Univerzita Hradec Králové

Pedagogická fakulta

Katedra aplikované kybernetiky

Deskové hry a hlavolamy jako motivace ve výuce algoritmizace a programování

Diplomová práce

Autor:	Bc. Pavel Tvrzník
Studijní program:	N7504 Učitelství pro střední školy
Studijní obor:	Učitelství pro střední školy – informatika Učitelství pro 2. stupeň ZŠ – anglický jazyk a literatura
Vedoucí práce:	PhDr. Michal Musílek, Ph.D.

Hradec Králové leden 2020

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem v seznamu použité literatury uvedl všechny prameny, které jsem při jejím psaní využil.

v Hradci Králové dne Bc. Pavel Tvrzník

Poděkování

Tímto bych chtěl poděkovat všem, kdo mě při psaní této diplomové práce podporovali, a samozřejmě také jejímu vedoucímu, PhDr. Michalu Musílkovi, Ph.D. Dále bych chtěl poděkovat všem, kdo si práci přečetli.

Anotace

TVRZNÍK, Pavel. *Deskové hry a hlavolamy jako motivace ve výuce algoritmizace a programování*. Hradec Králové, 2020. Diplomová práce na Pedagogické fakultě Univerzity Hradec Králové. Vedoucí diplomové práce Michal Musílek. 69 s.

Tato diplomová práce je zaměřena na deskové hry a hlavolamy a jejich využití k motivaci žáků při výuce algoritmizace a programování. Práce je rozdělena na část teoretickou a část praktickou. V rámci teoretické části je nejprve stručně popsána současná situace v oblasti výuky programování na českých školách. Poté je představen dětský programovací jazyk Scratch. Dále je nahlédnuto do historie významných deskových her a hlavolamů a podán přehled jejich typů. Následuje podrobný popis pravidel vybraných deskových her a hlavolamů, které byly následně autorem v rámci části praktické naprogramovány v programovacím jazyce Scratch. Tyto naprogramované deskové hry a hlavolamy jsou podrobně zdokumentovány v první kapitole části praktické. V praktické části je rovněž zpracováno dotazníkové šetření na téma výuky programování, které autor práce se svými žáky realizoval během ročního působení na ZŠ Sever v Hradci Králové. Výstupy z praktické části jsou k dispozici jako přílohy práce.

Klíčová slova

Deskové hry, hlavolamy, motivace, algoritmizace, programování, výuka informatiky, Scratch

Annotation

TVRZNÍK, Pavel. *Board Games and Puzzles as Motivation in Algorithmization and Programming Education*. Hradec Králové, 2020. Diploma Thesis at Faculty of Education at University Hradec Králové. Thesis supervisor Michal Musílek. 69 p.

This diploma thesis focuses on board games and puzzles and their use in motivating students during algorithmization and programming education. The thesis is divided into a theoretical part and a practical part. In the theoretical part, current state of programming education at Czech schools is briefly described. Then children's programming language called Scratch is introduced. After that, it continues with a glimpse into history of significant board games and puzzles, as well as an overview of their types. Afterwards, detailed descriptions of the rules of selected board games and puzzles are given. These board games and puzzles have been programmed by the author of the thesis, using Scratch programming language. In-depth documentation of these programs follows in the first chapter of the practical part. The practical part also covers a survey about programming education, which the author of the thesis performed during his one-year employment at ZŠ Sever in Hradec Králové. Outputs of the practical part are available as thesis attachments.

Keywords

Board games, puzzles, motivation, algorithmization, programming, IT education, Scratch

Obsah

Obsah 6

Úvod 8

1 Postavení programování v rámci českého školství 10

1.1 Zastoupení programování v RVP 10

1.2 Reforma ve výuce informatiky 10

1.3 Autorovy zkušenosti s výukou informatiky na ZŠ 11

1.4 Motivace pro programování 12

2 Scratch 14

2.1 Hlavní výhody a nevýhody aplikace Scratch 14

2.2 Scratch – používané termíny 16

3 Deskové hry a hlavolamy 18

3.1 Deskové hry 18

3.1.1 Historie deskových her 18

3.1.2 Typy deskových her 20

3.2 Hlavolamy 24

3.2.1 Historie hlavolamů 24

3.2.2 Typy hlavolamů 26

4 Konkrétní příklady deskových her a hlavolamů 29

4.1 Tic-tac-toe 29

4.2 Jezdcova procházka 30

4.3 Hanojské věže 32

5 Vlastní naprogramované deskové hry a hlavolamy v aplikaci Scratch 35

5.1 Tic-tac-toe 36

5.1.1 Sprity a proměnné 36

5.1.2 Program 37

5.2 Jezdcova procházka ... 41
5.2.1 Sprity a proměnné ... 41
5.2.2 Program ... 42
5.3 Hanojské věže ... 45
5.3.1 Sprity a proměnné ... 45
5.3.2 Program ... 47
6 Dotazníkové šetření na téma programování a jeho výuce ve škole ... 54
6.1 Výuka programovacího jazyku Scratch na ZŠ Sever ... 54
6.2 Základní údaje o dotazníku ... 55
6.3 Otázky v dotazníku ... 55
6.4 Všeobecné informace o respondentech ... 56
6.5 Odpovědi z dotazníku – uzavřené otázky ... 57
6.6 Vyhodnocení dotazníku – otevřené otázky ze sekce 3 ... 60
Závěr ... 63
Seznam použité literatury ... 65
Zdroje obrázků ... 68
Seznam příloh ... 69

Úvod

Diplomová práce se zabývá deskovými hrami a hlavolamy a jejich využitím ve výuce informačních technologií, a zvláště pak při výuce algoritmizace a programování. Cíle práce jsou:

- zvážit možnosti využití deskových her a hlavolamů k prohloubení zájmu žáků o programování
- naprogramovat vlastní verze vybraných deskových her a hlavolamů
- naprogramované deskové hry a hlavolamy začlenit do výuky informatiky a zhodnotit jejich přínos ve věci motivace žáků

Práce je rozdělena na část teoretickou, která sestává z prvních čtyř kapitol, a na část praktickou, která sestává ze zbylých dvou kapitol.

První kapitola teoretické části se zabývá didaktickými otázkami z oblasti výuky informatiky a programování a rozvoje algoritmického myšlení v rámci českého školství. V kapitole je rovněž nahlédnuto do současné situace okolo plánované reformy ve výuce informatiky na základních školách.

Druhá kapitola slouží k představení dětského programovacího jazyku Scratch, který je skvělým nástrojem pro výuku programování vhodným pro různé věkové kategorie, a který byl zvolen pro tvorbu programů v rámci části praktické.

Třetí kapitola přechází k deskovým hrám a hlavolamům. Jejím cílem je podat přehled typů deskových her a hlavolamů, včetně stručného náhledu do jejich všeobecné historie, a kategorií, podle nichž lze deskové hry, resp. hlavolamy dělit.

Účelem čtvrté a tedy poslední kapitoly teoretické části je podrobněji představit několik vybraných deskových her a hlavolamů, které byly následně autorem naprogramovány v rámci části praktické. U těchto konkrétních deskových her a hlavolamů je rovněž nahlédnuto do jejich historie, a pak samozřejmě podrobný popis jejich pravidel.

Pátá kapitola, která je první kapitolou části praktické, obsahuje autorské zpracování vybraných deskových her a hlavolamů v dětském programovacím jazyku Scratch, a to včetně vlastní grafiky. Zpracovanými hrami a hlavolamy jsou Tic-tac-toe (tedy Piškvorky na herním plánu s rozměry 3×3 pole), Jezdcova procházka (na šachovnici s rozměry 5×5 polí) a Hanojské věže (se šesti disky). Každé hře či hlavolamu náleží jedna podkapitola, která obsahuje části kódu, popis

principu jeho fungování a screenshoty ze samotné aplikace. Tyto tři programy jsou k nahlédnutí v přílohách diplomové práce.

Šestá kapitola je shrnutím výsledků dotazníkového šetření na téma programování a jeho výuky a názorů na přínosy využití deskových her a hlavolamů k ilustraci algoritmických postupů. Toto šetření autor práce provedl v rámci výuky informatiky na ZŠ Sever v Hradci Králové. Detailní výsledky dotazníkového šetření jsou rovněž k nahlédnutí v přílohách práce.

1 Postavení programování v rámci českého školství

Tato kapitola se zabývá situací okolo výuky informatiky a programování v rámci českého vzdělávacího systému.

První podkapitola odkazuje na český rámcový vzdělávací program a na zastoupení programování nejen v hodinách informatiky. Druhá podkapitola je zaměřena na plánovanou reformu ve výuce informatiky. Třetí podkapitola je zhodnocením reálné situace v českém školství. Toto zhodnocení je podloženo autorovými zkušenostmi s výukou informatiky na ZŠ Sever v Hradci Králové. Poslední podkapitola se zabývá motivací žáků, a následně je uvedeno několik aplikací, jimiž motivaci lze podpořit.

1.1 Zastoupení programování v RVP

Jedním ze specifik současného školství je období změn v koncepci výuky informačních a komunikačních technologií (ICT). Dle současného českého Rámcového vzdělávacího programu pro základní vzdělávání (RVP ZV), který platí od září 2017, jsou ICT zařazeny jako povinná součást základního vzdělávání jak na 1., tak na 2. stupni. [1]

Avšak zastoupení programování a rozvíjení algoritmického myšlení v RVP je nyní téměř nulové. Konkrétně slovo „programování“ v celém vydání RVP pro ZŠ z roku 2017 není obsaženo ani jednou. Algoritmy jsou zmiňovány v rámci Matematiky, a poté je algoritmické myšlení jednou zmíněno v rámci Informačních a komunikačních technologií ve spojitosti se schopnostmi žáka formulovat své požadavky a interagovat s počítačem [1].

Z toho lze usuzovat, že výuka informatiky v České republice není příliš inovativní, a tak se v rámci projektu iMyšlení schyluje k velké změně v celkové koncepci výuky informatiky, která české školství čeká. [2]

1.2 Reforma ve výuce informatiky

Tato reforma ve výuce informatiky, jejímž autorem je doc. PaedDr. Jiří Vaníček, PhD. z Katedry informatiky Pedagogické fakulty Jihočeské univerzity, a s níž je úzce spojen projekt iMyšlení, byla na sedmdesáti vybraných školách spuštěna už ve

školním roce 2018/2019. [3] Termín spuštění v celostátním měřítku se pak plánuje na září 2021. [4]

Podle tohoto plánu pro reformaci výuky informatiky mají děti být seznamovány s programováním už od první třídy, a stejně tak by mělo být rozvíjeno jejich logické a algoritmické myšlení. Na úkor tohoto zaměření by však nemalým dílem byla oslabena náplň, na kterou jsme v současném českém školství v hodinách informatiky zvyklí. Tato vize výuky informatiky je totiž do značné míry založena na předpokladu, že žáci základních škol práci s počítačem na uživatelské úrovni zvládají už sami, a tudíž v rámci hodin informatiky výuka látky jako práce s průzkumníkem Windows, práce s kancelářskými programy jako MS Word, MS Excel či MS PowerPoint či práce s grafickými editory není nutná. Naštěstí by však práce s uživatelským softwarem z hodin neměla být zcela odstraněna, ale do jisté míry přesunuta do ostatních předmětů. [5]

Celkovou ideu této reformace poměrně výstižně shrnuje následující citace:

„Počítač je dnes v každé domácnosti i firmě a koncept učení se toho, jak se vůbec ovládá, už je tak hodně zastaralý. I ty nejmenší děti už umí pracovat s moderními technologiemi a mnohdy dokonce lépe, než jejich rodiče. Schopnost ovládání počítače je zkrátka už samozřejmostí a je nutné přistoupit k další fázi informatické gramotnosti, kdy i ty nejmenší děti už musí vědět, jak počítače, stroje či programy fungují a jak být tím, kdo je tvoří.“ [2]

Ačkoli jej autor článku, z něhož úryvek výše pochází, pravděpodobně psal se zdravou mírou nadsázky, je nasnadě podotknout, že realita se od situace popsané v úryvku poněkud liší. A nejedná se pouze o samotné ovládání počítače, ale i o kybernetickou gramotnost, se kterou česká společnost není na příliš dobré úrovni – podle některých odhadů oproti jiným zemím zaostává i o deset let. [6]

1.3 Autorovy zkušenosti s výukou informatiky na ZŠ

Autor práce může z vlastní zkušenosti s roční praxí na ZŠ Sever v Hradci Králové potvrdit, že na základních školách v České republice jsou stále žáci, kteří doma počítač nemají (a nutno podotknout, že se nemusí jednat o jakkoliv sociálně vyloučenou lokalitu). Takoví žáci se musí naučit pracovat s počítačem právě ve škole. Samozřejmě, drtivá většina žáků stabilní přístup k počítači ve své domácnosti naštěstí má. Ale mnozí z nich valnými dovednostmi nedisponují –

naopak, mnozí z nich počítač doma využívají téměř výlučně k hraní her a sledování videí na YouTube. Není tedy výjimkou, že žáci často mají značné problémy jak s prací s kancelářským a grafickým softwarem, která představuje podstatnou část současné náplně výuky informatiky na základních školách, tak s naprosto základními úkony v průzkumníku souborů jako vytváření, kopírování, přesouvání či přejmenovávání souborů a složek. Často také nejsou schopni se řádně orientovat v souborovém systému a mnohdy neví, kam soubory, se kterými v uživatelských aplikacích pracují, vlastně ukládají. Je třeba zmínit, že tato skutečnost se týká i žáků na 2. stupni základní školy.

Tímto autor rozhodně nechce naznačovat, že by se programování ve školách nemělo vyučovat. Naopak, mít základní vhled do problematiky programování a algoritmizace je v dnešní době velmi žádoucí a pomáhá s celkovým rozvojem logického a technického uvažování. Autor se však domnívá, že někteří žáci na základních školách zkrátka se svými dispozicemi na povinnou výuku programování nestačí. Avšak je třeba si uvědomit, že takoví žáci obvykle mají značné problémy i v jiných hodinách, zvláště pak v ostatních předmětech, které jsou více založené na logickém myšlení, tedy hlavně v matematice a ve fyzice. Tento způsob myšlení je však do jisté míry důležitý i v jazycích, a to nejen v záležitostech mluvnice, tedy například ve větné skladbě či ve větném rozboru, ale i v oblasti slohu, který rovněž vyžaduje určitou strukturu a systematické postupování.

1.4 Motivace pro programování

Bohužel, žáci mající předpoklady pro zvládnutí látky programování často nemají příliš vysokou míru motivace pro učení se této látce (ale to koneckonců obvykle platí i o jiných vyučovacích předmětech). Tuto motivaci mohou povzbudit činnosti, které žáky baví, a u nichž mohou vidět okamžitý efekt. Z toho důvodu pro prostředí základní školy využití tradičních programovacích jazyků není příliš vhodnou variantou, a to jak pro svou abstraktnost, tak pro úsilí, které je potřeba vynaložit před možností spatřit první výsledky. Ovšem programování může nabídnout i různé způsoby, které jsou vhodnější pro motivování mladších žáků. Jako velmi kvalitní metoda se nabízí použití vzdělávacích online aplikací.

Jako příklad takové aplikace můžeme jmenovat například *Umíme programovat*, která je dostupná na adrese www.umimeprogramovat.cz. V této aplikaci se žáci mimo jiné mohou setkat s úlohami, které musí vyřešit pomocí utváření kódu. Kromě úloh, kde se kód píše pomocí programovacího jazyku Python, zde jsou i úlohy, které se řeší za použití přetahování bloků, což je alternativa ke kódu v podobě textu, kterou ocení hlavně mladší žáci. [7]

Další možností může být *Hour of Code* (hourofcode.com), kde se žáci setkávají se zadáními, které jsou typicky koncipovány jako hry či úkoly s postupně rostoucí obtížností. Ty opět typicky plní za pomoci přetahování bloků, čímž vytváří program. [8]

Zmíněné použití bloků je charakteristickým znakem mnohých dětských programovacích jazyků, které už nejsou určeny pouze k vyřešení předem dané úlohy, ale které lze využít k programování regulérních aplikací. Jedním takovým programovacím jazykem je *Scratch*, kterým se bude zabývat následující kapitola.

Kromě vhodné platformy pro výuku programování je k motivování žáků také nezbytné najít vhodná témata, která jsou dětem dobře známá a blízká. Při volbě tématu pro programování je nezbytné brát ohled na věk, znalosti a zájmy žáků, avšak jsou i témata, která jsou víceméně univerzální napříč věkovými skupinami. Jedním z takových témat jsou právě deskové hry a s nimi související hlavolamy, pro jejichž tvorbu je Scratch nepochybně vhodným nástrojem.

2 Scratch

Scratch je dětský programovací jazyk vhodný zejména pro tvorbu interaktivních příběhů, her a animací. Aplikace je projektem Lifelong Kindergarten Group v MIT Media Lab. Scratch je volně dostupný a je zcela zdarma. [9]

Scratch je aplikace fungující online, k jejímuž spuštění postačí jeden z podporovaný internetových prohlížečů s aktuální verzí Adobe Flash Player. Scratch je dostupný na adrese https://scratch.mit.edu. Existuje však i oficiální aplikace Scratch 3 Offline Editor, což je, jak už název aplikace napovídá, editor umožňující uživateli pracovat i offline. Offline aplikace je pochopitelně efektivnější než aplikace v prohlížeči, a to i z toho důvodu, že aplikace není 100% kompatibilní s některými prohlížeči, a to i tehdy, když má uživatel aktuální verzi Adobe Flash Playeru.[1] Scratch 3 Offline Editor je ke stažení zdarma na adrese https://scratch.mit.edu/download.

Na webu aplikace Scratch se uvádí, že aplikace je určena primárně pro děti ve věku od 8 do 16 let, ale mohou ji využívat všechny věkové kategorie. [9].

2.1 Hlavní výhody a nevýhody aplikace Scratch

Za výhody aplikace lze považovat např. to, že uživatel nemusí instalovat žádný software potřebný k jejímu chodu a vystačí si s internetovým prohlížečem (tedy za předpokladu, že má aktuální verzi Adobe Flash Playeru). Další výhodou je jednoduchost aplikace a její relativně intuitivní ovládání, které je vhodné nejen pro běžné uživatele, kteří nemají s programováním zkušenosti, ale samozřejmě i pro děti, na jejichž skupinu je Scratch cílen. Velkou předností aplikace je také to, že uživatel má neustále náhled na plochu svého programu a změny, které uživatel v programu udělá, se v tomto náhledu ihned projeví. Program lze snadno a rychle spouštět, aniž by jej bylo potřeba kompilovat.

[1] Například z autorovy vlastní zkušenosti bylo při použití prohlížeče Mozilla Firefox verze 60.0.2 (64 bitů) běžným bugem, že po umístění kurzoru do textového pole nebylo možné psát. Aby psaní bylo možné, bylo nutno otevřít v prohlížeči jiný panel, přepnout do něj, a následně přepnout zpět do panelu se Scratchem. Tento bug se však vyskytoval téměř při každém umístění kurzoru do textového pole, což pochopitelně způsobovalo značné potíže při práci, a použití online aplikace v tomto prohlížeči tedy bylo značně neefektivní.

Další obrovskou výhodou aplikace je samozřejmě její didaktický potenciál. Scratch je skvělý způsob, jak nenásilnou formou zprostředkovat první setkání dětí s programováním a jak formou blízkou hře rozvíjet jejich algoritmické myšlení. Kromě brány do světa programování může Scratch posloužit také k probuzení zájmu o matematiku, kterou děti obvykle nepovažují za příliš zajímavou či zábavnou. Vzhledem k tomu, že stejně jako každé programování je i Scratch z velké části prací s proměnnými a jejich porovnáváním a s výrokovou logikou, děti dostávají skvělou příležitost k tomu, aby v „nesmyslných písmenech", s nimiž se ve vyšších ročnících základní školy v hodinách matematiky setkávají, viděly hlubší význam, a celkově dospěly k pochopení důležitosti matematiky.

Aplikace ale samozřejmě má i nevýhody. Za ty lze považovat například neohrabanost přetahování bloků, se kterým se v porovnání s kódem ve formě textu pokročilým uživatelům pracuje mnohem hůře a pomaleji – avšak na druhou stranu pro děti a začátečníky je práce s bloky snazší a intuitivnější. V blocích, které aplikace používá, vzhledem k jejich netextové povaze nelze efektivně vyhledávat, a s tím souvisí i absence „najít a nahradit", což dokáže být obrovským problémem, protože v případě potřeby rozsáhlého vyhledávání a nahrazování musí uživatel vše provést manuálně. Aplikace bohužel nepodporuje ani základní univerzální klávesové zkratky jako například Ctrl+C, Ctrl+X, Ctrl+V, ale ani Ctrl+Z a Ctrl+Y, přičemž funkce „zpět" a „opakovat" ani aplikace neobsahuje. Další nevýhodou je značně neohrabaná manipulace s poli proměnných, a z toho důvodu se zpravidla jeví jako efektivnější používat jednotlivé proměnné, a to i ve větším množství. Zcela očividnou nevýhodu je také neměnná velikost scény programu, která má pevně dané rozměry, a to 480×360 pixelů. Všeobecnou a celkem relevantní nevýhodou pak samozřejmě je nevhodnost Scratche pro tvorbu větších projektů. Scratchi také schází oficiální kompilátor, takže bez použití aplikací třetí strany[2] není možné program vytvořený ve Scratchi, který je ve formátu sb2 (pro Scratch 2) nebo sb3 (pro Scratch 3), zkompilovat do EXE souboru, a uživatel je tedy odkázán na spouštění aplikací prostřednictvím online či offline prostředí aplikace Scratch.

[2] Nutno podotknout, že zmíněné aplikace třetí strany mají hodně daleko k dokonalosti, a uživateli se spíše než zkompilovat Scratch program podaří stáhnout si nějaký malware.

2.2 Scratch – používané termíny

Blok (Block) – Podobně jako jiné platformy pro výuku programování pro děti, Scratch nepracuje s kódem v podobě textu, ale v podobě bloků – bloky jsou tedy základním prostředkem pro tvorbu kódu programů v tomto jazyce. Obecně platí, že jeden blok odpovídá jednomu řádku kódu (resp. jednomu příkazu). Některé bloky obsahují parametry v podobě textových polí, do nichž lze vepsat hodnotu, nebo vložit blok operátoru. Scratch obsahuje operátory pro základní matematické operace, generování náhodného čísla, porovnávání a základní logické funkce: *and*, *or* a *not*. Scratch samozřejmě disponuje i bloky pro větvení (jak částečné, tak úplné) a pro cykly (podobně jako jiné programovací jazyky, i Scratch umožňuje použití typických cyklů: nekonečný cyklus, cyklus *for*, cyklus *while-do* a cyklus *do-while*). Scratch rovněž umožňuje z posloupností bloků utvářet i bloky vlastní.

Scéna (Stage) – Scéna u programů ve Scratchi tvoří pozadí uživatelského prostředí programu. Sama scéna může obsahovat svůj kód. Scéně pak náleží grafické soubory nazývané pozadí. Výchozím pozadím je bílá plocha. Pozadí může být i více a je možné je během běhu programu měnit. Rozměry scény ve Scratch programech jsou pevně dané, a to na 480×360 pixelů.

Pozadí (Backdrop) – Grafické soubory používané scénou udávající její vzhled. Pro scénu může být použito více pozadí, která se mohou měnit. Pozadí v podstatě funguje jako kostýmy pro scénu. Vhodné rozměry pozadí jsou 480×360 pixelů, jelikož to jsou pevně dané rozměry scény. Rozměry však mohou být i jiné, ale tehdy dojde k automatickému přizpůsobení.

Postava (Sprite)[3] – Interaktivní objekty, které ve Scratchi tvoří hlavní část programu. Téměř vše, co uživatel na scéně programu může vidět, jsou sprity. Sprity je možno pohybovat, měnit jejich kostýmy, měnit jejich rozměry apod. Důležitou vlastností spritů je také schopnost spouštět skripty po kliknutí na ně. Převážná většina kódu ve Scratch programech náleží právě spritům.

[3] Ačkoli česká lokalizace Scratche užívá termín „postava“, v této práci je používán termín „sprite“, jelikož pod pojmem „postava“ si většina lidí představí postavu jako je např. postava hráče či figury. To však může být poněkud zavádějící, protože sprite ve Scratch programech (a v počítačové grafice všeobecně) reprezentuje jakýkoliv grafický objekt (např. jednotlivá hrací pole, jednotlivé disky v hanojských věžích apod.), přičemž takový objekt ani nemusí být viditelný.

Kostým (Costume) – Grafické soubory používané sprity udávající jejich vzhled. Jak už bylo naznačeno výše, kostýmy a sprity mezi sebou mají v podstatě stejný vztah jako právě pozadí a scéna. Kostýmy lze v kódu dané postavy měnit nebo i přímo kódem upravovat – např. měnit jejich barvu, způsobit efekty jako rozostření, rozpixelování apod. Tyto efekty však jsou pouze dočasné a lze je pomocí jednoho příkazu zrušit a obnovit vzhled spritu do původní podoby daného kostýmu.

Skript (Script) – Kus kódu, k jehož spouštění dojde při nějaké události. Typickými spouštěči skriptů je spuštění programu (kliknutí na zelený praporek), kliknutí na sprite, stisknutí klávesy či obdržení zprávy.

Zpráva (Message) – Způsob, pomocí nějž mezi sebou jednotlivé komponenty programu mohou komunikovat. Každá zpráva má nějaké jméno a je vysílána spritem nebo scénou, přičemž toto rozeslání zprávy je pro celý program. Poté by v programu měl být alespoň jeden sprite obsahující skript, který se spustí právě po obdržení zprávy s daným jménem. K tomuto procesu slouží bloky *vyšli zprávu <jméno> (broadcast message <name>)* a po *obdržení zprávy <jméno> (when I receive <name>)*. Nejčastějším použitím zpráv je právě rozeslání zprávy jedním spritem a obdržení zprávy spritem jiným, následkem čehož dojde ke spuštění nějakého skriptu, avšak takový skript může mít i přímo ten sprite, který zprávu rozeslal – čímž tedy vlastně zprávu zasílá sám sobě.

3 Deskové hry a hlavolamy

V této kapitole budou poskytnuty obecné informace o deskových hrách a hlavolamech.

Kapitola je členěna do dvou podkapitol – první podkapitola pro deskové hry a druhá pro hlavolamy. Každá z těchto podkapitol je dále členěna na sekci zabývající se historií a na sekci nabízející přehled typů.

Na úvod kapitoly je rovněž vhodné předem upozornit, že jejím účelem není rozbor deskových her a hlavolamů z hlediska jejich didaktického využití. Jedná se o přehled historie a kategorií.

3.1 Deskové hry

Ačkoli se to na první pohled nemusí zdát, deskové hry se v současném světě těší velké oblibě, která každým rokem roste. Stejně tak každým rokem vychází mnohé nové deskové hry či rozšíření k deskovým hrám již existujícím. Pochopitelně se nejedná o klasické deskové hry jako šachy, dáma nebo mlýn, ale o tzv. „moderní deskové hry“. Moderní deskové hry jsou komerčně zaměřené, obvykle nejsou abstraktní, a tudíž mají konkrétní tématiku, a také zpravidla mají mnohem komplexnější pravidla než deskové hry klasické. Tyto vlastnosti však rozhodně nemusí zaručovat jejich kvalitu.

Pod pojmem „deskové hry“ se v dnešní době často rozumí právě moderní deskové hry, kterými se však v této práci příliš zabývat nebudeme – ovšem zmínit se o nich je nasnadě a v **sekci 3.1.2** bude stručně popsáno několik kategorií z oblasti moderních deskových her. Pro potřeby této práce se budeme zabývat hlavně deskovými hrami klasickými, resp. některými jejich variantami.

3.1.1 Historie deskových her

Deskové hry jsou fenoménem, který se vyvíjí již po dobu tisíců let. Nejstaršími doklady o existenci deskových her jsou artefakty nalezené v Sumeru. Jednalo se o ornamentální skříňky obsahující hrací kameny a herní kostky pyramidového tvaru. O těchto hrách se toho však ví poměrně málo. [10]

Nejstarší deskovou hrou, jejíž jméno známe, je Senet (nebo také Sen't či Senat), který se hrál ve starověkém Egyptě. Nejstaršími hmatatelnými objekty týkajícími

se Senetu jsou úlomky desky, která sloužila jako herní plán. Stáří těchto úlomků se odhaduje až k roku 3100 př. n. l. Senet, resp. osoby, které jej hrály, byly také vyobrazovány na nástěnných malbách v hrobkách. [11]

Senet se hrál na desce obsahující 30 čtvercových polí uspořádaných do třech řad po deseti. Dále se používaly dvě sady figurek či kamenů, přičemž každá sada jich obsahovala minimálně pět. Ačkoli přesná pravidla původního Senetu nejsou známa a jsou pouze předmětem dohadů, historici zabývající se Senetem Timothy Kendall a Robert Charles Bell provedli rekonstrukci jeho možných pravidel na základě úryvků z dochovaných písemných pramenů. [11]

Další velice starou hrou jsou vrhcáby. Ty pochází z Persie a jejich stáří je odhadováno na přibližně 5 000 let. Vrhcáby jsou strategickou deskovou hrou, v níž hraje svou roli i faktor náhody – používají se zde klasické šestistěnné hrací kostky. [12]

Hrou, jejíž historii lze sledovat až do starověkého Egypta, je mlýn. Dokladem je několik desek ze střešní krytiny z chrámu z thébské Kúrny, do nichž byly vyryty herní plány mlýnu. Mlýn se v průběhu věků také značně rozšířil, což dokazují zprávy o jeho existenci pocházející z antického Řecka a Říma. Mlýn se hrál také v Tróji v období trojských válek. [10] V současnosti existuje mnoho variant mlýnu lišících se počtem kamenů a velikostí a tvarem herního plánu. U nás je známá hlavně varianta s devíti kameny a plánem se čtyřiadvaceti poli, ale existuje i varianta se šesti kameny a s menším plánem nebo varianta se třemi kameny a plánem s pouhými devíti poli známá jako mlýnek. Od počtu kamenů jsou odvozeny anglické názvy pro varianty mlýnu: nine men's morris, six men's morris nebo three men's morris.

Velice významnou hrou je také Go, které pochází z Číny. Jeho stáří se odhaduje na 2 500 let, avšak hraje se dodnes, a to na celém světě. Go je velice kompetitivní a hraje se i na velice prestižní profesionální úrovni. [13]

Pokud by měla být jmenována nejznámější a nejvýznamnější desková hra s největší tradicí a prestiží v historii, na prvním místě by nepochybně byly šachy. Šachy mají svůj původ v Indii, kde se objevily mezi lety 280 a 550 n. l. Tehdejší hra se od své dnešní varianty pochopitelně poněkud lišila a musela projít jak vývojem, tak postupným šířením do ostatních kultur. Evoluce do současné podoby šachů probíhala v oblastech severní Afriky, Sicílie a Španělska, kam byla přinesena Araby

v 10. století. [14] Podobu moderních šachů tak, jak je známe dnes, měla tato hra už v roce 1475, přičemž poslední změny probíhaly v jižní Evropě. [15] V průběhu věků vzniklo obrovské množství variant šachů, přičemž některé z nich se v menší či větší míře těší oblibě i dnes. Tyto varianty jsou souhrnně nazývány exotické šachy, a konkrétními příklady mohou být např. jezdec proti králi, šachy bez matu, šachy pro více hráčů apod. [10] Z šachů také vychází různé šachové úlohy nebo jezdcova procházka, které jsou již na pomezí s hlavolamy.

Druhou nejznámější deskovou hrou užívající šachovnici je dáma. Uvádí se, že dáma byla vymyšlena někde ve 12. století na jihu Francie. Za předchůdce dámy se považuje hra Alquerque pocházející ze Středního východu, která v té době byla populární ve Španělsku. Podobně jako šachy, i dáma má mnoho různých variant. Z těch známějších lze jmenovat např. českou dámu, anglickou dámu (v britské angličtině označovanou jako *draughts* a v americké jako *checkers*), žravou dámu apod. [10]

Fenoménem, který se v dnešní době těší velké popularitě, jsou také moderní deskové hry. Ty se obvykle dělí na deskové hry evropského či německého typu (někdy také nazývané eurohry) a deskové hry amerického typu. Typickým znakem moderních deskových her je o poznání nižší míra abstrakce – tyto hry typicky mají nějaký příběh, figurky či kameny představují konkrétní postavy, součástí her často bývají různé kartičky představující specifické události apod. Navzdory menšímu poklesu v prodejích, který deskové hry zažívaly koncem 90. let a na přelomu tisíciletí, v 21. století jsou jejich prodeje a popularita na vzestupu [16], a to do takové míry, že se hovoří o tzv. „zlaté době deskových her“. [17]

3.1.2 Typy deskových her

Deskové hry lze rozlišovat podle několika kategorií, které se vzájemně nevylučují, avšak některé jejich kombinace jsou častější než jiné. Některých z těchto kategorií jsme se již dotknuli v úvodu této kapitoly, a v této sekci tyto pojmy budou rozvedeny o něco podrobněji. U každé z nich bude uvedeno několik charakteristických příkladů her. Dále je vhodné zmínit, že zde uvedené kategorie rozhodně nejsou kompletním seznamem, a ačkoli se tato označení deskových her běžně používají, jejich názvy nemusí být univerzálně platné.

Rovněž zde je uvedeno několik kategorií, které se ne vždy řadí mezi deskové hry, resp. jedná se o deskové hry pouze v širším slova smyslu.

Následující výčet kategorií čerpá z *Velké knihy deskových her* [10] a ze sekce *Categories* v článku *Board game* z anglické Wikipedie [16].

- **Klasické**
  - Starší hry, u nichž obvykle není znám autor, ani přesný rok vzniku
  - Tyto hry jsou zpravidla abstraktní, nemají tedy konkrétní tématiku nebo děj
  - Příslušenství k jejich hraní není specifické a lze jej použít i k hraní jiných klasických her – např. s klasickou šachovnici lze hrát šachy, dámu, jezdcovu procházku apod., s klasickými kameny lze hrát dámu, mlýn, vrhcáby apod.
  - Např. šachy, dáma, mlýn, Člověče, nezlob se!, piškvorky, go, …
- **Moderní**
  - Novější hry, u nichž známe autora a rok vzniku
  - Tyto hry mají zpravidla konkrétní tématiku a specifický děj
  - Jsou komerční
  - K jejich hraní je potřeba specifické vybavení jako herní plán, různé kartičky, figurky apod., avšak u některých z nich se používají i klasické figurky nebo hrací kostky (např. Monopoly)
  - Např. Monopoly, Dostihy a sázky, Osadníci z Katanu, Carcassone, Risk, Hra o trůny, …
- **Závodivé**
  - Hry, u nichž je cílem dostat svou figurku či figurky do předem určeného cíle dříve než soupeři
  - Rychlost figurek je obvykle určována hody kostkou
  - Vysoká míra náhody
  - Např. Člověče , nezlob se!, Pacheesi, Z pohádky do pohádky, ...
- **Strategické**
  - Hry, kde je cílem pomocí důmyslné strategie sebrat či zajmout soupeřovy figurky či kameny, obsadit konkrétní pozice či pole apod.
  - Nízká nebo žádná míra náhody
  - Často se jedná o hry s dokonalými informacemi

    - Jsou vhodné pro kompetitivní hraní
    - Např. šachy, dáma, mlýn, piškvorky, go, ...
- **Poziční**
    - Hry, kde je cílem pomocí důmyslné strategie obsadit konkrétní pozice či pole, utvořit řady kamenů apod.
    - Řada strategických deskových her se často považuje současně i za poziční
    - Např. mlýn, piškvorky, go, ...
- **Slovní**
    - Hry, kde je cílem z nějaké dostupné sady písmen skládat smysluplná slova či věty
    - Např. Scrabble, anagramy, křížovky, …
- **Vědomostní**
    - Hry, kde hrají velkou roli vědomosti, obvykle triviální
    - Mohou to být hry s tužkou a papírem, ale také hry s herním plánem a různým příslušenstvím jako jsou kartičky apod.
    - Např. jméno město zvíře věc, AZ kvíz, Souboj mozků, ...
- **Deskové hry evropského/německého typu (běžně označované jako Eurohry)**
    - Mají konkrétní tématiku, avšak udržují si nějakou míru abstrakce
    - Herní plány bývají variabilní, čehož je dosaženo postupným umísťováním „dlaždic“
    - Hráči mezi sebou obvykle mohou vyjednávat či obchodovat s různými zdroji
    - Hráči se obvykle nijak nevyřazují
    - Nízký nebo žádný faktor náhody
    - Např. Carcassone, Osadníci z Katanu, Puerto Rico, …
- **Deskové hry amerického typu**
    - Mají konkrétní tématiku a příběh
    - Dochází k přímému konfliktu mezi hráči
    - Obvykle dlouhá herní doba (běžně překračuje 4 hodiny)
    - Vyšší faktor náhody
    - Např. Arkham Horror, Twilight Imperium, Pandemic, …

- **Hádanky a hlavolamy**
    - Hry, kde není potřeba žádný herní plán či figurky (avšak může být použit), a stačí pouze tužka a papír
    - Obvykle určené pro jednoho hráče
    - Např. jezdcova procházka, kreslení tvarů jedním tahem, ...
- **Hry s tužkou a papírem**
    - Hry, kde není potřeba žádný herní plán či figurky, a stačí pouze tužka a papír
    - Mohou to být hry pro dva či více hráčů, ale také hry pro jednoho
    - Např. piškvorky, lodě, jméno město zvíře věc, ...
- **Kooperativní**
    - Hry, kde hráči nesoupeří proti sobě, ale spolupracují a snaží se porazit hru, která je ovládána obvykle faktorem náhody – např. hody kostkou, tažením kartiček z balíčku, apod.
    - Např. Pandemic, …
- **S faktorem náhody**
    - Hry, kde faktor náhody hraje nějakou roli, přičemž u některých her (např. Člověče, nezlob se!) je tato role naprosto zásadní a hráči mají minimální možnosti ovlivnění výsledku
    - Náhoda je obvykle určována např. hodem kostkou (nejčastější jsou kostky šestistěnné s hodnotami 1 až 6, avšak existují i hry, u nichž se používají i kostky s jinými počty stěn – např. čtyřstěnné, osmistěnné, desetistěnné, dvanáctistěnné, dvacetistěnné apod.; kostky mohou rovněž namísto číselných hodnot obsahovat symboly specificky pro danou hru, a některé jejich stěny nemusí obsahovat nic), tažením kartiček z náhodně zamíchaného balíčku apod.
    - Např. Člověče, nezlob se!, Monopoly, Dostihy a sázky, …
- **Bez faktoru náhody**
    - Hry, kde faktor náhody nehraje žádnou roli, takže záleží pouze na dovednostech hráčů
    - Např. šachy, dáma, piškvorky, …

- **S dokonalými informacemi**
    - Hry, kde všichni hráči v jakýkoliv daný okamžik mají přístup ke všem informacím o stavu hry
    - Např. Člověče, nezlob se!, šachy, dáma, piškvorky, …
- **S nedokonalými informacemi**
    - Hry, kde neplatí, že všichni hráči v jakýkoliv daný okamžik mají přístup ke všem informacím o stavu hry
    - Skryté informace mohou být např. karty v soupeřově ruce, pořadí karet v balíčku či balíčcích, tajně přidělené role hráčů, tajná rozhodnutí hráčů apod.
    - Např. Monopoly, Dostihy a sázky, většina karetních her, …

3.2 Hlavolamy

Jelikož se tato práce kromě deskových her zabývá i hlavolamy, je i jim věnována vlastní podkapitola. Hlavolamy lze opět dělit do několika kategorií, avšak v této práci se budeme zabývat hlavně hlavolamy tahovými. V **sekci 3.2.2** však budou uvedeny i jiné typy hlavolamů.

3.2.1 Historie hlavolamů

Podobně jako deskové hry, i hlavolamy jsou velice starým fenoménem sahajícím do 3. tisíciletí př. n. l.

Jedním z nejstarších známých typů hlavolamů jsou bludiště či labyrinty, které se těšily velké oblibě ve starověkém Egyptě a starověkém Řecku. Některá z nich jsou datována až do roku 2300 př. n. l. Taková bludiště a labyrinty často měly náboženský či spirituální význam. [18] Rozdíl mezi bludišti a labyrinty spočívá v počtu cest, kterými se lze vydat – labyrint má pouze jednu možnou cestu, která se nikde nerozdvojuje (z čehož vyplývá, že se v něm při dodržování pravidla pravé nebo levé ruky v podstatě nedá ztratit), zatímco bludiště mají cest více, cesty se mohou libovolně větvit, mohou se zde utvářet okruhy atd.[4] [19] Tato bludiště či

[4] Pokud bychom měli rozlišit bludiště a labyrinty z hlediska teorie grafů, bludiště by bylo představováno komplexním grafem sestávajícím z mnoha vrcholů, kde každý z nich představuje

labyrinty mohly mít jak kreslenou podobu, která tedy představuje zmenšený plánek, tak podobu fyzickou v životní velikosti, která je určená přímo k procházení. Taková fyzická bludiště a labyrinty jsou typicky pozemní útvary vytvořené vystoupnutou podlahou nebo dvěma odlišnými barvami dlažby. V takovém případě je možné z jakéhokoliv místa v bludišti či labyrintu vidět celou jeho plochu. Další podobou může být uzavřené bludiště či labyrint se stěnami obvykle tvořenými regulérními zdmi nebo živým plotem, avšak moderní bludiště a labyrinty mohou být tvořené i ze skla nebo zrcadel.

Velmi starým typem hlavolamu jsou hlavolamy skládací. Za nejstarší známou skládačku je považován Stomachion, který vymyslel řecký matematik Archimedes. Tento hlavolam tedy vznikl někde ve 3. století př. n. l. Jedná se o obrazec složený ze čtrnácti dílků různých tvarů a velikostí – jedenáct dílků má tvar trojúhelníků, dva čtyřúhelníků a poslední dílek má tvar pětiúhelníku. [20]

Jak už bylo zmíněno v předchozí podkapitole, existují variace na hru v šachy, které jsou na pomezí s hlavolamy. Klasickým příkladem je jezdcova procházka, která byla popsána už kolem roku 840 n. l. arabským učencem Al-Ádlím v *díle Kitáb aš-Šatrandž*, tedy *Knize o šachu*. [21] Jezdcově procházce se podrobněji bude věnovat **podkapitola 4.2**.

Ačkoli se obvykle nepovažují přímo za hlavolam, Jigsaw puzzle (u nás běžně označované jen jako „puzzle“) jsou i v dnešní době velmi populární skládačkou. První Jigsaw puzzle vytvořil v roce 1767 anglický kartograf John Spilsbury. Původně se jednalo o skládačky map, které sloužily hlavně ke vzdělávacím účelům, avšak v průběhu času se začaly objevovat i různé jiné motivy. Kromě znázorněných obrazců se změnil i materiál, z něhož se Jigsaw puzzle vyrábí – původní skládačky byly vyráběny ze dřeva, avšak v dnešní době se obvykle používá tvrdý karton nebo umělá hmota. [22]

Poměrně fascinujícím pojmem jsou polymina (někdy také polyomina). Ačkoli varianty polymin jsou známy již delší dobu, název polymino vymyslel až v roce 1953 americký matematik Solomon Golomb, který se studiem polymin zabývá takřka celý svůj život. O polyminech rovněž napsal řadu knih. [23] Polymina jsou

---

rozdvojení, a hran, přičemž se v grafu i mohou vyskytovat kružnice. Na druhou stranu labyrint by byl graf tvořený pouze dvěma vrcholy spojenými jednou hranou.

hranaté útvary složené z určitého počtu stejně velkých čtverců. Nejjednodušším polyminem je domino, které je složené ze dvou čtverců. Při připojování dalších čtverců vznikají složitější útvary: tromino, tetromino, pentomino atd. Typickou formou hlavolamu s polyminy je skládání nebo vyplňování, které může probíhat jak ve dvourozměrném, tak trojrozměrném prostoru. Polymina jsou však velmi významná i ve videoherním průmyslu: nejproslulejším příkladem je Tetris. Kromě Tetrisu, kde jsou tetromina (tedy polymina složená ze čtyř čtverců) ústředním motivem, bývají různé hádanky s polyminy ve videohrách obsaženy jako minihry.

Za jeden z nejznámějších a nejproslulejších hlavolamů lze považovat Rubikovu kostku. Tu vymyslel a v roce 1975 si nechal patentovat maďarský architekt a designer Ernő Rubik. Rubikova kostka dodnes zůstává pomyslnou celebritou mezi hlavolamy a dodnes existují nadšenci, kteří soutěží v různých variacích řešení tohoto hlavolamu. [24]

3.2.2 Typy hlavolamů

Hlavolamy taktéž lze rozlišovat podle několika kategorií. Tyto kategorie se opět nemusí vzájemně vylučovat, avšak tato kategorizace je v případě grafů konkrétnější než v případě deskových her.

Následující výčet kategorií čerpá z knihy *Jak vyrobit a vyřešit hlavolamy* [23] a ze sekce *Genres* v článku *Puzzle* z anglické Wikipedie [25].

- **Tahové (s postupnými kroky)**
    - Hlavolamy, kde od sebe lze jednoznačně odlišit jednotlivé tahy
    - Počet tahů vedoucích k vyřešení hlavolamu musí být konečný
    - U většiny hlavolamů je kromě jejich úspěšného vyřešení cílem i minimalizovat počet tahů
    - Např. Rubikova kostka, Hanojské věže, jezdcova procházka, Patnáctka Sama Loyda, bludiště a labyrinty, …
- **Bludiště a labyrinty**
    - Cílem je najít cestu do středu a ven, z jednoho konce na druhý apod.
    - Bludiště má více cest, cesty se mohou libovolně větvit, mohou se zde utvářet okruhy
    - Labyrint má pouze jednu možnou cestu, která se nikde nerozdvojuje

- **Mechanické**
    - Hlavolamy, které jsou obvykle tvořeny částmi, kterými lze pohybovat nebo je otáčet, případně je vysouvat a zasouvat
    - Řadu následujících kategorií (posuvné, skládací, rozkládací, rozplétací, na manuální zručnost) lze považovat za podtypy mechanických hlavolamů
    - Např. Rubikova kostka, skládačky, tajné skříňky, …
- **Posuvné**
    - Hlavolamy, které mají obvykle podobu desky obsahující jednotlivé čtverce nebo jiné geometrické útvary, kterými lze posunovat, a nějakou prázdnou část
    - Cílem těchto hlavolamů je nějakým způsobem posuvné součásti uspořádat (např. složit obrázek, vzestupně seřadit čísla, utvořit slova z písmen apod.)
    - Např. posuvné obrázky, slovní skládačky, Patnáctka Sama Loyda, …
- **Skládací**
    - Hlavolamy, kde je cílem z určitého množství dílů složit obrazec či trojrozměrný objekt
    - Mohou být dvourozměrné či trojrozměrné
    - Těmito díly je obvykle možné rotovat
    - Díly do sebe obvykle nějakým způsobem zapadají – např. výstupky a výklenky
    - Např. Jigsaw puzzle (u nás obvykle označované jen jako „puzzle“), polymina, tangramy, …
- **Rozkládací**
    - Hlavolamy, kde je cílem rozložit či případně otevřít nějaký objekt
    - Součásti takového hlavolamu jsou do sebe obvykle nějak důmyslně zaklíněny
    - Např. japonské tajné skříňky, ježek v kleci, …
- **Rozplétací**
    - Hlavolamy, jejichž podstata se podobá hlavolamům rozkládacím, ale typicky obsahují menší počet části, které bývají ohebné
    - Např. ježek v kleci, rozplétání drátů/provázků, Orbits, …

- **Na manuální zručnost**
    - Hlavolamy, které k vyřešení kromě mentálního zapojení vyžadují i určitou dávku manuální zručnosti či obratnosti
    - Např. bludiště s kuličkou, …
- **Šachové problémy**
    - Hlavolamy, které se řeší na šachovnici, obvykle za použití klasických možností pohybu šachových figur
    - Obvykle se jedná o šachové úlohy, kde je nejčastějším cílem dát z určitého rozestavení figur během konkrétního počtu tahů mat, ale je možné sem řadit i např. jezdcovu procházku
    - Tyto hlavolamy jsou z pochopitelných důvodů také tahové
- **S tužkou a papírem**
    - Hlavolamy, kde stačí tužka a papír (avšak tento papír obvykle již obsahuje nějaký specifický plán)
    - Obvykle se rovněž jedná o tahové hlavolamy
    - Např. sudoku, bludiště a labyrinty, jezdcova procházka, …
- **Neřešitelné**
    - Hlavolamy, které, jak už název jejich kategorie napovídá, nelze vyřešit
    - Řešením je v podstatě uvědomit si, že žádné řešení neexistuje, popř. dokázat, proč tomu tak je
    - Např. Sedm mostů města Královce, pět pokojů, tři nádoby, …

4 Konkrétní příklady deskových her a hlavolamů

V této kapitole bude uvedena trojice deskových her a hlavolamů, přičemž u každého z těchto tří zástupců je nejprve uveden krátký náhled do jeho historie, a následně podrobná pravidla a případný popis řešení či strategie. Deskové hry a hlavolamy uvedené v této kapitole byly v rámci praktické části práce vytvořeny v prostředí Scratch.

Výběr konkrétních deskových her a hlavolamů autor práce prováděl primárně na základě svého osobního vztahu k daným deskovým hrám a hlavolamům, avšak bylo nutné zvážit i to, zda pro jejich vytvoření bude programovací jazyk Scratch vhodnou platformou. Také bylo zváženo, zda dané deskové hry a hlavolamy budou žákům blízké, a zda jejich princip pro ně bude snadno pochopitelný.

4.1 Tic-tac-toe

Tic-tac-toe je „hra na papíře“ pro dva hráče. Jedná se o jednu z nejznámějších a zároveň nejjednodušších variant piškvorků, která se hraje na herním plánu 3×3. Z toho důvodu se jedná o variantu, která je opět nejvhodnější pro děti.

Jedná se o velice starou hru, jejíž historie sahá až do starověkého Egypta, kde byly nalezeny střešní tašky, které pravděpodobně sloužily jako herní desky ke hře podobné této variantě piškvorků. Jejich stáří je odhadováno na 1300 př. n. l. Podobná hra se také hrála ve starověkém Římě okolo 1. století př. n. l. V této variantě se používaly kameny, přičemž každý hráč měl vždy pouze tři a ty mohl přesouvat.[5] [26]

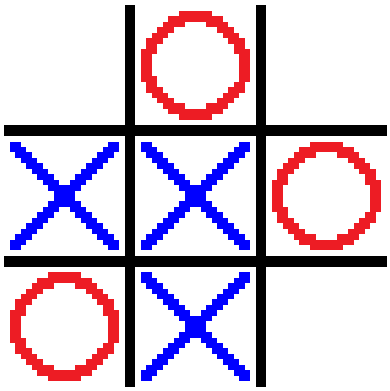
Jeden hráč používá křížky (obvykle modré) a druhý kolečka (obvykle červená). Hráči se střídají, přičemž při každém tahu příslušný hráč nakreslí do neobsazeného čtverečku svůj symbol. Vítězí hráč, který jako první utvoří horizontální, vertikální či diagonální řadu tří stejných symbolů.

Z důvodu velice malého herního pole je maximálním možným počtem tahů 9, kde začínající hráč má tahů 5 a jeho soupeř má tahy 4. Pokud oba hráči hrají

[5] Piškvorky na herním poli 3×3, tedy Tic-tac-toe, v podstatě odpovídají variantě mlýnu známé jako mlýnek či v angličtině three men’s morris.

správně, hra vždy skončí remízou. Tato varianta piškvorků tedy není příliš kompetitivní.

Obrázek 1 zachycuje průběh možné hry Tic-tac-toe. Momentálně by byl na tahu hráč používající modré křížky, a jeho logickými tahy by bylo obsadit buď levé horní, nebo pravé spodní pole. Hráč s červenými kolečky by následně obsadil druhé výše zmíněné pole, aby soupeři zabránil vytvořit řadu tří křížků. V tu chvíli by pak již bude rozhodnuto, že hra skončí remízou, protože už nebude možné řadu tří stejných symbolů utvořit.



Obrázek 1: Tic-tac-toe

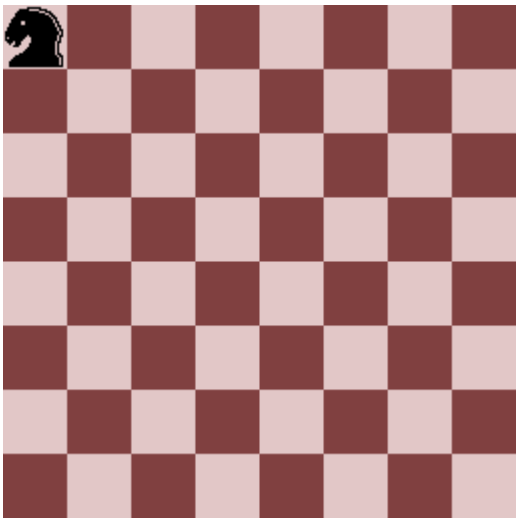
4.2 Jezdcova procházka

Jezdcova procházka je šachový problém pro jednoho hráče. Standardní varianta se hraje na klasické šachovnici o rozměrech 8×8 polí, a to s jednou figurkou jezdce. Hráč začíná umístěním jezdce na jakékoliv pole šachovnice a následně se snaží jezdcem, kterým táhne stejně jako klasický šachový jezdec do tvaru písmene L, navštívit všechna pole šachovnice, a to vždy právě jednou.

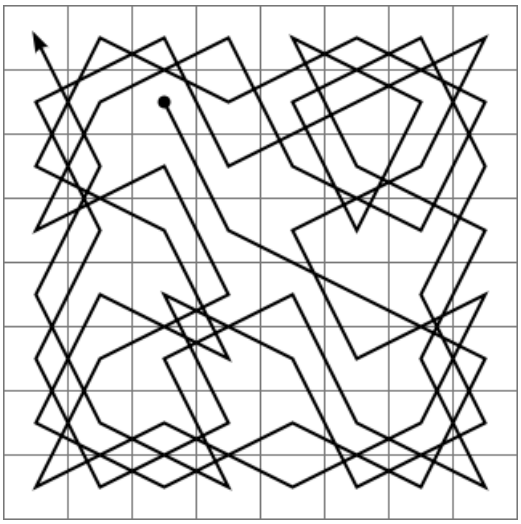
Ačkoli zde hovoříme o figurce jezdce a šachovnici, tento problém lze řešit i na obyčejném čtverečkovaném papíře za použití tužky – tato možnost se dokonce jeví jako efektivnější, protože hráč má dobrý přehled o tom, která pole již navštívil. Stejně tak herní plán nemusí mít rozměry zmíněných 8×8 polí, ale může být i větší nebo menší, nebo ani nemusí mít tvar čtverce. Jedná-li se o jezdcovu procházku na čtvercovém herním plánu, aby ji bylo možné dokončit, je nutné, aby čtverec měl rozměry alespoň 5×5 polí.

Jezdcova procházka může být buď úplná, tj. když jezdec skončí procházku na poli, odkud se může následujícím tahem vrátit na svou výchozí pozici, nebo neúplná, tj. když skončí kdekoliv jinde.

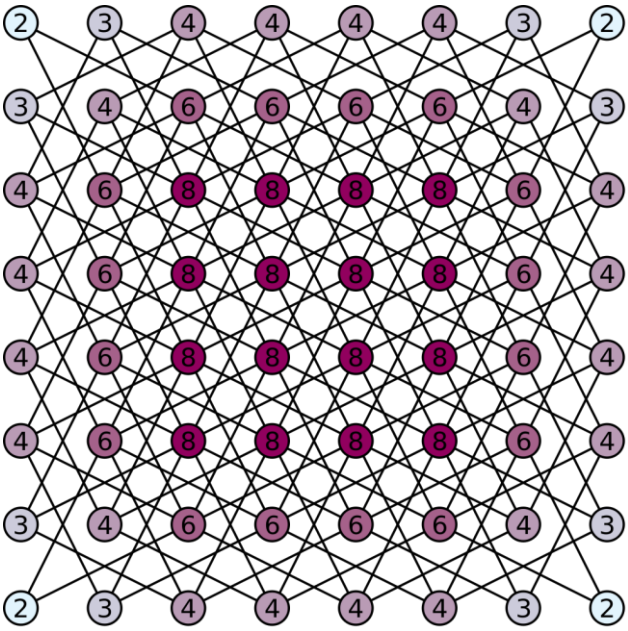
Problém jezdcovy procházky úzce souvisí s teorií grafů (ačkoli analogie s teorií grafů existuje i u řady dalších tahových hlavolamů), jelikož si šachovnici lze představit jako graf o 64 uzlech (nebo popř. jiném počtu v závislosti na rozměrech šachovnice), přičemž hrany tohoto grafu by představovaly tahy, které jezdec může z daného pole udělat. Toto znázornění jezdcovy procházky lze vidět na **obrázku 4**. Pokud ještě zůstaneme u terminologie používané v teorii grafů, řešením jezdcovy procházky je nalezení libovolné hamiltonovské cesty v takovém grafu, a v případě uzavřené jezdcovy procházky je řešením nalezení libovolné hamiltonovské kružnice. [27]



Obrázek 2: Jezdcova procházka



Obrázek 3: Jezdcova procházka – možné řešení



Obrázek 4: Jezdcova procházka – graf

4.3 Hanojské věže

Hanojské věže (někdy také Brahmovy věže) jsou tahový hlavolam pro jednoho hráče. Jedná se o velice známý hlavolam, na který je často odkazováno i v populární kultuře a jehož variace se často vyskytují jako puzzle (tedy hádanky) v různých videohrách.[6]

Hlavolam vymyslel v roce 1883 francouzský matematik Édouard Lucas na základě legendy o chrámu, který se nachází někde v Indii (odtud Brahmovy věže), nebo ve Vietnamu (odtud Hanojské věže), podle varianty legendy. V tomto chrámu stojí 3 kolíky a na nich dohromady 64 kotoučů. Kněží chrámu se snaží přemístit všechny kotouče z jednoho kolíku na druhý, přičemž vždy přemísťují pouze jeden kotouč, a nikdy nesmí položit větší kotouč na kotouč menší. Až se jim podaří hlavolam vyřešit, nastane konec světa. Detaily legendy se liší[7], ale princip zůstává stejný. [28]

[6] Z autorovy zkušenosti např. v počítačové hře *The Fish Fillets 2*, což je česká logická hra z roku 2007, nebo v počítačové hře *Mass Effect*, což je světově proslulá akční RPG hra z roku 2008. Další reference na tento hlavolam lze najít např. na anglické Wikipedii: https://en.wikipedia.org/wiki/Tower_of_Hanoi

[7] V *The Fish Fillets 2* legenda namísto o kněžích v chrámu v Indii nebo ve Vietnamu hovoří o třinácti nesmrtelných chobotnicích na dně Jihočínského moře. Jejich úděl vyřešit Hanojské věže a tím způsobit konec světa však zůstává.

Od legendy se odvíjí i pravidla hlavolamu: máme tři kolíky a na prvním z nich je určitý počet disků, které jsou seřazeny od největšího vespod po nejmenší nahoře. Naším cílem je za pomoci prostředního kolíku přemístit tyto disky na kolík poslední, přičemž při každém tahu můžeme přemístit pouze jeden disk, který se nachází na vrcholu některého z kolíků (nelze tedy brát disky zespoda), a ten následně musíme umístit na jiný kolik tak, aby nikdy větší disk nebyl umístěn nad diskem menším.

V případě Hanojských věží existuje jednoduchý algoritmus na jejich řešení. [28] A je počáteční kolík, B je odkládací kolík a C je cílový kolík.

Pokud je počet disků sudý:

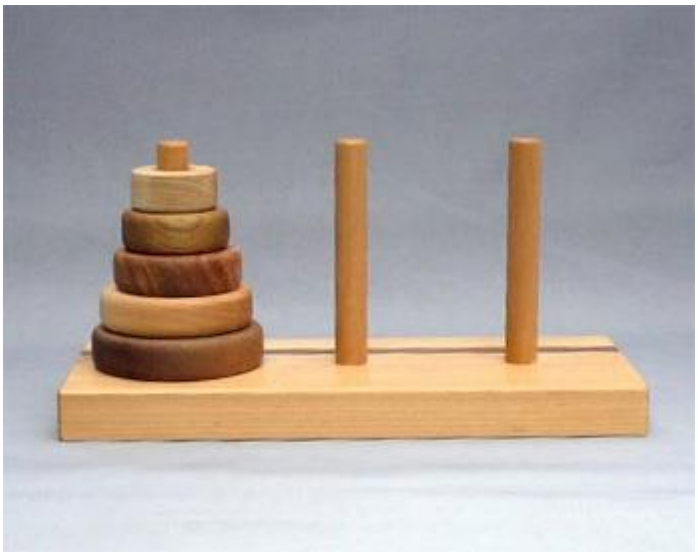
1) Udělej legální tah mezi kolíky A a B v jakémkoliv směru.
2) Udělej legální tah mezi kolíky A a C v jakémkoliv směru.
3) Udělej legální tah mezi kolíky B a C v jakémkoliv směru.
4) Opakuj kroky 1) až 3), dokud není vyřešeno.

Pokud je počet disků lichý:

1) Udělej legální tah mezi kolíky A a C v jakémkoliv směru.
2) Udělej legální tah mezi kolíky A a B v jakémkoliv směru.
3) Udělej legální tah mezi kolíky B a C v jakémkoliv směru.
4) Opakuj kroky 1) až 3), dokud není vyřešeno.

Pokud hráč postupuje podle kroků v algoritmu výše, vyřeší Hanojské věže nejmenším možným počtem tahů, který je vždy $2^n - 1$, kde n je počet disků.

Hanojské věže o pěti discích jsou k vidění na **obrázku 5**. Jelikož $2^5 - 1 = 31$, tento hlavolam lze vyřešit v 31 krocích.



Obrázek 5: Hanojské věže s pěti disky

5 Vlastní naprogramované deskové hry a hlavolamy v aplikaci Scratch

Tato kapitola se věnuje třem programům, které autor této diplomové práce vytvořil specificky pro její účely, a to v aplikaci Scratch 3 Desktop. Veškerá grafika použitá v těchto programech byla taktéž vytvořena autorem, a to specificky pro účely této práce.

Kapitola obsahuje tři podkapitoly, kde se každá z nich věnuje jednomu programu. Každá z těchto podkapitol je pak po krátkém úvodu dále členěna na sekce **Sprity a proměnné** a **Program**. V sekci **Program** jsou uvedeny i části kódu programu. Vzhledem k tomu, že Scratch nepracuje ve formě textu, kód použitý v této kapitole je přepisem textu bloků. Z programátorského hlediska je však tento „kód“ snadno čitelný a jeho pochopení by nemělo představovat problémy. Na závěr podkapitoly každého programu je pak vždy k vidění několik screenshotů z příslušné aplikace: snímek aplikace po jejím spuštění, snímek z průběhu hry, a nakonec snímek po jejím dokončení.

Autor práce se pro lepší srozumitelnost a snazší pochopení rozhodl používat Scratch v anglickém jazyce namísto jeho české lokalizace. Tato volba se pochopitelně projevuje nejen na screenshotech z editoru, ale i na přepisech samotného kódu. Autor se totiž domnívá, že anglická terminologie je pro programátory naprosto běžná, a to i pro ty, kteří angličtinu příliš neovládají. Terminologie používaná v české lokalizaci by tedy byla spíše na škodu než k užitku. Dalším důvodem pro použití angličtiny je pak estetická stránka.

Z těchto důvodů (a celkově pro zachování soudržnosti) se autor rozhodl používat anglický jazyk i v programech samotných, a to ve jménech proměnných, spritů či vlastních bloků. Angličtina byla použita i v uživatelském rozhraní, aby programy byly po zveřejnění dostupné více uživatelům.

Použitá programová řešení nemusí být vždy dokonalá či neskonale elegantní, avšak programy fungují tak, jak stanovují pravidla příslušné hry či hlavolamu, což byl hlavní účel.

Kromě toho je kód uspořádaný tak přehledně, jak to jen Scratch umožňuje. Ve všech programech byly dodržovány podobné postupy – např. inicializace všech globálních proměnných je vždy součástí kódu náležícímu scéně programu a je

provedena vždy po spuštění programu. Po inicializaci proměnných je zbytku programu rozeslána zpráva *new_game*, po jejímž obdržení dojde k inicializaci lokálních proměnných a resetu pozic a případně kostýmů všech spritů. Tyto úvodní procedury jsou součástí kódu všech tří programů.

Programy rozebírané v této kapitole rovněž byly autorem práce využity během jeho působení na ZŠ Sever ve školním roce 2018/2019 při výuce programovacího jazyka Scratch v rámci hodin informatiky. Programy byly žákům předvedeny jak pro svou motivační úlohu, tak za účely demonstrace pojmů souvisejícím s programovacím jazykem Scratch a s programováním a algoritmizací všeobecně. Ve spojení s těmito programy byli žáci seznámeni s pojmy program, algoritmus, cyklus a větvení.

5.1 Tic-tac-toe

Nejjednodušším z trojice programů je Tic-tac-toe. Tento program se řídí pravidly uvedenými v **podkapitole 4.1**.

Zjednodušenou verzi tohoto programu autor práce vytvářel s žáky během několika hodin informatiky. Toto zjednodušení spočívalo ve vypuštění kontroly ukončení hry.

5.1.1 Sprity a proměnné

Po spuštění programu uživatel uvidí mřížku o rozměrech 3×3. Tato mřížka není tvořena pozadím, které jsme u Tic-tac-toe ponechali bílé, ale devíti sprity pojmenovanými *1_1*, *1_2*, *1_3*, *2_1*, *2_2*, *2_3*, *3_1*, *3_2* a *3_3*, kde každý představuje jedno pole. Každý z těchto spritů má celkem tři kostýmy, které jsou nazvané *empty*, *cross* a *circle*. *Empty* je výchozí kostým pro prázdné pole a *cross* a *circle* jsou kostýmy pro pole obsazené příslušným hráčem. Kromě devíti spritů polí je v programu ještě sprite *win*, s nímž však hráči přímo neinteragují. Tento sprite slouží k ukázání bubliny s výsledkem hry.

Proměnných je v Tic-tac-toe celkem třináct. Devět proměnných, které jsou pojmenovány *1_1*, *1_2*, …, *3_3*, uchovávají status příslušných polí. Celkem nabývají tří hodnot: 0 pro prázdné pole, 1 pro pole obsazené křížkem a 2 pro pole obsazené kolečkem. Další proměnnou je *draw_counter*, která slouží k počítání odehraných tahů, aby bylo možné vyhodnotit po odehraných devíti tazích, kde žádný hráč

nevyhraje, výsledek hry jako remízu. Další proměnnou je *game_running*, která nabývá hodnot 0 a 1 a slouží k sledování, zda právě probíhá hra, nebo jestli hra už skončila. Dále zde je proměnná *player*, která sleduje, který hráč je na tahu – proměnná nabývá hodnot 1 pro hráče s křížky a 2 pro hráče s kolečky. Poslední proměnnou je *winner*, která slouží k určení vítěze – opět hodnota 1 pro hráče s křížky a 2 pro hráče s kolečky. V případě remízy hodnota proměnné w*inner* zůstane 0.

5.1.2 Program

Program náležící scéně provádí inicializaci globálních proměnných po spuštění programu (kliknutí na ikonu zeleného praporku) a po stisknutí klávesy ‚space', kterou dojde k zahájení nové hry. Všechny proměnné uchovávající status příslušných polí, tedy *1_1*, *1_2*, ..., *3_3*, jsou nastaveny na hodnotu 0. Proměnná *game_running* je nastavena na hodnotu 1, *draw_counter* na 0, *winner* na 0 a *player* na 1, jelikož začíná hráč s křížky. Nakonec je zbytku programu rozeslána zpráva *new_game*.

Přepis kódu náležícímu spritům jednotlivých polí vypadá následovně:

```
when I receive new_game
{
  go to x: -120, y: 120;
  switch costume to empty;
};

when this sprite clicked
{
  if game_running == 1 then
  {
    if 1_1 == 0 then
    {
      if player == 1 then
      {
        set 1_1 to 1;
        switch costume to cross;
      } else
      {
        set 1_1 to 2;
        switch costume to circle;
      };
      change draw_counter by 1;
```

```
      broadcast check_for_win;
    };
  };
}.
```

Po obdržení zprávy *new_game* nejprve dojde k resetu souřadnic daného pole a k nastavení jeho kostýmu na *empty*. Při kliknutí na sprite nejprve dojde k ověření, zda hra probíhá. Pokud ano (tedy pokud ještě žádný hráč nevyhrál nebo pokud hra neskončila remízou), dojde k ověření stavu příslušného pole. Pokud je pole prázdné, dojde ke zjištění, který hráč kliknutí provedl. Pokud to byl hráč 1, hodnota proměnné příslušného pole (v uvedeném kusu kódu *1_1*) se změní 1 a kostým pole se změní na *cross*. Pokud ne (z čehož tedy vyplývá, že to byl hráč 2), hodnota proměnné se změní na 2 a kostým pole se změní na *circle*. Po obsazení pole dojde ke zvýšení hodnoty proměnné *draw_counter* o 1 a nakonec se programu rozešle zpráva *check_for_win*, aby došlo ke kontrole, zda tímto tahem nemělo dojít k ukončení hry.

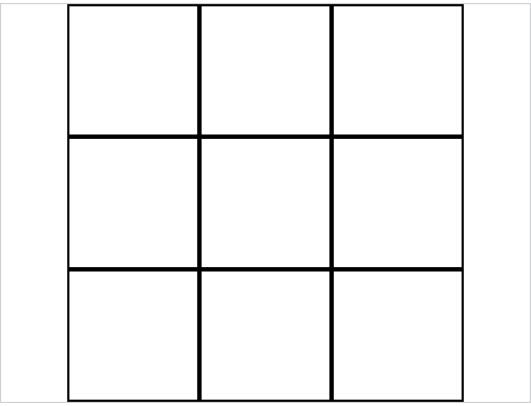
Kontrolu ukončení hry má na starost sprite *win*. Jemu náležící kód vypadá následovně:

```
when I receive new_game
{
  go to x: -58, y: -32;
  say '';
};

when I receive check_for_win
{
  if (1_1 == player and 1_2 == player and 1_3 == player) then
  {
    set winner to player;
  };
  if (2_1 == player and 2_2 == player and 2_3 == player) then
  {
    set winner to player;
  };
  if (3_1 == player and 3_2 == player and 3_3 == player) then
  {
    set winner to player;
  };
  if (1_1 == player and 2_1 == player and 3_1 == player) then
  {
    set winner to player;
  };
```

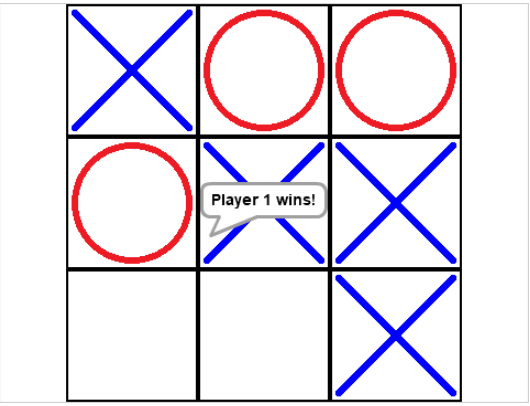
```
  if (1_2 == player and 2_2 == player and 3_2 == player) then
  {
    set winner to player;
  };
  if (1_3 == player and 2_3 == player and 3_3 == player) then
  {
    set winner to player;
  };
  if (1_1 == player and 2_2 == player and 3_3 == player) then
  {
    set winner to player;
  };
  if (1_3 == player and 2_2 == player and 3_1 == player) then
  {
    set winner to player;
  };
  broadcast check_for_win2;
};

when I receive check_for_win2
{
  if winner == 1 then
  {
    set game_running to 0;
    say 'Player 1 wins!';
  };
  if winner == 2 then
  {
    set game_running to 0;
    say 'Player 2 wins!';
  };
  if (winner == 0 and draw_counter == 9) then
  {
    set game_running to 0;
    go to x: -24, y: -32;
    say 'Draw!';
  };
  if player == 1 then
  {
    set player to 2;
  } else
    set player to 1;
  };
}.
```

Po obdržení zprávy *new_game* nejprve dojde k resetu souřadnic spritu (ačkoli je tento sprite tvořen pouze jedním pixelem průhledné barvy, jeho souřadnice určují, kde vyskočí bublina s textem) a k zrušení případné bubliny s textem (bublina zde vyskočí při dokončení hry a bez tohoto kroku by zde bublina zůstala i po zahájení hry nové). Po obdržení zprávy *check_for_win*, která je rozesílána sprity polí po jejich obsazení některým z hráčů, se spustí sekvence osmi podmínek (v Tic-tac-toe totiž existuje pouhých osm konfigurací pro vytvoření řady tří symbolů – tři horizontální, tři vertikální a dvě diagonální), které ověří, zda hráč, který právě pole obsadil, někde nemá vytvořenou trojici stejných symbolů, což by znamenalo jeho výhru. Nezávisle na tom, zda byla některá z těchto podmínek splněná, je rozeslána zpráva *check_for_win2*. Tato zpráva odkazuje na pokračování tohoto skriptu, který je pro větší přehlednost programu v aplikaci Scratch rozdělen na dvě části. Po obdržení zprávy *check_for_win2* je provedena sekvence ověření proměnné *winner*, k jejíž změně mohlo dojít v předchozím skriptu. Pokud některý z hráčů zvítězil, dojde ke změně hodnoty proměnné *game_running* na 0, aby se zabránilo možnosti obsazovat další pole. Poté vyskočí bublina s textem *„Player 1 wins!“* nebo *„Player 2 wins!“*. Následuje další podmínka, a ta je splněna tehdy, když je hodnota proměnné *winner* stále 0 a zároveň hodnota proměnné *draw_counter* 9, což znamená, že už je obsazeno všech devět polí, ale žádný hráč neutvořil řadu tří symbolů, a hra tudíž končí remízou. Při splnění této podmínky taktéž dojde ke změně hodnoty proměnné *game_running* na 0, k posunutí souřadnic spritu a nakonec vyskočí bublina s textem *„Draw!“*. Důvodem pro posun spritu je délka tohoto textu, který je kratší než *„Player 1 wins!“* nebo *„Player 2 wins!“* a díky posunu bublina vyskočí blíže středu herního pole. Následuje poslední podmínka, která opět zkontroluje, čí tah právě proběhl, a změní hodnotu proměnné *player* tak, aby následoval tah druhého hráče. Pokud nedošlo k výhře žádného z hráčů, ani k remíze, hra se nyní vrací do stavu, kdy čeká na kliknutí na další pole.

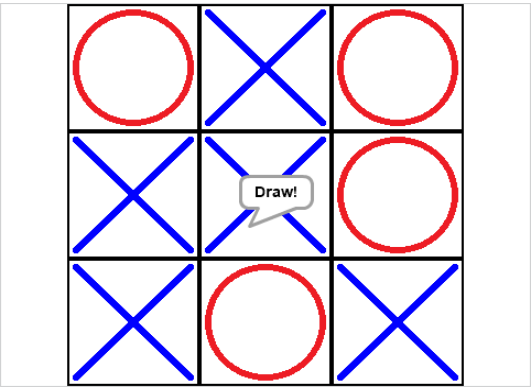
Obrázky 6, **7** a **8** na následující straně zachycují hru v několika možných fázích – na začátku hry, po vítězství hráče 1 a při remíze.



Obrázek 6: Scratch – Tic-tac-toe – prázdné pole



Obrázek 7: Scratch – Tic-tac-toe – Hráč 1 vítězí



Obrázek 8: Scratch – Tic-tac-toe – remíza

5.2 Jezdcova procházka

Druhým programem je Jezdcova procházka na šachovnici o rozměrech 5×5 polí. Až na rozměry šachovnice se tento program řídí pravidly uvedenými v **podkapitole 4.2**. Samotný program se nazývá *Knight's Tour 5×5*.

5.2.1 Sprity a proměnné

Podobně jako u Tic-tac-toe je i v Jezdcově procházce pozadí ponecháno bílé a šachovnice je tvořena jednotlivými sprity. Těch je dohromady dvacet pět a jsou pojmenovány podle stejného postupu jako sprity v Tic-tac-toe, tedy *1_1*, *1_2*, …, *5_5*. Třináct z těchto spritů, které představují světlá pole, obsahuje kostýmy *square_light* pro nenavštívené pole a *square_light_visited* pro pole navštívené. Zbylých dvanáct spritů, které představují pole tmavá, obsahuje kostýmy *square_dark* a *square_dark_visited*. Kromě spritů dvaceti pěti polí je zde ještě sprite samotného jezdce: *knight*. Ten obsahuje pouze jeden kostým, který se jmenuje rovněž *knight*.

V Jezdcově procházce je proměnných celkem pět.[8] Tyto proměnné se nazývají *legal_move*, *move_to_x*, *move_to_y*, *position* a *squares_visited*. Proměnná *legal_move* nabývá dvou hodnot – 0 a 1 – které určují, zdali posun na pole, které se hráč chystá navštívit, je legální, nebo ne. Proměnné *move_to_x* a *move_to_y* slouží k uložení souřadnic pole, které má být navštíveno, a na které se postava jezdce přesune. Proměnná *position* si pamatuje, na kterém poli se hráč momentálně nachází. Na začátku je hodnota této proměnné 0 (tj. jezdec ještě nebyl umístěn na šachovnici) a následně se mění na číslo daného pole – 11 pro pole *1_1*, 12 pro pole *1_2*, až po 55 pro pole *5_5*. Hlavním účelem této proměnné je, aby program mohl ověřit, zdali přesun jezdce na pole, které se hráč snaží navštívit, splňuje pravidlo pro jezdcův pohyb ve tvaru písmene L. Poslední proměnnou je *squares_visited*, která počítá, kolik polí již hráč navštívil – začíná tedy na hodnotě 0 a po úspěšném navštívení každého pole se zvyšuje o 1. Po nabytí hodnoty 25 je hra ukončena, jelikož hráč navštívil všech dvacet pět polí šachovnice.

5.2.2 Program

Program náležící scéně provádí inicializaci globálních proměnných po spuštění programu (kliknutí na ikonu zeleného praporku) a po stisknutí klávesy ‚space', kterou dojde k zahájení nové hry. Těmito proměnnými jsou *legal_move*, *position* a *squares_visited*, a všechny jsou nastaveny na 0. Poté je zbytku programu rozeslána zpráva *new_game*.

Přepis kódu náležícímu spritům jednotlivých polí vypadá následovně:

```
when I receive new_game
{
  go to x: -144, y: 144;
  switch costume to square_light;
};

when this sprite clicked
{
  if position == 0 then
```

[8] Ve starší verzi bylo kromě globálních proměnných použito i dvacet pět proměnných lokálních, kde každá náležela jednomu poli a uchovávala jeho status. Avšak řešení, které je použito v aktuální verzi programu, které nekontroluje proměnné, ale kostým daného pole, je úhlednější a celkově elegantnější.

```
  {
    set legal_move to 1;
  };
  if position == 23 then
  {
    set legal_move to 1;
  };
  if position == 32 then
  {
    set legal_move to 1;
  };
  if (costume_number == 1 and legal_move == 1) then
  {
    set position to 11;
    change squares_visited by 1;
    switch costume to square_light_visited;
    set move_to_x to x position;
    set move_to_y to y position;
    broadcast move_here;
  };
  set legal_move to 0;
}.
```

Po obdržení zprávy *new_game* nejprve dojde k resetu souřadnic daného pole a k nastavení jeho kostýmu na *square_light* v případě pole světlého a *square_dark* v případě pole tmavého. Při kliknutí na sprite je zkontrolována série podmínek ověřující hodnotu v proměnné *position*, čímž dojde k posouzení, zda tah, o který se hráč pokouší, je legální. První podmínka ověřuje, zda hodnota proměnné *position* není 0 (tzn. zda se nejedná o úplně první tah, kde hráč může jezdce umístit na jakékoliv pole) a pak následuje několik podmínek ověřující legalitu tahu v souladu s pravidly pro pohyb šachového jezdce.[9] Pokud je některá z těchto podmínek splněna, dojde k nastavení hodnoty proměnné *legal_move* na 1. Po této sérii podmínek následuje ještě jedná podmínka, která ověřuje právě proměnnou *legal_move* (pro splnění podmínky musí být její hodnota 1) a číslo aktuálního kostýmu daného pole (to musí být 1, což představuje kostým *square_light* pro pole světlé a *square_dark* pro pole tmavé). Pokud jsou obě tyto podmínky splněny,

[9] V uvedeném kódu jsou pole, z nichž je tah na toto pole legální, pouze dvě: *2_3* a *3_2*. Avšak u jiných polí je takový tah možný ze tří, čtyř, šesti, nebo osmi polí (viz **Obrázek 5: Jezdcova procházka – graf**). V jezdcově procházce na šachovnici o rozměrech 5×5 polí je takové pole pouze jedno, a sice to prostřední: *3_3*. Toto pole lze navštívit z polí *1_2*, *1_4*, *2_1*, *2_5*, *4_1*, *4_5*, *5_2* a *5_4*.

znamená to, že přesun jezdce na toto pole může proběhnout a dojde ke spuštění několika řádek kódu: proměnná *position* se nastaví na číslo tohoto pole (v případě pole *1_1* je hodnota 11, pro *1_2* to je 12 atd.), proměnná *squares_visited* se zvýší o 1, kostým pole se změní na kostým představující navštívené pole (*square_light_visited* pro pole světlé a *square_dark_visited* pro pole tmavé), proměnné *move_to_x* a *move_to_y* se nastaví na souřadnice pole (tyto souřadnice je třeba uložit do podoby globální proměnné, aby sprite *knight* představující figurku jezdce mohl přečíst souřadnice pole, na něž se má přemístit) a na závěr se rozešle zpráva *move_here*, která spustí skript pro přesun na toto pole u spritu *knight* (viz přepis kódu níže). Po provedení větve programu popsané výše se už jen hodnota proměnné *legal_move* opět změní na 0.

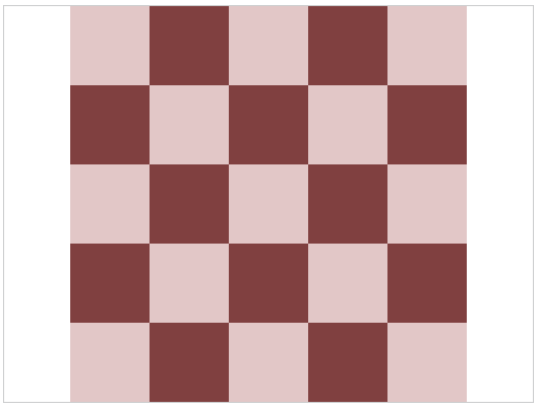
Nyní následuje přepis kódu spritu *knight*:

```
when I receive new_game
{
  hide;
};

when I receive move_here
{
  show;
  go to x: move_to_x, y: move_to_y;
  if squares_visited == 25 then
  {
    say 'Congratulations!' for 5 seconds;
  };
}.
```

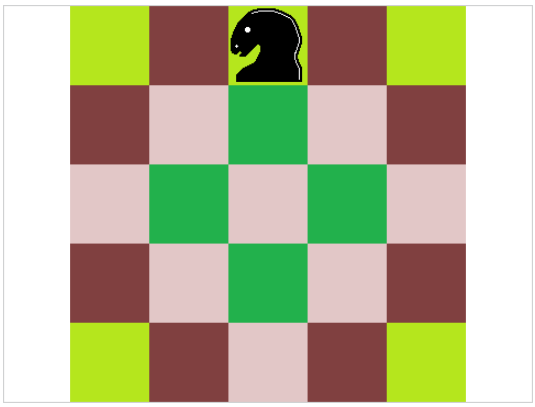
Po obdržení zprávy *new_game* dojde ke skrytí spritu jezdce. Takto skrytý čeká na obdržení zprávy *move_here*, která je zasílána sprity polí po kliknutí na ně a vyhodnocení tahu jako legálního.

Při obdržení zprávy *move_here* se spustí skript, který sprite jezdce nejprve zobrazí a následně přesune na souřadnice uložené v proměnných *move_to_x* a *move_to_y*. Poté dojde k ověření proměnné *squares_visited*, a pokud je její hodnota rovna 25 (tedy že jezdec navštívil všech dvacet pět polí šachovnice), u spritu jezdce se na pět sekund zobrazí bublina s textem *„Congratulations!“*.

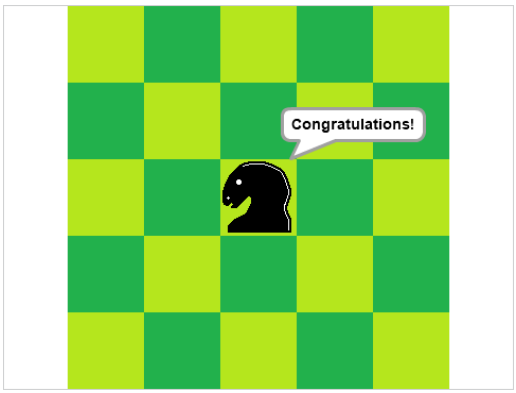
Obrázky 9, **10** a **11** na následující straně zachycují hru v několika fázích – při zahájení hry, kdy sprite jezdce zůstává skrytý, v průběhu hry po navštívení devíti polí (jezdec se právě nachází na poli *1_3*) a po úspěšném projití celé šachovnice.



Obrázek 9: Scratch – Jezdcova procházka – začátek



Obrázek 10: Scratch – Jezdcova procházka – průběh



Obrázek 11: Scratch – Jezdcova procházka – dokončeno

5.3 Hanojské věže

Třetím a tudíž posledním programem jsou Hanojské věže se šesti disky. Tento program se řídí pravidly uvedenými v **podkapitole 4.3**.

5.3.1 Sprity a proměnné

Tentokrát pozadí obsahuje hnědou podkladovou „desku“ a tři kolíky. Zbytek je bílý. Kolíky jsou však z důvodu, aby kliknutím na ně hráč udělal tah, graficky znázorněny i pomocí kostýmů jim náležejících spritů. Spritů je zde celkem devět – tři sprity pojmenované *pole1*, *pole2* a *pole3* představující jednotlivé kolíky a šest spritů pojmenovaných *disc1*, *disc2*, ..., *disc6* představující barevné disky, přičemž *disc1* představuje největší červený disk a čím vyšší číslo sprite disku má, tím menší je. Každý sprite má pouze jeden kostým – v případě kolíků to je jeden kostým *pole* a v případě disků to jsou opět *disc1*, *disc2*, ..., *disc6*.

V Hanojských věžích je použito celkem sedmnáct proměnných. Jedenáct proměnných – *moving_piece*, *pole1*, *pole1_discs*, *pole1_top*, *pole2*, *pole2_discs*, *pole2_top*, *pole3*, *pole3_discs*, *pole3_top* a i (proměnná pro použití v cyklech) – je

globálních. Zbylých šest lokálních proměnných se nazývá *on_pole* a každá z nich náleží jednomu z disků. Tyto lokální proměnné obvykle nabývají hodnot 1, 2 a 3, což značí, na kterém z kolíků se příslušný disk momentálně nachází. Kromě toho ještě mohou nabývat hodnoty 0, která je přiřazena poté, co je daný disk zdvižen a čeká na umístění na některý z kolíků. Z globálních proměnných proměnná *moving_piece* značí, který disk je momentálně zvednutý. Tato proměnná nabývá hodnoty 0 tehdy, když žádný z disků není zdvižený a následně 1, 2, 4, 8, 16 nebo 32 v případě, že je zdvižený některý z disků. Je zcela zřejmé, že se jedná o mocniny dvou, které zde jsou použity jako flagy, což hraje svou roli při součtu hodnot všech disků na daném kolíku. Tento součet hodnot je uložen v proměnných pole1, pole2 a pole3, kde každá z nich náleží jednomu z kolíků. Pro vysvětlení výpočtu hodnot může posloužit **tabulka 1**:

Tabulka 1: Hanojské věže – flagy disků

	fialový	modrý	zelený	žlutý	oranžový	červený
Decimální hodnota	1	2	4	8	16	32
Binární hodnota (6 bitů)	000001	000010	000100	001000	010000	100000

Díky použití těchto binárních flagů lze jedním číslem určit konkrétní konfiguraci disků momentálně nasazených na daném kolíku. Kolík bez disků má hodnotu 0, zatímco kolík se všemi šesti disky má hodnotu 63 (32+16+8+4+2+1; v binárním zápise 111111). Proměnná *pole1* má tedy na začátku hodnotu 63 a proměnné *pole2* a *pole3* mají hodnoty 0. Pokud má proměnná *pole3* hodnotu 63, znamená to, že se na cílový kolík hráči podařilo přesunout všech šest disků, a tudíž byl hlavolam úspěšně vyřešen. Proměnné *pole1_discs*, *pole2_discs* a *pole3_discs* pak uchovávají hodnotu podle počtu disků na daném kolíku, která je využívána pro výpočet y-souřadnice pro přesun disku po jeho umístění na daný kolík. Proměnné *pole1_top*, *pole2_top* a *pole3_top* pak uchovávají hodnotu podle nejvyššího disku na daném kolíku, která se při pokusu o umístění disku na daný kolík porovnává s hodnotou disku právě zdviženého.

5.3.2 Program

Program náležící scéně provádí inicializaci globálních proměnných po spuštění programu (kliknutí na ikonu zelené vlaječky) a po stisknutí klávesy ‚space', kterou dojde k zahájení nové hry. Všechny proměnné kromě těch související s prvním kolíkem jsou nastaveny na 0. Těmito proměnnými jsou *pole2*, *pole3*, *pole2_discs*, *pole3_discs*, *pole2_top*, *pole3_top* a *moving_piece*. Hodnota proměnné *pole1* je nastavena na 63 (celková hodnota všech šesti disků, viz popis výše), *pole1_discs* je nastavena na 6 (na kolíku je všech šest disků) a *pole1_top* je nastavena na 1 (nahoře je fialový disk, který má hodnotu 1). Poté je zbytku programu rozeslána zpráva *new_game*.

Ve srovnání s Tic-tac-toe a Jezdcovou procházkou se kód jednotlivých spritů v Hanojských věžích liší poněkud výrazněji. Nyní následuje první část přepisu kódu náležícímu spritu *disc6* (tedy nejmenšímu disku majícímu fialovou barvu):

```
when I receive new_game
{
  go to x: -152, y: 20;
  set on_pole to 1;
  go to front layer;
};

when this sprite clicked
{
  if moving_piece == 0 then
  {
    get_picked_up;
  };
};

define get_picked up
{
  set moving_piece to 1;
  set y to 152;
  if on_pole == 1 then
  {
    change pole1 by -1;
    change pole1_discs by -1;
    broadcast check_pole1_top;
  };
  if on_pole == 2 then
  {
```

```
    change pole2 by -1;
    change pole2_discs by -1;
    broadcast check_pole2_top;
  };
  if on_pole == 3 then
  {
    change pole3 by -1;
    change pole3_discs by -1;
    broadcast check_pole3_top;
  };
  set on_pole to 0;
};
```

Po obdržení zprávy *new_game* dojde k resetu souřadnic spritu a k nastavení lokální proměnné *on_pole* na 1. Kromě toho dojde k přesunu spritu do popředí, aby nedošlo k tomu, že bude zakrytý spritem kolíku.

Po kliknutí na sprite dojde k ověření hodnoty proměnné *moving_piece*. Pokud by nebyla 0, znamenalo by to, že některý z disků je již zvednutý, a tudíž by zvednutí dalšího disku bylo proti pravidlům. Pokud však žádný disk zvednutý není, dojde ke spuštění vlastního bloku *get_picked_up*.

Vlastní blok *get_picked_up* provede několik akcí. Nejprve nastaví hodnotu proměnné *moving_piece* na 1 (u jiných disků to je vždy jejich příslušná hodnota – viz **tabulka 1** výše). Poté se změní y-souřadnice spritu – tedy dojde ke zdvižení disku. Následují tři podmínky ověřující, ze kterého kolíku byl disk zdvižen. Po identifikaci kolíku dojde ke změně příslušných proměnných – např. u kolíku 1 se sníží hodnota proměnné *pole1* o 1 (u jiných disků by se opět jednalo o hodnotu uvedenou v **tabulce 1**) a *pole1_discs* o 1. Poté je rozeslána zpráva *check_pole1_top*, kterou zaregistruje sprite příslušného kolíku (tedy v tomto případě *pole1*) a ověří, který disk je momentálně vrchní.

V případě spritu *disc6* použití vlastního bloku nebylo nutné, jelikož kód bloku *get_picked_up* mohl být jednoduše obsažen ve větvení spuštěném po kliknutí na sprite. Avšak v případě ostatních disků použití vlastního bloku ušetří značnou část kódu a vede k celkově vyšší přehlednosti. Následující část kódu náleží spritu *disc1* (tedy největšímu disku majícímu červenou barvu):

```
when this sprite clicked
{
  if moving_piece == 0 then
  {
```

```
    if (on_pole == 1 and pole1_top > 16) then
    {
      get_picked_up;
    };
    if (on_pole == 2 and pole2_top > 16) then
    {
      get_picked_up;
    };
    if (on_pole == 3 and pole3_top > 16) then
    {
      get_picked_up;
    };
  };
};
```

U všech disků kromě toho nejmenšího je totiž v Hanojských věžích nutno kontrolovat, zdali se nad daným diskem nenachází žádný jiný disk, protože poté by jej nebylo možné zvednout (porušovalo by to pravidlo, že lze přesouvat vždy pouze jeden disk). Z toho důvodu je třeba po kliknutí na disk pomocí trojice podmínek zjistit, na kterém z kolíků se tento disk nachází. Rovněž je potřeba ověřit hodnotu horního disku na daném kolíku, k čemuž jsou použity proměnné *pole1_top*, *pole2_top* či *pole3_top*. Bez tohoto ověření by pak bylo možné zvednout například největší disk mající červenou barvu i tehdy, když by se nad ním nacházely i jiné disky.

Ke kódu náležícímu spritům disků se v této podkapitole ještě budeme vracet, avšak nyní je třeba přejít ke kódu náležícímu spritům kolíků, tedy *pole1*, *pole2* a *pole3*. Následující přepis kódu náleží spritu *pole1*, tedy výchozímu kolíku:

```
when I receive new_game
{
  go to x: -152, y: -16;
  go to back layer;
};

when this sprite clicked
{
  if moving_piece > 0 then
  {
    broadcast move_to_pole1;
  };
};

when I receive check_pole1_top
```

```
{
  if pole1 == 0 then
  {
    set pole1_top to 0;
  } else
  {
    set i to 1;
    repeat until i > 32
    {
      if pole1 mod (i * 2) == i then
      {
        set pole1_top to i;
        stop this script;
      } else
      {
        change i by i;
      };
    };
  };
}.
```

Po obdržení zprávy *new_game* dojde k resetu souřadnic spritu. Poté dojde k přesunu spritu do pozadí, aby nedošlo k tomu, že bude překrývat sprity disků.

Skript, který se spustí po kliknutí na sprite kolíku ověří, zdali je momentálně zdvižený některý z disků. K tomu potřebuje porovnat proměnnou *moving_piece* proti 0. Pokud má proměnná vyšší hodnotu (v případě našeho programu to mohou být hodnoty 1, 2, 4, 8, 16 nebo 32), je rozeslána zpráva *move_to_pole1* (v případě zbylých dvou kolíků *move_to_pole2* a *move_to_pole3*).[10]

Při obdržení zprávy *check_pole1_top* (resp. *check_pole2_top* a *check_pole3_top* v případě zbylých dvou kolíků), kterou rozesílají jednotlivé sprity disků, pokud jsou úspěšně zdviženy, následuje skript, jehož cílem je zjistit, jakou hodnotu má následující nejvyšší disk na daném kolíku. Zde se může jednat o hodnoty 0 (žádný disk na kolíku nezbývá), 2 (modrý disk), 4 (zelený disk), 8 (žlutý disk), 16 (oranžový disk) nebo 32 (červený disk). Hodnota 1 zde ani není zahrnuta, protože fialový disk může vždy být pouze nejvyšším diskem. Skript nejprve provede větvení ověřující, zdali na kolíku vůbec nějaký disk je. Pokud ano, tj. pokud proměnná *pole1* (resp. *pole2* nebo *pole3*) je rovna 0, hodnota proměnné

[10] Po obdržení této zprávy jednotlivé sprity disků spustí skript, kterým ověří, zdali se jedná o disk, který je právě zdvižený. Přepis těchto skriptů bude následovat později.

pole1_top[11] se rovněž nastaví na 0 a tím skript končí. Pokud ne, druhá větev nastaví hodnotu proměnné *i* na 1 a spustí se cyklus, který je opakován, dokud hodnota proměnné *i* nebude vyšší než 32. V každém opakování cyklu je provedeno ověření zbytku po vydělení *pole1* dvojnásobkem *i*. Pokud je zbytek roven *i*, hodnota proměnné *pole1_top* se nastaví na hodnotu *i* a poté se skript ukončí. V opačném případě se hodnota *i* zvýší o *i* (tedy se zdvojnásobí) a pokračuje se dalším opakováním cyklu. Po ukončení cyklu zůstane hodnota proměnné *pole1_top* na hodnotě nejvyššího disku.

Pro uvedení příkladu předpokládejme, že hodnota proměnné *pole1* bude 44.[12] Při prvním opakování cyklu je zbytek po vydělení 44 číslem 2 roven 0, takže se proměnná *i* zdvojnásobí a pokračuje se dalším opakováním. Při druhém opakování je zbytek po vydělení 44 číslem 4 opět roven 0 a dojde k dalšímu zdvojnásobení *i* a cyklus se opakuje. Při třetím opakování však zbytek po vydělení 44 číslem 8 je roven 4, takže podmínka je splněna a hodnota proměnné *pole1_top* je nastavena na 4, což odpovídá hodnotě zeleného disku. Skript je poté ukončen.

Toto poměrně elegantní řešení je další výhodou použití hodnot disků rostoucích s druhou mocninou čísla 2 popsané dříve v této podkapitole.

Nyní se vrátíme ke kódu disků, který se spustí po obdržení zprávy *move_to_pole1* (resp. *move_to_pole2* a *move_to_pole3*). Následující kód náleží konkrétně spritu *disc2*, tedy druhému největšímu disku majícímu oranžovou barvu:

```
when I receive move_to_pole1
{
  if moving_piece == 16 then
  {
    if (pole1_top == 0 or pole1_top > moving_piece) then
    {
      set on_pole to 1;
      change pole1_discs by 1;
      set moving_piece to 0;
      set x to -152;
      set y to (-172 + (32 * pole1_discs));
```

[11] Resp. *pole2_top* nebo *pole3_top*. Tyto paralelní proměnné pro zbylé kolíky dále již nebudou v hlavním textu vypisovány.

[12] Pro zajímavost, hodnota 44 odpovídá konfiguraci 32 + 8 + 4, na kolíku jsou tedy červený, žlutý a zelený disk.

```
      change pole1 by 16;
      set pole1_top to 16;
    };
  };
};
```

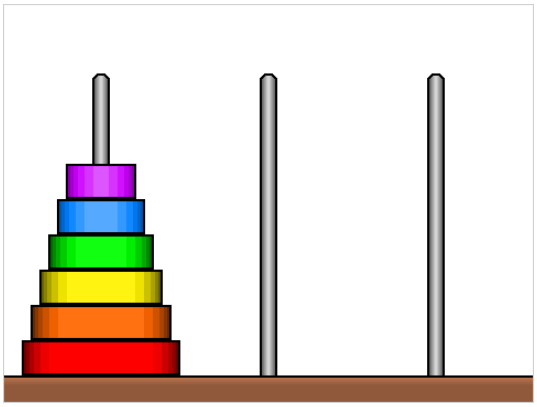
Po obdržení zprávy *move_to_pole1* nejprve dojde k ověření, zdali je tento disk právě zdvižený – v případě oranžového disku musí být hodnota proměnné *moving_piece* rovna 16 (pro hodnoty ostatních disků viz **tabulka 1**). Pokud je tato podmínka splněna, je rovněž potřeba ověřit, zdali tento disk může být na daný kolík umístěn. Aby bylo umístění disku legální, musí být daný kolík být buď prázdný – tedy hodnota proměnné *pole1_top* musí být rovna 0 – nebo se na jeho vršku musí nacházet větší disk – tedy hodnota proměnné *pole1_top* musí být větší než hodnota zdviženého disku, což je v případě oranžového disku 16. V případě spritu *disc1*, tedy největšího disku červené barvy, lze disk umístit pouze tehdy, když je hodnota proměnné *pole1_top* rovna 0, protože největší disk nemůže být pro svou velikost umístěn na jiný z disků. Oproti tomu u spritu *disc6*, tedy nejmenšího disku fialové barvy, tato podmínka může být zcela vypuštěna, protože je možné jej umístit kamkoliv, ať už je daný kolík prázdný, či obsahuje jakoukoliv konfiguraci disků, jelikož všechny jsou větší.

Pokud jsou podmínky pro umístění tohoto disku splněny, dojde k sérii změn v hodnotách příslušných proměnných. V tomto případě je hodnota lokální proměnné *on_pole* nastavena na 1 (v případě zbylých dvou kolíků by to pochopitelně bylo 2, resp. 3). Hodnota *pole1_discs* je zvýšena o 1 a hodnota *moving_piece* je nastavena na 0. Poté dojde k přesunu samotného spritu: x-souřadnice se změní na −152 (v případě druhého a třetího kolíku by to bylo 0, resp. 152) a y-souřadnice se změní na −172 zvýšených o 32 (což je výška spritů všech disků) za každý disk na daném kolíku (takže y-souřadnice disků umístěných na kolících mohou být −140, −108, −76, −44, −12 nebo 20). Po přesunu spritů je ještě zvýšena hodnota proměnné *pole1* o 16 (tedy o hodnotu disku) a *pole1_top* nastavena rovněž na 16.

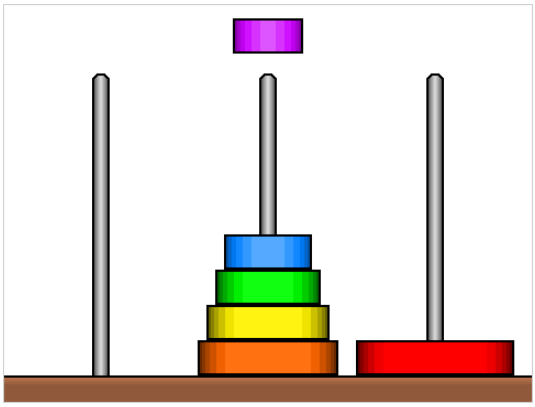
V případě přesunu nejmenšího disku fialové barvy na třetí kolík (tedy cílový kolík úplně vpravo) by ještě následovalo rozeslání zprávy *check_for_win*, po jejímž obdržení by sprite *pole3* ověřil, zdali je hodnota proměnné *pole3* rovna 63 (tedy

součtu hodnot všech disků), což by znamenalo výhru a zobrazila by se bublina s textem *„Congratulations!“*.

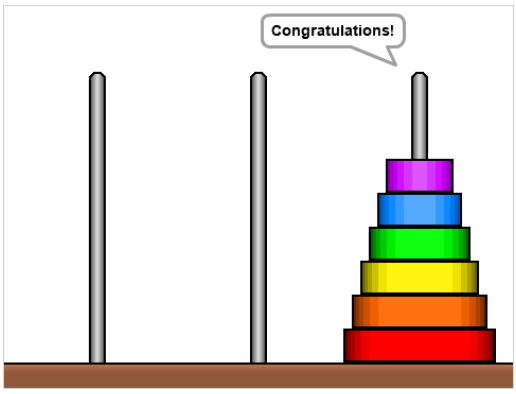
Obrázky 12, **13** a **14** zachycují hru v několika fázích – při zahájení hry, kdy jsou všechny disky umístěny na prvním kolíku, v průběhu hry, kdy už byl červený disk umístěn na cílový kolík a momentálně byl z prostředního kolíku zdvižen fialový disk, a po ukončení hry úspěšným přemístěním všech disků na cílový kolík.



Obrázek 12: Scratch – Hanojské věže – začátek



Obrázek 13: Scratch – Hanojské věže – průběh



Obrázek 14: Scratch – Hanojské věže – dokončeno

6 Dotazníkové šetření na téma programování a jeho výuce ve škole

Během svého působení na pracovní pozici učitele na ZŠ Sever na adrese Lužická 1208 v Hradci Králové autor práce ve třídách 6. A, 6. B a 7. B, ve kterých vyučoval informatiku, mimo běžnou náplň předmětu vyučoval i programovací jazyk Scratch. Koncem dubna 2019 mezi žáky těchto tříd došlo k dotazníkovému šetření na téma programování a jeho výuce ve škole (seznam otázek z dotazníku je k nahlédnutí v **podkapitole 6.3** a výběr některých odpovědí v **podkapitole 6.4**).

6.1 Výuka programovacího jazyku Scratch na ZŠ Sever

ZŠ Sever je sportovně zaměřená škola, kde tzv. sportovní třídy (C) mají zvýšenou dotaci vyučovacích hodin tělesné výchovy a tudíž mají ve výsledku o jednu hodinu informatiky týdně méně. Právě v hodinách informatiky, kterou mají 6. A, 6. B a 7. B oproti svým spolužákům ze 6. C a 7. C navíc, autor práce Scratch vyučoval.[13]

V hodinách informatiky věnovaným jazyku Scratch měli žáci výše zmíněných tříd možnost spatřit autorovy programy popsané v **kapitole 5**. Tyto naprogramované hry a hlavolamy měli žáci rovněž možnost si zahrát a samozřejmě jim bylo umožněno nahlédnout do jejich kódu. Jak již bylo zmíněno v předchozí kapitole, zjednodušenou verzi Tic-tac-toe (viz **podkapitola 5.1**) rovněž autor práce s žáky v několika po sobě jdoucích hodinách informatiky společně vytvářel, avšak u mnoha žáků se i zjednodušená verze tohoto programu ukázala být jako příliš složitá, a jejich celkový zájem o programovací jazyk Scratch byl nevalný. Naštěstí se však našli i žáci, kteří vyučovanou látku chápali a jejichž zájem o ni byl jasně viditelný. Většina těchto žáků vykazovala celkově vyšší úroveň dovedností v oboru informatiky a celkově jejich průměr i v rámci jiných předmětů byl velmi dobrý, z čehož lze usuzovat, že u nadanějších žáků je zájem o programování vyšší než u žáků průměrných či podprůměrných. Na téma korelace

[13] 7. A měla namísto výuky programovacího jazyku Scratch výuku psaní naslepo prostřednictvím programu ZAV v rámci grantového projektu I-KAP ve spolupráci s Obchodní akademií Hradec Králové.

mezi zájmem o programování a celkovým nadáním žáků však v rámci této práce žádné šetření provedeno nebylo, a v našem případě se tedy jedná pouze o autorův úsudek založený na spolupráci s těmito žáky trvající po dobu jednoho školního roku.

6.2 Základní údaje o dotazníku

Dotazníkové šetření na téma programování a jeho výuce ve škole bylo provedeno prostřednictvím aplikace Google Forms (Formuláře Google). Samotný soubor Google Forms byl pojmenován *Scratch – dotazník*.

Formulář byl strukturován do pěti tematicky souvisejících celků po třech až pěti otázkách. Tyto celky byly nazvány *Všeobecné informace*, *Zkušenosti s programováním*, *Programátorské pojmy*, *Výuka programování* a *Celkový názor na programování*.

Otázek bylo dohromady dvacet. Některé otázky měly uzavřené odpovědi (typicky ano–ne), zatímco některé měly odpovědi otevřené. U těchto otázek s otevřenou odpovědí budou uvedeny pouze některé z povedenějších odpovědí, odpovědi typu „nevím“ zde z pochopitelných důvodů příliš rozebírány nebudou.

Na úvod byly žákům sděleny základní instrukce týkající se dotazníku a jeho tématu. Rovněž byli upozorněni, že pojmy v sekci 3 (mimo jiné i přímo v dotazníku nazvané *Programátorské pojmy*), které se žáci mají pokusit co nejvýstižněji popsat, souvisí s programováním.

6.3 Otázky v dotazníku

1) Všeobecné informace

- **a) Kolik je ti let?**
- **b) Jsi chlapec, nebo dívka?**
- **c) Hlásil(a) ses nebo máš v plánu se hlásit na gymnázium?**
- **d) Víš, co bys v dospělosti chtěl(a) dělat za povolání? Pokud ano, jaké?**

2) Zkušenosti s programováním

- **a) Setkal(a) ses před Scratchem už někdy s programováním?**
- **b) Pokud ano, s jakým programovacím jazykem či jazyky?**
- **c) Pokud jsi programoval(a), jaký byl rozsah tvých dovedností?**

d) Pokud jsi v minulosti něco naprogramoval(a), jaký program považuješ za svůj největší úspěch?

3) Programátorské pojmy

a) Jak chápeš pojem „program“?

b) Jak chápeš pojem „algoritmus“?

c) Jak chápeš pojem „cyklus“?

d) Jak chápeš pojem „větvení“?

4) Výuka programování

a) Myslíš, že by se programování mělo ve školách vyučovat? Pokud ano, mělo by být povinné?

b) Myslíš, že je Scratch vhodné prostředí pro výuku programování?

c) Který či které z programů, co jsme v hodinách vytvářeli, ti utkvěl(y) v paměti nejvíce? Co tě na nich zaujalo, co sis zapamatoval(a) apod.?

d) Vyvolalo v tobě vytváření her ve Scratchi nějaký zájem o programování?

e) Chtěl(a) bys ve výuce programování pokračovat?

5) Celkový názor na programování

a) Myslíš, že je dovednost programovat užitečná?

b) Chtěl(a) bys umět dobře programovat?

c) Chtěl(a) bys vyzkoušet i jiné programovací jazyky?

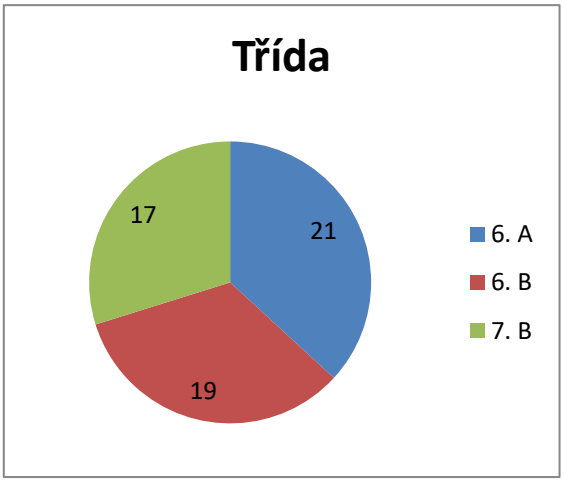
6.4 Všeobecné informace o respondentech

První část dotazníku sloužila k podání základních údajů o dětech samotných a zatím nesouvisela s programováním. Třída, do které daný respondent chodí, byla odvozena z času vyplnění dotazníku.

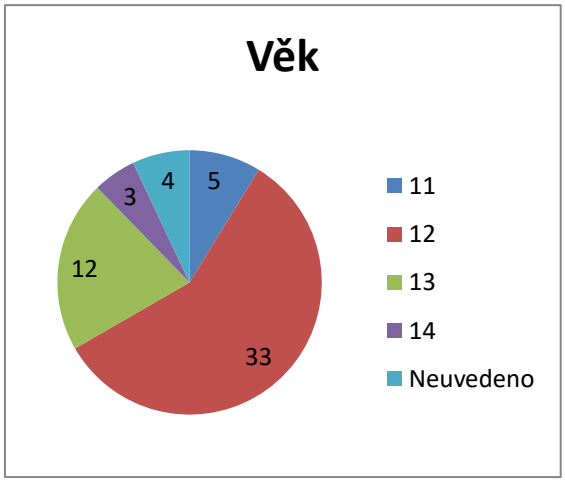
Grafy v této podkapitole byly vytvořeny na základě odpovědí z těchto otázek v dotazníku:

Tabulka 2: Všeobecné informace o respondentech

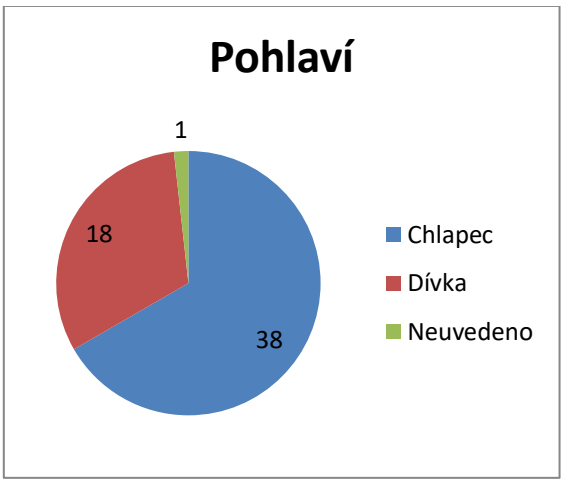
Graf	Znění otázky v dotazníku
Třída	N/A
Věk	Kolik je ti let?
Pohlaví	Jsi chlapec, nebo dívka?
Zájem o studiu na gymnáziu	Hlásil(a) ses nebo máš v plánu se hlásit na gymnázium?



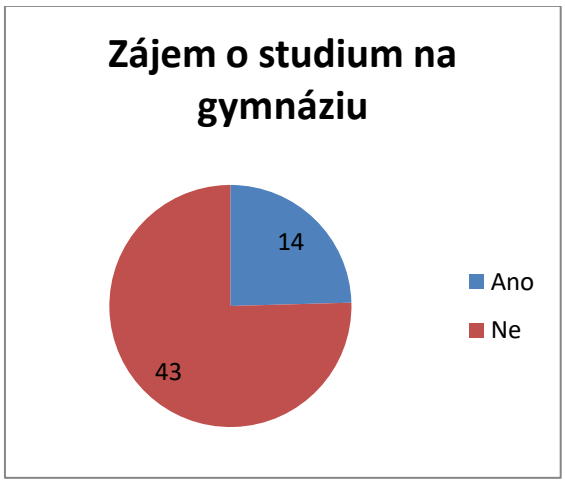
Obrázek 15: Graf – Třída



Obrázek 16: Graf – Věk



Obrázek 17: Graf – Pohlaví



Obrázek 18: Graf – Zájem o gymnázium

Kromě údajů znázorněných grafy výše byli v první části dotazníku žáci také dotazováni, zdali ví, jaké by chtěli v budoucnosti mít povolání. Odpovědi na tuto otázku však byly volné, takže by znázorňování pomocí grafu nebylo příliš vhodné (nemluvě o skutečnosti, že různí žáci v podstatě stejnou odpověď napsali několika různými způsoby a s různými chybami). Pro konkrétní odpovědi viz **příloha 1**.

6.5 Odpovědi z dotazníku – uzavřené otázky

Tato podkapitola obsahuje souhrn odpovědí na otázky již související s programováním a jeho výukou ve škole. Kromě grafů znázorňující vzorky odpovědí na otázky s uzavřenými odpověďmi zde bude i vyhodnocení odpovědí na některé z otázek s odpověďmi otevřenými po jejich přizpůsobení pro znázornění pomocí grafů.

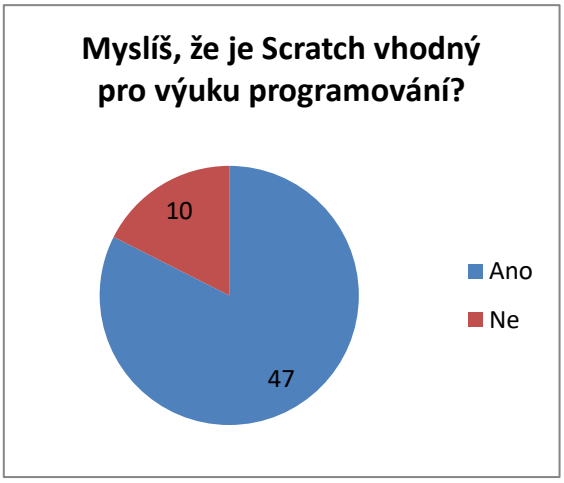
Grafy v této podkapitole byly vytvořeny na základě odpovědí z těchto otázek v dotazníku:

Tabulka 3: Uzavřené otázky

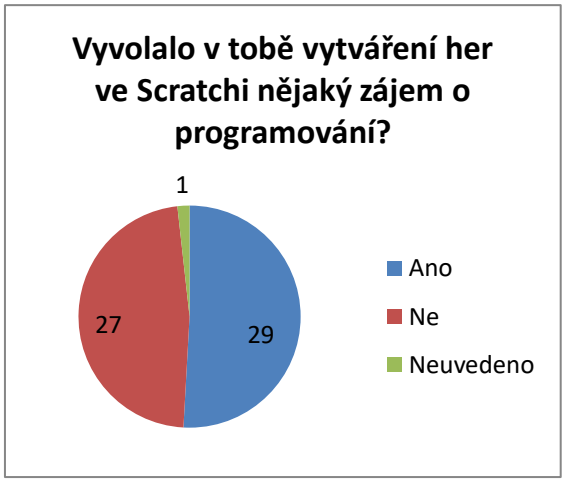
Graf	Znění otázky v dotazníku
Předchozí zkušenosti s programováním	Setkal(a) ses před Scratchem už někdy s programováním?
Názor na výuku programování ve škole	Myslíš, že by se programování mělo ve školách vyučovat? Pokud ano, mělo by být povinné?
Myslíš, že je Scratch vhodný pro výuku programování?	Myslíš, že je Scratch vhodné prostředí pro výuku programování?
Vyvolalo v tobě vytváření her ve Scratchi nějaký zájem o programování?	totéž
Chtěl(a) bys ve výuce programování pokračovat?	totéž
Myslíš, že je dovednost programovat užitečná?	totéž
Chtěl(a) bys umět dobře programovat?	totéž
Chtěl(a) bys vyzkoušet i jiné programovací jazyky?	totéž

Obrázek 19: Graf – Programování – zkušenosti

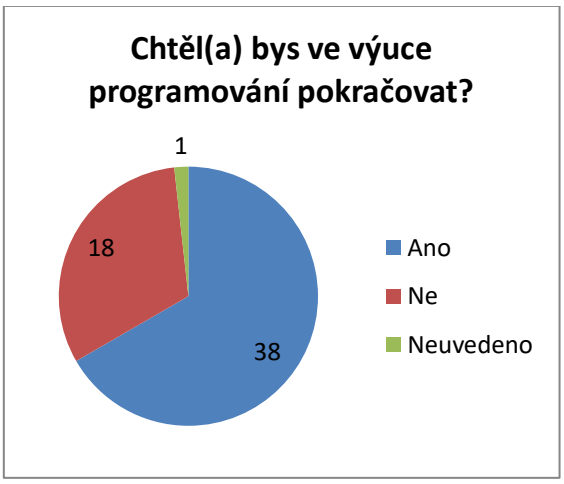
Obrázek 20: Graf – Programování – výuka



Obrázek 21: Graf – Scratch ve výuce

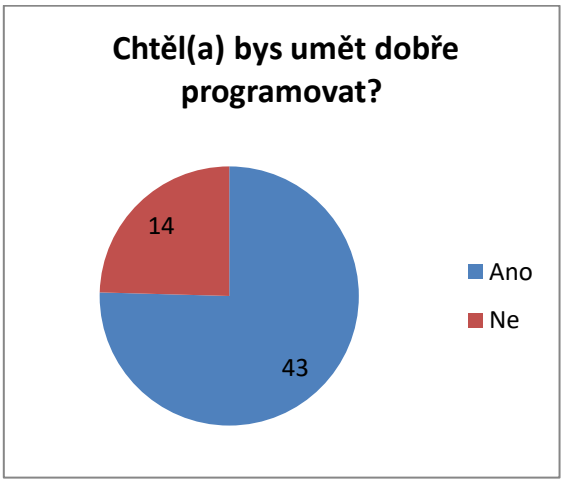


Obrázek 22: Graf – hry a zájem o programování

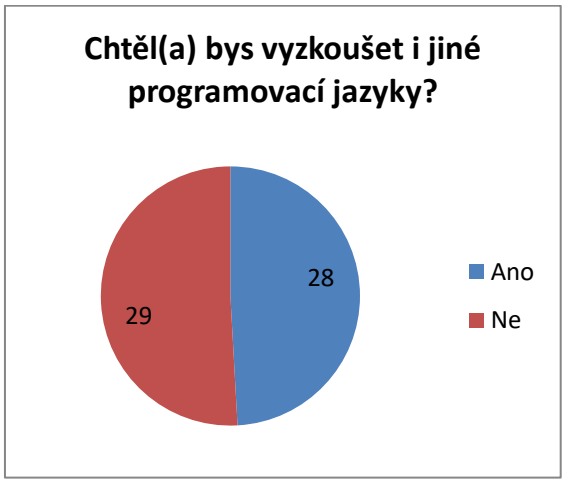


Obrázek 23: Graf – Pokračování ve výuce progr.

Obrázek 24: Graf – Užitečnost programování



Obrázek 25: Graf – Chuť umět programovat



Obrázek 26: Graf – Jiné programovací jazyky

Z grafů výše je patrné, že žáci povětšinou shledávají programování užitečnou dovedností, a že se domnívají, že programování by se mělo na školách vyučovat, avšak obvykle pouze jako volitelný předmět. Žáci se také domnívají, že je Scratch vhodné prostředí pro výuku programování, alespoň tedy jeho základů. Mnozí by ve

výuce programování i chtěli pokračovat, a stejně tak by chtěli umět dobře programovat. Avšak zájem o jiné programovací jazyky byl smíšený. Stejně tak byly smíšené odpovědi na poměrně podstatnou otázku, a to, zdali vytváření her ve Scratchi vyvolalo v žácích zájem o programování – 29 žáků zvolilo „Ano“, 27 „Ne“ a 1 neodpověděl.

Kromě odpovědí v grafech výše dotazník obsahoval ještě otevřenou otázku, která následovala po otázce *2.a) Setkal(a) ses před Scratchem už někdy s programováním?*, ve znění *2.b) Pokud ano, s jakým programovacím jazykem či jazyky?* Drtivá většina žáků na tuto otázku neodpovídala, odpovídala negativně, nebo znění otázky nepochopila, avšak našli se tři respondenti, jejichž odpovědi byly pozitivní: jeden odpověděl "C#", další "Python" a poslední "CMD, netcat, scratch, google" (s výjimkou Scratche se sice nejedná o programovací jazyky, ale stále se v porovnání s ostatními jedná o nadprůměrnou odpověď).

6.6 Vyhodnocení dotazníku – otevřené otázky ze sekce 3

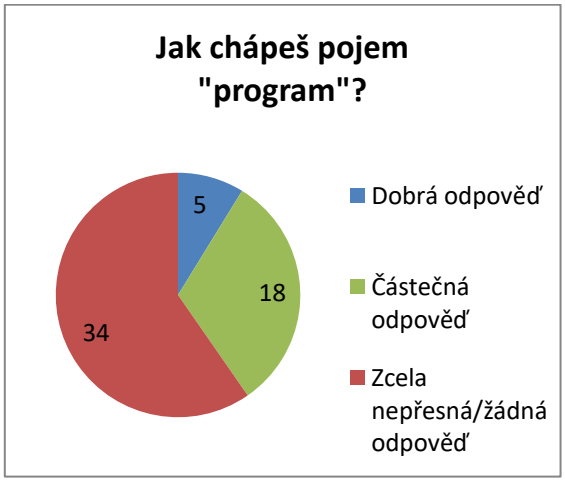
Tato podkapitola shrnuje vyhodnocení zbylých otázek obsažených v dotazníku, tedy těch, které měly otevřené odpovědi. Z toho důvodu jsou grafy, které následují, spíše ilustrační a slouží k hrubému shrnutí odpovědí dotazovaných žáků. Tyto grafy byly vytvořeny na základě odpovědí konkrétně na tyto otázky:

Tabulka 4: Otevřené otázky ze sekce 3

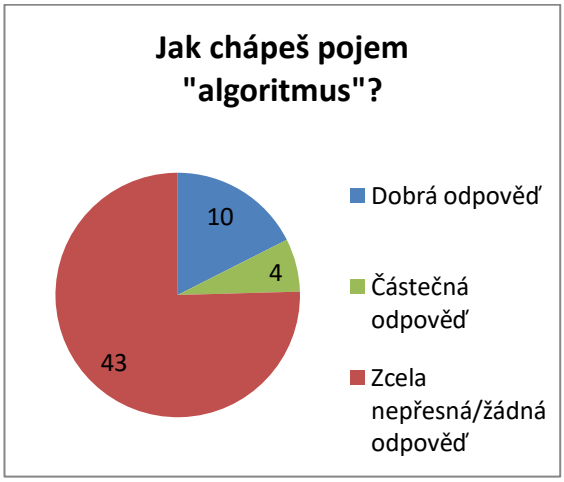
Graf	**Znění otázky v dotazníku**
Program	Jak chápeš pojem „program“?
Algoritmus	Jak chápeš pojem „algoritmus“?
Cyklus	Jak chápeš pojem „cyklus“?
Větvení	Jak chápeš pojem „větvení“?

Aby bylo možné grafy vytvořit, byly odpovědi na otázky na pojmy související s programováním spojeny do několika kategorií podle jejich výstižnosti. Použité kategorie jsou „Dobrá odpověď“, „Částečná odpověď“ a „Zcela nepřesná/žádná odpověď“. Varianta, že žádná odpověď nebyla uvedena, byla u těchto otázek poměrně častá – je však nutné brát v úvahu, že to ne vždy muselo znamenat, že daný respondent nezná odpověď (v takových případech obvykle rovněž mohli uvést „nevím“ apod., což mnozí dělali), a připustit, že se některým žákům jednoduše nechtělo odpovídat na otázky s otevřenou odpovědí, jelikož to již

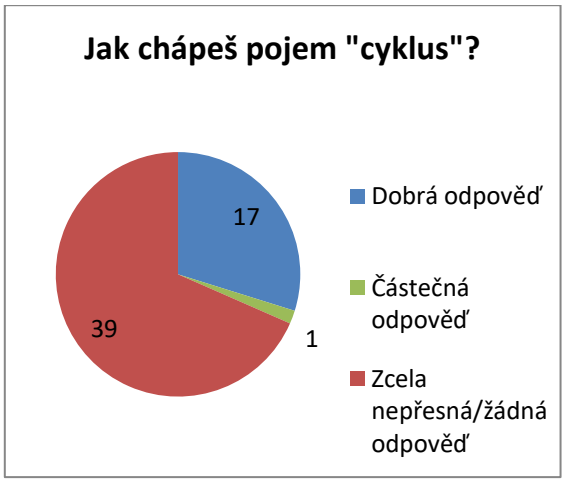
vyžaduje o dost vyšší míru vykonaného úsilí než kliknutí na „Ano“ nebo „Ne“. U některých jedinců bylo z jejich odpovědi rovněž patrné, že si zadání otázky ani nepřečetli, nebo že zcela nedávali pozor při zadávání úvodních instrukcí – např. odpovědi zcela nesouvisející s programováním byly zvláště časté u otázky *3.d) Jak chápeš pojem „větvení“*? Konkrétní odpovědi na otázky v podobě .xlsx souboru automaticky vygenerovaného prostřednictvím Google Forms jsou k nahlédnutí v **příloze 1**, a odpovědi po roztřídění do kategorií uvedených výše jsou pak k nahlédnutí v **příloze 2**. Tyto roztříděné odpovědi jsou uvedeny ve znění a s pravopisnými a gramatickými chybami tak, jak byly vyplněny v dotazníku přímo žáky. V případě, že nebyla uvedena žádná odpověď, nebo že byla uvedena odpověď typu „nevím“ nebo „nedokážu vysvětlit“, daná odpověď byla umístěna do sloupce „Zcela nepřesná/žádná odpověď“ jako <bez odpovědi>.



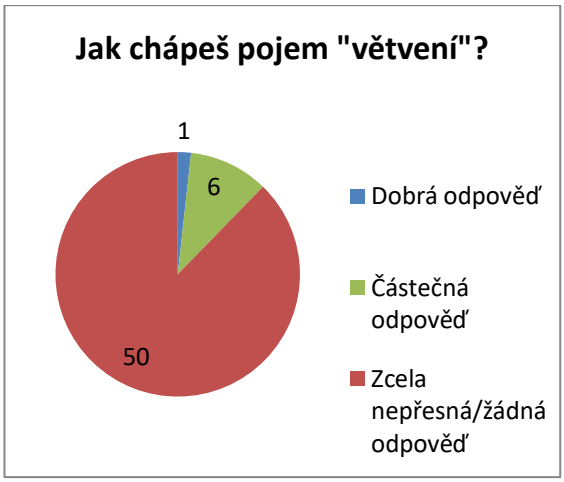
Obrázek 27: Graf – Pojem „program“



Obrázek 28: Pojem „algoritmus“



Obrázek 29: Pojem „cyklus“



Obrázek 30: Pojem „algoritmus“

Z grafů výše je patrné, že špatné a žádné odpovědi na otevřené otázky ze sekce 3 suverénně převažují. Za nejlepší lze považovat odpovědi na otázku *3.c) Jak chápeš*

pojem „cyklus“?, kde vyšlo 17 dobrých odpovědí a 1 částečná. Středně dopadly otázky *3.b) Jak chápeš pojem „algoritmus“?*, kde bylo 10 dobrých odpovědí a 4 částečné, a *3.a) Jak chápeš pojem „program“?*, kde bylo 5 dobrých odpovědí a pak poměrně vysoký počet odpovědí částečných, kterých bylo 18. Nakonec nejhorší vzorek odpovědí byl u otázky *3.d) Jak chápeš pojem „větvení“?*, kde byla jako dobrá uznána jediná odpověď. Částečných odpovědí pak bylo 6.

Závěr

Tato diplomová práce se z různých úhlů pohledu zabývala deskovými hrami a hlavolamy a jejich využitím ve výuce algoritmizace a programování. Cíle práce, kterými bylo zvážit možnosti využití deskových her a hlavolamů ve výuce programování, naprogramovat vlastní verze vybraných deskových her a hlavolamů, následně je zapojit do výuky, a na závěr zhodnotit jejich přínos, byly naplněny.

V první kapitole teoretické části práce byl nejprve nabídnut pohled do současné situace ohledně výuky informatiky a programování v rámci českého školství. Byly uvedeny podklady z RVP, bylo podáno shrnutí plánované reformy ve výuce informatiky na českých školách, dále bylo autorem práce podáno zhodnocení reálné situace ve výuce informatiky a na závěr bylo uvedeno několik aplikací určených k motivaci žáků a k prohloubení jejich zájmu o programování.

Druhá kapitola byla stručným představením programovacího jazyka Scratch, který následně hrál podstatnou roli v rámci části praktické. Po všeobecném úvodu byly zmíněny některé z výhod a nevýhod tohoto programovacího jazyka, a následně byla uvedena základní terminologie, kterou Scratch používá.

Ve třetí kapitole byly stručně představeny deskové hry a hlavolamy z pohledu jejich historie a kategorizace. V části zabývající se historií bylo nahlédnuto do historie nejvýznamnějších a nejznámějších deskových her a hlavolamů. Druhá část kapitoly následně poskytnula přehled běžně používaných kategorií, podle nichž se deskové hry a hlavolamy rozlišují, přičemž u každé z těchto kategorií byl uveden jako příklad jeden nebo více relevantních zástupců.

Ve čtvrté a tudíž poslední kapitole teoretické části práce byla podrobněji rozebrána historie a pravidla varianty piškvorek Tic-tac-toe, šachového problému jezdcovy procházky a hlavolamu Hanojské věže. Tyto konkrétní deskové hry a hlavolamy byly autorem práce zvoleny mimo jiné z důvodu, aby byly, pokud možno, blízké žákům, a aby byly vhodné pro platformu programovacího jazyku Scratch, ve kterém tyto tři deskové hry a hlavolamy byly autorem práce vytvořeny v rámci části praktické.

Pátá kapitola, která je první kapitolou praktické části, představila autorem vytvořené programy výše zmíněných deskových her a hlavolamů v programovacím

jazyce Scratch. Programy byly vždy nejprve popsány z pohledu hráče či řešitele hlavolamu, a následně byl poskytnut náhled do přepisu kódu. Tyto programy byly rovněž využity během autorova působení na ZŠ Sever ve školním roce 2018/2019 při výuce programovacího jazyka Scratch v rámci hodin informatiky za účelem motivování žáků a k demonstraci pojmů souvisejícím s programováním. U většiny žáků byla pozorovatelná určitá míra zájmu a motivace, avšak našli se i tací, které se zaujmout nepodařilo. I tak však lze toto využití programů v hodinách hodnotit za přínosné.

Šestá a poslední kapitola je vyhodnocením dotazníkového šetření na téma programování, kterému byli podrobeni žáci ze tří tříd (6. A, 6. B a 7. B) na ZŠ Sever. Kapitola obsahuje seznam otázek z dotazníku, metodiku vyhodnocování, grafy znázorňující výsledky a slovní shrnutí. Z výsledků prezentovaných v této kapitole je patrné, že většina žáků považuje programování za užitečné a s jeho výukou ve škole souhlasí. Bohužel, většina žáků s probíranou látkou měla nemalé problémy, a to jak v rovině teoretické, tak v rovině praktické.

I tak však lze prohlásit, že se deskové hry a hlavolamy při výuce algoritmizace a programování jeví jako vhodný námět. Jelikož se jedná o téma, které je blízké široké vrstvě obyvatelstva, jsou také poměrně univerzální, a rovněž se nabízí široké spektrum jejich využití. Tato práce se soustředila hlavně na programování deskových her a hlavolamů samotných a jejich propojení s výukou, přičemž jejich praktický přínos ve výuce spočíval ve zvýšení zájmu žáků o učivo a snadnější pochopení základních principů algoritmizace a programování.

Seznam použité literatury

[1] Řídicí výbor k Inkluzi (MŠMT) a Národní ústav pro vzdělávání. *Rámcový vzdělávací program pro základní vzdělávání*. MŠMT ČR [online]. Praha: MŠMT, 2017 [cit. 2. 7. 2019]. Dostupné z: http://www.msmt.cz/file/43792/

[2] JAVŮREK, Karel. *České školství čeká revoluce v informatice. Děti se budou učit programovat a logicky myslet*. Connect.cz – Profesionální IT a byznys [online]. ©2018 [cit. 2. 7. 2019]. Dostupné z: https://connect.zive.cz/clanky/ceske-skolstvi-ceka-revoluce-v-informatice-deti-se-budou-ucit-programovat-a-logicky-myslet/sc-320-a-195437/default.aspx

[3] *Česká škola: Na vybraných školách od září startuje velká reforma ve výuce informatiky, za dva roky se má spustit všude*. Česká škola [online]. ©2018 [cit. 2. 7. 2019]. Dostupné z: http://www.ceskaskola.cz/2018/08/na-vybranych-skolach-od-zari-startuje.html

[4] VANÍČEK, Jiří a redakce Živě.cz. *Jak nově učit informatiku na školách: Otázky a odpovědi k nutné změně - 2. kapitola*. Živě.cz – O počítačích, IT a internetu [online]. ©2018 [cit. 3. 7. 2019]. Dostupné z: https://www.zive.cz/clanky/jak-nove-ucit-informatiku-na-skolach-otazky-a-odpovedi-k-nutne-zmene/odpovedi-na-nejdulezitejsi-otazky/sc-3-a-196201-ch-116258/default.aspx

[5] ENDRŠTOVÁ, Michaela. *Nový způsob výuky informatiky nemá děti učit Word a Excel, ale programování a stavění robotů. Učitelé se ho ale bojí*. Hospodářské noviny (IHNED.cz) [online]. ©2018 [cit. 3. 7. 2019] Dostupné z: https://archiv.ihned.cz/c1-66384790-novy-zpusob-vyuky-informatiky-nema-deti-ucit-word-a-excel-ale-programovani-ci-staveni-robotu-zatim-ale-u-ucitelu-budi-spis-obavy

[6] ČÍŽEK, Jakub. *Český CyberCon 2018: Kybernetická gramotnost je tragická. Jsme roky pozadu*. Živě.cz – O počítačích, IT a internetu [online]. ©2018 [cit. 2. 7. 2019]. Dostupné z: https://www.zive.cz/clanky/cesky-cybercon-2018-kyberneticka-gramotnost-je-tragicka-jsme-roky-pozadu/sc-3-a-195283/default.aspx

[7] *Umíme programovat – Procvičujte programování hrou či na příkladech* [online]. ©2019 [cit. 3. 7. 2019]. Dostupné z: https://www.umimeprogramovat.cz/

[8] *Learn*. Hour of Code [online]. ©2019 [cit. 3. 7. 2019]. Dostupné z: https://hourofcode.com/cz/learn

[9] *Scratch - About*. Scratch [online]. ©2019 [cit. 11. 4. 2019] Dostupné z: https://scratch.mit.edu/about

[10] ZAPLETAL, Miloš. *Velká kniha deskových her*. Praha: Mladá fronta, 1991. Volný čas, sv. 6. ISBN 80-204-0188-1.

[11] *The Game of Senet*. Game Cabinet [online]. [cit. 19. 5. 2018]. Dostupné z: http://www.gamecabinet.com/history/Senet.html

[12] Wikipedia contributors. *Backgammon*. Wikipedia, the free encyclopedia [online]. [cit. 21. 5. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Backgammon

[13] Wikipedia contributors. *Go*. Wikipedia, the free encyclopedia [online]. [cit. 21. 5. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Go_(game)

[14] *Hindi and the origins of chess | ChessBase*. Chess News | ChessBase [online]. [cit. 21. 5. 2018] Dostupné z: https://en.chessbase.com/post/hindi-and-the-origins-of-chess

[15] HOOPER, David a Ken WHYLD. *The Oxford Companion to Chess*. 2nd ed. New York: Oxford University Press, 1992. ISBN 0-19-866164-9.

[16] Wikipedia contributors. *Board game*. Wikipedia, the free encyclopedia [online]. [cit. 19. 05. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Board_game

[17] DUFFY, Owen. *Board games' golden age: sociable, brilliant and driven by the internet*. The Guardian [online]. 2014 [cit. 4. 4. 2019]. Dostupné z: https://www.theguardian.com/technology/2014/nov/25/board-games-internet-playstation-xbox

[18] *Chronological History of Puzzles: A Timeline by SiamMandalay*. Siam Mandalay [online]. ©2019 [cit. 18. 7. 2019]. Dostupné z: https://www.siammandalay.com/blogs/puzzles/95858310-chronological-history-of-puzzles-a-timeline

[19] SAWARD, Jeff. *Mazes or Labyrinths... What's the difference & what types are there?*. Labyrinthos [online]. ©2018. [cit. 18. 7. 2019]. Dostupné z: http://www.labyrinthos.net/Labyrinth%20Typology.pdf

[20] LEWIS, Hazel. *The Stomachion – The world's oldest maths puzzle*. Maths Careers [online]. ©2019. [cit. 07. 12. 2019]. Dostupné z:

https://www.mathscareers.org.uk/article/the-stomachion-the-worlds-oldest-maths-puzzle/

[21] *Jezdcova procházka*. Algoritmy.net [online]. [cit. 14. 12. 2019]. Dostupné z: https://www.algoritmy.net/article/1361/Jezdcova-prochazka

[22] WILLIAMS, Anne D. *History of Puzzles*. PuzzleWarehouse.com [online]. ©2019 [cit. 6. 7. 2019]. Dostupné z: https://www.puzzlewarehouse.com/history-of-puzzles/

[23] VEJMOLA, Stanislav. *Jak vyrobit a vyřešit hlavolamy: domino a polymina, tangramy, drátové hlavolamy, hanojské věže, čínské kroužky, ukládací hlavolamy, Rubikovy kostky: 52 úloh*. Praha: Grada, 2007. ISBN 978-80-247-2013-5.

[24] KOTTOVÁ, Anna. *Hrají si s ní děti i dospělí, Rubikova kostka oslavila 40 let*. iROZHLAS [online]. 2015 [cit. 7. 12. 2019]. Dostupné z: https://www.irozhlas.cz/clovek/hraji-si-s-ni-deti-i-dospeli-rubikova-kostka-oslavila-40-narozeniny_201501310933_akottova2

[25] Wikipedia contributors. *Puzzle*. Wikipedia, the free encyclopedia [online]. [cit. 20. 5. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Puzzle

[26] Wikipedia contributors. *Tic-tac-toe*. Wikipedia, the free encyclopedia [online]. [cit. 21. 5. 2018]. Dostupné z https://en.wikipedia.org/wiki/Tic-tac-toe

[27] Wikipedia contributors. *Knight's Tour*. Wikipedia, the free encyclopedia [online]. [cit. 21. 5. 2018]. Dostupné z https://en.wikipedia.org/wiki/Tic-tac-toe

[28] Wikipedia contributors. *Tower of Hanoi*. Wikipedia, the free encyclopedia [online]. [cit. 21. 5. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Tower_of_Hanoi

Zdroje obrázků

Obrázek 1: vlastní tvorba

Obrázek 2: vlastní tvorba

Obrázek 3: *Jezdcova procházka*. Algoritmy.net [online]. [staženo 15.5.2018]. Adresa obrázku: https://www.algoritmy.net/image/id/39912

Obrázek 4: Wikipedia contributors. *Knight's tour*. Wikipedia, the free encyclopedia [online]. Adresa obrázku: https://en.wikipedia.org/wiki/Knight%27s_tour#/media/File:Knight's_graph_showing_number_of_possible_moves.svg

Obrázek 5: STORER, James A. *Towers Of Hanoi*. Jim Storer Puzzles [online]. [staženo 14.12.2019]. Adresa obrázku: https://www.cs.brandeis.edu/~storer/JimPuzzles/MANIP/TowersOfHanoi/TowersOfHanoiPhoto.jpg

Obrázky 6–14: vlastní tvorba – screenshoty z vytvořených Scratch programů

Obrázky 15–30: vlastní tvorba – grafy vytvořené v MS Word

Seznam příloh

Příloha 1: Scratch dotazník - odpovědi.xlsx

Příloha 2: Scratch dotazník - sekce 3.xlsx

Příloha 3: Tic-tac-toe.sb3

Příloha 4: Knight's Tour 5x5.sb3

Příloha 5: Tower of Hanoi.sb3