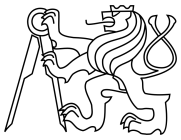
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Zásuvné moduly aplikace Dráček III - výuka základů algoritmizace
Student:	Ondřej Slabý
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 základů algoritmizace.
4. Navrhněte alespoň 5 nových modulů, které budou podporovat výuku základů algoritmizace.
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 6. února 2017

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

Bakalářská práce

Zásuvné moduly Dráček III – výuka základů algoritmizace

Ondřej Slabý

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 kolegům Jaroslavu Rybovi a Jaroslavu Štěpánovi za spolupráci.

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

Slabý, Ondřej. *Zásuvné moduly Dráček III – výuka základů algoritmizace*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce je pokračováním projektu Dráček, který se zaměřuje na podporu výuky dětí prvního stupně základní školy pomocí tabletů. Důležitou částí projektu je právě aplikace na tablety s operačním systémem Android. Hlavním cílem práce bylo vytvořit nové výukové moduly, které rozšiřují aplikaci o výuku základů algoritmizace. Druhotným cílem bylo otestovat uživatelské rozhraní stávajících i nových modulů s žáky základní školy. Vlastní práce obsahuje analýzu, návrh a implementaci pěti nových modulů pro aplikaci Dráček. Součástí každého modulu je kromě samotného cvičení také editor pro tvorbu nových a úpravu existujících zadání. Moduly jsou připraveny k běžnému používání, nicméně před jejich nasazením bude vhodné sjednotit jejich grafické provedení.

Klíčová slova výuková aplikace, testování s dětmi, testování uživatelského rozhraní, Dráček, modul, výuka základů algoritmizace, Android, tablet

Abstract

This thesis is a continuation of the project Dráček, which aims to support elementary school education using mobile devices. The project consists of a modular Android application, which then uses specific modules to provide exercises for specific education fields. The main goal of this thesis was to create new modules, extending the current module set to provide exercises focused on learning the basics of algorithmization. The secondary goal was to subject both new and old modules to usability testing with elementary school pupils. The thesis contains analysis, design and implementation of five new modules for the application. Each module also contains an editor, which enables the user to create and modify the tasks for each exercise. Created modules are ready for use, although their user interface should first be unified with all other modules.

Keywords education application, testing with children, usability testing, Dráček, module, basics of algorithmization, Android tablet

Obsah

Seznam obrázků

Úvod

Děti se od stále nižšího věku učí používat elektronická zařízení, jako jsou počítače a v poslední době také tablety. Tato zařízení se dětem v některých případech dostávají do rukou již na základních školách. Problémem je však nedostatek aplikací pro tablety, které by se zaměřovaly na specifická odvětví výuky a zároveň by byly určené pro žáky základních škol. Projekt Dráček se zabývá právě tvorbou takových aplikací.

Dráček I byl projekt, jehož cílem bylo vytvoření výukové aplikace pro děti s poruchami učení. Po jeho úspěšném dokončení vznikl navazující projekt Dráček II, jehož hlavním úkolem bylo převést aplikaci Dráček I na mobilní zařízení s operačním systémem Android. Jednotlivá témata výuky jsou řešená pomocí zásuvných modulů pro hlavní aplikaci.

Projekt Dráček III má za úkol rozšířit soubor modulů o další témata, jmenovitě matematiku, fyziku a základy algoritmizace. Tato práce se zaměřuje na základy algoritmizace. Výsledné moduly by kromě usnadnění výuky tématu na základních školách měly také zvýšit zájem žáků o programování.

Zároveň s touto prací se na další dvě zmíněná témata soustředí práce kolegů, se kterými jsem spolupracoval:

- Jaroslav Štěpán – výuka fyziky [4]
- Jaroslav Ryba – výuka matematiky [11]

Kapitola 1

Cíl práce

Cílem této práce je vytvořit alespoň 5 nových zásuvných modulů pro aplikaci Dráček II, které budou zaměřené na výuku základů algoritmizace. Zároveň je cílem podrobit nově vytvořené moduly a jeden z existujících modulů uživatelskému testování s dětmi prvního stupně základní školy. Výsledky tohoto testování spolu s analýzou současných řešení a výuky tématu, na které je tato práce zaměřena, budou základem pro tvorbu požadavků pro nové moduly. Podle vytvořených požadavků bude vytvořen návrh a následně implementace pětice nových modulů, které budou podrobeny vhodnému testování, včetně uživatelského testování za spolupráce s Usability laboratoří na Fakultě informačních technologií, ČVUT.

Kapitola 2

Analýza

Při výběru zaměření jednotlivých nových modulů jsem vybral kromě algoritmizace i další témata pevně spjatá se základy algoritmizace: matematickou logiku a převody mezi číselnými soustavami. Na všechna tři témata celkem vytvořím pět nových modulů. Pro téma algoritmizace a matematické logiky vytvořím po dvou nových modulech a pro téma převodů mezi soustavami vytvořím jeden modul.

2.1 Výuka témat na základních školách

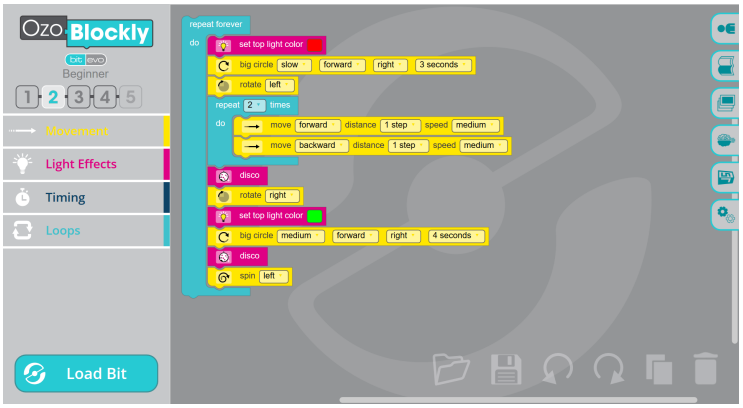
Všechny moduly budou zaměřené na první stupeň základní školy, stejně jako moduly, které byly dosud pro projekt Dráček vytvořeny. Tato kapitola se zabývá upřesněním zvolených témat a zjišťováním jejich zastoupení na základních školách, zejména na prvním stupni.

2.1.1 Základy algoritmizace

V tomto tématu jsem se zaměřoval zejména na tvorbu jednoduchých algoritmů v podobě skupiny po sobě jdoucích příkazů. Tyto příkazy by měly mít jednoduchou, intuitivní formu. Výsledný algoritmus by měl být tvořen za účelem dosažení předem určeného cíle, který by byl pro jedno cvičení pevně daný. V rámcovém vzdělávacím programu pro základní vzdělání je sice výuka informačních a komunikačních technologií zařazena do povinné výuky pro 1. a 2. stupeň, nicméně popsané části problematiky se týká pouze jeden bod cílového zaměření této vzdělávací oblasti, "schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení."[12]

Výuka algoritmizace tímto způsobem na základních školách probíhá téměř výhradně až na druhém stupni. Často je realizována pouze formou volitelného předmětu [13] nebo kroužku [14], ve kterém si děti mohou vyzkoušet například jednoduché programování robotů. Na obrázku 2.1 je ukázka programu OzoBlocky, který lze použít k programování robota používaného v kroužku

programování. Pro první stupeň neexistuje předmět, který by se alespoň tvorbou jednoduchého algoritmu zabýval.



Obrázek 2.1: Snímek programu OzoBlocky [1]

2.1.2 Matematická logika

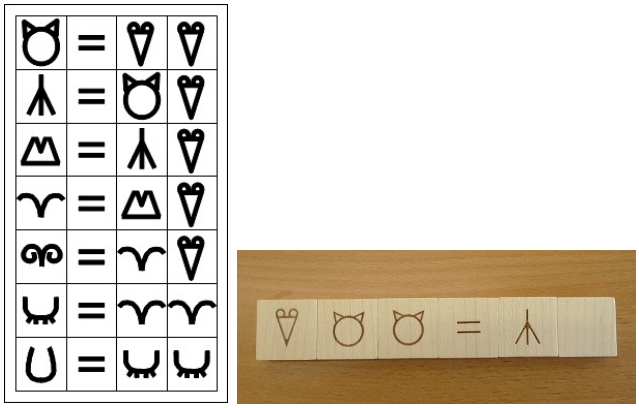
Pro toto téma jsem zvolil výuku základních logických operátorů, jako jsou například logické operátory AND a OR. Jejich použití je relativně jednoduché a lze pomocí nich tvořit nejen jednoduché podmínky, jejich kombinací lze vytvořit také složitější logické výrazy. Rámcový vzdělávací program se o logice zmiňuje v sekci pro matematiku a její aplikace. Je zde kladen důraz na rozvíjení logického uvažování žáků. Jde zde však o jiné odvětví logiky, vyučované často formou doplňování číselných řad [12].

Na základních školách se logické operátory prakticky nevyučují. Pokud ano, tak až na druhém stupni součástí doplňujícího vzdělávacího oboru formou povinně volitelného předmětu. Tento předmět se zaměřuje na obecné téma matematické logiky od pojmů, jako je výrok, pravdivostní hodnota nebo implikace, po řešení jednoduchých příkladů. Jeho cílem je žáky do problematiky povrchově uvést, naučit je význam jednotlivých pojmů a naučit je řešit příklady pomocí tabulky pravdivostních hodnot [15]. Předmět se tedy zabývá více tématy, než by pro první stupeň bylo vhodné.

2.1.3 Převody mezi soustavami

V tomto tématu jsem zjišťoval stav výuky převodů mezi číselnými soustavami, zejména s bázemi 2, 8 a 16, na základních školách. Rámcový vzdělávací program se touto problematikou nezabývá [12], nedá se tedy předpokládat velké zapojení tématu na prvním stupni základních škol.

Téma je také vyučováno až na druhém stupni základní školy, kde se jedná zejména o součást výuky předmětů zaměřených na informatiku a práci s počítačem, jde tedy o převody do soustavy s bází 2 [16]. K vytvoření vhodného



Obrázek 2.2: Tabulka převodů a ukázka příkladu cvičení Děda Lesoň [2]

cvičení pro převody soustav budu používat materiály založené na Hejného metodě výuky matematiky, zejména cvičení Děda Lesoň, které se zaměřuje na řešení rovnic. Čísla jsou reprezentována ikonou místo číslice [17], obvykle jako na levé části obrázku 2.2. Obvyklým úkolem ve cvičení je například rozdělení jedné sady ikon do dvou tak, že obě sady mají stejnou hodnotu. Ukázka takového cvičení formou rovnice je na pravé části obrázku 2.2. Dalším příkladem je přidání či odebrání ikony z jednoho ze dvou sad tak, že obě sady mají stejnou hodnotu.

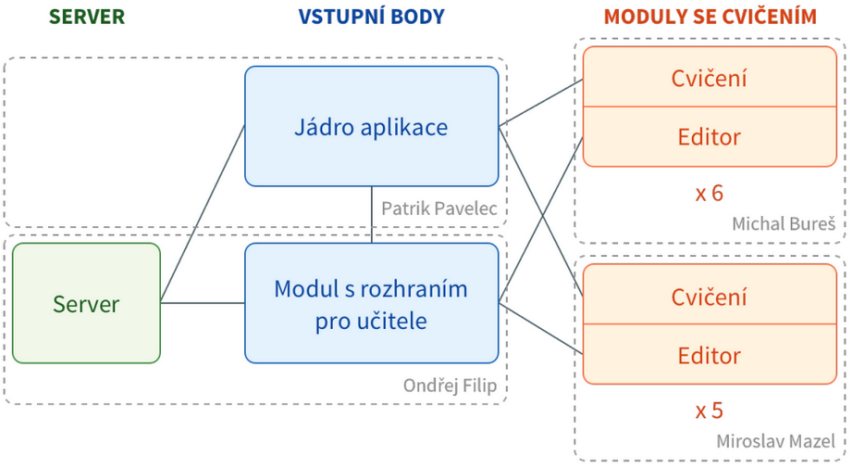
Druhou možností výuky převodů číselných soustav je již dříve zmíněný doplňující vzdělávací program, opět formou povinně volitelného předmětu pro druhý stupeň základní školy. Předmět sice nemá výraznou spojitost s programováním (nezabývá se moc soustavami, jejichž základ je mocnina dvou), na druhou stranu však zahrnuje i například historický pohled na číselné soustavy a další témata [15].

2.2 Analýza modulů Dráčka II

Jedním z cílů projektu Dráček II bylo vytvořit první sadu modulů pro výslednou aplikaci na platformu Android. Tyto moduly jsou zaměřené výhradně na práci s poruchami učení, které se u dětí mohou vyskytovat [18]. Tvorbou modulů se zabývají dvě práce projektu Dráček II, jejichž výsledkem bylo dohromady devět modulů.

2.2.1 Architektura modulů

Všechny moduly Dráčka II jdou rozdělené na dvě části: editor a samotné cvičení. Tyto části jsou spouštěné jádrem, které zajišťuje nejen stahování samotných modulů, ale také získání jednotlivých zadání pro právě zvolené cvičení ze serveru. Získané zadání následně předává modulu cvičení, který umožní uživa-



Obrázek 2.3: Schéma aplikace dráček [3]

teli zadání vyplnit. Výslednou úspěšnost uživatele spolu s časem, jaký uživateli vyplňování cvičení zabralo, vrací jednotlivé moduly zpět jádru, které je následně ukládá na server. Zadání, které jsou vytvořena či upravována editorem jsou rovněž posílána zpět jádru k uložení, čímž jsou pak zpřístupněna všem uživatelům. Rozdělení a komunikace mezi jednotlivými částmi je znázorněna na obrázku 2.3.

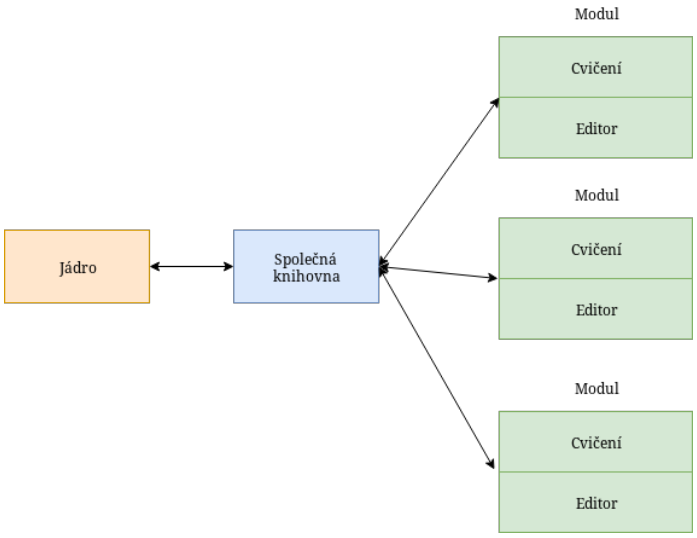
2.2.2 Oddělení do knihoven

Hlavní dobrou stránkou obou prací projektu Dráček II, které se zaměřují na tvorbu modulů, je vytvoření knihovny pro společné funkce všech modulů. Jde například o části komunikace s klientskou aplikací, dialogy, aj. Toto rozdělení je vyobrazeno na obrázku 2.4.

V práci Miroslava Mazla navíc autor oddělil i další části modulů, například zobrazování hodin, od samotných modulů do samostatných knihoven, čímž zvýšil přehlednost projektu a umožnil zobrazení hodin použít i v jiném modulu, pokud by to bylo potřeba. Tento přístup bude vhodné použít i v modulech projektu Dráček III, zejména oddělení komunikace s klientskou aplikací do vlastní knihovny.

2.2.3 Nedostatky modulů

Všem modulům lze vytknout jisté nedostatky, které by měly být před jejich použitím odstraněny. Těmto nedostatkům je třeba se v nových modulech vyhnout.



Obrázek 2.4: Rozdělení aplikace do knihovny a modulů

- **Nekompatibilita s jádrem** V době dokončení první sady modulů ještě nebyla dokončena klientská aplikace, se kterou měly moduly komunikovat za účelem získávání a ukládání zadání a hlášení výsledků cvičení. Důsledkem je, že před použitím je třeba všechny moduly upravit tak, aby správně s konečnou verzí klientské aplikace komunikovaly. Jedná se hlavně o načítání zadání. Vzhledem k tomu, že během projektu Dráček III už je klientská aplikace hotova, problémy s komunikací by se v nových modulech neměly vyskytovat.

- **Neodladěné kritické chyby** Druhým hlavním problémem jsou nedostatky na straně uživatelského rozhraní. V některých modulech je nemalá část grafického rozhraní, která při dotyku způsobuje neočekávané ukončení aplikace. Podobnému problému je třeba v nových modulech zamezit, nejlépe testováním interakce se všemi částmi grafického rozhraní.

- **Pomalé reakce** V některých modulech je problémem pomalá reakce na určité vstupy. Modul nedává uživateli najevo, že něco zpracovává a není tak jasné, jestli nejde o chybu.

- **Nízká tolerance přesunů a dotyků** Některé moduly vyžadují přesnou interakci. V opačné případě nedojde ke správnému zaznamenání odpovědi, což vzbuzuje dojem, že bylo cvičení vyplněno špatně, i když tomu tak nemuselo být.

2.3 Testování modulů Dráčka II

Moduly Dráčka II nebyly v době jejich tvorby otestovány po stránce interakce dětí s uživatelským rozhraním, pouze po stránce tvorby a úpravy modulů učiteli. Bylo tedy potřeba takové testování provést, abychom se mohli poučit z případných problémů, které by se u modulů mohly objevit.

2.3.1 Testovaný modul

K testování jsem si vybral modul hodiny, který se zabývá výukou analogových hodin. Princip cvičení, které modul realizuje, spočívá v tažení hodinové, popřípadě minutové ručičky na správné místo podle aktuálního zadání.

Na ciferníku hodin mohou být zobrazeny čísla reprezentující hodiny. Mimo ciferník pak mohou být čísla minut. Tento způsob byl autorem zvolen, aby bylo možné zobrazit minuty i hodiny najednou [18]. Před podrobením modulu uživatelskému testování je potřeba grafickou stránku modulu upravit tak, aby byl pro děti, se kterými bude testování prováděno, atraktivnější.

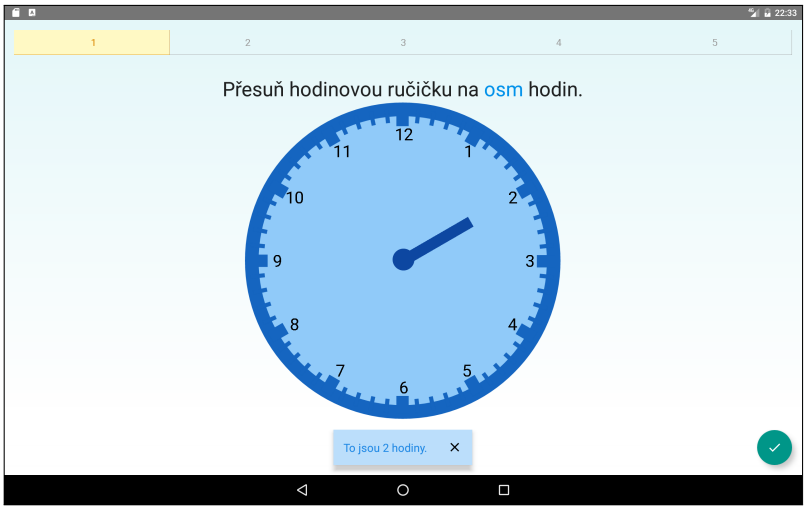
Dne 24.11.2016 proběhlo uživatelské testování s dětmi prvního stupně základní školy, při kterém jsem zpozoroval řadu problémů s tímto modulem.

2.3.2 Testovací formuláře

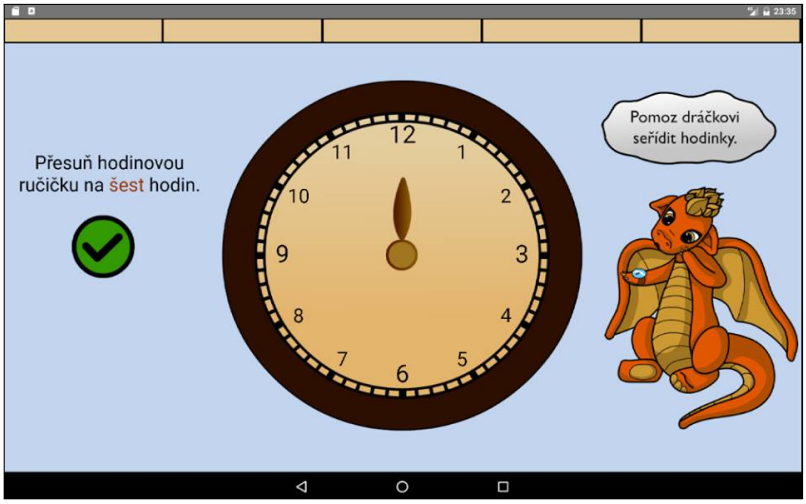
Předtím, než mohlo být testování zahájeno bylo nutné vytvořit vstupní a výstupní dotazníky pro testované uživatele. Cílovou skupinou byly děti prvního stupně základní školy. Součástí projektu Dráček II byla vytvořena sada formulářů právě pro uživatelské testování s dětmi, nicméně výsledné formuláře byly zbytečně dlouhé a složité pro děti na vyplnění. Nově jsme vytvořili jak vstupní tak výstupní formuláře. Nové vstupní formuláře se zabývají hlavně tím, jak je dítě seznámeno s tabletem a hraním na chytrých zařízeních a tím, jak je dítě seznámeno s učivem, na které jsou testované moduly zaměřené. Do výstupních formulářů jsme zahrnuli otázky, které se zabývají obecným dojmem ze cvičení a schopností pochopit, co je po dítěti požadováno. Zároveň je zde zahrnuta důležitá poslední otázka, mající za cíl zjistit, zda by dítě rádo cvičení vyplňovalo i mimo školu. Všechny dotazníky jsou součástí příloh. následující otázky:

2.3.3 Úprava modulu před testováním

Grafická stránka modulu Hodiny byla upravena za účelem uživatelského testování, na obrázku 2.5 je stará verze modulu a na obrázku 2.6 je nově upravená verze. Cílem úpravy bylo zpříjemnit vzhled modulu pro děti přidáním avatara Dráčka a dalšími úpravami.



Obrázek 2.5: Původní vzhled modulu Hodiny



Obrázek 2.6: Nový vzhled modulu Hodiny

2.3.4 Výsledky testování modulu Hodiny

Díky uživatelskému testování jsem na modulu hodiny zjistil následující nedostatky:

- **Přísný požadavek na přesnost dotyku** Hlavním problémem byly příliš úzké ručičky hodin, díky čemuž dítě ručičku prstem často minulo.

- **Matoucí posun hodinové ručičky** Často dětem dělal problém také posun hodinové ručičky, který znázorňuje uplynulou část hodiny podle

nastavených minut. Děti nevěděly, která hodina je nastavená a proč jim nejde nastavit přesně hodina, kterou chtějí.

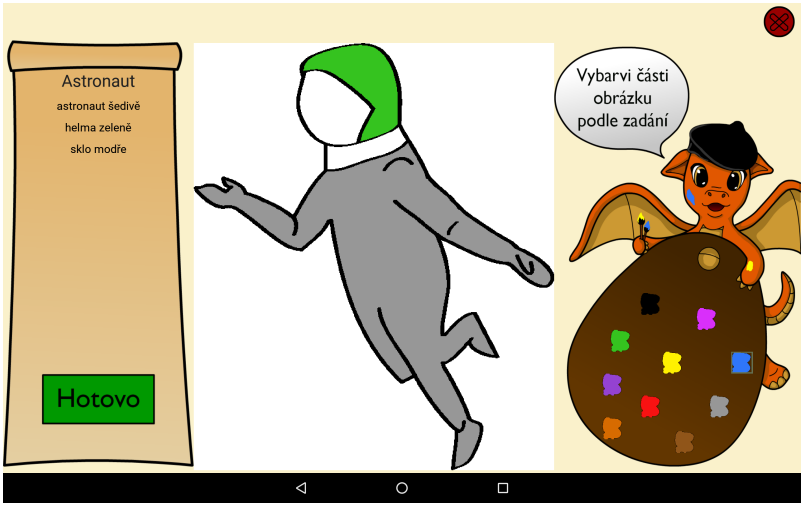
- **Nejasné rozlišení ručiček** Často se stávalo, že děti nevěděly, která z ručiček reprezentuje hodiny a která reprezentuje minuty.

2.3.5 Výsledky testování dalších modulů

Kromě modulu Hodiny byly do testování zahrnuty další dva moduly z projektu Dráček II a aplikace Jdu do školy, která je volně dostupná na portálu Google Play, pro referenci. Jediný z modulů, u kterého se nevyskytly žádné problémy, byl modul Otáčení.

Modul Vybarvování

Při testování modulu Vybarvování, který vznikl součástí projektu Dráček II, se vyskytovaly zejména problémy, které byly již zmíněné jako nedostatky modulů. Cílem cvičení tohoto modulu je vybarvit zadaný obrázek podle poskytnutých instrukcí. Vzhled testovaného modulu je na obrázku 2.7



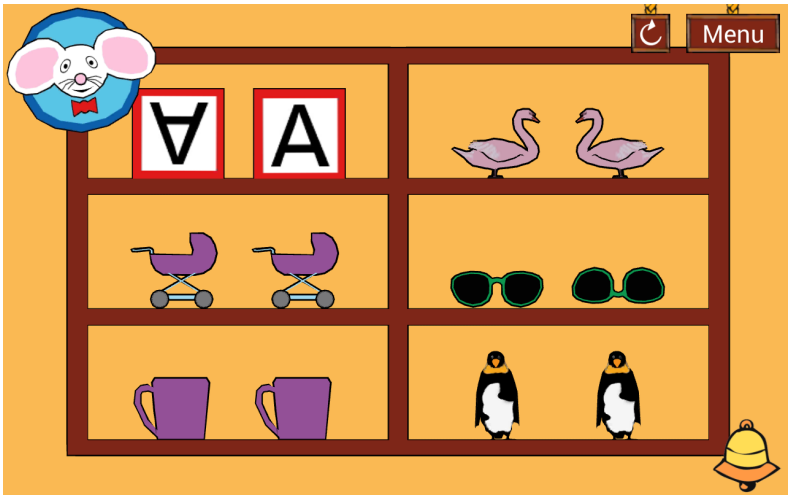
Obrázek 2.7: Vzhled testovaného modulu Vybarvování, převzato z [4]

- **Malá tolerance přijmutí odpovědi** Často se při testování modulu vyskytovala situace, kdy dítě přesunulo obrázek na koš jen částečně a modul takovou odpověď nepřijal, ani neoznačil za špatnou. Dítě se následně pokusilo vybrat obrázek jiný, často již nesprávný, což pak vedlo k dalšímu zmatení.
- **Dlouhé načítání obrázků** Vzhledem k tomu, že při přechodu na další sadu obrázků se modul na chvíli zastaví a nedává najevo, že pracuje,

došlo k situaci, kdy dítě vícekrát potvrdilo své vybarvení a tím přeskočilo další obrázek zadáním špatného řešení.

- **Moc vysoká podobnost dvou barev** Tento problém vznikl až úpravou modulu před vlastním testováním, nelze ho tedy přisuzovat tvůrci modulu. Dětem se občas pletla růžová barva s barvou fialovou.

Aplikace Jdu do školy



Obrázek 2.8: Cvičení v aplikaci Jdu do školy, převzato z [5]

Za účelem srovnání existujících modulů a jejich testování s již hotovou aplikací, která se zaměřuje na výuku dětí, jsme k dodatečnému testování zvolili aplikaci Jdu do školy. V této aplikaci je uživatel provázen mluvícím avatarem, který dává instrukce a hodnotí výsledky. Vzhled aplikace je na obrázku 2.8. Při testování této aplikace jsem si všiml několika problémů.

- **Zdlouhavý mluvený doprovod** Často předtím, než mohlo dítě začít úkol, který mu aplikace předložila, plnit, muselo vyčkat, než doprovázející avatar domluví. Čekání bylo moc dlouhé a dítě se již v průběhu snažilo úkol splnit.

- **Nemožnost opravit se při špatné odpovědi** Aplikace nedovolila dítěti opravit jeho špatnou odpověď, přestože se o to snažilo a zdlouhavě vysvětlovala, proč byla jeho odpověď špatná.

I přes tyto nedostatky byla aplikace Jdu do školy mezi testovanými dětmi nejpopulárnější, zejména díky zvukovému doprovodu a animacím, které v modulech Dráčka chybí.

Problémům, kterých jsem si při testování již existujících modulů Dráčka všiml, je potřeba se v dalších modulech vyhnout. Zejména je potřeba vyhnout se dlouhému čekání a tolerovat menší přesnost vstupu uživatele.

2.4 Současný stav výukových aplikací

Pro každé z vybraných témat jsem zvolil několik nejvhodněji vypadajících aplikací na platformu Android. Při analýze jsem zkoumal pouze řešení, která jsou dostupná zdarma v úplné nebo omezené formě na portálu Google Play. Hodnotil jsem následující parametry: podporu českého jazyka, aktivitu vývoje řešení, stoupající obtížnost úloh a přívětivost uživatelského rozhraní. Pro většinu hodnocení je maximální hodnota 5, v případě podpory českého jazyka jsou uděleny další dva body. Maximální možné celkové hodnocení je 17 bodů.

2.4.1 Algoritmizace

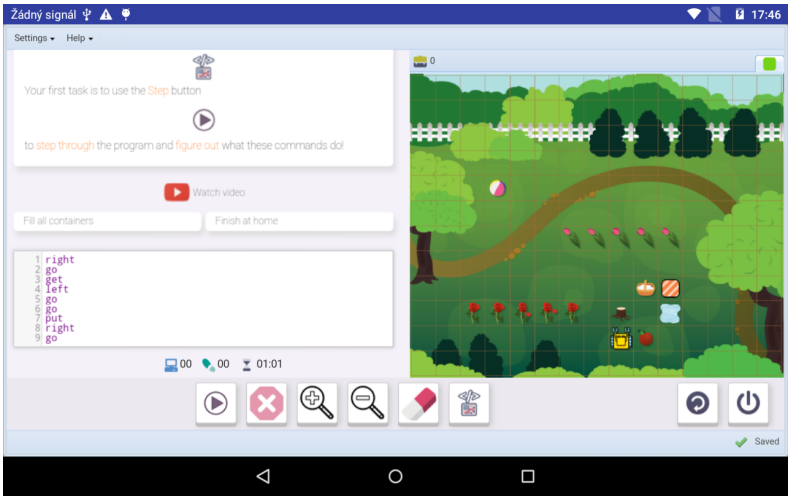
Robot Karel

Karel je robot, který se umí pohybovat po předem stanovené mřížce a vykonávat základní příkazy KROK, VLEVO-VBOK, POLOZ a ZVEDNI. Součástí programovacího jazyka jsou také další příkazy, které umožňují podmí-něné provádění, cykly a další příkazy, které neovlivňují stav samotného robota. Jazyk podporuje definici vlastních příkazů a tak tvořit složitější funkční celky. Veškeré programování je formou textu [19].

Pro zařízení na platformě Android existuje řada implementací Karla. Tyto verze však sdílí jeden zásadní problém. Pokud je pro spuštění použito zařízení bez hardwarové klávesnice a je potřeba použít dotykovou klávesnici, na menších zařízeních zabere klávesnice velkou část obrazovky. Důležité části aplikace, jako je zobrazení plochy, po které se robot pohybuje, pak nejsou zcela vidět a při psaní kódu je třeba často dotykovou klávesnici zavírat. Grafické provedení aplikace je k vidění na obrázku 2.9.

Jako reprezentanta jsem vybral implementaci Karel Coding: Code Hour [6], která má podporu pro český jazyk. Zároveň poskytuje postupně stoupající obtížnost. V době psaní této práce však poslední verze aplikace je z března roku 2016.

Parametr	Hodnota
Podpora českého jazyka	2
Aktivita vývoje	3
Stoupající obtížnost	5
Přívětivost uživatelského rozhraní	2
Celkové hodnocení	**12**



Obrázek 2.9: Snímek aplikace Karel Coding: Code Hour [6]

Scratch

Scratch je programovací jazyk zaměřený na vizuální programování. Cílovou věkovou skupinou jsou děti od 8 do 16 let. Lze v něm tvořit například hry a animace[20]. Scratch je inspirován stavebnicí Lego, zejména jednoduchostí skládání kostek a jasnou vizuální nápovědou, které kostky lze spolu skládat. Například pro kontrolní bloky, jako jsou například cykly nebo podmínky, slouží v jazyce Scratch kostky ve tvaru písmene C, aby bylo jasné, že do nich lze vložit další kostky [21].

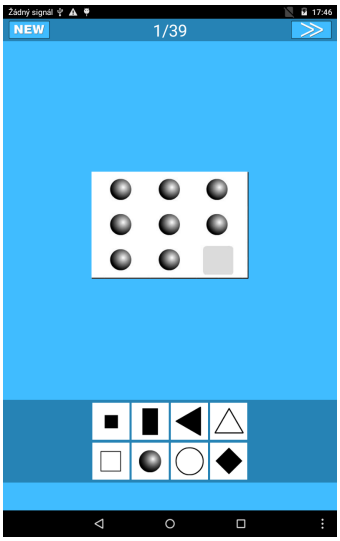
Obrázek 2.10: Snímek aplikace ScratchJr [7]

Verze prostředí jazyka pro platformu Android je dobře ovladatelná, jelikož jde pouze o přesouvání kostek a není vyžadováno žádné psaní textu. Vzhled aplikace je k vidění na obrázku 2.10. Tím se vyhýbá problému popsaném u implementací jazyka Karel. Poslední aktualizace aplikace je v době psaní této práce ze září roku 2016 [7]. V základní variantě ale nedává jazyk programátorovi specifické problémy k vyřešení. Soustředí se na kreativitu a tvorbu vlastních projektů, ideálně ve spolupráci s existující komunitou.

Parametr	Hodnota
Podpora českého jazyka	0
Aktivita vývoje	4
Stoupající obtížnost	1
Přívětivost uživatelského rozhraní	2
Celkové hodnocení	**7**

2.4.2 Matematická logika

Na téma logiky existuje řada výukových her, nicméně tyto hry se zaměřují pouze na doplňování řad, popř. podobné logické hádanky. Žádná z her, které jsem našel, se však nezabývá tématy matematické logiky, jako jsou logické spojky či predikáty. Často se také vyskytují aplikace, které uživateli měří IQ na základě otázek na logické doplnění řady obrázků, čísel či písmen. Jednou z takových aplikací je IQ Test [8], jejíž vzhled je k vidění na obrázku 2.11, ale kvůli příliš malé shodě s tématem, na které se zaměřuji, jsem takové aplikace dále neanalyzoval.



Obrázek 2.11: Snímek aplikace IQ Test [8]

Nejblíže hledanému řešení byla aplikace matematická logika [22], která však není formou hry a pouze umožňuje prohlížet teoretické informace. Pro téma matematické logiky jsem nenašel žádného vhodného kandidáta na analýzu.

2.4.3 Převody mezi soustavami

Pro procvičování převodů čísel mezi různými soustavami existuje pro platformu Android několik řešení.

Binary Challenge

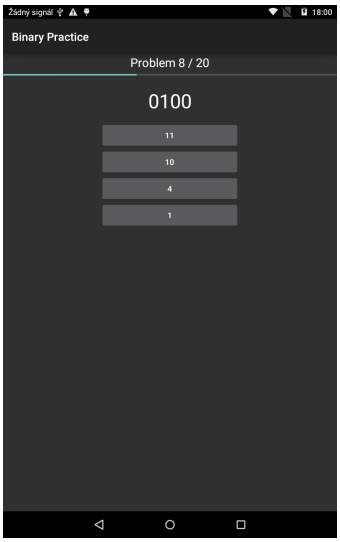
Jedním z nich je aplikace Binary Challenge, která umožňuje uživateli procvičovat použití soustav se základem 2, 4 a 8. Aplikace je uživatelsky přívětivá, nicméně neobsahuje český překlad, tudíž je potřeba použít cizojazyčnou verzi. Uživatelské rozhraní aplikace je na obrázku 2.12. Obtížnost je nastavitelná, dobrými výsledky si lze odemknout další úrovně obtížnosti a módy hry. Aplikace má v době psaní této práce poslední aktualizaci v listopadu 2016 [9].

Obrázek 2.12: Snímek aplikace Binary Challenge [9]

Parametr	Hodnota
Podpora českého jazyka	0
Aktivita vývoje	5
Stoupající obtížnost	4
Přívětivost uživatelského rozhraní	4
Celkové hodnocení	**13**

Binary Practice

Další aplikací, která se však soustředí pouze na soustavu se základem 2, je Binary Practice. V této aplikaci jde vždy o převod jednoho čísla z dvojkové do desítkové soustavy. Jsou k dispozici tři úrovně obtížnosti, které si uživatel musí sám zvolit. Řešení problému probíhá zvolením jedné z několika možností, pro uživatelské rozhraní je použitý základní systémový vzhled, viz obrázek 2.13. Aplikace je dostupná pouze v anglickém jazyce. Poslední aktualizace byla provedena v březnu roku 2013 [10].



Obrázek 2.13: Snímek aplikace Binary Practice [10]

Parametr	Hodnota
Podpora českého jazyka	0
Aktivita vývoje	1
Stoupající obtížnost	2
Přívětivost uživatelského rozhraní	3
Celkové hodnocení	**6**

2.4.4 Shrnutí hodnocení

Ani jedna z testovaných aplikací nedosáhla plného hodnocení. V kategorii programování vyšla díky podpoře českého jazyka nejlépe aplikace Robot Karel, která však body opět ztratila kvůli nepřívětivosti svého uživatelského rozhraní. Aplikace Binary Challenge byla nejlepší nalezenou aplikací v kategorii převodů mezi soustavami, nicméně neměla podporu českého jazyka. V následující tabulce je souhrn bodových zisků veškerých testovaných aplikací.

Parametr	Robot Karel	Scratch	Binary Challenge	Binary Practice
Podpora českého jazyka	2	0	0	0
Aktivita vývoje	3	4	5	1
Stoupající obtížnost	5	1	4	2
Přívětivost uživatelského rozhraní	2	2	4	3
Celkové hodnocení	**12**	**7**	**13**	**6**

2.5 Nové moduly Dráčka III

Na základě zkoumaných aplikací a metod výuky budu vytvářet následujících pět nových modulů pro aplikaci Dráček.

2.5.1 Labyrint

Modul labyrint se bude snažit přiblížit základy algoritmizace tím, že uživateli umožní z dostupných akcí sestavit jednoduchý algoritmus. Na hrací ploše bude jednoduchý labyrint. Postava, která bude umístěna uvnitř labyrintu by se měla postupným provedením akcí v algoritmu dostat z labyrintu ven. Modul bude snažit zdůraznit možnost různých řešení pro jeden problém, tedy postavu půjde z labyrintu dostat různými posloupnostmi akcí. Akce budou reprezentovány bloky, které bude možné přesunout.

2.5.2 Labyrint 2

Modul Labyrint 2 bude pouze modifikací předchozího modulu. V tomto modulu bude již k dispozici sestavená posloupnost akcí, které postava vykoná. Tyto akce však postavu nepovedou ven z labyrintu. Cílem uživatele bude odstranit nesprávné akce z posloupnosti tak, aby se postava dostala ven.

2.5.3 Skákací panák

Modul Skákací panák bude založen na stejnojmenné dětské hře a bude zaměřený na základy matematické logiky. Na hrací ploše bude nakreslený panák, po kterém bude postava skákat. Panák však nebude hotový. Podle určitých pravidel, jak by postava měla skákat, bude úkolem uživatele doplnit volná místa v panákovi tak, aby byla pravidla splněna.

2.5.4 Skákací panák 2

Modul Skákací panák 2 bude modifikací modulu Skákací panák. Cílem uživatele je, na rozdíl od předchozího modulu, odstranit políčka z panáka, která porušují daná pravidla.

2.5.5 Převody soustav

Modul převody soustav bude mít za cíl procvičování převodů mezi číselnými soustavami pomocí ikon namísto číslic. Uživatel bude mít vždy k dispozici jednoduše reprezentovaný převod soustav, podle kterého bude převádět předložené hodnoty z jedné soustavy na druhou. Hodnoty budou vždy reprezentovány ikonami a pro odpověď bude možné zvolit ikony z druhé soustavy.

2.6 Souhrn požadavků

Na základě požadavků jádra, analýzy existujících aplikací a uživatelského testování předchozích modulů byly vytvořeny následující požadavky.

Společné funkční požadavky

- **FR1** Modul bude umět načíst zadání poskytnutá jádrem.
- **FR2** Modul oznámí uživateli informaci o správnosti jeho řešení potom, co jej uživatel potvrdí.
- **FR3** Modul bude počítat procentuální hodnocení ze všech zadání a měřit čas jejich řešení.
- **FR4** Cvičení bude možné v modulu ukončit před vyřešením, rozpracované cvičení nebude uloženo.
- **FR5** Editor bude schopen načítat zadání poskytnutá jádrem a ukládat zadání do jádra.
- **FR6** Editor bude umožňovat tvorbu nových zadání.
- **FR7** Editor bude umožňovat úpravu existujících zadání.
- **FR8** Editor bude umožňovat mazání existujících zadání.

Společné nefunkční požadavky

- **N1** Moduly budou provedeny českém jazyce.
- **N2** Moduly budou implementovány pro platformu Android 5.0.
- **N3** Hry budou cílené svým vzhledem a prezentací na žáky prvního stupně základní školy.

Labyrint

- **FRL.1** Na hrací ploše budou umístěné zdi, východ a jiné elementy labyrintu.
- **FRL.2** Na hrací ploše bude na jednom místě postava.
- **FRL.3** Uživatel bude moct poskytnuté akce poskládat do algoritmu.
- **FRL.4** Uživatel bude moct akce z neběžícího algoritmu odebírat.
- **FRL.5** Uživatel bude moct hotový algoritmus spustit a sledovat postavu v pohybu.

Labyrint 2

- **FRL2.1** Na hrací ploše budou umístěné zdi, východ a jiné elementy labyrintu.
- **FRL2.2** Na hrací ploše bude na jednom místě postava.
- **FRL2.4** Uživatel bude moct akce z neběžícího algoritmu odebírat.
- **FRL2.5** Uživatel bude moct vrátit algoritmus po původního stavu.
- **FRL2.6** Uživatel bude moct hotový algoritmus spustit a sledovat postavu v pohybu.

Skákací panák

- **FRP.1** Na hrací ploše bude nakreslený nedokončený skákací panák.
- **FRP.2** Uživatel může z poskytnutých možností doplnit panáka.
- **FRP.3** Uživatel může odstranit své doplnění.
- **FRP.4** Uživatel může potvrdit své řešení.

Skákací panák 2

- **FRP2.1** Na hrací ploše bude nakreslený hotový skákací panák.
- **FRP2.2** Uživatel může na panákovi označit špatně doplněné části.
- **FRP2.3** Uživatel může zrušit své označení.
- **FRP2.4** Uživatel může potvrdit své řešení.

Převody soustav

- **FRS.1** Uživateli bude po celý průběh cvičení zobrazena nápověda na převod.
- **FRS.2** Uživatel bude moct zvolit z několika možností převodu.
- **FRS.3** Uživatel bude moct potvrdit svoji volbu.

Kapitola 3

Návrh

V této kapitole je nejprve popsána knihovna, pomocí které budou všechny moduly komunikovat s jádrem. Následuje návrh jednotlivých modulů a editorů k nim příslušících.

3.1 Návrh společné knihovny

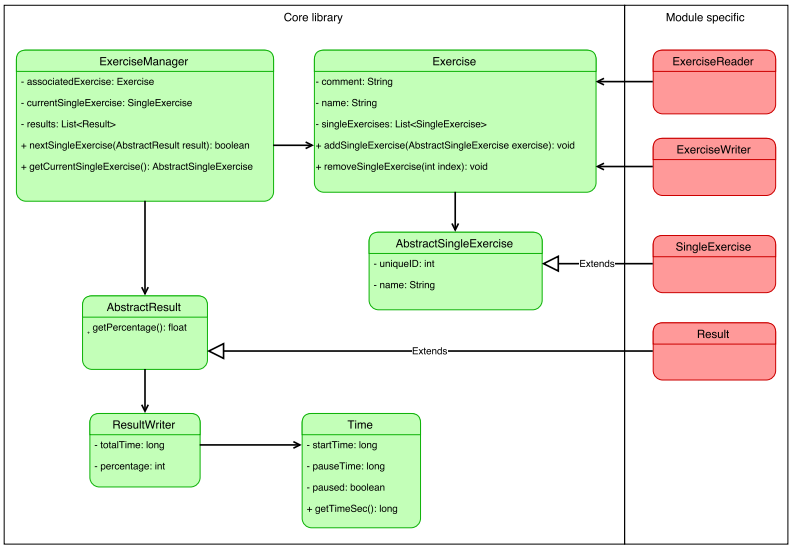
Vzhledem k tomu, že část funkcí budou mít všechny moduly společnou, rozhodli jsme se tyto funkce oddělit do samostatné knihovny. Jedná se hlavně o komunikaci s jádrem, načítání a ukládání cvičení a obecně rozhraní, která musí všechny moduly splňovat. Knihovnu jsme se rozhodli rozdělit do dvou vrstev: aplikační vrstva a prezentační vrstva.

3.1.1 Aplikační vrstva

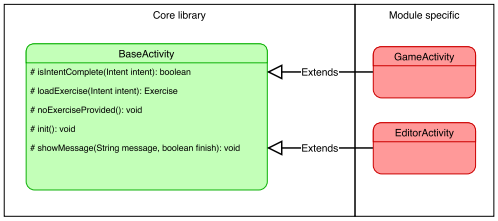
Součástí této vrstvy je třída Exercise, kterou používají všechna cvičení ke správě celkových zadání. Jednotlivé úkoly v zadáních dědí od třídy AbstractSingleExercise. Každé cvičení musí implementovat vlastní třídu ExerciseReader a ExerciseWriter, které zařizují načítání, resp. ukládání zadání. Ve všech modulech musí být také implementace abstraktní třídy AbstractResult, která poskytne procentuální úspěšnost řešení za účelem hodnocení. Schema tříd je na diagramu 3.1.

3.1.2 Prezentační vrstva

Tato vrstva obsahuje pouze třídu, od které dědí všechny moduly a jejich editory. Je zde provedena základní kontrola zadání a následně je spuštěno samotné cvičení či editor, jak je vidět na diagramu 3.2.



Obrázek 3.1: Diagram aplikační vrstvy, převzato z [4]

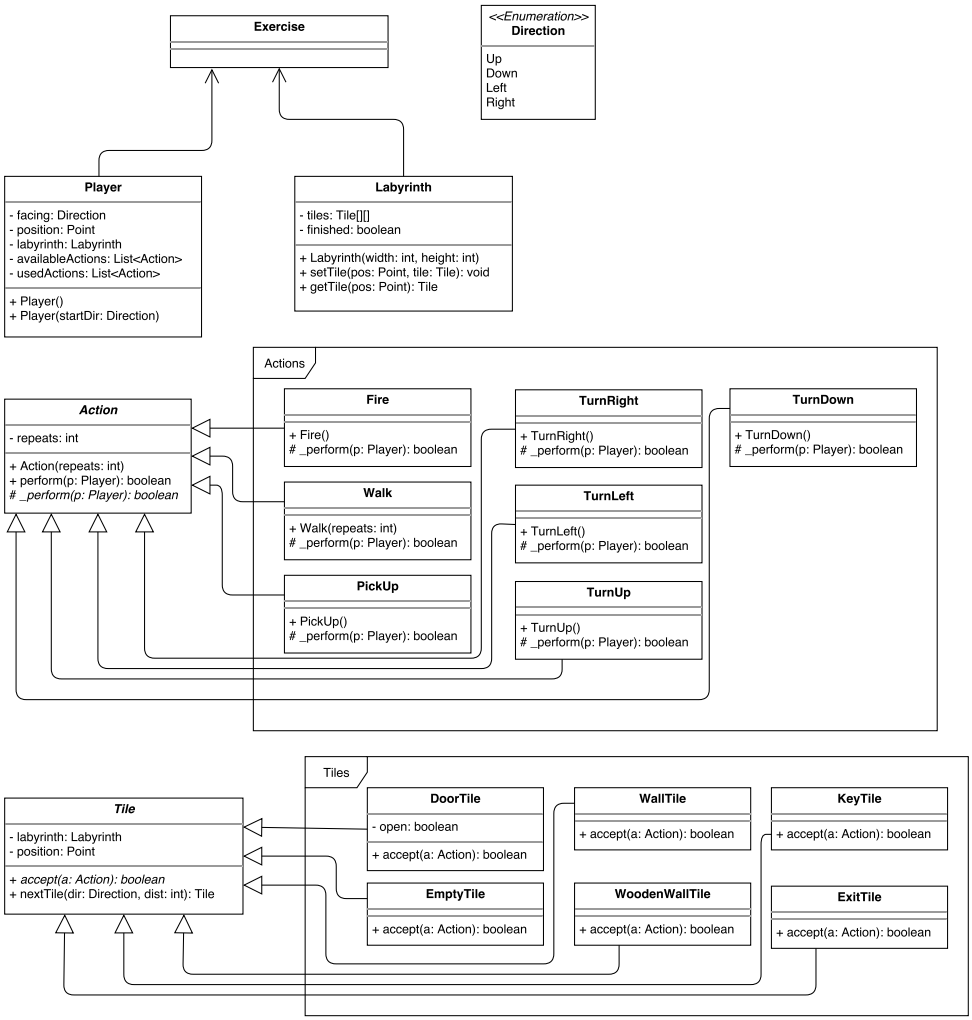


Obrázek 3.2: Diagram prezentační vrstvy, převzato z [4]

3.2 Návrh modulu Labyrint

Pro tento modul je nejprve potřeba navrhnout, o jaký labyrint se bude jednat. Jednou z možností je mřížka skládající se z čtvercových polí, kde každé pole reprezentuje jedno místo, na které se může postava přesunout. Každé pole je pak možné samostatně nastavit. Tento způsob je jednoduchý na implementaci, nicméně neumožňuje velkou úroveň volnosti ve tvorbě labyrintů.

Druhou možností je umožnit volné umístění částí labyrintu do hracího prostoru. Tento přístup oproti má oproti mřížce výhodu podstatně více možností pro tvorbu labyrintu. Na druhou stanu by podstatně zvýšil složitost řešení zadaného labyrintu, jelikož by algoritmus sestavený uživatelem měl podstatně menší toleranci drobných nepřesností. Zároveň by byla tvorba jednotlivých příkazů pro použití ve tvorbě algoritmu složitější, protože by bylo potřeba specifikovat například délku kroku postavy. Vzhledem k těmto komplikacím jsem se rozhodl labyrint reprezentovat mřížkou polí. Jednotlivá pole bude možné nastavit na následující možnosti:



Obrázek 3.3: Diagram tříd modulu Labyrint

- Země, volné místo, kam postava může stoupnout,
- Kamenná zeď,
- Dřevěná zeď,
- Klíč na zemi,
- Dveře se zámkem,
- Východ z labyrintu.

Dále je třeba rozhodnout, jaké akce bude postava schopná vykonat. Při tvorbě zadání pro cvičení bude možné tyto akce částečně upravovat a tím

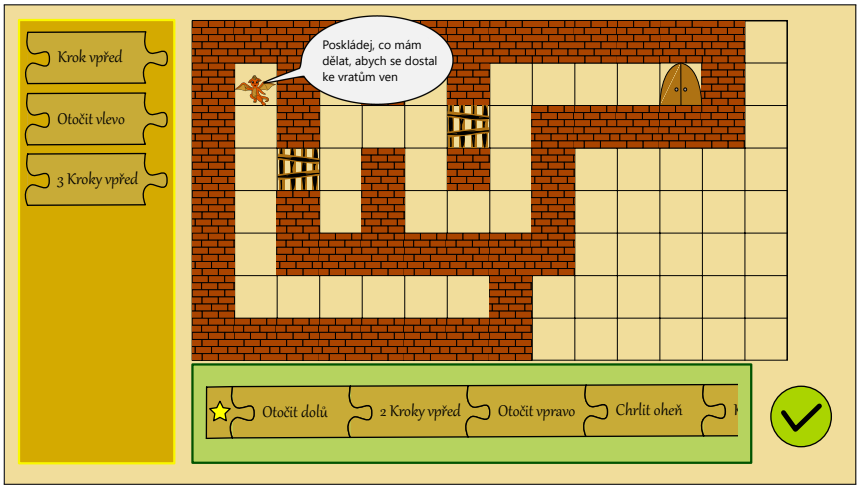
učinit zadání složitějším. Vzhledem k tomu, že postavou bude avatar Dráčka, lze zařadit také tématicky vhodné akce. Postava bude schopna:

- Učinit daný počet kroků vpřed,
- Otočit se vlevo,
- Otočit se vpravo,
- Otočit se nahoru,
- Otočit se dolů,
- Chrlit oheň, čímž může zničit dřevěné zdi,
- Sebrat klíč ze země,
- Otevřít dveře pomocí sebraného klíče.

Všechny zmíněné části jsem rozdělil do tříd znázorněných na diagramu 3.3, na kterém je vidět složení labyrintu a všechny součásti zadání.

Poslední otázkou je výpočet procentuálního skóre. Rozhodl jsem se skóre počítat jako poměr jedné k počtu, kolikrát byl sestavený algoritmus spuštěn.

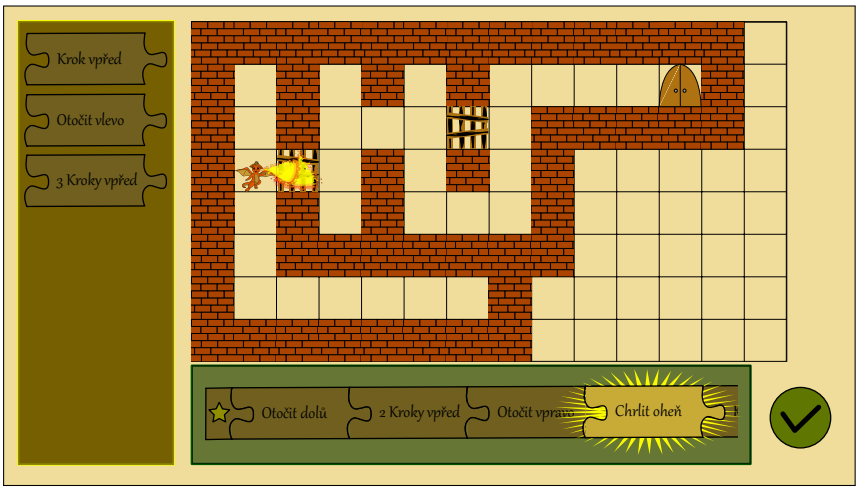
3.2.1 Vzhled modulu



Obrázek 3.4: Wireframe cvičení Labyrint

Hlavní část prostoru modulu bude zaujímat samotný labyrint, kde bude zobrazen jeho počáteční stav. Na levé straně bude lišta s akcemi, které jsou pro právě řešený labyrint dostupné. Tyto akce bude možné tahem přesunout do lišty pod zobrazením labyrintu, kde budou zobrazeny akce, které by měla

postava vykonat. Akce ve spodní liště bude možné tahem přerovnat. Jejich vykonávání bude probíhat ve stejném pořadí, v jakém jsou akce v liště srovnány. Vzhled cvičení je na obrázku 3.4.



Obrázek 3.5: Wireframe cvičení Labyrint v průběhu algoritmu

Potom, co se uživatel rozhodne, že je spokojen se svým sestaveným algoritmem, spustí algoritmus kliknutím na potvrzující tlačítko v pravé dolní části modulu. Následně dojde k přehrání jednotlivých akcí. Průběžný stav labyrintu bude možné v modulu sledovat. Právě prováděná akce bude ve spodní liště zvýrazněna, což by mělo pomoci při hledání případných chyb v algoritmu. Při provádění algoritmu nebude možné jej upravovat ani do něj přidávat další akce. Navržený vzhled probíhajícího cvičení je na obrázku 3.4.

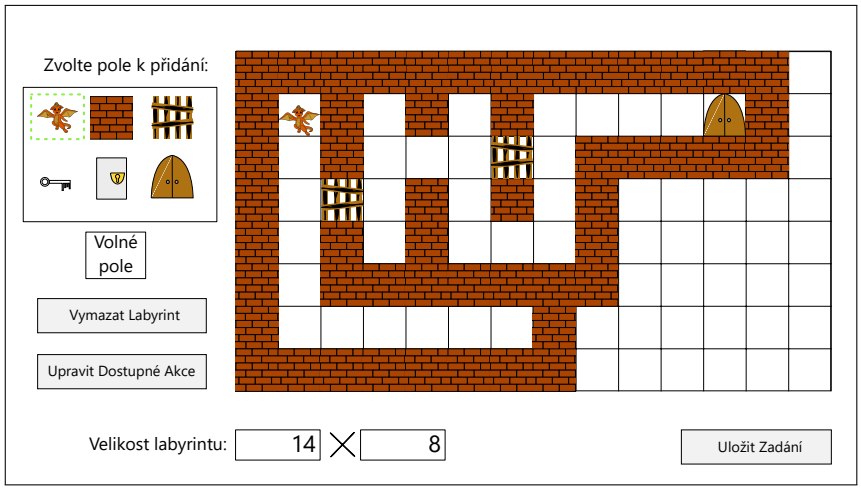
Jakmile postava dojde do východu z labyrintu, bude cvičení ukončeno vítězným oznámením úspěšného splnění.

3.2.2 Editor modulu

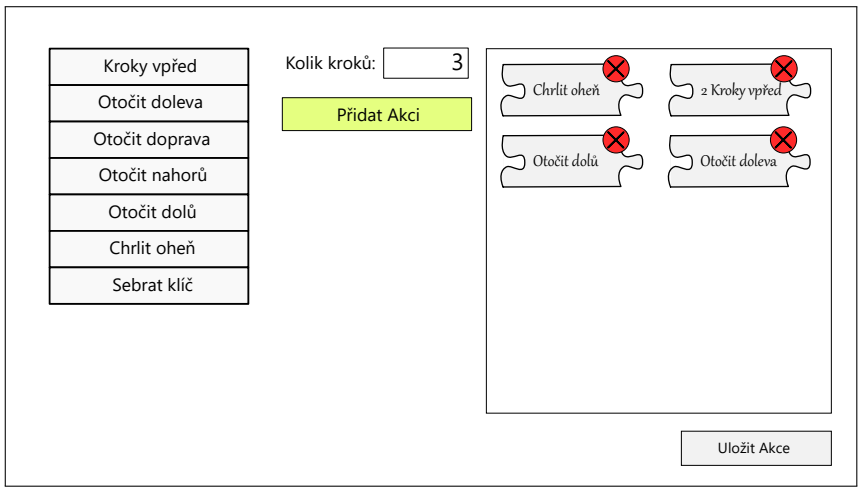
Pro modul je potřeba vytvořit také editor zadání, který umožní učiteli tvořit nové labyrinty a upravovat již existující. Editor se bude skládat ze dvou částí: editoru labyrintu a editoru dostupných akcí pro editovaný labyrint.

Tvorba samotného labyrintu bude probíhat vyplňováním volných polí jednotlivými dotyky. Při dotyku bude na volné, popřípadě již jinou možností obsazené pole umístěná právě zvolená možnost, kterou si bude uživatel moct vybrat v postranní liště. Modul bude obsahovat kontrolu průchodnosti labyrintu pouze nezávisle na dostupných akcích, protože akce budou voleny zvlášť. Návrh rozhraní pro tvorbu labyrintu je na obrázku 3.6.

Akce, které budou při řešení cvičení dostupné, bude možné přidávat jednotlivě kliknutím v levé liště do seznamu dostupných akcí v pravé liště. Pokud přidanou akcí bude učinění kroku vpřed, bude možné upravit počet kroků,



Obrázek 3.6: Wireframe editoru labyrintu



Obrázek 3.7: Wireframe editoru příkazů k použití na vyřešení labyrintu

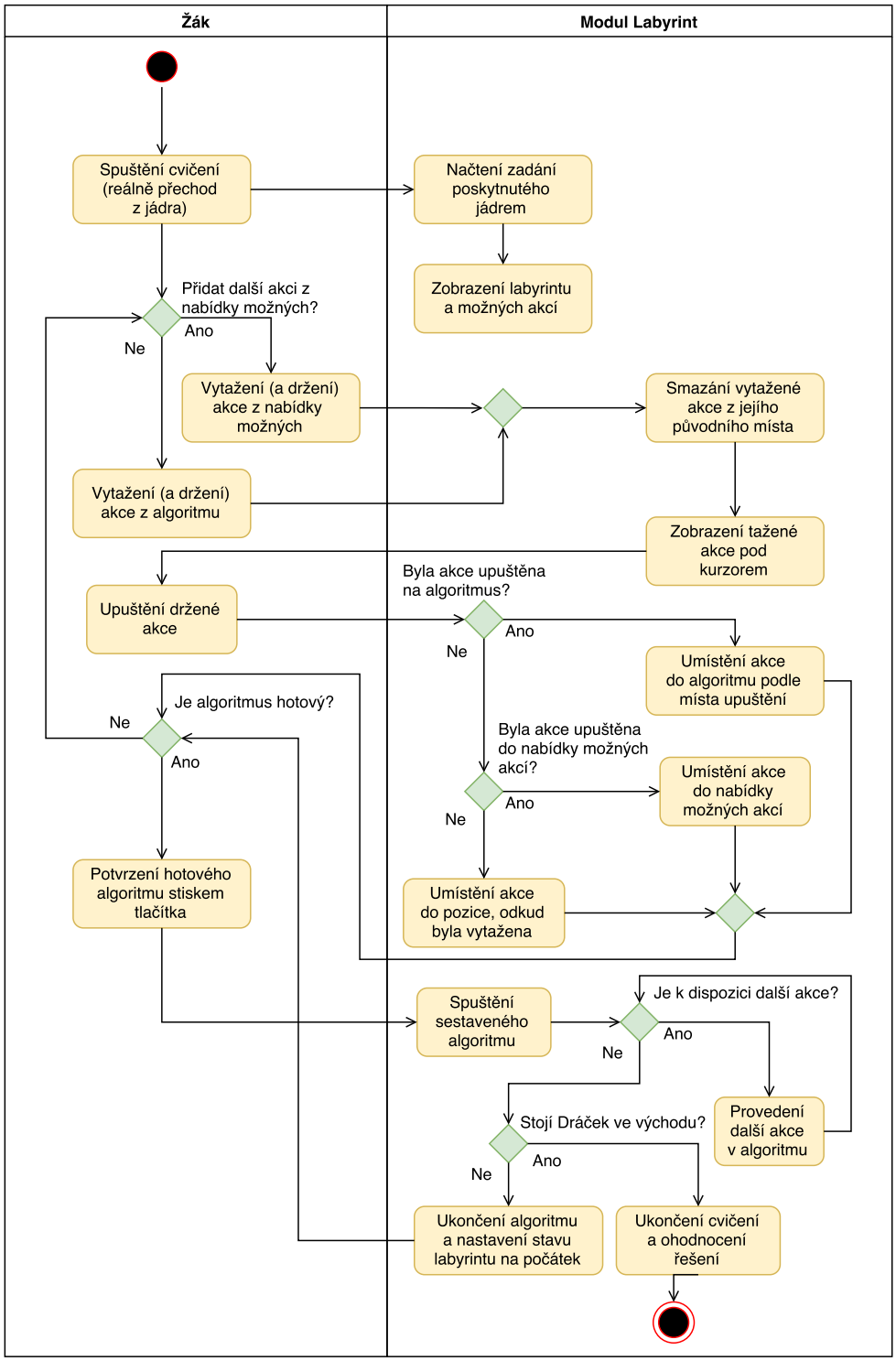
které postava při vykonání akce učiní. Vzhledem k tomu, že cvičení může umožňovat více rozdílných řešení, nebude učitel schopen v editoru nastavit správné řešení. Na obrázku 3.7 je vidět návrh vzhledu části editoru pro úpravu akcí, které má řešitel cvičení k dispozici.

3.2.3 Chování modulu

- **Kliknutí na Dráčka** → zobrazení dočasné bubliny s nápovědou nad dráčkem
- **Kliknutí na tlačítko spuštění algoritmu** → spuštění sestaveného algoritmu pro Dráčka
- **Vytažení akce z dostupných akcí** → smazání vytažené akce z dostupných akcí, zobrazení vytažené akce pod kurzorem
- **Upuštění akce algoritmu** → přidání nesené akce do algoritmu na místo kurzoru, smazání nesené akce ze zobrazení pod kurzorem
- **Vytažení akce z algoritmu** → smazání vytažené akce z algoritmu, zobrazení vytažené akce pod kurzorem
- **Upuštění akce do dostupných akcí** → přidání nesené akce do dostupných akcí, smazání nesené akce ze zobrazení pod kurzorem
- **Kliknutí na jinam** → žádná reakce

3.2.4 Průběh modulu

Po přechodu do modulu z jádra je načten labyrint ze zadání poskytnutého jádrem. Následně je uživateli umožněno sestavit algoritmus, který by zadání měl vyřešit. Uživatel se sám rozhoduje, kdy algoritmus spustí a nechá tak modul vyhodnotit jeho správnost. Celý průběh je znázorněn na obrázku 3.8.



Obrázek 3.8: Diagram aktivity modulu Labyrint

3.3 Návrh modulu Labyrint 2

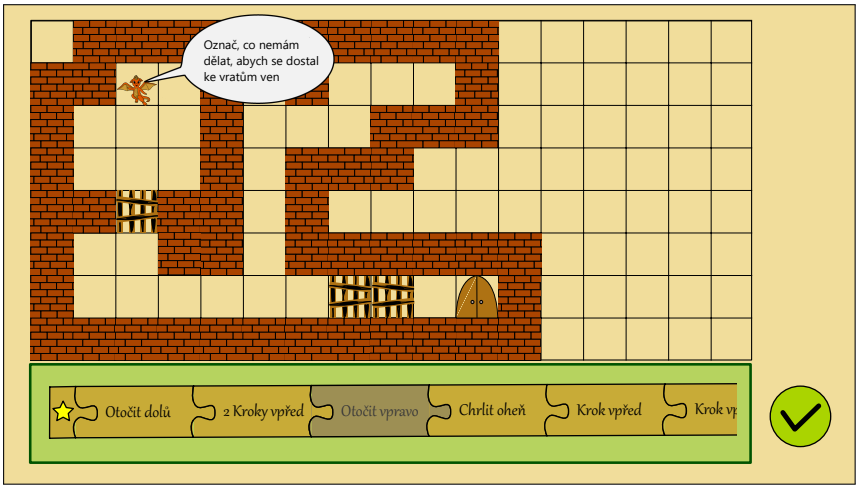
Tento modul vychází ve velké části z modulu Labyrint. Využívá stejných tříd. Díky tomu je již část otázek, které se týkají návrhu modulu, zodpovězena. Modul Labyrint 2 se bude lišit hlavně úkolem ve cvičení. Místo skládání algoritmu bude uživateli předložen již hotový algoritmus, ve kterém se však budou vyskytovat chyby. Uživatel musí odstranit chybné akce a tím algoritmus opravit.

Akce, které se v algoritmu můžou vyskytovat, se od modulu Labyrint neliší. Stejná jsou také možná pole, která se mohou v labyrintu vykytovat. Při tvorbě zadání lze tím pádem snadno vycházet ze zadání pro modul Labyrint, pouze s tím rozdílem, že je třeba v zadání již připravit celý algoritmus.

3.3.1 Vzhled modulu

Na rozdíl od modulu Labyrint se v modulu Labyrint 2 nevyskytuje lišta na levé straně obrazovky, jelikož složený algoritmus je již součástí zadání. Hlavní část obrazovky opět zabírá samotný labyrint a ve spodní části se vyskytuje lišta s algoritmem.

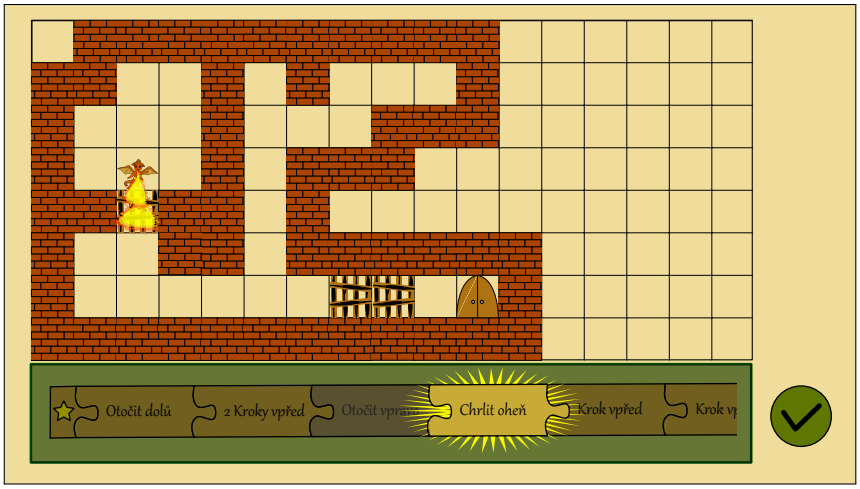
Pro odstraňování jednotlivých akcí jsem vybíral mezi dvěma způsoby. Jedním způsobem bylo akci úplně odstranit. Tím by akce zmizela z algoritmu a bylo by jasné, že nebude provedena. Tento způsob však nedovoluje uživateli rozmyslet si svoji volbu a později se opravit. Bylo by tedy zapotřebí buď cvičení s každým neúspěchem restartovat, nebo umožnit uživateli restartovat cvičení ručně.



Obrázek 3.9: Wireframe cvičení Labyrint 2 na začátku cvičení

Druhým možným způsobem odstraňování akcí je pouze na akci vizuálně naznačit, že nebude provedena, ale ponechat ji na svém místě, jak je zná-

zorněno na wireframu 3.9. Tento přístup umožní uživateli svoji volbu měnit opětovným stiskem akce a není proto potřeba umožnit restart cvičení. Zvolené akce budou při následném provádění algoritmu přeskočeny, jak je vidět na obrázku 3.10. Za účelem jednoduššího ovládání jsem zvolil tuto možnost.



Obrázek 3.10: Wireframe cvičení Labyrint 2 v průběhu algoritmu

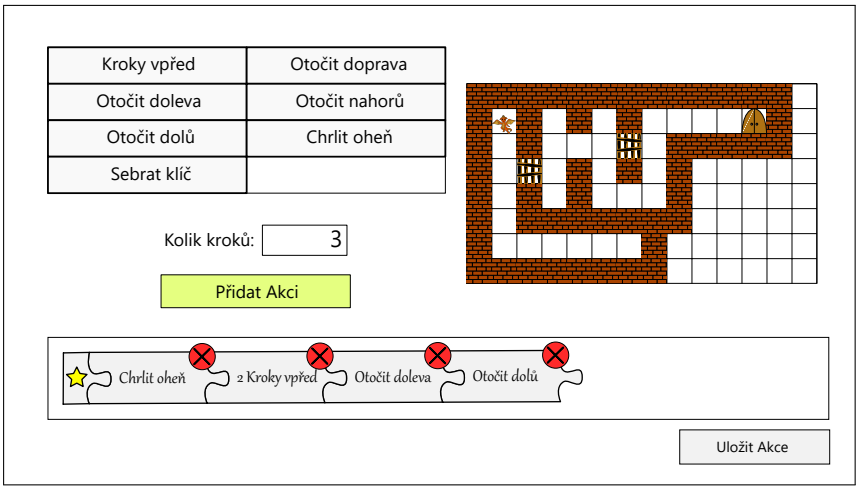
3.3.2 Editor modulu

Část editoru pro tvorbu labyrintu se od modulu Labyrint nebude lišit. Rozdíl bude až při editaci akcí pro zadání, jelikož kromě výběru akcí bude také zapotřebí složit algoritmus, který by měl být ve cvičení opraven. Nelze tedy použít editor akcí z modulu Labyrint. Kromě možnosti akce skládat je také vhodné při editaci akcí poskytnout uživateli náhled labyrintu, pro který akce edituje.

V editoru akcí bude uživatel moct přidat novou akci na konec algoritmu. Po přidání bude možné akci přesunout tahem na jinou pozici a tím algoritmus upravit. Na obrázku 3.11 lze vidět návrh upraveného editoru akcí.

3.3.3 Chování modulu

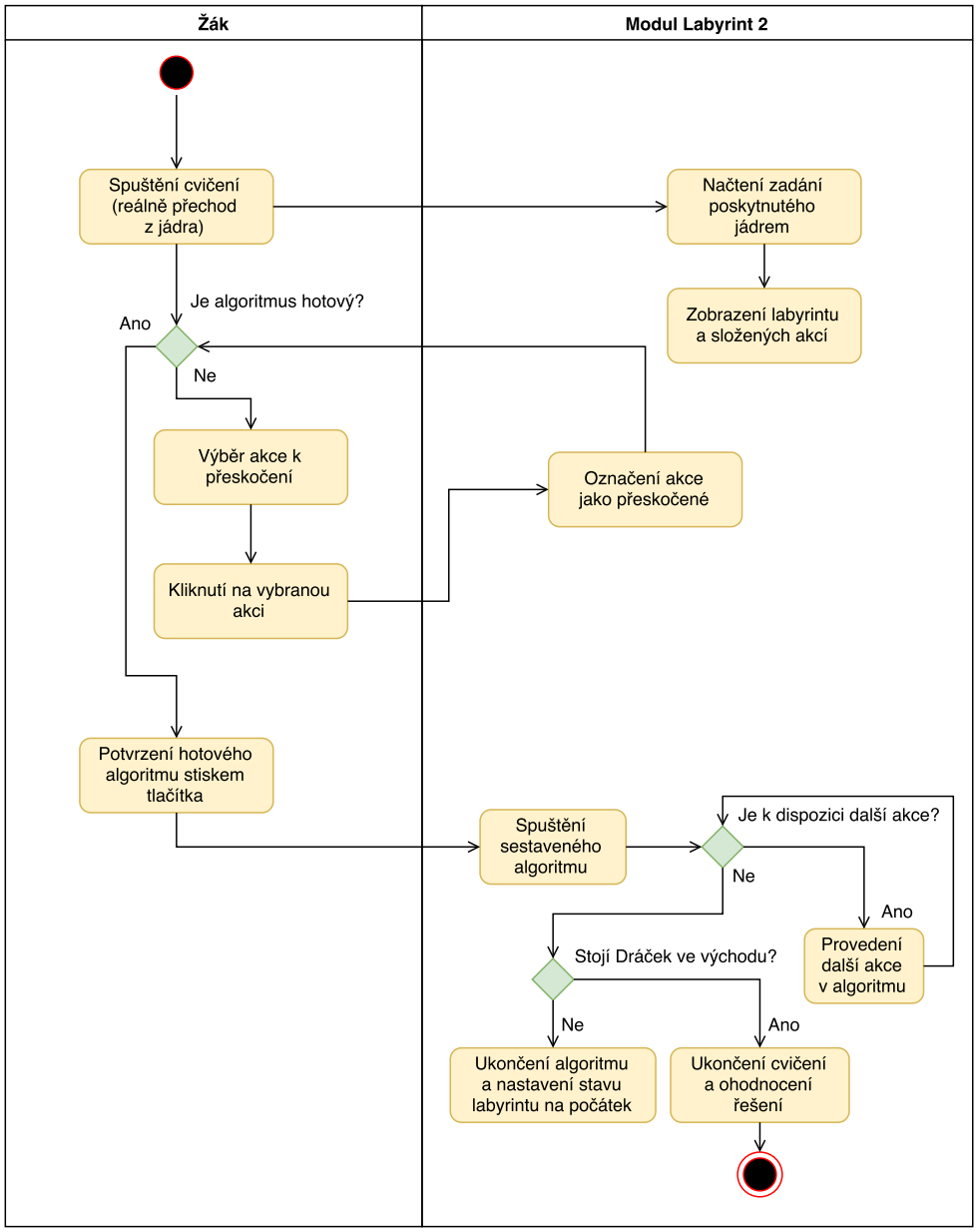
- **Kliknutí na Dráčka** → zobrazení dočasné bubliny s nápovědou nad dráčkem
- **Kliknutí na tlačítko spuštění algoritmu** → spuštění opraveného algoritmu pro Dráčka
- **Kliknutí na akci v algoritmu** → zešednutí akce, znázornění, že bude akce při běhu algoritmu přeskočena
- **Kliknutí na jinam** → žádná reakce



Obrázek 3.11: Wireframe editoru příkazů a chybného algoritmu pro cvičení

3.3.4 Průběh modulu

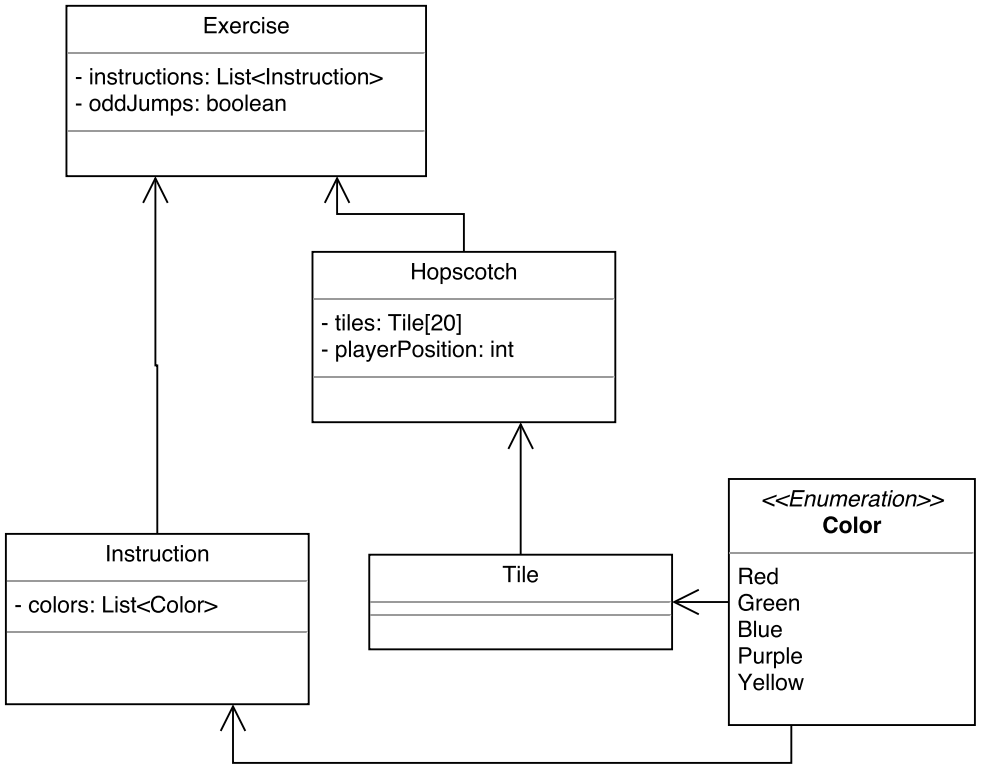
Průběh tohoto modulu je velice podobný průběhu modulu Labyrint. Důležitou změnou je, že uživatel algoritmus nesestavuje, pouze označuje akce, které se nemají vykonat. Celý průběh je znázorněn na obrázku 3.12.



Obrázek 3.12: Diagram aktivity modulu Labyrint 2

3.4 Návrh modulu Skákací panák

Cílem hráče v tomto modulu bude vybarvit skákacího panáka tak, aby Dráček, který bude přes panáka skákat po nastaveném počtu políček, skákal pouze na barvy nebo skupiny barev, které jsou v zadání. Je tedy potřeba rozhodnout, přes jakého panáka bude skákat. Je také potřeba zvolit barvy, které bude mít



Obrázek 3.13: Diagram tříd modulu Skákací panák

uživatel k dispozici pro vybarvení panáka a které se mohou objevit v zadání. Třídy modulu jsou na diagramu 3.13

Pro vzhled panáka se naskýtají dvě možná řešení. První možností je uživatele, který bude vytvářet zadání, nechat vybrat obrázek s podobou panáka. Vybraného panáka by bylo potřeba dále zpracovat a rozdělit na jednotlivá pole. Komplikací zde je rozpoznat zamýšlené pořadí skákání, které by muselo být například označeno barvou nebo jiným lehce rozpoznatelným způsobem, což by znamenalo složitější instrukce pro tvorbu zadání. U panáka může být také předpokládaná možnost skočit na dvě políčka najednou, což by komplikovalo nejen tvorbu zadání, ale také případné animace Dráčka.

Druhou možností je použít vždy stejného, předem určeného panáka. Tento přístup zjednodušuje tvorbu zadání, je totiž potřeba zadat již jen po kolika polích bude Dráček skákat a na jaká pole může skákat. Je také předem jasné pořadí a lze snadno předejít problémům s různými skoky, což umožňuje snazší případnou animaci. Nevýhodou tohoto přístupu je menší množina možných zadání. Pro jednoduchost implementace jsem zvolil možnost vždy stejného panáka. Nevýhodu zvoleného řešení lze částečně odstranit dostatečným počtem polí a možných pravidel.

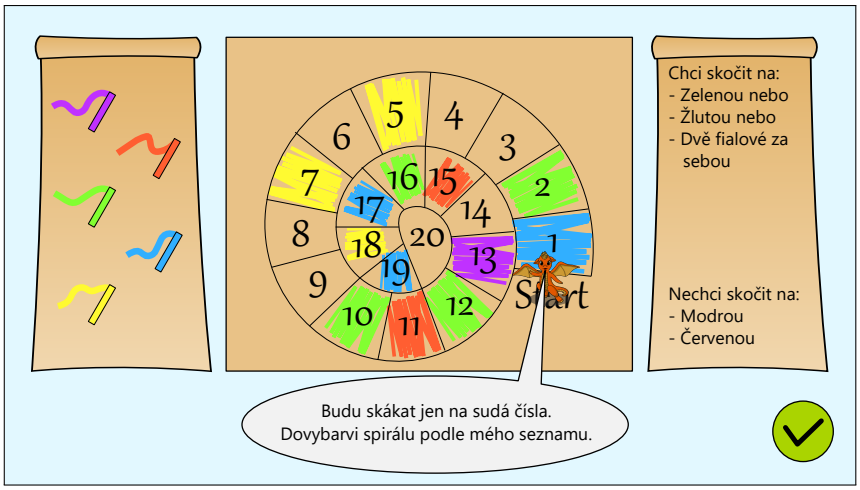
K vybarvování panáka jsem zvolil pět odlišných barev: červenou, zelenou, modrou, fialovou a žlutou. Při případné grafické úpravě vzhledu cvičení není problém barvy změnit, nicméně měly by být od sebe dostatečně odlišné, neměly by se například spolu vyskytovat možnosti fialová a růžová.

3.4.1 Vzhled modulu

Uživatelské rozhraní cvičení, které je vidět na obrázku 3.14, jsem rozdělil do tří hlavních částí:

- **Levá lišta s barvami** Zde jsou zobrazeny barvy, které lze dotykem vybrat. Zvolená barva bude použita při vybarvování panáka.
- **Prostřední část s panákem** Prostřední část obsahuje na začátku cvičení nevybarveného panáka a Dráčka, který poskytuje nápovědu a říká, jak velké skoky bude konat.
- **Pravá lišta se zadáním** V pravá části cvičení je k dispozici zadání, které má být vybarvením panáka splněno. Pod zadáním je umístěno tlačítko, kterým lze potvrdit vyplnění.

Cvičení v průběhu je k vidění na obrázku 3.15.

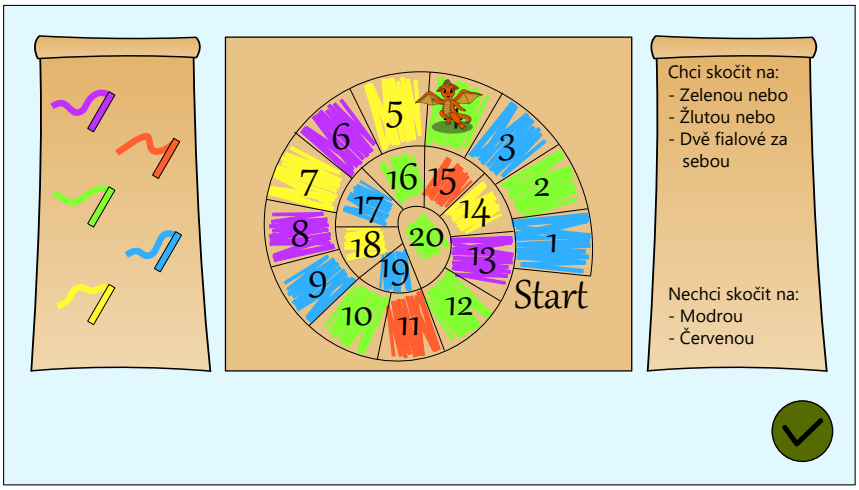


Obrázek 3.14: Wireframe cvičení Skákací panák na začátku cvičení

Pro vzhled panáka jsem zvolil spirálu, jelikož je velikostně vhodná do prostřední části a umožňuje umístit dobrý počet polí k vybarvení.

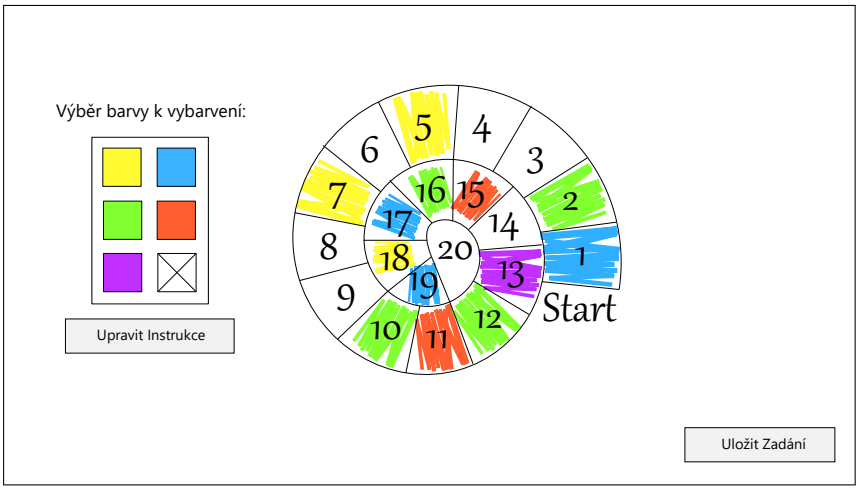
3.4.2 Editor modulu

Editor zadání pro modul Skákací panák je rozdělen na dvě hlavní části. V první části je možné editovat počáteční barevné vyplnění spirály, které v průběhu



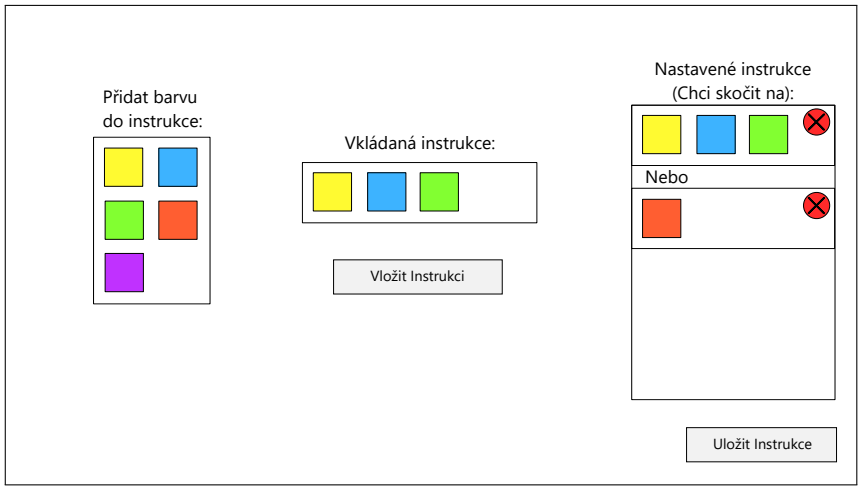
Obrázek 3.15: Wireframe cvičení Skákací panák v průběhu hodnocení

cvičení nebude možné změnit. V levém panelu, jak lze vidět na obrázku 3.16, jsou k dispozici všechny barvy a možnost odbarvení pole. Zároveň je pod panelem tlačítko pro přechod k editaci pravidel, které je při plnění zadání potřeba splnit.



Obrázek 3.16: Wireframe editoru panáka pro cvičení Skákací panák

V Editoru pravidel je možné volením příslušných barev v levém panelu tvořit pravidla, která určují, že by Dráček měl skočit na zvolené barvy v jejich nastaveném pořadí. Vytvořené pravidlo lze pak přidat do seznamu pravidel ve cvičení tlačítkem pod prostředním panelem. Vytvořená pravidla jdou odstranit příslušným křížkem v seznamu pravidel. Celé rozhraní je na obrázku 3.17.



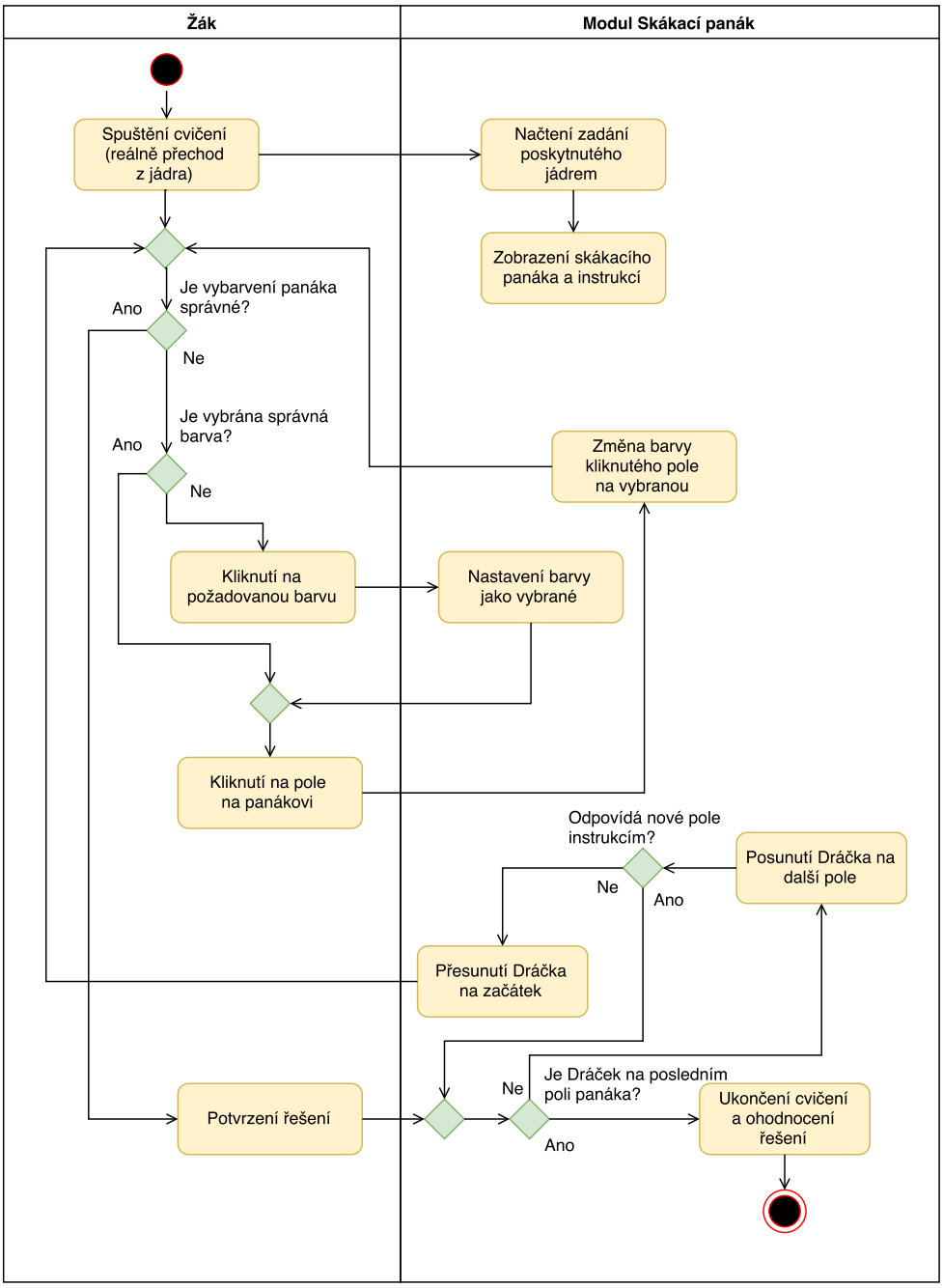
Obrázek 3.17: Wireframe editoru pravidel pro cvičení Skákací panák

3.4.3 Chování modulu

- **Kliknutí na Dráčka** → zobrazení dočasné bubliny s nápovědou nad dráčkem
- **Kliknutí jednu z barev v levé liště** → nastavení barvy, pomocí které bude dále možné vybarvovat panáka
- **Kliknutí na políčko na panákovi** → vybarvení políčka na panákovi právě zvolenou barvou
- **Kliknutí na potvrzující tlačítko** → Dráček začne skákat po panákovi a kontrolovat vyplnění
- **Kliknutí na jinam** → žádná reakce

3.4.4 Průběh modulu

Po přechodu do modulu z jádra je načteno nastavení spirály a instrukce pro uživatele. Následně může uživatel vyplnit spirálu a potvrdit své vyplnění. Modul poté vyhodnotí správnost vyplnění. Celý průběh je znázorněn na obrázku 3.18.



Obrázek 3.18: Diagram aktivity modulu Skákací panák

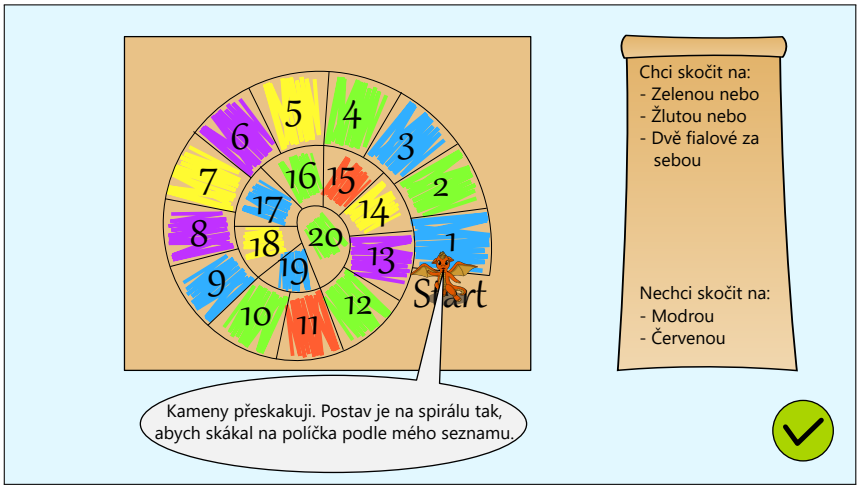
3.5 Návrh modulu Skákací panák 2

Modul Skákací panák 2 je pouze malou úpravou modulu Skákací panák a využívá stejných tříd. Zatímco cíl se mezi moduly neliší a uživatel stále musí zajistit, aby Dráček skákal pouze na správné barvy, je výrazný rozdíl v tom, jak musí k cíli uživatel dospět. Místo dovybarvování panáka je cílem uživatele umístit kameny, které bude Dráček při skákání přes panáka přeskakovat, tak, aby jejich umístěním dosáhl splnění zadaných pravidel.

Panák může být ze zadání vybarven stejnými barvami, jako v modulu Skákací panák. Na rozdíl od předchozího modulu je však vždy panák vybarven zcela.

3.5.1 Vzhled modulu

V uživatelském rozhraní modulu Skákací panák 2 oproti předchozímu modulu již není potřeba pravá lišta s barvami, jelikož panák bude v zadání vždy vybarven. Zbytek prvků je posunut doleva, čímž je částečně zaplněno vzniklé místo. V počátečním stavu jinak modul vypadá stejně jako modul Skákací panák, což je vidět na obrázku 3.19.

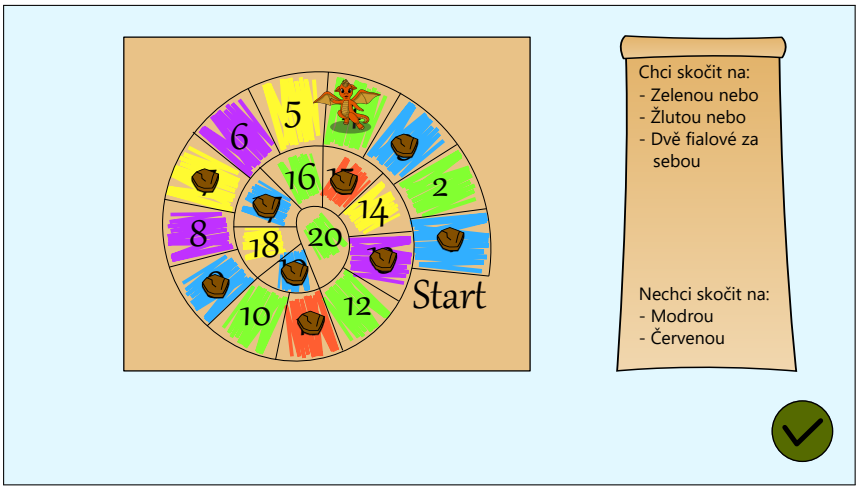


Obrázek 3.19: Wireframe cvičení Skákací panák 2 na začátku cvičení

Vyplňování cvičení probíhá formou umísťování kamenů kliknutím na příslušné pole. Opětovným kliknutím je kámen opět odebrán. Vyplněné cvičení v průběhu hodnocení je na obrázku 3.20.

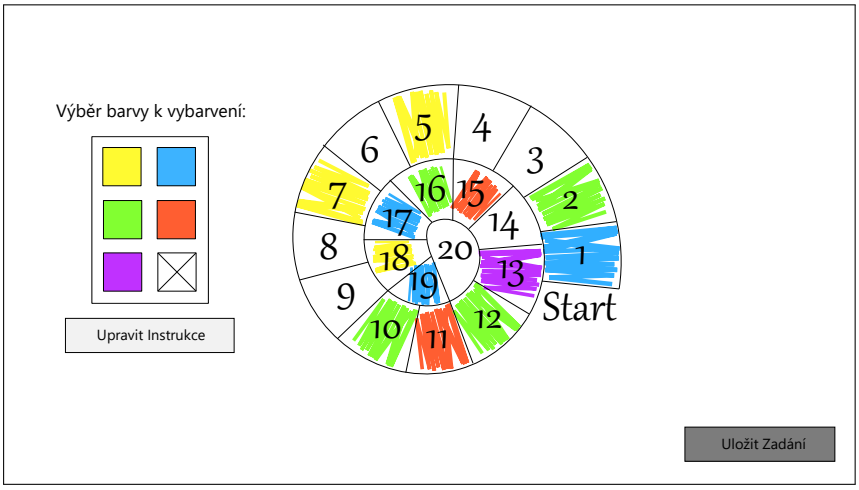
3.5.2 Editor modulu

Editor tohoto modulu se svoji funkčností téměř nebude lišit od editoru modulu Skákací panák. Jedinou změnou bude požadavek na kompletní vybarvení



Obrázek 3.20: Wireframe cvičení Skákací panák 2 v průběhu hodnocení

panáka. Pokud panák nebude zcela vybarven, nebude možné aktuální zadání uložit. Editor v tomto stavu je na obrázku 3.21.



Obrázek 3.21: Wireframe editoru panáka pro modul Skákací panák 2 s nekompletním vybarvením

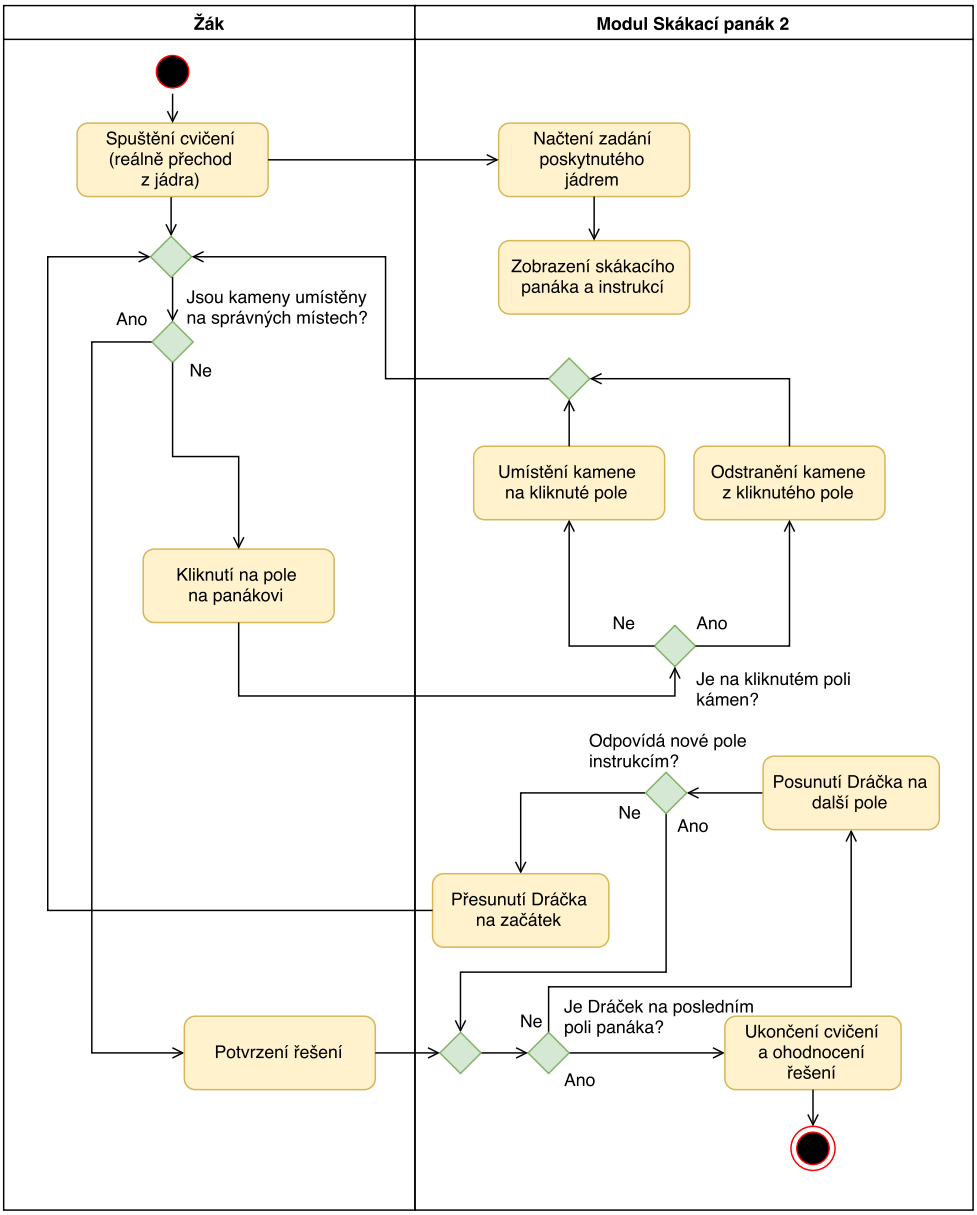
3.5.3 Chování modulu

- **Kliknutí na Dráčka** → zobrazení dočasné bubliny s nápovědou nad dráčkem
- **Kliknutí na volné pole na panákovi** → umístění kamene, který Dráček přeskočí, na příslušné pole

- **Kliknutí na pole na panákovi s kamenem** → odebrání kamene z příslušného pole
- **Kliknutí na potvrzující tlačítko** → Dráček začne skákat po panákovi a kontrolovat vyplnění
- **Kliknutí na jinam** → žádná reakce

3.5.4 Průběh modulu

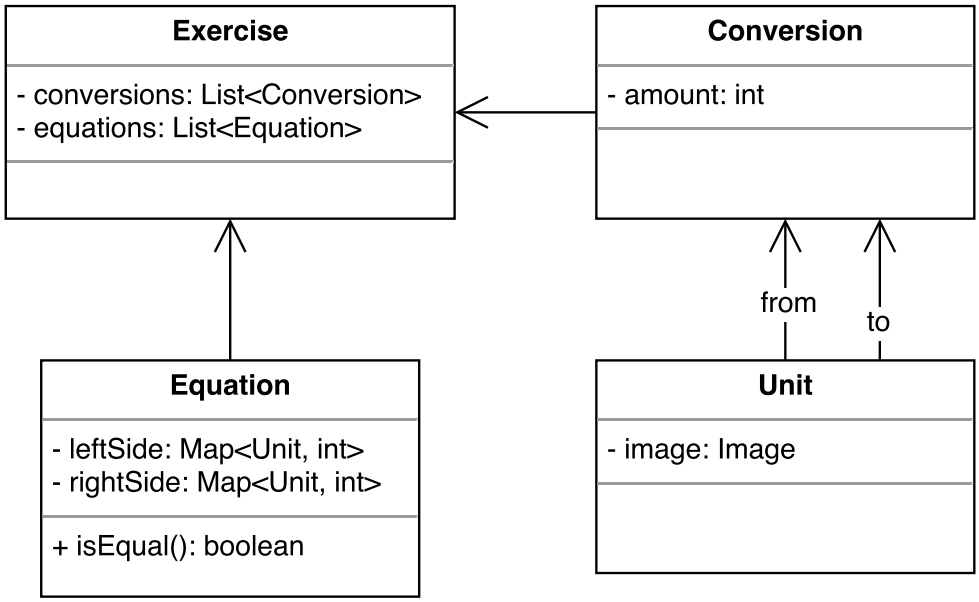
Modul probíhá podobně, jako modul Skákací panák s tím rozdílem, že uživatel nemůže spirálu vybarvovat, ale pokládat na ní kameny, které Dráček přeskočí. Celý průběh je znázorněn na obrázku 3.22.



Obrázek 3.22: Diagram aktivity modulu Skákací panák 2

3.6 Návrh modulu Převody soustav

V modulu Převody soustav je cílem uživatele doplnit zadané rovnice tak, aby platily. Modul je založen na cvičení Děda Lesoň podle Hejného metody výuky matematiky [17]. Jednotlivá čísla v rovnicích jsou reprezentována obrázky a operace mezi nimi je vždy sčítání. Pro tento modul bylo potřeba určit, jaké obrázky budou použity pro jednotlivá čísla. Konečné hodnoty obrázků však



Obrázek 3.23: Diagram tříd modulu Převody soustav

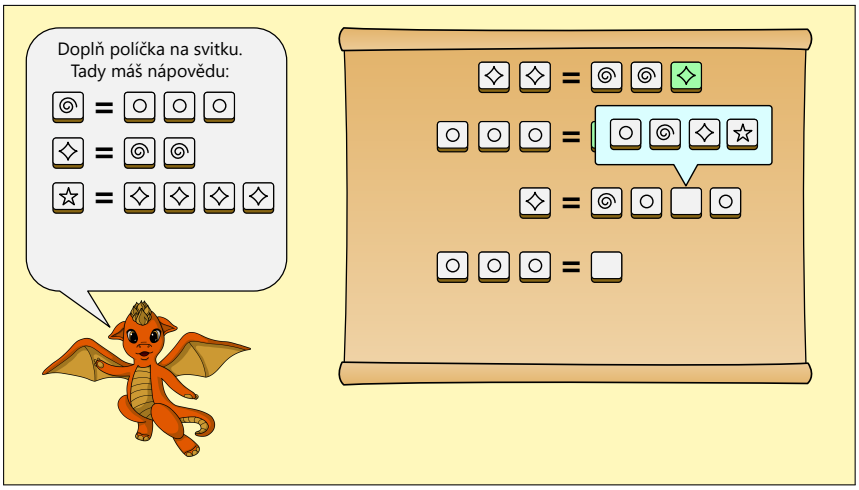
bude možné určit při tvorbě zadání. Třídy, ze kterých se modul skládá, jsou na diagramu 3.23.

První možností pro výběr obrázků je nechat tvůrce zadání samotného vybrat, které obrázky chce, aby reprezentovaly které číslo. Tento přístup umožňuje tvůrci zadání použít budoucím řešitelům již známé obrázky. Dále je jednodušší a smysluplnější odlišit v různých zadáních různé hodnoty obrázků. Snadněji je tedy možné vytvářet větší počet rozmanitých zadání.

Druhou možností je obrázky, které bude možné pro reprezentaci čísel použít, zvolit předem a tvůrce zadání pouze nechat z daných obrázků vybrat. Zde je však výrazná nevýhoda v tom, že obrázky nemohou být snadno přizpůsobeny potřebám uživatelů. Mít předem připravené obrázky je ale jednodušší řešení. Přestože se jedná o složitější možnost, pro ostatní výhody jsem se rozhodl umožnit nahrávání vlastních obrázků.

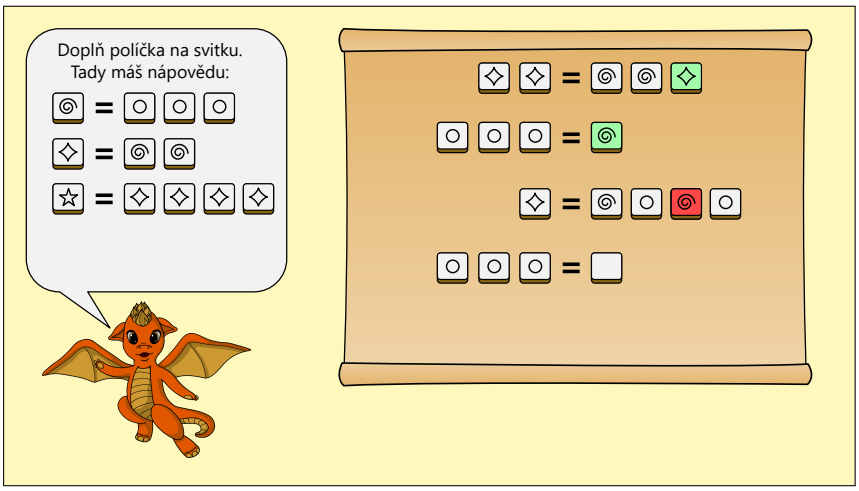
3.6.1 Vzhled modulu

Na levé straně uživatelského rozhraní modulu je Dráček s nápovědou. Na stejném místě se vyskytuje také zadání převodů ve formě jednoduchých rovnic, na jejichž levé straně je vždy jen jeden obrázek a na pravé straně vždy určitý počet jiného obrázku. Nikdy v jednom zadání převodu nebudou celkem více než dva různé obrázky. Napravo od Dráčka je seznam příkladů, které je ke splnění cvičení potřeba správně vyplnit. Po kliknutí na nedoplněné pole se uživateli zobrazí nabídka všech možných obrázků, ze kterých musí vybrat ten správný. Vzhled uživatelského rozhraní je na obrázku 3.24.



Obrázek 3.24: Wireframe cvičení Převody soustav s výběrem možného doplnění

Špatně zvolené odpovědi budou v rovnicích zvýrazněny červeně, zatímco správné odpovědi budou mít zelenou barvu. Ukázka špatně zodpovězeného příkladu je na obrázku 3.25. Po správném vyplnění všech dostupných příkladů je cvičení úspěšně dokončeno.

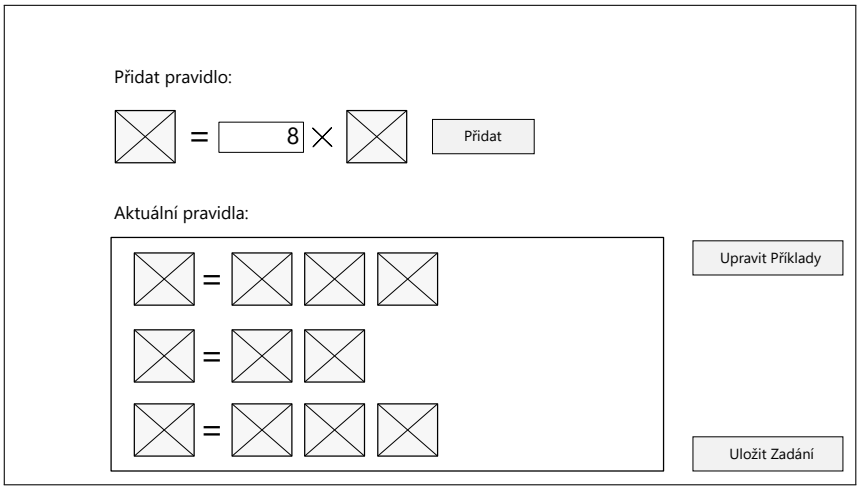


Obrázek 3.25: Wireframe cvičení Převody soustav se špatným vyplněním

3.6.2 Editor modulu

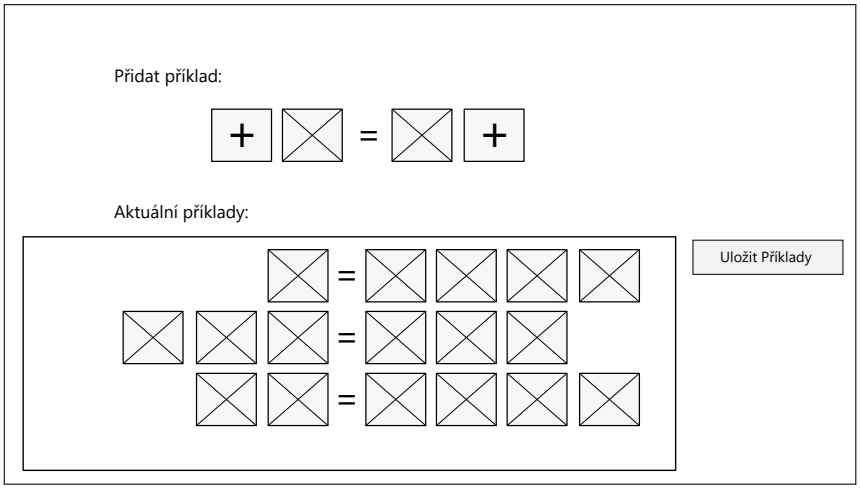
Editor modulu Převody soustav se skládá z dvou částí: editoru převodů a editoru rovnic. V editoru převodů, který je na obrázku 3.26, je možné nastavit

obrázek pro právě přidávanou jednotku. Po jeho nastavení je možné určit počet a jednotku, kterou je možné nově přidávanou jednotku převést. Přidané jednotky je možné odstranit, pokud nejsou použity v jiných jednotkách.



Obrázek 3.26: Wireframe editoru převodů pro cvičení Převody soustav

Editor rovnic se velmi podobá editoru převodů. Není zde již potřeba zvolit obrázek pro jednotku a počet je přidáváním obrázků možné určit na obou stranách rovnice. Uživatelské rozhraní této části editoru je na obrázku 3.27.



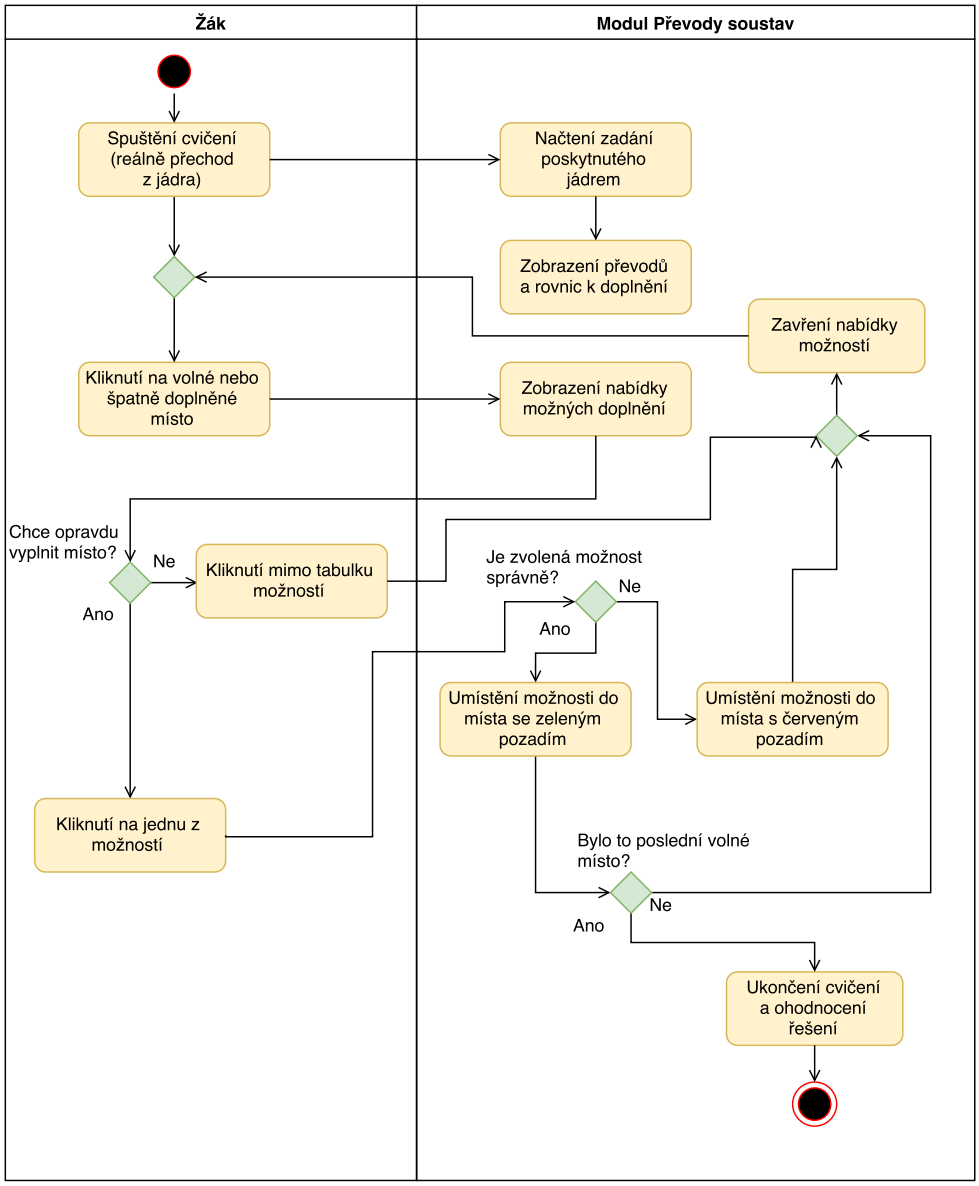
Obrázek 3.27: Wireframe editoru rovnic pro cvičení Převody soustav

3.6.3 Chování modulu

- **Kliknutí na volné pole v rovnici** → zobrazení nabídky možných doplnění
- **Kliknutí na jedno z možných doplnění** → doplnění vybraného obrázku do rovnice
- **Kliknutí na jinam** → žádná reakce, případné zavření nabídky možných doplnění

3.6.4 Průběh modulu

Po přechodu do modulu z jádra modul načte nastavené převody a rovnice, které zobrazí uživateli k vyplnění. Uživatel jednotlivé rovnice kliknutím na volná místa a následně kliknutím na zvolenou možnost vyplní. Každé vyplnění je hodnoceno samostatně a okamžitě. Celý průběh je znázorněn na obrázku 3.28.



Obrázek 3.28: Diagram aktivity modulu Převody jednotek

Kapitola 4

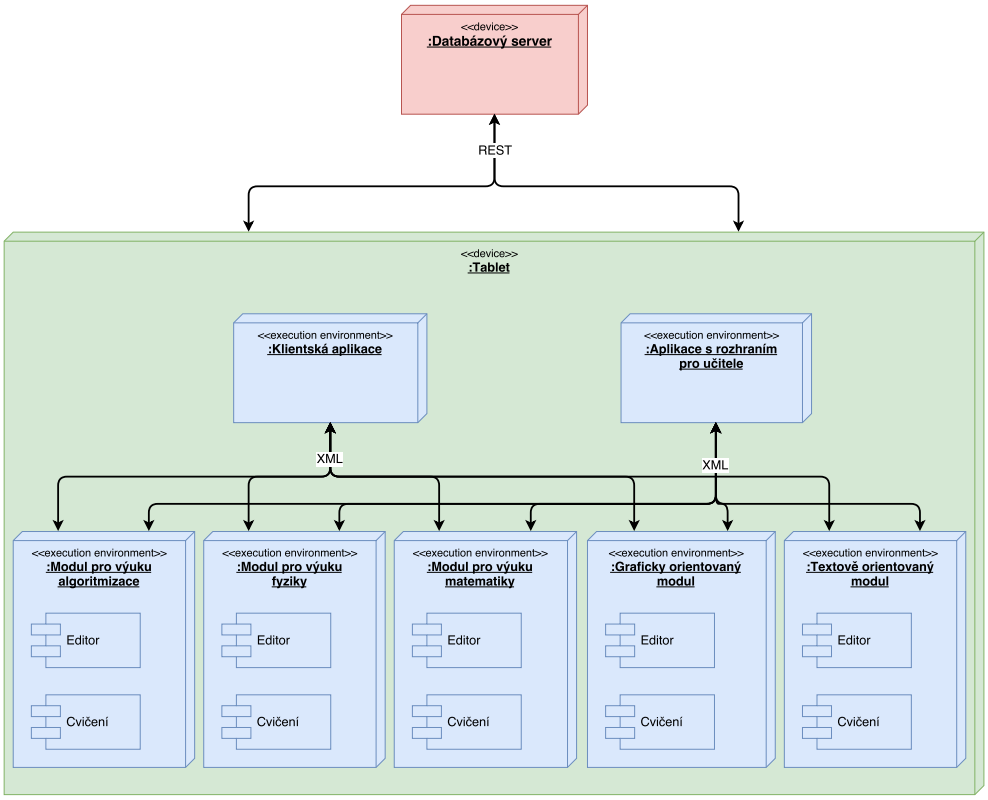
Implementace

V této kapitole se budu zabývat nasazením hotových modulů, knihovnou společnou pro všechny moduly, spojením dvojicí podobných modulů do jedné aplikace, jak jednotlivé moduly nainstalovat a používat. Popíši, jak připravit serverovou aplikaci a přidat do ní nový modul. Dále uvedu, jak připravit prostředí pro kompilaci a další vývoj mých modulů. Nakonec bude v kapitole uvedeno, jak lze změnit grafické provedení modulů, jelikož navržený vzhled není finální.

4.1 Schéma nasazení

Serverová část aplikace Dráček je nasazena na samostatném (virtuálním) zařízení, se kterým následně komunikují veškerá zařízení s klientskou částí aplikace. Komunikace probíhá pomocí přenosu prostřednictvím HTTP. Klientské části aplikace jsou dvě: jádro, které je určené pro studenty, a učitelská aplikace, ve které je možné upravovat a tvořit zadání. Obě klientské části dále spouští jednotlivé moduly. Ty poskytují vlastní editor a logiku samotných cvičení. Celkový přehled je na obrázku 4.1.

Zatímco serverová část bude nejspíše i v budoucnu vyžadovat ruční instalaci, klientské části, včetně jednotlivých modulů, mohou využít platformy Google Play Store ke snadné instalaci.



Obrázek 4.1: Schéma nasazení serveru, klientů a modulů, převzato z [4]

4.2 Společná knihovna

Vzhledem k tomu, že nemalá část kódu bude sdílena mezi všemi moduly projektu Dráček 3, rozhodli jsme se s kolegy Jaroslavem Štěpánem a Jaroslavem Rybou tuto funkcionalitu oddělit do knihovny, kterou následně všichni použijeme. Implementace společné knihovny vychází ze společné knihovny pro moduly Michala Bureše. Je v ní implementována zejména komunikace s klientskou aplikací, obsahuje ale také například společné části uživatelského rozhraní.

4.2.1 Spojení podobných modulů

Moduly byly navrhované po dvojicích, kdy druhý modul je vždy malou úpravou modulu prvního. Z tohoto důvodu jsem se rozhodl tyto dvojice modulů spojit do jedné aplikace. Díky tomu, že klientské části si informaci o tom, jak jednotlivé moduly a jejich editory spouštět, stahují ze serveru, je možné tento způsob spojení realizovat.

Alternativní možnost oddělení společného kódu do samostatné knihovny, kterou by příslušné moduly používaly, jsem nevolil, jelikož moduly sdílejí veškerou logiku cvičení a liší se hlavně uživatelským rozhraním.

4.3 Instalační příručka

Pro tvorbu nových modulů, či úpravu modulů vytvořených v této práci, je třeba nejprve nainstalovat následující software:

- **Android Studio** Oficiální IDE pro vývoj na systém Android od společnosti Google. Nainstalovat lze pomocí volně dostupného instalátoru, který lze nalézt na jeho oficiálních stránkách.
- **Android SDK** Instalaci SDK lze provést pomocí vývojového prostředí Android Studio, pro účely automatizovaného sestavení lze také na jeho oficiálních stránkách nalézt samostatný instalátor. Všechny moduly využívají Android SDK verze 25 se zpětnou kompatibilitou až k SDK verze 21, která odpovídá systému Android 5.0 (Lollipop).
- **Gradle** Build systém, který je využíván Android SDK, obvykle není třeba instalovat samostatně, jelikož je součástí prostředí Android Studio. Samostatná instalace je vhodná v případě automatizovaného sestavení, kdy není třeba instalovat kompletní vývojové prostředí.

Dále je zapotřebí získat zdrojové kódy společné knihovny, které jsou dostupné na serveru GitLab FIT ČVUT [23]. Aby bylo možné knihovnu použít v novém modulu, je třeba upravit následující soubory projektu:

- **settings.gradle** Upravit následovně první řádku

  ```
  include ':app', ':corelibrary'
  ```

  a přidat řádku

  ```
  project(':corelibrary').projectDir =
   new File(settingsDir, '../DracekModulesCore/corelibrary')
  ```

  Cestu ke knihovně (DracekModulesCore) lze libovolně upravit.

- **build.gradle** Přidat do sekce závislostí řádku

  ```
  compile project(':corelibrary')
  ```

Moduly vytvořené v této práci jsou nastavené tak, že očekávají složku s knihovnou právě v nadřazené složce.

4.3.1 Instalace serverové aplikace

Součástí práce Filipa Ondřeje projektu Dráček 2 je také návod na nasazení serverové aplikace [24]. Ten však počítá s tím, že je k dispozici již existující instalace, ze které je možné soubory zkopírovat. V případě, že jsou k dispozici pouze zdrojové soubory aplikace, je třeba provést dodatečné kroky:

- **Příprava prostředí** Nejprve je potřeba nainstalovat softwarové závislosti tak, jak je popsáno v původním návodu. Může zde nastat problém, že balíček

  ```
  libapache2-mod-auth-mysql
  ```

  nebude v novějších operačních systémech k dispozici kvůli nekompatibilitě s aktuální verzí webového serveru Apache. V takovém případě, jelikož jsem nedohledával, zda je balíček stále potřeba, doporučuji použít starší verzi systému, kde se balíček stále nachází.

- **Composer** Manažer závislostí Composer je potřeba kvůli stažení frameworku, na kterém je aplikace postavena, a kvůli stažení závislostí aplikace nainstalovat. Instalaci lze provést podle oficiálních instrukcí.

- **Stažení frameworku** Před nahráním zdrojových souborů serverové aplikace je třeba stáhnout sandbox projekt frameworku Nette. K tomu lze využít manažer Composer příkazem

  ```
  composer create-project nette/sandbox dracek-nette
  ```

  kdy posledním argumentem je cesta, kde bude uložena serverová aplikace.

- **Nahrání souborů serveru** Soubory serveru je potřeba nahrát do složky, kam byl stažen sandbox projekt. V případě konfliktu lze použít soubor serverové aplikace. Dále je potřeba podle původního návodu provést nastavení serveru.

- **Stažení závislostí serveru** Po nahrání souborů je nutné příkazem

  ```
  composer install
  ```

  ve složce serveru stáhnout veškeré závislosti.

- **Úprava práv** K tomu, aby server mohl fungovat správně, je potřeba, aby uživatel, pod kterým poběží proces Apache, měl přístup k následujícím složkám (relativně k složce projektu):

  ```
  ./www/images
  ./www/exercises
  ./www/scores
  ./www/module\_exercise
  ```

- **Nastavení Apache** Vzhledem k tomu, že serverová aplikace využívá přepisovací pravidla, je potřeba zapnout modul přepisování pomocí příkazu

  ```
  a2enmod rewrite
  ```

  Nakonec je potřeba restartovat webový server Apache.

4.4 Uživatelská příručka

Pro veškeré interakce s editory a cvičeními je předpokládáno, že uživatel má nainstalovanou příslušnou klientskou část (jádro nebo učitelskou aplikaci).

4.4.1 Instalace modulů

V době psaní této práce není možné moduly automaticky stáhnout z klientské části. Automatická instalace funguje za předpokladu, že modul, který má být nainstalován, je v instalovatelné formě (.apk) umístěn ve složce

`Download/dragon/`

v interním úložišti zařízení. Instalační soubor modulu musí být pojmenován podle identifikátoru modulu v databázi.

Moduly lze také instalovat zcela ručně spuštěním příslušných instalačních souborů. V takovém případě nezáleží na pojmenování instalačního souboru.

4.4.2 Modul Labyrinth

V tomto modulu je cílem uživatele dostat Dráčka z jeho počáteční pozice do východu z labyrintu. Toho musí docílit složením správného algoritmu z dostupných akcí. Dráček umí krom chůze také chrlit oheň, čímž může z labyrintu odstranit dřevěnou zeď, která stojí před ním. Akcemi lze Dráčka také otočit, nebo mu přikázat sebrat klíč, který automaticky použijte při průchodu zavřenými dveřmi.

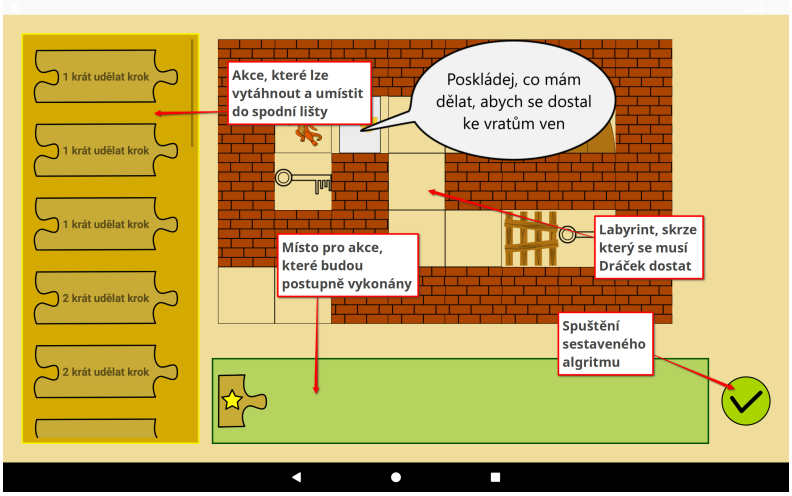
Pokud provedení sestaveného algoritmu nedostane Dráčka do východu z labyrintu, je labyrint vrácen do stavu, v jakém byl před začátkem provádění algoritmu.

Ovládání cvičení

Uživatelské rozhraní cvičení, které je na obrázku 4.2, skládá dvou hlavních částí:

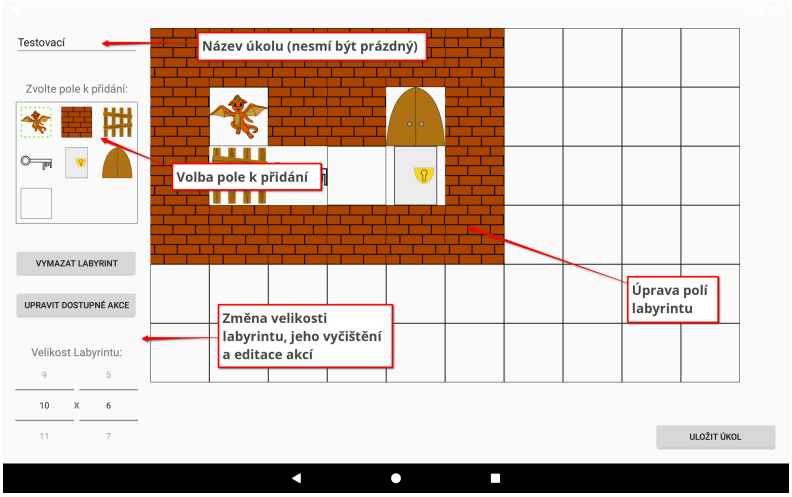
- **Seznam dostupných akcí** Svislý seznam v levém rohu obrazovky, ze kterého lze tažením vyjmout libovolnou akci a umístit ji do algoritmu upuštěním nad seznamem akcí algoritmu. Akce bude umístěna za akci, nad kterou byla upuštěna.
- **Seznam akcí algoritmu** Vodorovný seznam ve spodní části obrazovky. Akce budou po spuštění algoritmu vykonávány v pořadí, v jakém jsou umístěny v tomto seznamu. Lze je tažením vyjmout a změnit tím jejich pořadí.

Po složení algoritmu je možné jej spustit stiskem zeleného tlačítka v pravém dolním rohu obrazovky.



Obrázek 4.2: Uživatelské rozhraní cvičení Labyrint

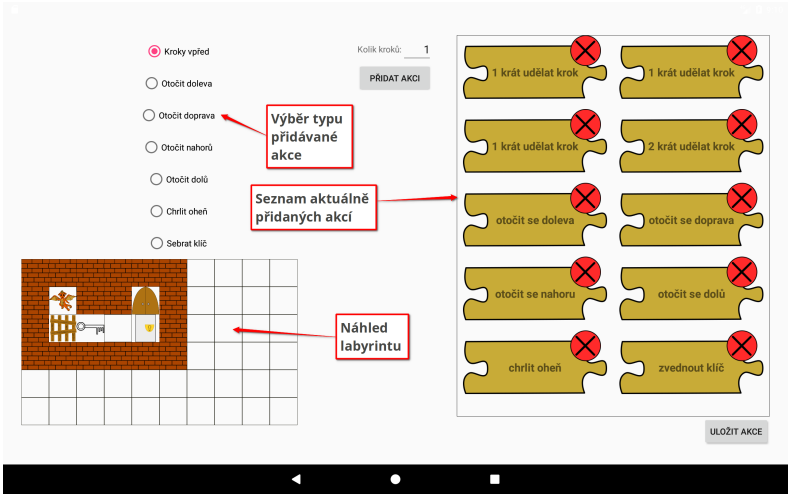
Ovládání editoru



Obrázek 4.3: Uživatelské rozhraní editoru labyrintu s popisem

Editor umožňuje pro jedno zadání tvořit libovolný počet úkolů. Úvodní sekce editoru, ve které je možné zvolit úkol k editaci, požádat o vytvoření nového úkolu, smazat existující úkol a uložit celé zadání, je společná pro všechny moduly.

Uživatelské rozhraní editoru má dvě části: editor labyrintu, jehož rozhraní je s popisem na obrázku 4.3, a editor dostupných akcí, který je na obrázku 4.4. Editor labyrintu obsahuje v levé části obrazovky (shora):



Obrázek 4.4: Uživatelské rorzhraní editoru akcí s popisem

- **Název úkolu** Orientační název aktuálně editovaného úkolu, který je v editoru otevřen.
- **Výběr pole** Volba pole, které lze kliknutím do labyrintu na zvolenou pozici umístit.
- **Vymazat labyrint** Možnost nastavit všechna pole v labyrintu na prázdná.
- **Nastavit dostupné akce** Otevře editor dostupných akcí.
- **Nastavení velikosti labyrintu** Změní velikost labyrintu se zachováním existujících polí.

Editor dostupných akcí obsahuje výběr pro volbu, kterou akci do seznamu dostupných akcí přidat. V prostřední části editoru je možné nastavit počet opakování kroků pro příslušnou akci a tlačítkem zvolenou akci přidat. V pravé části editoru se nachází aktuální seznam dostupných akcí, které je kliknutím na křížek možné odstranit. Pro lepší přehled při přidávání akcí je také součástí editoru pohled na aktuální nastavení labyrintu.

4.4.3 Modul Labyrinth 2

Tento modul je variací modulu Labyrint. Cíl je zde stejný, není však možné přidávat do algoritmu nové akce, pouze označovat ty, které se nemají vykonat. Algoritmus následně za běhu přeskakuje označené akce.

Ovládání cvičení

Cvičení Labyrint 2 má téměř stejné uživatelské rozhraní, jako modul Labyrint. Zásadním rozdílem je absence seznamu dostupných akcí. Ze seznamu akcí algoritmu již nelze jednotlivé akce vytahovat. Kliknutím na akci ji je pouze možné označit k přeskočení. Zbytek rozhraní je nezměněn.

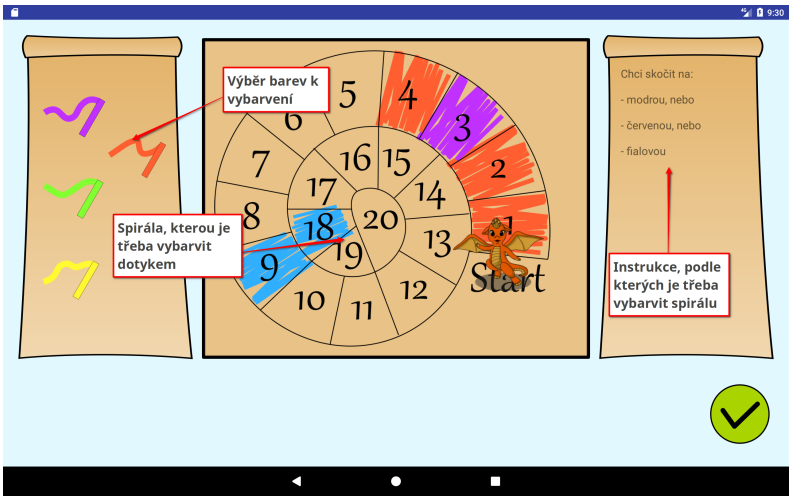
Ovládání editoru

Editor modulu Labyrint 2 vychází z editoru modulu Labyrint. Úprava labyrintu zůstává stejná, rozdílný je pouze editor dostupných akcí. Místo pouhého seznamu je zde seřazený seznam akcí tak, jak budou prováděny v algoritmu. Lze je tažením přeskupit a kliknutím na křížek odstranit.

4.4.4 Modul Skákací panák

Cílem tohoto modulu je vybarvit skákacího panáka reprezentovaného spirálou podle zadaných pravidel. K dispozici je na výběr pět odlišných barev a některá z polí jsou již předvybarvená. Na spirále se vyskytuje dvacet polí, na které bude Dráček postupně po potvrzení řešení skákat. Dráček skáče vždy na všechna pole, nebo jen na sudá, nebo lichá. Pokud Dráček skočí na jiné pole, než je řečeno v zadání, vrátí se na původní pozici. Spirála zůstane vybarvená tak, jak byla před potvrzením řešení.

Ovládání cvičení



Obrázek 4.5: Uživatelské rozhraní cvičení Skákací panák

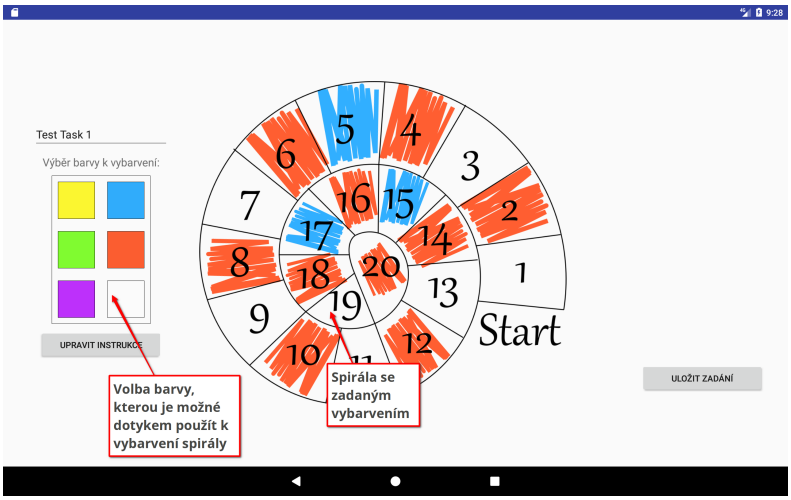
Rozhraní cvičení Skákací panák, které je na obrázku 4.5, lze rozdělit na dvě hlavní části:

- **Výběr barvy** Na levé straně obrazovky je kliknutím možné vybrat jednu z pěti nabízených barev. Po jejím zvolení bude barva použita při vybarvování spirály.
- **Spirála** V prostřední části obrazovky je umístěna spirála, která se skládá z dvaceti polí. Každé pole lze kliknutím vybarvit právě zvolenou barvou.

V pravé části rozhraní jsou k dispozici instrukce, podle kterých by měla být spirála vybarvena. Po vybarvení spirály lze řešení potvrdit stiskem zeleného tlačítka v pravém dolním rohu obrazovky.

Ovládání editoru

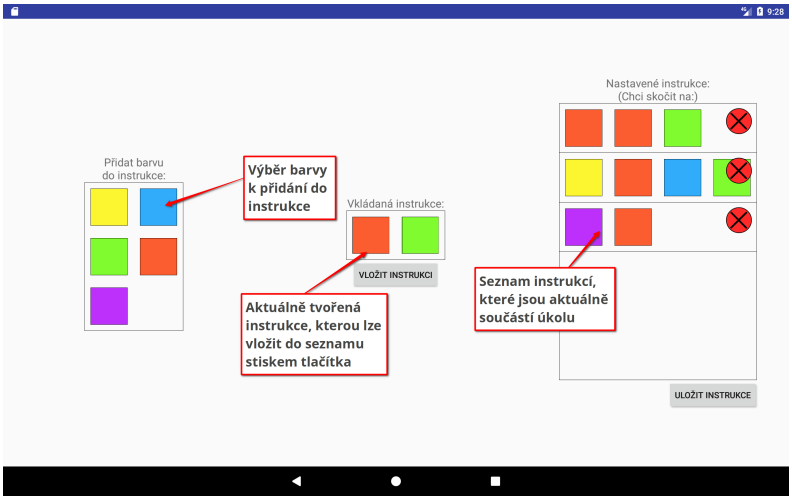
Úvodní částí editoru modulu je výběr úkolu k editaci, stejně jako u modulu Labyrint.



Obrázek 4.6: Uživatelské rorzhraní editoru panáka s popisem

Samotný editor se skládá ze dvou částí: úpravy spirály, který je k vidění na obrázku 4.6, a nastavení instrukcí, které je na obrázku 4.7. V části pro úpravu spirály je také možné nastavit název úkolu. K vybarvení spirály je k dispozici kromě všech pěti barev navíc možnost pole odbarvit. Po zvolení barvy lze kliknutím vybarvovat jednotlivá pole.

Část nastavení instrukcí obsahuje vlevo výběr barev, které lze do instrukce vložit. Uprostřed je vidět aktuálně skládaná instrukce, na jejíž barvy bude muset Dráček skákat za sebou, a tlačítko pro její přidání. Na pravé straně jsou již vytvořené instrukce v seznamu. Každou hotovou instrukci lze odstranit



Obrázek 4.7: Uživatelské rozhraní editoru instrukcí s popisem

stiskem kliknutím na křížek. U spodního rohu rozhraní je možné nastavit po kterých polích bude Dráček skákat.

4.4.5 Modul Skákací panák 2

Modul Skákací panák 2 je pouze úpravou modulu Skákací panák. Opět je cílem splnit zadané instrukce. Místo vybarvování spirály je však součástí zadání již plně vybarvená spirála a úkolem je položit kameny, které Dráček přeskočí, na pole tak, aby byly splněny instrukce.

Ovládání cvičení

Uživatelské rozhraní cvičení se od předchozího modulu liší tím, že již nelze zvolit barvu k vybarvení spirály. Místo toho je možné kliknutím na pole spirály položit na něj kámen. Po vyplnění spirály lze potvrdit řešení tlačítkem v pravém dolním rohu. V případě špatného vyplnění umístěné kameny zůstávají na svém místě.

Ovládání editoru

Jedinou změnou v editoru cvičení oproti modulu Skákací panák je znemožnění uložení úkolu se spirálou, která není plně vybarvena.

4.4.6 Modul Převody soustav

V tomto modulu má uživatel za cíl doplnit všechny zadané rovnice tak, aby platily podle zadaných převodů. Převody jsou zadány vždy ve tvaru větší

jednotky rovné určitému počtu menší jednotky. Každé doplnění je hodnoceno okamžitě, cvičení je dokončeno úspěšném vyplnění všech rovnic.

Ovládání cvičení

Hlavní částí uživatelského rozhraní cvičení je zobrazení rovnic na pravé straně obrazovky s volnými místy k vyplnění. Po kliknutí na volné místo se nad ním zobrazí nabídka pro výběr jedné ze zadaných jednotek. Jednotku lze vybrat kliknutím. Následně nabídka pro výběr zmizí a jednotka je doplněna na zvolené místo se zeleným pozadím, pokud byla vybrána správně, a červeným pozadím, pokud ne.

Na levé straně uživatelského rozhraní je k dispozici nápověda a žádané převody, podle kterých mají být rovnice vyplněny.

Ovládání editoru

Editor modulu se skládá z části nastavení převodů a části zadání rovnic. V první části je možné tvořit nové přechody vyplněním hodnot:

- **Větší jednotka** Na levé straně rovnice lze kliknutím na volné políčko zvolit obrázek, který bude reprezentovat větší jednotku v převodu.
- **Volba počtu** Za rovnítkem je číselné nastavení počtu menší jednotky převodu.
- **Menší jednotka** Za volbou počtu je možné kliknutím na volné políčko zvolit obrázek menší jednotky.

Přidané převody jsou zobrazena v seznamu pod nastavením nového převodu. Kliknutím na křížek v seznamu vymaže příslušný převod. Pro úpravu rovnic k řešení je třeba přesunout se do druhé části editoru kliknutím na tlačítko Upravit Příklady.

Druhá část editoru obsahuje možnost přidání nové rovnice postupným přidáváním jednotek na libovolnou stranu rovnice. Přidat jednotku lze stiskem plusu na zvolené straně. Všechny rovnice jsou zobrazeny v seznamu pod možností přidání a opět je lze kliknutím na křížek smazat.

4.5 Úprava grafického provedení

Aktuální verze modulů obsahují grafické provedení, které není finální. V budoucnu budou by moduly měly být po grafické stránce sjednoceny. Pro úpravu vzhledu jednotlivých prvků je třeba nahradit soubory ve složce příslušného projektu:

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

Za účelem změny rozložení jednotlivých prvků je nutné upravit soubory v projektové složce:

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

Soubory doporučuji pro jednoduchost upravovat pomocí IDE Android Studio. U rozložení je ale třeba zachovat nastavené identifikátory prvků.

Některé moduly obsahují animace. Pro jejich úpravu je třeba změnit části kódu, které tyto animace vytvářejí, zejména pokud požadovaná nová animace obsahuje například několik obrázků, které se mají střídat. V tabulce 4.5 jsou soubory projektů a animace, které obsahují. Všechny cesty jsou relativně ke složce v příslušném modulu:

```
app/src/main/java/cz/cvut/fit/dracek/
```

Soubor	**Animace**
Moduly Labyrint a Labyrint 2 (složka labyrinthwalk)	
LabyrinthFragment.java	Animace pohybu Dráčka v labyrintu, změny polí v labyrintu.
BuildExerciseActivity.java	Animace resetu labyrintu po dokončení neúspěšného algoritmu.
SkipExerciseActivity.java	Animace resetu labyrintu po dokončení neúspěšného algoritmu (Labyrint 2).
Moduly Skákací panák a Skákací panák 2 (složka hopscotch)	
HopscotchFragment.java	Animace skákání Dráčka po polích panáka

4.6 Omezení vyvinutých prototypů

Téměř veškerá navržená funkcionalita modulů a jejich editorů byla implementována. U modulů Labyrint a Labyrint 2 prozatím chybí kontrola v editoru, která by měla zaručit splnitelnost vytvořeného labyrintu bez ohledu na dostupné akce. Tato funkce chybí jednak z důvodu špatného plánování a nedostatku času, stojí ale také za zamyšlení, jestli by nebylo lepší z hlediska výkonu aplikace kontrolovat splnitelnost pomocí dostupných akcí.

Dalším nedostatkem je grafické zpracování modulů, které je třeba v budoucnu před jejich nasazením dokončit. Nejvíce se nedostatek projevuje u modulů s labyrintem, ve kterých může současná grafika mást uživatele.

Kapitola 5

Testování

Každý z modulů byl podroben několika druhům testů. Jednalo se o unit testy, testování integrace s jádrem a aplikací pro učitele, systémový test pro každý modul a uživatelské testování s dětmi.

5.1 Unit testy

Jednotkové (unit) testy kontrolují správnost funkce částí kódu samostatně. Ve většině případů se zaměřují na jednotlivé metody tříd, jejich výsledky, popřípadě i vedlejší efekty. V kontextu vývoje pro systém Android navíc existuje rozdělení unit testů na lokální unit testy, které jsou spouštěny na nativním JVM systému, a instrumentované unit testy, které jsou naopak spouštěny na připojeném zařízení s operačním systémem Android, nebo na emulátoru [25].

Pro realizaci unit testů jsem použil nástroje poskytnuté spolu s instalací IDE Android Studio. Jedná se hlavně o testovací framework JUnit. Ke kontrole vedlejších efektů jsem použil mockovací knihovnu Mockito.

Testoval jsem hlavně implementaci průběhu jednotlivých cvičení. Například v modulu Labyrint jsem testoval jednotlivé akce a jejich výsledný efekt, zejména úpravu polí v labyrintu. Tyto testy mi pomohly odhalit chyby vzniklé při úpravě vyhodnocování akcí. Pomocí instrumentovaných testů jsem kontroloval funkcionalitu tříd, kterými v modulech zajišťuji zobrazování centrálních prvků cvičení, jako je například skákací panák.

5.2 Integrační testy

Integrační testy zjišťují, zda jednotlivé komponenty spojené do jednotného celku mezi sebou správně komunikují, za účelem odhalení problémů týkajících se interakce komponent.

Testoval jsem komunikaci mých modulů s jádrem a aplikací pro učitele. Aby bylo testování možné, bylo nejdříve potřeba spustit testovací server apli-

kace Dráček. Dále bylo potřeba upravit jak jádro, tak aplikaci pro učitele. Ve verzích, které byly použity pro testování, byla pevně daná adresa serveru, kterou bylo potřeba změnit. Navíc bylo v jádru nutné opravit způsob, jakým stahuje a posílá zadání modulu, aby odpovídal testovací implementaci v jádru, která používala pevně specifikované zadání.

Výsledky integračního testování mi pomohly opravit drobné chyby v ukládání a načítání zadání, nicméně díky spolupráci s kolegy Jaroslavem Rybou a Jaroslavem Štěpánem bylo propojení s jádrem a aplikací pro učitele úspěšné.

5.3 Systémové testy

Systémové testy probíhají po dokončení integračních testů a kontrolují všechny části systému spojené jako celek. Po otestování integrace všech modulů jak s jádrem, tak s aplikací pro učitele, jsem úspěšně otestoval celkový proces práce s modulem od vytvoření nového zadání, jeho splnění skrz jádro, úpravu zadání a následné opětovné splnění.

Systémové testy potvrdily funkční spolupráci mezi modulem a klientskými aplikacemi.

5.4 Uživatelské testování

Uživatelské testování se zaměřuje na interakci uživatele s rozhraním softwaru. Testování má za úkol zjistit, zda je uživatelské rozhraní přívětivé a zda se snadno používá. Uživatelské testy budou probíhat ve spolupráci s Usability laboratoří na Fakultě informačních technologií, ČVUT.

Testování bude naplánováno na jeden den, kdy budou testovány k tomu připravené moduly ze všech tří skupin vyvíjených pro projekt Dráček 3. Subjekty uživatelského testování budou žáci prvního stupně základní školy. Po přivítání žáků moderátorem s nimi bude vyplněn vstupní dotazník a následně budou umístěni do monitorované místnosti. Zde s nimi moderátor projde zvolený scénář, zatímco vývojář testovaného modulu bude vzdáleně průběh testování sledovat. Nakonec moderátor s žáky vyplní výstupní dotazník.

Veškeré materiály k uživatelskému testování jsou k dispozici na `https://goo.gl/iD0aB9`.

5.4.1 Testovací dotazníky

Pro uživatelské testování nových modulů bylo zapotřebí vytvořit nové vstupní a výstupní dotazníky. Oba dotazníky je možné vyplnit elektronicky. Vstupní dotazník má za úkol zjistit vztah žáků k předmětům, kterých se testované moduly týkají, a zkušenosti žáka s podobnými zařízeními. Na základě vyplnění vstupního dotazníku budou vybrány scénáře, podle kterých bude testování po-

kračovat. Výstupní dotazníky se žáka ptají na snadnost ovládání testovaných modulů a celkové dojmy.

5.4.2 Výsledky testování

Z organizačních důvodů a po dohodě s vedoucím práce proběhne uživatelské testování až po odevzdání práce. Výsledky budou předloženy u obhajoby práce.

Závěr

Tato práce se zabývala rozšířením počtu modulů pro aplikaci Dráček o pět nových modulů zaměřených na výuku základů algoritmizace. V analýze jsem zhodnotil stav aktuálně dostupných aplikací, které se zabývají výukou algoritmizace, matematické logiky a převodu číselných soustav. Zjistil jsem, že žádná z analyzovaných aplikací není vhodná pro výuku dětí na prvním stupni základní školy. Součástí analýzy bylo také otestování jednoho z modulů vzniklých pro projekt Dráček 2. Výsledkem této analýzy a na základě Rámcového vzdělávacího programu pro základní vzdělání jsem zformuloval požadavky pro pět nových modulů.

Na základě analýzy jsem navrhl nové moduly, spolu s editory pro tvorbu zadání do jednotlivých modulů. Nové moduly jsem cílil pro žáky prvního stupně základní školy, což se odráží na jednoduchém ovládání cvičení. Všechny výsledné moduly prošly jednotkovými, integračními a systémovými testy, které pomohly odstranit chyby v implementaci. Za účelem testování byly upraveny aplikace jádra a aplikace pro učitele. Při spouštění testovacího serveru jsem narazil na nedostatky v návodu na nasazení serveru a úpravy tohoto návodu jsem uvedl v kapitole implementace.

Všechny body zadání byly splněny, nicméně vytvořené moduly nejsou ve finálním stavu, zejména z hlediska grafického provedení, které by bylo vhodné sjednotit s ostatními moduly. Zároveň v některých modulech není implementována veškerá navržená funkcionalita, převážně z časových důvodů.

Do budoucna bych chtěl v práci na modulech pokračovat rozšířením jejich možností pro zadání. Pro moduly Labyrint a Labyrint 2 bych chtěl přidat další druhy akcí a nová pole, jako například tlačítka nebo i nepřátele na vyhýbání. V modulech Skákací panák a Skákací panák 2 bych chtěl zavést automatické vytvoření spirály podle zadané velikosti.

Literatura

[1] Evollve, Inc.: *OzoBlocky*. [online]. [cit. 2017-4-16]. Dostupné z: `http://ozoblockly.com/editor`

[2] Bomerová: *Děda Lesoň*. [online]. [cit. 2017-1-9]. Dostupné z: `http://www.bomerova.cz/deda-leson`

[3] BUREŠ, M.: *Výuková aplikace Dráček II - zásuvné moduly II*. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

[4] ŠTĚPÁN, J.: *Zásuvné moduly aplikace Dráček III - výuka fyziky*. Praha, 2017. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

[5] TechSophia: *Jdu do školy!* [online]. [cit. 2017-4-16]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.techsophia.schoolreadiness`

[6] NC Lab inc.: *Karel Coding: Code Hour*. [online]. [cit. 2017-1-9]. Dostupné z: `https://play.google.com/store/apps/details?id=example.com.hocapp`

[7] Foundation, S.: *ScratchJr*. [online]. [cit. 2017-1-9]. Dostupné z: `https://play.google.com/store/apps/details?id=org.scratchjr.android`

[8] Games, A.: *IQ Test*. [online]. [cit. 2017-2-16]. Dostupné z: `https://play.google.com/store/apps/details?id=com.ame3.iq2`

[9] increatly: *Binary Challenge*. [online]. [cit. 2017-1-9]. Dostupné z: `https://play.google.com/store/apps/details?id=com.increatly.binarychallenge`

[10] KB, S.: *Binary Practice*. [online]. [cit. 2017-1-9]. Dostupné z: `https://play.google.com/store/apps/details?id=com.stevekb.binary`

[11] RYBA, J.: *Zásuvné moduly aplikace Dráček III - výuka matematiky.* Praha, 2017. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

[12] *Rámcový vzdělávací program pro základní vzdělání.* [online]. [cit. 2017-1-9]. Dostupné z: http://www.nuv.cz/uploads/RVP_ZV_2016.pdf

[13] ZŠ Korenského: *O naší škole.* [online]. [cit. 2017-1-9]. Dostupné z: http://www.zskorenskeho.cz/skola/

[14] STC Ostrava: *Vzdělávací programy - základní školy (1. stupeň).* [online]. [cit. 2017-1-9]. Dostupné z: http://skola.stcostrava.cz/cs/vzdelavaci-programy-zakladni-skoly-1-stupen

[15] Křížová, S.: *Učivo, matematická témata.* [online]. [cit. 2017-1-9]. Dostupné z: http://docplayer.cz/13654912-Ucivo-matematicka-temata-1-matematicka-logika.html

[16] ZÁKLADNÍ ŠKOLA HLUK: *Převody číselných soustav.* [online]. [cit. 2017-1-9]. Dostupné z: http://webcache.googleusercontent.com/search?q=cache:NGeJ1KPeMfkJ:www.zshluk.cz/dokumenty/informatika/prevodciselnychsoustav.pdf+&cd=9&hl=en&ct=clnk&gl=cz

[17] Bomerová, E.: *Děda Lesoň.* [online]. [cit. 2017-1-9]. Dostupné z: http://bomerova.cz/deda-leson

[18] MAZEL, M.: *Dragon II - plugins I.* Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

[19] VEJVODA, M.; RYTÍŘ, M.: *Programovací jazyk KAREL.* Český Krumlov, 2001, ISBN 99972-03-09-7. Dostupné z: http://pckarel.sweb.cz/pdf/pjkarel.pdf

[20] LIFELONG KINDERGARTEN GROUP AT THE MIT MEDIA LAB: *Scratch - Imagine, Program, Share.* [online]. [cit. 2016-12-12]. Dostupné z: https://scratch.mit.edu/about/

[21] RESNICK, M.; SILVERMAN, B.; KAFAI, Y.; aj.: Scratch. In *Communications of the ACM*, ročník 52, 2009-11-01, ISSN 00010782, s. 60–67, doi:10.1145/1592761.1592779. Dostupné z: http://portal.acm.org/citation.cfm?doid=1592761.1592779

[22] Sidorov, K.: *Matematická logika.* [online]. [cit. 2017-1-9]. Dostupné z: https://play.google.com/store/apps/details?id=com.do_apps.catalog_704

[23] KB, S.: *Binary Practice*. [online]. [cit. 2017-1-9]. Dostupné z: `https://play.google.com/store/apps/details?id=com.stevekb.binary`

[24] FILIP, O.: *Výuková aplikace Dráček II - serverová část a rozhraní pro učitele*. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

[25] *Test your app | Android Studio*. [online]. [cit. 2017-1-9]. Dostupné z: `https://developer.android.com/studio/test/index.html`

Příloha **A**

Seznam použitých zkratek

XML Extensible markup language

IDE Integrated development environment

HTTP Hypertext transfer protocol

Příloha **B**

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Slabý_Ondřej_2017.pdf` ..... text práce ve formátu PDF