ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

# POROVNÁNÍ SOFTWAROVÉHO A UNPLUGGED PROSTŘEDÍ PRO VÝUKU BLOKOVÉHO PROGRAMOVÁNÍ

BAKALÁŘSKÁ PRÁCE

**Jana Kotrbatá**

*Učitelství pro základní školy, obor Učitelství informatiky pro základní školy*

Vedoucí práce: Mgr. Zdeněk Lomička

**Plzeň 2020**

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 30. června 2020

.....

vlastnoruční podpis

Ráda bych poděkoval Mgr. Zdeňkovi Lomičkovi za odborné konzultace, cenné rady a připomínky při tvorbě bakalářské práce.

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta pedagogická

Akademický rok: 2018/2019

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Jana KOTRBATÁ** |
| Osobní číslo: | **P17B0368P** |
| Studijní program: | **B1001 Přírodovědná studia** |
| Studijní obor: | **Informatika se zaměřením na vzdělávání** |
| Téma práce: | **Porovnání softwarového a unplugged prostředí pro výuku blokového programování.** |
| Zadávající katedra: | **Katedra výpočetní a didaktické techniky** |

## Zásady pro vypracování

1. Porovnejte softwarové a unplugged prostředí pro výuku blokového programování a jejich charakteristické vlastnosti.
2. Představte vhodné zástupce softwarového a unplugged prostředí pro výuku blokového programování.
3. Zmapujte aktuální stav využití softwarového a unplugged prostředí pro výuku blokového programování na zvolených školách.

Rozsah bakalářské práce: **30 – 50**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

BACKUS, J. W., BAUER, F. L., GREEN, J., KATZ, C., MCCARTHY, J., PERLIS, A. J., RUTISHAUSER, H., SAMELSON, K., VAUQUOIS, B., WEGSTEIN, J. H., WIJNGAARDEN, A. van, WOODGER, M. Revised Report on the Algorithmic Language Algol 60. [online]. Dostupné z: https://www.masswerk.at/algol60/report.htm

BAU, D., GRAY, J., KELLEHER, C., SHELDON, J., TURBAK, F. Learnable Programming: Blocks and Beyond. Communications of the ACM, Vol. 60 No. 6, Pages 72-80. DOI:10.1145/3015455. [cit. 13.06.2019]. Dostupné z: https://m-cacm.acm.org/magazines/2017/6/217743-learnable-programming/fulltext?mobile=true

MOHAMAD, S. N. H., PATEL, A., LATIH, R., QASSIM, Q. S., LIU, N., TEW, Y. Block-based programming approach: challenges and benefits. In: Proceedings of the 2011 International Conference on Electrical Engineering and Informatics [online]. IEEE, 2011, 2011, s. 1-5 [cit. 2019-06-13]. DOI: 10.1109/ICEEI.2011.6021507. ISBN 978-1-4577-0753-7. Dostupné z: http://ieeexplore.ieee.org/document/6021507/

GROVER, S., JACKIW, N., LUNDH, P. Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. Computer Science Education [online]. 2019, 29(2-3), 106-135 [cit. 2019-06-13]. DOI: 10.1080/08993408.2019.1568955. ISSN 0899-3408. Dostupné z: https://www.tandfonline.com/doi/full/10.1080/08993408.2019.1568955

Vedoucí bakalářské práce: **Mgr. Zdeněk Lomička**
Katedra výpočetní a didaktické techniky

Datum zadání bakalářské práce: **13. června 2019**
Termín odevzdání bakalářské práce: **30. června 2020**

Západočeská univerzita v Plzni
L.S.
Fakulta pedagogická

**RNDr. Miroslav Randa, Ph.D.**
děkan

**PhDr. Tomáš Jakeš, Ph.D.**
vedoucí katedry

V Plzni dne 25. června 2019

# OBSAH

# ÚVOD

Tato bakalářská práce pojednává o vývojových prostředích blokového programování. Téma jsem si zvolila mimo jiné proto, že si myslím, že je tato forma programování velmi výhodná pro výuku na základní škole, zejména z důvodu jeho jednoduchosti. Navíc jsem samotné blokové programování znala spíše okrajově a chtěla jsem se o něm dozvědět více. V souvislosti s tématem a zásadami vypracování jsem pak práci využila i pro porovnání rozdílu mezi softwarovým a unplugged prostředím, aby se i čtenář mohl seznámit s jejich úskalími a přednostmi, a zároveň ke zmapování situace na základních školách.

Obsah této práce se skládá ze čtyř hlavních kapitol, ve kterých se snažím čtenáře seznámit s problematikou dnešní doby, a to výuky programování na základní škole. Snažím se zde vysvětlit, co a proč je vhodné se v dnešní době učit. V první kapitole s názvem Algoritmizace zmiňuji, co je to algoritmus a jak se tvoří jeho schematické zobrazení.

V kapitole Programování se nachází rychlé seznámení s tím, co je to programování, od kdy je tento pojem známý a k čemu slouží. Tento pojem zde popisuji proto, že v dnešní době nejde o to, zda někdo umí programovat v nějakém programovacím jazyce, ale zejména o to myslet – logicky, v souvislostech, v návazných krocích, což souvisí s vytvářením algoritmů, a tedy s algoritmickým myšlením. Dále se zde snažím vysvětlit, proč je důležité se učit programovat a nahlížím zde do návrhu nového rámcově vzdělávacího programu.

Další kapitola se zabývá blokovým programováním. Vysvětluji zde jeho charakteristiku, a jaký je rozdíl mezi softwarovým a unplugged prostředím, které takový způsob programování využívají. Následně popisuji vybrané zástupce těchto prostředí.

V poslední kapitole popisuji zjišťování stavu na základních školách, pro které jsem si zvolila elektronickou formu dotazníku. V této kapitole se snažím vysvětlit zamýšlenou logiku postupu respondenta při vyplňování, popisuji zde přípravu dotazníku, jeho rozeslání, sběr odpovědí a jejich následné vyhodnocení.

# 1 ALGORITMIZACE

Dnes často používané slovo algoritmus existuje již od 9. století, kdy se objevilo v názvu knihy postupů pro počítání s čísly, kterou napsal arabský matematik Muhammad ibn Músá al Chwárizmí. Pojem algoritmus se často spojuje hlavně s výpočetní technikou a spousta lidí pod tímto slovem vidí nejasný obraz spojený s programováním, jehož smysl je pro běžného člověka téměř nepochopitelný. A přitom se s algoritmy setkáváme každý den. Každý máme nějaký algoritmus proto, abychom se dostali z domu do práce nebo školy. Algoritmy používáme i při vaření kávy, čaje či přípravě různých pokrmů. Algoritmus bychom totiž mohli charakterizovat jako jakýsi postup či recept. V knize Informatika tvrdí Brookshear, Smith a Brylow, že *„Algoritmus je uspořádaná sada jednoznačných a proveditelných kroků, která definuje konečný proces.“* [1]

Ve snaze o řešení nějaké situace či problému je třeba, aby někdo nejprve vymyslel postup takového řešení, tedy algoritmus, a následně jej přepsal do kódu, který počítač umí přečíst, tedy počítačového programu.

Aby se skutečně jednalo o algoritmus, musí mít takový postup následující vlastnosti:

**Konečnost:** Algoritmus musí v konečném počtu kroků dospět do určitého konečného stavu a přinést výsledek v rozumném čase.

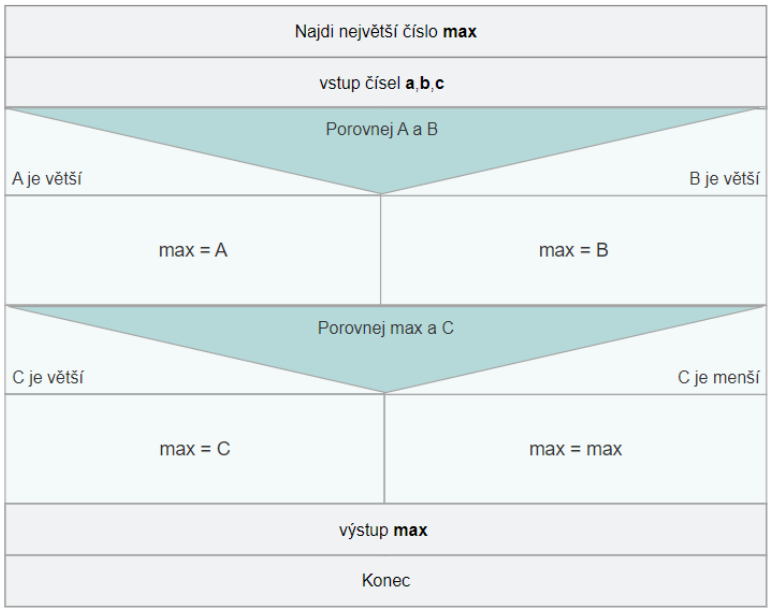
**Hromadnost** (nebo také **obecnost** či **opakovatelnost**) algoritmu znamená, že nemá řešit konkrétní problém, ale obecně podobné problémy. Například při dělení dvou čísel, algoritmus nemá řešit pouze příklad dvou konkrétních čísel, ale je možné jej opakovaně využít i na jiná čísla.

**Determinovanost:** Algoritmus tvoří různé stavy a tyto stavy jsou tvořeny zpracovanými daty v jeho krocích, které na sebe navazují. Pro stejné vstupy jsou pokaždé stejné výstupy.

**Elementárnost** (též **jednoduchost** či **srozumitelnost**) popisu: Algoritmus je tvořen sledem konečného počtu instrukcí, které jsou pro vykonavatele srozumitelné a jasné a může podle nich postupovat.

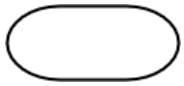
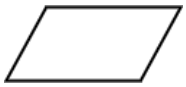
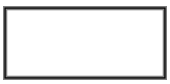
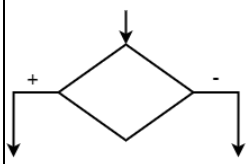
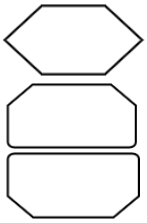
**Rezultativnost:** Kroky algoritmu vedou vždy ke konečnému stavu, výsledku. [2][3]

Stejný postup, jako pro běžné opakující se činnosti platí i pro počítačové programy. Je třeba vědět, co má program dělat, a následně jeho tuto činnost srozumitelně popsat. Algoritmus pro program může být popsán graficky nebo textově. Pro lidské chápání je grafické vyjádření algoritmu mnohem přehlednější, protože je vyjádřeno jednoznačnými a srozumitelnými symboly. Často se používají strukturogramy, vývojové diagramy a UML, ale i jiné způsoby. Strukturogram je úspornější řešení vývojového diagramu bez značek, je kombinací grafického a textového znázornění. UML je grafický jazyk pro navrhování nebo vizualizaci programů a nejčastěji se používá pro databáze. Na obrázku 1 je znázorněn algoritmus pomocí strukturogramu zjišťující nejvyšší vstupní číslo. Porovnávají se zde tři vstupní hodnoty, proměnné **a**, **b**, **c**, pomocí sledu podmínek. První zjišťuje, zda je větší **a** nebo **b,** dále se vyhodnotí nejvyšší hodnota a ukládá se do proměnné s názvem **max**. Proměnná **max** vstupuje do podmínky, kde se porovnává s proměnou **c.** Následně se vyhodnotí, která hodnota je nejvyšší, a algoritmus se ukončí.




Obrázek 1 Strukturogram

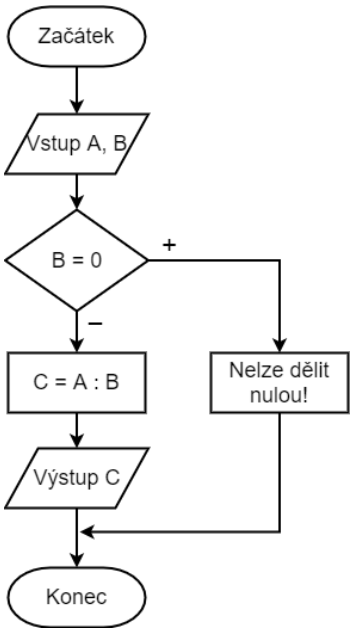
Vývojové diagramy se skládají z různých značek a popisků. Vývojový diagram, jak popisuje ve své knize paní inženýrka Pšeníčková [2], musí obsahovat začátek a konec. Diagram popisuje, jak bude daný kód vypadat, a tedy znázorňuje existující možnosti v programování. Na základě popisu či nákresu nějaké činnosti může programátor vytvořit program, který zohlední zadané požadavky a vyhoví vstupním podmínkám. A právě převod původní myšlenky z jejího popisu do výsledného programu vyžaduje znalost programování. V tabulce 1 je možné vidět jednotlivé značky vývojového diagramu a jejich funkce.

| | Popis | Značka |
|---|---|---|
| Začátek a konec | Bez nastavení začátku by algoritmus nemohl proběhnout a bez konce bychom se dostali k zacyklení programu, to znamená, že by vykonával danou úlohu donekonečna. |  |
| Načtení dat a zobrazení výstupu | Aby znázorňovaný algoritmus zpracoval data, potřebujeme zobrazit jejich načtení. A nejen k tomu slouží značka kosodélníku, ale i k zobrazení výstupu zpracovaných dat. |  |
| Zpracování hodnot | Pro zpracování vstupních hodnot, jako například sečtení dvou čísel, se používá značka pro zpracování. |  |
| Podmínka | Při běhu programu potřebujeme, aby algoritmus řešil podmínky. K tomu nám slouží značka pro podmínku, která nám rozděluje program na dvě větve. Na větev, kdy je podmínka splněna (značena např. ano, 1, true, zde se znaménkem „+“) a na větev, kdy splněna není (např. ne, 0, false, zde se znaménkem „-“). |  |
| Cykly | Pokud potřebujeme v programu nějakou činnost opakovat, tak potřebujeme znázornit cyklus. Existuje několik cyklů: cyklus se známým počtem opakování, cyklus s podmínkou na začátku a cyklus s podmínkou na konci. |  |
| Spojka | Jestliže je vývojový diagram příliš dlouhý a nevejde se vcelku na jednu stránku, můžeme jej rozdělit a v místě rozdělení zakreslit spojovací značku. |  |
| Podprogram | Dále pro přehlednost můžeme použít značku znázorňující podprogram, který je možný použít vícekrát a tím zamezit redundanci. |  |

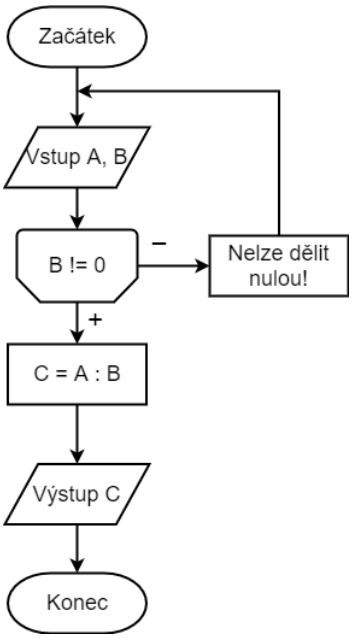
Tabulka 1 Značky vývojového diagramusss

Na obrázku 2 můžeme vidět jednoduchý vývojový diagram pro dělení dvou čísel A a B, kde je vyznačený začátek algoritmu pomocí příslušné značky, ze které se vstupuje do podmínky zajišťující, aby nedošlo k dělení nulou. To znamená, že se zde řeší výpočet A : B. Na tomto diagramu je vidět, že pokud vstupní hodnota B se rovná nule, tak se provede větev se znaménkem „+“, kde je výstup „Nulou nelze dělit“, a dále se program ukončí. [2]

Druhý diagram na obrázku 3 znázorňuje řešení stejného výpočtu pomocí cyklu s podmínkou na konci. Oproti předešlému diagramu se zde algoritmus neukončí, když vstupní hodnota B je rovna 0, ale opakuje se zde načítání hodnot, dokud se B nerovná 0.




Obrázek 3 Algoritmus pro výpočet A:B,




Obrázek 2 Algoritmus pro výpočet A:B s opakováním

# 2 PROGRAMOVÁNÍ

Nacházíme se v době digitální revoluce, kde se snažíme veškeré procesy práce urychlit a díky rozvoji moderních technologií se nám to daří zrealizovat. Nedílnou součástí vytváření a zdokonalování nových technologií je právě programování. Také na současném trhu práce můžeme vidět, že obsahuje mnoho nabídek související s programováním. A proto si myslím, že je třeba se na programování a rozvíjení algoritmického myšlení zaměřit už na základní škole. Žák může kromě zkušeností získat i nový koníček, jehož náplň využije nejen v běžném životě, ale i v průběhu jeho studia na střední škole, kde je často programování nedílnou součástí informatiky. Algoritmické myšlení nepřináší přínosy pouze ve výuce programování, ale ve veškerých oblastech života, kde je třeba se rozhodovat a zvolit správný postup. Toto myšlení nepoužívá pouze programátor, ale využívá jej i zedník, který nezačne nejprve skládat cihly bez toho, aniž by si vytvořil pojidlo a umisťoval jej mezi ně. Nebo kuchař, který si nejprve nakoupí suroviny, a poté vaří podle promyšleného postupu či receptu.

Mezi nejstarší způsob programování řadíme udávání instrukcí stroji pomocí děrných štítků. První myšlenka děrných štítků se objevila začátkem 18. století, v podání Jeana Falcona. Tyto děrné štítky zrealizoval a používal je pro programování tkalcovských stavů. Tkalcovské stavy se neuchytily, ale touto myšlenkou se dále zabývali i jiní vynálezci, ale dlouho se nedařilo nikomu z nich uspět. Začátky programovatelných strojů byly velmi krušné hlavně kvůli tomu, že se stroje stávaly konkurencí v ruční výrobě. Na počátku 19. století se ve Francii dosáhl značného úspěchu tkalcovský stav od vynálezce Josepha M. Jacquarda. V té době bylo těchto strojů v provozu kolem 11 000. Jacquardův tkalcovský stav uměl tkát látky podle předem připravených, po sobě jdoucích děrných štítků. Tyto děrné štítky byly spojené v jeden pás, který obsahoval vzor tkaniny pomocí dírek. Pás působil na jehly, ovládající proces tkaní. Tento vynález ovlivnil řadu lidí, kteří tuto myšlenku následně využili u svých vynálezů jako například tvůrce návrhu prvního mechanického počítače Charles Babbage. Jeho návrh vznikl pár let po Jacquardově tkalcovském stavu, ale nebyl zrealizován. Přestože se Babbageho mechanický počítač ve společnosti neujal, jeho obdivovatelka a přítelkyně, označovaná za první programátorku, hraběnka Ada Lovelance, tento stroj nejen docenila, ale rozvinula jeho myšlenku o čtyři poznámky, které ovlivnily vývoj počítačů.

První poznámka předběhla dobu o 100 let, protože pojednávala o stroji, který má možnost být naprogramován či přeprogramován k různým úlohám. Hraběnčiny matematické

a analytické schopnosti jí pomohly vymyslet konkrétní funkce pro zpracování námořních tabulek. V druhé poznámce je popsaný analytický stroj zaměřený na zpracování a vyhodnocování různých informací, které nemusí být pouze matematické a číselné, ale mohou být vyjádřeny jakýmikoliv znaky jako například slovy či hudbou. Ve třetí poznámce Ada popisuje algoritmus počítačového programu. Sepsala zde sérii jednotlivých operací a sestavila schéma, které znázorňovalo, jak úlohu zakódovat do stroje. Zavedla zde již pojmy jako podprogram, cyklus, knihovna, které se v dnešní době běžně při programování používají. Je označována za první programátorku na světě, protože uměla tyto pojmy zakreslit a popsat. Ve čtvrté poznámce zpracovala vizualizaci zadávání programu do počítače pomocí nákresů a tabulek. [4][5]

Ke konci 19. století byl vytvořen stroj IBM, jež sloužil pro sčítání lidu ve Spojených státech amerických. Stroj byl vytvořen německým vynálezcem se jménem Herman Hollerith. IBM četl data pomocí děrných štítků. [4]

Pro naprogramování moderních zařízení nepotřebujeme v dnešní době děrné štítky, ale některá zařízení jsou vytvořena tak, že se dají programovat pomocí rozpoznávání barev, čar, obrazů, zvuků či různých fyzikálních veličin. Nejčastěji se pro programování využívají různé programovací jazyky.

## 2.1 PROGRAMOVACÍ JAZYK

Programovací jazyk si můžeme představit jako jakýkoliv nám známý jazyk pro verbální komunikaci. Stejně jako cizí jazyky mohou mít mezi sebou jistou podobnost, tak i jednotlivé programovací jazyky si mohou být velmi podobné. Čím se jazyky liší, je jejich syntaxe a sémantika. Syntaxe programovacího jazyka je souhrn pravidel, určující znaky, jejich uspořádání do skupin a struktury celého programu. Sémantika neboli takzvaný slovník daného jazyka se zabývá významem slov a znaků podle definovaných syntaktických pravidel. [6]

Oproti verbálním jazykům je programovací jazyk vytvořen uměle pro komunikaci s počítačem. Počítači dáváme instrukce, které tvoří počítačový program. Vytváření počítačového programu neboli programování vychází z procesu algoritmizace.

## 2.2 DĚLENÍ PROGRAMOVACÍCH JAZYKŮ

Pokud již máme vytvořený zápis algoritmu, například formou strukturogramu nebo vývojového diagramu, je vhodné zapřemýšlet nad volbou vhodného jazyka. Záleží na tom, zda chceme použít takový, ze kterého vycházejí méně chybové a rychlejší programy, ale je složitější na vývoj, nebo zda chceme ušetřit čas a využít jazyk, jenž je lépe čitelný a je k němu připraveno mnoho knihoven.

Programovací jazyky se dají dělit podle několika kritérií. Nejobecnější dělení je podle míry abstrakce, kde se dělí na vyšší a nižší programovací jazyky. Vyšší programovací jazyky se dělí na procedurální a neprocedurální. Procedurální ještě obsahují dělení na strukturované a objektově orientované. Neprocedurální se člení na funkcionální a logické. A další dělení je podle překladu na strojový kód na kompilované a interpretované. [6]

### 2.2.1 VYŠŠÍ A NIŽŠÍ

Vyšší programovací jazyk oproti nižšímu by měl být pro člověka lépe čitelný, srozumitelný a kratší. Počítač ale rozumí nižšími programovací jazykům, a aby porozuměl těm vyšším, je potřeba tyto vyšší jazyky překládat nebo jiným slovem kompilovat do nižších jazyků. Mezi vyšší programovací jazyky patří téměř všechny, kromě jazyka symbolických adres Assembler a strojového kódu.

U nižších programovacích jazyků je kladen důraz na přesnou syntaxi. Program vytvořený nižším jazykem obsahuje instrukce, které jsou příkazy procesoru, a proto programy

vytvořeny těmito jazyky jsou nepřenositelné na jiný procesor. I přes tuto nevýhodu mohou být programy velmi efektivní. [6]

### 2.2.2 PROCEDURÁLNÍ A NEPROCEDURÁLNÍ

Procedurální (imperativní) jsou jazyky, které přesně v krocích popisují to, co se má stát a jak se to má stát. Tento typ jazyka jsou například Pascal, FORTRAN, BASIC, C, a tedy nízko úrovňové systémové jazyky. Od procedurálních jazyků se odvíjí také objektově orientované jazyky, které jsou založeny na imperativních principech, ale navíc doplněné o abstrakce využití objektů. Tato kategorie je dnes nejrozšířenější a největší zástupci jsou Java, JavaScript, C++, C#.

Neprocedurální jazyky specifikují to, co se má stát, ale nespecifikují, jak se to má stát. Tyto jazyky se také nazývají funkcionální, protože se skládají z funkcí. Tento typ je méně populární, ale má svá využití, například v matematice. Některé principy funkcionálních jazyků se přenesly i do objektově orientovaných jazyků. [6][7]

### 2.2.3 KOMPILOVANÉ A INTERPRETOVANÉ

Kompilované jazyky musí být před spuštěním přeloženy do strojového kódu (nižšího jazyka). To vyústí v lepší optimalizaci na daný hardware a tím vyšší výkon. Výhodou toho je, že ještě před spuštěním nám překladač může dát mnoho informací o správnosti kódu, jak syntaktické, tak v některých případech i logické. Dále jsou kompilované jazyky většinou silně typované například jazyk C#. Typovaný jazyk znamená, že se zakládá na datových typech, které přesně určují, jakého typu jsou používané proměnné. Datové typy jsou například: string (textový řetězec), integer (celé číslo), float (desetinné číslo), boolean (logická hodnota) a další. Ve spojení s rychlostí a kontrolou chyb tyto jazyky tvoří spolehlivé prostředí pro rychlé systémové programy, spolehlivé informační systémy a mobilní aplikace. Nevýhodou je nepřenositelnost a vyšší bariéra pro začátečníky oproti interpretovaným a jazykům.

Interpretované jazyky jsou překládány až za chodu programu. A na rozdíl od kompilovaných se nemusí předem překládat do nižšího jazyka, což může zpomalovat chod programu. Hlavním využitím interpretovaných jazyků se nachází na webu, jak na straně klienta např. JavaScript, tak na straně serveru např. PHP. Dále pak se nachází ve skriptech a jednodušších programech (Python). Velkou výhodou je tedy přenositelnost, kdy skripty nebo webové stránky můžeme spustit v široké škále podporovaných prostředí a prohlížečů bez dalších

modifikací týkající se dané platformy. Nevýhodou je ve většině případů netypovost, což může vést k větší chybovosti a k faktu, že nevíme, zda je kód spustitelný, než se opravdu spustí. [6][7]

## 2.3 PROČ SE UČIT PROGRAMOVAT

### 2.3.1 AKTUÁLNOST

Již v roce 2008 měl pan Vystavěl v pedagogickém článku s názvem Informatika a informační a komunikační technologie myšlenku *„Považuji za důležité, aby se vyučovaly aktuální věci. Žáci pak vidí, že je to něco smysluplného, že se učí pro život a ne pouze pro školu. Ve výuce programování se to odráží především ve volbě platformy a programovacího jazyka, potažmo také vývojového prostředí.“* [8] Je třeba výuku přizpůsobovat dnešní době a vést žáky k takovému způsobu myšlení, které v této době uplatní. Je třeba se přizpůsobit i trhu práce, ve kterém velmi vzrůstá poptávka po pozicích ve vývoji digitálních technologií, kde nachází velké uplatnění třeba právě zmíněné algoritmické myšlení či přímo programování. Je pochopitelné, že ne každý žák se stane programátorem, ale je třeba, aby se s tímto pojmem alespoň setkal. Tyto myšlenky měli významní lidé naší historie, a to například Steve Jobs prohlašující: *"Všichni v této zemi byste se měli učit programování, protože vás to naučí myslet."* Nebo známý teoretický fyzik Stephen Hawking, který prohlásil: *„Pokud chcete odhalit tajemství vesmíru, nebo se chcete věnovat kariéře v 21. století, základy počítačového programování jsou zásadní dovedností, kterou byste se měli naučit.“* [9]

### 2.3.2 RVP

RVP je zkratka pro rámcové vzdělávací programy, které byly zavedeny v České republice zákonem č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborném a jiném vzdělávání. RVP obsahuje konkrétní cíle, formy, délku a povinný obsah vzdělávání. Tyto atributy jsou zde rozepsány nejen všeobecně, ale i odborně podle zaměření oboru vzdělání. Lze zde najít organizační uspořádání, profesní profil, podmínky průběhu a ukončování vzdělávání a zásady pro tvorbu školních vzdělávacích programů. Zaměřuje se zde na speciální vzdělávání, kde jsou popsány materiální, personální, organizační a bezpečnostní podmínky pro žáky se speciálními vzdělávacími potřebami. Školní vzdělávací programy jsou pak uplatněním zásad RVP pro konkrétní školy na základě jejich možností a specifik. [10]

V RVP platném od roku 2017 se nachází kapitola s názvem „Pojetí a cíle základního vzdělávání“, jež pojednává o koncepci a cílech základního vzdělávání, klíčových kompetencí a vzdělávacích oblastech. Vzdělávací oblasti jsou rozdělené na:

- jazyk a jazyková komunikace
- matematika a její aplikace
- informační a komunikační technologie
- člověk a jeho svět
- člověk a společnost
- člověk a příroda
- umění a kultura
- člověk a zdraví
- člověk a svět práce
- a doplňující vzdělávací obory

Pro náplň této bakalářské práce je zásadní oblast informační a komunikační technologie, která má umožňovat všem žákům získat základní dovednosti výpočetní techniky a moderních technologií. Na prvním stupni by se žáci měli naučit základní práci s počítačem a jeho základními periferiemi jako monitor, klávesnice, myš, reproduktory. Měli by se dozvědět, jak chránit data před ztrátou, poškozením a zneužitím. Žáci by se měli od vyučujícího dozvědět zásady bezpečnosti práce s hardwarem a softwarem, a také prevenci zdravotních rizik spojených s dlouhodobým používáním počítače. Při vyhledávání informací na internetu by se měli naučit používat vhodná klíčová slova a rychlé cesty, hledat v knihovnách a různých databázích. Nalezené informace by měli umět zpracovat v textovém editoru a měli by zvládnout i základní funkce grafického editoru. Prostřednictvím internetu by se měl naučit komunikovat e-mailem, chatováním nebo telefonováním. Na druhém stupni se rozvíjí zpracování informací v textových, grafických editorech a přibývají zde i tabulkové editory. V těchto editorech se uplatňují základní estetická a typografická pravidla. Řeší se zde práce s informacemi v souladu se zákony a duševním vlastnictvím. Při vyhledávání informací se volí různé zdroje a vyhodnocují se

jednoduché vztahy mezi údaji. Učí se zde prezentovat pomocí různých prezentačních programů a multimédií.[10]

RVP by bylo vhodné v pravidelných intervalech přezkoumávat a zajišťovat jeho aktuálnost. Je vhodné hlídat působení společenských změn, zohlednit poslední výsledky vědeckého výzkumu, využití nových technologií či zkušenosti ze škol, které by signalizovaly, že RVP neobsahuje dostatečně efektivní postupy pro výuku atp. Kvůli aktuální možnosti digitálních technologií k výše uvedené charakteristice oblasti informační a komunikační technologie a očekávaným výstupům přibyla v návrhu revize RVP z roku 2018 oblast informatické myšlení a s ní i téma algoritmizace a programování. V tomto návrhu se nachází tabulka očekávaných výstupu podle stupně vzdělání od předškolního věku po střední školy. [11]

Také se zde změnil přístup k výuce informačních a komunikačních technologií. Podle nového návrhu by se měla učit i v ostatních oblastech výuky. A je to proto, že digitální technologie pronikají do různých oblastí a činností člověka. Tyto oblasti a činnosti by žáci nedokázali pochytit v jednom předmětu. A je nejlepší se jim věnovat v předmětech, které souvisejí s danými oblastmi a činnostmi. Toto by mělo žákům přispět i k využívání výpočetní techniky v životě 21. století. Tento přístup výuky spočívá v tom, že každý jednotlivý předmět přispěje do celku digitální gramotnosti (viz obrázek 4). [11]


Obrázek 4 Schéma rozvoje digitální gramotnosti v předmětech, Zdroj: Návrh revizí v ICT, 2018

# 3 BLOKOVÉ PROGRAMOVÁNÍ

První bloky v programování se začaly objevovat v imperativním jazyce ALGOL, ze kterého vychází mnoho dnes používaných jazyků jako např. PASCAL a C. Bloky jsou v tomto jazyce logicky ohraničené slovy **begin** a **end**, které se dají přeložit jako začátek a konec. ALGOL 60 je zkratka pro Algorithmic Language, jež vznikl v 60. letech 20. století. Tento jazyk popisuje jednotlivé bloky tak, že se dají bloky vzájemně skládat bez nutnosti řešení jejich návaznosti. A to především kvůli tomu, že se zde nachází lokální proměnné, které jsou využívané právě v jednom bloku a v jiných blocích používat nejdou. [12]

Způsob blokového programování je něco mezi component-based a end-user programováním. Component-based programování je založeno na vývoji software pomocí předem vytvořených komponent. Komponenta může být softwarový balík, webová služba, nebo modul, zapouzdřující sadu souvisejících funkcí. Tyto komponenty bývají části objektově orientovaného jazyka v podobě objektů. Výhodou komponent by měla být znovu použitelnost. End-user programování je vývoj týkající se činnosti nástrojů, které umožňují koncovým uživatelům vytvořit nebo upravit software a jeho automatizované chování bez znalosti nějakého programovacího jazyka. Blokové programování navíc umožňuje integraci několika programovacích bloků pro vývoj některých aplikací v krátkém časovém období. Hlavní myšlenkou blokového programování je, že pro vývoj aplikací by do sebe měly programovací bloky zapadat formou lega. Jsou k dispozici bloky řešící určité úlohy, uživatelé je pak mohou upravovat a vytvářet s nimi aplikace přizpůsobené jejich potřebám. Bloky se chovají jako příkazy v kódu programu, a v textové formě se proto označují jako programátorské závorky. [13]

Dále se v této kapitole budu zabývat vizuálním blokovým programováním a jeho prostředími.

## 3.1 PROČ SE UČIT BLOKOVÉ PROGRAMOVÁNÍ

Na začátek je třeba zdůraznit několik výhod pro vývoj aplikací, jež jsou: nižší náklady, rychlost, opakovaná použitelnost, nejsou nutné profesionální IT dovednosti, volnost při přidávání API do aplikace. [13]

Blokové programování lze brát jako určitou formu propedeutiky programování. Propedeutika je příprava ke studiu určitého oboru.[14] Tímto programováním se žák může

velmi dobře připravit na klasické programování. Jak již bylo zmíněno v předešlé kapitole, souvisí klasické programování se zápisem programového kódu. Tento zápis je možné usnadnit pomocí různých podporujících editorů, které dokáží programátorovi ulehčit práci v napovídání, dokončování a zvýrazňování kódu. Ale přes veškerá usnadnění zde neodpadá znalost syntaxe a jednotlivých příkazů. A učení se psaní programového kódu může snadno žáka odradit a demotivovat v rozvoji algoritmizace a programováni.

Blokové programování je na rozdíl od klasických jazyků v grafické podobě, je intuitivní, vizuální a názorné, a tím pro děti snadněji pochopitelné než obyčejný kód. Žáci se nemusí zabývat způsobem zápisu, ale místo toho se mohou soustředit na jednotlivé části programu a jejich logickou návaznost. Takže pro vývoj programu stačí analyzovat postup pro daný proces, a tedy vytvářet algoritmus. [13]

I přesto, že prostředí pro programování založené na blocích je většinou názorné a intuitivní, mívají žáci často problém s pochopením proměnných, cyklů a výrazů. A tedy pro efektivnější a rychlejší výuku programování byly vytvořeny i jiné propedeutické aktivity, které by měly předcházet i blokovému programování, a to tzv. non-programming (neprogramovací) aktivity, které mají sloužit právě k pochopení proměnných, výrazů, cyklů a ostatních oblastí programování. Tyto aktivity se nazývají VELA a můžeme je najít v několika prostředích blokového programování jako např. Scratch, Snap! a Blockly. Název je zkratka obsahující výrazy:

**Variables** jsou pojmenované hodnoty, které se mohou měnit.

**Expressions** jsou výrazy pro práci s proměnnými.

**Looping** znázorňují opakování nějaké činnosti.

**Abstraction** jsou struktury, jejichž vnitřní logiku není při použití potřeba znát.

Každá z VELA aktivit má být modulární a samostatná tak, aby mohla být použita v jakémkoliv úvodu programování. VELA učební cíle jsou, aby se studenti naučili, jak jsou programy prováděny postupně a v souvislostech. [15]

Ve studii, kde se srovnávali studenti, kteří se učili jazyky C# nebo Java, bylo dokázáno, že ti, kteří prošli kurzem v aplikaci Scratch, chápali mnohem lépe oblasti programování a postupovali rychleji než ti, kteří kurzem neprošli. Dále proběhla studie zaměřená na přenos učení, kde byly upraveny kurzy na jazyk Java, kde se studenti nejprve učili v

prostředí Alice. Absolventi kurzu v Alice měli o 10 % lepší výkon než ti, kteří se rovnou učili samotný jazyk Java. V prostředí Alice se totiž programuje blokově, ale studenti při programování vidí i kód algoritmu v jazyce Java. [16]

## 3.2 ROZDÍL MEZI SOFTWAROVÝM A UNPLUGGED PROSTŘEDÍM

V předešlé kapitole zmiňované VELA aktivity jsou vlastně převážně unplugged aktivity, jež mohou být jednoduše implementovány do učebnic informatiky. Unplugged, z anglického slova "odpojený", znamená ve slovním spojení s prostředím, že neprobíhá přímo na počítači, ale mimo něj. Oproti tomu softwarové prostředí probíhá především na počítači či na jiném zařízení jako například tablet nebo mobilní telefon. Unplugged programování je řešeno formou kartiček nebo destiček, které se skládají jako puzzle. Tato prostředí se dají přirovnat stavebnicím, které jsou řešeny formou zábavných naučných her.

Někteří jedinci mohou mít obavy z výuky na počítači, protože se bojí vysoké obtížnosti a neporozumění. Unplugged programování může podporovat sebevědomí, protože k němu mají mladší jedinci blíže. Výhodou této metody programování je také to, že není třeba vlastnit počítač a tím je nižší investice do škol, výuka nemusí bezprostředně probíhat v počítačové učebně. Softwarové prostředí je ale výhodnější v tom, že může obsahovat více předpřipravených bloků kódu a také větší prostor pro rozmanitost v programování. Ve výzkumu výuky pomocí unplugged aktivit bylo zjištěno, že v nich žáci dosahovali lepších výsledků než v aktivitách přímo na počítači. Není známo, čím to je způsobeno, zda sebevědomím žáků či atmosférou ve výuce bez počítače. [17]

I přesto, že je unplugged prostředí často popisováno jako prostředí, jež neprobíhá na žádném digitálním zařízení, není to úplně pravda. Samotné algoritmy se sice mohou skládat pomocí herních destiček, ale jejich ověření i výstup může probíhat třeba právě na digitálních zařízeních. Na nich může být i rychlejší a přesnější, případně se zajímavě řešenou zpětnou vazbou. V dalších podkapitolách budou někteří zástupci softwarového a unplugged prostředí představeni.

## 3.3 ZÁSTUPCI SOFTWAROVÉHO PROSTŘEDÍ

Zástupců softwarových prostředí existuje mnohem více než zástupců unplugged prostředí. Jako nejznámější a nejpopulárnější prostředí se dá považovat aplikace Scratch, která je blíže popsána v další kapitole.


Obrázek 5 Logo Scratch, Zdroj: [18]

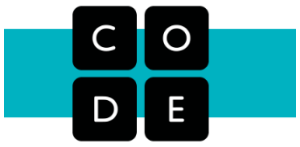
Společnost Google vytvořila JavaScriptovou knihovnu, zaměřenou na vizuální blokové programování, která je volně dostupná, a proto ji využívají i jiné aplikace. Tato knihovna je popsána v další podkapitole, kde je popsána aplikace zaměřená na výuku programování s názvem Blockly Games, rovněž od společností Google. [19]


Obrázek 6 Logo Blockly Games,
Zdroj: [31]

**Code.org**

Code.org je nezisková organizace zabývající se rozšiřováním přístupu k informatice. Tato organizace prohlašuje, že se snaží do informatiky přimět zapojit i ženy a menšiny, a také uvádí, že jejich kurzy používají desítky milionů studentů. Díky spolupráci s mnoha pedagogy vytváří vlastní vzdělávací kurzy. Komunita kolem tohoto projektu je velmi široká a je snaha ji dále rozšiřovat už jen z toho důvodu, aby mělo smysl aplikaci rozvíjet v co nejvíce jazycích a byla tak globálně dostupnější. Prostředí Code.org také využívá připravenou knihovnu Blockly od společnosti Google. Nabízené a zdarma dostupné kurzy projektu jsou rozděleny do věkových kategorií a je na uživateli, které si vybere. Pro mládež mohou být velmi přitažlivé aktivity zaměřené na moderní témata jako Minecraft, Frozen, Hvězdné války a různé známé hudebníky. U jednotlivých her se nachází i příručka pro učitele, většinou v anglickém jazyce stejně jako některé kurzy. I webové stránky jsou

převážně v angličtině, ale je vidět, že na češtině pracují. Veškeré aktivity probíhají v prohlížeči, a proto není aplikace nikterak hardwarově náročná. [20]




Obrázek 7 Logo CODE.org,
Zdroj: [20]

**OzoBlockly**

Již zmiňovaná knihovna Blockly je využívána v prostředí pro programování ozobotů, které je dostupné pro počítače jako webová aplikace nebo jako aplikace pro telefony či tablety pro Android i iOS. Ozoboty jsou malé roboty, které se dají programovat podle kreslení čar a barevných příkazů. Tyto příkazy jsou jednoduché a snadno pochopitelné pro děti od 6 let. Pro čtení čar Ozobot používá senzory, které čtou černou čáru, po níž jezdí a podle barevných značek vykonává různě aktivity jako třeba otáčení a jiné pohybové prvky, pípání a blikání. Vodící čáry a znaky si žáci kreslí sami na papír, nebo používají předpřipravené vytištěné značky. Díky OzoBlockly je možné Ozoboty programovat pomocí blokového programování. Na oficiálních stránkách je možné najít několik inspirací pro využití Ozobotů jak doma, tak i ve výuce. Také se zde dá najít sekce, kde se nachází hry zaměřené na vzdělávání. Volně dostupné hry jsou tři a každá obsahuje deset úrovní, přičemž k nim není třeba vlastnit Ozobota, protože je zde předpřipravené okno pro grafický výstup kódu. Ale hned pod tímto oknem se nachází tlačítko pro načtení kódu do Ozobota, a tedy se dá tento vytvořený program zrealizovat i fyzicky. Pro používání Ozobota při těchto hrách je třeba jej nastavit, což se provede pomocí obrazovky, na které se bude programovat. Ozobot se přiloží snímačem na obrazovku se stoprocentním jasem a ten se pomocí toho, co načte z obrazovky, spáruje se zařízením a následně se na něm mohou spouštět vytvořené programy.[21]


Obrázek 8 Logo OzoBlockly,
Zdroj: [21]

**Micro:bit**

Velmi zajímavá interaktivní pomůcka pro využití ve výuce je zařízení Micro:bit. Jedná se o mikropočítač, kterému se dají rozšiřovat jeho vlastnosti pomocí doplnění různých vnějších obvodů. Ale i jako samotná destička může být velmi zajímavě využita při programování. Naprogramovat se dá pomocí vlastního prostředí, které je velmi podobné Scratch. Je dostupné na oficiálních stránkách, kde je vytvořené knihovnou Blockly (díky tomu je editor česky), nebo také v mobilní aplikaci dostupné na Android i iOS. Výstupem programu může být obraz zobrazený na matici 5x5 červených LED diod, nebo i zvuk a vibrace. Tato destička obsahuje tlačítka, která po naprogramování vytvoří z destičky například ovladač na autíčka. Také obsahuje akcelerometr a kompas pro zjišťování polohy, které vedou k širší škále kreativního využití. Pro nahrání programu na Micro:bit je třeba USB kabel, díky kterému se připojí k počítači, a do příslušné složky se vložit vyexportovaný soubor na disk s názvem MICROBIT. Při používání mobilní aplikace lze program nahrát bezdrátově pomocí Bluetooth. Pro využití Micro:bit prostředí pro programování není třeba vlastnit mikropočítač, protože je zde grafický výstup přímo ve vývojářském prostředí. Na stránkách microbit.org se nachází projekty, které mohou uživatele inspirovat a naučit, jak zacházet s Micro:bitem. Také je zde pod záložkou „Lekce“ dvanáct okruhů, které obsahují lekce zaměřené na dané téma. Tyto okruhy jsou: zpracování dat, digitální flashcards, elektrické vodiče, aktivity, hudební micro:bit, přírodní umění, animace sopky, výpočetní základy, zdravotnická technika, úvod do kryptografie, úvod do kybernetické bezpečnosti, učebna smyslů.[22]

Obrázek 9 Logo Micro:bit,
Zdroj: [22]

**MIT App Invetor**

Jak lze vyčíst z názvu byl vytvořen v Massachusetts Institute of Technology (MIT) za účelem zjednodušení vývoje mobilních aplikací pro operační systém Android. Na oficiálních stránkách můžeme najít informace a instrukce pro učitele, a také čtyři hlavní výukové programy, které se zabývají začátky vývoje, kreslení, tvorby hry „Mole mash“ a příručka vytváření mobilních aplikací pro mládež. Nachází se zde mnohem více výukových bloků,

ale bohužel nejsou v češtině. Vývojové prostředí vypadá jako klasické prostředí pro vývoj Android aplikací. Obsahuje záložku pro design aplikace, kde se nachází mobilní telefon, do kterého uživatel může přetahovat různé komponenty, jako jsou tlačítka, zaškrtávací políčka, popisky apod. Dále je zde rozvržení grafického návrhu aplikace, média, mapy, kreslení a animace, senzory a jiné mobilní funkcionality. Další záložka obsahuje klasické vývojové prostředí s bloky, vytvořené knihovnou Blockly, které určuje logiku, a tedy funkčnost jednotlivých komponent a celé aplikace.[23]


Obrázek 10 Logo MIT App Inventor,
Zdroj: [23]

**Snap!**

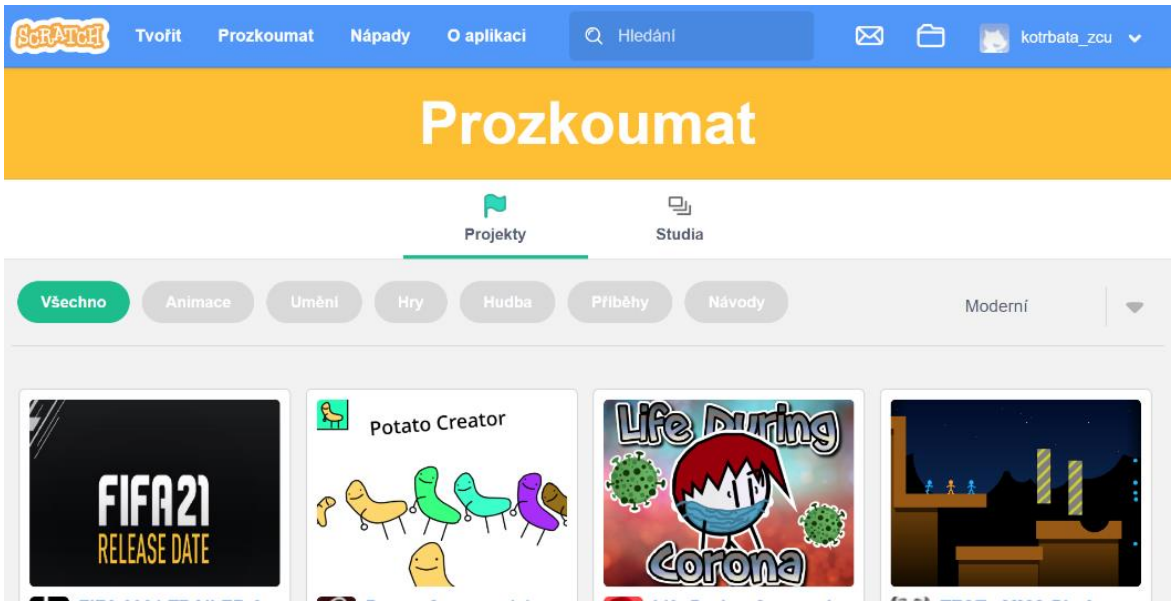
Toto je rozšířená reimplementace aplikace Scratch umožňující si vytvořit své vlastní bloky. Především se zaměřuje na úvod do informatiky středního a vysokoškolského vzdělání. Díky implementaci v jazyce JavaScript je možné Snap! používat na jakémkoliv zařízení pomocí webového prohlížeče. Není zde kladen důraz na poutavou grafiku jako v aplikaci Scratch, která je zaměřena hlavně na mladší populaci. Webové stránky Snap! nejsou k dispozici v češtině, ale vývojové prostředí je možné do češtiny přepnout. Na stránkách se na rozdíl od aplikace Scratch nenachází tutoriály pro výuku, ale pouze publikované projekty komunitou. Dále je k dispozici dané vývojové prostředí, fórum, kde je popsáno jak Snap! funguje, možnost podpory, a nakonec je zde záložka s plánovanými konferencemi, ve které je možné si zjistit, kdy konference probíhají a lze zde na ně koupit vstupenky. [24]


Obrázek 11 Logo Snap!,
Zdroj: [24]

### 3.3.1 SCRATCH

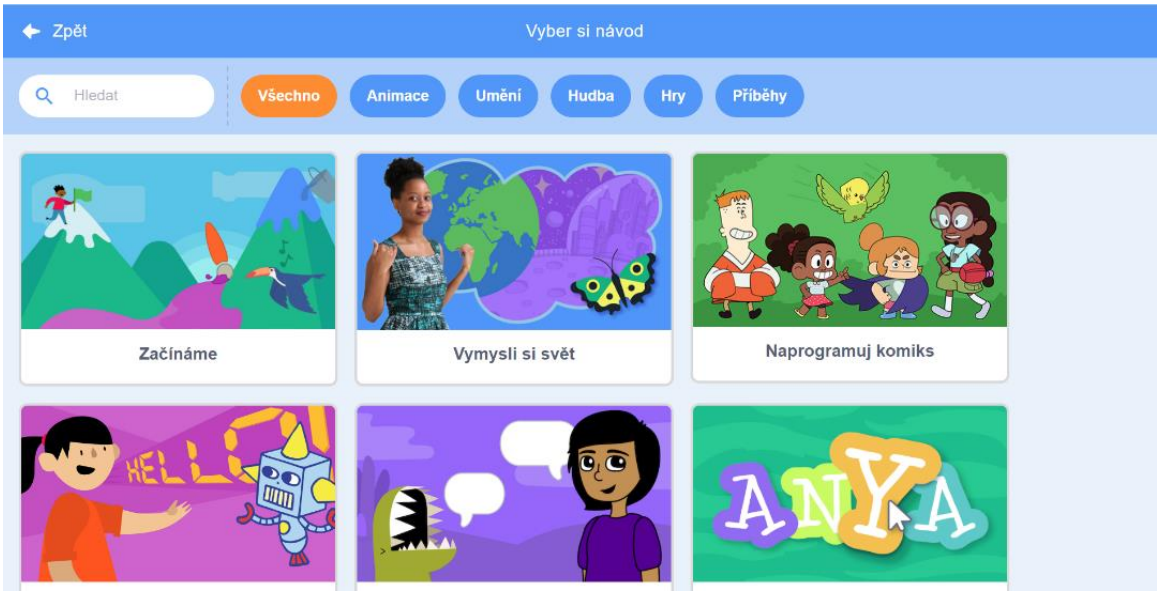
Scratch je softwarová aplikace s grafickým rozhraním, která podporuje kognitivní vývoj a motivaci prováděním aktivit nejen při vyučování. Toto softwarové prostředí je vytvořeno organizací Lifelong Kindergarten Group v MIT Lab. Jeho výhodou není pouze to, že je bezplatně poskytované, ale také, že je dostupné ve více než čtyřiceti jazycích a tím propojuje více lidí a vytváří větší komunitu. Aplikace Scratch je zaměřena na programování interaktivních příběhů, her a animací, které je možno sdílet se svojí komunitou. Napomáhá svým uživatelům s rozvojem algoritmického myšlení a tvořivosti. Je určena pro věkovou hranici 8 až 16 let, ale používají ji lidé jakéhokoliv věku, a díky nim se vytváří mnoho projektů, které mohou inspirovat ostatní uživatele. Momentálně je k dispozici online prostředí, které lze spustit v prohlížeči, a také offline prostředí, jež může být nainstalováno na zařízení a lokálně na něm fungovat. [18]

Online prostředí je výhodné v tom, že se uživatel nemusí starat o místo v paměti a je přímo propojen s komunitou. Po kliknutí na záložku „Tvořit“ v hlavním menu, nebo po kliknutí na tlačítko „Začni vytvářet“ přímo na úvodní stránce: https://scratch.mit.edu/, Scratch poskytuje úvodní lekci pro začátečníky. Na úvodní stránce projektu se nachází několik dalších záložek v menu, jako například „Prozkoumat“, kde se vyhledávají veškeré projekty, které jsou přehledně rozděleny na kategorie, jak je vidět na obrázku 12.




Obrázek 12 Záložka Prozkoumat na stránce Scratch, Zdroj: aplikace Scratch

Po kliknutí na projekt si lze vyzkoušet výstup projektu v náhledovém okně, přečíst si návod k použití a také nějaké poznámky od autora. Je zde i vidět počet zhlédnutí a počet lidí, kterým se daný projekt líbí. K dispozici je možnost dát najevo svůj názor kliknutím na srdíčko nebo přidáním komentáře. Pokud by někoho zajímala realizace tohoto projektu, může nahlédnout dovnitř a prozkoumat jednotlivé komponenty projektu a jejich kód. Kdo by chtěl projekt modifikovat, může si vytvořit jeho kopii a pracovat na něm jako na svém projektu, jenž se dá uložit. Jako další záložka v hlavním menu je „Nápady“, kde Scratch obsahuje návody na aktivity, které uživatele učí základním věcem jako animace, tvoření obrázků, zvuků, příběhů, her a podobně. Je jich tu hned několik a jsou také přehledně rozděleny (viz obrázek 13).

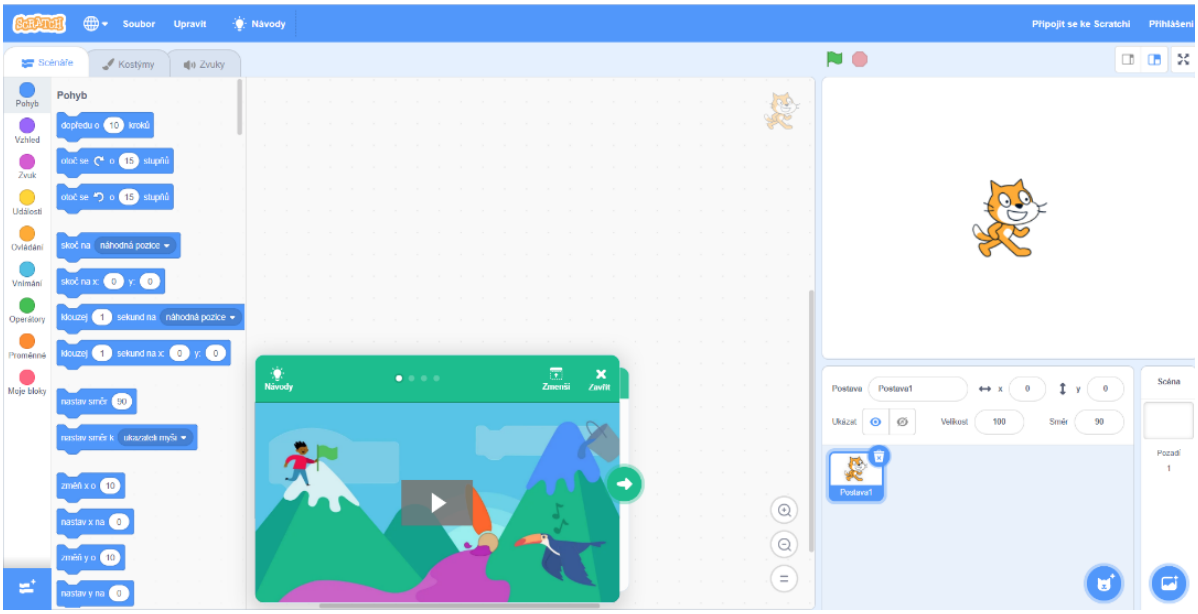



Obrázek 13 Nápady na stránce Scratch, Zdroj: aplikace Scratch

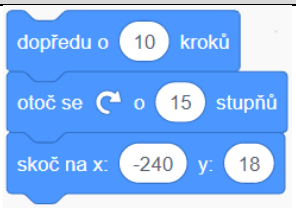
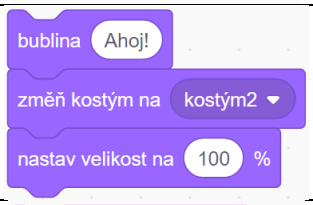
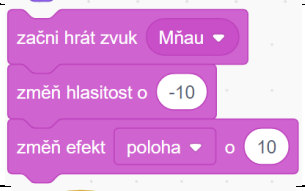
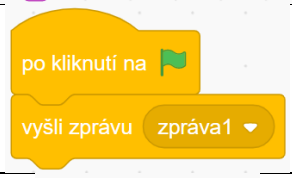
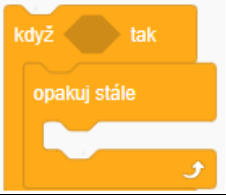
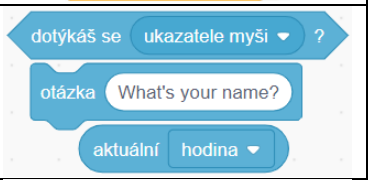
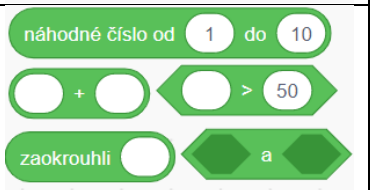
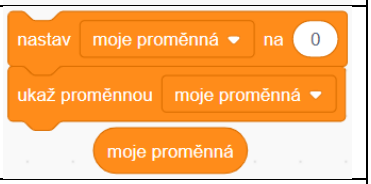
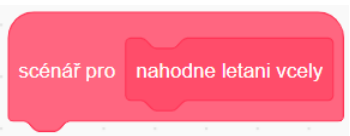
Podobně jako u prvního tvoření se u těchto aktivit zobrazí video tutoriál, který uživatele vede k danému výsledku. Tutoriál se objeví přímo na pracovní ploše jako pohyblivá karta s videi nebo animacemi, se kterou lze libovolně pohybovat, aby bylo možné při tutoriálu pracovat. Video v tutoriálu se může otevřít na celé okno, jako je to v běžných video přehrávačích zvykem. Zde obsažená videa jsou kreativně zpracována, ale jejich nevýhodou je, že jsou namluvena v anglickém jazyce. Pokud je zde nastavený český jazyk, obsahují české titulky. To ale nemusí být pro studující překážkou, protože zde je přesně graficky znázorněno, co má uživatel udělat. Jako další záložka v menu je „O aplikaci“ obsahující základní informace o aplikaci. Vedle záložky se nachází vyhledávání, díky kterému lze vyhledávat různé projekty podle klíčových slov. Pokud je uživatel přihlášen, nachází se vedle

vyhledávání ikonka dopisu, jež obsahuje zprávy s informacemi od aplikace. Vedle této ikonky se nachází obrázek složky, kde jsou uživatelovy projekty. Tyto projekty jsou soukromé, nebo sdílené s komunitou. Jako poslední věc v menu se nachází uživatelský účet, kde se lze po kliknutí dostat na profil, do složky vlastních projektů, nastavení účtu, nebo odhlášení.

Při vstupu do prostředí, například pomocí záložky „Tvořit“, je prostředí velmi intuitivní a přehledné díky svému rozložení a barevnému provedení. V sekci „Scénáře“ se nachází na levé straně logicky rozdělené bloky do různě zbarvených skupin, které jsou zaznamenané v tabulce 7.

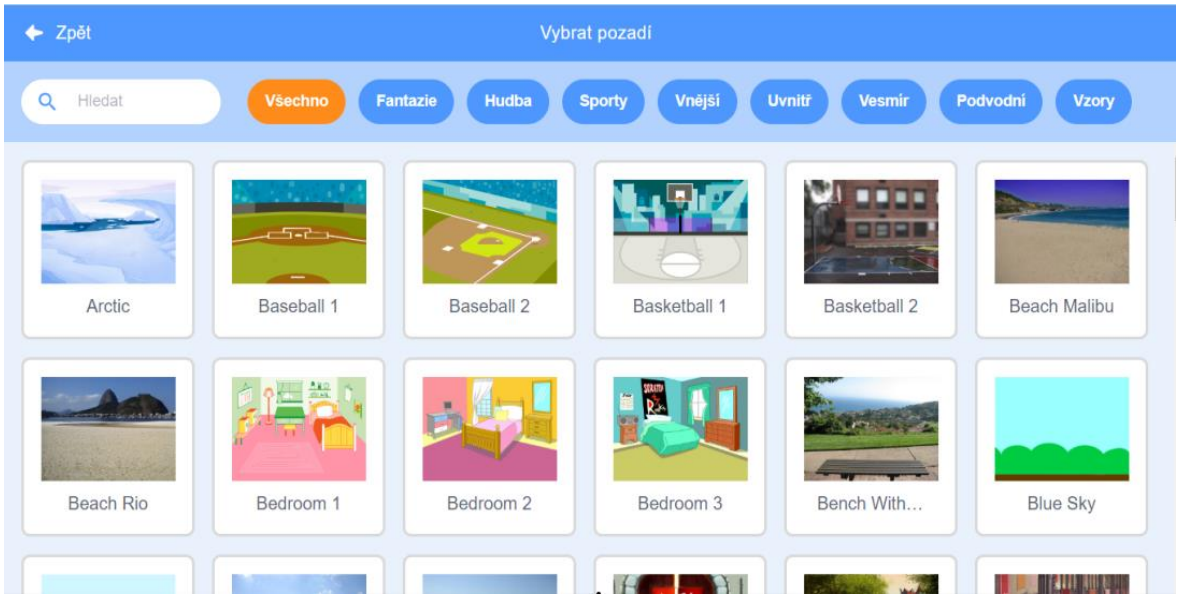



Obrázek 14 Prostředí Scratch, Zdroj: aplikace Scratch

| Skupiny | Vysvětlivka příkazů | Příklad příkazu |
|---|---|---|
| Pohyb | Nastavují posuny a pohyby po osách X a Y a také rotaci. |  |
| Vzhled | Slouží pro nastavení různých efektů jako přidání textových bublin, zvětšování a zmenšování objektů, měnění jejich barev a také jejich skrývání. |  |
| Zvuk | Přehrávají zvuk a také jej mění pomocí různých efektů. |  |
| Události | Tvoří začátek algoritmu a určuje, kdy se má jaká část kódu spustit. |  |
| Ovládání | Umožňuje větvení programu pomocí podmínek a zajišťuje opakování pomocí cyklů. |  |
| Vnímání | Umožňuje interakci s různými objekty. Poskytuje vytváření podmínek na různé akce, které se dějí. Nastavují se zde také proměnné. |  |
| Operátory | Zprostředkovávají užití aritmetických a logických operátorů. Jsou zde i některé matematické funkce jako například zaokrouhlení a generování náhodného čísla. |  |
| Proměnné | Vytvářejí, nastavují, mění, skrývají a ukazují proměnné. Také se zde dá uložit více hodnot do seznamu. |  |
| Moje bloky | Pro opakovanou použitelnost a přehlednost je zde umožněno vytvářet vlastní bloky kódu, což bychom mohli přirovnat k metodám z oblasti objektově orientovaných jazyků. |  |

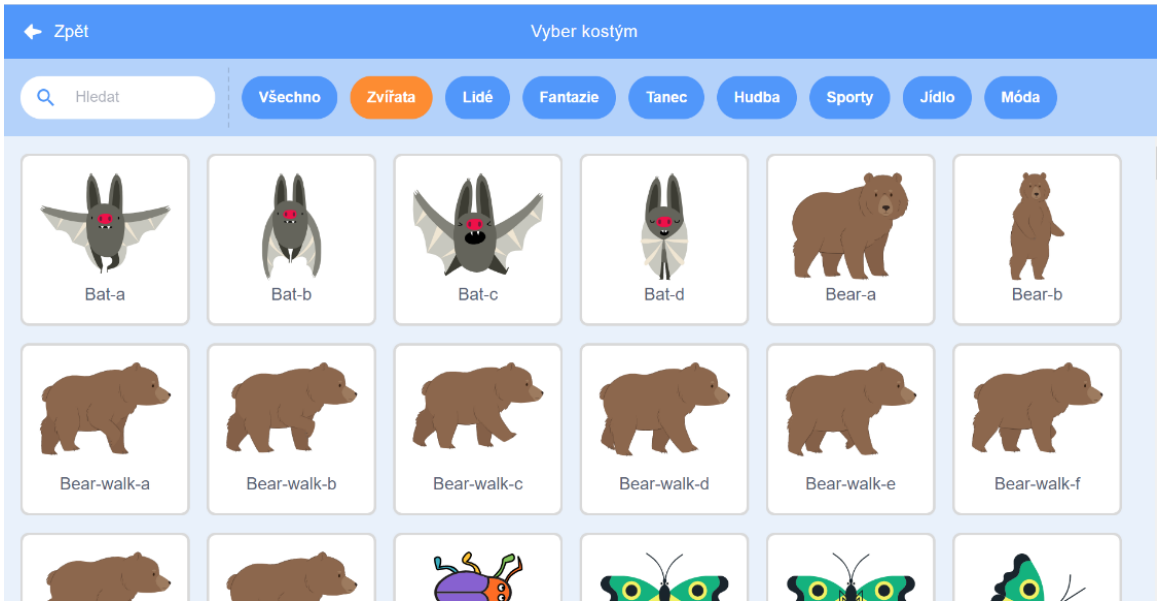
Tabulka 2 Popis příkazů v aplikaci Scratch

Vedle příkazů je pracovní plocha, kde se kód skládá a vedle ní výstup složeného algoritmu v okně se scénou programu. Pod tímto oknem se nachází informace o postavách a pozadí scény. U informací lze nalézt tlačítka pro přidání nové postavy a pro výběr či tvorbu pozadí. Pozadí je možnost nahrát ze zařízení nebo vygenerovat náhodně z databáze anebo si ho v databázi vyhledat. Stejně jako u kostýmů a zvuků je zde i mnoho obrázků na pozadí, jež jsou také přehledně rozděleny (viz obrázek 15).




Obrázek 15 Pozadí na stránce Scratch, Zdroj: aplikace Scratch

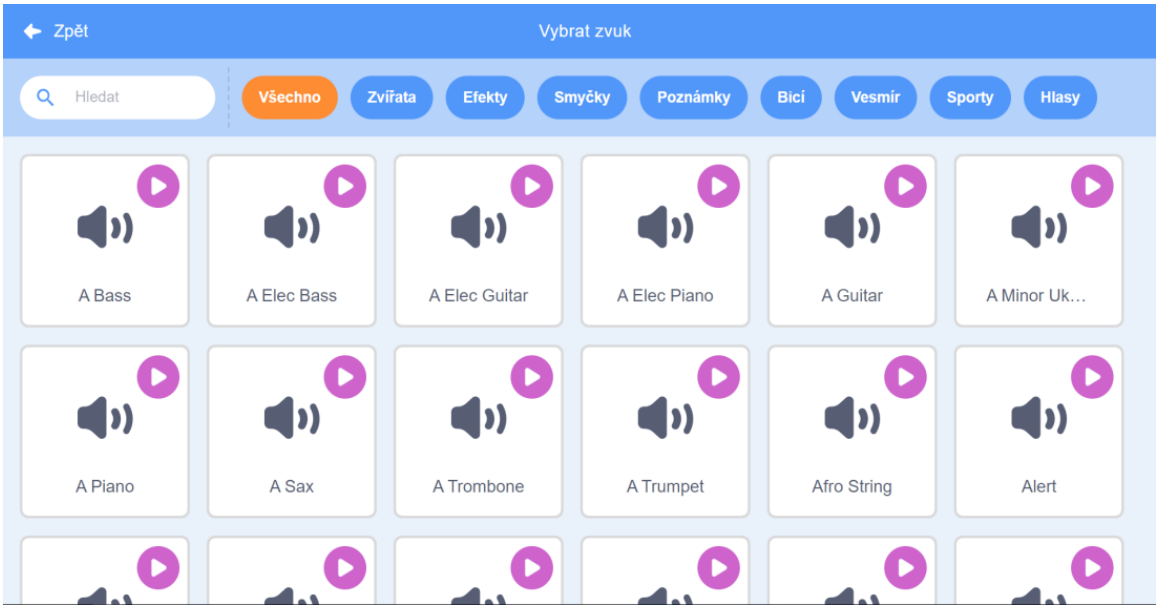
Algoritmy se vytváří pomocí „drag and drop“, to znamená, že se bloky přetahují myší do pracovní plochy. Používají se k ovládání takzvané postavy. Jednotlivé postavy se dají přirovnat jednomu objektu z oblasti objektově orientovaných jazyků, protože mohou mít svůj vlastní kód pro svou funkci v projektu. Každá postava má kromě kostýmu (vzhledu) i svůj scénář (tj. programovací prostor) a také zvuk. Základní postavou je kočka, ale může to být jakékoliv zvíře, postava, rostlina, obrázek, číslo nebo tvar. Nejen, že postav a kostýmů je tu nespočet předpřipraveno, ale nachází se zde i grafický editor, kde se dá kostým upravit nebo nakreslit svůj vlastní. Také je zde možnost nahrát obrázek přímo ze zařízení, a to buď již existující anebo vyfocený kamerou. Rozdělení kostýmů je zobrazeno na obrázku 16.




Obrázek 16 Kostýmy na stránce Scratch, Zdroj: aplikace Scratch

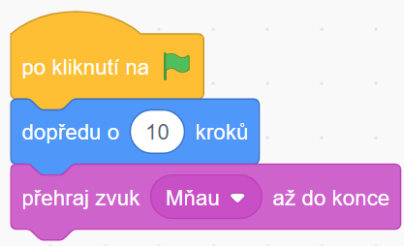
Po kliknutí na zvolený kostým se objeví v grafickém editoru našeho projektu u vybrané postavy a je možnost s ní dále pracovat. Takto rozděleny jsou i postavy, které lze programovat nezávisle na ostatních objektech.

Kromě kostýmů, postav a pozadí si zde uživatelé mohou upravovat zvuky v záložce „Zvuky“ používaných ve svých programech. Databáze nabízí uživateli mnoho předpřipravených zvuků, nebo možnost použít své vlastní. Stejně jako u kostýmů se zvuky najdou v záložce v levém sloupci pomocí tlačítka, které uživatele přesměruje do jejich databáze (viz obrázek 17). Pokud se najede myší na tlačítko přehrát, zvuk se přehraje a pokud je třeba ho použít, tak se po kliknutí převede do projektu, kde se zobrazí ve zvukovém editoru.




Obrázek 18 Zvuky na stránce Scratch, Zdroj: aplikace Scratch

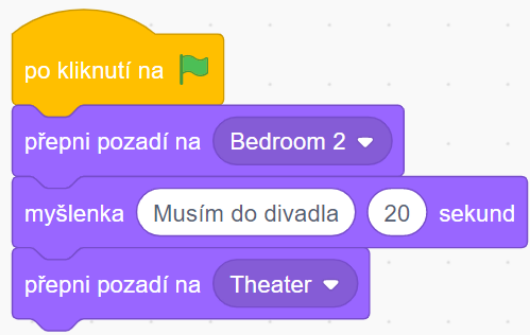
Hned na začátku prvního projektu tutoriál znázorňuje postup při vytváření algoritmu, jak jednotlivé bloky příkazů fungují a jak se skládají. V prvním videu je ukázáno, že algoritmus musí mít začátek, který určuje, kdy se program spustí. Ukázka je demonstrována na pohybu kočky a následnému mňouknutí, kde se program spustí po kliknutí na zelenou vlaječku. Také je zde vysvětleno, že lze snadno měnit pořadí bloků, že se s nimi dá pohybovat společně a že se bloky po přetáhnutí vlevo mimo okno smažou. Tutoriál poukazuje na to, mít v projektu i více postav, které mohou mít svůj vlastní scénář, a tedy svůj program. Vedle zelené vlaječky se nachází červený osmiúhelník nápadně připomínající značku ”stop”, který slouží k zastavení programu. Na obrázku 18 je vidět, že po kliknutí na zelenou vlaječku, kočka nejprve popojde o 10 kroků. Jeden krok je 1 pixel, což znamená, že se kočka posune na ose X o 10 px vpravo.




Obrázek 17 Seznámení se s tvorbou programu, Zdroj: aplikace Scratch

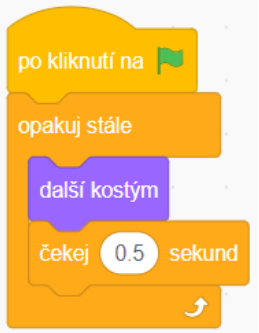
Po videu následují animace, které upřesňují to, že postava provede vytvořený program po klinutí na samotný příkaz. Samotný běh programu je třeba začít blokem pro určení začátku spuštění, jak je popsáno dříve. Poslední fáze tutoriálu vede uživatele k tomu, aby pokračoval v návodech dále. Následující nabídnuté návody jsou poskytnuty postupně

a logicky tak, aby žák v návaznosti objevil všechny vlastnosti aplikace Scratch. Jeden z nich se týká využití pozadí, které žáka motivuje k vytváření příběhů. Video znázorňuje výběr několika pozadí pro použití v daném příběhu. Ukazuje algoritmus určující, že po kliknutí na vlaječku se nastaví pozadí, pak se přidá k postavě bublina s textem na určitou dobu a po uplynutí se přepne na jiné pozadí. Přepínání pozadí lze v oblastech programování přirovnat k naplňování proměnné. Vlastně i v příkazu pro myšlenku se naplňuje proměnná textem (viz obrázek 19).




Obrázek 19 Naplnění proměnné myšlenka,
Zdroj: aplikace Scratch

Tato jednoduchost programu motivuje k tomu, aby žák chtěl získat více informací pro tvorbu svého příběhu a k získání dalších souvislostí, jež logicky navazují. Pro návaznost úkolů je Scratch připraven na konci návodu, kde odkazuje na tutoriály zabývající se dalšími funkcemi bloků kódu a možnostmi ve vývoji. Je zde například tutoriál popisující tvorbu animace. V animaci se žáci dostanou k další oblasti programování, a to k cyklům. K rozpohybování postavy je důležité mít alespoň dva kostýmy, popisující změnu pohybu. Jestliže je známý počet opakování pohybu, je potřeba zvolit blok pro cyklus s počtem opakování. Pokud není počet opakování známý, je zde cyklus, který nekončí. Pro rozpohybování postavy s podmínkou, aby se pohybovala, dokud se něco nestane, je zde připravený blok. Pro nastavení plynulosti pohybu je třeba použít blok s příkazem „čekej“. Znázornění algoritmu pro neustálý pohyb můžeme vidět na obrázku 20.




Obrázek 21 Nekonečný cyklus,
Zdroj: aplikace Scratch

K cyklům neodmyslitelně patří podmínky, sloužící k rozdělení kódu na splňující a nesplňující větev. Podmínky by se jistě neobešly bez výrokové logiky, a to konkrétně **and** (a zároveň) a **or** (anebo). Pro určování podmínek jsou důležité operátory jako **+**, **-** , **=**, **<**, **>**. Program pro podmínku a využití výrokové logiky i operátorů ukazuje obrázek 21.


Obrázek 20 Podmínka s operátory, Zdroj: aplikace Scratch

Na obrázku je vidět, že po kliknutí na vlaječku se vygeneruje náhodné číslo od 1 do 100 a uloží se do proměnné s příslušným názvem. Po vygenerování se tato proměnná vypíše v bublině, abychom viděli, zda podmínka funguje. Potom následuje podmínka, která ověřuje, zda je dané číslo větší nebo rovno číslu 50. Pokud ano, tak postava zamňouká a změní barvu, pokud ne, zobrazí se u postavy bublinka s myšlenkou „Už padne číslo větší nebo rovno (>=) 50?“.

Je zde nejen možnost vytvářet své proměnné, ale také seznamy, jež můžeme přirovnat k polím. Stejně jako pole slouží k ukládání více hodnot, ke kterým se přistupuje pomocí číselných indexů (v prostředí Scratch se používají od 1 do počtu prvků). Pro demonstraci

fungování takového pole ve Scratch je vytvořen projekt, kde se řeší výpis seznamu jmen lidí ve skupině nacházející se na této adrese https://scratch.mit.edu/projects/394609834. Projekt obsahuje 6 objektů: tlačítko, reproduktor a 4 postavy. Při spuštění programu vlaječkou se zobrazí tlačítko, které po stisknutí naplní „Seznam jmen“ čtyřmi jmény a posléze tlačítko uživateli řekne, ať zmáčkne mezerník, po 5 sekundách tlačítko zmizí. Po stisknutí mezerníku se ukáží u jednotlivých postav jména a nastaví se vytvořená proměnná s názvem iterace na číslo 0, pak následuje cyklus se známým počtem opakování, do kterého vstupuje délka seznamu jmen. V tomto cyklu se nejprve čeká 4 sekundy, abychom viděli v bublině postupně jdoucí jména. V cyklu se nastaví proměnná iterace na `iterace + 1`. Daná iterace vstoupí do příkazu, jenž vypisuje položky seznamu podle pozice. Výpis je pomocí bubliny zobrazující se u reproduktoru. Příkazy v cyklu proběhnou čtyřikrát, přičemž délka seznamu je 4, proměnná se v průběhu cyklu mění na číslo o jedna větší, a tím pádem cyklus projde a vypíše všechny položky v seznamu.

Základní koncepty programování, jako jsou cykly, podmínky, proměnné a pole, se zde žáci naučí velmi snadno díky dobré volbě příkladů. Prostředí dokáže dobře motivovat svým kreativním zpracováním, tutoriály a již hotovými projekty. Díky aplikaci se uživatel naučí mnoho věcí jako například základní matematické pojmy, souřadnice, náhodná čísla, navrhování projektů, analyzování, a jiné.

### 3.3.2 BLOCKLY

Blockly je JavaScriptová knihovna, obsahující editor vizuálních kódů, používající se ve webových a mobilních aplikacích. Je open source, což znamená, že kdokoliv tuto knihovnu může rozvíjet a používat ve svých vlastních aplikacích. Výhoda pro tvoření prostředí pro blokové programování u jazyka JavaScript je, že běží na straně klienta a není závislý na serverové části. Knihovna podporuje několik programovacích jazyků a jejich syntaxe je zabalena do grafických bloků jako například proměnné, logické výrazy, cykly a další. Nejpoužívanější a nejoblíbenější podporované jazyky jsou:

- JavaScript,
- Python,
- PHP,
- Lua,
- Dart.

Implementace této knihovny je vysvětlena a popsána na stránkách pro vývojáře. Jsou zde tři návody pro použití ve vlastní aplikaci, a to:

- pro webovou aplikaci
    - https://developers.google.com/blockly/guides/get-started/web
- pro aplikaci na Android
    - https://developers.google.com/blockly/guides/get-started/android
- pro aplikaci na iOS
    - https://developers.google.com/blockly/guides/get-started/ios

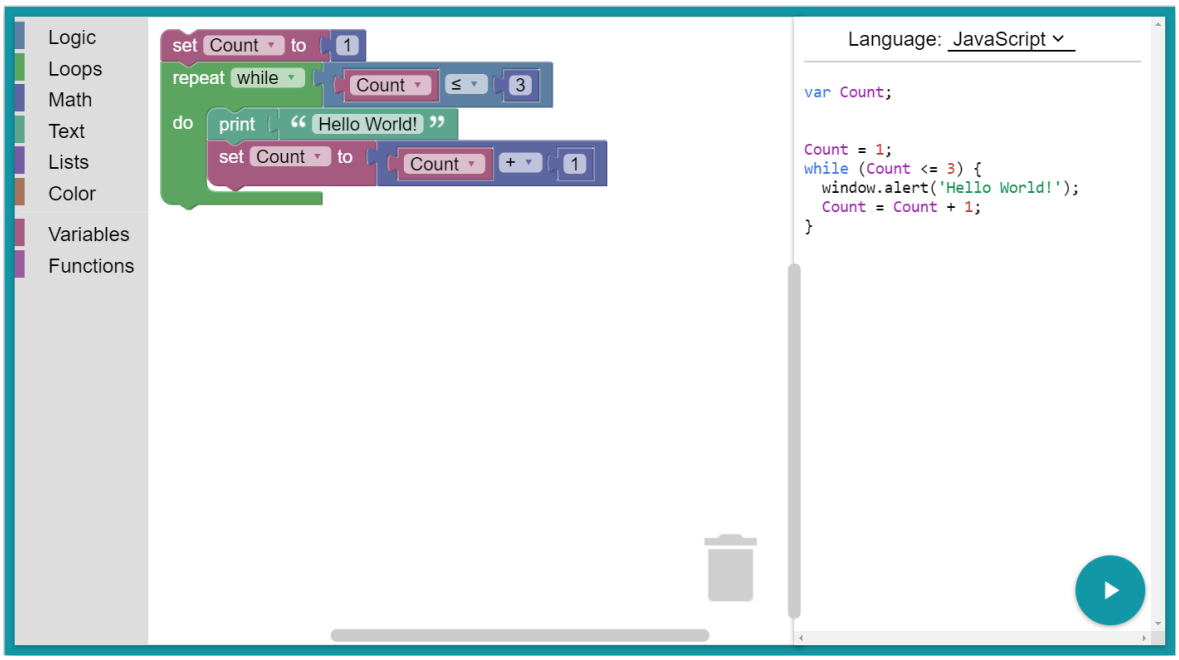
Díky své dobré implementaci a cenové politice (je zdarma) je Blockly využíváno mnoha vzdělávacími projekty. Příklady prostředí vytvořené pomocí knihovny Blocky se nachází v tabulce 3.

| | |
|---|---|
| **Blockly Games** | Obrázek 22 Logo BlocklyGames, Zdroj: [31] |
| **App Inventor** | Obrázek 23 Logo AppInventor, Zdroj: [23] |
| **Code. org** | Obrázek 24 Logo CODE, Zdroj: [20] |
| **OzoBlockly** | Obrázek 25 Logo OzoBlockly, Zdroj: [21] |
| **Micro:bit** | Obrázek 26 Logo Micro:bit, Zdroj: [22] |

Tabulka 3 Vzdělávací projekty vytvořené pomocí knihovny Blockly

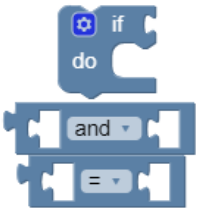
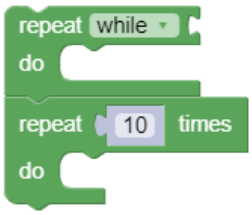
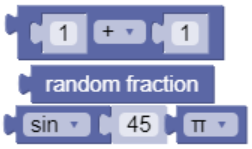
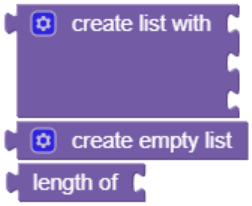
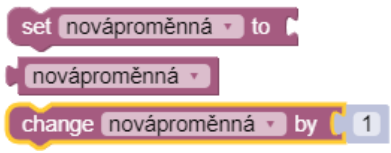
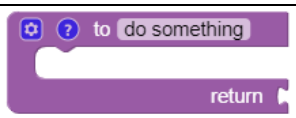
Samotné Blockly je možné si vyzkoušet na stránkách https://developers.google.com/blockly. Nachází se tam okno (viz Obrázek 27) zobrazující na levé straně skupiny bloků kódu, vpravo od nich vývojové okno, kde se bloky kódu skládají, a zcela vpravo prostor překládající použité bloky do různých programovacích

jazyků (primárně je nastaven JavaScript). Pro smazání bloků stačí dané bloky uchopit a přetáhnout je do levého sloupce anebo do koše nacházejícího se v pravém rohu vývojového okna. V pravém dolním rohu celého okna je tlačítko pro spuštění vytvořeného algoritmu. Při vstupu je na stránce předpřipravený program pro výpis často využívaného pozdravu „Hello World“ nacházející se v cyklu. Po stisknutí tlačítka play nám tedy vyskočí okno a po kliknutí na tlačítko ok, vyskočí znovu, dokud se nám nezobrazí třikrát, jak je určeno v cyklu. [19]




Obrázek 27 Ukázka prostředí Blockly, Zdroj: [19]

Bloky do sebe zapadají jako puzzle a stejně jako v aplikaci Scratch se k sobě přichytávají a dá se s nimi pohybovat jako s celkem. V této knihovně jsou také barevně odlišeny podle funkcionality a jejich rozdělení je vidět v tabulce 4.

| Skupiny | Vysvětlivka příkazů | Příklad příkazu |
|---|---|---|
| Logické | Nachází se zde podmínky spolu s aritmetickými a logickými operátory, jež jsou v podmínkách používané. |  |
| Cykly | Obsahují bloky zajišťující opakování pomocí cyklů. |  |
| Matematické | Umožňují používat základní matematické funkce, které můžeme najít v kalkulačce, a i jiné, jako například náhodné číslo. |  |
| Text | Umožňují práci s textem (vstup a výstup) a obsahují funkce s ním spojené. |  |
| Seznamy | Dovolují v programu vytvářet, vypisovat, získávat a modifikovat záznamy v poli. |  |
| Barvy | Zde jsou připravené bloky s příkazy pro generování barev. Může se zde také vygenerovat náhodná barva anebo vytvořit barevný přechod. |  |
| Proměnné | Tyto bloky vytváří, nastavují a mění proměnné. |  |
| Funkce | Je možné zde vytvořit svou vlastní funkci a dále ji používat v kódu jako jeden blok. |  |

Tabulka 4 Popis příkazů v prostředí Blockly

Přímo Google podporuje výuku blokového programování pomocí aplikace Blockly Games. Blockly Games obsahuje řadu vzdělávacích her zaměřených na výuku dětí bez předchozích zkušeností s programováním. Hry je možné si stáhnout a používat je offline. Jsou zdarma,

tím snadno dostupné, a lze je jednoduše nasadit do výuky. Jednotlivé hry jsou navrženy tak, že se žák učí postupně zvládnout oblasti programování. Aplikace žáka vede k pochopení daných úkolů, radí mu, na co má kliknout a občas poradí i postup řešení. Úlohy většinou obsahují ve sloupci s bloky pouze takové, které jsou potřebné pro řešení daného úkolu. Při použití dostatečného počtu bloků ke splnění algoritmu ostatní bloky v daném sloupci přestanou být aktivní a nejdou použít. Vždy, když přibyde nový blok, aplikace zobrazí vysvětlivku, aby žák pochopil, k čemu slouží. Aplikace obsahuje kromě první seznamovací úlohy dalších 10 úrovní, jejichž obtížnost se stupňuje, a to nutí žáka vymýšlet složitější návrhy pohybů k dosažení cíle. Již po první úrovni je žákovi jasné, jak má postupovat. Mezi jednotlivými úrovněmi se může přepínat a přeskakovat. Vždy při běhu programu aplikace zvýrazňuje v danou chvíli probíhající bloky.

Při vstupu do online aplikace, nacházející se na stránkách https://blockly.games/ se nahoře zobrazí název, logo a odkaz na informace pro učitele. V pravém horním rohu je možná volba českého jazyka. Hlavní část stránky jsou velmi jednoduše po sobě jdoucí hry znázorněné pomocí kulatých ikonek, připomínající cestu od základu, přes použití bloků, až po vytvoření hry Rybník. Kulaté ikonky jsou ohraničeny šedou barvou, která po splnění hry zezelená a značí, na kolik procent je úkol splněn. Cesta obsahuje osm her s názvy odvíjejícími se od obsahu hry. Po úspěšném splnění úkolu vyskočí okno s gratulací vyřešení úkolu a ukáže, jak by vypadal daný program v JavaScriptu. Také se zeptá, zda chce žák pokračovat na další úroveň.

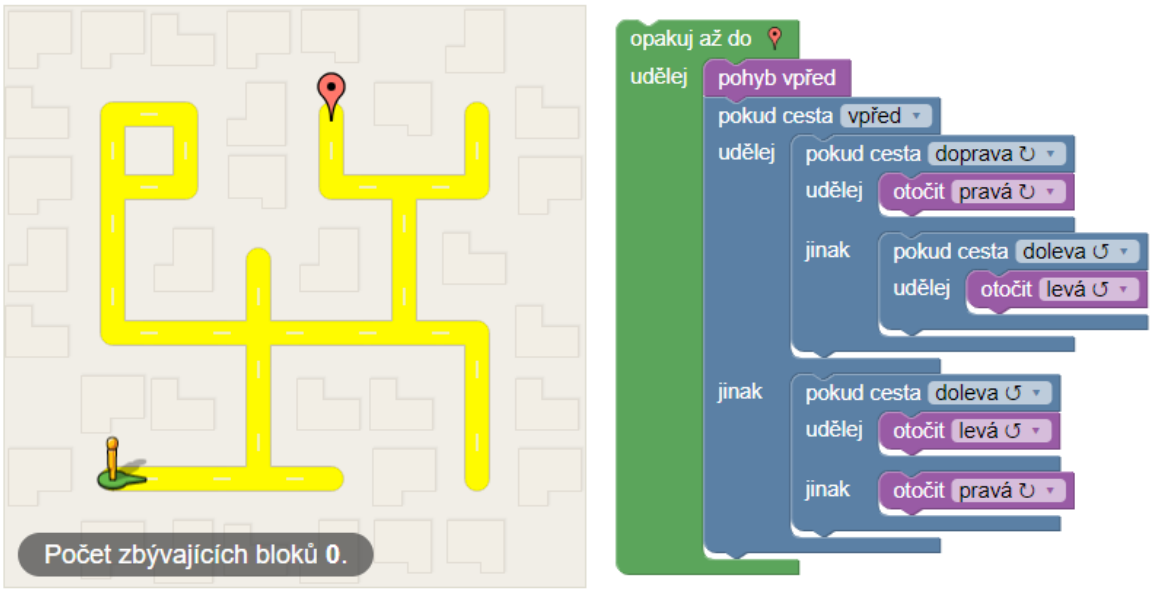
**Skládačka**

Tato hra je zaměřena na rychlý úvod do práce s bloky, kde se žák seznámí s jejich skládáním. Úkol je zpracovaný tak, že z počátku vyskočí okno se zadáním úkolu spolu s příkladem, jak úlohu vyřešit. Úkolem je přiřadit k blokům s názvy zvířat fotografii daného zvířete a přiřadit k bloku jejich vlastnosti. Plocha obsahuje čtyři bloky, k tomu čtyři fotografie a osm vlastností, všechny bloky do sebe zapadají. I během plnění úkolu je zde možnost přepnutí do programovacího jazyka, vedle této možnosti je tlačítko s nápovědou, sousedící s tlačítkem, které kontroluje zpracování úkolu. Po kliknutí na nápovědu se ukáže stejné okno se zadáním úkolu jako na začátku. Pokud úkol není vyřešen správně, tak se po kliknutí na tlačítko „Zkontrolovat odpovědi“ zobrazí dialogové okno popisující chybné bloky, a ty se v pracovní ploše zvýrazní. Jestliže je úkol splněn správně, tak se bloky rozpohybují do kola

a vyskočí dialogové okno s textem „Výborně! Všech 16 bloků je umístěno správně.“. Po stisknutí tlačítka „ok“ v okně se úkol ukončí a vrací se zpět do menu, kde se ikona tohoto úkolu ohraničí zelenou barvou, protože je úkol splněn.

**Bludiště**

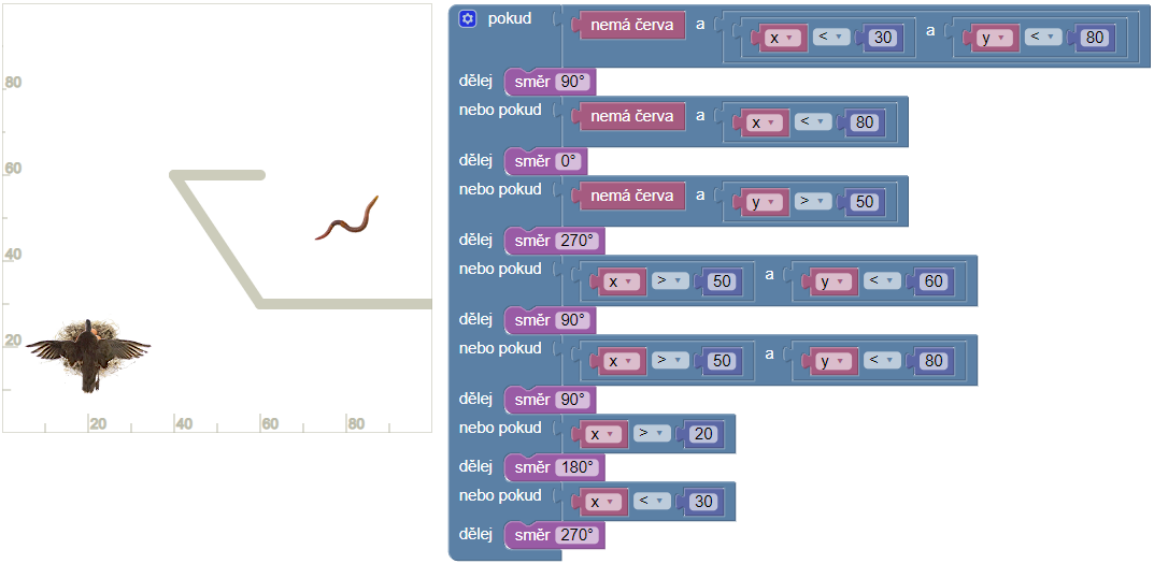
Bludiště učí žáka příkazy spojené s pohybem, ke kterým se zde postupně přidávají i podmínky a cykly. V okně pro výstup se nachází bludiště s panáčkem a vyznačeným cílem, kam se má panáček dostat. V první úrovni je třeba k dosažení cíle pouze dvakrát panáčka posunout vpřed. U sloupce s bloky se objeví okno s úkolem, jež tkví v tom, že se má panáček dostat z bludiště. Přetažením bloku do pracovní plochy aplikace ukáže na tlačítko „Spusť program“ a vede žáka ke spuštění algoritmu, po kterém zjistí, zda složil algoritmus správně. U posledního, a tedy nejtěžšího úkolu aplikace radí, čemu by se měl věnovat. Po uplynutí nějakého času a pokusů žáka stráveného nad posledním úkolem, vyskočí dialogové okno, že úkol je příliš těžký, a žák může přejít na další úkol a vrátit se k této úrovni později. Jestliže žák projde devíti úrovněmi, měl by zcela chápat podmínky, cykly a pohyby. Na obrázku 28 lze vidět obtížnost poslední úrovně. Vlevo je výchozí pozice panáčka a vpravo funkční algoritmus.




Obrázek 28 Poslední úroveň ve hře Bludiště, Zdroj: aplikace BlocklyGames

### Pták

Zde je v okně výstupu kachna, žížala a ptačí hnízdo s vejci. Úkolem je změnit směr ptáka, aby ulovil červa. V první úrovni je již blok předpřipravený na ploše a úkolem je změnit pouze úhel ptáka. V bloku se dá směr měnit číselně ve stupních anebo na jednotkové kružnici, jenž znázorňuje směr úhlu. Ve čtvrté úrovni přibyde do výstupního okna souřadnice osy X a v páté se k této ose přidá osa Y. Úloha je zaměřená na procvičení podmínek a cyklů, k naučení logických a aritmetických operátorů, a také k orientaci v rovině v kartézském prostoru souřadnic. Poslední úroveň je též velmi obtížná a znovu zde aplikace nabídne pokračování na další úkol a možnost se vrátit zpět. Obtížnost této úrovně můžete vidět na obrázku 29.




Obrázek 29 Poslední úroveň ve hře Pták, Zdroj: aplikace BlocklyGames

### Želva

Tato úloha se zabývá obkreslováním tvarů připravených v okně výstupu. Je zde možnost zrychlovat a zpomalovat spuštěný program, aby uživatel mohl lépe sledovat průběh algoritmu. Žák si zde procvičí cykly, prostorovou představivost a naučí se pracovat s perem a nastavováním barev. Do deváté úrovně má žák k dispozici tři záložky bloků a to želva, barva a smyčky. V desáté úrovni dostane žák volnou ruku ke kreslení a přibydou zde další záložky bloků a to logika, smyčky, matika, seznamy, proměnné a procedury. Pod výstupním oknem přibydou tlačítka, první je pro inspiraci, kde po kliknutí se nachází galerie toho,

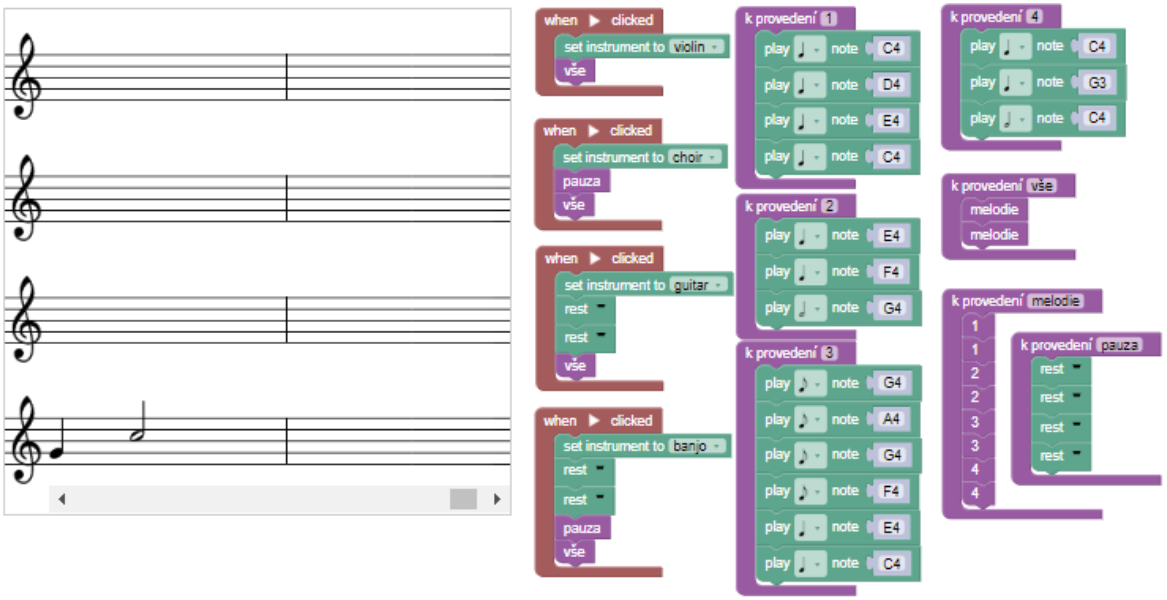
co nakreslili ostatní a druhé tlačítko slouží k nahrání žákovo výtvoru do databáze. Po kliknutí na jakýkoliv obrázek v galerii se může žák podívat na kód a inspirovat se.

**Film**

Film je úvodem do matematických rovnic, zprostředkovávajících tvorbu animací. Pro splnění první úrovně je zde předkreslený panáček, skládající se ze tří obdélníků (ruce, tělo) a jednoho kruhu (hlava). Ve výstupovém okně se nachází souřadnice os X a Y, sloužící pro kreslení tvarů. Úkolem je nakreslit pomocí bloků již předkresleného panáčka. Od druhé úrovně se animuje a postupně se přidávají nové funkce. Na konci je znovu přístup k fantazii a uživatel tvoří svůj film, který může poté publikovat.

**Hudba**

Tato hra učí skládat hudbu, ve výstupu je notová osnova pro zapisování not. Pro realizaci algoritmu tvořící melodii jsou bloky s notami. První úroveň se zabývá skládáním jednotlivých bloků not za sebou. Po spuštění programu se tyto noty přehrají v melodii. Kromě poslední úrovně, kde si žák může skládat, co chce, se řeší pouze jedna skladba s názvem Frere Jacques a na závěr ji žák složí celou i s různými nástroji jako housle, kytara a bendžo (viz obrázek 30). K ušetření výpisu jednotlivých not se zde učí využívat funkce, které je pak možno zavolat, a tím vytvořit přehlednější kód. Žák se zde seznámí se základními prvky skládání hudby, a to s notovou osou a vybranými notami, jež se vyučují v hudební výchově.




Obrázek 30 Poslední úroveň ve hře Hudba, Zdroj: aplikace BlocklyGames

### Rybník s instruktorem

Rybník s instruktorem je hra uvádějící hráče do textového programování, zabývající se potopením soupeře pomocí střelby. Úrovně se střídají mezi blokovým a textovým programováním, a tedy nejprve je zde ukázán příklad v blocích a posléze je úkolem zapsat kód ručně v jazyce JavaScript. Takže pokud si žák není jist, jak program zapsat v JavaScriptu, může přepnout do předešlé úrovně a inspirovat se. Poslední úroveň je opět ta nejtěžší a je k ní přidána dokumentace syntaxe jazyka JavaScript, kde jsou anglicky popsány logické, matematické funkce, cykly a funkce týkající se rybníku. V této úrovni je třeba využít cyklus `while` pro opakování střelby s plaváním a podmínku, kde je třeba řešit pohyb pomocí funkce, která nám zjišťuje polohu na ose X. Protože protivník utíká a může se střílet pouze do určité vzdálenosti. Řešení je možné vidět na obrázku 31.

```
1
2  while (true) {
3    if(getX() <= 50){
4    swim(45);
5    }else{
6      stop();
7    }
8    cannon(45, scan(45));
9
10 }
11
```

Obrázek 31 Poslední úroveň ve hře Rybník s instruktorem, Zdroj: aplikace BlocklyGames

### Rybník

Zde na rozdíl od hry Rybník s instruktorem přibydou 4 kachny a to hráč, věž, počtář a odstřelovač. Nejsou zde úrovně, ale je to otevřená soutěž o nejchytřejší kachnu. Žák si může naprogramovat, co chce, pomocí bloků anebo jazyka JavaScript.

## 3.4 ZÁSTUPCI UNPLUGGED PROSTŘEDÍ

Zástupců unplugged prostředí je značně méně, ale na trhu je spousta deskových her přímo či nepřímo podporujících rozvoj algoritmického myšlení, jen tolik v člověku neevokují podobnost s programováním. Tyto hry jsou například:

**Twin Tin Bots**

Tato strategická hra byla vydána společnosti Flatlined Games a je navržena pro dva až šest hráčů, kteří mají své roboty, které se programují ke shromažďování cenných krystalů. Hra je rozvržena do hexagonální mřížky (viz obrázek 32), kde jsou různě rozmístěné krystaly. Každý hráč má svou příkazovou základnu, kde skládá jednoduché instrukce pro své dva roboty jako: pohyb vpřed, zahnout doleva, zahnout doprava, vyzvednout krystal, uložit krystal nebo zapnout robota. Zásah do programu jednoho ze dvou robotů může být proveden pouze jeden. Následně se provedou programy obou robotů. Hra končí po určitém počtu vstupů krystalů do hry nebo až nějaký hráč dovrší určitého počtu bodů a samozřejmě hráč s nejvyšším počtem bodů vyhrává.[25]


Obrázek 32 Twin Tin Bots, Zdroj: [25]

**Wings of War**

Kolekce modulárních her Wings of War dříve vydávaná společností Nexus Editrice a nyní vydávaná společností Ares Games je zaměřena na simulaci vzdušného boje z 20. století. Hry obsahují kombinaci karetní, deskové a figurkové hry. Figurky jsou zde pro zmíněnou simulaci a jsou to miniaturní modely letadel. Karty zde symbolizují různé možnosti manévrování a typy poškození (viz obrázek 33). Manévrování souvisí s typem letadla a je

dobré se s ním nejprve seznámit. Pokud letadla opustí vymezený prostor, tak jsou považována za havarovaná. Cílem hry je přežít a nejlépe i sestřelit soupeře. [26]


Obrázek 33 Wings of War, Zdroj: [26]

**Robo Rally**

Společnost Wizards of the Coats vydala hru RoboRally, jež je zaměřená na programování robotů pro přežití v nebezpečné továrně a dosažení určitých bodů. Roboty se programují pomocí předem rozdaných náhodných karet určujících pohyb a směr. Hrací plocha znázorňuje továrnu s různými dopravními pásy, laserovými paprsky, jámami, drtiči a jinými překážkami. Hrací plocha s jejími prvky je vidět na obrázku 34. Cílem hry je uspořádat pět karet tak, aby se robot pohyboval po mapě bezpečně, aniž by se mu něco stalo a sebral co nejvíce bodů. Také si při pohybu může vypomoci různými zbraněmi a předměty, které škodí spoluhráčům. [27]


Obrázek 34 Robo Rally, Zdroj: [27]

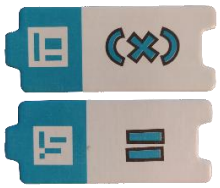
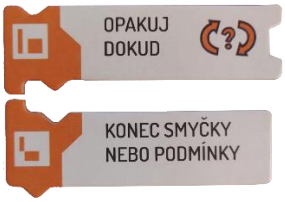
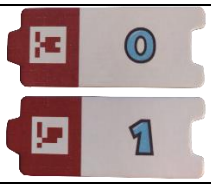
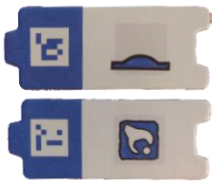
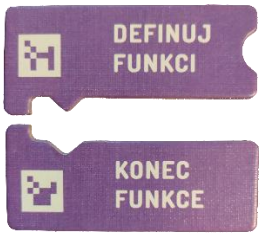
### 3.4.1 SCOTTIE GO!

Tato interaktivní hra byla vytvořena týmem polských učitelů ve spolupráci polského centra Poznańskie Centrum Superkomputerowo-Sieciowego. Zaměřuje se na vzdělávání zábavným způsobem kombinací fyzických prvků s digitální realitou aplikace Scottie Go! Forma hry by mohla být provedena plně v aplikaci, ale tvůrci se snaží mládež přimět se odpoutat od chytrých telefonů, se kterými tráví většinu času. Mobilní telefon nebo tablet se zde využívá pouze jako průvodce příběhem a pro kontrolu a výstup kódu. Pro tvoření algoritmu jsou zde k dispozici kartonové kartičky zapadající do sebe. Znázorňují funkci a speciální znak pro čtení kódu pomocí daného zařízení, podporující operační systémy Android, iOS i Windows.

U každé hry je důležitý příběh a v této hře je děj z roku 2030, který je o mimozemšťanovi se jménem Scottie, jehož loď se srazila se satelitem obíhajícím kolem Země. Trosky lodě jsou rozmístěné po celé Zemi a hráč má za úkol je najít, aby se Scottie mohl vrátit zpátky, a při tom čelit různým překážkám. Hra obsahuje 91 úkolů rozdělených do 10 modulů. Tyto moduly jsou různě rozmístěny po mapě světa.[28]

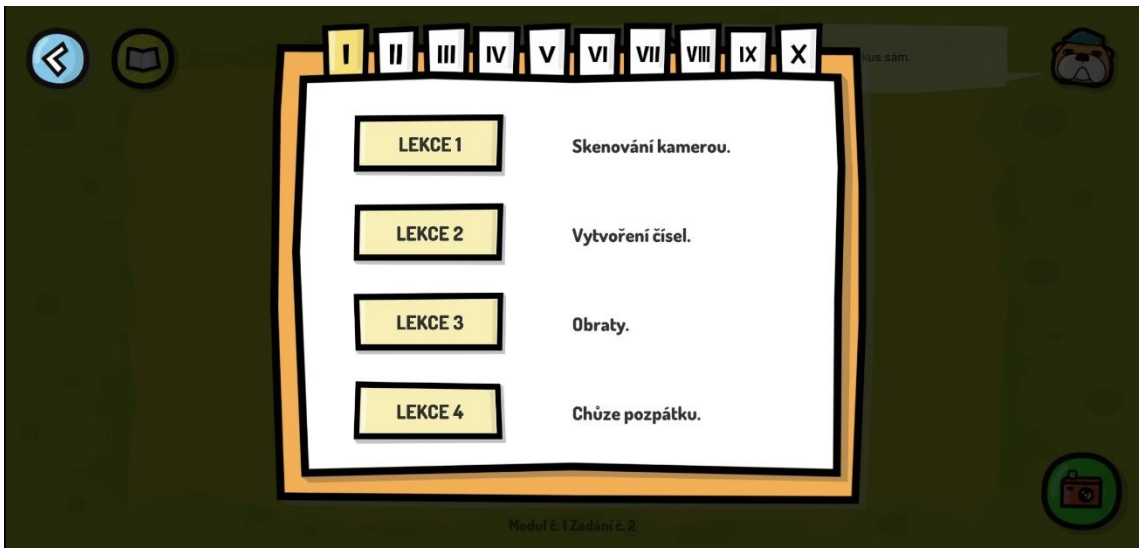
Balení hry obsahuje hrací desku, kde se kód skládá. Dále se zde nachází příručka, s vysvětlením bloků a použití této hry a návod, jak postupovat při instalaci aplikace na zařízení. Po rozbalení hry je třeba, aby si hráč bloky vyloupal z kartonových desek a složil si je do příslušných kategorií pro jejich přehledné hledání a následné použití.

Bloky jsou zde přehledně rozděleny do kategorií podle barev a nachází se v krabici s popisky. Po chvíli hráč intuitivně ví kam pro dané bloky sáhnout. Jejich rozdělení je znázorněno v tabulce 5.

| Skupiny | Příkazy | Příklady příkazů |
|---|---|---|
| | ▪ začátek<br>▪ konec |  |
| | ▪ kroky, skoky, zvedání<br>▪ skoky, zvedání, umisťování<br>▪ aktivování, proměnné, kreslení |  |
| | ▪ proměnné X a Y<br>▪ plus, mínus, rovná se<br>▪ zde, zepředu<br>▪ zprava, zleva, zezadu<br>▪ stále |  |
| | ▪ začátek podmínky/smyčky<br>▪ konec nebo rozšíření podmínky/smyčky<br>▪ přeruš |  |
| | ▪ čísla 1-2<br>▪ čísla 3-5<br>▪ čísla 6-9, 0 |  |
| | ▪ prázdné, překážka, vyvýšení<br>▪ znak<br>▪ předmět k sebrání, pole akce |  |
| | ▪ stvoření funkce<br>▪ vyvolání funkce<br>▪ funkce: A<br>▪ funkce. B<br>▪ funkce: C |  |
| | ▪ ovládání postav |  |

Tabulka 5 Popis příkazů v prostředí Scottie Go!

Po spuštění aplikace se objeví menu, kde si hráč vytvoří profil se jménem. Po přihlášení se pod svým jménem uvidí hezky zpracovaný obrázkový příběh, znázorňující havárii mimozemšťana. Po skončení animace se objeví mapa světa, přes kterou se hráč dostává do hry. Na této mapě se nachází v levém horním rohu dvě ikonky, jedna je nastavení a druhá obsahuje deset úrovní (viz obrázek 35), ve kterých se nachází lekce související s hrami. Lekce jsou přehledně vypracovány tak, že se v levé části ukáže animace s popisem pro danou problematiku a v pravé části je znázorněn algoritmus pomocí karet.



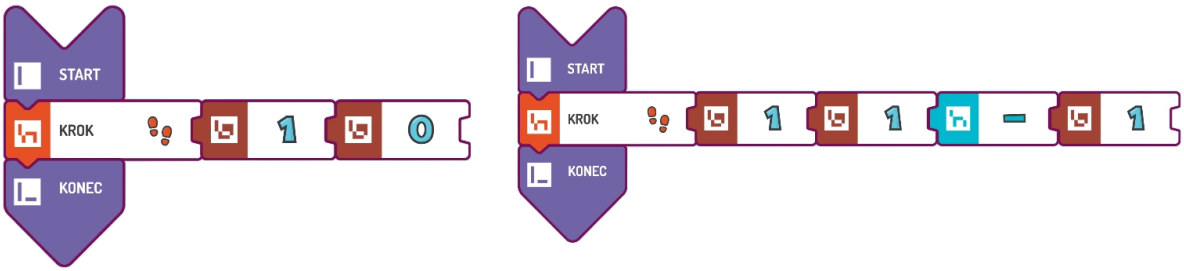

Obrázek 35 Lekce, Zdroj: aplikace Scottie Go! Edu

Algoritmy se zde tvoří tak, že musí mít začátek a konec a mezi těmito bloky pak jednotlivé příkazy, jinak program neproběhne.

### Úroveň I

**Lekce 1** seznamuje uživatele se skenováním kódu. Popisuje, jak má postupovat při skenování a že po správném naskenování má stisknout tlačítko play pro spuštění načteného programu.

**Lekce 2** vysvětluje, jak se tvoří dvouciferná čísla. Vytvářejí se spojením dvou karet s čísly nebo kartami s čísly společně s kartou pro součet nebo rozdíl. Obě možnosti jsou znázorněny na obrázku 36.




Obrázek 36 Tvorba čísel, zdroj: aplikace Scottie Go! Edu

**Lekce 3** popisuje pohyb do stran pomocí karet „OBRAT VLEVO“ a „OBRAT VPRAVO“.

**Lekce 4** se zabývá couváním, které se tvoří příslušnými kartami.

### Úroveň II

**Lekce 1** uvádí blok „ZVEDNI“, pomocí kterého může Scottie zvedat různé předměty, jako například hlemýždě. Pokud se tato karta použije na poli, kde nic není, tak je to považováno za chybu.

**Lekce 2** předvádí, že se mimozemšťan může setkat s překážkou a že jí může přeskočit pomocí karty „SKOČ“.

### Úroveň III

**Lekce 1** zobrazuje zkrácení kódu pomocí cyklů. Radí kam karty znázorňující opakování vložit s čísly pro určení počtu opakování.

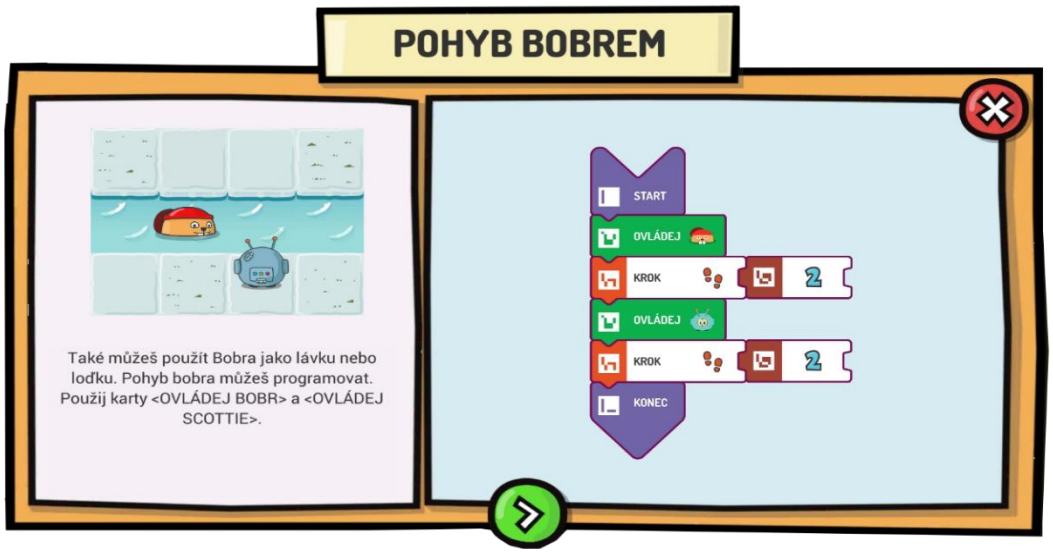
**Lekce 2** upozorňuje na to, že některé části v programu se nemusí opakovat a že se tyto části mají umístit mimo cyklus.

**Lekce 3** vysvětluje použití vnořeného cyklu v jiném cyklu, který nazývají zahnízděným.

### Úroveň IV

**Lekce 1** seznamuje hráče s tím, že může přecházet po bobrech jako po mostech.

**Lekce 2** rozvíjí lekci 1 tím, že bobr se dá ovládat a nemusí být použit pouze jako lávka, ale i jako loďka (viz obrázek 37).




Obrázek 37 Programování více postav, Zdroj: aplikace Scottie Go! Edu

**Lekce 3** ukazuje, že pro rychlejší pohyb je vhodnější plout na bobrovi.

**Lekce 4** vysvětluje, že pokud se Scottie nachází na bobrovi, kopíruje jeho pohyb.

### Úroveň V

**Lekce 1** uvádí blok „POUŽIJ“, pomocí kterého mimozemšťan například rozsvítí maják, přičemž musí stát na sousedním poli a čelem k němu.

**Lekce 2** radí žákovi, jak použít kládu, aby mohl přejít přes vodu. Pro položení klády přes vodu musí Scottie stát na políčku před vodní trhlinou a použít karty „ZVEDNI“ a „UMÍSTI“.

**Lekce 3** doporučuje používat ledové kry pro přechod přes vodu. Tyto kry se dají posouvat pomocí bobrů.

### Úroveň VI

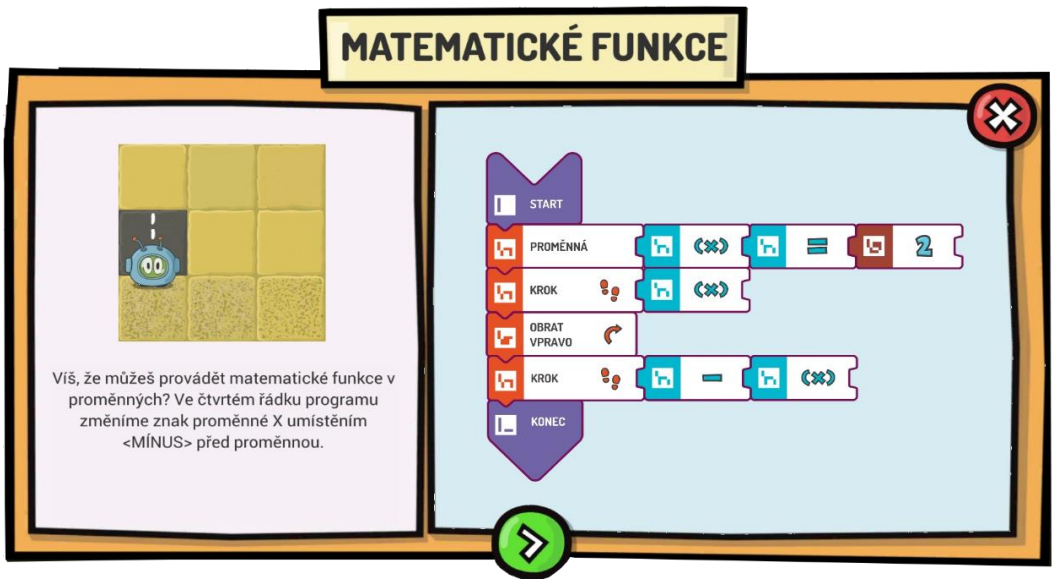
**Lekce 1** znázorňuje cyklus začínající kartou „OPAKUJ DOKUD“ a ukončen kartou „KONEC SMYČKY NEBO PODMÍNKY“. V cyklu je nejprve kontrolovaná podmínka, a pokud je podmínka splněna, tak se vykonají příkazy uvnitř cyklu.

**Lekce 2** rozvíjí předešlou lekci a poukazuje na možnosti podmínek v cyklu.

### Úroveň VII

**Lekce 1** vysvětluje pojem proměnná přirovnáním k nádobě, do které se dají vkládat různá data a počítač může data prohlížet a měnit. Postupně ukazuje příklady, jak proměnných využít a poukazuje na to, že se nejprve musí naplnit a dále použít v kódu.

**Lekce 2** se zabývá matematickými funkcemi s proměnnými, jak lze vidět na obrázku 38.




Obrázek 38 Matematické funkce s proměnnými, Zdroj: aplikace Scottie Go! Edu

### Úroveň VIII

**Lekce 1** uvádí žáka do podmínek. Přirovnává podmínku k rozhodování se v běžném životě takto: „Když prší, oblékneš si mikinu a bundu. Když neprší, tak jen mikinu.“ Je zde také vysvětleno a znázorněno, že je někdy třeba podmínku použít uvnitř cyklu.

### Úroveň IX

**Lekce 1** rozvíjí předešlou úroveň o druhou větev podmínky sloužící pro stav, kdy podmínka splněna není.

**Lekce 2** prohlubuje předešlou lekci o příkazy s více podmínkami. Také zdůrazňuje, že pokud se nějaká z podmínek splní, další podmínky se nekontrolují a pokračuje se v programu dál.

### Úroveň X

**Lekce 1** poukazuje na to, že pokud se opakují několikrát po sobě některé kroky v kódu, tak je program zbytečně dlouhý a na řadu přichází použití funkcí.

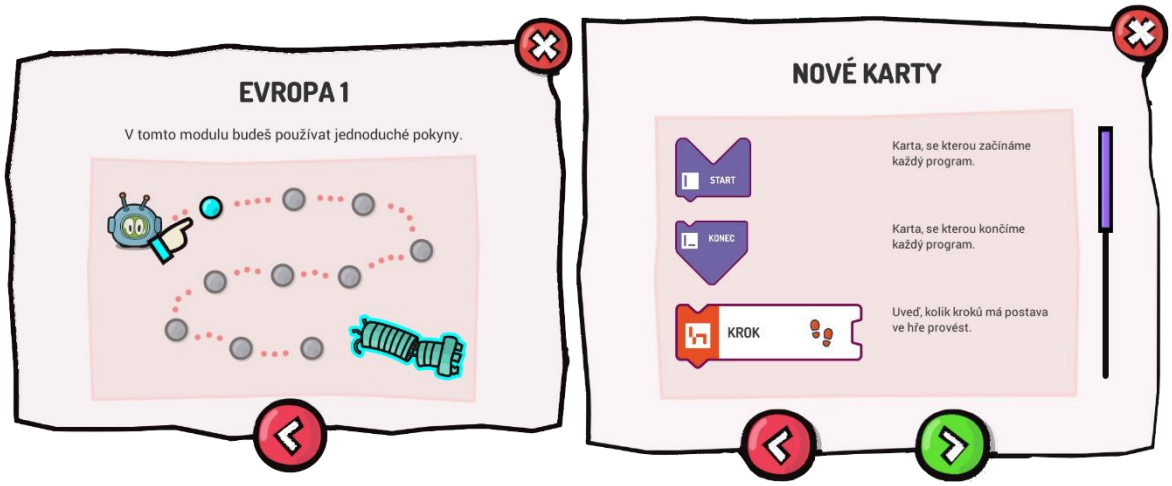
**Lekce 2** seznamuje s kartou „KRESLI“, pomocí které může Scottie kreslit cestičky. Stejná karta zapíná a vypíná režim kreslení.

Na mapě se nachází 10 modulů rozmístěných různě po mapě, obsahujících několik her. Do těchto her se vstupuje postupně a jsou znázorněny jako cesta, kde jsou jednotlivé kroky znázorněny pomocí koleček, vedoucích k věci, kterou Scottie chce získat. Tato kolečka jsou po splnění modrá nebo zlatá. Pokud jsou zlatá, uživatel získal plný počet bodů, jestliže jsou modrá, hráč splnil úkol, ale ne nejefektivněji. Jsou zde tři úrovně úspěšnosti, které se nachází pod danými kolečky v podobě hvězdiček. Pokud jsou jednotlivé kroky splněny, uživatel se do nich může vrátit a popřípadě je opravit nebo jen zkusit znovu. Ve hře se nachází obdélníková plocha rozdělena na 6 x 8 čtverečků, po kterých se mimozemšťan pohybuje a na kterých se objevují různé překážky a předměty. Díky těmto čtverečkům může hráč snadno počítat kroky a tím jednoduše sestavit program.

Moduly obsahují několik úrovní, kde se zvyšuje náročnost jejich řešení. Stejně jako do jednotlivých úrovní se do modulů postupuje podle obtížnosti a jsou seřazeny takto:

1. Evropa 1

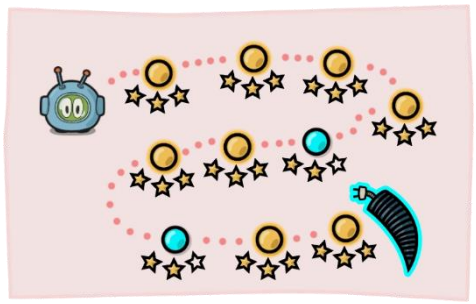
   První modul se zabývá jednoduchými instrukcemi, jako jsou kroky a otáčení. Na obrázku 39 jsou vidět dialogová okna prvního modulu. Nejprve se zobrazí okno s názvem „Nové karty“, kde aplikace seznamuje hráče s kartami, které bude používat. Po prohlídnutí seznámení se s bloky se objeví okno s názvem „Evropa 1“, kde vyobrazena cesta k součástce, kterou chce Scottie získat pro spravení své lodě. Modré kolečko znázorňuje aktivní úkol. Při vstupu do ostatních modulů se nachází podobná dialogová okna.




Obrázek 39 Dialogová okna v prvním modulu, Zdroj: aplikace Scottie Go! Edu

2. Evropa 2

Druhou získanou součástku je možné vidět na obrázku 40, kde je také vidět hodnocení jednotlivých úkolů pomocí hvězdiček. Zde Scottiemu přibývají bloky určující sbírání předmětů a skákání.



Obrázek 40 Absolvovaná cesta k druhé součástce, Zdroj: aplikace Scottie Go! Edu

3. Jižní Amerika

V tomto modulu se hráč učí používat cykly od jednoduchých po vnořené. Na obrázku 41 je zadání posledního úkolu v tomto modulu. Hráč zde používá bloky z předešlých modulů plus nové cykly. Zde má za úkol sebrat co nejefektivněji všechna ptačí pera. Na obrázku je možné vidět, že vlevo nahoře se nachází ikony pro menu a pro lekce. Vpravo dole je ikona fotoaparátu, která slouží pro skenování kódu.


Obrázek 41 Poslední úkol v modulu Jižní Amerika, Zdroj: aplikace Scottie Go! Edu

4. Severní Amerika 1

   Programování se zde ztěžuje tím, že přibývá možnost ovládání jiných postav a to bobrů. Hráč musí používat bloky pro sběr a kroky. Úkolem je zde použít cyklus a bobra tak, aby Scottie získal veškeré zlato na hrací ploše.

5. Antarktida

   Oproti předešlému modulu je zde třeba řešit zvedání a pokládání předmětů, aby se Scottie mohl dostat k majákům a rozsvěcovat je. Přibývají zde pro snazší cestování i další bobři, které hráč může ovládat.

6. Asie

   Zde přibývají podmíněné cykly s bloky „opakuj dokud“ a s tím přibývají bloky určující překážky, předměty pro sběr a prázdná místa a také zde přibývají bloky pro určení, z jakého směru má Scottie zjišťovat, kde se dané věci nachází.

7. Austrálie 1

   Hráč se zde učí pracovat s matematickými funkcemi, ukládat hodnoty do proměnných a používat je pro úsporu kódu. Také si zde vytváří cesty pro rychlejší cestování.

8. Austrálie 2

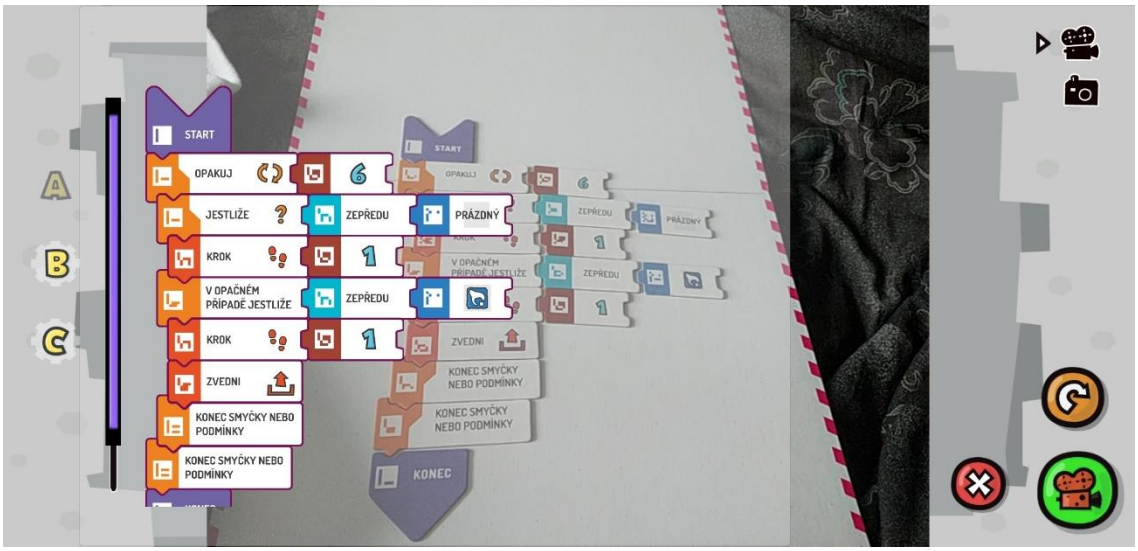
   Tento modul se zabývá podmínkami a přibývá blok s názvem „POLE AKCE“. Scottie zjišťuje, kde může sázet rostliny. Pokud najde místo, kde je možné zasadit, tak je zasadí.

9. Severní Amerika 2

   Tento modul rozšiřuje podmínky a zabývá se jejich větvením. Používá se zde blok „V OPAČNÉM PŘÍPADĚ“ a blok pro přidání další podmínky. Na obrázku 42 je vidět toto větvení, které probíhá v cyklu. Jestliže je před Scottiem prázdno, tak pokračuje o krok vpřed. Pokud se před ním nachází předmět ke sběru, tak jde o krok vpřed a příslušnou věc sebere a tato logika se díky cyklu opakuje šestkrát. Na tomto obrázku je také vidět, že se vlevo nachází písmena A, B, C, obsahující uložené funkce a po kliknutí na ně je možné je zobrazit. Tyto funkce se ale hráč učí až v následujícím modulu. Vedle funkcí je vidět načtený kód v aplikaci pomocí kamery spolu s obrazem z kamery. Vpravo nahoře se nachází dvě ikony: kamera a fotoaparát. Tyto ikony slouží k načítání programu do aplikace. Kamerou je snazší načíst delší kód, ale i s fotoaparátem je dlouhý kód

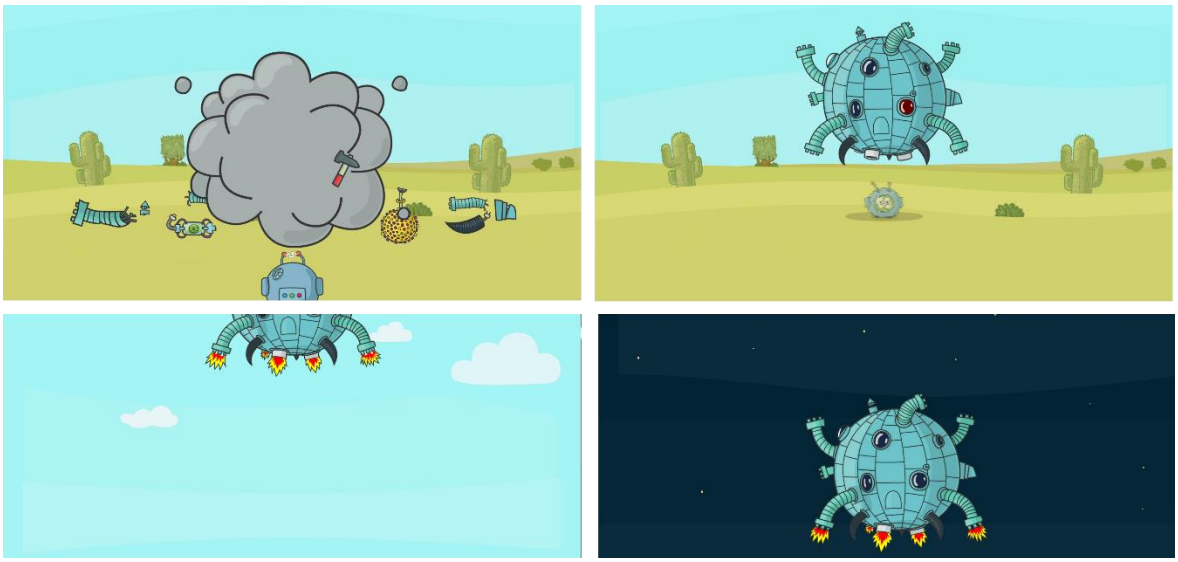
možné načíst, záleží na uživateli, která cesta k načtení kódu je mu pohodlnější. Pod těmito ikonami se nachází tlačítka pro znovunačtení, pro zrušení načítání a tím vrácení se zpět do úkolu a tlačítko pro dokončení skenování a spuštění příslušného programu.

10. Afrika




Obrázek 42 Načítání kódu do aplikace, Zdroj: aplikace Scottie Go! Edu

Poslední modul se zabývá vytvářením pořádku v parku a znovu sázením rostlin a budováním cest pomocí bloků definujících funkce. Po posledním úkolu Scottie získá finální potřebnou součástku pro spravení lodě, díky čemuž pak odlétá domů (viz obrázek 43).



Obrázek 43 Odlet Scottieho domů, Zdroj: aplikace Scottie Go! Edu

Tato hra má u žáků velký úspěch, což je možné vidět na výsledcích výzkumu, který proběhl v Makedonii na 120 žácích z různých základních škol, kteří používali Scottie Go! ve výuce

podobu dvou měsíců. Po skončení této dvouměsíční výuky byli žáci dotázáni na několik subjektivních názorů, které můžete spolu s výsledky vidět v tabulce 6. [29]

| | Silně souhlasím | Souhlasím | Neutrální | Nesouhlasím | Rozhodně nesouhlasím |
|---|---|---|---|---|---|
| Prostředí hry se snadno používá. | 103 | 18 | / | / | / |
| Bylo jednoduché se hru naučit hrát. | 109 | 11 | 1 | / | / |
| Bavil jsem se při hraní hry Scottie Go! | 104 | 16 | 1 | / | / |
| Hra mi přijde atraktivní. | 102 | 18 | 1 | / | / |
| Mohu ji hrát společně s přáteli. | 111 | 8 | 2 | / | / |
| Je zajímavé se učit pomocí Scottie Go! | 107 | 14 | / | / | / |
| Tato cesta výuky je velmi poutavá. | 99 | 21 | 1 | / | / |
| Líbí se mi tento způsob učení. | 103 | 14 | 4 | / | / |
| Jsem rád za tuhle zkušenost. | 100 | 20 | 1 | / | / |

Tabulka 6 Výsledky výzkumu, Zdroj: [29]

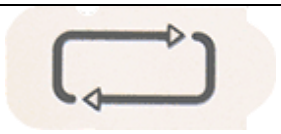
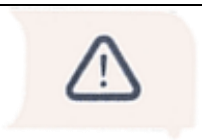
Scottie Go! zlepšuje schopnost analyticky a logicky myslet, vyvíjí algoritmickou intuici. Učí, jak řešit složité problémy a jak pracovat ve skupině, podporuje rozvoj kompetencí ve výuce programování v raných fázích vzdělávání.

### 3.4.2 OSMO CODING

Společnost Tangible Play vytváří řadu her s názvem Osmo pro inspiraci nejmladších k praktickému využití moderních technologií. K dispozici je primárně v angličtině, ale podporuje i jiné jazyky (čeština mezi ně bohužel nepatří). Doporučený věk je 5 až 12 let. Aplikace s hrami jsou k dispozici pouze pro iOS a Fire tablety. Osmo coding je zaměřeno na programování pomocí plastových magnetických bloků. Společnost uvádí, že bloky byly ovlivněny více než 50 lety vysokoškolského výzkumu v oblasti kódování a byly navrženy tak, aby poskytly uživateli nejlepší možný úvod do světa programování. [30]

Osmo coding obsahuje základní jednotku, plastový nástavec se zrcadlem a plastové magnetické kartičky, které do sebe zapadají. Do základní jednotky se umístí tablet s danou aplikací na výšku a na fotoaparát daného zařízení se přiloží plastový nástavec se zrcadlem, aby aplikace mohla číst bloky ze stolu. Tato realizace je provedená proto, aby žák viděl výstup kódu v reálném čase. Bloky související s určováním směru jsou vyrobeny tak, aby se směrovou šipkou mohlo otáčet podle směru, který chceme, aby blok znázorňoval. Díky tomu Osmo obsahuje méně bloků, které jsou odlišeny barvami, a jejich funkcionalitu si lze prohlédnout v tabulce 7.

Hry nejprve hráče interaktivně seznámí s použitím bloků a s prostředím aplikace. Při tvorbě kódu je důležité bloky skládat před tablet, aby je tablet mohl snímat. Algoritmus klasicky začíná shora a pro jeho spuštění je důležitý blok s tlačítkem play, který se dává na konec kódu. Po stisknutí tohoto tlačítka se spustí vytvořený program, jehož výstup je možné pozorovat, jak je již zmíněno právě na tabletu. Při tvorbě kódu zařízení snímá v reálném čase, a proto je možné vidět hráčem tvořené instrukce. To hráči může velmi urychlit proces tvoření programu.

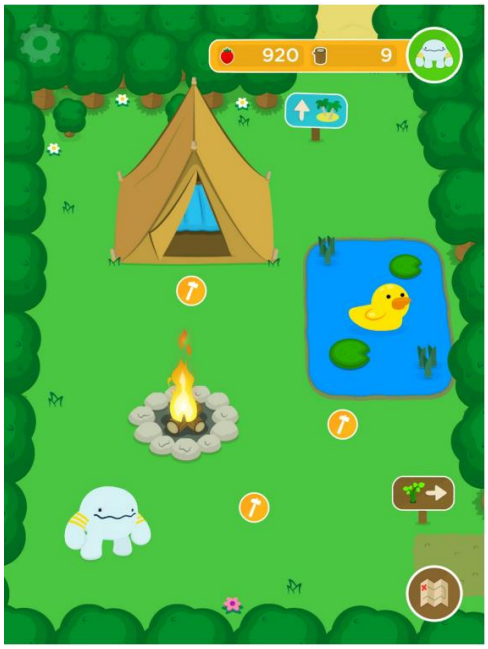
| Vysvětlivka příkazů | Příkazy |
|---|---|
| Slouží pro spuštění programu; dává se na konec vytvořeného kódu. |  |
| Blok pro pohyb; pokud je třeba změnit směr, stačí otočit pohyblivou šipkou. |  |
| Skákání, u kterého platí, že pokud je třeba měnit směr skoku, stačí otočit šipkou. |  |
| Pomocí tohoto bloku je umožněno Awbiemu používat předměty. Také se zde dá měnit směr pomocí kolečka. |  |
| Po určitém počtu sebraných hvězdiček může Awbie použít tento magický blok, který mu až do vyčerpání hvězdiček umožňuje získávat větší počet jahod. |  |
| Tento blok je určen pro opakování; pokud se použije takto, tak se akce opakuje jednou. |  |
| Pomocí tohoto bloku se může vytvořit cyklus s podmínkou. Anebo pouze podmínka pro ošetření toho, aby postava například nespadla do vody. |  |
| Se hrou Coding Jam přibývá blok pro pauzu. |  |
| Tento blok má ještě dvě varianty a slouží pro uložení delšího kódu a následnému volání. Funguje jako v programování funkce. |  |
| Bloky s čísly jsou zde pro určení počtu opakování. |  |
| Tyto dvě postavy určují začátek kódu dané postavy ve hře Coding Duo |  |

Tabulka 7 Vysvětlení příkazu v prostředí Coding Osmo

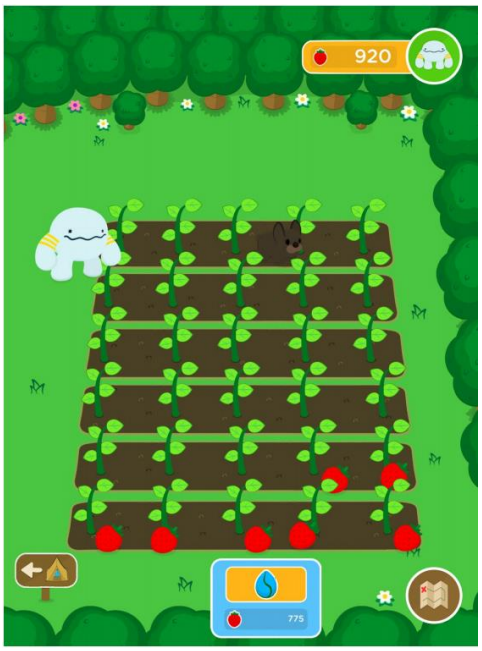
Osmo coding hry jsou rozděleny do třech kategorií:

**Začátečník – Coding Awbie:**

Tato hra představuje dětem koncepty logiky a řešení problému pomocí skládání bloků do kroků, jež rozpohybují postavu jmenující se Awbie. Postava žije uprostřed lesa, miluje jahody, díky kterým si může kupovat a vylepšovat svojí životní úroveň. Awbie začíná se spacím pytlem a ohněm. Jak je vidět na obrázku 45 může vlastnit jezírko a má své vlastní pole (viz obrázek 44), kde může pěstovat své oblíbené ovoce.



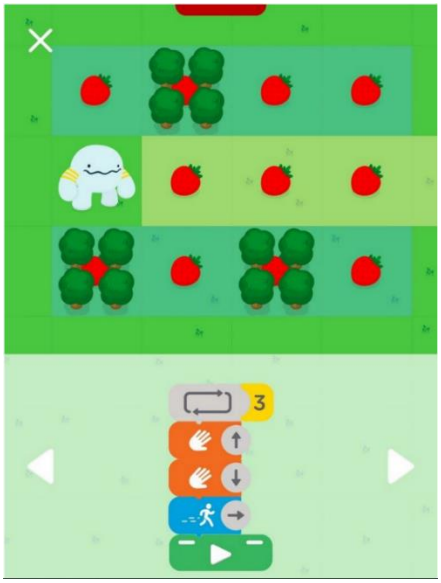

Obrázek 44 Awbieho bydlení, Zdroj: aplikace Osmo Coding Awbie




Obrázek 45 Awbieho pole, Zdroj: aplikace Osmo Coding Awbie

Vpravo dole se nachází mapa, prostřednictvím které se Awbie dostává do hry, kde plní úkoly, které se objevují v průběhu hry. Primárně má Awbie sesbírat jahody, jejichž sazenice získává při pohybu na mapě, a čím lepší program hráč vymyslí, tím více jahod dostane. Vyššího počtu jahod se docílí, když se použije magický blok, který navyšuje počet sebraných jahod. Tato magická síla se objeví v levé části displeje ve sloupcovém ukazateli, ve kterém ubývá fialový obsah a s tím i magická síla. Awbieho dobrodružství zahrnuje pět různých map, obsahujících několik úrovní, do kterých hráč může vstoupit, pokud jsou odemčené. Po klepnutí na danou úroveň se zobrazí dosažené skóre v úrovni a také počet shromážděných věcí.

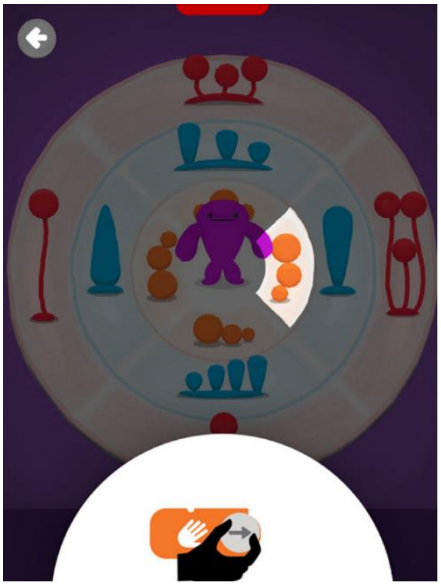
V aplikaci je výstup znázorněn na čtverečkované mapě. Tyto čtverečky symbolizují počty kroků, které má postava absolvovat pro dosažení cíle (viz obrázek 46). Mapa obsahuje mnoho grafických prvků, symbolizujících překážky a věci pro sběr. Na mapě jsou zvýrazněné kroky podle hráčem sestaveného programu v podobě zesvětlených čtverečků. Hrací pole je velmi rozsáhlé a je možnost si jej prohlížet pomocí přejíždění po obrazovce prstem, aby se mohl kód tvořit několik kroků dopředu.




Obrázek 46 Sestavený program s jeho výstupem, Zdroj: aplikace Osmo Coding Awbie

**Středně pokročilý – Coding Jam:**

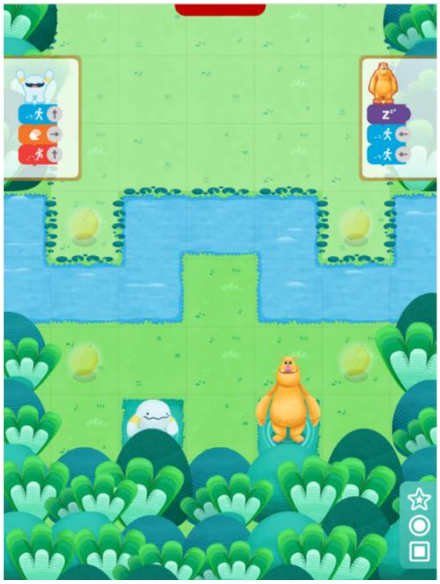
Zde si hráči mohou složit svou vlastní hudbu a rozvinout si smysl pro rytmus. Hra je plná hudebních světů, které se dají odemknout a prozkoumat. Odemykání jednotlivých světů spočívá ve vytváření jamů a dokončení úrovní v tréninkovém režimu. Každý hudební svět má jedinečné postavy a hudební objekty, které mají různé zvuky a programují se pomocí bloků. Své jamy může hráč sdílet s komunitou, kde se mohou lidé i navzájem hodnotit pomocí srdíček. Oproti Coding Awbie zde přibývá takzvaný odpočívací blok, který se používá proto, aby mohly být v hudbě pauzy. Pro nastavení delší pauzy je vhodné použít blok pro cyklus. Pro náročnější skladby je potřeba využít bloky pro ukládání a volání funkcí. Skládání hudby je zde jednoduché a funguje tak, že postava stojící uprostřed hracího pole se dotýká věcí podle určování směru na daném bloku. Každý předmět v okolí postavy má jiný tón či zvuk a je zde možnost být velmi kreativní.



Obrázek 47 Prostředí tvorby hudby,
Zdroj: aplikace Osmo Coding Awbie

**Pokročilý – Coding Duo:**

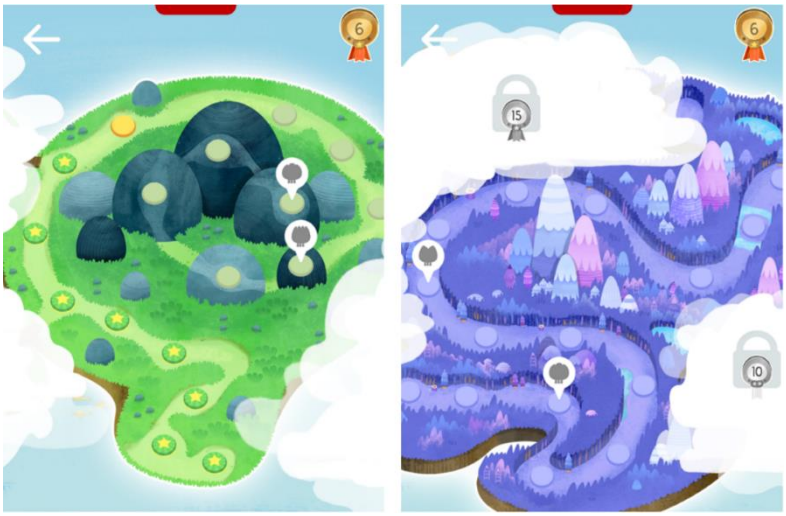
Zde čelí dobrodružství Awbie spolu s přátelským monstrem, které se jmenuje Mo. Tato hra využívá všechny bloky, které byly v předešlých hrách, ale přibývají zde bloky právě těchto dvou přátel, a tím je zde umožněno programovat obě postavy naráz a žák se je zde naučí synchronizovat. Jakmile hráč dokončí výuku v tutoriálu, odemkne se ostrov, který je třeba prozkoumat a vyřešit zde hádanky, najít mazlíčky a nasbírat angrešt a odznaky.



Obrázek 48 Coding Duo, Zdroj:
aplikace Osmo Coding Duo

V aplikaci při výběru her se nachází vzducholoď a dva ostrovy. Vzducholoď je nejen vstupní tutoriál, ale i domov vědce a zachráněných mazlíčků. Také se zde nachází nashromážděné

angrešty, za které je možné nakupovat různé ozdoby a předměty pro zachráněné mazlíčky. Jako další místa jsou zde ostrovy Awbie's Island a Mo's Island (viz obrázek 49).



Obrázek 49 Ostrovy s úrovněmi hry, Zdroj: aplikace Osmo Coding Duo

## 3.5 POROVNÁNÍ JEDNOTLIVÝCH PROSTŘEDÍ

Pro zahrnutí daného prostředí do výuky je nejspíše nejvýhodnější Scratch, protože má velmi rozsáhlou komunitu nejen běžných uživatelů, ale i učitelů, kteří společně komunikují a vytvářejí úkoly zaměřené na programování. Scratch obsahuje VELA aktivity, které pomáhají rozvíjet algoritmické myšlení. Je zde možnost být kreativní, podporuje češtinu a také je zdarma. V Blockly Games oproti Scratch chybí plná čeština a je zde menší možnost kreativity, protože žák většinou musí striktně sledovat zadání. Výhodou ale je, že učitel nemusí vymýšlet, nebo vyhledávat jednotlivé úlohy pro vyučované programovací konstrukty. Ty jsou zde již připraveny a pro objasnění problematiky jsou dostačující. Obě tato softwarová prostředí jsou přehledně zpracována, jsou zdarma a stačí k nim pouze internetový prohlížeč, takže zahrnutí do výuky informatiky by mělo být jednoduché.

U zmíněných unplugged prostředí je největší nevýhodou jejich cena. Prostředí Scottie Go! je stejně drahé jako jednotlivé hry Osmo Coding. Osmo přináší další investice, a to nákup zařízení, protože je třeba mít zařízení s operačním systémem iOS nebo Fire. Scottie Go! je k dispozici na Android i na iOS, takže se nemusí řešit operační systém a s tím drahou investici. V dnešní době již žáci často vlastní chytré telefony, které by mohli ve výuce a při práci s ním využít. U Osma je výhodou, že je mnohem rozsáhlejší, obsahuje více úkolů a aktivit. Obě prostředí evokují v žácích pocit, že hrají deskovou hru díky skládání bloků ve formě kartiček. Žáci jsou mnohem motivovanější, protože je hra spjata s chytrým zařízením.

Na obrázku 50 je porovnání ohledně oblastí programování u softwarových aplikací Scratch s Blockly Games a unplugged aplikací Scottie Go! s Osmo coding. Žlutá půl hvězda symbolizuje, že danou věc v tabulce nesplňuje stoprocentně. U Scottie Go! se hvězda nachází u operátorů, protože zde nejsou přímo operátory jako znaménka větší, menší a nerovná se.

Oblasti programování

| | SCRATCH | BLOCKLY GAMES | SCOTTIE GO! | OSMO |
|---|---|---|---|---|
| Cykly | ✔ | ✔ | ✔ | ✔ |
| Operátory | ✔ | ✔ | ½ | ✖ |
| Podmínky | ✔ | ✔ | ✔ | ✔ |
| Pole | ✔ | ✔ | ✖ | ✖ |
| Proměnné | ✔ | ✔ | ✔ | ✖ |
| Funkce | ✔ | ✔ | ✔ | ✔ |

Obrázek 50 Porovnání aplikací v oblastech programování

Pro smysluplné nasazení do výuky je vhodné porovnat vlastnosti prostředí jako například, zda jsou dostupná zdarma a na jakých zařízeních dané aplikace běží (viz obrázek 51). U Scratch se dá polemizovat nad tím, zda má výukové lekce zaměřené přímo na oblasti programování. Při porovnání s Blockly Games lze říct, že zde na první pohled není přímá linie na návaznost oblastí programování. U Blockly Games není možnost tvoření vlastního kódu s vlastním výstupem, v některých úrovních má uživatel volnější ruku, ale pořád je usměrněn na dané téma.

Vlastnosti prostředí

| | SCRATCH | BLOCKLY GAMES | SCOTTIE GO! | OSMO |
|---|---|---|---|---|
| Dostupné zdarma | ✔ | ✔ | ✖ | ✖ |
| Obsahuje výukové lekce zaměřené na oblasti programování | ◢ | ✔ | ✔ | ✔ |
| Možnost tvoření vlastního kódu | ✔ | ◢ | ✖ | ✖ |
| Dostupnost v češtině | ✔ | ◢ | ✔ | ✖ |
| Dostupnost iOS | ✔ | ✔ | ✔ | ✔ |
| Dostupnost Android | ✔ | ✔ | ✔ | ✖ |
| Dostupnost desktop prohlížeč | ✔ | ✔ | ✖ | ✖ |

Obrázek 51 Porovnání aplikací v jejich vlastnostech

# 4 DOTAZNÍKOVÉ ŠETŘENÍ

Tato kapitola popisuje průzkum na základních školách, který se zaměřuje na obeznámenost s blokovým programováním a jeho vývojovými prostředími a také jejich využitím. Bylo vybráno 200 základních škol z plzeňského kraje, které byly osloveny pomocí e-mailu s vloženým odkazem na elektronický dotazník.

## 4.1 DOTAZNÍK

Tento dotazník je určen pro učitele informatiky 2. stupně na ZŠ. Slouží ke zmapování aktuálního stavu využití softwarového a unplugged prostředí pro výuku blokového programování. Dotazník je vytvořen pomocí Google Forms proto, že je volně dostupný, lze snadno vytvořit a vyhodnocuje odpovědi do přehledných grafů.

Dotazník obsahuje několik podmínek, podle kterých se pohybuje v devíti sekcích. Tyto sekce jsou jedna po druhé (bez ohledu na postup procházení) popsány v podkapitole „Sekce dotazníku“. Jak dotazníkem prochází respondent je znázorněno v Příloha I.

### 4.1.1 SEKCE DOTAZNÍKU

U jednotlivých sekcí se vždy nachází odkaz na přílohu, kde je možné vidět, jak daná sekce vypadá.

**1)** Příloha II obsahuje první sekci dotazníku „Víte, co je blokové programování?“. Tato otázka je vytvořena proto, aby přizpůsobila další dotazy danému respondentovi. Bylo by zbytečné se dotyčného ptát, jaká zná vývojová prostředí, pokud by nevěděl, co pojem blokové programování znamená. Jestliže respondent odpoví ano, tak postupuje do druhé sekce. Když odpoví ne, pokračuje do čtvrté sekce.

**2)** Druhá sekce (viz Příloha III) slouží ke zjištění, zda je blokové programování součástí výuky informatiky na škole dotázaného. Bylo třeba zjistit, jestli respondent využívá blokové programování ve výuce nebo ne, aby se vyhnul zbytečným otázkám. V druhé sekci se nachází šest otázek a s tím i šest podmínek. U první a čtvrté vybrané odpovědi se přejde na sekci třetí. Po vybrání druhé nebo páté odpovědi se přejde na sekci pátou. A u třetí a sedmé odpovědi se přejde na sekci sedmou.

**3)** Třetí sekce se zabývá blokovým programováním ve výuce a je znázorněna v příloze IV. Respondent je dotázán na to, jak používá blokové programování, jestli za odměnu, za trest, nebo je to náplní předmětu. Také se zde zjišťuje jeho subjektivní názor na blokové programování pomocí otázek s číselnými škálami. A pomocí volných odpovědí, které nejsou povinné, aby neodrazovaly od vyplňování dotazníku. Tyto otázky byly položeny proto, že je důležitý postoj učitele k blokovému programování jako takovému. Když jeho postoj bude kladný, bude pravděpodobně blokové programování využívat ve výuce sám od sebe, bude ho používat rád a také kvalitněji než někdo, kdo k němu nemá pozitivní postoj. Po této sekci se automaticky přechází na sekci pátou.

**4)** Čtvrtá sekce stručné popisuje blokové programování a jeho přínos (viz Příloha V). Tato sekce byla vytvořena pro respondenty, kteří odpoví, že neví, co je to blokové programování. Také zohledňuje situaci, kdy někteří blokové programování znají, ale neznají ho pod tímto slovním spojením a jsou upozorněni, že pokud si uvědomili, že blokové programování znají, tak ať se vrátí o sekci zpět. Dotázaným, kteří se seznámili s blokovým programováním až takto, jsou položeny otázky, zda se jim tato forma programování líbí a zda by ji začlenili do výuky. Také je jim ponechán prostor k vyjádření ve volné odpovědi. Jestliže se dotázaný nevrátí o lekci zpět, tak automaticky pokračuje na sedmou sekci.

**5)** V páté sekci je nejprve vysvětlen rozdíl mezi unplugged a softwarovým prostředím, aby dotázaní mohli odpovídat na následující otázky (viz Příloha VI). Nejprve je zde otázka ohledně povědomí o aplikacích, kde respondent odpovídá pomocí výběrů, u kterých se nachází loga, názvy a hypertextové odkazy na dané aplikace. Loga by měla nejen zpestřit dotazník, ale také pomoci respondentovi ke vzpomenutí si na dané prostředí. Odkazy jsou využity k tomu, aby se dotázaný v případě zájmu mohl jednoduše podívat na příslušné prostředí. Učitel může vybrat možnost „jiné“ a dopsat i další aplikaci či pomůcku, kterou zná. Další otázky směřují ke zjištění, jak často respondent používá softwarové a jak často unplugged prostředí, které aplikace používá a jak je spokojený s používajícími aplikacemi. Je též dotázán na názor ohledně grafického zpracování, návaznosti ve výuce, motivaci a intuitivnost při práci. Na konci jsou dvě volné nepovinné otázky zaměřené na to, proč používají

dané aplikace a co jim na daných aplikacích chybí, co by změnili. Z této sekce se přechází přímo na následující šestou sekci.

**6)** Šestá sekce (viz Příloha VII) se zaměřuje na oblasti programování a je zde dotaz na jeho zkušenost s prostředím, jestli tyto oblasti dokáže objasnit. Sekci uzavírá otázka „Jaký výstup podle Vašich zkušeností více motivuje žáky pro napsání algoritmu v blokovém programování?“ a lze zde vybrat více odpovědí. Po této sekci následuje osmá sekce.

**7)** Příloha VIII obsahuje sedmou sekci, což je jiná varianta páté sekce určena pro respondenty, kteří nepoužívají blokové programování vůbec. Touto sekcí se zjišťuje, jestli ti, kteří blokové programování nepoužívají, mají o některých aplikacích alespoň povědomí. Je zde tedy obdoba první otázky z páté sekce. Dále je zde popsán rozdíl mezi softwarovým a unplugged prostředím a položená otázka:“ Jaké prostředí na Vás působí jako nejvíce vhodné pro využití ve výuce?“. U odpovědí se nacházejí obrázky reprezentující daná prostředí. Na konci je možnost přejít na čtvrtou sekci, pokud si respondent chce přečíst rychlé seznámení s blokovým programováním. Jestliže dotázaný nechce přejít na čtvrtou sekci, tak automaticky pokračuje na následující osmou sekci.

**8)** Předposlední sekce (viz Příloha IX) se týká informací o respondentovi a škole. Jsou zde otázky na věk učitele, na počet žáků na hodině a na to, kde se škola nachází. Po této sekci se pokračuje do poslední deváté sekce.

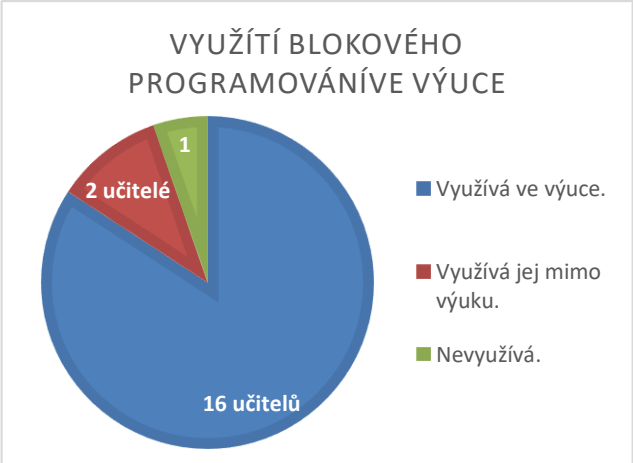
**9)** Poslední devátá sekce obsahuje poděkování „Děkuji Vám za Váš čas a přeji příjemné dny a motivované žáky.“ (viz Příloha X).

## 4.2 VÝSLEDKY DOTAZNÍKOVÉHO ŠETŘENÍ

Dotazník přináší zajímavé výsledky, které jsou znázorněny pomocí grafů, nacházející se v příloze XI. Na konci ledna bylo osloveno pomocí e-mailové služby 200 základních škol. Z těchto oslovených odpovědělo třicet dva respondentů. Osmnáct respondentů patřilo do věkové kategorie 35–49 let. Devět učitelů odpovědělo ve věku 50–65 let. A pět odpovědí přišlo od učitelů mladších 35 let. Většina odpovědí pocházela z městských škol, a to ve 21 případech.

Již na začátku dotazníku bylo zřejmé, že třináct oslovených nevědělo, co je blokové programování. Více než polovina respondentů (19) věděla, co je blokové programování a třináct z nich používá blokové programování ve výuce, protože je součástí výuky na jejich škole. Dále tři učitelé používají blokové programování ve výuce i přesto, že není náplní předmětu informatiky. Pouze dva učitelé používají blokové programování jenom pro vlastní potřebu. Je pozitivní, že jen jeden učitel odpověděl, že blokové programování vůbec nepoužívá a ani není součástí výuky na škole, kde vyučuje.


Graf 1 Povědomí o pojmu blokové programování

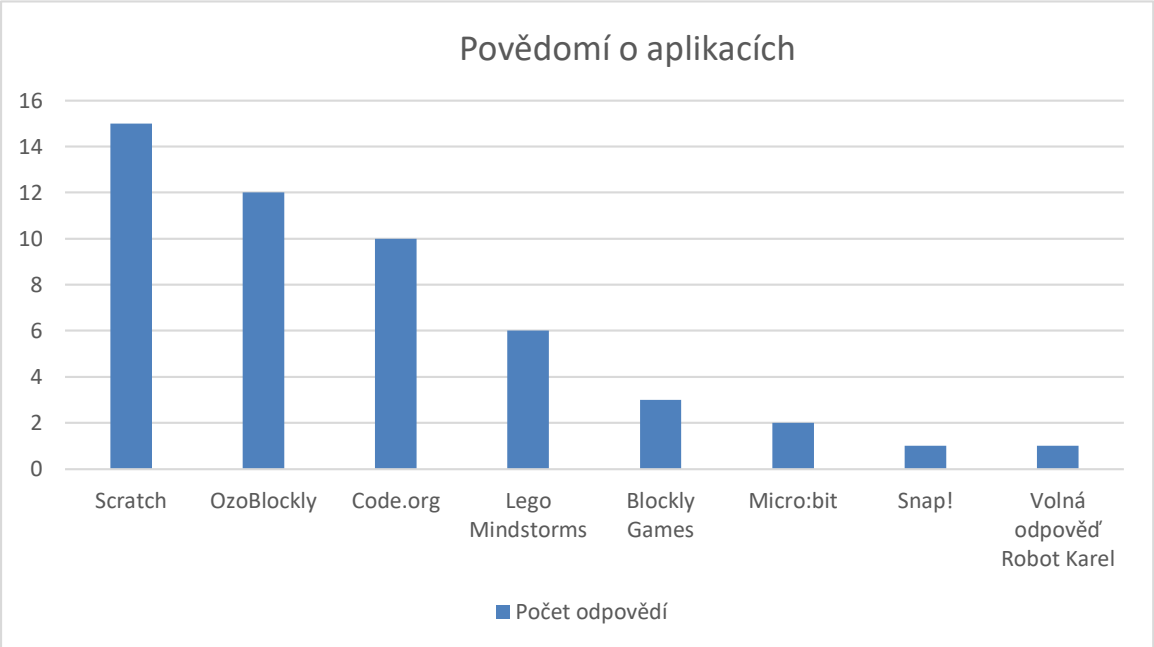



Graf 2 Využití blokového programování ve výuce

Z další části je patrné, že blokové programování je spíše zábavná forma výuky, protože nikdo nepoužívá tento způsob programování za trest. Devět dotázaných odpovědělo, že jej využívají pro zpestření výuky. V sedmi případech bylo zodpovězeno, že toto programování je součástí výuky. Jedna odpověď se týkala používání tohoto programování za odměnu a při volnočasových aktivitách. Většina učitelů (8 dotázaných) naprosto souhlasí, že je pro žáky tento způsob programování intuitivní, sedm učitelů si vybralo na stupnici od jedné do čtyř číslo tři. Zbytek si na této stupnici vybralo číslo dva. Přičemž číslo jedna znamená vůbec a číslo čtyři znamená velmi. Při otázce mířené na motivaci k sestavování algoritmů si čtyři

dotázaní myslí, že sestavování algoritmů pomocí bloků žáky motivuje více než psaní zdrojového kódu. Zbytek odpovědělo stupněm 3. Stejný počet dotázaných učitelů (4 dotázaní) souhlasí s tím, že by se toto programování mělo zahrnout do výuky. Zbytek znovu odpověděl stupněm 3. U volné odpovědi týkající se vnímání pozitiv blokového programování se projevilo devět respondentů, kde se často opakují odpovědi o rozvoji informatického, logického a programátorského myšlení. Líbí se jim také, že není nutno znát přesnou syntaxi a dá se pomocí tohoto programování oživit výuku díky známým a kreativním prostředím. Jako negativa vnímají, že popis bloků je v každém prostředí jiný, což žáky mate. Také postrádají doplnění vlastních příkazů. To, že není třeba znát přesnou syntaxi, vnímají zároveň i jako nevýhodu, protože si žák neosvojí přesnou práci s programovacím jazykem. Učitelé také vnímají jako negativní to, že žáci takovou formu aktivity berou jako hru a zkoušejí různé kombinace bez toho, aniž by se nad programem zamysleli. Také se zde objevily poznámky ve smyslu, že pro využití blokového programování je náročné na přípravu na výuku a že slabší jedinci si s úkoly neporadí.

Otázky zaměřené přímo na aplikace daného prostředí ukázaly, že nejznámějším prostředím je Scratch, protože tuto aplikaci si zde vybralo patnáct z osmnácti dotázaných. Na druhé místo s dvanácti odpověďmi se zařadila aplikace OzoBlockly, na třetí místo se zařadilo prostředí Code.org s deseti odpověďmi. Další obsazená místa je možné vyčíst z grafu 3.




Graf 3 Povědomí o aplikacích

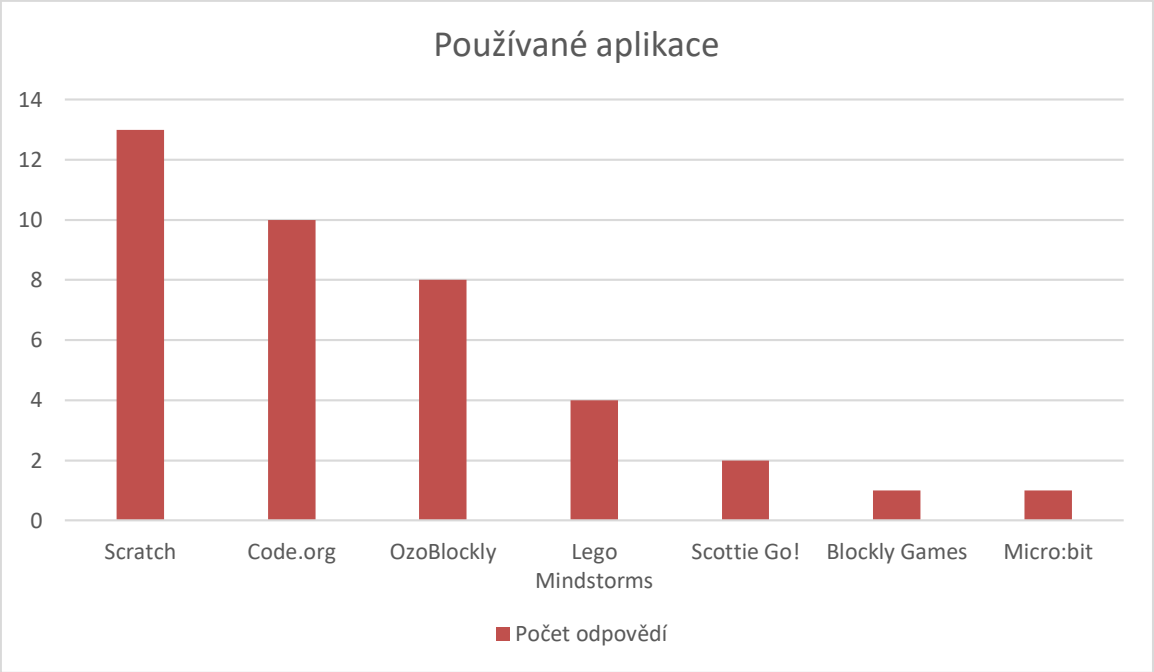
Softwarové prostředí používá osm dotázaných měsíčně. Zbytek dotázaných jednotlivě odpovědělo, že softwarové prostředí používají takto:

- „Jako tematický celek v jednom ročníku 2. stupně“
- „5x – 8x ročně“
- „Příležitostně (pouze jako ukázka)“
- „Zatím se seznamuji“
- „Několik hodin během roku v jednom vyučovacím bloku“
- „8 vyučovacích hodin za rok“
- „Někdy týdně někdy měsíčně (záleží na probíraném tématu, párkrát během pololetí, denně a týdně)“

Dvanáct učitelů odpovědělo, že unplugged prostředí nepoužívá vůbec. Měsíčně používají unplugged prostředí dva dotázaní. Ostatní jednotlivě odpověděli takto:

- „1x - 3x ročně  podle potřeby (většinou týdně).“
- „2 vyučovací hodiny před vlastním programováním.“
- „Záleží na probíraném tématu. U mladších častěji (ale většinou týdně), pak již jen podle potřeby.“

Nejpoužívanějším prostředím podle dotazníku je Scratch s třinácti odpověďmi, na druhém místě se nachází, s deseti odpověďmi Code.org. Na třetím je OzoBlockly, které si vybralo osm učitelů. Veškerá vybraná prostředí je možné vidět v grafu 4.




Graf 4 Používané aplikace

S těmito prostředími jsou většinou respondenti spokojeni. Většina učitelů je spokojena s grafickým zpracováním, návazností, motivací a intuitivností prostředí, jež používají. Nejčastěji tato prostředí používají díky jejich dostupnosti. Dále také proto, že dané prostředí bylo první, se kterým se setkali. Někteří dokonce prostředí používají díky funkčnosti. Na jimi používaných aplikacích by změnili programovací jazyk, větší zásobu předpřipravených úkolů pro učitele a podporu operačních systémů.

U motivace výstupů je nejčastější názor, že žáky motivuje nejvíce fyzický výstup jako je pohyb robota, vibrace, zvuky atd. Osm učitelů si myslí, že žáky motivuje virtuální výstup (na obrazovce). Dva dotázání mají zkušenost, že záleží na věku a na typu žáka. Zbylí dva učitelé nemají zkušenost s fyzickým výstupem. Dvanáct učitelů si myslí, že je vhodná kombinace softwarového a unplugged prostředí ve výuce. Dva učitelé by volili prostředí softwarové.

# ZÁVĚR

Bakalářská práce nejprve je čtenáře seznamuje s pojmem algoritmus a s jeho znázorněním. V kapitole Algoritmizace je vysvětleno, že se lidé s algoritmy setkávají každý den a není třeba ho spojovat pouze s počítači a programováním. Následující kapitola pojednává již o programování samotném. Popisuje programovací jazyky důležité pro komunikaci s počítačem a jejich podobnost běžným jazykům, kvůli jejich syntaxi a sémantice. Dále zmiňuje, proč je třeba se učit programovat na základní škole. Je zde uvedeno, že je potřeba se držet s dobou a rozvojem technologií, a proto je důležité se soustředit na aktuální požadavky. Tuto myšlenku má i rámcový vzdělávací program nabízející revizi, ve které je popsáno, že by se prvky informatiky měly vhodně objevovat ve všech předmětech výuky.

V kapitole Blokové programování je dané programování popsáno spolu s rozdílem mezi softwarovým a unplugged prostředím a jeho charakteristickými vlastnostmi. Kapitola byla rozšířena o zástupce daných prostředí, kteří jsou zmíněni i v dotazníkovém šetření popsaném na konci této práce. Byly zde zmíněny softwarové prostředí Scratch, Code, Micro:bit, Blockly Games, OzoBlockly, Lego Mindstorms, MIT App Inventor a Snap! Z unplugged prostředí pak Scottie Go! a Osmo Coding. Také bylo zmíněno, že na trhu existuje spousta deskových her pro rozvoj algoritmického myšlení a určování příkazů jako Twin Tin Bots, Wings of War a Robo Rally.

Dotazník byl navržen tak, aby mohl zmapovat aktuální stav využití daných prostředí ve výuce na předem vybraných školách. Překvapivě bylo blokové programování celkem populární, protože z 32 odpovědí dané programování zná 19 učitelů a 13 z nich ho využívá ve výuce. Nejčastější odpovědí bylo, že blokové programování používají jako její zpestření, z čehož lze soudit, že je tato forma programování pro žáky zábavná.

Výsledky této práce mohou posloužit pedagogům pro seznámení s výukovými pomůckami, které ještě neznal, a mohly by ho namotivovat ke zpestření výuky. A tím žáky přimět vhodně používat moderní technologie tak, aby ho rozvíjely.

## RESUMÉ

Tato bakalářská práce popisuje porovnání softwarového a unplugged prostředí pro výuku blokového programování. Skládá se ze čtyř kapitol Algoritmizace, Programování, Blokové programování a Dotazníkové šetření. Nejprve obsahuje rychlé seznámení s pojmy algoritmus a programování. Tyto pojmy jsou vysvětleny, aby bylo lépe pochopitelné, k čemu slouží popsaná a porovnávaná prostředí blokového programování. Před popisem daných prostředí je vysvětlen pojem blokové programování spolu s rozdílem mezi unplugged a softwarovým prostředím. Závěrem práce se čtenář seznámí s přípravou dotazníku využití blokového programování na školách a jeho výsledkem.

Klíčová slova: blokové programování, programování, unplugged, algoritmus, algoritmizace, algoritmické myšlení, myšlení, prostředí, aplikace

## RESUME

This bachelor thesis deals with comparing software and unplugged environments for teaching block-based programming. Thesis consist of four chapters Algorithmization, Programming, Block Programming and Questionnaire Survey. At first, it provides a quick introduction to the concepts of algorithms and programming. For better understanding of these concepts, the block-based programming enviroments are described in detail. Then follows a thorough description of the differenece between the software and unplugged enviroments. Finally, there is an introduction to the creation of the questionnaire, its logic and outputs.

Keywords: block-based programming, programming, unplugged, algorithm, algorithmization, algorithmic thinking, thinking, environment, application

## SEZNAM LITERATURY


1. BROOKSHEAR, J., SMITH D., BRYLOW D. *Informatika*. Brno: Computer Press, 2013. ISBN 978-80-2513-805-2.
2. PŠENČÍKOVÁ, J. Algoritmizace. Kralice na Hané: Computer Media, 2007. ISBN:80-86686-80-9.
3. LESSNER, D., LÁNA, M., TOMKOVÁ, M., HAUT, J. Základy informatiky pro střední školy, 2018 JUČ [online]. Dostupné z: https://popelka.ms.mff.cuni.cz/~lessner/mw/index.php/U%C4%8Debnice/Algoritmus/Co_je_to_algoritmus
4. Computer History Museum © 2020. 1801: PUNCHED CARDS CONTROL JACQUARD LOOM, 2015 [online]. Dostupné z: https://www.computerhistory.org/storageengine/punched-cards-control-jacquard-loom/
5. Computer History Museum © 2020. ADA LOVELACE, [online]. Dostupné z: https://www.computerhistory.org/babbage/adalovelace/
6. MÜLLER, K. Programovací jazyky. Praha: Vydavatelství ČVUT, 2002. ISBN 80-01-02458-x.
7. COOKE, A. 2012. Introduction To Computer Languages, [online] Dostupné z: http://www.acooke.org/comp-lang.html
8. VYSTAVĚL, R. Informatika a informační a komunikační technologie 2008, dostupné z: https://clanky.rvp.cz/clanek/c/G/2646/moderni-vyuka-programovani.html/
9. KOZÁK, P. Nástroje pro výuku programování 2015, dostupné z: https://spomocnik.rvp.cz/clanek/20019/NASTROJE-PRO-VYUKU-PROGRAMOVANI.html
10. RVP [online]. Dostupné z: http://www.nuv.cz/t/rvp
11. Revize RVP [online]. Dostupné z: http://www.nuv.cz/t/rrvp
12. BACKUS, J. W., BAUER, F. L., GREEN, J., KATZ, C., MCCARTHY, J., PERLIS, A. J., RUTISHAUSER, H., SAMELSON, K., VAUQUOIS, B., WEGSTEIN, J. H., WIJNGAARDEN, A. van, WOODGER, M. Revised Report on the Algorithmic Language Algol 60. [online]. Dostupné z: https://www.masswerk.at/algol60/report.htm
13. MOHAMAD,S. N. H., PATEL, A., LATIH, T., QASSIM, Q.S.,LIU, N., TEW, Y. Block-based programming approach: challenges and benefits. In: Proceedings of the 2011 International Conference on Electrical Engeneering and Informatics [online]. IEEE, 2011, s. 1-5[cit. 2019-06-13]. DOI: 10.1109/ICEEI.2011.6021507. ISBN 978-1-4577-0753-7. Dostupné z: http://ieeexplore.ieee.org/document/6021507/
14. PRŮCHA, J., WALTEROVÁ E., MAREŠ J. Pedagogický slovník. 7., aktualiz. a rozš. vyd. Praha: Portál, 2013. ISBN 978-80-262-0403-9.

15. GROVER, S., JACKIW, N. LUNDH, P. Conceprs before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. Computer Science Education [online]. 2019, 29(2-3), 106-135 [cit. 2019-06-13]. DOI: 10.1080/08993408.2019.1568955. ISSN 0899-3408. Dostupné z: https://www.tandfonline.com/doi/full/10.1080/08993408.2019.1568955
16. BAU, D., GRAY, J., KELLEHER, C., SHELDON, J., TURBAK, F. Learnable Programming: Blocks and Beyond. Communications of the ACM, Vol. 60 No. 6 Pages 72-80. DOI:10.1145/3015455. [cit. 16.06.2019]. Dostupné z: https://cacm.acm.org/magazines/2017/6/217743-learnable-programming/fulltext
17. HERMANS, F., AIVALOGLOU, E. 2017. To Scratch or not to Scratch?. In Proceedings of WiPSCE '17, Nijmegen, Netherlands, November 8–10, 2017, 8 pages. DOI: 10.1145/3137065.3137072 [online]. Dostupné z: https://dl.acm.org/doi/abs/10.1145/3137065.3137072
18. Scratch [online]. Dostupné z: https://scratch.mit.edu/
19. Blockly [online]. Dostupné z: https://developers.google.com/blockly
20. Code [online]. Dostupné z: https://code.org/
21. OzoBlockly [online]. Dostupné z: https://ozobot.com/ozoblockly
22. Micro:bit [online]. Dostupné z: https://microbit.org/
23. MIT App Inventor [online]. Dostupné z: https://appinventor.mit.edu/
24. Snap! [online]. Dostupné z: https://snap.berkeley.edu/
25. Twin Tin Bots [online]. Dostupné z: https://www.flatlinedgames.com/Games/twin-tin-bots
26. Wings of War [online]. Dostupné z: https://www.aresgames.eu/games/ww1-wings-of-glory-line
27. Robo Rally [online]. Dostupné z: https://avalonhill.wizards.com/games/robo-rally
28. Scottie Go! [online]. Dostupné z: https://scottiego.com/en/
29. VIDENOVIK, M., TRAJKOVIK, V., USING SCOTTIE GO! AS GAME BASED LEARNING TOOL FOR COMPUTATIONAL THINKING COURSE, DOI: 10.21125/iceri.2018.1304, [online]. Dostupné z: https://www.researchgate.net/publication/329197065_USING_SCOTTIE_GO_AS_GAME_BASED_LEARNING_TOOL_FOR_COMPUTATIONAL_THINKING_COURSE
30. Osmo [online]. Dostupné z: https://www.playosmo.com/en/coding/
31. BlocklyGames [online]. Dostupné z: https://blockly.games/

## SEZNAM OBRÁZKŮ

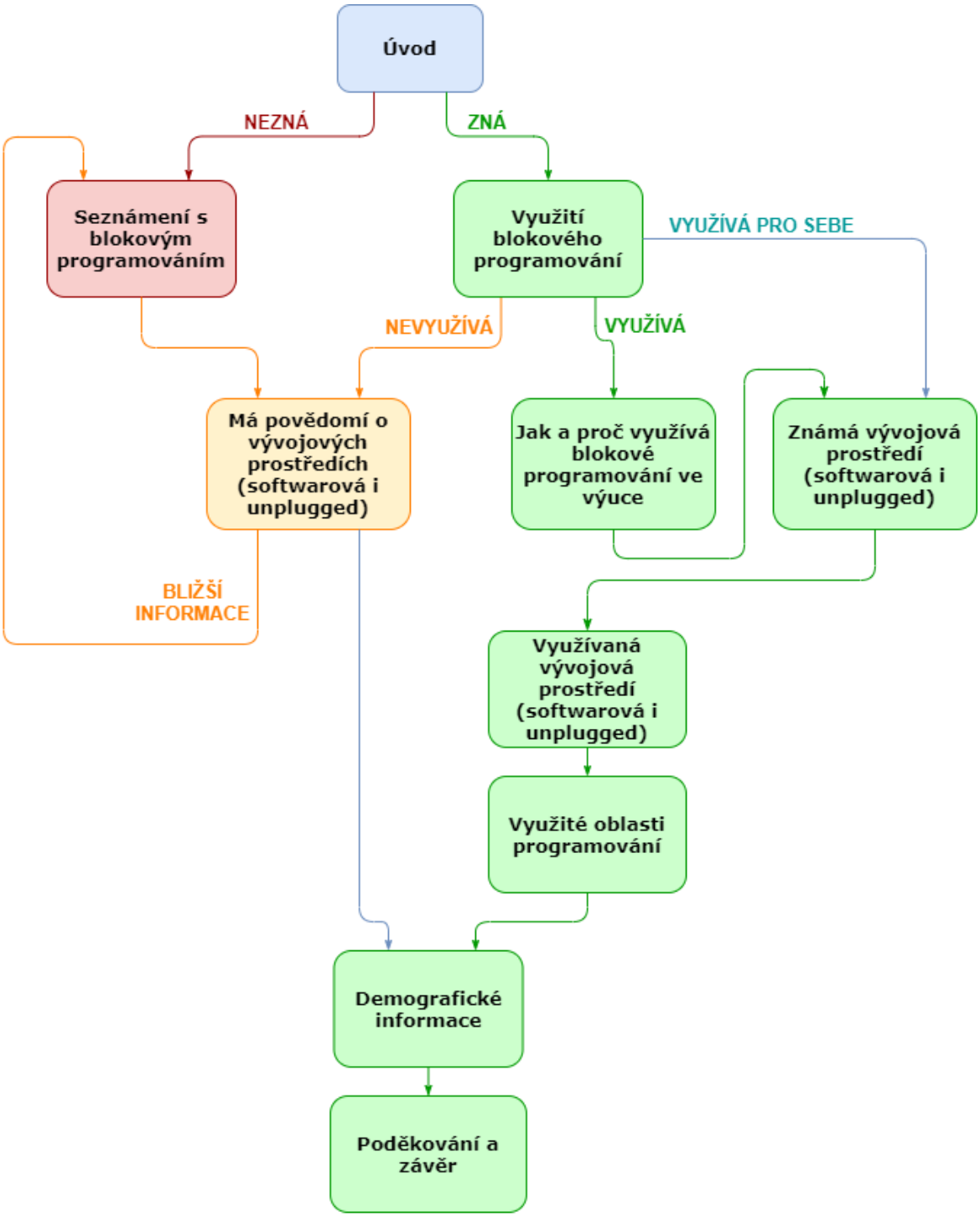
# SEZNAM TABULEK

## SEZNAM GRAFŮ

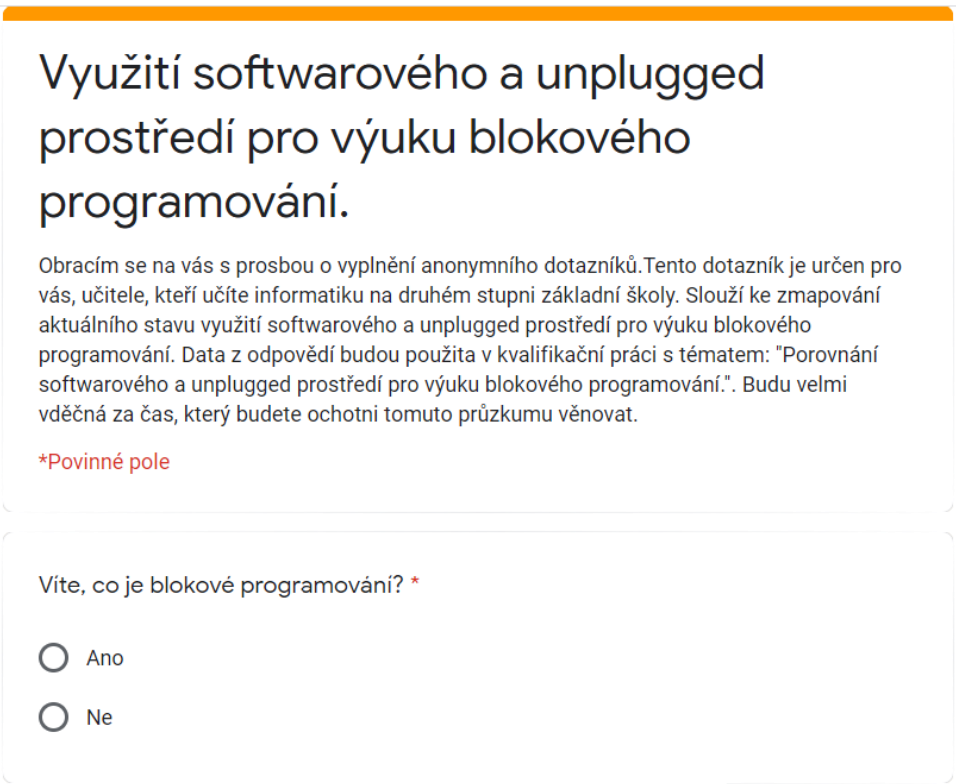
## SEZNAM PŘÍLOH

# PŘÍLOHY

**Příloha I - Logika dotazníku**



**Příloha II - Sekce 1**



Využití softwarového a unplugged prostředí pro výuku blokového programování.

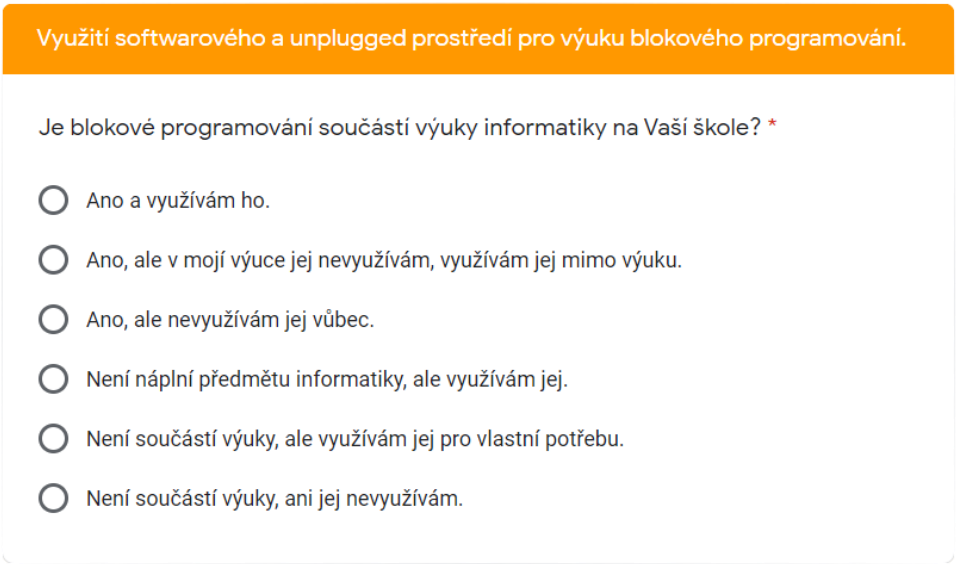
Obracím se na vás s prosbou o vyplnění anonymního dotazníků.Tento dotazník je určen pro vás, učitele, kteří učíte informatiku na druhém stupni základní školy. Slouží ke zmapování aktuálního stavu využití softwarového a unplugged prostředí pro výuku blokového programování. Data z odpovědí budou použita v kvalifikační práci s tématem: "Porovnání softwarového a unplugged prostředí pro výuku blokového programování.". Budu velmi vděčná za čas, který budete ochotni tomuto průzkumu věnovat.

*Povinné pole

Víte, co je blokové programování? *

- Ano
- Ne

**Příloha III - Sekce 2**



Využití softwarového a unplugged prostředí pro výuku blokového programování.

Je blokové programování součástí výuky informatiky na Vaší škole? *

- Ano a využívám ho.
- Ano, ale v mojí výuce jej nevyužívám, využívám jej mimo výuku.
- Ano, ale nevyužívám jej vůbec.
- Není náplní předmětu informatiky, ale využívám jej.
- Není součástí výuky, ale využívám jej pro vlastní potřebu.
- Není součástí výuky, ani jej nevyužívám.

**Příloha IV- Sekce 3**

Blokové programování ve výuce

Jak využíváte blokové programování ve výuce? *

- [ ] Jako plnohodnotnou součást výuky.
- [ ] Pro zpestření výuky.
- [ ] Za odměnu.
- [ ] Volnočasové aktivity.
- [ ] Za trest.
- [ ] Jiné: _____

Považujete způsob blokového programování pro žáky za intuitivní? *

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| Vůbec | ○ | ○ | ○ | ○ | Velmi |

Myslíte, že blokový způsob programování žáky motivuje k sestavení algoritmů více než psaní zdrojového kódu? *

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| Vůbec | ○ | ○ | ○ | ○ | Velmi |

Považujete za vhodné blokové programování zahrnout do výuky? *

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| Vůbec | ○ | ○ | ○ | ○ | Velmi |

Jaká vnímáte pozitiva využívání blokového programování?

Vaše odpověď

Jaká vnímáte negativa využívání blokového programování?

Vaše odpověď

**Příloha V - Sekce 4**

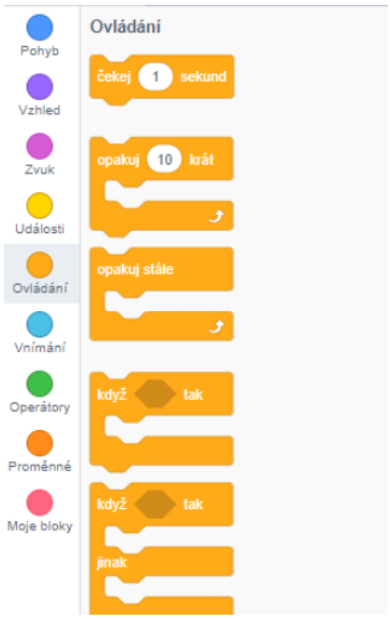
## Rychlé seznámení s blokovým programováním

Zdrojový kód je zde definován bloky, které mají různé vlastnosti. Jsou rozděleny na logické, matematické, textové, na cykly, podmínky a jiné bloky. Ty do sebe zapadají a strukturované programování umožňuje jejich vzájemné vnořování, což umožňuje používat lokální proměnné a funkce.

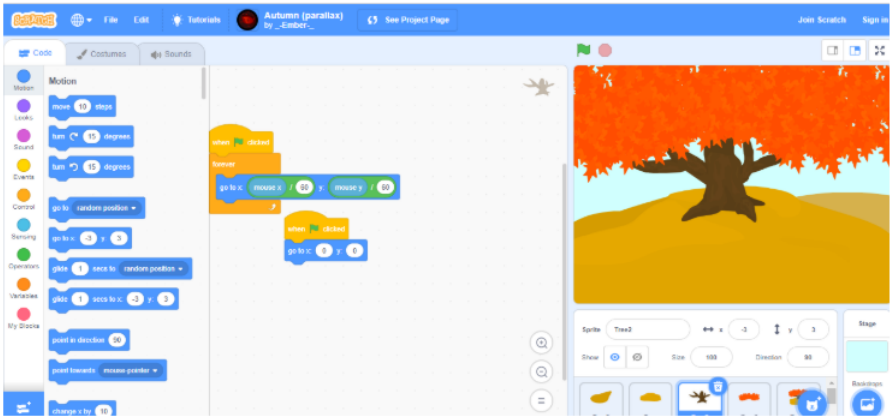
### Blok

Blok specifikuje část kódu programu a chová se, jako by byl jedním příkazem. Bloky jsou zjednodušením syntaktických pravidel zápisu zdrojového kódu programu. Jinými slovy, obrázkový blok nahrazuje jeden i více příkazů.

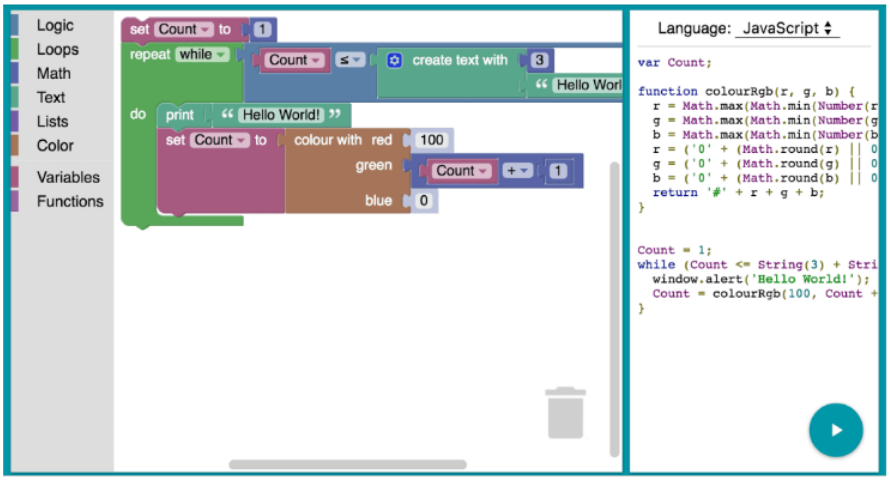
Bloky v prostředí Scratch




Prostředí Scratch



Zde je vidět, jak použití programových bloků opticky zjednodušilo kód zapsaný v pravé části.




Zde můžete zhlédnout rychlé seznámení s blokovým programováním v prostředí od Google.


### Přínos

Vzbuzuje zájem, představivost a podporuje logické myšlení. Toto programování je intuitivní, vizuální a názorné. A tím pro děti snadněji pochopitelné než obyčejný kód.

### Přínos

Vzbuzuje zájem, představivost a podporuje logické myšlení. Toto programování je intuitivní, vizuální a názorné. A tím pro děti snadněji pochopitelné než obyčejný kód.

**Pakliže tímto zjišťujete, že blokové programování vlastně znáte, tak prosím klikněte na tlačítko zpět a zvolte patřičnou odpověď.**

Líbí se Vám podstata blokového programování? *

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| Vůbec | ○ | ○ | ○ | ○ | Velmi |

Považujete za vhodné zahrnout blokové programování do výuky? *

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| Vůbec | ○ | ○ | ○ | ○ | Velmi |

Chcete-li doplnit informace k předešlým otázkám, nebo máte nějaký postřeh k blokovému programování obecně, využijte prosím prostor zde:

Vaše odpověď

**Příloha VI - Sekce 5**

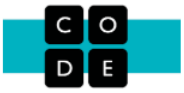
## Vývojová prostředí

Unplugged programování narozdíl od programování v softwarovém prostředí neprobíhá na žádném zařízení jako PC, smartphone, tablet... Často je řešeno formou kartiček nebo destiček, které se skládají jako puzzle. Nicméně může být zařízení využíváno k vyhodnocení připraveného programu.

Znáte některá z uvedených aplikací? Pokud ano vyberte je. Pokud znáte jiné, vyplňte v možnosti "Jiná..." *


☐ Scratch https://scratch.mit.edu/


☐ Blockly https://developers.google.com/blockly




☐ Code.org https://code.org/


☐ micro:bit https://microbit.org/


☐ OzoBlockly https://ozoblockly.com/


☐ Lego Mindstorms - EV3 Programmer https://www.lego.com/cs-cz/themes/mindstorms/learntoprogram


☐ MIT App Inventor https://appinventor.mit.edu/


☐ Snap! https://snap.berkeley.edu/

Které z uvedených aplikací využíváte? Pokud využíváte jiné, vyplňte v možnosti "Jiná..." *

☐ Scratch

☐ Blockly

☐ Code.org

☐ micro:bit

☐ OzoBlockly

☐ Lego Mindstorms - EV3 Programmer

☐ MIT App Inventor

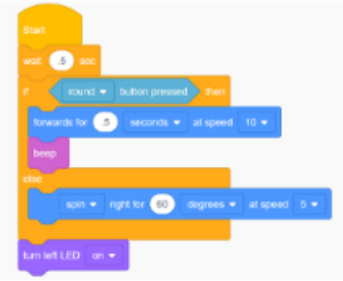
☐ Snap!

☐ Osmo (Unplugged)

☐ Scottie Go! (Unplugged)

☐ Jiné:

Jak často používáte softwarové prostředí? *




- ◯ Vůbec
- ◯ Měsíčně
- ◯ Týdně
- ◯ Denně
- ◯ Jiné: _____

Jak často používáte unplugged prostředí? *

- ◯ Vůbec
- ◯ Měsíčně
- ◯ Týdně
- ◯ Denně
- ◯ Jiné: _____

☐ Neznám žádné

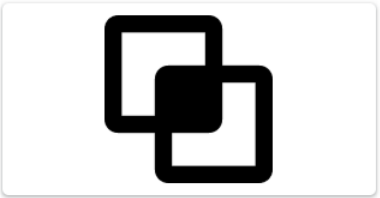
☐ Jiné:

Unplugged programování narozdíl od programování v softwarovém prostředí neprobíha na žádném zařízení jako PC, smartphone, tablet... Ale může být formou kartiček, které se skládají jako puzzle.

Jaké prostředí na Vás působí jako nejvíce vhodné pro využití ve výuce? *

○ Softwarové

○ Unplugged



○ Kombinace obou prostředí

Chcete si přečíst rychlé seznámení s blokovým programováním? *

○ Ano

○ Ne

Které z uvedených aplikací využíváte? Pokud využíváte jiné, vyplňte v možnosti "Jiná..." *

☐ Scratch

☐ Blockly

☐ Code.org

☐ micro:bit

☐ OzoBlockly

☐ Lego Mindstorms - EV3 Programmer

☐ MIT App Inventor

☐ Snap!

☐ Osmo (Unplugged)

☐ Scottie Go! (Unplugged)

☐ Jiné:

Jak jste spokojen/a s prostředím, které používáte? *

| | Vůbec | Spíše ne | Spíše ano | Velmi | Nevím |
|---|---|---|---|---|---|
| Grafické zpracování | ○ | ○ | ○ | ○ | ○ |
| Návaznost | ○ | ○ | ○ | ○ | ○ |
| Motivace | ○ | ○ | ○ | ○ | ○ |
| Intuitivita | ○ | ○ | ○ | ○ | ○ |

Jaký máte důvod, že používáte právě tyto aplikace?

Vaše odpověď

Co Vám na Vámi využívaných aplikacích chybí, co byste změnil/a?

Vaše odpověď

**Příloha VII - Sekce 6**

Oblasti programování

Dokáže na základě Vašich zkušeností Vámi používané prostředí dostatečně objasnit oblasti programování? *

| | Vůbec | Spíše ne | Spíše ano | Velmi | Nevím |
|---|---|---|---|---|---|
| Proměnné | ○ | ○ | ○ | ○ | ○ |
| Operátory | ○ | ○ | ○ | ○ | ○ |
| Podmínky | ○ | ○ | ○ | ○ | ○ |
| Cykly | ○ | ○ | ○ | ○ | ○ |
| Pole | ○ | ○ | ○ | ○ | ○ |

Jaký výstup podle Vašich zkušeností více motivuje žáky pro napsání algoritmu v blokovém programování? *

- [ ] Na obrazovce (virtuální)
- [ ] Fyzický (pohyb robota, vibrace, zvuky, svítící diody na plošném spoji, ...)
- [ ] Jiné:

**Příloha VIII - Sekce 7**

Vývojová prostředí

Viděl/a jste někdy některá z uvedených aplikací? Pokud ano vyberte je. Pokud znáte jiné, vyplňte v možnosti "Jiná..." *

☐ Scratch https://scratch.mit.edu/

☐ Blockly https://developers.google.com/blockly

☐ Code.org https://code.org/

☐ micro:bit https://microbit.org/

☐ OzoBlockly https://ozoblockly.com/

☐ Lego Mindstorms - EV3 Programmer Lego Mindstorms - EV3 Programmer

☐ MIT App Inventor https://appinventor.mit.edu/

☐ Snap! https://snap.berkeley.edu/

☐ Osmo (Unplugged) https://www.playosmo.com/en/coding/

☐ Scottie Go! (Unplugged) https://dejtemipevnybod.cz/2019/03/programujeme-scottieho-scottie-go/

☐ Neznám žádné

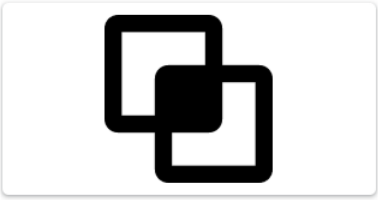
☐ Jiné:

Unplugged programování narozdíl od programování v softwarovém prostředí neprobíha na žádném zařízení jako PC, smartphone, tablet... Ale může být formou kartiček, které se skládají jako puzzle.

Jaké prostředí na Vás působí jako nejvíce vhodné pro využití ve výuce? *

○ Softwarové

○ Unplugged



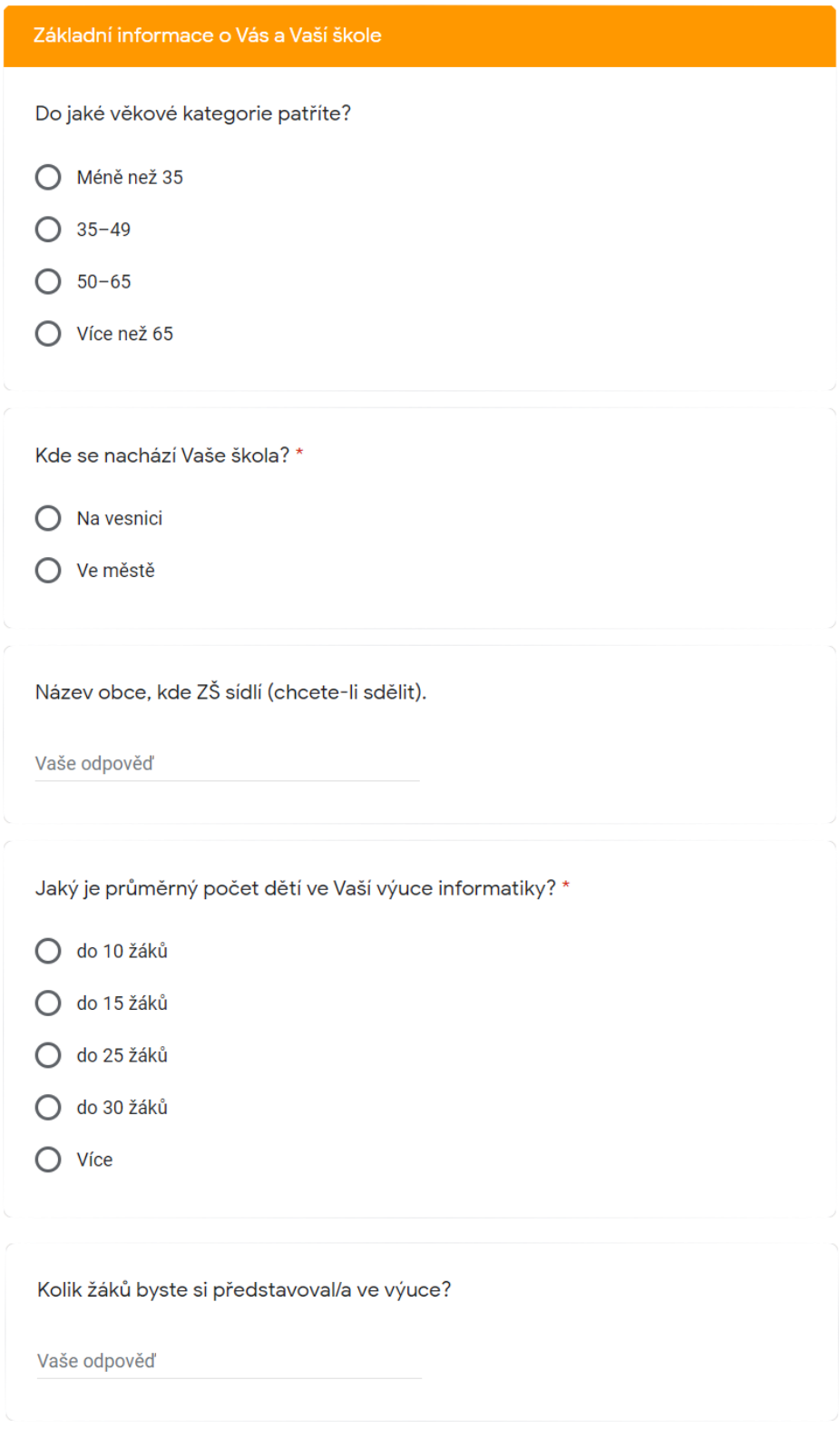
○ Kombinace obou prostředí

Chcete si přečíst rychlé seznámení s blokovým programováním? *

○ Ano

○ Ne

**Příloha IX - Sekce 8**



Základní informace o Vás a Vaší škole

Do jaké věkové kategorie patříte?

- Méně než 35
- 35–49
- 50–65
- Více než 65

Kde se nachází Vaše škola? *

- Na vesnici
- Ve městě

Název obce, kde ZŠ sídlí (chcete-li sdělit).

Vaše odpověď

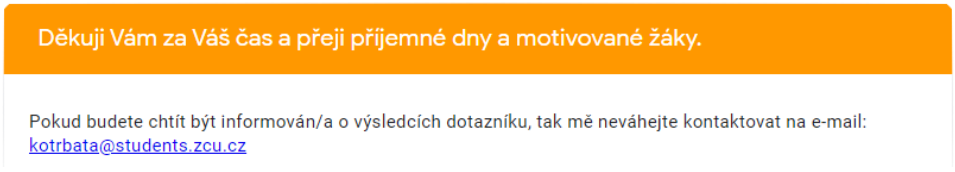
Jaký je průměrný počet dětí ve Vaší výuce informatiky? *

- do 10 žáků
- do 15 žáků
- do 25 žáků
- do 30 žáků
- Více

Kolik žáků byste si představoval/a ve výuce?

Vaše odpověď

**Příloha X - Sekce 9**



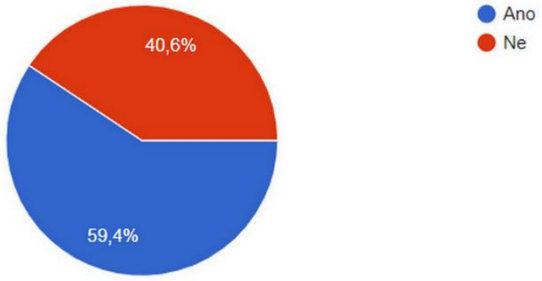
Děkuji Vám za Váš čas a přeji příjemné dny a motivované žáky.

Pokud budete chtít být informován/a o výsledcích dotazníku, tak mě neváhejte kontaktovat na e-mail: kotrbata@students.zcu.cz

**Příloha XI - Výsledky dotazníku**

Víte, co je blokové programování?

32 odpovědí




Využití softwarového a unplugged prostředí pro výuku blokového programování.

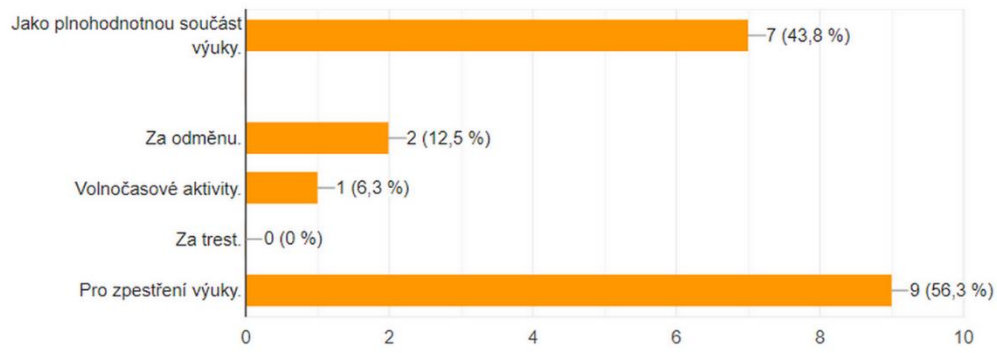
Je blokové programování součástí výuky informatiky na Vaší škole?

19 odpovědí


Blokové programování ve výuce

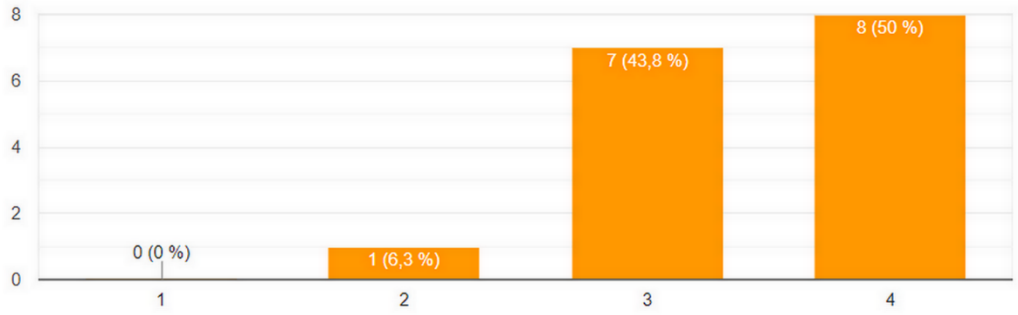
Jak využíváte blokové programování ve výuce?

16 odpovědí



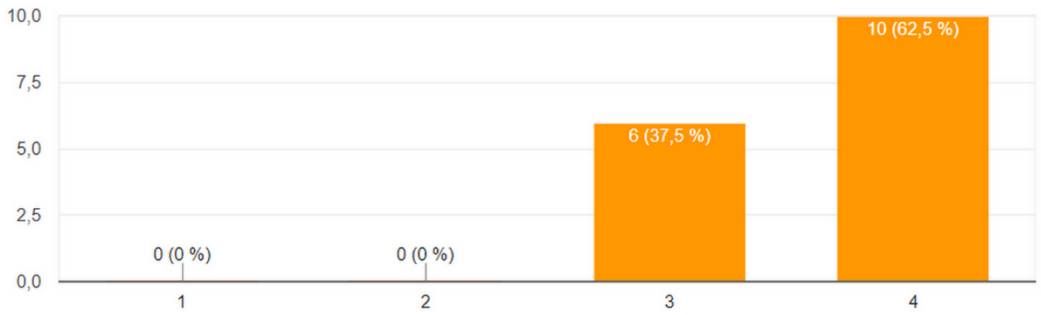
Považujete způsob blokového programování pro žáky za intuitivní?

16 odpovědí



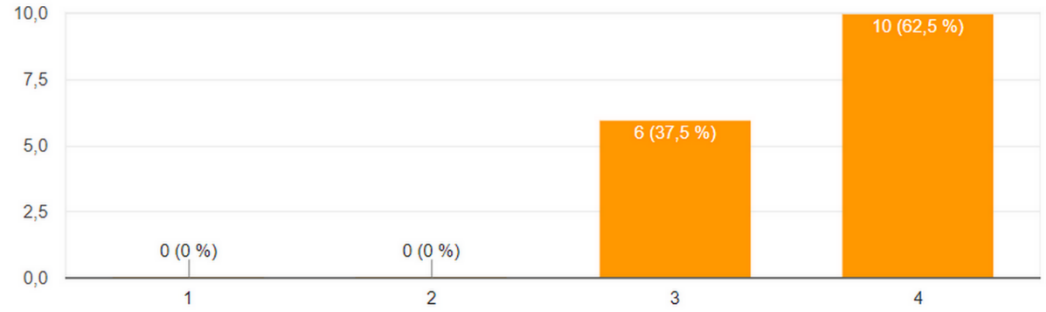

Myslíte, že blokový způsob programování žáky motivuje k sestavení algoritmů více než psaní zdrojového kódu?

16 odpovědí




Považujete za vhodné blokové programování zahrnout do výuky?

16 odpovědí



Jaká vnímáte pozitiva využívání blokového programování?

9 odpovědí

- rozvoj logického a informatického myšlení
- Většinou se jedná o jednoduché nástroje pro naučení se základů programování. Většinou se jedná o jednoduché grafické a kreativní prostředí, ve kterém se dobře orientuje i pracuje jak dětem tak dospělým.
- Oživení výuky.
- Není nutné znát přesnou syntaxi příkazů. Úspora času a práce.
- Jednoduché osvojení algoritmů pomocí her.
- Rozvoj informatického myšlení
- rozvíjí programátorské myšlení
- žáci snáze pochopí základní principy programování
- rozvoj logického myšlení

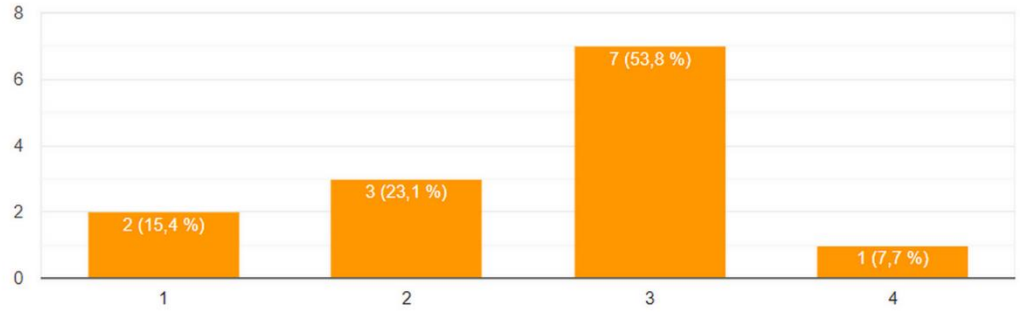
Jaká vnímáte negativa využívání blokového programování?

7 odpovědí

- Možnost používat je předem dané příkazy (není možnost doplnit vlastní příkazy). Ale na úrovni kterou ve škole používáme se nejedná o žádné omezení.
- Složitá příprava učitele.
- Není nutné znát přesnou syntaxi příkazů. Chybí návyk na přesnou práci.
- Někteří žáci to berou jen jako hru a metodu pokus / omyl a nepřemýšlí nad obsahem.
- slabší žáci si s ním neporadí
- nevnímám
- popis bloků v různých prostředích je jiný a žáky mate

Považujete za vhodné zahrnout blokové programování do výuky?

13 odpovědí




Chcete-li doplnit informace k předešlým otázkám, nebo máte nějaký postřeh k blokovému programování obecně, využijte prosím prostor zde:

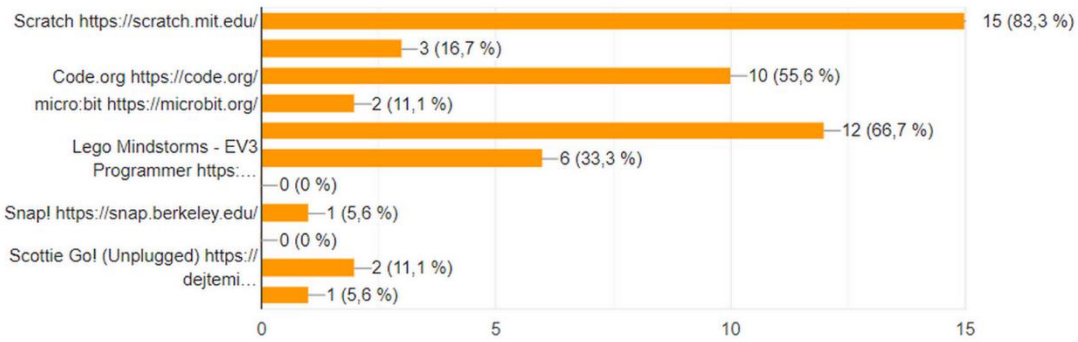
1 odpověď

Na webu www.umimeprogramovat.cz je na toto skvělá 'hra'. Děti velmi baví, pro některé je toto již obtížné (občas maji problém i s sipkovou či želvičkou).

## Vývojová prostředí

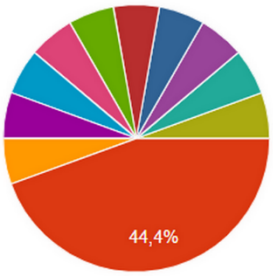
Znáte některá z uvedených aplikací? Pokud ano vyberte je. Pokud znáte jiné, vyplňte v možnosti "Jiná..."

18 odpovědí



Jak často používáte softwarové prostředí?

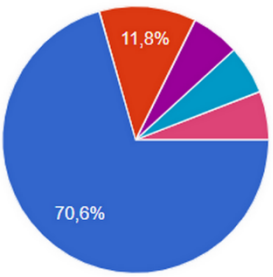
18 odpovědí




- Vůbec
- Měsíčně
- Týdně
- Denně
- jako tématický celek v jednom ročníku…
- 8 vyučovacích hodin za rok
- Párkrát do pololetí
- Záleží na probíraném tématu. Někdy je to týdně, někdy měsíčně.
- Zatím se seznamuji.
- 5x - 8x ročně
- příležitostně, ukázka žákům k seznámení
- několikrát za pololetí

Jak často používáte unplugged prostředí?

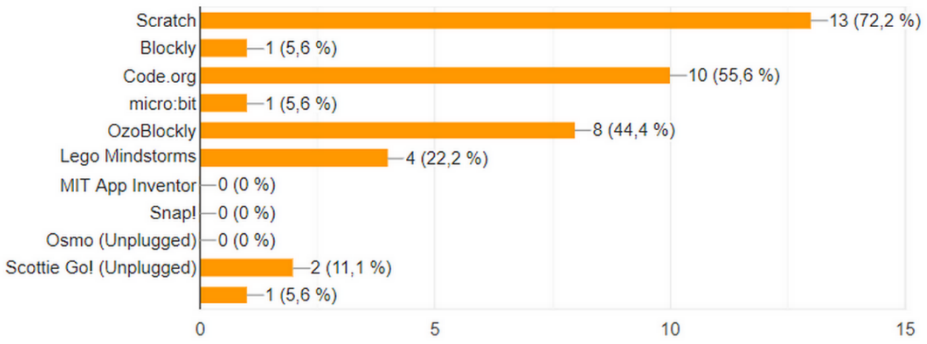
17 odpovědí



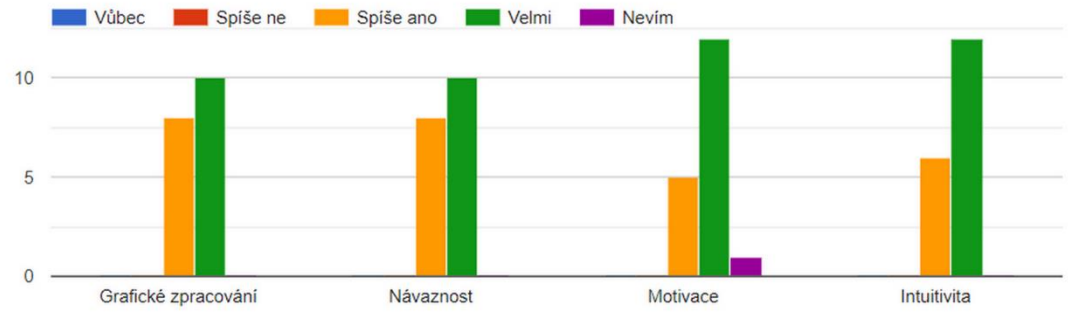

- Vůbec
- Měsíčně
- Týdně
- Denně
- 2 vyučovací hodiny před vlastním programováním
- Záleží na probíraném tématu. U mladších častěji (ale většinou týdně), pak již jen podle potřeby.
- 1x - 3x ročně

Které z uvedených aplikací využíváte? Pokud využíváte jiné, vyplňte v možnosti "Jiná..."

18 odpovědí



Jak jste spokojen/a s prostředím, které používáte?




Jaký máte důvod, že používáte právě tyto aplikace?

11 odpovědí

| |
|---|
| Jsou dostupné |
| Snadná dostupnost |
| Skvělé na pochopení základních příkazů, podmínek, cyklů a rekurzivního volání. |
| Byla jsem na semináři o scratchi a legu. |
| Byly první, které jsem začal používat. |
| Buď jsou přístupné online a zdarma - ty preferuji i z hlediska využitelnosti žáky. Scottie GO, MicroBit a Mindstorms mám pro zpestření výuky, protože jsme si je pořídila soukromě. |
| Asi byla jedna z prvních, ke kterým jsem se dostala. Děkuji za užitečné odkazy na další. |
| ne |
| přehlednost |
| zakoupeny ozoboty |
| dozvěděla jsem se o nich a vyhovují mi, tak jsem neměla potřebu hledat jiné |

Co Vám na Vámi využívaných aplikacích chybí, co byste změnil/a?

9 odpovědí

| |
|---|
| Nevím |
| Podpora novějších operačních systémů bez použití DosBoxu. |
| Podrobný manuál. Zásobárna úkolů pro učitele. |
| Chybí vypracovaná metodika. A pokud je, tak se musí pracně hledat. Metodika by měla být udělána dříve, než začne tento způsob výuky (a obsahu informatiky) povinný! |
| Možná větší zásobník připravených úkolů. |
| Zatím nic, opravdu jsem na začátku. |

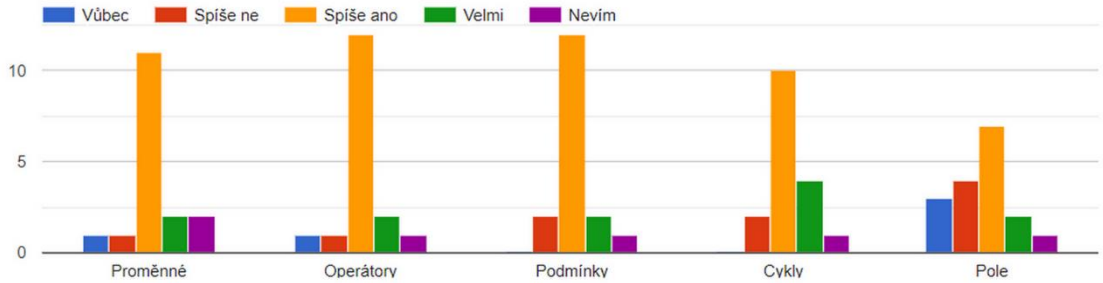
nic

nic mě nenapadá

jazyk

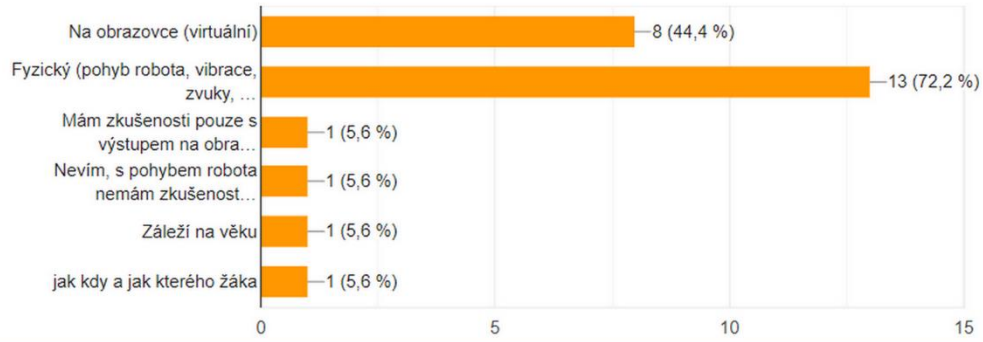
## Oblasti programování

Dokáže na základě Vašich zkušeností Vámi používané prostředí dostatečně objasnit oblasti programování?




Jaký výstup podle Vašich zkušeností více motivuje žáky pro napsání algoritmu v blokovém programování?

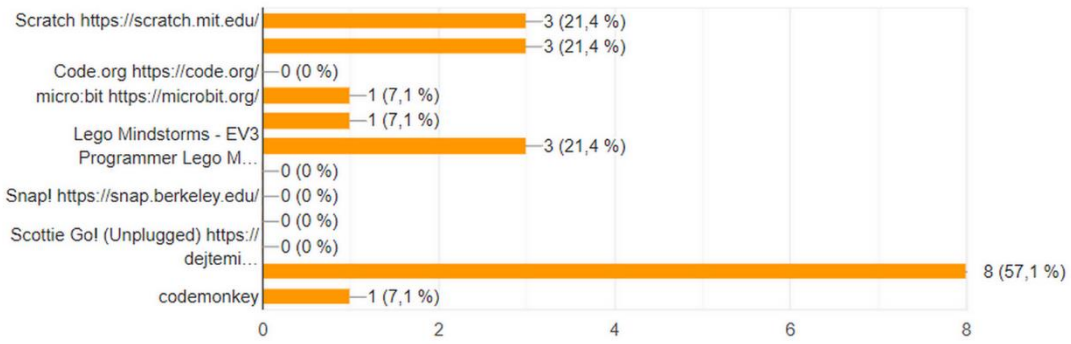
18 odpovědí




## Vývojová prostředí

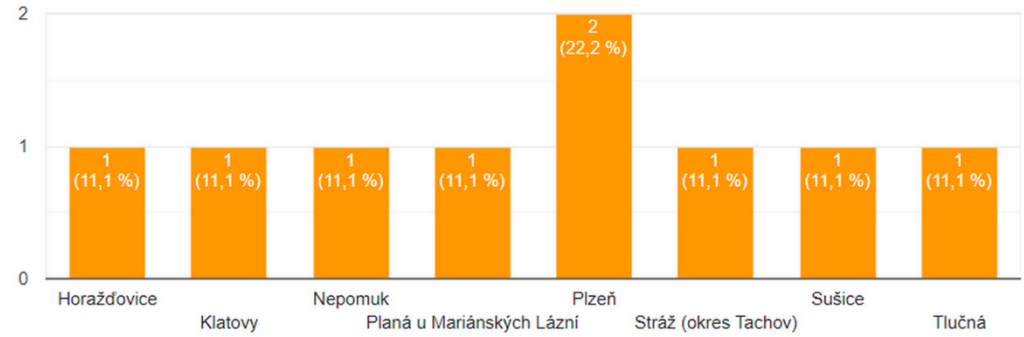
Viděl/a jste někdy některá z uvedených aplikací? Pokud ano vyberte je. Pokud znáte jiné, vyplňte v možnosti "Jiná..."

14 odpovědí



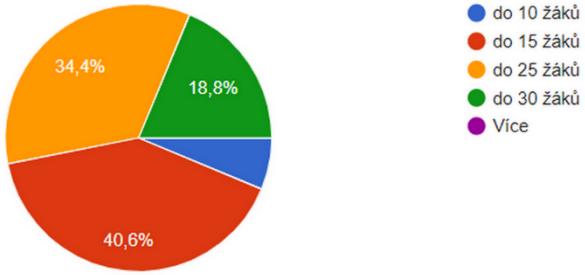
Název obce, kde ZŠ sídlí (chcete-li sdělit).

9 odpovědí




Jaký je průměrný počet dětí ve Vaší výuce informatiky?

32 odpovědí




Kolik žáků byste si představoval/a ve výuce?

27 odpovědí