průměrným výsledkem (10/30) i nejlepším výsledkem (13/30) současných aplikací velmi výrazný posun a stanovený cíl vylepšení současného stavu se tedy podařilo splnit.

Tato práce splnila všechny body zadání. Za zvláště povedenou považuji úpravu původního modulu, což se projevilo i u uživatelského testování. I vytvoření základu pro další vývoj se zdařilo a celá aplikace Dráček se tímto posunula značně blíže k nasazení do reálného provozu. Na druhou stranu se ale sluší zmínit i věci, které nedopadly natolik dobře. Převážně se jedná o grafické zpracování modulů, které je velmi strohé a nedokonalé, a testování, které neprobíhalo dostatečně intenzivně, a v navržených modulech tak mohou být neodhalené chyby.

Do budoucna lze na této aplikaci ještě provést mnoho další práce, ta však

Tabulka 5.1: Matematika příklady zhodnocení

jazyk	čeština	(5b)
chyby, problémy	dobré, ale může obsahovat skryté chyby	(4b)
nápaditost	střední	(3b)
velikost	více oblastí, v rámci aplikace Dráček mnoho modulů	(5b)
podpora pedagogů	plná, výsledky i úprava zadání	(5b)
gamifikace	názorné grafické ztvárnění, částečně herní prvky (baloons)	(3b)
celkové hodnocení	25/30	

již svým rozsahem přesahuje rozsah této bakalářské práce. Například je možné vytvořit další moduly nebo zajistit funkčnost aplikace pod dalšími operačními systémy než Android. Bylo by také vhodné přidat do všech modulů hlasové pokyny a animace. Také by v budoucnu bylo vhodné některé moduly rozšířit, kupříkladu modul Magické trojúhelníky, který je velmi konkrétní, a tak má více než dost prostoru pro takovéto rozšíření.

Literatura

[1] Kids Learning Math Lite. https://play.google.com/store/apps/details?id=com.honeybee.android.kidsmathlite&hl=en, viděno: 2017-01-02.

[2] Learn Math. https://play.google.com/store/apps/details?id=nuprotec.aprendematematicas&hl=en, viděno: 2017-01-02.

[3] Child Learn Math 1st 2nd grade. https://play.google.com/store/apps/details?id=air.childLearnMath.edugames.vn&hl=en, viděno: 2017-01-02.

[4] Matematika příklady. http://www.pmq-software.com/sw/cz/android/priklady-z-matematiky/, viděno: 2017-01-02.

[5] Filip, O.: *Výuková aplikace Dráček II – Serverová část a rozhraní pro učitele*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarofého inženýrství, 2016.

[6] Štěpán, J.: *Zásuvné moduly aplikace Dráček III - výuka fyziky*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarofého inženýrství, 2017.

[7] Slabý, O.: *Zásuvné moduly aplikace Dráček III - výuka základů algoritmizace*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarofého inženýrství, 2017.

[8] Jeřábek, J.: *Rámcový vzdělávací program pro základní vzdělávání: s přílohou upravující vzdělávání žáků s lehkým mentálním postižením*. Praha: Výzkumný ústav pedagogický v Praze, 2005, iSBN 80-87000-02-1.

[9] Bureš, M.: *Výuková aplikace Dráček II – zásuvné moduly II*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarofého inženýrství, 2016.

[10] Mazel, M.: *Dragon II — Plugins I.* Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra softwarofého inženýrství, 2016.

[11] Horník, L.: *Matematika hrou: Sbírka úloh pro 1.roč.ZŠ*. Praha: Státní pedagogické nakladatelství, 1994, iSBN 80-04-26393-3.

[12] a Miroslav Červenka, J. M.: *Matematika hrou: sbírka úloh pro 2. ročník základní školy*. Praha: Státní pedagogické nakladatelství, 1993, iSBN 80-04-26362-3.

[13] Software Testing Fundamentals. `http://softwaretestingfundamentals.com/unit-testing/`, viděno: 2017-05-05.

[14] Experience Ux. `http://www.experienceux.co.uk/faqs/what-is-usability-testing/`, viděno: 2017-05-05.

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

SDK Software development kit

API Application programming interface

IDE Integrated development environment

RVP Rámcový vzdělávací program základního vzdělávání

Příloha **B**

Obsah přiloženého paměťového zařízení

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF