Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

DIPLOMOVÁ PRÁCE

Programování na základní škole a rozvoj algoritmického myšlení žáků

Programming in elementary school and development of students´ algorithmic thinking

Bc. Lucie Milichovská

Vedoucí práce:	PhDr. Jiří Štípek, Ph.D.
Studijní program:	Učitelství pro střední školy
Studijní obor:	Učitelství všeobecně vzdělávacích předmětů pro základní školy a střední školy - informační a komunikační technologie

2020

Odevzdáním této diplomové práce na téma Programování na základní škole a rozvoj algoritmického myšlení žáků potvrzuji, že jsem ji vypracovala pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Sloup v Čechách, 2. 5. 2020

Poděkování

Tímto bych chtěla poděkovat PhDr. Jiřímu Štípkovi, Ph.D. za vedení diplomové práce a poskytovanou zpětnou vazbu při tvorbě úloh.

UNIVERZITA KARLOVA
PEDAGOGICKÁ FAKULTA
Katedra informačních technologií a technické výchovy

ZADÁNÍ DIPLOMOVÉ PRÁCE

akademický rok 2019/2020

Jméno a příjmení studenta: Bc. Lucie Milichovská

Studijní program: N7504 Učitelství pro střední školy

Studijní obor: Informační a komunikační technologie

Název tématu práce v českém jazyce:

Programování na základní škole a rozvoj algoritmického myšlení žáků

Název tématu práce v anglickém jazyce:

Pokyny pro vypracování:

- Seznámit se s pojmem algoritmické myšlení a možnostmi jeho rozvoje u vybrané skupiny dětí školního věku s pomocí softwarových prostředků.
- Zmapovat soudobé přístupy, softwarové prostředky a nástroje pro rozvoj algoritmického myšlení dětí.
- Pro zvolenou věkovou skupinu vybrat vhodné nástroje či prostředky pro programování a připravit ucelený soubor aktivit orientovaných na rozvoj algoritmického myšlení.
- Ověřit navržený soubor aktivit v praxi.

Vedoucí práce: PhDr. Jiří Štípek, Ph.D.

Datum zadání práce: 17.9.2019

V Praze dne: 17.9.2019

.....

PhDr. Jiří Štípek, Ph.D.
vedoucí katedry

Zadání převzal(a) dne:

podpis .....

ABSTRAKT

Tato diplomová práce se zabývá rozvojem algoritmického myšlení a výukou programování u žáků na základní škole. Věnuje se dostupným možnostem a nástrojům využitelných v hodinách.

Praktická část práce je zaměřena na dětský programovací jazyk Scratch, což je jeden z nástrojů určených pro využití ve výuce. Hlavním cílem práce je vytvoření ucelené sbírky úloh orientovaných na rozvoj algoritmického myšlení žáků ve věku 9 – 12 let. Úlohy mají postupně gradující charakter a žáci nepotřebují žádné předchozí zkušenosti s programováním ani programovacím jazykem Scratch, zároveň jsou koncipovány tak, aby je žáci zvládli vyřešit i bez pomoci učitele. Všechny úlohy byly ověřeny na skupině žáků dané věkové kategorie. Pro možnost dalšího využití je sbírka úloh dostupná formou webové prezentace.

KLÍČOVÁ SLOVA

algoritmické myšlení, programování na základní škole, nástroje pro rozvoj algoritmického myšlení žáků, dětské programovací jazyky, Scratch

ABSTRACT

This thesis deals with development of algorithmic thinking and teaching programming in elementary school. It focuses on available ways and tools suitable for classes.

The practical part of the thesis is focused on children's programming language Scratch, which is one of the tools designed to be used for teaching. The main goal is to create comprehensive collections of tasks that develop algorithmic thinking of pupils aged 9 - 10 years. The tasks get more complex gradually so that the pupils don't need any previous programming experience. Also they are designed so that they can be solved without the assistance of a teacher. All the tasks were checked against a set of pupils in the given age range. The collection of tasks is also made available as a web presentation for the ease of further use.

KEYWORDS

algorithmic thinking, programming at primary school, learning tools and programming languages, children's programming languages, Scratch

Obsah

Úvod

Tato diplomová práce se věnuje problematice rozvoje algoritmického myšlení žáků základních škol. Jde o téma, které je v posledních letech předmětem zájmu řady odborníků i učitelské veřejnosti a kterému je přikládán stále větší význam.

Možností, jak se se žáky základní školy věnovat rozvoji algoritmického myšlení a programování, se nabízí několik. Výuka může probíhat za pomoci klasických programovací jazyků, které mají výhodu v tom, že je mohou žáci využít i v pozdějším věku, někteří se s nimi setkají při dalším studiu nebo v zaměstnání. Další možností je využití nástrojů určených přímo pro výuku žáků. Těchto nástrojů mají učitelé k dispozici širokou škálu. Do této kategorie patří robotické hračky, robotické stavebnice, předpřipravené hry zaměřené na rozvoj algoritmického myšlení a dětské programovací jazyky.

Tato práce se zaměřuje na dětský programovací jazyk Scratch, který byl navržen pro rozvoj algoritmického myšlení. Scratch nabízí široké možnosti pro výuku programování, umožňuje žákům vlastní volnou tvorbu, je přehledný, má intuitivní ovládání a poskytuje téměř okamžitou zpětnou vazbu formou spustitelného programu, díky tomu mají žáci možnost vidět, co jejich kód dělá.

Praktická část práce obsahuje ucelenou sbírku úloh určených pro žáky v 4. – 6. třídě, kteří s programováním začínají. Obtížnost úloh postupně graduje a díky tomu, žáci postupně rozšiřují své dovednosti. Tyto úlohy obsahují zadání pro žáky ve dvou verzích, u první verze se jedná o pracovní postup tvorby a u druhé verze se jedná pouze o stručný popis toho co má program dělat, zároveň je vždy součástí přesné řešení určené pro učitele a také je k dispozici seznam problémů, či komplikací, které mohou v každé úloze nastat. Všechny úlohy byly ověřeny v rámci zájmového kroužku danou věkovou skupinou. Aby úlohy byly dále využitelné vznikla webová prezentace dostupná na adrese http://scratch.milichovska.cz, kde jsou všechny úlohy volně k dispozici.

1 Cíl práce

Tato práce se zabývá rozvojem algoritmického myšlení a výukou programování žáků základní školy.

V teoretické části se zabývám dostupnými možnostmi, jak žáky na základní škole seznámit s programováním a rozvojem algoritmického myšlením.

V praktické části se následně zabývám tvorbou ucelené sbírky úloh a jejich ověřením v praxi.

1.1 Cíl práce

Cílem práce je sestavit ucelený soubor aktivit orientovaných na rozvoj algoritmického myšlení žáků ve věku 9 až 12 let a následně jej ověřit v praxi.

Tato sbírka by měla obsahovat úlohy, které by měly být zaměřené na žáky navštěvující 4. – 6. třídu základní školy, protože v tomto věku, již ovládají dovednosti jako čtení, psaní a základní ovládání počítače a jsou již dostatečně vyspělí, aby pochopili základní koncepty algoritmizace pomocí nástrojů pro výuku programování. Sbírka úloh by měla předpokládat nulové dosavadní zkušenosti s programováním, protože žáci v tomto věku se na základní škole s programováním obvykle setkají poprvé. Úlohy by měly být navrženy tak, aby žáci byli schopni je zpracovávat samostatně nebo v rámci výuky za malé pomoci učitele. Cílem úloh by mělo být zábavnou formou rozvíjet algoritmické myšlení žáků, protože výuka, která žáky baví je zároveň motivuje k dalšímu poznávání. Sbírka vytvořených úloh by měla být dostupná pro učitele k využití ve výuce.

1.2 Vymezené úkoly

Úkol 1: Zmapovat možnosti rozvoje algoritmického myšlení u žáků základní školy

- Prozkoumat dostupné možnosti nástrojů určených pro programování a rozvoj algoritmického myšlení žáků.
- Z těchto možností vybrat vhodný nástroj pro tvorbu úloh.

Úkol 2: Sestavit sbírku úloh ve vybraném dětském programovacím jazyce

- Navrhnout ucelený soubor úloh, které budou vhodné pro žáky navštěvující 4 - 6. třídu základní školy.
- Sestavit pro tyto úlohy zadání pro žáky, řešení a metodické poznámky pro učitele, seznam možných problémů a úlohy naprogramovat.

Úkol 3: Sbírku úloh ověřit na skupině žáků příslušné věkové skupiny

- Ověřit vytvořené úlohy v rámci zájmového kroužku programování a nechat je žáky samostatně řešit.
- Na základě problémů a komplikací, které se vyskytnou, vhodně upravit zadání úloh a případně doplnit seznam možných problémů a komplikací.
- Pozorovat žáky při práci v následujících aspektech: orientace ve vybraném prostředí, samostatná práce žáků na úlohách, práce se zadáním a časová náročnost úloh.

Úkol 4: Zajistit vhodnou formu prezentace úloh

- Zvolit vhodnou formu prezentace úloh, aby mohly být znovu použity případně i dalšími učiteli.
- Následně sbírku úloh publikovat zvolenou formou prezentace.

1.3 Metodologie

První část plnění vymezených úkolů souvisí se zmapováním soudobých možností, jak zapojit programování a rozvoj algoritmického myšlení do výuky žáků základní školy. Následně z dostupných možností vybrat vhodný nástroj pro rozvoj algoritmického myšlení žáků.

Druhá část plnění vymezených úkolů řeší tvorbu uceleného souboru úloh. Úlohy budou určeny pro věkovou skupinu žáků navštěvující 4.-6. třídu. Žáci nemusí mít žádné vstupní znalosti a zkušenosti v oblasti programování.

Třetí část plnění vymezených úkolů ověřuje sestavené úlohy v praxi. Jejich ověřování bude probíhat v rámci zájmového kroužku programování, který navštěvuje 8 žáků. Na základě ověřování úloh vznikne seznam možných problémů a komplikací při řešení navržených úloh.

Při ověřování úloh v praxi dojde k pozorování žáků v následujících aspektech: orientace žáků ve vybraném prostředí, samostatná práce žáků na úlohách, práce se zadáním a časová náročnost úloh. Na základě pozorování budou tyto aspekty zhodnoceny, což poskytne informace o vhodnosti výběru nástroje a adekvátní obtížnosti úloh pro zvolenou věkovou skupinu žáků.

Čtvrtá část plnění vymezených úkolů spočívá ve vytvoření vhodné prezentace sbírky úloh, aby mohla být opětovně kýmkoliv využita. A to jako celá sbírka nebo pouze jednotlivé vybrané úlohy.

2 Teoretická část

Tato část práce se zabývá začleněním informačních a komunikačních technologií do výuky. Dále termíny informatické myšlení, algoritmické myšlení a programování, jejich souvislostmi s výukou.

Dále jsou zde popsány soudobé možnosti a nástroje pro možnost programování a rozvoj algoritmického myšlení u žáků základní školy.

2.1 Informační technologie na základní škole

2.1.1 Pojem informační technologie

Pro Informační technologie je využívána zkratka IT, pojem je překladem anglického pojmu Information technology. Jedná se o odvětví techniky, které se zabývá prací s informacemi, jejich tvorbou, shromažďováním, zpracováváním, vyhledáváním, uchováváním, zpřístupňováním, organizováním a využíváním, zároveň řeší automatizaci zmíněných procesů.

Pojem informační technologie byl využíván na počátku počítačové éry ve vzdělávání (ve 2. pol. 20. století). S rychlým rozvojem výpočetní techniky byl tento termín nahrazen vhodnějším termínem informační a komunikační technologie. (Maněnová, 2012)

Pojem informační a komunikační technologie, pro který se využívá zkratka ICT nebo IKT, vychází z anglického pojmu Information and Communication Technologies, a zahrnuje veškeré technologie využívané pro práci s informacemi a komunikaci. Obě tyto odvětví techniky zahrnují hardwarové i softwarové vybavení počítače.

2.1.2 Informační a komunikační technologie ve výuce

Rychlý rozvoj oboru informační a komunikační technologie velmi ovlivňuje současnou společnost. Vědění je v dnešní době produkováno a distribuováno pomocí informačních a komunikačních technologií do vzdělávacího procesu, a proto jsou značná očekávání spojená s implementací ICT do škol. (Maněnová, 2012)

Informační a komunikační technologie jsou tedy do výuky stále více začleňovány, a to nejen v samostatném předmětu, ale v rámci všech předmětů. Technologie vzbuzují u žáků zájem a zvyšují jejich motivaci věnovat se dané činnosti. Neměly by však být pouze cílem výuky, ale zejména prostředkem výuky určeným k dosažení cílů.

Informační a komunikační technologie ve výuce plní různé funkce a také ovlivňují vztah mezi učitelem a žáky. V knize Učitelé a technologie autoři uvádí pět možných způsobů začlenění technologií do výuky. (Zounek, Šedová, 2009)

- ICT jako nosič obsahu (využití pro výklad látky)
- ICT jako extenze (rozšíření smyslové nebo mentální schopnosti uživatelů)
- ICT jako pracovní nástroj (učitelé i žáci pomocí technologií vytváří své výstupy)
- ICT jako testovací nástroj (testování žáků pomocí technologií učitelům usnadňuje práci s opravováním testů)
- ICT jako kulisa a doplněk (použití technologií jako oživení či zpestření hodiny)

Využití informačních a komunikačních technologií ve výuce ovlivňuje i pojetí role učitele, který přijímá roli poradce a průvodce žáků při učení. Učitel reaguje na potřeby žáků a pro žáky se stává mediátorem (organizuje, monitoruje a hodnotí interakci žáků s ICT, ale zůstává v pozadí, do činnosti žáků vstupuje pouze, když nastane problém) nebo partnerem (učitel se dostává na úroveň žáka a řeší podobné úlohy za použití ICT).

2.2 Programování na základní škole

2.2.1 Informatické myšlení

Pojmem Informatické myšlení je českým překladem pojmu Computational thinking, používaného v zahraničí. Jedná se o univerzální a na různé oblasti aplikovatelnou dovednost. Stále neexistuje jednotná definice. Za velmi zjednodušenou definici pojmu lze považovat, že informatické myšlení je schopnost *„myslet jako informatik při řešení problémů“*. (Lessner, 2015)

Jedná se o myšlenkové postupy využité při formulování a řešení problémů. Součástí informatického myšlení by měly být následující schopnosti:

- pochopení aspektů, které je možné řešit strojově
- vyhodnocování souvislostí mezi informatickými prostředky a problémem
- porozumění možnostem a případným omezením informatických prostředků
- používání informatických prostředků novými způsoby nebo v nových situacích
- používání informatického myšlení v jakékoliv oblasti. (Wing, 2010)

Velmi využívanou definicí Informatického myšlení vytvořila Mezinárodní společnost pro technologie ve vzdělávání - International Society for Technology in Education (ISTE) a Asociace učitelů informatiky - Computer Science Teachers Association (CSTA). Oblíbenost této definice spočívá v její struktuře, charakteristika Informačního myšlení je zde dostatečně konkrétní, aby umožnila plánování výukových aktivit. (Lessner, 2015)

Podle definice ISTE a CSTA by měl žák zvládnout:

- formulovat problémy tak, aby bylo možno je vyřešit za pomoci počítače
- logicky organizovat a analyzovat data
- reprezentovat data pomocí modelů a simulací
- automatizovat řešení pomocí algoritmického myšlení (jako posloupnost kroků)
- najít a prozkoumat různá řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů
- zobecnit nalezené řešení a zvládnout jej použít k řešení dalších problémů (ISTE a CSTA, 2011)

Zároveň během zvládnutí tohoto procesu by měl žák:

- najít sebejistotu při řešení problémů
- vytrvat i při řešení složitějších problémů
- vypořádat se s otevřenými a nejednoznačně formulovanými problémy
- být schopný řešit otevřené problémy
- zvládnout komunikovat a spolupracovat s ostatními při řešení společných problémů (ISTE a CSTA, 2011)

Žáci by se měli učit soustředit se na samostatné hledání a analýzu problémů a jejich následné řešení za pomoci počítače, a ne se pouze seznamovat s hotovými postupy. Informatické myšlení umožňuje řešit problémy, které jsou otevřené a komplexní, tedy není možné jejich přímočaré a mechanické řešení.

Informatické myšlení umožňuje racionální rozhodování i v obtížnějších situacích a napomáhá dopady rozhodnutí předvídat. Znásobuje možnosti každého člověka ve světě, ve kterém roste význam informací a kde je výpočetní technika všudypřítomná. Pomáhá přitom technickému vývoji, jeho důsledkům, rychlosti, i principiálním limitům porozumět. Jeho cílem není výchova populace programátorů, ale je v různé míře užitečné pro každého. (Lessner, 2015)

2.2.2 Algoritmické myšlení

Pro pochopení pojmu Algoritmické myšlení, je třeba seznámit se s pojmy *algoritmizace* a *algoritmus*. Algoritmizace se zabývá návrhem, analýzou a konstrukcí algoritmů. Algoritmus je přepis posloupnosti kroků, které vedou k danému cíli, typicky tedy vyřešení problému. Algoritmem se tedy rozumí postup splňující následující požadavky:

- Elementárnost (algoritmus se skládá z jednoduchých kroků)
- Konečnost (počet kroků musí být konečný)
- Obecnost (neřeší jen jeden konkrétní problém, ale obecnou třídu podobných problémů)
- Determinovanost (po každém provedeném kroku musí být jasné, kterým krokem se pokračuje, případně že algoritmus skončil)
- Výstup (algoritmus po skončení své činnosti má alespoň jeden výstup a tím tvoří odpověď na problém, který řešil)

Algoritmy jsou tvořeny třemi základními strukturami (= algoritmické struktury) a to posloupností příkazů neboli kroků seřazených v konkrétním pořadí. Větvením neboli rozhodováním podle toho, zda je určitá podmínka splněna či nikoliv. A cykly neboli opakováním určitých kroků, počet opakování může být pevný nebo daný podmínkou.

Pojem algoritmus je často spojován s využitím výpočetní techniky, čímž většině lidí přijde jejich tvorba složitá a mají ji spojenou s odborným vzděláním. Lidé si často neuvědomují, že se s algoritmy setkávají ve svém každodenním životě. Většinou v pasivní formě, kdy za předpokladu dodržení předepsaných kroků dosahují požadovaných výsledků. Příkladem algoritmu, ke kterému není třeba využití výpočetní techniky, může být kuchařský recept. Také technologie, se kterými se člověk denně setkává jsou řízeny algoritmy, jako například telefony, semafory či spotřebiče v domácnosti. (Svoboda, 2018)

Algoritmické myšlení je dle jedné z definic soubor schopností, které člověk užije ke konstruování algoritmů a porozumění jim. Do těchto schopností patří: přesné určení problému, analýza problému, nalezení příkazů odpovídajících danému problému, sestavení postupu pomocí příkazů, přemýšlení o souvislostech v rámci problému a zvýšení efektivity nalezeného řešení. Algoritmické myšlení také mimo výše uvedených schopností vyžaduje

vlastnosti jako abstraktní myšlení, logické myšlení, strukturované myšlení, tvořivost a schopnost řešit problémy. (Futschek, 2006)

Schopnost algoritmického myšlení však není jen o vytváření postupů, ale neméně důležité je těmto postupům opravdu porozumět. (Vaníček, 2016) Jen tak je člověk schopný nacházet nejefektivnější řešení, ladit jej a případně v něm nacházet chyby.

2.2.3 Programování

Programování je proces tvorby programu, který obsahuje následující kroky: analýza problému, pochopení problému, nalezení vhodného algoritmu a zápis zdrojového kódu v cílovém programovacím jazyce a jeho testování. Cílem programování je nalezení sekvence instrukcí, které počítač dokáže provést a tím vyřešit daný problém.

Každý program vyžaduje programátora, který jej vytvoří a zapíše, a také procesor, který vytvořený program vykoná. Program pracuje se zadanými daty, jejichž vlastnosti jsou mu jasně popsány a také jsou mu jasně dány příkazy, které mají být s daty provedeny. (Klement, 2002)

Programovací jazyk

Programovací jazyk je komunikačním nástrojem mezi programátorem a počítačem a je prostředkem pro zápis algoritmů. Algoritmus zapsaný v určitém programovacím jazyce se nazývá program.

Jedná se o uměle vytvořené jazyky, sloužící k tvorbě počítačových programů. Mají danou syntaxi neboli systém pravidel a symbolů, kterými se řídí formální zápis programu a sémantiku, jež určuje význam programu. (Klement, 2002)

Programovací nástroj

Programovací nástroje slouží k usnadnění práce programátorům, tím že usnadňují vývoj programů. Jedním z programovacích nástrojů jsou vývojová prostředí (= rozhraní pro snazší vývoj programu v konkrétním programovacím jazyce).

2.2.4 RVP pro základní školy v souvislosti s programováním

RVP je zkratkou pro Rámcový vzdělávací program, jedná se o obecně závazný rámec pro tvorbu školních vzdělávacích programů škol. V České republice jsou popsány a ukotveny Školským zákonem. (Zákon č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborném a jiném vzdělávání) Stanovují konkrétní cíle, formy, délku a povinný obsah vzdělávání, organizační uspořádání vzdělání, profesní profil, podmínky průběhu a ukončování vzdělávání a zásady pro tvorbu školních vzdělávacích programů. (RVP ZV, 2017)

Rámcový vzdělávací program definuje klíčové kompetence, které představují souhrn základních vědomostí, dovedností, schopností, postojů a hodnot důležitých pro osobní rozvoj a uplatnění každého člena společnosti. Tyto kompetence mají žáka připravit na další vzdělávání a uplatnění ve společnosti. S informatickým myšlením souvisí Kompetence k řešení problémů, která se týká rozpoznání problémových situací, vyhledávání informací, řešení problémů, ověřování řešení a kritického myšlení.

Dále Rámcový vzdělávací program definuje vzdělávací oblasti. Přičemž Informační a komunikační technologie jsou jednou z nich. Očekávaným výstupem žáka po absolvování prvního a druhého stupně povinné školní docházky jsou uživatelské dovednosti obsluhy počítače, komunikace pomocí počítače, práce s určitým programovým vybavením počítače, jako je například textový či tabulkový editor, vyhledávání informací a také znalost bezpečné a legální práce s technologiemi. Časová dotace pro tuto vzdělávací oblast je na prvním stupni 1 hodina a na stupni druhém také 1 hodina, z toho důvodu již není možné v této oblasti najít prostor pro rozvoj dalších digitálních kompetencí.

Rozvoj digitální gramotnosti v posledních letech zaznamenal velký vzestup a programování a algoritmické myšlení je jednou z jeho oblastí, o které se stále více mluví. Mnoho odborníků je přesvědčeno, že v tak digitálně vyspělém prostředí, které dnes mladé lidi obklopuje nemůže člověk zůstat pouhým uživatelem a pozorovatelem. Měl by pochopit, jak všudypřítomné technologie fungují. Bohužel, ze všech oblastí digitálních dovedností čelí tato oblast v České republice zřejmě největšímu nepochopení a předsudkům. Mnozí lidé, včetně učitelů, si programování představují jako kódování, a že jeho výuka spočívá pouze ve psaní kódu. Nicméně díky možnostem, které jsou k dispozici, je rozvoj programování

a algoritmického myšlení na základní škole založen na hře a tvořivosti. Je až s obdivem, že znění vzdělávací oblasti Informační a komunikační technologie v Rámcovém vzdělávacím programu pro základní vzdělávání se od svého vzniku v roce 2004 nezměnilo. (Neumajer, 2017)

V roce 2018 vydal Národní ústav pro vzdělávání dokument Návrh revizí rámcových vzdělávacích programů v oblasti informatiky a informačních a komunikačních technologií. (Národní ústav pro vzdělávání, 2018) Tento dokument mimo jiné vymezuje pojmy „Digitální gramotnost“ a „Informatické myšlení“.

Dle výše zmíněného dokumentu je digitální gramotnost souborem digitálních kompetencí (vědomostí, dovedností, postojů, hodnot), které jedinec potřebuje k bezpečnému, ke kritickému, sebejistému a tvořivému využívání digitálních technologií při práci, při učení, ve volném čase i při svém zapojení do společenského života.

Informatické myšlení je v tomto dokumentu definováno jako způsob uvažování, který jedinci umožňuje rozpoznávat informatické aspekty světa a využívat informatických prostředků k porozumění a uvažování o přirozených i umělých systémech a procesech. Informaticky myslící žák je schopen při řešení nejrůznějších situací:

- rozpoznat a formulovat problémy s ohledem na jejich řešitelnost
- získat, zaznamenat, uspořádat, strukturovat a předávat data a informace
- rozložit systémy a procesy na části, odhalovat jejich vztahy a strukturu a modelovat situace
- vytvořit a formulovat postupy a řešení, která lze přenechat k vykonání jinému člověku nebo stroji
- vytvářet formální popisy skutečných situací a pracovních postupů
- testovat, analyzovat, vyhodnocovat, porovnávat a vylepšovat uvažovaná řešení

V této revizi je zpracována v rámci očekávaných výstupů oblast s názvem Algoritmizace a programování, která popisuje, co by měl žák v souvislosti s touto oblastí zvládnout.

Očekávané výstupy pro 1. a 2. stupeň základní školy v oblasti algoritmizace a programování (Národní ústav pro vzdělávání, 2018)

1. stupeň ZŠ	2 stupeň ZŠ
Žák přečte textový nebo symbolický zápis algoritmu a vysvětlí jeho jednotlivé kroky.	Žák po přečtení jednotlivých kroků algoritmu nebo programu vysvětlí celý postup; určí problém, který je daným algoritmem řešení.
Žák popíše jednoduchý problém, navrhne a popíše jednotlivé kroky jeho řešení.	Žák rozdělí problém na jednotlivě řešitelné části a navrhne a popíše kroky k jejich řešení.
Žák upraví připravený postup pro obdobný problém; ověří správnost jím navrženého postupu, najde a opraví v něm případnou chybu.	Žák upraví daný algoritmus pro jiné problémy, ověří správnost postupu navrženého i někým jiným, najde a opraví v něm případnou chybu.
Žák rozpozná různé algoritmy, které vedou ke stejným výsledkům.	Žák navrhne různé algoritmy pro řešení problému; vybere z více možností vhodný algoritmus pro řešený problém a svůj výběr zdůvodní.
Žák v blokově orientovaném programovacím jazyce sestaví program; program otestuje a opraví v něm případné chyby.	Žák v blokově orientovaném programovacím jazyce sestaví přehledný program pro vyřešení zadaného problému; program otestuje a opraví v něm případné běhové a logické chyby.
Žák rozpozná opakující se vzory, používá opakování a připravené podprogramy; používá události ke spouštění podprogramů.	Žák používá opakování, větvení programu, proměnné, podprogramy s parametry; používá události k paralelnímu spouštění podprogramů.

Tabulka 1 - Očekávané výstupy pro 1. a 2. stupeň základní školy v oblasti algoritmizace a programování

2.2.5 Programování ve výuce i mimo ni

Programování a algoritmické myšlení je tedy na základě aktuálního znění Rámcového vzdělávacího programu zařazeno pouze nad rámec běžné výuky, neznamená to však, že se s ním žáci nemohou na základní škole setkat.

Mnohé školy se v rámci výuky zapojují do celosvětové kampaně Hodina kódu (https://hourofcode.com/). Cílem této kampaně je, aby se žáci zábavnou formou seznámili s programováním a algoritmickým myšlením. Školám jsou poskytnuty materiály nevyžadující předchozí přípravu či znalosti žáků ani učitelů. Ministerstvo školství, mládeže a tělovýchovy tuto aktivitu vítá. (MŠMT, 2016)

Další kampaní pro podporu programování je například Code Week (https://codeweek.eu/) jehož cílem je zviditelnit programování a ukázat veřejnosti, jak je díky programování možné realizovat myšlenky. Jedná se zejména o propagační kampaň, která nechává volné ruce dobrovolníkům v tom, co v rámci ní budou s žáky tvořit. (Lessner, 2014)

Kromě aktivit zaměřených na programování a rozvoj algoritmického myšlení, které jsou realizovány během výuky, ať už v rámci kampaně nebo díky vlastní iniciativě učitele, se žáci mohou s touto problematikou setkat jako s náplní mimoškolních aktivit, typicky zájmových kroužků. Tyto mimoškolní aktivity jsou pořádány buď jako součást nabídky zájmových kroužků v rámci školy, nebo jsou nabízeny soukromými firmami.

Bylo by chybou si myslet, že cílem výuky programování na základní škole má být vychovat z žáků budoucí programátory. Primárním cílem je naučit žáky algoritmickému myšlení, dále pak analýze problému a dovedení myšlenky k formalizovanému návrhu algoritmu. (Pitner, 2000)

Dalším důležitým aspektem zařazení programování do výuky je podpora tvořivosti, konstruktivního a kreativního myšlení žáků. Každé dítě je ze své podstaty zvídavé a pro mnoho dětí je naplňující, když pomocí počítače mohou zrealizovat svůj nápad, či převést svou myšlenku do fungující podoby.

2.3 Možnosti výuky programování pro žáky na základní škole

Existuje celá řada možností a nástrojů, za pomoci kterých, je možné žáky učit programování a rozvíjet jejich algoritmické myšlení. Tyto nástroje lze dělit na dvě základní kategorie.

Profesionální nástroje a programovací jazyky

První z nich jsou nástroje profesionální, jejichž cílovými uživateli jsou profesionální programátoři. (Pittner, 2000)

Jejich výhodou je to, že se žáci se od začátku učí programovat v prostředí, které pak mohou využít i mimo školu, například v zaměstnání. Výsledkem jejich práce je univerzální, použitelný a dále šiřitelný program. Žáci se také neučí zlozvyků (ne úplně správným zápisům, které mnohdy obsahují výukové nástroje pro usnadnění). Možnosti toho, co lze pomocí profesionálních programovacích jazyků a nástrojů vytvořit nejsou v podstatě nijak omezeny.

Mezi nevýhody patří poměrně složitý způsob ovládání s ohledem na věk žáků a nutnost zvládnutí základních principů programování předem. Výsledky práce nejsou vždy pro žáky dostatečně atraktivní, protože pokud nejsou vyloženy základní programové konstrukce, těžko se učiteli podaří vymyslet nějaké příklady, které by studentům připadaly zajímavé. Počáteční výklad bývá obvykle provázen drilovými úlohami, které žáci řeší většinou mechanicky bez dostatečné motivace si vše zapamatovat. (Pecinovský, 2001)

Do této kategorie patří například Java, C, C++, Python, JavaScript, PHP, Scala a tak dále.

Nástroje a programovací jazyky určené pro výuku

Druhou kategorií jsou speciální výukové nástroje a programovací jazyky. Jedná se o nástroje, které jsou cíleně vytvořeny pro výuku programování a rozvoj algoritmického myšlení u žáků. Obvykle se jedná o předpřipravené prostředí, ve kterém žáci pomocí ať už skládání bloků příkazů, nebo zjednodušeným zápisem příkazů, tvoří pro ně zajímavý produkt.

Výhodou je jednoduchost těchto nástrojů. Ovládání a zápis kódu je zajištěn tak, aby jej zvládli i mladší uživatelé. Tato výhoda však přináší poměrně omezené možnosti v tom, co můžeme za pomoci nástroje udělat, a v některých případech i ne úplně korektní zápis. Pro žáky jsou právě díky jednoduchosti ovládání, konkrétnosti (dělají něco vizuálně konkrétního, například stačí kousek kódu a postava nebo robotická hračka se hýbe) a téměř okamžité zpětné vazbě, velice atraktivní. Žáci snáze dosahují cíle a rychleji získávají pocit úspěchů.

Zásadní roli zde má motivace žáků. Programování jednoduchých her je jednou z možností, jak žáky v rámci výuky programování motivovat. Díky nástrojům určeným přímo pro výuku programování jsou studenti více motivováni k tvořivé činnosti a rychleji pochopí základní pojmy a algoritmické konstrukce. (Kanáliková, 2015)

Nepřináší pouze výhody, jak už bylo zmíněno výše, jejich možnosti jsou omezené. Mohou, ač třeba nechtěně, učit žáky určitým chybám nebo zlozvykům, které následně v klasických programovacích jazycích nebudou fungovat. Jsou uzavřené samy do sebe. (Pittner, 2000)

Nástroje a programovací jazyky pro výuku jsou tedy vhodné zejména pro rozvoj algoritmického myšlení s ohledem na věk žáků, protože díky nim se seznámí se základními principy programování, pochopí je, a vše probíhá pro žáky zábavnou, motivující a zajímavou formou.

Podrobněji se prostředky na rozvoj algoritmického myšlení zabývá kapitola 2. 4. Nástroje pro výuku programování.

2.3.1 Věk žáků

Žáci na základní škole prochází rychlým vývojem, mezi žákem v první třídě a ve třídě deváté je obrovský rozdíl. Je zcela samozřejmé, že k výuce programování je nutné přistupovat způsobem zohledňujícím věk.

Úroveň abstraktního myšlení, potřebného pro pochopení základních konceptů programování může být s ohledem na věk rozdílná. Švýcarský psycholog Jean Piaget vytvořil teorii kognitivního vývoje, ve které definoval čtyři základní stádia vývoje:

- senzomotorické stadium (věk 0–2 roky) – dítě dokáže odlišit sebe od objektů, rozeznává sebe jako aktivního činitele a objevují se počátky záměrného jednání
- předoperační stadium (věk 2–7 let) – dítě se učí využívat jazyk, objekty reprezentuje pomocí představ a slov, předměty třídí dle jednoho rysu (barva, tvar, materiál), myšlení je egocentrické (tzn. nevnímá názory druhého)
- stadium konkrétních operací (věk 7–12 let) – dítě dokáže logicky přemýšlet v operacích, objektech, událostech, předměty třídí podle různých vlastností a dokáže je logicky seřadit (nejtmavší – nejsvětlejší, největší – nejmenší)
- stadium formálních operací (věk 12 let a výše) – dítě dokáže myslet logicky o abstraktních pojmech a systematicky testuje hypotézy; zabývá se abstrakcí, budoucnostní, ideologickými problémy (Piaget, 2014)

Schopnost pochopit koncepty programování samozřejmě nevychází pouze z věku žáků, jsou zde i důležité aspekty individuálního vývoje, sociálního prostředí, vlastní motivace a mnoha dalších. Věk je měřitelnou jednotkou a může usnadnit orientaci ve vývoji žáka, ale vždy je potřeba si uvědomit, že každý žák je individuální osobnost.

Při výuce jakéhokoliv předmětu je nutné přihlížet k individuálním zvláštnostem ve schopnostech, vědomostech, dovednostech, zkušenostech, ale i k životní situaci, vývojovým zvláštnostem a problémům, které přináší změny v žákově životě. Mimo to je nutné zohlednit i věkovou vyspělost a zralost. (Smékal, 2006)

Na základní škole je pozornost soustředěna zejména na zvládání základních algoritmizačních obratů, jako jsou posloupnosti příkazů, větvení a cykly. V 4.-5. třídě je

vhodné žáky seznámit zejména s hotovými programy, jednoduchými příkazy a sekvencemi příkazů. V 6.-7. třídě se žáci seznamují s cykly, větvením a proměnnými. (Kopecká, 1998, zmíněno v Pitner, 2000)

Seznamovat žáky s programováním a algoritmickým myšlením je možné už od předškolního věku. Nicméně je nezbytné zvážit, kdy a jaký nástroj využít. Pokud je výběr některého z nástrojů příliš uspěchaný, může to zkomplikovat situaci žákům do budoucna, protože ve chvíli, kdy jsou dostatečně vyspělí na to, aby pro ně daný nástroj měl maximální přínos, mohou mít pocit, že již dané aktivity v něm dělali. Ovšem podobný problém může nastat i ze strany druhé, zařadíme-li se daný nástroj příliš pozdě, žákům může připadat příliš dětský a z toho důvodu nezajímavý. (Kalaš, 2017)

2.3.2 Základní principy výuky programování

Základní principy programovaní ve svých pracích či článcích popisuje několik autorů.

Desatero zásad výuky algoritmizace a programování podle Pittnera (Pittner, 2000)

- Učitel musí teoreticky zvládat algoritmizaci, praktické programování i prostředky k tomu určené
- Systematicky myslet by učitel měl učit všechny žáky, programování a algoritmizaci pouze zájemce
- Cíle musí být předem ujasněny a podle nich se volí metodika a nástroje
- Na žáky je potřeba nespěchat, musí být pro danou činnost dostatečně vyspělí
- Pozor na učení špatných zvyků, lepší než naučit špatně, je nenaučit vůbec
- Měly by být využity adekvátní prostředky vedoucí přímo k cíli
- Je nutné zdůrazňovat podstatné a odstínit to od nepodstatného
- Důležité pojmy by měli žáci pochopit už během výuky a není dobré je nechávat na samostudium
- Pokud je to nutné v rámci pochopení problému, není špatně “klesnout pod odbornou úroveň”
- Učitel musí mít vlastní programátorské zkušenosti

Pecinovský definoval zásady, které by měl učitel při výuce programování dodržovat následovně (Pecinovský, 2001)

- Žák by měl mít možnost co nejdříve tvořit programy, nejlépe informace vstřebá, pokud má možnost si danou věc vyzkoušet
- Používat pouze prvky jazyka, které již byly vyloženy
- Informace žákům dávkovat postupně a po malých soustech
- Dobrovolné příklady musí vyžadovat použití nových poznatků
- Příklady musí být zajímavé
- Žáci se musí naučit programy nejen vytvářet, ale i ladit
- Žáci by měli během výuky řešit i netriviální problémy

Kalaš v souvislosti s nástrojem ScratchMaths zmiňuje následující zásady (Kalaš, 2017)

- Používat principy 5E (Model 5E se věnuje badatelsky orientované výuce. Skládá se z 5 částí a v češtině by bylo možné jej označit 5Z: zapojení, zkoumání, zobecnění, zpracování a zhodnocení (Kuncová, 2019))
- Postupovat od přímé manipulace přes přímou akci k řízení programem, odizolovaného k složenému, od jednoho (příkazu, vstupu, postavy, kostýmu ...) ke dvěma, případně více
- Žák by měl rozumět každému bloku dříve, než jej použije
- Žáci nejvíce ocení konstrukty, které sami objeví
- Žáci si potřebují konstrukt zažít a vyzkoušet
- Všechny konstrukty je potřeba používat opakovaně a v různých souvislostech
- Žáci by měli být vedeni k tomu, aby skripty vysvětlovali a uvažovali o nich blok po bloku, i jako o celku.

2.4 Nástroje pro výuku programování

Nástrojů jejichž pomocí se žáci na základní škole mohou učit programovat existuje široká škála. Většina se snaží základní principy algoritmického myšlení zprostředkovat zábavnou formou. Obvykle formou hry. Forma hry má smysluplný výukový potenciál. Současně tato prostředí zprostředkovávají okamžitou nebo poměrně rychlou zpětnou vazbu, což je pro něj motivující k další činnosti.

Některé nástroje mají propracované vývojové prostředí do takové míry, že žák nemusí psát žádný programový kód a jen přesouvá příkazy reprezentované grafickou formou (například ikony, puzzle), konstrukce a události pomocí funkce uchopení a přesunutí příkazu na požadovanou pozici ve zdrojovém kódu. Jiné naopak vedou žáky k psaní jednoduchých příkazů. (Kozák, 2015)

Výše zmíněné přístup k tvorbě programů se v literatuře často nazývá blokové programování. Blokové programování patří mezi nejvyužívanější formy programování za pomoci nástrojů určených pro výuku. Je určené především ke snazšímu pochopení problematiky programování. Podporuje počáteční kroky ve výuce programování, algoritmické myšlení a obsahuje jednoduché prostředí pro vstup do programátorského světa. (Horváthová, Kněžník, 2019)

Nástroje pro výuku programování a algoritmického myšlení je možné rozdělit na následující kategorie:

- Robotické hračky
- Robotické stavebnice
- Výukové hry
- Dětské programovací jazyky

Vzhledem k široké nabídce programů, nástrojů a možností, je potřeba zvolit vhodný úměrně věku, schopnostem a dosavadním zkušenostem žáků. Pokud zvolíme špatně, je riziko, že dojde situaci, kdy žáci budou považovat příslušné prostředí za příliš dětské a triviální nebo naopak za příliš složité, což je může od samotného programování odrazovat. (Kalaš, 2017)

2.4.1 Robotické hračky a robotické stavebnice

Edukační robotika, je odvětvím robotiky úzce propojené s pedagogikou. Jedná se o silný a flexibilní vzdělávací nástroj s velkým motivačním faktorem, který umožňuje studentům prostřednictvím grafických nebo textových programovacích jazyků řídit a kontrolovat chování robotů. (Tocháček, Lapeš, 2012)

Robotické hračky a stavebnice cílí na širokou skupinu žáků a na trhu jich je k dispozici mnoho. Díky tomu, že se jedná o hračky jsou pro žáky zcela konkrétní, mohou si na ně sáhnou, mohou je držet v ruce. Žákem naprogramované příkazy realizuje přímo tato hračka.

Robotické programovatelné hračky představují nově rozvíjející se odvětví robotiky. Cílovou skupinou se stávají děti předškolního věku, žáci základních, ale i středních škol. Učitelům se tak naskýtá možnost zapojení tohoto motivačního prvku do edukace napříč všemi stupni vzdělání. Přinášejí do výuky možnost rozvoje algoritmického myšlení, ale také zvýšení motivace žáků k učení, rozvoj kreativity a samostatného myšlení, mimo to podporují i rozvoj pravo-levé orientace, orientace v síti nebo mapě. (Vaňková, Pítrová, 2018)

Existují roboti, kteří zaujmou a jejich ovládání zvládne i dítě v předškolním věku (například Bee-Bot) a proti tomu najdeme na trhu i roboty, kteří jsou určeni spíše pro středoškoláky (například Asuro). Pro použití na základní škole jsou vhodné hračky například Bee-Bot, Blue-Bot, Ozobot, Edison a další. Liší se vzhledem, možnostmi a složitostí ovládání a programování.

Robotické stavebnice se od robotických hraček liší možností vlastní konstrukce. Čímž je kromě algoritmického myšlení rozvíjí jemnou motoriku, kreativitu žáků při hledání alternativních řešení, jejich tvůrčí schopnosti, představivost i fantazii. (Kvapilová, 2016)

Z kategorie robotických stavebnic jsou velice známé stavebnice od firmy LEGO, pro žáky nižšího stupně Lego WeDo 2.0 a pro žáky vyššího stupně Lego Mindstorms. Ale existují i další druhy stavebnic jako například Tetrix, Makeblock nebo Merkur.

Pokud tyto hračky či stavebnice mají být využity ve výuce, je nutné je pořídit, což představuje poměrně velké pořizovací náklady v řádech několika tisíc korun. Cena jednoho Blue-Bota je přibližně 3000 Kč a sada stavebnice Lego WeDo 2.0 stojí přibližně 4500 Kč.

2.4.2 Předpřipravené hry

Jedná se o softwarový nástroj pro rozvoj algoritmického myšlení, jejich cílem je formou hraní hry žáky seznámit se základními principy programování. Kód je sestavován buď na principu skládání bloků nebo žák musí jednoduché příkazy psát. Pomoci příkazů, pohybujeme postavou, která má nějaký konkrétní úkol například dojít na určitý bod, projít bludiště, sebrat určité předměty a podobně.

Složitost úkolů, které musí žák řešit, se postupně zvyšuje. Žáci nepotřebují žádné vstupní znalosti programování. Z počátku programuje pouze sekvence příkazů typu „jdi vpřed“, „otoč se vpravo“, následně se učí úkol řešit efektivnějším způsobem za použití cyklů. Obtížnost se zvyšuje postupným procházením levelů hry, cílem hry však není dosáhnout co nejvyššího levelu, ale rozvíjet žákovo algoritmické myšlení.

Tyto hry bývají obvykle zdarma a dostupné on-line. Ovšem programování je omezeno pouze na vyřešení předpřipravené úlohy a příliš neumožňuje zapojit žákovu fantazii.

Do této kategorie patří například hry Run Marco, Lightbot, Galaxy Codr, CodeMonkey a mnohé další.

2.4.3 Dětské programovací jazyky

Dětské programovací jazyky jsou speciálně vytvořená vývojová prostředí usnadňující výuku programování. Žáci zde mají volnou možnost tvorby. Lze v nich snadno a rychle vytvořit vlastní program, což má pro žáky motivační efekt.

Historie dětských programovacích jazyků sahá až do roku 1967, kdy byl americkým matematikem, informatikem a pedagogem Seymourem Papertem představen programovací jazyk LOGO. Jedná se o želvu, která na základě příkazů kreslí tvary do písku. Tento programovací jazyk má celou řadu implementací a některé z nich jsou populární dodnes. (Musílek, 2012)

Tyto programovací jazyky podporují kreativitu a fantazii žáků. Nabízí zjednodušené prostředí pro výuku. Jejich hlavním záměrem je rozvoj algoritmického myšlení. Žáci se nemusí příliš zabývat způsobem zápisu a mohou se soustředit na jednotlivé části programu a jejich logickou návaznost. Zápis kódu je realizován buď na principu skládání bloků

nebo je zapisován zjednodušeně textově. Často jsou k dispozici v českém jazyce, což zejména mladším žákům usnadní práci. Velká část těchto dětských programovacích jazyků je k dispozici zdarma.

Jedním z nejznámějších soudobých nástrojů z této skupiny je programovací jazyk Scratch. Nejedná se však o jediný nástroj tohoto typu, existuje jich celá řada, například KODU, Stencyl, Scoolcode a mnoho dalších.

Scratch

Programovací jazyk Scratch byl navržen v roce 2005 na Massachusetts Institute of Technology, skupinou vedenou profesorem Mitchelem Resnickem. Jedná se o vizuální jazyk (kód je tvořen skládáním grafických elementů) určený pro výuku programování, vycházející z programovacího jazyka LOGO. (Musílek, 2013)

Výhody programovacího jazyka Scratch

Jedná se vizuální programovací jazyk a žáci kód skládají pomocí bloků jako puzzle. Tento systém skládání kódu má výhodu v tom, že nelze udělat chybu v syntaxi. Je k dispozici v českém jazyce (Scratch je dostupný ve více než 40 jazycích) a žáci se nemusí seznamovat s dalšími věcmi nad rámec základní myšlenky. Celé prostředí je přehledné.

Učí žáky práci s objekty, v každém programu mají postavy, kterým mohou programovat vlastnosti a chování. Také podporuje kreativitu žáků, protože nabízí, kromě přednastavených postav a pozadí, tvorbu vlastních. Pro tvorbu vlastních postav a pozadí disponuje jednoduchým grafickým editorem a umožňuje nahrát i zcela vlastní postavy a pozadí vytvořené v jiném programu.

Po stránce programování umožňuje veškeré základní koncepty jako jsou posloupnosti příkazů, větvení, cykly a proměnné. Díky komunitě vývojářů se program neustále vyvíjí a od verze 2.0 je možné využívat i komplikovanější prvky, jako jsou například funkce a procedury. (Krejsa, 2014)

Umožňuje široké možnosti toho, co, jak a kdy mohou postavy dělat, což u žáků vzbuzuje zájem a nadšení. Vzhledem k popularitě Scratche je k dispozici kompatibilita s jinými projekty určenými pro výuku programování, jako je například LEGO.

Scratch je zcela zdarma a lze jej používat v cloudové verzi, kdy jediné, co žák musí udělat je přijít na web (https://scratch.mit.edu/) a může začít programovat. Vytvořený projekt si může uložit do svého počítače. Scratch nabízí i vytvoření uživatelského účtu, kdy žák může své projekty ukládat a sdílet s ostatními. Ovšem počítá i s možností, že internetové připojení nemusí být k dispozici a nabízí možnost stažení vývojového prostředí přímo do počítače.

Žák prakticky okamžitě získává zpětnou vazbu, vidí to, co naprogramoval, což jej motivuje pracovat dál.

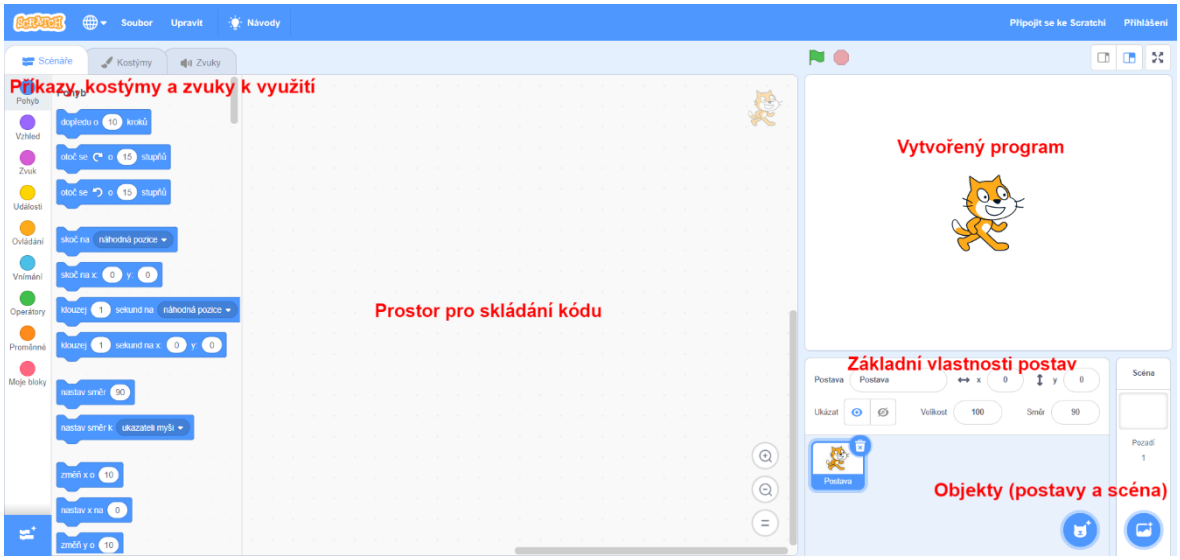
Nevýhody programovacího jazyka Scratch

Vzhledem k tomu, že se jedná o výukový programovací jazyk, je nutné zachovat adekvátní poměr mezi jednoduchostí i pro úplného začátečníka, který je obvykle ještě školákem, a množstvím možností, které jazyk poskytuje. Pokročilému programátorovi mohou některé možnosti chybět, ale začátečník by je stejně nejspíš nevyužil, i když by k dispozici byly. Při využití tohoto jazyka ve výuce je cílem, aby žák pochopil základní principy programování.

Prostředí Scratch

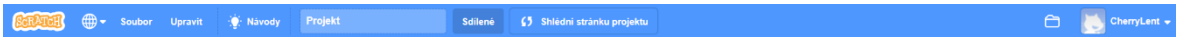
Jednou z hlavních předností programovacího jazyka Scratch 3.0 je prostředí, ve kterém je program sestavován. Toto prostředí je přehledné a intuitivní. Nabízí široké možnosti toho, co mohou uživatelé vytvořit.

Prostředí je rozděleno do čtyř částí. Vlevo se nachází příkazy, kostýmy a zvuky, které je možné využít, mezi jednotlivými záložkami je možné překlikávat. Uprostřed je prostor pro skládání kódu. Po pravé straně můžeme vidět a spustit vytvořený program, zároveň se zde nachází prostor pro používané objekty a vidíme zde jejich základní vlastnosti.



Obrázek 1- Prostředí Scratch

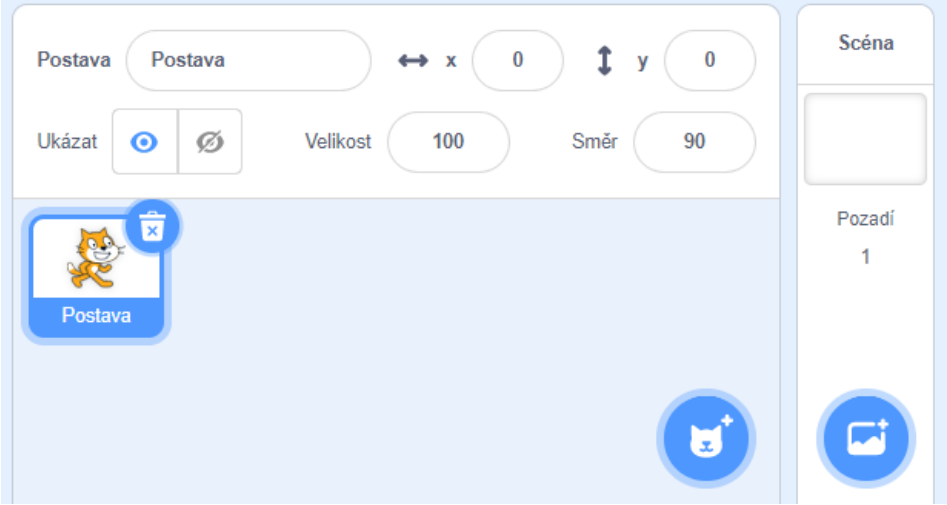
Prostředí zároveň disponuje horní lištou, která umožňuje volbu jazyka, práci se soubory, nápovědu a přihlášení. V přihlášeném režimu se nabízí více možností práce se souborem, krom toho, že jej můžeme načíst z počítače, či uložit do počítače, tak je možné soubor pojmenovat, sdílet, ukládat na náš účet a vytvořit jeho kopii. Ukládání projektů probíhá také jednou za čas automaticky, takže není potřeba mít strach, že bychom o celý projekt (v případě přihlášení) přišli.



Obrázek 2- Scratch přihlášený režim

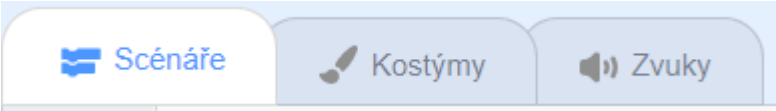
Objekty programovacího jazyka Scratch

Objekty ve Scratchi rozumíme postavy a scény (pozadí). Tyto objekty můžeme buď vybírat z nachystané knihovny, nebo je nahrát z počítače, či je vytvořit přímo ve Scratchi. Scény jsou prostředím, ve kterém se uživatelův program odehrává a postavy jsou věci, které mají určité vlastnosti, jako je pozice, velikost a podobně.



Obrázek 3- Scratch postavy a scény

Scéna i každá postava má vzhledy (kostýmy/pozadí), zvuky, a především vlastní kód (scénáře), který vzhledy a zvuky může využívat.



Obrázek 4 – Scratch možnosti postav a scény

Příkazy programovacího jazyka Scratch

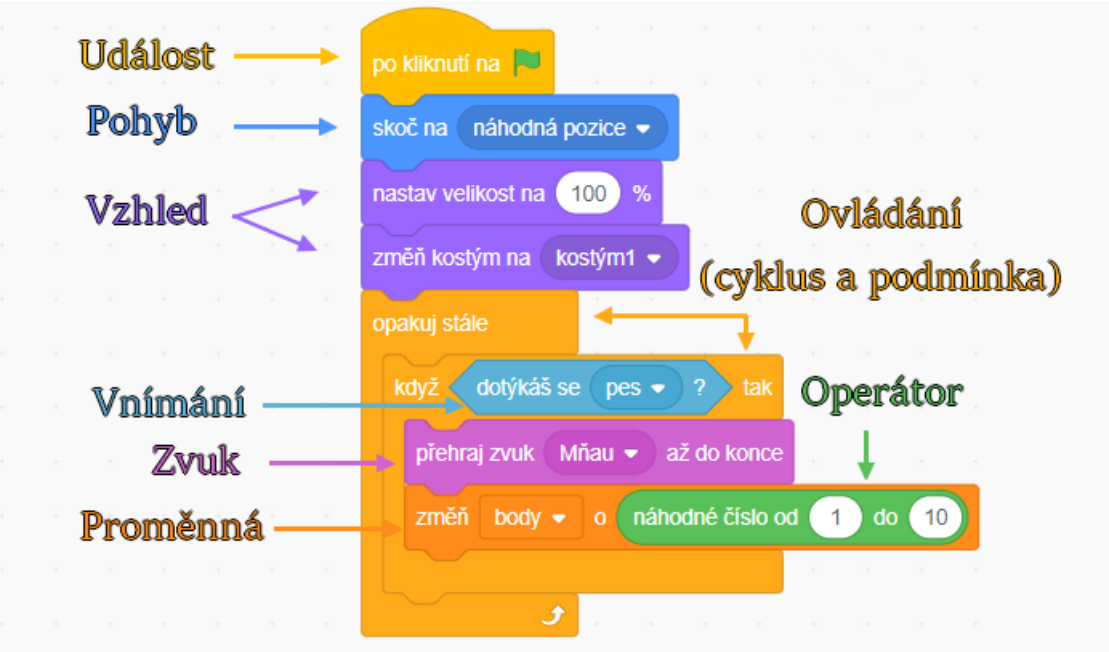
Příkazy, které uživatel může využít pro tvorbu scénářů jsou rozdělené a barevně odlišené podle činnosti, které se týkají a zároveň se odlišují tvarem z důvodu, že jejich skládání probíhá na principu puzzle, kdy do sebe jednotlivé dílky zapadají, což uživateli dává jistou nápovědu, co lze a co ne. Skládání kódu funguje na principu drag and drop, neboli chytni a pusť.

Kategorie příkazů:

- Pohyb (označený modrou barvou) obsahuje příkazy určené pro pozicování (hodnota osy X a Y) a hýbání postav. Příkazy pro pohyb mají k dispozici pouze postavy, scéna tuto sadu nemá.
- Vzhled (označený fialovou barvou) obsahuje příkazy pro úpravu či definování vzhledu postav, zároveň se zde nachází možnost mluvení postav, respektive toho, že nad postavou se objeví bublina s textem.
- Zvuk (označený růžovou barvou) obsahuje příkazy určené pro práci se zvukem.
- Události (označené žlutou barvou) obsahují vstupní body, tedy definují kdy se scénář pod nimi má začít vykonávat.
- Ovládání (označené oranžovou barvou) obsahuje konstrukce pro cykly a podmínky, do kterých se následně vkládají další argumenty, obvykle z kategorie vnímání nebo operátory.
- Vnímání (označené světle modrou barvou) obsahuje příkazy, které jsou schopné vyhodnotit, zda nastala nějaká situace. Většinu těchto příkazů nelze využít samostatně, ale vkládají se obvykle například do podmínek.
- Operátory (označené zelenou barvou) obsahují klasické matematické a logické operátory. Také je nelze zapojit do kódu samostatně, ale vkládají se na vhodné místo, jako součást jiných příkazů.
- Proměnné (označené tmavě oranžovou barvou) umožňují práci s proměnnou. Uživatel si může vytvořit libovolné množství proměnných. Scratch odlišuje proměnnou globální (= proměnná pro všechny postavy) a proměnnou lokální (= proměnná jen pro konkrétní postavu). Zároveň je možné tvořit seznamy.

- Moje bloky (označené tmavě růžovou barvou) umožňují tvorbu zcela vlastních funkcí.

Dále je možné přidat další sady příkazů promocí rozšíření, zde se nachází například sady pro další zvukové efekty, překladač, pero, která umožní malování, a několik dalších možností.



Obrázek 5 – Scratch ukázka kódu

Scoolcode

Jedná se o dětský programovací jazyk svým vzhledem a vlastnostmi velmi připomínající Scratch. Hlavním rozdílem je to, že není k dispozici zdarma, ale je vyhrazen pouze pro žáky navštěvující kroužek programování, v rámci kterého, je využíván. (https://www.logiscool.com/cz/scoolcode)

Některé funkcionality jsou přizpůsobeny právě potřebám daného kroužku, například obsahuje různé palety příkazů s ohledem na věk žáků (například žáci na nižším stupni základní školy využívají k nastavení směru postavy prvek hodin tedy postavu můžeme otáčet od 0 do 60 minut, zatímco žáci na vyšším stupni základní školy k rotaci využívají stupně, stejně jako Scratch.)

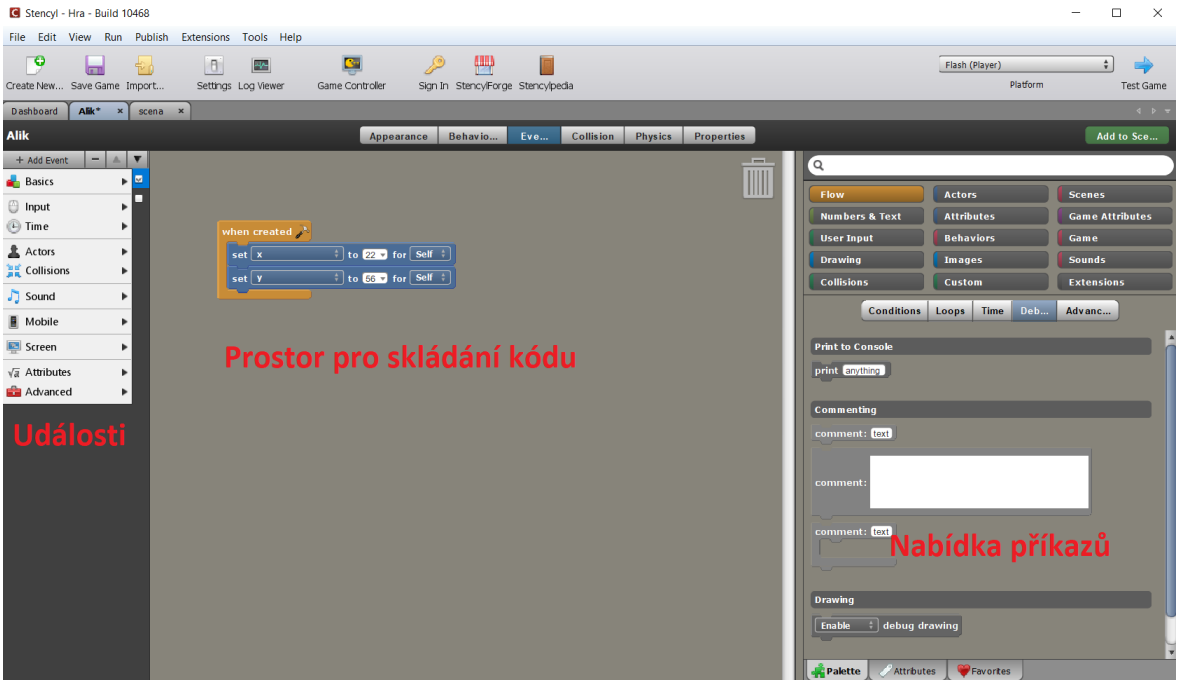
Zápis kódu umožňuje jak blokově, tak textově, opět to, jaké možnosti žák dostane k dispozici závisí na věkové skupině. Je k dispozici v českém jazyce a řešený cloudově, ale nelze pracovat bez přihlášení.

Obrázek 6 - Prostředí Scoolcode

Stencyl

Stencyl je vývojové prostředí, které umožňuje vytvářet aplikace pro počítače, web, telefony a tablety. Umožňuje tvorbu rozsáhlých her. (Červený, 2016)

Prostředí je svým vzhledem opět podobné předchozím zmíněným dětským programovacím jazykům, ale vyžaduje instalaci a není dostupné v českém jazyce (nabízí 14 jiných jazyků). Ovládání je poměrně intuitivní, nicméně vlastní tvorbě předchází několik nastavování. Prostředí neobsahuje předpřipravenou knihovnu postav a scén, je nutné nahrát vlastní. Tvorba programu probíhá na principu skládání bloků. V základní verzi je toto prostředí dostupné zdarma.



Obrázek 7 – Prostředí Stencyl

Kodu Game Lab

Kodu je dětský programovací jazyk vytvořený firmou Microsoft, je vytvořený speciálně pro tvorbu vizuálně působivých her pomocí konzole Xbox nebo počítače s operačním systémem Windows. Jeho prostředí je navržené pro potřeby mladších dětí. Žáci si v tomto programu staví vlastní 3D svět a celková grafika prostředí je blízká komerčním hrám, které žáci základních škol hrají. (Klimová, Horváthová, 2017).

Vývojové prostředí vyžaduje instalaci, je k dispozici v českém jazyce a je zdarma. U tohoto prostředí je kladen důraz na vizuálně působivé zpracování, poskytuje uživateli velké množství nápovědy ke každému úkonu, zároveň je jednoduché a navržené pro potřeby mladších dětí. Zápis kódu je realizován na blokovém principu výběru obrázkově znázorněných příkazů pomocí dlaždic. Tvorba her probíhá v několika krocích:

1. Tvorba prostředí hry (úpravy terénu)
2. Zavedení objektů a postav
3. Programování objektů

Obrázek 8 - Kodu tvorba prostředí hry

Obrázek 9 - Kodu přidávání objektů

Obrázek 10 - Kodu programování

2.4.4 Zhodnocení nástrojů

Každý nástroj má objektivní vlastnosti. Před výběrem programu, respektive nástroje je nutné jej posoudit na základě stanovených kritérií a až poté jej pořídit a využít ve výuce. (Vaníček, 2004)

Autoři posuzují výukové aplikace podle různých parametrů a kritérií. Vaníček uvádí následující kritéria: validita, čeština, obsahová správnost, věkově přiměřené didaktické metody, robustnost, motivační aspekty, možnost změny nastavení parametrů, nápověda, zpracovaná metodika použití, cena, styl výuky, a podobnost tomu, co uživatel zná. (Vaníček, 2004)

Brdička uvádí jako nejdůležitější prvky pro posuzování kvality programu první dojem, cenu, nápovědu, uživatelské rozhraní, jazyková lokalizace, přístupnost, diferencovaný přístup k žákům, a plnění stanovených cílů. (Brdička 1995)

Klement uvádí jako hlavní kritéria ovládání, terminologickou správnost a aktuálnost obsahu, interaktivitu, přítomnost procvičovacích prvků, přehlednost a snadnou ovladatelnost, nabídku více možných řešení, soulad s příslušným kurikulem, způsob využití ve výuce a uplatňování mezipředmětových vztahů. (Klement, 2005)

Dustin Le uvádí v souvislosti s hodnocením webových stránek pro podporu programování dětí jako podstatné parametry jednoduchost prostředí, vzhled programu a prostředí, efektivitu, zábavnost a cenu. (Le, 2015)

Výše uvedení autoři se shodnou na tom, že pro výběr vhodného nástroje je důležité prostředí nástroje nebo aplikace a také motivační efekt, tyto dvě kritéria v různých podobách uvádí všichni čtyři. Tři autoři se shodnou na tom, že podstatným kritériem je cena. Dále dva autoři uvádí jako podstatný parametr jazyk.

Zhodnocení výše uvedených kategorií nástrojů (robotické hračky, robotické stavebnice, předpřipravené hry a dětské programovací jazyky) dle následujících parametrů:

- Prostředí vývoje
- Motivační efekt a zpětná vazba
- Omezení pro žákovu vlastní tvorbu
- Cena
- Jazyk

Parametr vývojového prostředí vhodného pro výuku splňují všechny kategorie nástrojů, protože se obvykle jedná o tvorbu kódu na principu skládání bloků, či psaní jednoduchých příkazů, zároveň jsou tyto nástroje navrženy tak, aby jejich využití ve výuce bylo možné.

Motivační efekt a zpětnou vazbu poskytují také všechny uvedené kategorie nástrojů. U robotických hraček a stavebnic je výsledkem práce žáků pohybující se robot. U předpřipravených her a dětských programovacích jazyků, žáci získávají zpětnou vazbu po spuštění kódu na obrazovce počítače.

Rozdíl mezi nástroji je patrný v omezení pro žákovu tvorbu. Nejméně omezující jsou dětský programovací jazyky, v nich má žák mnoho možností vlastní tvorby a velkou paletu příkazů, které je možné využít. Nejvíce omezující jsou předpřipravené hry, kde žák pouze plní zadaný úkol – například dojít na určené místo. U robotických hraček a stavebnic je žákova vlastní tvorba omezena vzhledem a konstrukcí robota, u stavebnic pak množstvím dostupných kostiček.

Výrazným rozdílem je jejich cena, zatímco robotické hračky a stavebnice stojí za jeden kus v řádech tisíců korun a pro využití ve výuce je nutné mít kusů více, naopak předpřipravené hry a dětské programovací jazyky jsou často k dispozici zdarma, buď v plné verzi, nebo mají možnost vyzkoušení s omezenou licencí a případně je možné jejich zakoupení.

Jazyk je věcí konkrétního nástroje, také záleží na tom, jak probíhá skládání kódu, pokud se jedná pouze o symboly, tak jazyk použití ve výuce neovlivňuje, pokud však žák

musí pracovat s texty (bloky obsahující text nebo přímo psaní příkazů) jsou pro danou věkovou kategorii vhodnější nástroje s českou jazykovou mutací.

Z výše uvedených kritérií vyplývá, že pro komplexní rozvoj algoritmického myšlení jsou velmi vhodné dětské programovací jazyky, protože jsou přímo pro rozvoj algoritmického myšlení navrženy, umožňují žákovy téměř neomezenou vlastní tvorbu a mimo jiné díky tomu je práce s nimi pro žáky motivující. Jejich výhodou také je, že mnoho z nich je k dispozici zdarma.

V kapitole 2.4.3 jsou představeny čtyři dětské programovací jazyky Scratch, Scoolcode, Stencyl a Kodu Game Lab. V následující tabulce jsou stručně porovnány na základě těchto parametrů:

- Vývojové prostředí
- Omezení pro žákovu vlastní tvorbu
- Cena
- Jazyk

Nástroj	Vývojové prostředí	Omezení pro žákovu vlastní tvorbu	Cena	Jazyk
Scratch	Programování probíhá pomocí skládání bloků obsahující textové příkazy, kategorie příkazů jsou barevně odlišené. Program může obsahovat více objektů, které mohou mít vlastní funkcionalitu.	Možnosti tvorby jsou prakticky neomezené. Je k dispozici mnoho příkazů. Nástroj obsahuje předpřipravenou galerii postav a scén, umožňuje pomocí grafického editoru tvorbu vlastních nebo je možnost nahrát postavy či scény z počítače	Zdarma	Čeština
Scoolcode	Programování probíhá pomocí skládání bloků obsahující textové příkazy, kategorie příkazů jsou barevně odlišené. Program může obsahovat více objektů, které mohou mít vlastní funkcionalitu.	Možnosti tvorby jsou prakticky neomezené. Je k dispozici mnoho příkazů. Nástroj obsahuje předpřipravenou galerii postav a scén nebo je možnost nahrát postavy či scény z počítače	Pouze v rámci kroužku. 1500 Kč/měsíc	Čeština
Stencyl	Programování probíhá pomocí skládání bloků obsahující textové příkazy, kategorie příkazů jsou barevně odlišené. Program může obsahovat více objektů, které mohou mít vlastní funkcionalitu. V porovnání se Scratchem a Scoolcodem však není až tak intuitivní.	Možnosti tvorby jsou prakticky neomezené. Je k dispozici mnoho příkazů. Postavy a scény jsou nahrávány z počítače (neobsahuje vlastní galerii). Umožňuje tvorbu složitějších programů než Scrach a Scoolcode. Ve verzi PRO umožňuje export aplikace pro desktopové nebo mobilní využití.	Základní verze zdarma Verze PRO 4000 Kč/rok (sleva pro studenty/učitele/ školu)	Angličtina
Kodu Game Lab	Programování probíhá pomocí skládání bloků obsahující symbolické příkazy. Je zde kladen velký důraz na vizuálně působivé zpracování. Obsahuje velké množství nápovědy.	Možnosti tvorby omezené dostupnými postavami a příkazy.	Zdarma	Čeština

Tabulka 2 – Porovnání dětských programovacích jazyků

3 Praktická část

Cílem praktické části práce bylo vytvořit ucelenou sbírku úloh za pomoci vybraného výukového nástroje, určených pro věkovou skupinu žáků navštěvujících 4. – 6. třídu základní školy. Tyto úlohy následně ověřit se skupinou žáků z důvodu ověření náročnosti a nalezení případných problémů a komplikací při řešení úloh.

Všechny úlohy jsou dostupné online na webových stránkách: http://scratch.milichovska.cz/.

3.1 Zvolený výukový nástroj

Pro tvorbu úloh byl zvolen dětský programovací jazyk Scratch, který má mnoho výhod. Umožňuje široké možnosti pro vlastní tvorbu. Pro žáky je velice atraktivní, že získávají téměř okamžitou zpětnou vazbu ve formě vizuálně zajímavé a funkční hry, kterou mohou kdykoliv spustit a vyzkoušet. Zároveň prostředí, ve kterém k vývoji programu dochází je přehledné a intuitivní. Vzhledem k tomu, že skládání kódu probíhá formou skládání bloků stejně jako kostičky puzzle neboli na principu, že všechny dílky do sebe musí zapadat, pokud nezapadají puzzle nelze složit, přináší to žákům jistou nápovědu, co lze spojit a sestavit a co ne, také díky tomu není možné udělat chybu v syntaxi. Ale samozřejmě žáci musí zapojit i své logické a algoritmické myšlení, aby dokázali sestavit funkční kód, který dělá přesně to, co bylo zamýšleno.

Scratch má širokou komunitu, která se jeho vývojem zabývá a stále jej zdokonaluje. Také mě k jeho využití vedly praktické pohnutky. Asi hlavním pozitivem je v tomto směru to, že je zcela zdarma a díky cloudové verzi připravený ihned k použití. Není potřeba nic stahovat a instalovat, stačí přijít na webovou stránku (https://scratch.mit.edu/) a je možné začít okamžitě programovat. Druhým praktickým důvodem je dostupnost v českém jazyce, což je u zvolené věkové skupiny příjemné, protože žáci si příkazy dokáží přečíst a plnohodnotně jim samostatně porozumět.

S testovací skupinou žáků jsem zvolila možnost registrace a následného přihlašování, tedy každý žák měl svůj vlastní účet, na který se vždy na začátku lekce přihlásil a mohl tvořit. Účty jsme společně založili na první lekci kroužku, není to vůbec náročné, všichni žáci to zvládli bez sebemenších problémů. Pro případ, že by žáci nevlastnili email, jsem měla

nachystaný e-mail erární (není problém jeden e-mail využít k několika účtům), nicméně potřeba jej použít nenastala, protože všichni žáci uvedli e-mail vlastní. Tento krok není nezbytný. Je však praktické, že žáci mají všechny své naprogramované úlohy pohromadě na jednom místě, mohou se k nim vracet, případně je doma dodělat nebo vylepšit a v neposlední řadě ukázat rodičům co vytvořili, což jim obvykle činí velkou radost. Samozřejmě musíme vycházet z předpokladu, že si do příští lekce budou pamatovat své přístupové údaje.

3.2 Cílová skupina

Úlohy jsou cíleny pro žáky navštěvující 4. – 6. třídu základní školy, což odpovídá věkové kategorii 9 až 12 let. Tento věk je pro začátky výuky programování a rozvoji algoritmického myšlení ideální, protože takto staří žáci již mají základní schopnosti a dovednosti k tomu potřebné, jako je například schopnost číst, ovládají základní práci s počítačem, dokáží přemýšlet v souvislostech a logicky myslet.

Zde si dovolím odkázat se na Piagetovu teorii kognitivního vývoje, ve které autor definuje čtyři vývojová stádia dítěte. Výše zmíněná skupina žáků spadá do třetího vývojového stádia konkrétních operací, jenž je věkově vymezené od 7 do 12 let a důležitými procesy jsou zde logické myšlení a operování s abstraktními pojmy. Dítě je schopno pochopit identitu. Dokáže třídit objekty podle několika charakteristik a provádí experimenty s objekty nikoliv však zatím systematicky. Zároveň uvedu i čtvrté vývojové stádium, a to formálních operací, které počíná věkem 11 let a výše. Zde jsou důležitými procesy abstraktní a formálně logické operace. Dítě se už nemusí opírat jen o smyslovou skutečnost, při experimentování systematicky obměňuje proměnné a hledá pravidla. Dokáže se vyrovnat i se situacemi, se kterými se doposud nesetkalo. Operace jsou spojovány ve složitější struktury a dítě s nimi dokáže pracovat oběma směry. (Piaget, 2014)

Kromě toho, že je dítě v tomto věku na příslušnou aktivitu dostatečně psychicky vyspělé a disponuje potřebnými dovednostmi, tak má stále dětskou hravost a zvídavost. Tedy jsou pro něj aktivity s nástroji určenými pro výuku programování a rozvoj algoritmického myšlení zajímavé, atraktivní a zábavné. Zvolený výukový nástroj Scratch je pro danou věkovou skupinu vhodný.

Při výuce programování v dětském programovacím jazyce na základní škole jsem se již setkala s žáky od první až do deváté třídy. Pokud byli žáci příliš mladí, tedy děti navštěvující 1. – 3. třídu, často jsem se setkala s absencí základních dovedností, například že si neuměli přečíst potřebné informace a příkazy nebo neuměli ovládat počítač, a i když úlohy byly adekvátně obtížné, aby je žáci zvládli, tak nedostatečné potřebné dovednosti je v tom poměrně limitovaly a v některých případech zcela demotivovaly dále pracovat. Naopak u žáků starších, tedy 7. – 9. třída jsem se již několikrát setkala s tím, že jim aktivity ve výukovém programovacím prostředí přišly příliš jednoduché a dětské a práce je bavila jen chvíli.

Samozřejmě každý žák je individuální osobnost. Vždy záleží na individuálním vývoji žáka, jeho schopnostech, dosavadních zkušenostech a vlastní motivaci.

Pro realizaci úloh není potřeba, aby žáci měli předchozí zkušenost s programováním nebo s programovacím jazykem Scratch. Postupně se seznamují s prostředím, možnostmi Scratche a algoritmickými strukturami.

Testovací skupina

Vytvořené úlohy byly oběrovány na skupině žáků navštěvujících zájmový kroužek zaměřený na rozvoj programování a algoritmického myšlení. Kroužek je určený pro výše zmíněnou věkovou skupinu 4. – 6. třída základní školy.

Testovací skupina čítala celkem 8 žáků. Jistý vliv na jejich práci s úlohami může mít to, že si kroužek sami zvolili, tedy se v testovací skupině vyskytují žáci, kteří mají o tuto aktivitu zájem. U většiny aktivit měli všichni žáci velice podobné tempo, vše se snažili řešit samostatně, jen výjimečně požádali o radu a vždy minimálně polovina stíhala i takzvané bonusové úlohy.

Věkové rozložení žáků v testovací skupině bylo následující: 6. třída 3×, 5. třída 3×, 4. třída 2×. Na všech lekcích bylo přítomno vždy minimálně 6 žáků.

Žáci byli úplní začátečníci, čtyři žáci uvedli, že se již s nějakým prostředím pro výuku programování setkali (typicky se jednalo o předpřipravené hry typu Run Marco), ale pouze jeden žák zmínil, že se již setkal přímo se Scratchem. Nicméně jejich vstupní zkušenosti se zhruba po prvních dvou až třech úlohách zcela sjednotily.

3.3 Struktura úloh

Úlohy mají jednotnou strukturu a je na učiteli, kolik a jaké informace dá žákům k dispozici. Všechny úlohy mají charakter hry, kterou žák může ovládat a případně si ji zahrát.

Obtížnost úloh pozvolna graduje. Žáci se nejprve seznámí s výukovým nástrojem Scratch, jeho prostředím a se základní algoritmickou strukturou - posloupností příkazů, které program vykonává. Postupně se přidávají cykly a podmínky. V několika posledních úlohách se žáci seznámí s proměnnou a jejím využitím.

Příběh a náhledy

Každá úloha má k dispozici krátký motivační příběh doplněný náhledem, tento příběh popisuje, co se v dané hře děje a případně jak jej uživatel může ovládat. náhledy jsou určeny k tomu, aby vznikla jistá představa o tom, jak finální výsledek vypadá, aniž bychom ji museli spouštět.

Cíl hry, vstupní požadavky na žáka a nové dovednosti

Tato část úlohy je podstatná pro učitele, specifikuje cíl hry, také to kterým algoritmickým strukturám se daná úloha věnuje a jaké nástroje programovacího jazyka Scratch bude žák využívat.

Vstupní požadavky na žáka říkají, co by žáci již měli znát pro úspěšné a samostatné řešení úlohy. Úlohy na sebe postupně navazují, tedy se předpokládá, že to, s čím se seznámili v první úloze využijí i v těch následujících.

Nové dovednosti popisují, s čím se žák v úloze nově seznámí. Jejich schopnosti jsou postupně nabalovány a ve většině úloh se nachází něco nového, s čím se žák musí seznámit. Nicméně by žákům nemělo činit problém vymyslet to samostatně.

Vstupní požadavky a nové dovednosti jsou rozděleny na kategorie *Algoritmizace* a *Scratch*.

Zadání

Každá úloha obsahuje zadání ve dvou zněních. První odpovídá pracovnímu postupu, jak úlohu vyřešit a druhé je jen nástin toho, co se očekává, že bude program umět. Jako nejvíce obecné zadání by bylo možné využít i pouze příběh hry.

Zadání pro žáky v krocích odpovídá pracovnímu postupu, nicméně žákům není nadiktováno vše do posledního detailu. Mnoho věcí musí vymyslet sami, v zadání je v jednotlivých krocích řečeno, co by se mělo udělat a v jakém pořadí, aby žák dosáhl funkčního programu. Jedná se o doporučený postup, v jakém pořadí mají žáci jednotlivé kroky realizovat, z toho důvodu je postup očíslovaný. Nové nebo složitější záležitosti jsou v postupu zmíněny ve formě nápovědy, věci, které by mohly žáka překvapit, nebo se domnívám, že by bylo vhodné na to žáky upozornit, zmiňuji v postupu jako doporučení.

Zadání pro žáky ve formě úkolu je mnohem méně konkrétní, je zde v bodech popsáno, co se od žáka a výsledného programu očekává. Postup, jaký žák zvolí je čistě na něm. I zde se může vyskytnout nápověda či doporučení, ale v mnohem menší míře.

Obě zadání jsou doplněna o bonusový úkol nebo bonusové úkoly, ty jsou určeny pro rychlejší žáky, aby se poté, co vytvoří program podle základního zadání nenudili.

Řešení a metodické poznámky

Tato část je určena výhradně pro učitele, jedná se o přesný postup řešení každé úlohy, doplněný náhledy programu, případnými doporučeními či vysvětlivkami, proč je daná věc vytvořena konkrétním způsobem.

I v této části se předpokládá, že věci, které byly využity v úlohách předchozích už žák i učitel znají, tedy není třeba je v každé úloze znovu vysvětlovat.

Možné problémy a komplikace

Tato část vznikla na základě ověřování úloh skupinou žáků a je výčtem problémových situací, které mohou v žákově řešení nastat. Cílem je usnadnit učiteli nalézt problém a pomoci žákovi s jeho vyřešením. Jedná se o situace, které se vyskytly při testování těchto konkrétních úloh skupinou žáků navštěvujících zájmový kroužek. případně jsou to situace, se kterými jsem se již při výuce někdy setkala a je vhodné na ně upozornit.

3.4 Zapojení úloh do výuky

Učitel má několik možností, jak tuto sadu úloh zapojit do výuky. Záleží zejména na žácích, jejich věku, zkušenostech, schopnostech a nadšení, zároveň však mají vliv i vnější faktory, jako počet žáků ve třídě a čas, který aktivitě může být věnován. Úlohy jsou vymyšlené tak, aby je zvládli žáci řešit i samostatně. Bez bonusových úkolů a agendy kolem (zapnutí počítače, přihlášení a tak dále) by měl průměrně stačit čas 60 minut, u některých úloh je to méně u jiných více, záleží hlavně na konkrétní skupině žáků

Testovací skupina žáků řešila úkoly samostatně, občas však někteří potřebovali poradit nebo dopomoci, tento stav byl však spíše výjimečný. Časová dotace pro jednu výukovou lekci kroužku činí 90 minut, přičemž v polovině se vždy udělá 5–10 minut přestávky a agenda na začátku také zhruba 5 minut zabere, tedy čistý čas pro práci žáků je 75 minut, nic jim však nebrání pracovat i během přestávky. Jak jsem zmínila výše, za tuto dobu více než polovina žáků zvládla vypracovat zadání včetně bonusových úkolů.

Je potřeba myslet na to, že aby výuka byla efektivní, žáci by si měli na řešení přicházet sami, pokud jen opíší kód z tabule, tak to nebude mít pro žáka téměř žádný přínos.

Za nejvhodnější způsob považuji nechat žáky pracovat samostatně s tím, že učitel je jim po celou dobu k dispozici, aby zodpovídal jejich dotazy, případně jim poradil, nikoliv však napsal kód za ně. Já osobně po žácích vyžaduji, že pokud se chtějí na něco zeptat, musí dotaz promyslet a formulovat konkrétně.

Zároveň jsem ve sbírce úloh zvolila právě dvě varianty zadání, aby bylo možné co nejvíce přizpůsobit práci individualitě žáka. Domnívám se, že pro žáka ve čtvrté třídě je, obzvláště ze začátku, vhodnější využít zadání ve formě pracovního postupu, žák navštěvující šestou třídu, nebo žák, který už má s programováním ve Scratchi zkušenosti, může více ocenit zadání formou úkolu, kde má mnohem větší prostor pro vlastní uvažování a kreativitu.

Mimo jiné je na zvážení, zda žákům ukázat jen textové zadání nebo i náhledy či před nimi krátce zahrát výslednou hru. Já využívám možnost promítnout finální produkt a popsat jej krátkým příběhem, poté poskytnout žákům textové zadání (ve verzi dle mého uvážení). Žáci jej dostávají v tištěné podobě. Tento přístup je namotivuje a zároveň si lépe dovedou

představit co se po nich vlastně chce, nicméně může to mít i jistou nevýhodu v tom, že je mírně potlačena jejich vlastní kreativita.

Nevylučuji však ani možnost využít úlohy jako doprovodný materiál k výuce, kdy učitel pracuje s žáky společně, v tomto směru, ale považuji za důležité nechat žáky vytvořit samostatně věci, které již znají a společně pracovat jen na věcech nových. Nicméně ani v tomto případě by výuka neměla sklouznout pouze k opisování kódu z tabule, je potřeba se třídy ptát, aby na řešení přicházeli v podstatě sami a jen se pak společně realizovalo. Zde může nastat problém, že tempo nebude vyhovující všem žákům, pomalejší budou tápat a rychlejší se budou nudit.

3.5 Přehled úloh

Přehled úloh slouží ke snazší orientaci v tom, jaké koncepty jsou v dané úloze využity. Kategorie přehledu vychází z rozdělení kategorií příkazů ve Scratchi. Stejné koncepty jsou v každém sloupci barevně označeny pro lepší orientaci.

Úkol	Pohyb	Vzhled	Zvuk	Události	Ovládání	Vnímání	Operátory	Proměnné
Všudypřítomná kočička	skok na náhodnou pozici	velikost mluvící bublina	x	klávesnice start	x	x	x	x
Denní a noční život	pomocí šipek	změna kostýmu změna pozadí	x	klávesnice start změna pozadí	x	x	x	x
Ples	pomocí šipek pozicování	x	ano	klávesnice start kliknutí	x	x	x	x
Módní salón	pozicování	velikost změna kostýmu	x	klávesnice start kliknutí	x	x	x	x
Upovídaná čarodějnice	pomocí šipek pozicování	velikost mluvící bublina	ano	klávesnice start	nekonečný cyklus podmínka když	dotyk	x	x
Žralok	pomocí šipek klouzání	velikost změna kostýmu skrývání a ukazování postav	x	klávesnice start	nekonečný cyklus podmínka když čekání po určitý čas	dotyk	x	x
Poslušný pejsek	pomocí šipek pozicování pomocí opakování	velikost změna kostýmu mluvící bublina skrývání a ukazování postav	x	klávesnice start	nekonečný cyklus cyklus s daným počtem opakování podmínka když podmínka když/tak jinak	dotyk	x	x
Predátor a kořist	pomocí myši klouzání	velikost změna kostýmu změna pozadí	x	start změna pozadí	nekonečný cyklus podmínka když	dotyk	x	x

Opice	pomocí myši skok na náhodnou pozici	velikost skrývání a ukazování postav	x	start	nekonečný cyklus podmínka když	dotyk	x	ano
Balóny	samovolný náhodným směrem	velikost	x	start kliknutí	nekonečný cyklus	x	náhodné číslo	ano
Vesmírná mise	pomocí šipek samovolný náhodným směrem skok na náhodnou pozici	velikost mluvící bublina změna pozadí skrývání a ukazování postav	x	klávesnice start změna pozadí	nekonečný cyklus podmínka když čekání s danou podmínkou cyklus s danou podmínkou	dotyk	náhodné číslo porovnání hodnot	ano
Dostihy	pozicování pomocí opakování	velikost změna kostýmu mluvící bublina	x	klávesnice start	nekonečný cyklus cyklus s daným počtem opakování podmínka když čekání po určitý čas čekání s danou podmínkou	dotyk	náhodné číslo porovnání hodnot	ano

Tabulka 3 – Přehled úloh

3.6 Úlohy

3.6.1 Úkol 1: Všudypřítomná kočička

Příběh hry

Seznamte se s kočičkou v programu Scratch! Nejprve se po levé straně podívejte na příkazy a pročtěte si je. Už asi tušíte, že všechny tyto věci může vaše postavička dělat, stačí je jen ve vhodném pořadí poskládat do prostoru pro program. Tak pojďme začít. Naučíme něco kočičku!

Obrázek 11 - Všudypřítomná kočička náhled 1

Obrázek 12 - Všudypřítomná kočička náhled 2

Cíl hry

Žák se seznámí s programovacím jazykem Scratch a naučí se používat základní algoritmickou strukturu - posloupnost příkazů.

Vstupní požadavky na žáka

- Žák se pomocí této hry seznamuje s prostředím Scratch, není třeba žádná předchozí zkušenost s tímto prostředím.
- Poznámka: V tomto úkolu jsou v rámci zadání v krocích použité barvy odpovídající barvám příkazů ve Scratchi, aby to žákům co nejvíce usnadnilo orientaci.

Nové dovednosti

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele
- **Scratch:** seznámení s programem, události klávesnice, událost po startu, náhodná pozice, nastavení směru, způsob otáčení, velikost postavy, mluvící bublina

Zadání pro žáky v krocích

1. Využijeme výchozí postavu zrzavé kočičky.
2. Kočička se umí přesouvat na náhodné místo po stisknutí mezerníku.
3. Kočička se umí otáčet vlevo a vpravo na stisk šipky doleva a doprava.

   Nápověda: musíme nastavit směr, kterým chceme, aby se otočila.

   Doporučení: nechceme mít kočku hlavou dolů, pomůže nám k tomu nastavit vhodný způsob otáčení, který jí nadefinujeme po stisknutí zelené vlaječky, tedy tlačítka Start.
4. Kočička umí měnit velikost: na šipku nahoru se zvětšuje a na šipku dolů se zmenšuje.
5. Není to však obyčejná kočička, a dokonce umí i mluvit, na stisk klávesy a nám říká po nějakou dobu “Ahoj”.
6. Nauč kočičku, aby po spuštění hry (Nápověda: zelená vlaječka) obnovila velikost na 100 %.

7. Bonusové úkoly:
    a. Na jiné klávesy může kočička říkat jiné věci.
    b. Prozkoumej, jak se liší myšlenka od bubliny a co se stane, když zvolíš variantu příkazu bez počtu sekund.
    c. Dokážeš ke své hře přidat pozadí?
    d. Občas se nám kočka schová půlkou těla za okraj. Dokážeš to vyřešit?

Zadání pro žáky ve formě úkolu

- Kočička volí svou pozici náhodně.
- Kočička se umí na šipku doleva a doprava otáčet vlevo a vpravo. Také zařídíme, aby nebyla vzhůru nohama.
- Kočička umí měnit velikost, šipkou nahoru a dolů. Zároveň po spuštění hry svou velikost obnoví.
- Dokonce je to i mluvící kočička, po stisku klávesy a vytvoří na několik sekund bublinku se slovem “Ahoj”.
- Bonusové úkoly:
    - Na jiné klávesy může kočička říkat jiné věci.
    - Prozkoumej, jak se liší myšlenka od bubliny a co se stane, když zvolíš variantu příkazu bez počtu sekund.
    - Dokážeš ke své hře přidat pozadí?
    - Občas se nám kočka schová půlkou těla za okraj. Dokážeš to vyřešit?

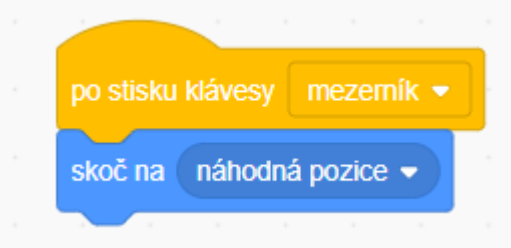
Program

https://scratch.mit.edu/projects/325905218/

Řešení a metodické poznámky

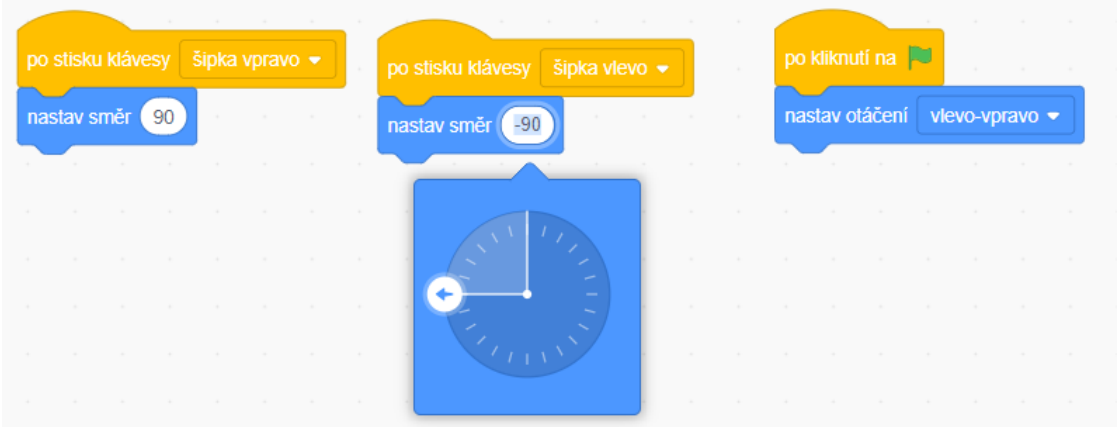
1. Postava kočičky je již předem vložená, můžeme rovnou programovat. Nejprve však žáky necháme samostatně prozkoumat příkazy po levé straně, aby se lépe zorientovali a hledání příkazů pro ně bylo snazší. Případně je možné příkazy probrat společně.

2. Po stisku klávesy mezera skočí kočička na náhodnou pozici.



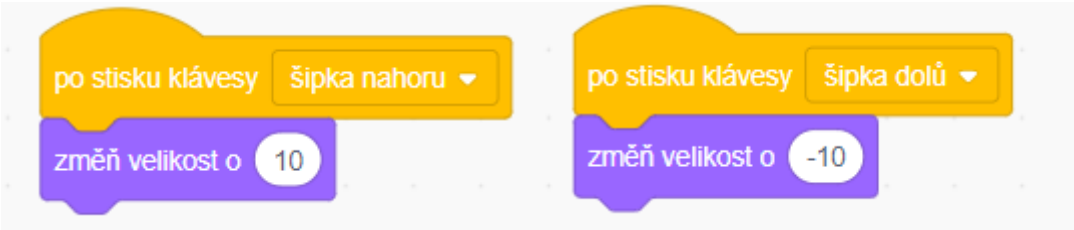
Obrázek 13 - Náhodná pozice

3. Po stisknutí klávesy doprava se nastaví směr vpravo (90) a doleva vlevo (-90). Přesná čísla není nutné znát, program má k dispozici vizuální kolečko nastavení směru, které se nám automaticky ukáže při přetažení příkazu a kliknutí do prostoru s číslem pro směr. Abychom neměli kočičku hlavou dolů, je potřeba *nastavit způsob otáčení "vlevo-vpravo"*, vhodné je tento příkaz umístit za událost Po startu hry (*Po kliknutí na zelenou vlaječku*), ale není zcela špatně jej umístit ani za jednotlivé otáčecí příkazy.



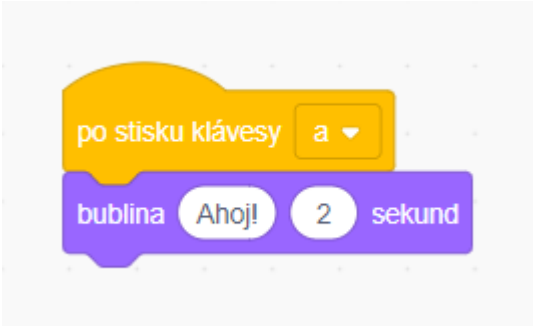
Obrázek 14 - Nastavení směru

4. Změna velikosti pomocí šipek nahoru (zvětšení + 10) a dolů (zmenšení -10).



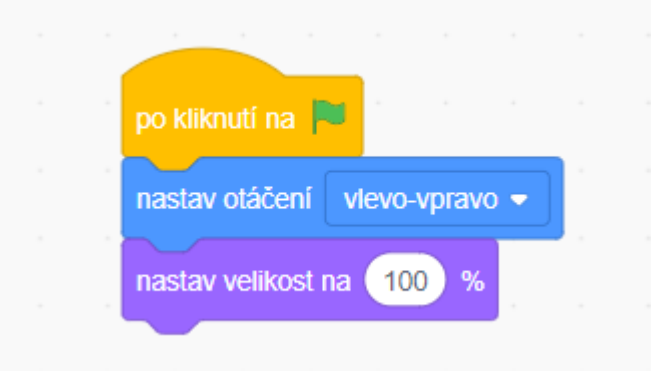
Obrázek 15 - Změna velikosti postavy (zvětšení, zmenšení)

5. Mluvení kočičky *po stisku klávesy A*, chceme, aby říkala "Ahoj" jen určitý počet sekund (Využijeme příkaz *bublina s omezeným počtem sekund*), když bychom zvolili verzi bez času, říkala by "Ahoj" už pořád.



Obrázek 16 - Mluvení postavy (bublina)

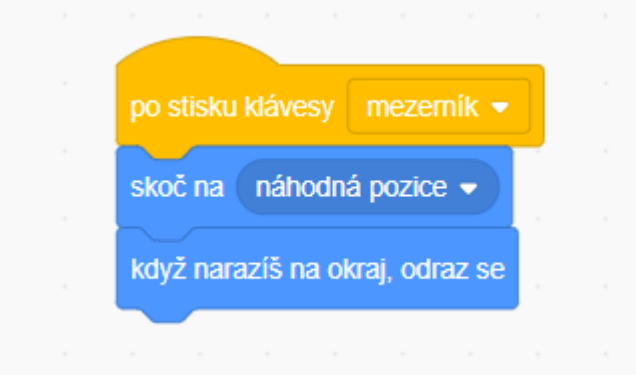
6. Po startu (*Po kliknutí na zelenou vlaječku*) kočička nastaví svou velikost na 100 %.



Obrázek 17 - Obnova velikosti

7. Bonusové úkoly:
    a. Jiná slova na jiné klávesy je totéž, jako “Ahoj” po stisku klávesy A.
    b. Myšlenka a bublina je malé cvičení pro žáky, aby se lépe seznámili s tím, co od kterého příkazu přesně získají. Princip naprogramování je však stejný jako slova “Ahoj” po stisku klávesy A.
    c. Změna pozadí je detailně řešena v úkolu 2.
    d. Schovávání kočky za okraj vyřešíme přidáním příkazu *když narazíš na okraj odraz se* k již existujícímu *po stisku klávesy mezera*.

    (Pozor, v tomto případě je vhodné mít nastavený způsob otáčení vlevo-vpravo, jinak se nám kočička bude nevhodně natáčet, ale to bychom měli mít vyřešené z předchozích kroků.)



Obrázek 18 - Odraz od okrajů

Možné problémy a komplikace

- Žákům se nedaří spojovat bloky - pokud mají přidané správné bloky, ale program nefunguje, zkontrolujte, zda jsou správně spojené.
- Kočička je po stisku klávesy šipka vlevo otočená hlavou vzhůru - chybí nastavit způsob otáčení vlevo-vpravo.

3.6.2 Úkol 2: Denní a noční život

Příběh hry

Motýl si vesele létá po lese. Uživatel se může rozhodnout, že už je čas, aby byla noc, v tu chvíli se změní pozadí na noční. Z motýla se stane netopýr a opět létá po lese, když si uživatel rozmyslí, že už je čas, aby byl den, změní se pozadí a z netopýra se opět stane motýl.

Obrázek 19 - Denní a noční život náhled 1 (den)

Obrázek 20 - Denní a noční život náhled 2 (noc)

Cíl hry

Žák se seznámí s výběrem postav, jejich kostýmy, pozadím a prací s nimi, naučí se systematicky pojmenovávat objekty a ovládat pohyb postavy pomocí šipek. Zároveň si upevní algoritmickou strukturu posloupnost příkazů.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele
- **Scratch:** události klávesnice, událost po startu

Nové dovednosti

- **Algoritmizace:** systematické pojmenovávání objektů
- **Scratch:** práce s postavami a jejich kostýmy (výběr, pojmenování, změna), práce s pozadím (výběr, pojmenování, změna), pohyb postavy pomocí šipek

Zadání pro žáky v krocích

1. Vyber denní pozadí a postavu motýla.

   Doporučení: pozadí přejmenuj na "Den" a podívej se, jaké má postava kostýmy - záložka Kostýmy, jeden můžete přejmenovat na "Motýl", přebytečné kostýmy postavy nebudou potřeba, tedy je možné smazat.

2. Zajisti pohyb motýla pomocí šipek.

   Nápověda: stejně jako člověk se pohybuje pomocí kroků, stejným způsobem se pohybují postavy v našem programu.

   Doporučení: aby nám motýl nelétal hlavou vzhůru použijeme způsob otáčení vlevo- vpravo.

3. Zařiď, aby se nám motýl neschovával za okraj herní plochy.

   Nápověda: potřebujeme, aby se motýl od okrajů odrážel.

4. Přidej noční pozadí.

   Doporučení: pozadí přejmenujte na "Noc".

5. K postavě přidej kostým netopýra.

   Doporučení: kostým přejmenuj na “Netopýr”.

6. Zařiď, aby po stisku klávesy **N** nastala **Noc** a po stisku klávesy **D** nastal **Den**
   Nápověda: přepnutí pozadí.

7. Ve dne je naše postava motýlem, v noci je netopýrem.

   Nápověda: změna kostýmu.

8. Doporučení: bylo by dobré, aby program věděl, jestli po startu hry má být den nebo noc a zároveň zda má být po startu hry postava motýlem nebo netopýrem.

9. Bonusové úkoly:

   a. Další pozadí a vhodné postavy: například město + člověk, voda + ryba, hrad + princezna a tak dále.

Zadání pro žáky ve formě úkolu

- Postava se pohybuje pomocí šipek, bylo by dobré, aby se neschovávala za okraje.
- Ve dne je naše postava motýlem, v noci netopýrem.

  Doporučení: je vhodné si kostýmy i pozadí adekvátně pojmenovat, usnadní to další práci.

- Noc nastane po stisku klávesy N, den po stisku klávesy D.
- Bonusové úkoly:

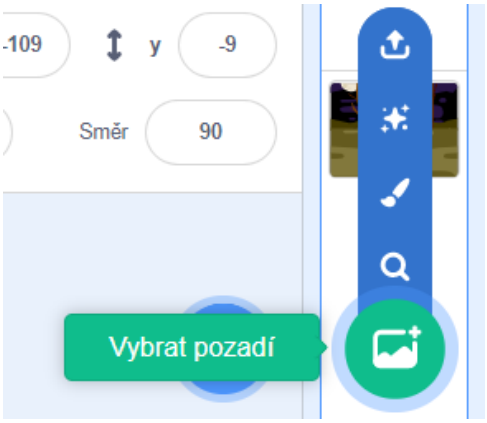
  - Další pozadí a vhodné postavy: například město + člověk, voda + ryba, hrad + princezna a tak dále.

Program

https://scratch.mit.edu/projects/323168481/

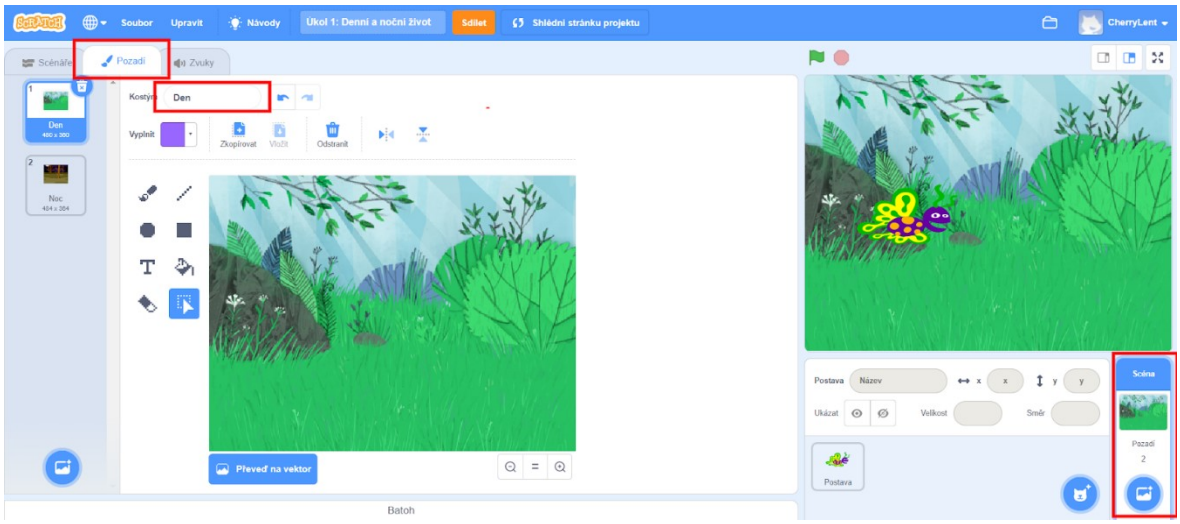
Řešení a metodické poznámky

1. Vybereme vhodné pozadí. V knihovně pozadí je k dispozici celé škála druhů pozadí, pro tento úkol je nejvhodnější denní příroda.



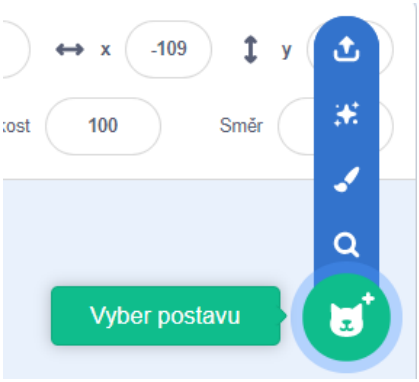
Obrázek 21 - Výběr pozadí

2. Z důvodu přehlednosti, a zejména s ohledem na další části úkolu, je toto pozadí vhodné přejmenovat na "Den". Klikneme vpravo dole na sekci pozadí a následně vlevo nahoře na záložku Pozadí. Kde vidíme všechna vybraná pozadí a můžeme s nimi pracovat.



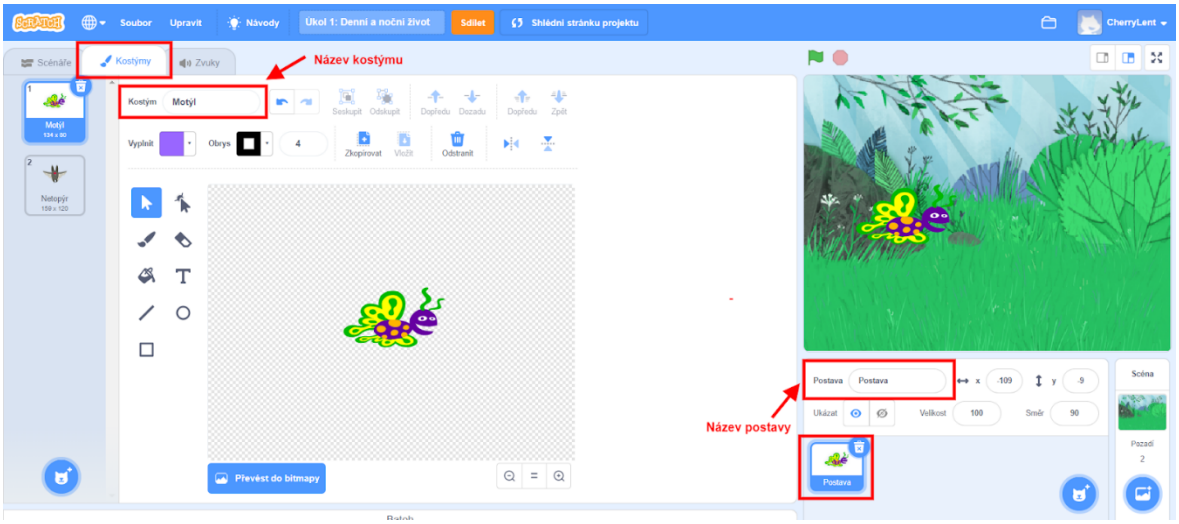
Obrázek 22 - Dostupná pozadí a jejich přejmenování

3. Vybereme postavu motýla. V knihovně postav nalezneme dva různé motýly a jednoho si vybereme.



Obrázek 23 - Výběr postavy

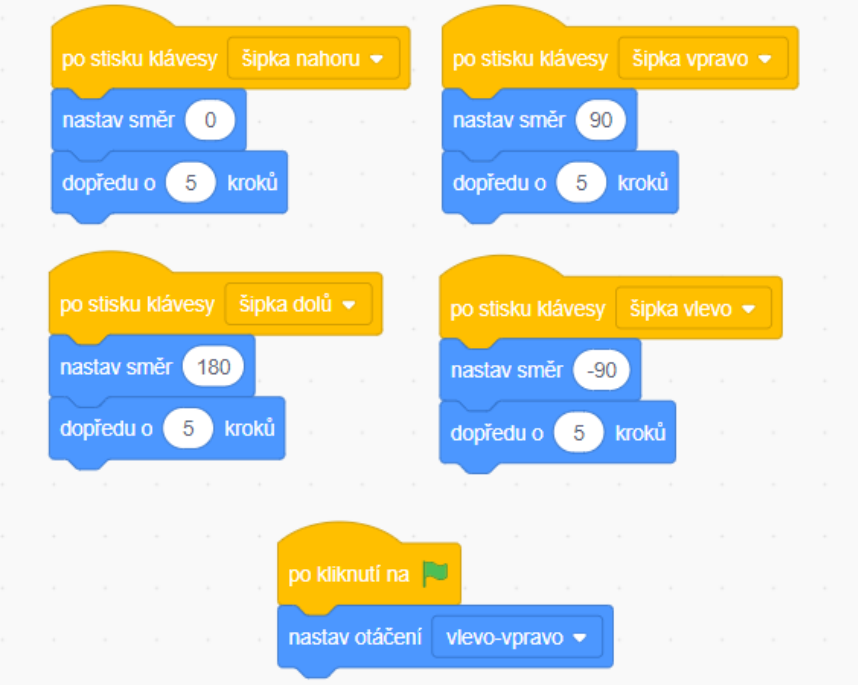
4. Opět je vhodné, postavu přejmenovat, nicméně u postav je to složitější, protože musíme rozlišovat postavu samotnou (= postavička, kterou programujeme) a její kostýmy (= to, jak naše postava aktuálně vypadá), jedna postava může mít libovolný počet kostýmů. V tuto chvíli musíme mít kliknuto na postavu, kterou chceme měnit a nad její ikonkou můžeme změnit jméno postavy. Dále pak vlevo nahoře vybereme záložku Kostýmy, kde můžeme spravovat všechny kostýmy, které daná postava má. Motýl může mít ve výchozím nastavení kostýmů více, nám bude stačit jen jeden motýl (a jeden netopýr, kterého přidáme později), přebytečné motýlí kostýmy je možné smazat.



Obrázek 24 - Dostupné kostýmy a přejmenování postavy i kostýmů

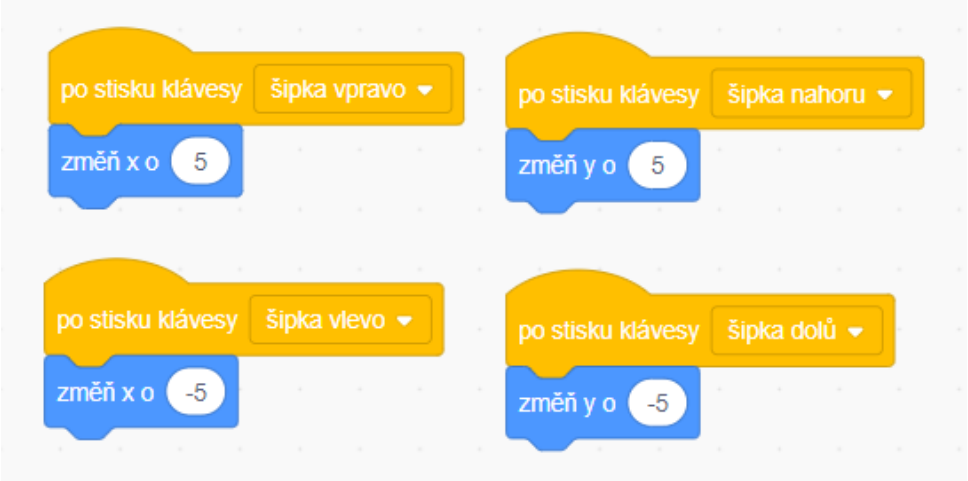
5. Pohyb postavy pomocí šipek: Vždy, když chceme postavu programovat, musíme ji mít vybranou (kliknutím na její ikonku vpravo dole) a také je potřeba být v záložce Scénáře, ve které najdeme všechny příkazy. Existují dvě varianty, jak zajistit pohyb postavy na šipky. Vždy však začínáme příkazem *po stisku klávesy*. (Poznámka: lepší a pro žáky v tuto chvíli pochopitelnější mi přijde způsob A, ale správné a funkční jsou oba dva.) V obou případech musíme zvolit adekvátní počet kroků/posun, ideální je 5-10.

    a. Nastavení směru a posunu o x počet kroků. Směr nastavíme podle toho, jakou šipku zrovna programujeme, Scratch nabízí k tomuto účelu posuvné kolečko, takže není třeba, aby žák znal stupně. Při tomto řešení je důležité nezapomenout zařadit příkaz *po kliknutí na start* (zelená vlaječka) *nastav otáčení vlevo-vpravo*, jinak by se postavička pohybovala doleva hlavou dolů.



Obrázek 25 - Pohyb postavy na šipky (nastavení směru a posun o daný počet kroků)

b. Nastavení posunu po ose X a Y (s problematikou pozicování a osy X a Y se žáci poprvé setkají v úkolu 3). V tomto případě nemusíme zařazovat otáčení vlevo-vpravo, protože postava se nikam neotáčí. Při posunu vlevo a vpravo měníme X a při posunu nahoru a dolů měníme Y. Pokud bychom chtěli, aby nám postava na šipku vlevo necouvala musíme u šipek vlevo a vpravo nastavit směr, stejně jako u varianty A, a pak tedy též musíme zařadit příkaz na nastavení způsobu otáčení.



Obrázek 26 - Pohyb postavy na šipky (změna osy X a Y)

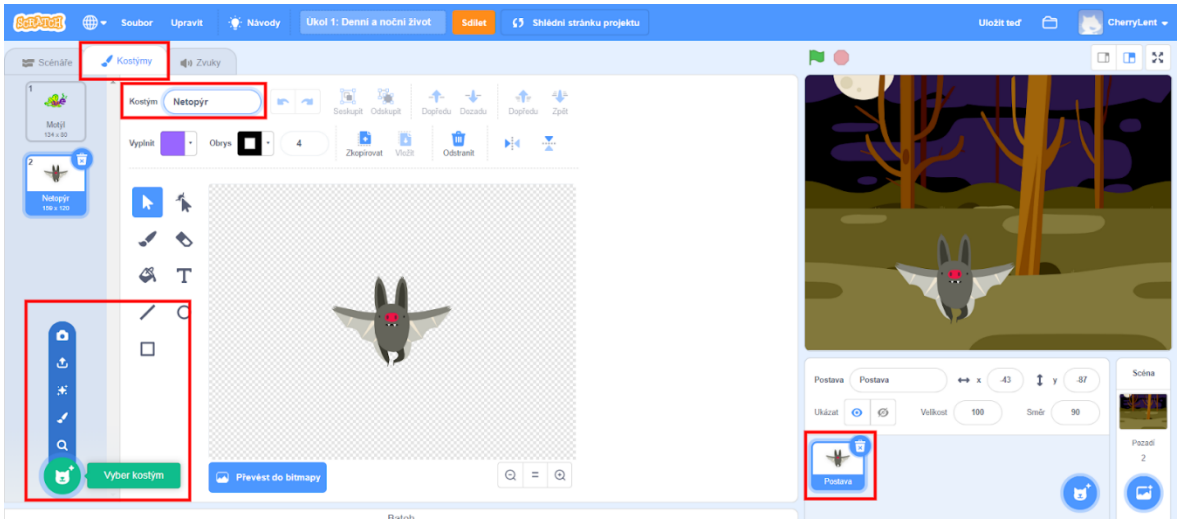
6. Můžeme si všimnout, že pokud s motýlem v tuto chvíli zajedeme k okraji, tak se nám schová. Tento problém se snadno vyřeší příkaz *když narazíš na okraj, odraz se*, který zařadíme ke každé události *po stisku klávesy*, tedy po každém posunu je zkontrolováno, zda se postava nedotýká okraje.

   Poznámka: Samozřejmě by bylo možné tento příkaz zařadit do nekonečného cyklu, který by se vyhodnocoval po celou dobu běhu programu, nemuseli bychom mít 4×, ten stejný příkaz, nicméně bereme v potaz to, že žáci v tuto chvíli cykly neznají.

7. Přidáme noční pozadí a vhodně jej přejmenujeme. Postup je stejný, jako při výběru prvního pozadí.

8. K postavě přidáme kostým netopýra. U postavy, v záložce kostýmy nalezneme ikonku Vyber kostým, nový kostým vhodně přejmenujeme.

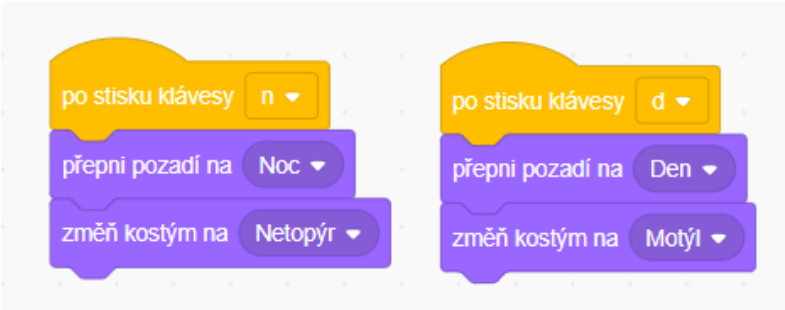
   Poznámka: Stejný účel pro tuto hru by splnilo i řešení, když bychom netopýra přidali jako druhou postavu a ve dne jej skryli a v noci ukázali, nicméně toto řešení by bylo značně neefektivní, vzhledem k tomu, že netopýr se chová úplně stejně jako motýl, tedy bychom tentýž kód museli psát dvakrát u obou postav.



Obrázek 27 - Přidání dalšího kostýmu

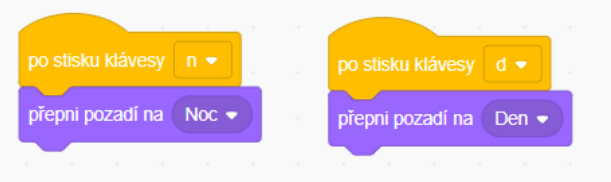
9. Po stisku klávesy **N** se přepne pozadí na **Noc** a po stisku klávesy **D** na **Den**. Zároveň ve dne má postava kostým motýla a v noci netopýra. Zde se naskýtají dvě řešení, kdy „způsob a“ je jednodušší (méně příkazů) a pro tento úkol zcela dostačující.

   a. Po stisku klávesy se změní zároveň pozadí i kostým postavy.

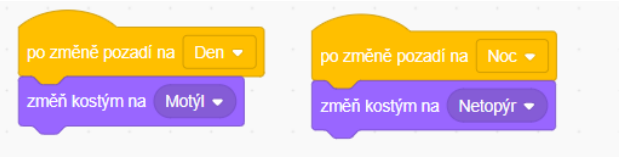


Obrázek 28 - Změna pozadí a kostýmu po stisku klávesy

b. Po stisku klávesy se změní pozadí a jako reakce na změnu pozadí se změní kostým.



Obrázek 29 - Změna pozadí po stisku klávesy



Obrázek 30 - Změna kostýmu po změně pozadí

10. Zároveň je dobré, abychom určili, zda po spuštění programu bude den nebo noc. To zařídíme přidáním příkazu *přepni pozadí na Den/Noc* za událost *po kliknutí na start* (tuto událost už bychom v programu měli mít, zejména pokud jsme pohyb řešili variantou A, pokud ji tam máme, nemusíme tentýž příkaz přidávat znovu). Stejně tak je dobré definovat, jaký kostým bude mít postava na začátku hry, tedy po startu zařadíme ještě příkaz *změň kostým na Motýl/Netopýr*.
11. Bonusový úkol:
    a. Přidání dalších pozadí a kostýmů - fantazii se meze nekladou, vše funguje obdobně jako netopýr s motýlem.

Možné problémy a komplikace

- Postava létá/chodí hlavou dolů - chybí nastavit způsob otáčení.
- Postava se neotáčí správným směrem - směr otáčení není nastaven správně nebo vůbec.
- Postava se pohybuje moc pomalu nebo moc rychle - je potřeba upravit počet kroků nebo posun na ose X/Y.
- Po startu hry máme noc a motýla/den a netopýra – není nastaveno kterým pozadím a kostýmem má hra začínat.

3.6.3 Úkol 3: Ples

Příběh hry

Ocitli jste se v sále, kde bude probíhat krásný ples. Vaším úkolem je zajistit skvělou hudbu a pozvat tanečníky.

Obrázek 31 - Ples náhled

Cíl hry

Žák se seznámí s pozicováním postav a prací se zvuky. Zjistí, že více objektů může reagovat na stejný vstup uživatele.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, velikost postavy, pohyb postavy pomocí šipek

Nové dovednosti

- **Algoritmizace:** více objektů může reagovat na stejný vstup uživatele
- **Scratch:** pozicování, událost po kliknutí, práce se zvukem

Zadání pro žáky v krocích

1. Vyber vhodnou scénu, kde se bude ples odehrávat.
2. Vyber dva hudební nástroje, v případě potřeby uprav jejich velikost a umísti je na vhodnou pozici.
    a. Nápověda: tuto pozici je důležité definovat příkazem, který obsahuje souřadnice, kde se postava nachází, aby její pozice byla vždy po spuštění hry stejná.
3. Vyber dvě postavy, které spolu budou tančit, v případě potřeby, uprav jejich velikost. Těmto postavám také definuj pozici, aby po spuštění hry vždy startovaly na stejném místě a stály vedle sebe.
4. Zajisti pohyb postav na šipky.
    a. Nápověda: postavy se pohybují společně úplně stejným způsobem - tančí spolu.
5. Zajisti, aby hudební nástroje vydávaly po kliknutí na ně zvuk.
6. Bonusové úkoly:
    a. Více hudebních nástrojů.
    b. Další pár tanečníků (začíná na jiné pozici než první pár, také se pohybují na šipky nebo na jiné klávesy například WASD).

Zadání pro žáky ve formě úkolu

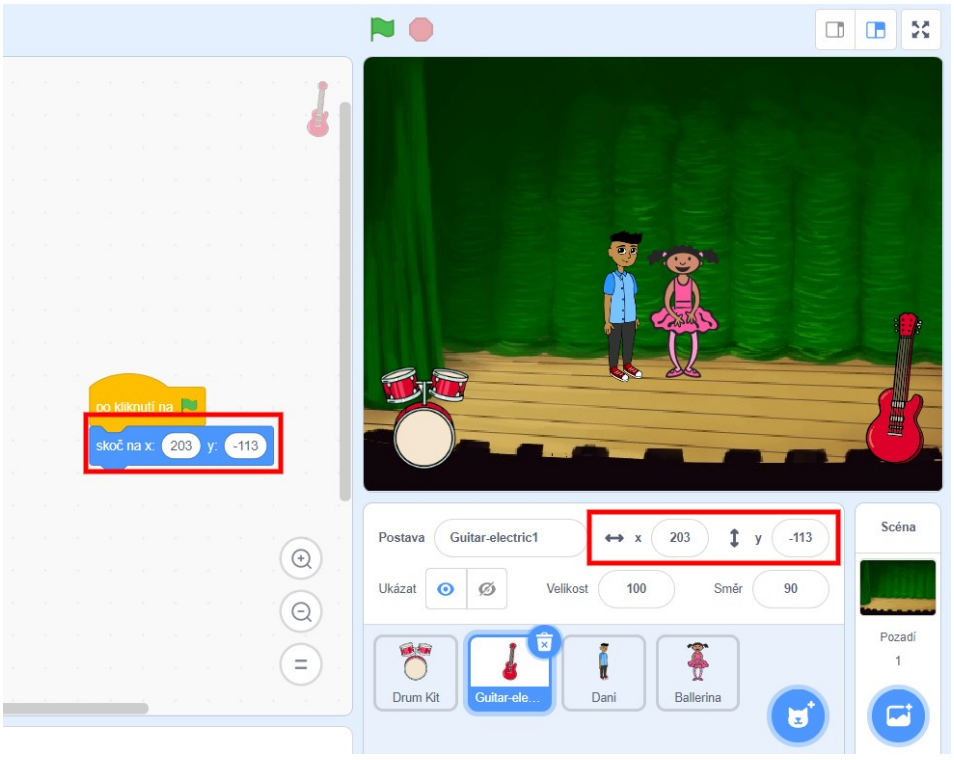
- Vytvoříme krásnou plesovou scénu s vhodným pozadím a dvěma hudebními nástroji, které se nachází na vhodných místech. (Pozici nástrojů je nutné definovat příkazem po spuštění hry). Tyto hudební nástroje po kliknutí na ně přehrají zvuk.
- Žádný ples by se neobešel bez tanečníků, tito tanečníci stojí vedle sebe a pohybují se na šipky. (Pohyb tanečníků je stejný - tančí spolu.)
- Bonusové úkoly:
    - Více hudebních nástrojů.
    - Další pár tanečníků (začíná na jiné pozici než první pár, také se pohybují na šipky nebo na jiné klávesy například WASD).

Program

https://scratch.mit.edu/projects/334369622/

Řešení a metodické poznámky:

1. Vybereme vhodné pozadí.
2. Vybereme dva libovolné hudební nástroje, pokud je to potřeba upravíme jim velikost pomocí příkazu *nastav velikost na x %*.
3. Pozicování: program Scratch umožňuje postavy po scéně libovolně přetahovat pomocí kurzoru myši, umístíme tedy přetažením postavu na požadované místo. Vedle názvu postavy si všimneme souřadnic, na kterých se aktuálně nachází. Je nutné tyto souřadnice definovat příkazem. (Pokud bychom to neudělali tak po dalším spuštění hry se postava nebude nacházet na námi zvoleném místě). Využijeme příkazy *po kliknutí na start (zelená vlaječka)* *skoč na pozici X Y*. Do X a Y zapíšeme souřadnice, na kterých chceme, aby postava začínala. (Tyto souřadnice se doplní automaticky podle aktuální polohy, pokud se tak nestane můžeme je vždy přepsat z příslušného místa pod náhledem výsledné hry.)



Obrázek 32 - Pozicování postav

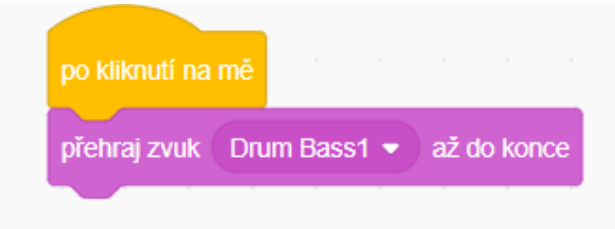
4. Dva vybrané hudební nástroje umístíme, výše popsaným způsobem, na vhodné pozice.

5. Vybereme dvě postavy osob, které spolu budou tančit, v případě potřeby upravíme velikost postav. (Je vhodné, aby byly podobně velké.) Tyto postavy napozicujeme vedle sebe na vybrané místo. Pozici opět musíme definovat (*skoč na pozici X Y*).

6. Obě postavy se pohybují pomocí šipek (pohyb na šipky je vysvětlen v úkolu 2). Postavy se pohybují zcela identicky

   Poznámky: počet kroků, o který se postavy posunou, směr a klávesy na které pohyb provedou musí být u obou postav stejný, nezapomeňte upravit *způsob otáčení* buď na "vlevo-vpravo" nebo "neotáčet", jinak by se tanečníci pohybovaly hlavou dolů.

7. Hudební nástroje po kliknutí na ně ozvučíme. Každý hudební nástroj má vlastní sadu zvuků, které můžeme využít. *Po kliknutí na nástroj* se *přehraje vybraný zvuk*.



Obrázek 33 - Přehrát zvuk

8. Bonusové úkoly:
   a. Další nástroje fungují obdobně, jako ty již existující.
   b. Další pár tanečníků funguje také obdobně jako pár, který již v programu máme. Důležité je nový pár nechat začínat na jiné pozici než první pár a zároveň dodržet počet kroků, tedy rychlost, kterou se oba tanečníci v páru pohybují. Pohyb postav je možný buď také na šipky nebo například na klávesy WASD.

Možné problémy a komplikace

- Postavy se pohybují hlavou dolů - není nastaven způsob otáčení vlevo-vpravo.
- Po startu hry jsou všechny postavy uprostřed (nebo na jiných nevhodných pozicích) - pozice není definována příkazem *skoč na pozici x y*.
- Tanečníci se nehýbou společně - zkontrolujeme příkazy pro pohyb. Může být špatně nastavený směr, počet kroků nebo klávesa, na kterou se má pohyb stát.
- Zvuk není slyšet - zkontrolujeme zda jsou zapnuté reproduktory, případně jestli je zvuk správně zařazen v kódu.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.

3.6.4 Úkol 4: Módní salón

Příběh hry

Vaše postava navštívila módní salón a musí si vybrat co na sebe oblékne. Pomůžete jí zajistit to, aby si mohla vše vyzkoušet?

Obrázek 34 - Módní salón náhled 1

Obrázek 35 - Módní salón náhled 2

Cíl hry

Žák si procvičí již získané dovednosti. (Použití posloupnosti příkazů, reakce na vstup uživatele, práci s postavami a jejich kostýmy, práci s velikostí postav a pozicování.)

Vstupní požadavky na žáka

- Algoritmizace: posloupnost příkazů, reakce na vstup uživatele
- Scratch: události klávesnice, událost po startu, událost po kliknutí, práce s postavami a pozadím, práce s kostýmy, velikost postavy, pozicování

Nové dovednosti

- Tato úloha slouží k procvičení dosud získaných znalostí a dovedností.

Zadání pro žáky v krocích

1. Vyber vhodnou scénu.
2. Přidej postavu, kterou budeme oblékat, a tuto postavu napozicuj na vhodné místo, případně uprav její velikost.

   Nápověda:

   a. Pro celý tento program se ti bude hodit ve výběru postav záložka Móda.

   b. Postava vhodná k oblékání má celkem tři kostýmy, vyber si takový, který se ti líbí - ostatní můžeš smazat.

   c. Pokud upravuješ velikost postavy, mysli na to, že bude potřeba stejným způsobem upravit velikost veškerého oblečení.
3. Přidej oblečení (alespoň 5 různých druhů), umísti jej na vhodné pozice pro výběr. Případně uprav jeho velikost.

   Doporučení: oblečení přidávej a programuj postupně.

4. Funkce oblečení:
    a. Po kliknutí: postava si jej oblékne na sebe.

       Nápověda: skočí na vhodnou pozici.

    b. Po stisku mezerníku: postava si jej sundá a oblečení se vrátí na původní pozici (funguje pro všechny kusy oblečení stejně).

    c. Po stisku písmene: oblečení mění kostýmy, pokud jich má víc. Například na klávesu K se mění kostýmy kalhotám, na klávesu S šatům a tak dále. (Písmena můžeš použít jakákoliv)
5. Bonusové úkoly:
    a. Více kusů oblečení.

    b. Oblékání postavy v závislosti na pozadí. (například: pozadí se změní na zimní a postava na sebe automaticky dostane zimní oblečení)

Zadání pro žáky ve formě úkolu

- Vytvoř módní salón s postavou, kterou je možné oblékat, a nabídkou minimálně pěti kusů oblečení.
- Po kliknutí na dané oblečení si jej postava oblékne.
- Některé kusy oblečení mají různé kostýmy, zajisti, aby se mohly měnit.
- Zajisti, aby se postava dokázala svléknout a oblečení se vrátilo na původní místo.
- Bonusové úkoly:
    - Více kusů oblečení.
    - Oblékání postavy v závislosti na pozadí. (například: pozadí se změní na zimní a postava na sebe automaticky dostane zimní oblečení)

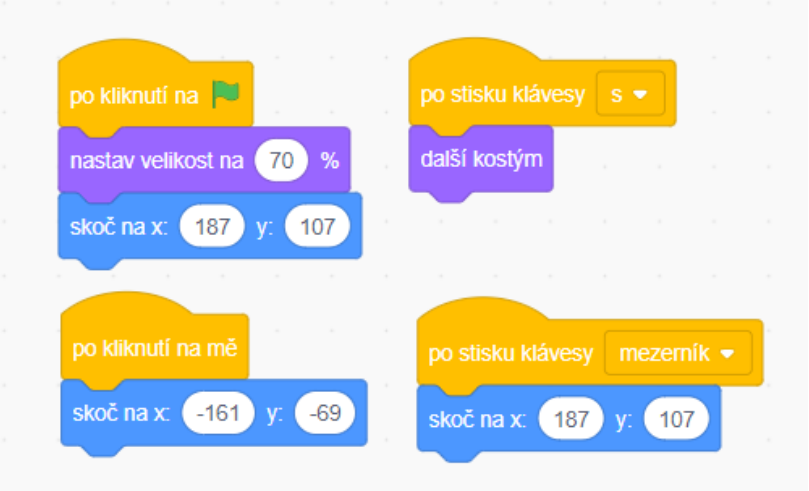
Program

https://scratch.mit.edu/projects/334704836/

Řešení a metodické poznámky

1. Vybereme vhodné pozadí.
2. Výběr postavy: Z kategorie Móda zvolíme postavu ve spodním prádle (protože je vhodná k oblékání), tato postava má tři kostýmy, můžeme zvolit ten, který se nám líbí a ostatní smazat. Postavu umístíme na vhodnou pozici příkazem *skoč na pozici X Y* a v případě potřeby upravíme její velikost příkazem *nastav velikost na %*. (Pozor: v případě úpravy velikosti postavy bude nutné upravit velikost i u všech kusů oblečení.)
3. Program oblečení: (doporučuji programovat jednotlivé kusy oblečení postupně, nejprve přidat šaty a k nim vytvořit potřebnou funkcionalitu, následně kalhoty a tak dále)
    a. Po spuštění hry (zelená vlaječka): oblečení se nachází na vhodné výchozí pozici (*skoč na pozici X Y*) a případně je mu upravena velikost (*nastav velikost na %*).
    b. Po kliknutí na mě: oblečení se přesune na postavu. Nejprve jej přetažením umístíme na cílovou pozici na postavě, čímž zjistíme potřebné souřadnice pro využití v příkazu *skoč na pozici X Y*. (Pozor: jako vhodný příkaz by se mohl zdát i "skoč na postavu", nicméně by to fungovalo jen u některých kusů oblečení - např. tričko, problém by nastal například u bot, které by se v tomto případě objevovaly na břiše postavy.)
    c. Po stisku mezerníku: oblečení se vrátí na pozici na které začínalo, tedy opět *skoč na pozici X Y* a zadané souřadnice budou stejné, jako po spuštění hry. (Tento příkaz bude fungovat u všech kusů oblečení stejně, tedy po stisku mezerníku se postava celá svlékne, samozřejmě můžeme i pro každý kus oblečení využít jinou klávesu, ale to by mohlo být pro žáky příliš zmatečné.)
    d. Po stisku písmene: některé kusy oblečení mají několik kostýmů, pomocí příkazu *další kostým* je můžeme měnit. Pro každý kus oblečení využijeme jiné písmeno z důvodu, že v případě, kdy bychom kostým měnili pro všechny kusy oblečení stejným způsobem (po stisku stejné klávesy), měnily

by se všechny kostýmy naráz, tedy bychom nedosáhli případných požadovaných kombinací. Písmena zvolíme dle vlastního uvážení. (Ve vzorovém projektu: S - šaty, K - kalhoty, B - boty, C - čepice)



Obrázek 36 - Program oblečení

4. Bonusové úkoly:
    a. Další kusy oblečení fungují úplně stejným způsobem.
    b. Přidáme další pozadí, které například *po stisku klávesy* měníme. K příslušnému kusu oblečení (například zimní čepici) doplníme příkaz *po změně pozadí na zima skoč na X Y*, kdy hodnota X a Y odpovídá umístění oblečení na postavě.

Možné problémy a komplikace

- Problémy s pozicováním - je potřeba překontrolovat, zda jsou pozice zadány správně.
- Oblečení postavě nesedí (je jí moc velké nebo malé) - velikost postavy byla upravena, ale velikost oblečení zůstala 100 %, nebo naopak. Je potřeba upravit velikost oblečení stejně jako ji má upravenou postava (% musí být stejná).
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.

3.6.5 Úkol 5: Upovídaná čarodějnice

Příběh hry

Vešli jste do pokoje mladé čarodějnice, která vlastní několik kouzelných zvířátek. Má je moc ráda, proto vždy, když k nim přijde, se s nimi vzájemně pozdraví.

Obrázek 37 - Upovídaná čarodějnice náhled 1

Obrázek 38 - Upovídaná čarodějnice náhled 2

Cíl hry

Žák se seznámí s nekonečným cyklem a neúplnou podmínkou.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, velikost postavy, pohyb postavy pomocí šipek, práce se zvukem, mluvící bublina, pozicování

Nové dovednosti

- **Algoritmizace:** nekonečný cyklus, neúplná podmínka
- **Scratch:** opakuj stále, podmínka když, vyhodnocení dotyku

Zadání pro žáky v krocích

1. Vyber vhodnou scénu.

   Doporučení: v celém tomto programu se nám bude hodit záložka Fantazie.

2. Přidej postavu čarodějnice, v případě potřeby uprav její velikost a zajisti její pohyb na šipky.

3. Čarodějnice vlastní minimálně tři zvířátka nebo jiné kamarády, ale bude lepší je přidávat postupně. Zvíře přidáme, v případě potřeby upravíme velikost a napozicujeme.

   Nápověda: pokud se dívá špatným směrem můžeme jej otočit.

   Doporučení: všechny postavy v tomto programu pojmenuj českými názvy.

4. Čarodějnice svá zvířátka pozdraví, když se jich dotkne.

   Nápověda: čarodějnice si neustále ověřuje, jestli se zvířete dotýká, když k doteku dojde, pozdraví jej.

5. Zvířátka zdraví svou čarodějnici. Zde prosím respektuj pokyn učitele, zda je možné využít zvuky, nebo budou zvířátka zdravit také bublinou, stejně jako čarodějnice.

6. Nápověda: pokud se vám nelíbí, že se čarodějnice schovává pod zvířátka, můžeš jí naprogramovat, že má jít do popředí.

7. Bonusové úkoly:
    a. Více zvířátek.
    b. Zvířátka mají určitě různé kostýmy, mohou je tedy celý program měnit a vypadat, že se pohybují na místě.

Zadání pro žáky ve formě úkolu

- Vytvoř čarodějný pokoj s minimálně třemi zvířátky nebo jinými postavami.
- Čarodějnice se po pokoji prochází a když se se zvířátkem setká, pozdraví jej a ono jí na oplátku odpoví. Zde prosím respektuj pokyn učitele, zda je možné využít zvuky, nebo budou zvířátka zdravit také bublinou, stejně jako čarodějnice.
- Bonus: Bonusové úkoly:
    - Více zvířátek.
    - Zvířátka mají určitě různé kostýmy, mohou je tedy celý program měnit a vypadat, že se pohybují na místě.

Program

https://scratch.mit.edu/projects/336083616/

Řešení a metodické poznámky

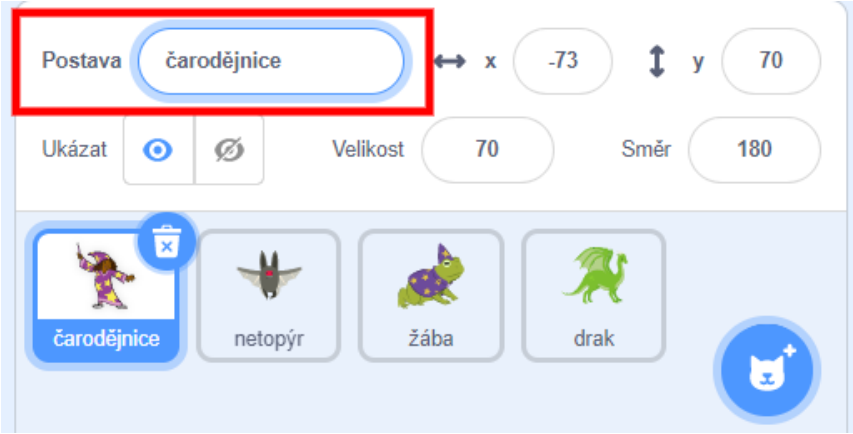
1. Vybereme vhodné pozadí.

   Doporučení: v celém tomto programu se nám bude hodit záložka Fantazie.

2. Přidáme postavu čarodějnice, upravíme její velikost a zajistíme její pohyb na šipky (pohyb na šipky je vysvětlen v úkolu 2). Aby se čarodějnice neschovávala za postavy umístíme ji do popředí pomocí příkazu *přejdi na vrstvu popředí* (tento krok je však možné doplnit kdykoliv později).

3. Vybereme postavy zvířátek nebo jiné, upravíme jejich velikost a umístíme je na vhodné pozice. Chceme, aby hra měla alespoň 3 interagující zvířátka, ale bude lepší je přidávat postupně, kvůli přehlednosti. Pokud potřebujeme zvíře otočit nastavíme mu způsob otáčení vlevo-vpravo a poté zvolíme směr, kterým chceme, aby se dívalo (*nastav způsob otáčení vlevo vpravo, nastav směr*).

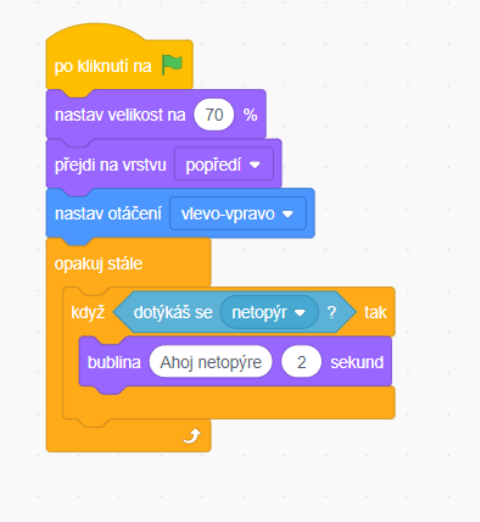
4. V tomto programu je dobré si postavy nazvat česky. Až budeme vytvářet podmínky, bude určitě přehlednější, když postavy budou mít námi zvolené názvy.



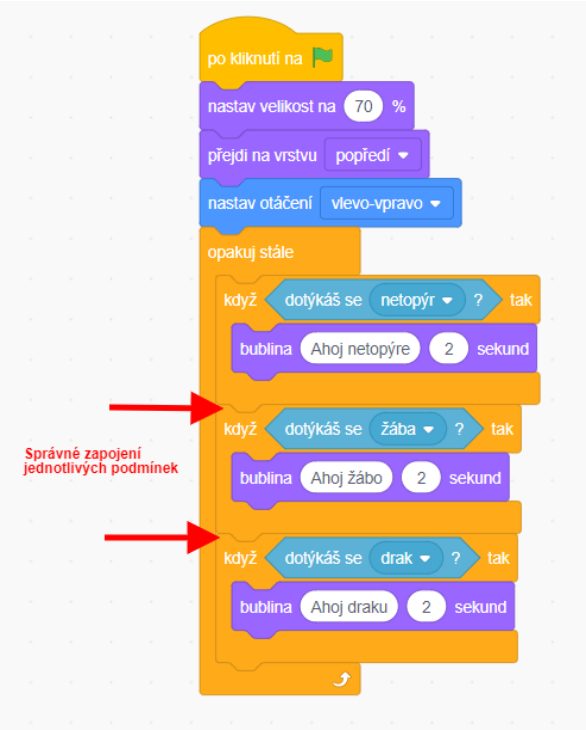
Obrázek 39 - Přejmenování postavy

5. Komunikace čarodějnice se zvířaty: U postavy čarodějnice *po stisku startu (zelená vlaječka)* použijeme nekonečný cyklus *opakuj stále* a do něj vložíme jednoduchou podmínku *když dotýkáš se …* . Podmínky umístíme do jednoho nekonečného cyklu, musíme dát pozor na to, aby jednotlivé podmínky byly zapojené pod sebou, kdybychom je zapojili jako vnořené, vyhodnocoval by se jen současný dotyk například žáby a netopýra s čarodějnicí. Bylo by možné i řešení pro každé zvíře

u čarodějnice vytvořit samostatný nekonečný cyklus s podmínkou, ale muselo by to být vždy umístěno pod samostatné *po stisknutí startu*. V každém případě po splnění podmínky čarodějnice na pár sekund zobrazí bublinu s pozdravem.



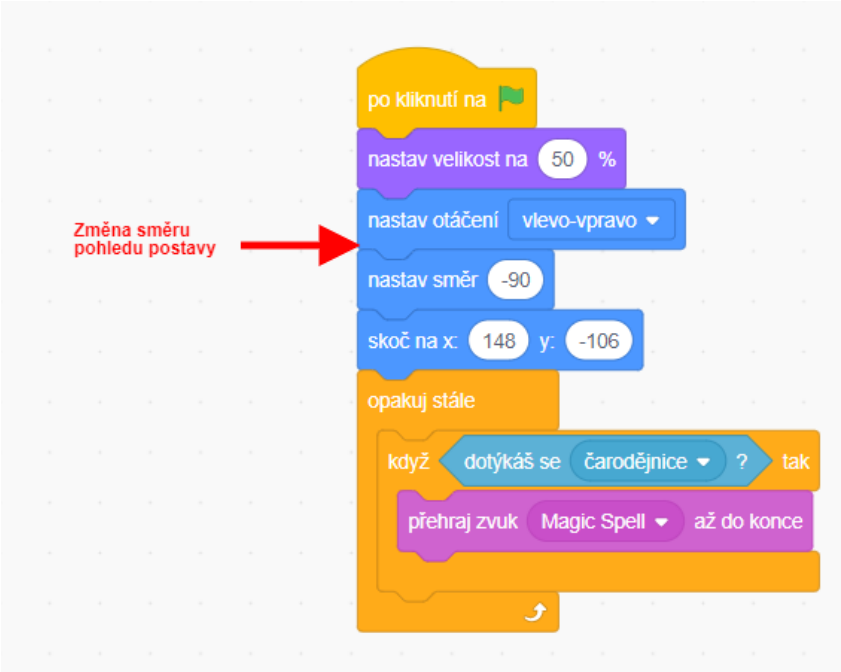
Obrázek 40 - Interakce postav – čarodějnice (opakuj stále a podmínka když)



Obrázek 41 - Interakce postav – čarodějnice (více podmínek a jejich správné zapojení)

6. Stejným způsobem vytvoříme podmínky u jednotlivých zvířat na pozdrav čarodějnice. Dobře zvažte, zda dětem umožníte použít zvuky, nebo i pozdrav zvířat realizujete pomocí bubliny.

   Poznámka: Berte v potaz, že 10 neustále „kvákajících“ počítačů (interakce probíhá po celou dobu, co se postavy dotýkají) nemusí být zrovna ideální, na druhou stranu žáci mají zvuky obvykle rádi. Jako kompromis lze využít zvuk v kombinaci příkazem *čekej x sekund*, pak počítače nebudou vydávat zvuky úplně nepřetržitě. Jako nejideálnější cestu bych doporučila použití sluchátek ke každému počítači, žáci uspokojí svou touhu po využití zvuků a zároveň nebude ve třídě nesnesitelný rámus.

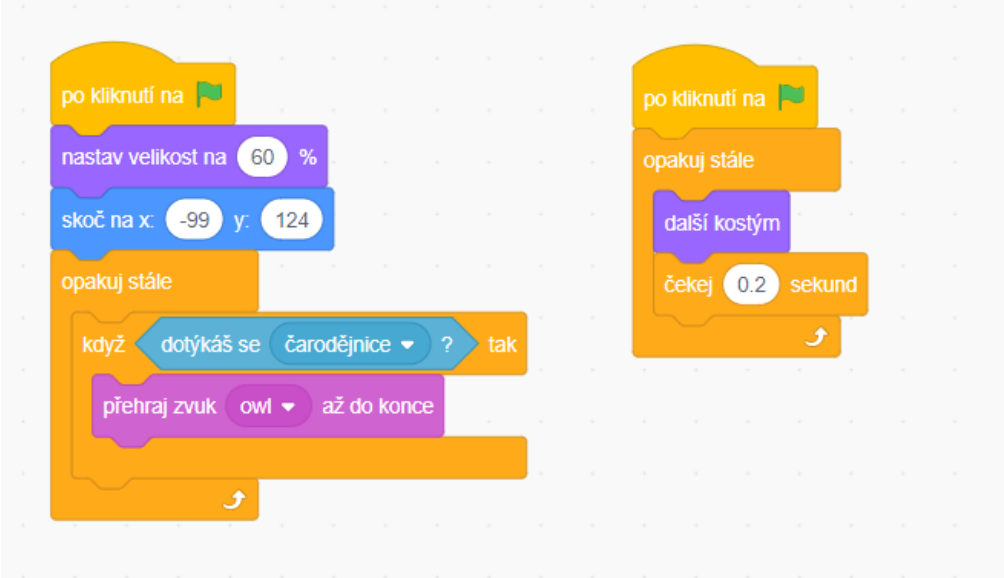


Obrázek 42 - Interakce postav – zvířátka

7. Bonusové úkoly

   a. Další zvířata fungují stejným způsobem jako ta, která již v programu máme.

   b. Neustálá změna kostýmů: spočívá v zařazení nekonečného cyklu pro změnu kostýmu. K tomuto účelu použijeme samostatné *po kliknutí na start (zelená vlaječka)* a do něj vložíme nekonečný cyklus (*opakuj stále*) obsahující změny kostýmů (*další kostým*) a k tomu krátké čekání (*čekej x sekund*). Čekání použijeme z důvodu toho, že jinak by se nám kostýmy měnily příliš rychle. Když bychom změnu kostýmu s čekáním umístili do cyklu, ve kterém máme interakci s čarodějnicí, nastal by problém v případě, když by program byl zrovna v čekající fázi, tak by na čarodějnici zvíře okamžitě nezareagovalo. Samozřejmě pokud máme nastavené čekání například na 0.2s, tak by si uživatel třeba ani nevšiml, že interakce nepřišla hned, ale v případě, že by čekání bylo například 1s, tak už by to mohlo vadit.



Obrázek 43 - Efekt dynamických postav (nepřetržitá změna kostýmů)

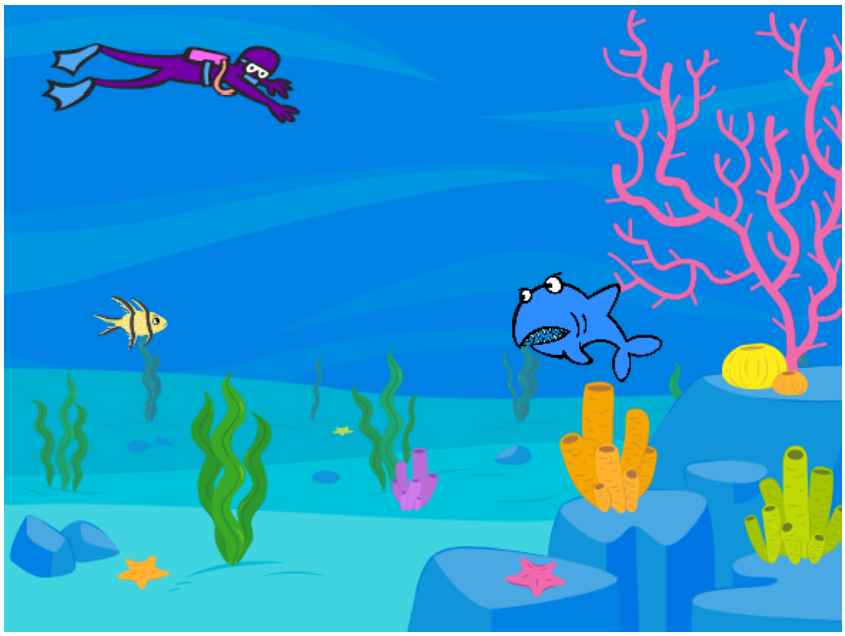
Možné problémy a komplikace

- Postava se pohybuje hlavou dolů - není nastaven způsob otáčení vlevo - vpravo.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.
- Interakce čarodějnice nefunguje se všemi zvířaty správně - podmínky jsou špatně zapojené nebo není správně definováno, čeho se má dotýkat.

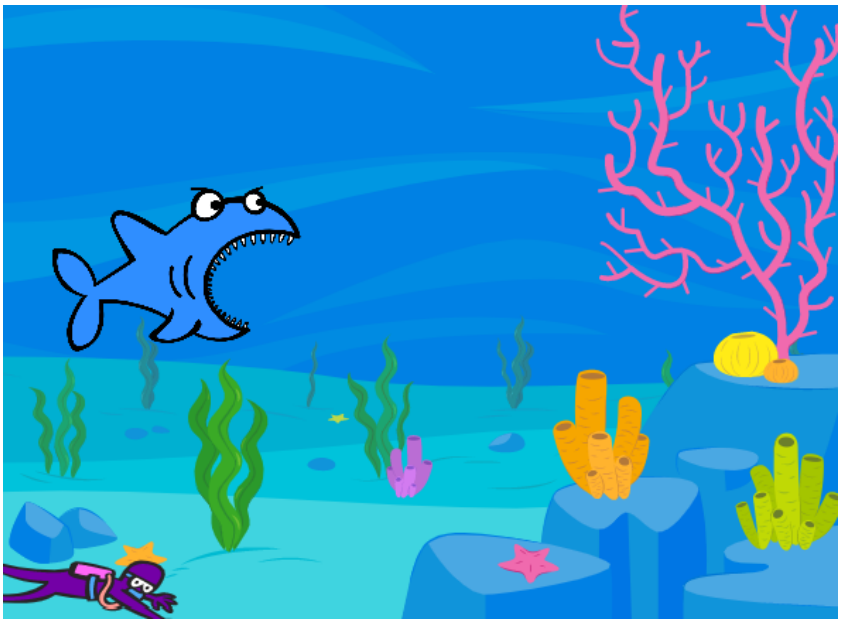
3.6.6 Úkol 6: Žralok

Příběh hry

Nacházíte se na dně oceánu. Bydlí zde žralok a rybičky, žralok potřebuje růst, proto musí jíst rybičky. Ale pozor na potápěče!



Obrázek 44 - Žralok náhled 1



Obrázek 45 - Žralok náhled 2

Cíl hry

Žák si upevní použití nekonečného cyklu a neúplné podmínky a seznámí se s využitím čekání po určitý čas, skrýváním postav a samovolným pohybem postavy.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, neúplná podmínka
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, práce s kostýmy, velikost postavy, pohyb postavy pomocí šipek, náhodná pozice, opakuj stále, podmínka když, vyhodnocení dotyku

Nové dovednosti

- **Algoritmizace:** čekání po daný čas
- **Scratch:** čekej x sekund, pohyb pomocí klouzání, skrývání a ukazování postav

Zadání pro žáky v krocích

1. Vyber vhodnou scénu.
2. Přidej postavu žraloka, uprav jeho velikost a zajisti jeho pohyb na šipky.
3. Přidej postavu ryby, uprav její velikost, zajisti, aby startovala na náhodné pozici a po celou dobu programu se klouzala náhodným směrem vhodnou rychlostí.
4. Žralok rád jí ryby:
    a. Žralok: otevře pusu, zavře pusu a trošku se zvětší.
    b. Ryba: skryje se, chvíli počká a objeví se jinak barevná na náhodné pozici.

    Nápověda: Budeš potřebovat jedno stálé opakování pro pohyb ryby a druhé stálé opakování pro dotek se žralokem, pokud by se vše dalo do jednoho a ryba byla zrovna ve fázi klouzání se, nemohla by vyhodnotit podmínku doteku se žralokem. Předtím než se ryba skryje využijte příkaz *čekej 0.1s,* je to důležité kvůli tomu, aby si i žralok stihl vyhodnotit, že rybu snědl než ryba zmizí.

5. Přidej potápěče, který začíná na náhodné pozici a pohybuje se stejným způsobem jako ryba, ale trošku pomaleji.
6. Žralok si musí dát na potápěče pozor, vždy když se jej začne dotýkat, tak se žralok začne maličko zmenšovat.
7. Bonusové úkoly:
    a. V moři je více ryb naráz a každá žraloka zvětší o jinou velikost.
    b. V moři je více pro žraloka nebezpečných objektů. Mohou jej zmenšovat, vrátit na výchozí velikost, nebo cokoliv tě napadne.

Zadání pro žáky ve formě úkolu

- Vytvoř podmořský svět, ve kterém žije žralok, kterého ovládáš pomocí šipek, a ryba, která stále plave. Žralok potřebuje růst a k tomu mu pomáhá to, že jí ryby.
- Zároveň si žralok musí dávat pozor na potápěče, protože při každém dotyku s ním se zmenšuje.
- Bonusové úkoly:
    - V moři je více ryb naráz a každá žraloka zvětší o jinou velikost.
    - V moři je více pro žraloka nebezpečných objektů. Mohou jej zmenšovat, vrátit na výchozí velikost, nebo cokoliv tě napadne.

Program

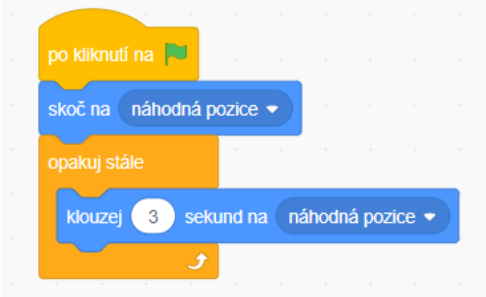
https://scratch.mit.edu/projects/336343710/

Řešení a metodické poznámky

1. Vybereme podmořské pozadí.
2. Doporučení: Pro tento projekt je vhodné postavy pojmenovat česky, kvůli přehlednosti při řešení interakce postav.
3. Přidáme postavu žraloka, upravíme jeho velikost a zajistíme jeho pohyb na šipky. Aby se žralok neschovával za ostatní postavy umístíme jej do popředí (*přejdi na vrstvu popředí)*.
4. Přidáme postavu ryby, upravíme její velikost, ryba startuje na náhodné pozici (*skoč na náhodnou pozici*) a po scéně se neustále klouže na náhodné pozice (*klouzej x sekund na náhodnou pozici*). Čas pro klouzání můžeme určit dle uvážení, čím vyšší číslo tím pomalejší pohyb, čím nižší číslo, tím rychlejší pohyb. Zároveň by bylo dobré naprogramovat, aby se vždy ryba po startu ukázala (*ukaž se*), je to z důvodu, že když bychom program zastavili zrovna ve fázi, kdy je ryba skrytá (snědená žralokem), neměla by se jak objevit.

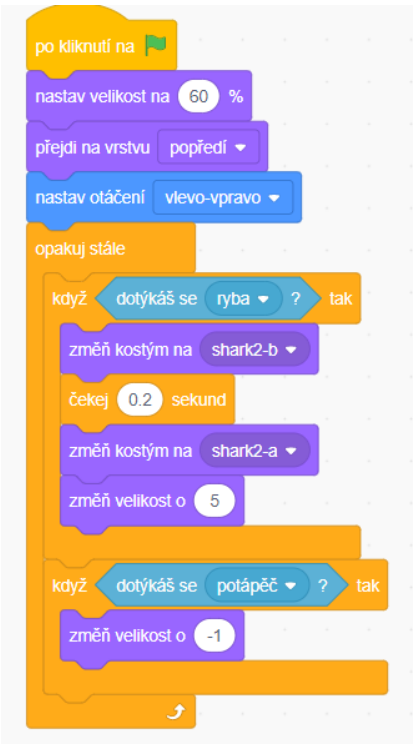
Obrázek 46 - Pohyb klouzáním (ryba)

5. Přidáme postavu potápěče, který startuje na náhodné pozici a pohybuje se stejným způsobem jako ryba (klouzáním), akorát trochu pomaleji (čas klouzání zvolíme vyšší číslo, než máme u ryby).



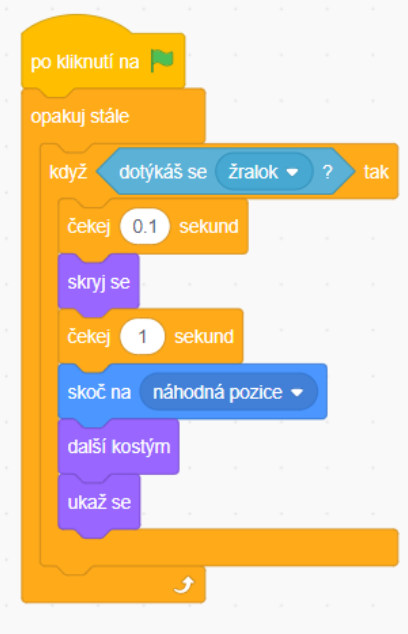
Obrázek 47 - Pohyb klouzáním potápěč

6. Interakce postav:
    a. U žraloka: Žralok stále vyhodnocuje, jestli se ryby dotýká, pokud ano tak otevře pusu, pár setin počká, zavře pusu (*změna kostýmu*) a zvětší se. Zároveň vyhodnocuje, zda se dotýká potápěče, v tom případě se po celou dobu dotyku zmenšuje.



Obrázek 48 - Žralok (interakce s rybou a potápěčem)

b. U ryby: Ryba stále (*opakuj stále*) vyhodnocuje, zda se dotýká žraloka, po kontaktu se žralokem 0.1 s počká (*čekej 0.1 s*) z důvodu, aby i žralok měl dost času vyhodnotit, že se společně dotkli (pokud bychom toto čekání vynechali mohla by nastat situace, že by ryba zmizela, ale žralok to vůbec nezaregistroval), skryje se (*skryj se*), počká, skočí na náhodnou pozici, změní kostým a ukáže se (*ukaž se*). U ryby použijeme samostatný nekonečný cyklus (*opakuj stále*) pro pohyb a samostatný pro interakci se žralokem, pokud bychom i podmínku pro pohyb dali do jednoho nekonečného cyklu a ryba zrovna byla ve fázi klouzání nemohla by zároveň vyhodnotit podmínku interakce se žralokem.



Obrázek 49 - Ryba (interakce se žralokem)

7. Bonusové úkoly fungují na stejném principu, jako ryba a potápěč. To, jak se budou nové postavy pohybovat a jaká bude jejich interakce je na fantazii žáků.

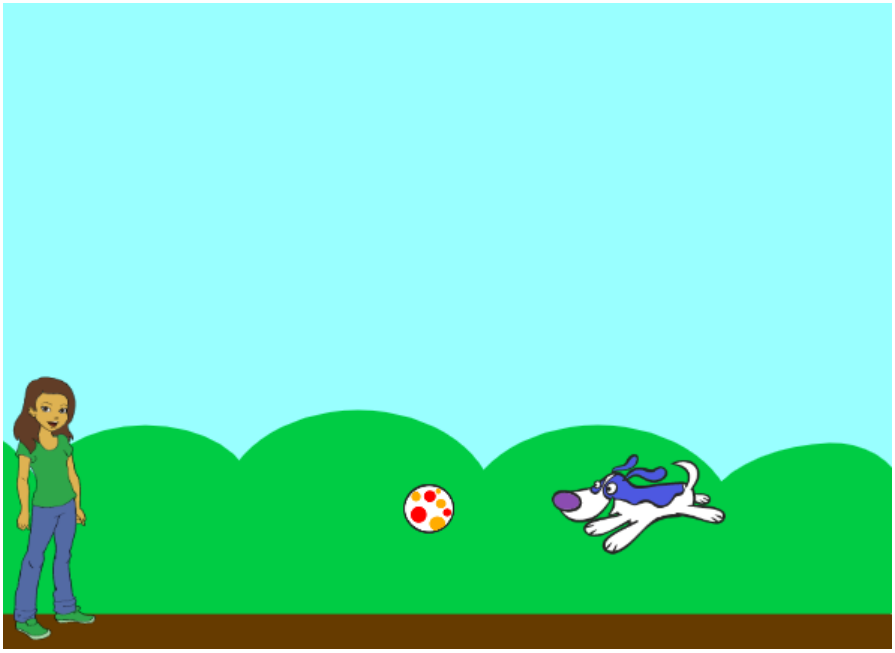
Možné problémy a komplikace

- Postava se pohybuje hlavou dolů - není nastaven způsob otáčení vlevo - vpravo.
- Žralok se schovává pod ostatní postavy - není nastaveno, že má být v popředí.
- Ryba není po startu vidět - u ryby chybí po startu *ukaž se* a program zrovna skončil se skrytou rybou.
- Ryba se skryje dřív, než žralok stihne zareagovat - doplňte k rybě čekání 0.1s předtím, než se skryje.
- Žralok je po startu hry obří nebo miniaturní - po startu není nastavena výchozí velikost žraloka.
- Ryba či potápěč se pohybují příliš rychle nebo příliš pomalu - je nevhodně nastavený čas po který se má postava klouzat. Čím vyšší číslo tím pomalejší pohyb, čím nižší číslo, tím rychlejší pohyb.

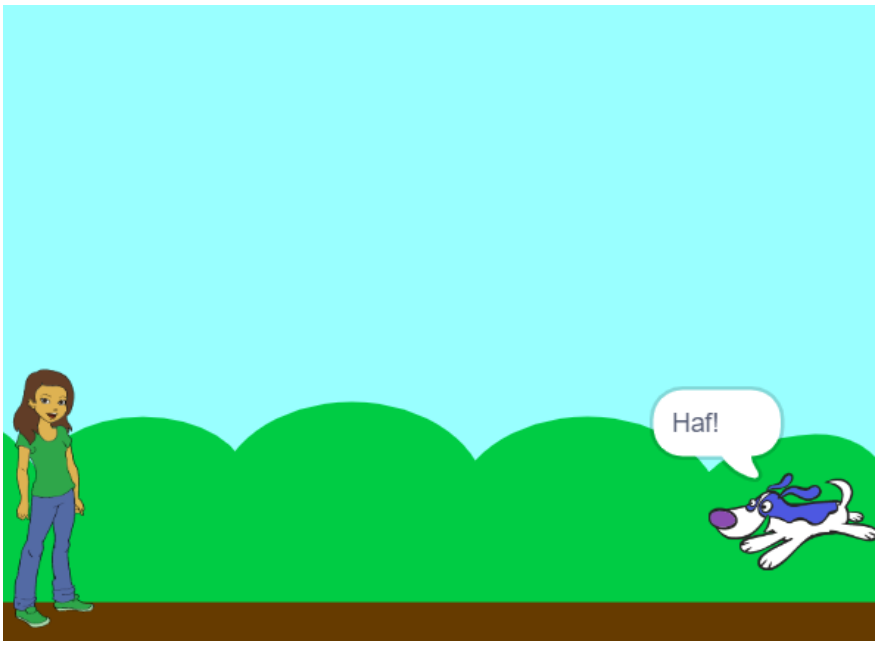
3.6.7 Úkol 7: Poslušný pejsek

Příběh hry

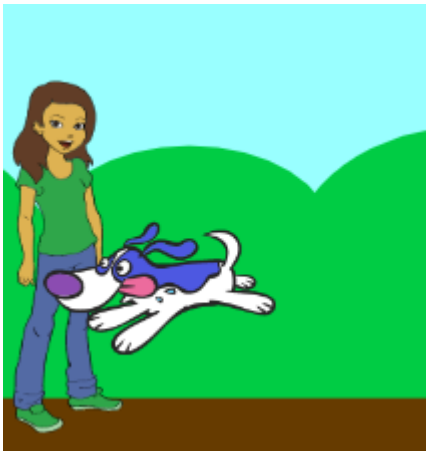
Páníček a jeho pes si vyšli na procházku do parku. Pejsek má moc rád chytání míčků, tak mu je páníček hází. Zároveň má pejsek rád páníčka, proto jej vždy v jeho blízkosti olízne.



Obrázek 50 - Poslušný pejsek náhled 1



Obrázek 51 - Poslušný pejsek náhled 2



Obrázek 52 - Poslušný pejsek náhled 3

Cíl hry

Žák se seznámí s cyklem s daným počtem opakování a úplnou podmínkou.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, neúplná podmínka, čekání po daný čas
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, práce s kostýmy, velikost postavy, pohyb postavy pomocí šipek, pozicování, opakuj stále, podmínka když, vyhodnocení dotyku, mluvící bublina, skrývání a ukazování postav

Nové dovednosti

- **Algoritmizace:** cyklus s daným počtem opakování, úplná podmínka
- **Scratch:** opakuj x krát, podmínka když … tak … jinak

Zadání pro žáky v krocích:

1. Vyber vhodnou scénu.
2. Přidej postavu člověka, v případě potřeby uprav její velikost a napozicuj ji na vhodné místo.
3. Postava hází míč: Přidej míč, v případě potřeby uprav velikost, který po stisknutí mezerníků postava hodí. Míč je vidět jen když letí, jinak je skrytý.

   Nápověda zde využiješ příkaz *opakuj "číslo" krát*, počet opakování musíš vyladit tak, aby míč doputoval až k okraji scény.

4. Přidej psa (ideálně takového, který umí vyplazovat jazyk), v případě potřeby uprav jeho velikost a zajisti pohyb psa na šipky.
5. Pes chytá míč, když jej chytne řekne Haf!
6. Pes má rád svého páníčka, vždy v jeho blízkosti vyplázne jazyk, jinak má jazyk schovaný.

   Nápověda: Bude se hodit podmínka: *když ... tak jinak*
7. Bonusový úkol:
   a. Postava může házet i jiné věci (například jídlo) a pes při jeho chycení může mít jiné reakce (například mňam, nebo se olíznout).

Zadání pro žáky ve formě úkolu

- Pes pohybující se na šipky je se svým páníčkem na procházce.
- Páníček mu hází míč a pes jej chytá.
- Pes má páníčka rád, proto má v jeho blízkosti vyplazený jazyk, aby jej mohl olíznout.
- Bonusový úkol:
  - Postava může házet i jiné věci (například jídlo) a pes při jeho chycení může mít jiné reakce (například mňam, nebo se olíznout).

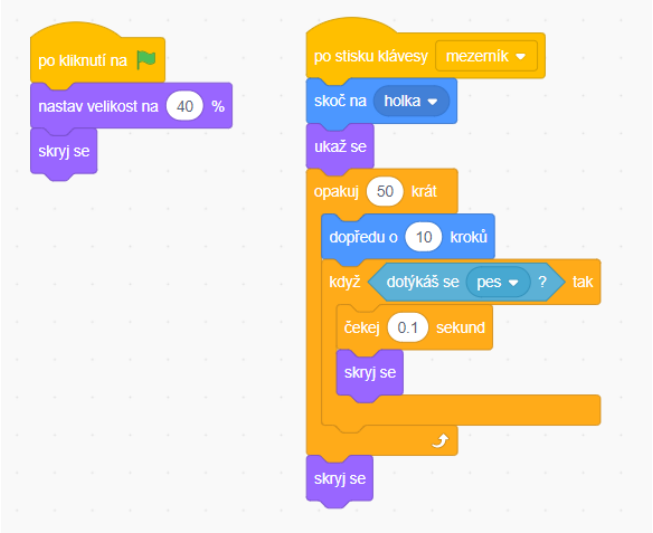
Program

https://scratch.mit.edu/projects/336999028/

Řešení a metodické poznámky

1. Vybereme vhodné pozadí.
2. Přidáme postavu člověka, napozicujeme ji a upravíme její velikost.
3. Přidáme postavu psa, upravíme jeho velikost a zajistíme jeho pohyb na šipky, zároveň je dobré pomocí příkazu *přejdi na vrstvu popředí*, zajistit, aby se pes neschovával za ostatní postavy.

4. Program míče:
    a. Míč je po startu skrytý, v případě potřeby upravíme jeho velikost.
    b. Po stisku mezerníku skočí na postavu, ukáže se a pomocí cyklu s daným počtem opakování se posouvá po scéně, na což využijeme příkaz *opakuj x krát* a do něj vložíme *dopředu o x kroků*. Po skončení tohoto cyklu se skryje.
        i. Doporučení: nejprve naprogramujte pohyb, počet opakování a kroků musíte vyladit dle vašich potřeb, až poté řešte interakci míče se psem.
    c. Při dotyku se psem se míč skryje. Tuto podmínku můžeme vložit buď do cyklu s daným počtem opakování, který je pro pohyb míče, nebo vytvořit samostatný nekonečný cyklus (*opakuj stále*). Stejně jako v minulém programu před skrytím míče počkáme 0.1s, kvůli tomu, aby stihl zareagovat i pes.

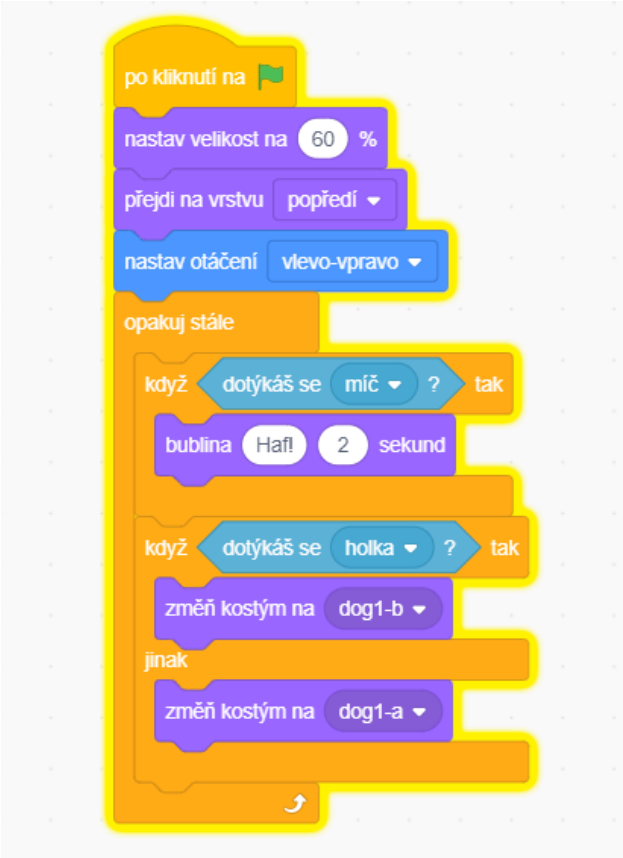


Obrázek 53 - Program míče

5. Program psa:

    a. Při dotyku s míčem pes řekne “Haf” (*bublina “Haf” 2 sekundy*), využijeme jednoduchou podmínku když.

    b. Při dotyku s postavou pes změní kostým na vyplazený jazyk, ale když se postavy nedotýká, má kostým s jazykem schovaným. Vyžijeme podmínku *když dotýkáš se postavy tak změň kostým na jazyk vyplazený*, *jinak změň kostým na jazyk zandaný* (podmínka *když ... tak ... jinak ...*)



Obrázek 54 - Program psa

6. Bonusový úkol:

    a. Hod jiných věcí vyřešíme přidáním nového objektu (postavy). Ke psovi přidáme další podmínky dotyku stejným způsobem jako máme pro míč akorát pes udělá něco jiného než “Haf”.

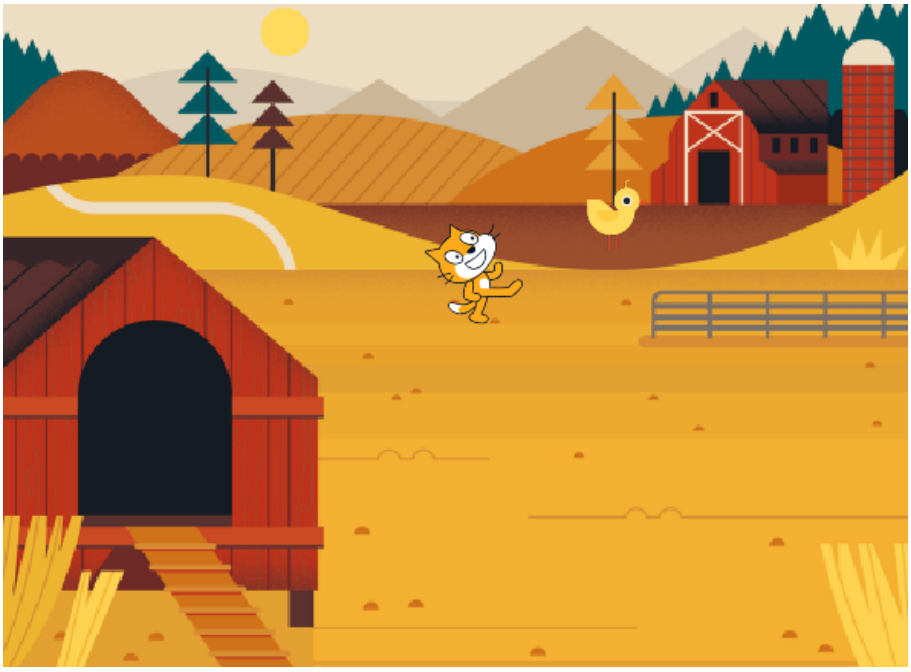
Možné problémy a komplikace

- Pes chodí hlavou dolů - není nastaven způsob otáčení vlevo - vpravo.
- Míč se nám neukázal nebo zůstal stát, tam kde nechceme - není definováno, kdy se má ukázat a kdy skrýt. Po startu je skrytý a po stisku mezerníku se ukáže, letí a buď se po daném počtu kroků a opakování skryje nebo k jeho skrytí dojde po chycení psem.
- Pes se schovává za ostatní postavy - je potřeba jej umístit do popředí.
- Míč se skryje dřív, než pes štěkne - chybí prodleva 0.1s, aby dotyk s jistotou stihl vyhodnotit i pes
- Je vhodné vybrat takovou postavu psa, která zvládne vyplazovat jazyk, abychom mohli pohodlně naprogramovat kontakt s člověkem. Pokud byla zvolena jiná postava než pes vyplazující jazyk, vymyslíme nějakou alternativní verzi toho, co bude postava dělat v přítomnosti člověka.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.

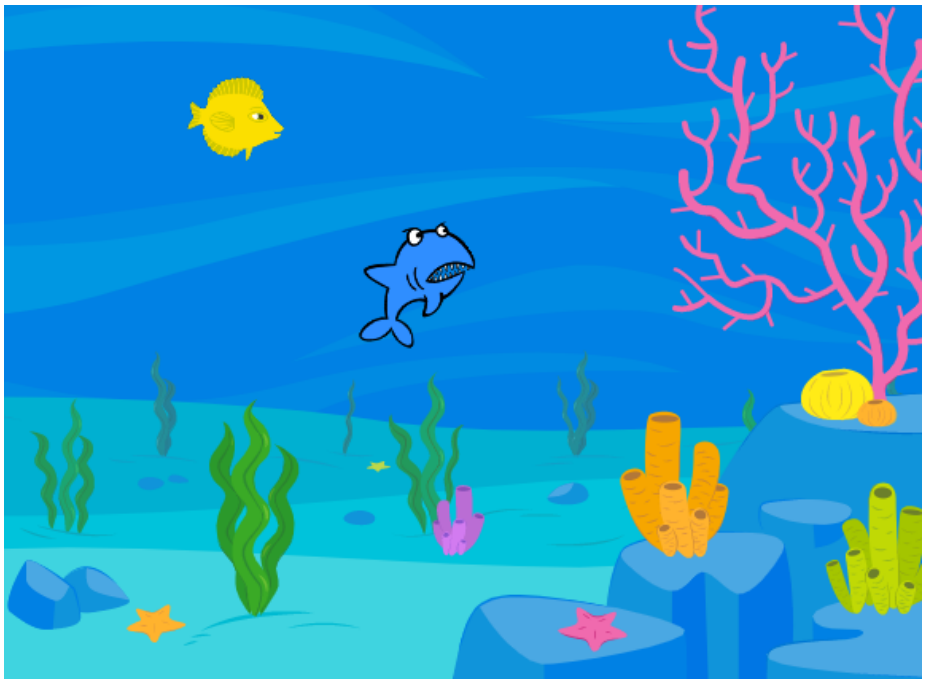
3.6.8 Úkol 8: Predátor a kořist

Příběh hry

V přírodě jsou některá zvířata predátory a jiná jejich kořistí. Stejně tak je to i v této hře.



Obrázek 55 - Predátor a kořist náhled 1



Obrázek 56 - Predátor a kořist náhled 2

Obrázek 57 - Predátor a kořist náhled 3

Obrázek 58 - Predátor a kořist náhled 4

Cíl hry

Žák se naučí ovládat postavu pomocí kurzoru myši a zopakuje si práci s kostýmy a pozadím.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, neúplná podmínka, čekání po daný čas
- **Scratch:** událost po startu, práce s postavami a pozadím, práce s kostýmy, velikost postavy, náhodná pozice, opakuj stále, podmínka když, vyhodnocení dotyku, čekej x sekund, pohyb pomocí klouzání

Nové dovednosti

- **Algoritmizace:** tato úloha je procvičovací
- **Scratch:** pohyb postavy pomocí myši (následování kurzoru)

Zadání pro žáky v krocích

1. V celé hře dbej na to, aby kombinace pozadí, predátor a kořist dávala smysl.
2. Vyber vhodnou scénu.

   Nápověda: pro hru jich budeš potřebovat několik, ale začneme jen s jednou.
3. Přidej predátora a zajisti jeho pohyb tak, aby se posouval za kurzorem myši.
4. Přidej kořist, která začíná na náhodné pozici a náhodně se klouže po scéně.
5. Pokud predátor kořist uloví, objeví se nové pozadí, nový predátor a nová kořist.

   Doporučení: dbej na vhodné pojmenování kostýmů a pozadí.

   Nápověda: aby predátor nechytil kořist hned znovu, nastav v cyklu krátké čekání po dotyku.
6. Doporučení: Aby hra nebyla tak snadná, dbej na vhodnou velikost a rychlost pohybu predátora i kořisti.
7. Hra bude mít minimálně 5 možných scén, tedy 5 rozdílných pozadí, predátorů a kořistí. Přičemž je stále potřeba myslet na to, aby to dávalo z přírodovědného hlediska smysl.

8. Bonusové úkoly:
    a. Více scén.
    b. Velikost postav měnící se v závislosti na scéně.
    c. Více kořistí na jednoho predátora. (Stále však stačí chytit jen jednu z nich.)

Zadání pro žáky ve formě úkolu

- Predátor pohybující se za kurzorem myši loví kořist ve vhodném prostředí, když ji chytí prostředí se změní a jiný predátor loví jinou kořist.
- Hra bude mít minimálně 5 možných scén, tedy 5 rozdílných pozadí, predátorů a kořistí. Přičemž je stále potřeba myslet na to, aby to dávalo z přírodovědného hlediska smysl.
- Bonusové úkoly:
    - Více scén.
    - Velikost postav měnící se v závislosti na scéně.
    - Více kořistí na jednoho predátora. (Stále však stačí chytit jen jednu z nich.)

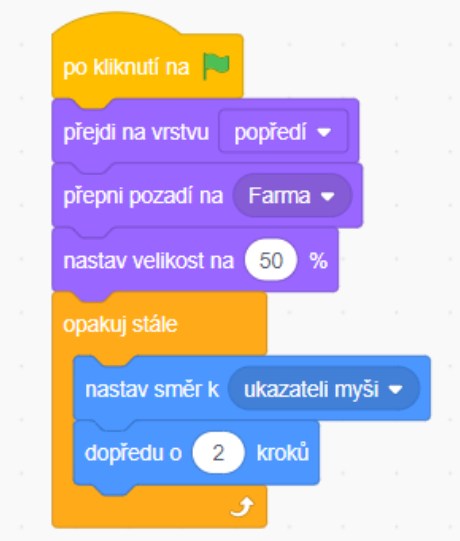
Program

https://scratch.mit.edu/projects/337328769/

Řešení a metodické poznámky

1. Vybereme vhodné pozadí. (Poznámka: do budoucna jich budou mít více, ale stačí začít s jedním.)
2. Doporučení: V celé této hře je vhodné dbát na správné pojmenovávání postav, kostýmů a pozadí. Jinak velmi snadno vzniknou zmatky.

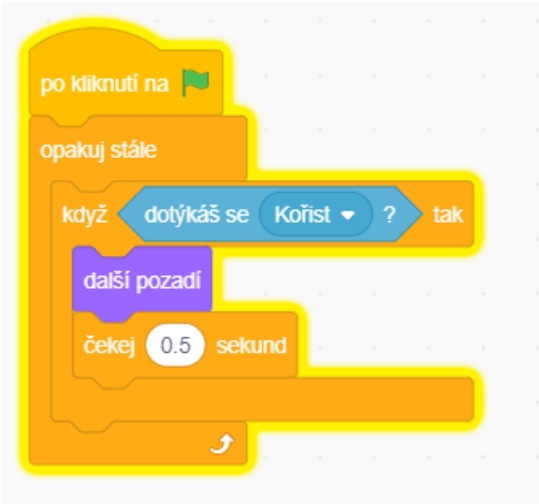
3. Přidáme postavu predátora, upravíme jeho velikost a zajistíme pohyb za kurzorem myši - postava se stále posouvá směrem ke kurzoru (*nastav směr k ukazatel myši a dopředu o x kroků*), rychlost pohybu (=počet kroků) zvolíme takovou, aby byla hra dobře hratelná. Postavu predátora umístíme do popředí, aby se neschovával pod kořist a také je důležité definovat pozadí, kterým hra začíná, aby se nám nestalo, že máme na začátku nevhodné postavy na nevhodném pozadí. (Poznámka: Volbu pozadí po startu hry můžeme definovat kdekoliv v programu.)



Obrázek 59 – Pohyb za kurzorem myši

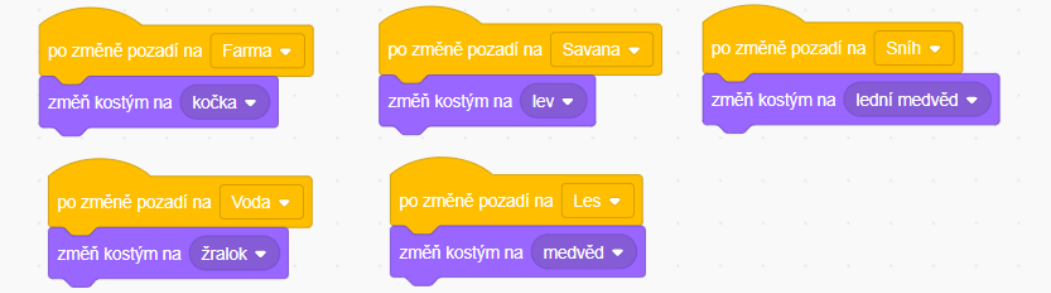
4. Přidáme postavu kořisti, která startuje na náhodné pozici, upravíme její velikost a naprogramujeme samovolný pohyb. Využijeme příkaz *opakuj stále*, který bude obsahovat *klouzání na náhodnou pozici po určitou dobu*, protože to je jednou z nejsnazších cest, jak samovolný pohyb realizovat, a žáci jej již znají z úkolu se žralokem a rybou.

5. Pokud se predátor dotkne kořisti, změní se pozadí na další. Zde je vhodné dát krátké čekání, aby se nám nestalo, že predátor svou kořist chytí hned znovu. Můžeme využít samostatné *po startu opakuj stále*, nebo podmínku vložit do části kódu, která realizuje pohyb za kurzorem myši.

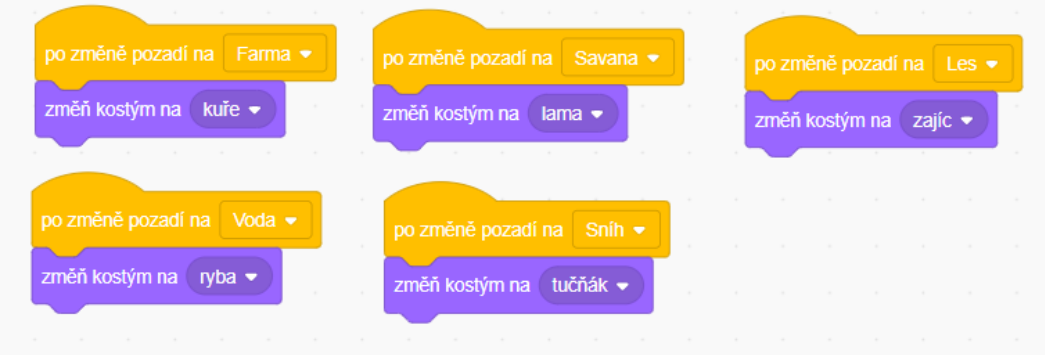


Obrázek 60 - Dotyk predátora a kořisti

6. V závislosti na tom, jaké je pozadí se mění kostým u predátora i u kořisti.



Obrázek 61 - Změna kostýmu predátor



Obrázek 62 - Změna kostýmu kořist

7. Bonusové úkoly:
    a. Více scén, tedy více kombinací pozadí, predátor a kořist. Vše funguje totožně jako doposud.
    b. Postavy mění v závislosti na pozadí velikost. Po změně konkrétního pozadí, kromě toho, že změníme predátorovi i kořisti kostým, přidáme ještě příkaz *nastav velikost na x %*.
    c. Přidáme další postavu kořisti s různými kostýmy. U predátora přidáme další podmínku *když dotýkáš se kořist2 a* její obsah bude stejný jako u kořisti 1, tedy změna pozadí. Kořist2 na základě změny pozadí také mění kostýmy.

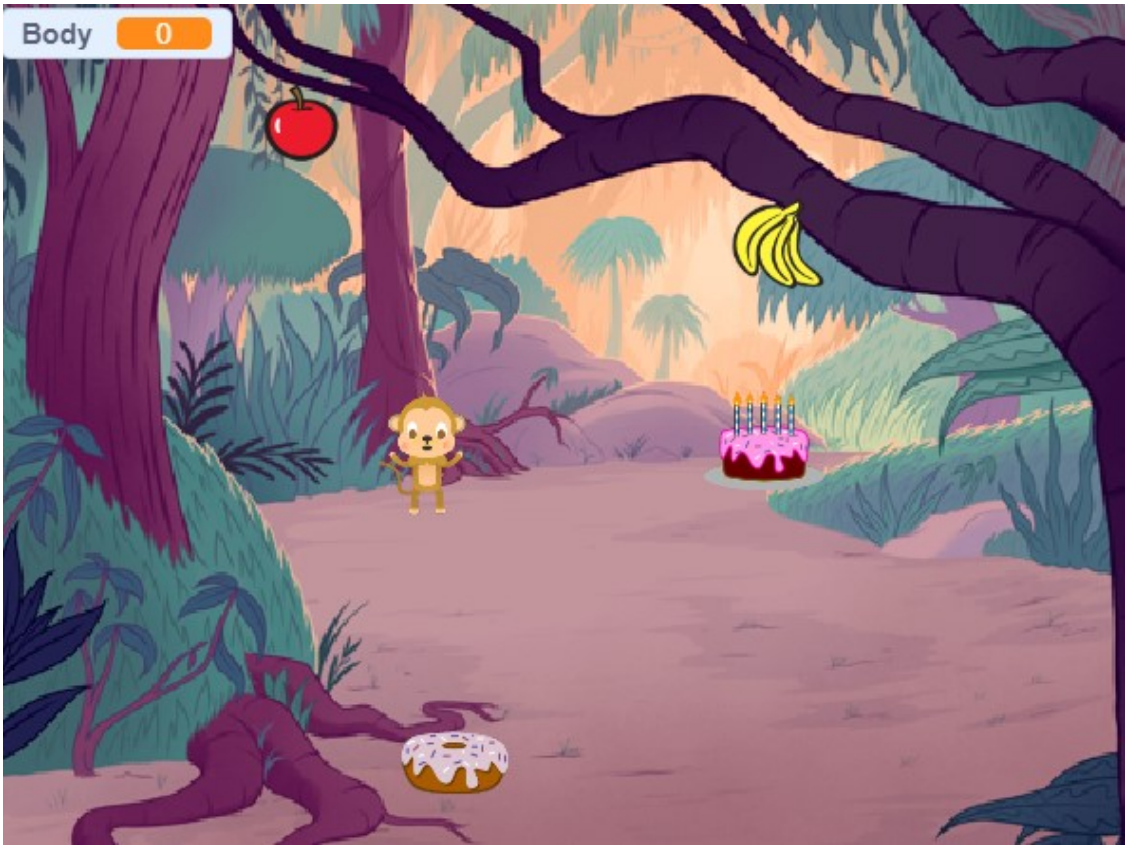
Možné problémy a komplikace

- Pohyb za kurzorem myši - žáky může napadnout, že by postava rovnou skákala na kurzor myši, což by samozřejmě také šlo, ale zrovna tato hra by díky tomu byla extrémně snadná.
- Na začátku hry máme jiné postavy, než bychom na daném pozadí chtěli - není definováno pozadí po startu hry.
- Predátor chytá kořist neustále za sebou, tedy se stále mění pozadí - nenastavili jsme krátkou prodlevu po chycení, aby měla nová kořist čas utéct.
- Predátor se pohybuje příliš rychle/pomalu - upravíme počet kroků o který se má posunout. Čím nižší číslo, tím pomalejší pohyb.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.
- Kořist se pohybuje příliš rychle/pomalu - upravíme dobu, po kterou se má klouzat na novou pozici. Čím nižší číslo, tím rychlejší pohyb.

3.6.9 Úkol 9: Opice

Příběh hry

V džungli žije opice, která má moc ráda banány, nepohrdne však ani jablky. Nicméně si musí dávat pozor na sladkosti, které tam neohleduplní turisté zanechávají, obzvláště nebezpečný je pro opici dort.



Obrázek 63 - Opice náhled

Cíl hry

Žák se seznámí s proměnnou.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, neúplná podmínka, čekání po daný čas
- **Scratch:** událost po startu, práce s postavami a pozadím, velikost postavy, pozicování, opakuj stále, podmínka když, vyhodnocení dotyku, čekej x sekund, skrývání a ukazování postav

Nové dovednosti

- **Algoritmizace:** proměnná
- **Scratch:** pohyb postavy pomocí myši (skoč na kurzor myši), práce s proměnnou

Zadání pro žáky v krocích

1. Vyber vhodnou scénu.
2. Přidej opici, uprav její velikost a zajisti, aby se stále nacházela tam, kde je kurzor myši.
3. Přidej banány, uprav jejich velikost a zajisti, aby se objevovaly na náhodných pozicích, vždy tam chvíli vydržely, poté se skryly a zůstaly chvíli skryté. Následně se zase objeví…
4. Založ proměnnou “Body”, nastav ji na 0 a zajisti, aby byla zobrazená.

   Nápověda: proměnnou můžeme zakládat jak u postav, tak u scény chceme, aby naše proměnná byla společná pro všechny postavy ve hře.
5. Když opice chytí banán, dostaneme bod a banán se skryje.
6. Přidej další objekty (jablko, koblihu, dort) se stejným chováním jako mají banány (mohou být vidět rozdílnou dobu).
7. Když opice chytí banán dostaneme 3 body, když jablko tak 1 bod, když koblihu tak o 1 bod přijdeme a když chytí dort naše bodové skóre se vynuluje.
8. Zajisti, aby se jídlo neobjevovalo všechno naráz.
9. Bonusové úkoly:
   a. Více jídla s různým bodovým hodnocením.
   b. Opice vyjádří radost/smutek z chyceného jídla.

Zadání pro žáky ve formě úkolu

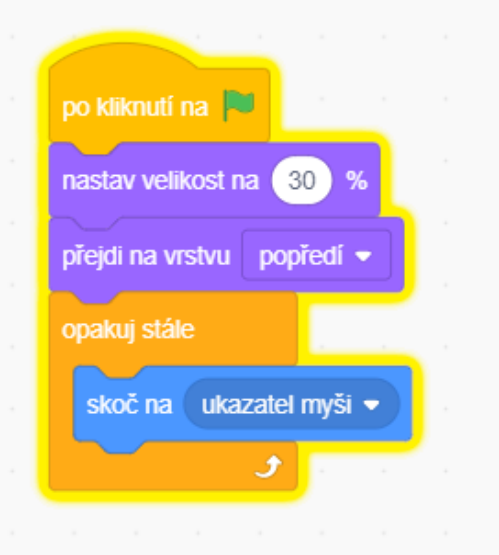
- Opice pohybující se na kurzoru myši je velice hladová. Za každé snědené jídlo dostane body. Je pro ni vhodné zejména ovoce (za něj se body přidávají) naopak sladkosti jí velmi škodí (za ně se jí body ubírají nebo dokonce vynulují).
- Jídlo se objevuje na náhodných místech.
- Bonusové úkoly:
    - Více jídla s různým bodovým hodnocením.
    - Opice vyjádří radost/smutek z chyceného jídla.

Program

https://scratch.mit.edu/projects/338197186/

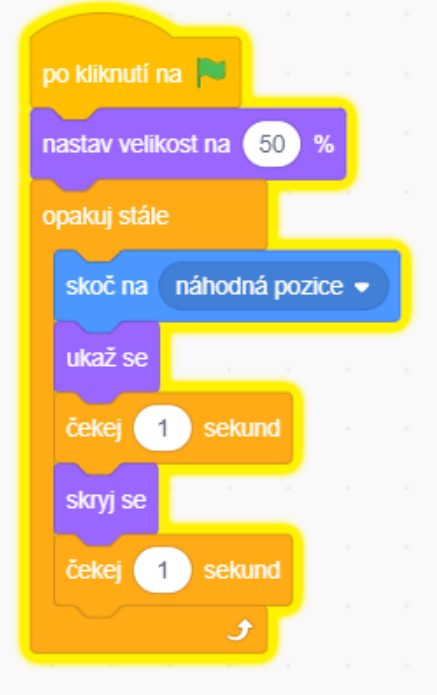
Řešení a metodické poznámky

1. Vybereme vhodné pozadí.
2. Přidáme postavu opice, upravíme její velikost a umístíme ji do popředí (aby se neschovávala pod jídlo). Následně zajistíme, aby se postava stále nacházela na kurzoru myš pomocí příkazů *opakuj stále skoč na ukazatel myši*.



Obrázek 64 - Pohyb na kurzor myši

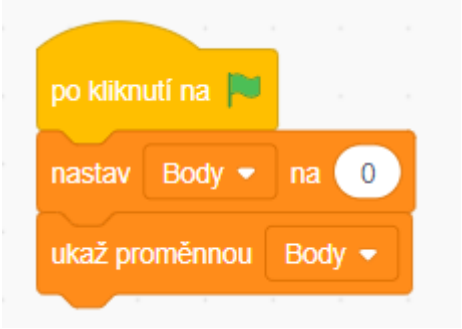
3. Přidáme postavu banánu a upravíme jeho velikost. Banán se vždy na chvíli objeví na náhodné pozici a poté zase skryje, toto chování opakuje po dobu celé hry. Čím kratší dobu je banán vidět, tím je hra obtížnější, protože je náročnější jej chytit.



Obrázek 65 - Náhodné zobrazování ovoce

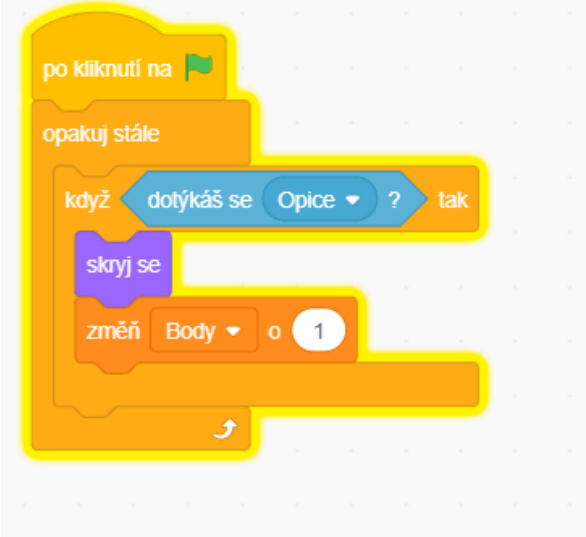
4. Založíme proměnnou body: V záložce Proměnné zvolíme vytvoř proměnnou. Tuto proměnnou můžeme zakládat jak u scény, tak u postav. Chceme však, aby ji znaly všechny postavy (jedná se o proměnnou globální, tím však žáky nezatěžujeme). Pokud ji zakládáme u scény tak je globální vždy, pokud u některé z postav je potřeba zvolit "pro všechny postavy". Tuto novou proměnnou nastavíme na 0 a ukážeme ji.

   Doporučení: v rámci přehlednosti programu je dobré proměnné definovat u scény, abychom vždy hned věděli, kde jejich sklad máme, vyhneme se tak i volbě, jestli bude globální či lokální, protože proměnné založené u scény jsou vždy globální.



Obrázek 66 - Založení proměnné

5. Zisk bodů za chycení: K ovoci naprogramujeme podmínku *opakuj stále když dotýkáš se opice*, která bude obsahovat změnu skóre (*změň body o…*) a skrytí daného ovoce.



Obrázek 67 - Zvýšení skóre

6. Přidáme další jídlo se stejným chováním jako banán, jen bodový zisk je různý:
    a. banán +3 body (*změň Body o 3*)
    b. jablko +1 bod (*změň Body o 1*)
    c. kobliha -1 bod (*změň Body o -1*)
    d. dort anuluje body (*nastav Body na 0*)
7. Asynchronní objevování jídla:
    a. můžeme nastavit každému jídlu rozdílnou dobu kdy je vidět a kdy je skryté
    b. můžeme před tím, než spustíme cyklus pro objevování jídla dát rozdílně dlouhou prodlevu
    c. můžeme využít operátor náhodné číslo (ten však žáci v tuto chvíli zatím neznají) a realizovat pomocí něj variantu A nebo B
8. Bonusové úkoly:
    a. Další jídlo funguje stejně jako to, které již v programu máme.
    b. Opice reaguje na chycení jídla - doplníme podmínku u opice *když dotýkáš se…* a vymyslíme co se má stát.

Možné problémy a komplikace

- Opice se schovává pod ovoce - není nastaveno, aby opice byla v popředí
- Body se neukazují nebo je program nezná - nemáme proměnnou vytvořenou nebo definovanou nebo nemáme nastaveno její zobrazení (*ukaž proměnnou*).
- V bonusovém úkolu opice nestihne reagovat dřív, než se jídlo skryje - doplníme k jídlu prodlevu 0.1s předtím než se skryje.
- Jídlo je zobrazeno příliš krátce a rychle se přesouvá jinam - je nastavena příliš krátká nebo žádná doba čekání.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.

3.6.10 Úkol 10: Balóny

Příběh hry

Kolikrát dokážeš chytit balón jedné barvy, než bude úplně maličký?

Obrázek 68 - Balóny náhled 1

Obrázek 69 - Balóny náhled 2

Cíl hry

Žák se seznámí s využitím více proměnných v jednom programu a naučí se vytvořit samovolný pohyb.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, proměnná
- **Scratch:** událost po startu, událost po kliknutí, práce s postavami, velikost postavy, pozicování, opakuj stále, práce s proměnnou

Nové dovednosti

- **Algoritmizace:** více proměnných
- **Scratch:** samovolný pohyb, náhodné číslo

Zadání pro žáky v krocích

1. Přidej postavu balónu jedné konkrétní barvy, který se bude stále pohybovat po scéně. Vždy jsme programovali pohyb klouzáním, nyní promysli, zda by to šlo i jinak.

   Nápověda: Budou se hodit příkazy pro nastavení směru a nově využijeme operátor náhody - náhodné číslo.

2. Když na balón klikneme (chytíme jej), přičte nám body pro konkrétní barvu balónu. Následně se přesune na jinou pozici, zmenší se, náhodně změní směr, kterým se pohybuje.

3. Celkem budou balóny tři: modrý, žlutý a fialový. Balóny se chovají stejně, nicméně každý si sám počítá počet, kolikrát byl chycen.

4. Každý balón se pohybuje jinou rychlostí.

5. Bonusový úkol:

   a. Čím je balón menší tím se pohybuje rychleji.

      Nápověda: Budou potřeba další proměnné pro rychlost balónů.

Zadání pro žáky ve formě úkolu

(při využití tohoto zadání, je vhodné ukázat žákům hotovou hru)

- Tři balóny se pohybují po scéně. Vždy jsme programovali pohyb klouzáním, nyní promysli, zda by to šlo i jinak.

  Nápověda: Budou se hodit příkazy pro nastavení směru a nově využijeme operátor náhody - náhodné číslo.

- Poté co balón chytíme přičte body své barvě, změní směr, změní pozici a zmenší se.
- Bonusový úkol:
  - Čím je balón menší, tím se pohybuje rychleji.

    Nápověda: Budou potřeba další proměnné pro rychlost balónů.

Program

https://scratch.mit.edu/projects/339790539/

Řešení a metodické poznámky

1. Přidáme postavu balónu. Je vhodné si jej pojmenovat (ideálně barvou, kterou má) a smazat mu nepotřebné kostýmy. Zároveň mu *nastavíme velikost na 100 %*.
2. Pohyb balónu:
    a. Balón se pohybuje náhodným směrem: *nastav směr* a *náhodné číslo*.

    Poznámka: to, jaké číslo vložit do rozmezí pro náhodné číslo můžeme zjistit snadno na kruhu výběru směru, že tam najdeme nejnižší a nejvyšší hodnotu (-180 až 180). Směr se nastaví pouze jednou mimo cyklus, tímto směrem se balón vydá.

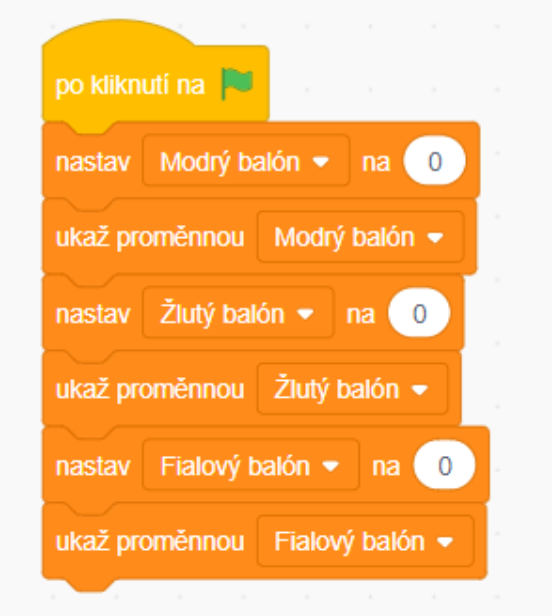
    b. Samotný pohyb zajistíme přidáním cyklu obsahujícím *posun dopředu o daný počet kroků* (i zde můžeme využít prvek náhody, čímž zajistíme, že každý balón se pohybuje náhodnou rychlostí) a příkazem *když narazíš na okraj, odraz se*.

    c. Je dobré nastavit *způsob otáčení vlevo-vpravo*, aby balón nelétal podivně otočený.

Obrázek 70 - Pohyb balónu (nastavení směru, posun, odraz)

3. Chytání balónu:

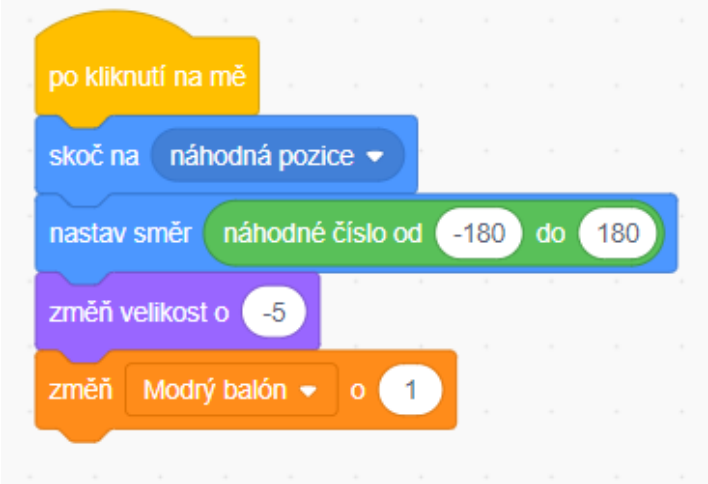
   a. Pro každou barvu balónu budeme mít samostatnou proměnnou. Proměnné musíme založit a zobrazit.



Obrázek 71 - Proměnná pro každý balón

   b. Balón chytíme kliknutím na něj.

   c. Skočí na náhodnou pozici, změní náhodně směr, zmenší se a změní proměnnou pro svou barvu. (Poznámka: pořadí příkazů, které balón udělá po chycení může být zcela libovolné, vše se provede v mžiku.)



Obrázek 72 - Chycení balónu

4. Bonusový úkol:
    a. Pro bonusový úkol bude potřeba založit další proměnné (rychlost modrý, rychlost žlutý, rychlost fialový), tyto proměnnou vložíme do počtu kroků u balónu dané barvy, o které se má balón posunout a zároveň se při každém chycení balónu dané barvy budou zvyšovat.

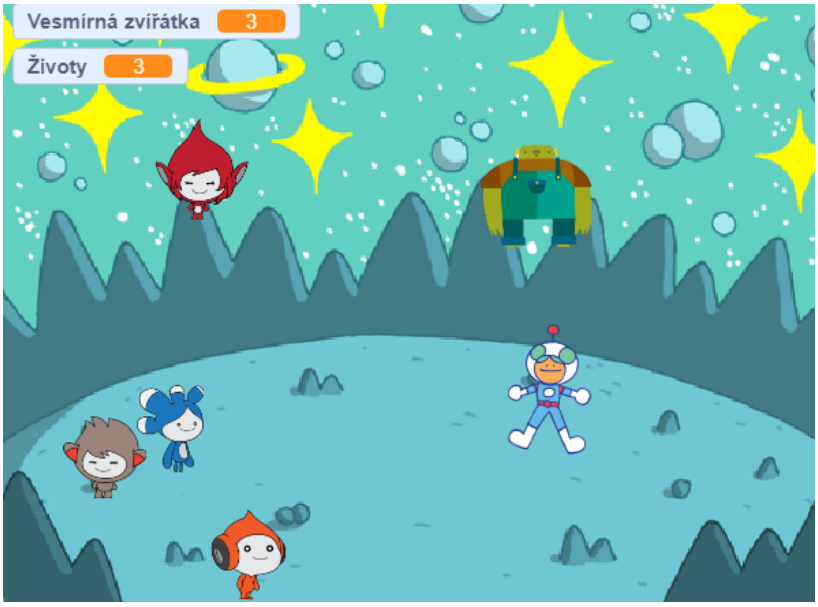
Možné problémy a komplikace

- Balón začíná příliš malý - po startu chybí definování velikosti.
- Balón se pohybuje zvláštně vzhůru nohama a podivně nakřivo - je vhodné nastavit způsob otáčení vlevo-vpravo
- Balón se pohybuje sekavě, pořád mění směr - příkaz pro změnu směru musí být umístěn mimo cyklus. Balón si nejprve nastaví směr a poté nastane neustálé opakování pohybu.
- Balón dojede ke kraji a tam se zasekne - v cyklu chybí příkaz *když narazíš na okraj odraz se*.
- Body za chycení balónů se neukazují nebo je program nezná - nemáme proměnné vytvořené nebo definované nebo nemáme nastaveno jejich zobrazení (*ukaž proměnnou*).

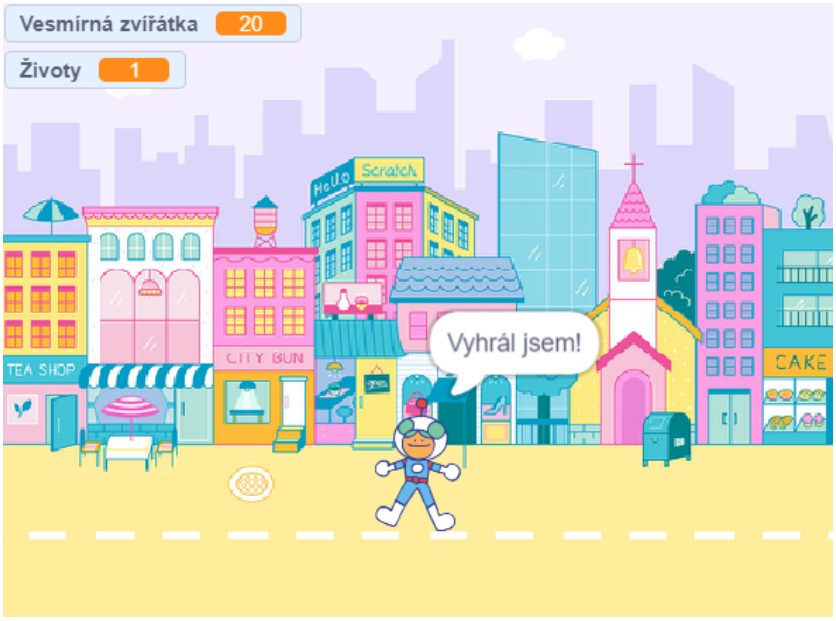
3.6.11 Úkol 11: Vesmírná mise

Příběh hry

Jsi kosmonautem vyslaným do vesmíru a tvým úkolem je posbírat 20 vesmírných zvířátek, poté se můžeš vrátit zpátky na Zem. Ale pozor na zlého obra, když kosmonautovi sebere všechny životy, zůstane navždy uvězněn ve vesmíru.



Obrázek 73 - Vesmírná mise náhled 1



Obrázek 74 - Vesmírná mise náhled 2

Obrázek 75 - Vesmírná mise náhled 3

Cíl hry

Žák se seznámí s relačním operátorem (je rovno), čekáním s danou podmínkou a cyklem s danou podmínkou.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, neúplná podmínka, proměnná
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, velikost postavy, pohyb postavy pomocí šipek, náhodná pozice, opakuj stále, podmínka když, vyhodnocení dotyku, práce s proměnnou, samovolný pohyb postavy, skrývání a ukazování postav

Nové dovednosti

- **Algoritmizace:** relační operátor (je rovno), čekáním s danou podmínkou, cyklus s danou podmínkou
- **Scratch:** operátor porovnání, čekej dokud nenastane, opakuj dokud nenastane

Zadání pro žáky v krocích

1. Vyber vesmírné pozadí.
2. Přidej postavu kosmonauta, uprav jeho velikost a zajisti jeho pohyb na šipky.
3. Přidej postavy vesmírných zvířátek (Giga, Pico, Nano, Tera) a zajisti jejich samovolný pohyb.

   Doporučení: Samovolný pohyb by bylo lepší naprogramovat stejným způsobem jako v minulém úkolu s balónky (nastavení směru a neustálý pohyb), protože následně budeš mít díky tomu jednodušší situaci.
4. Poté co je zvířátko chyceno kosmonautem započítá se bod.

   Nápověda: Musíš ošetřit, aby se bod přidal jen jeden za každé chycení.
5. Když kosmonaut chytí 20 zvířátek může se vrátit na zem, zvířátka už se dále nezobrazují a nelze je chytat.

   Nápověda: Bude se hodit příkaz *čekej dokud nenastane*.
6. Kosmonaut má omezené množství životů a ve vesmíru je „škodič“, který jej o ně chce připravit, pokud se mu to povede dřív, než kosmonaut zvládne posbírat zvířátka musí uznat svou prohru.

   Nápověda: Mohl by tě zajímat příkaz *opakuj dokud nenastane*.
7. Bonusové úkoly:
   a. Zvířátka se pohybují náhodnou rychlostí.
   b. Jednou za čas se ve hře objeví prvek, který kosmonautovi život přidá.

Zadání pro žáky ve formě úkolu

- Kosmonaut ve vesmíru musí posbírat 20 vesmírných zvířátek, která tam žijí a samovolně se pohybují.
- Poté co jich 20 uloví může se vrátit na zem.
- Kosmonaut má omezené množství životů a ve vesmíru je škodič, který jej o ně chce připravit, pokud se mu to povede dřív, než kosmonaut zvládne posbírat zvířátka musí uznat svou prohru.
- Bonusové úkoly:
    - Zvířátka se pohybují náhodnou rychlostí.
    - Jednou za čas se ve hře objeví prvek, který kosmonautovi život přidá.

Program

https://scratch.mit.edu/projects/340137561/

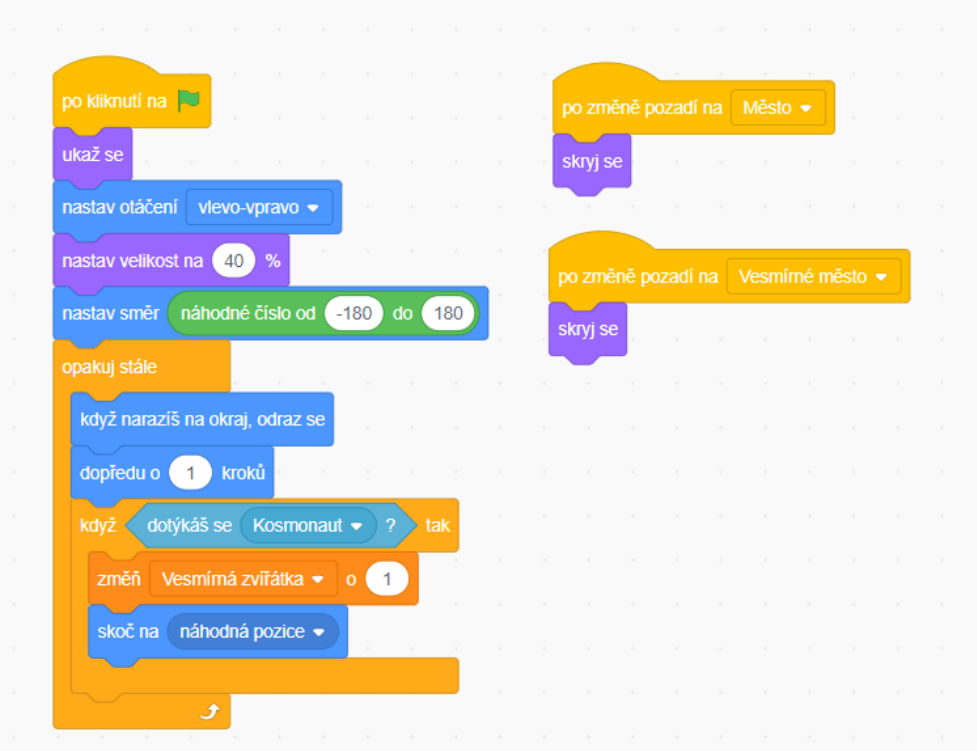
Řešení a metodické poznámky

1. Vybereme vesmírné pozadí.
2. Přidáme postavu kosmonauta, upravíme jeho velikost, umístíme ho do popředí a zajistíme jeho pohyb na šipky.
3. Přidáme postavy vesmírných zvířátek (příšerky: Giga, Pico, Nano, Tera), upravíme jejich velikost, zajistíme start na náhodné pozici. (Poznámka: kdyby postavy nestartovaly na náhodné pozici, vyhodnotily by se změny bodů/životů hned po startu, protože by se mohlo stát, že by postavy začaly všechny na jednom místě.) Následně vytvoříme samovolný pohyb. Pohyb již umíme udělat dvěma způsoby
    a. Klouzáním (tato varianta pro daný program není úplně ideální, protože poté nám to bude trochu komplikovat situaci při zajištění, aby se body za chycení nepřidaly vícekrát. Nicméně to není neřešitelný problém.
    b. Nastavením náhodného směru (*nastav směr náhodné číslo)* a následným pohybem vpřed a zajištěním, aby se postava odrážela od okrajů. (Stejně jako

u předchozího úkolu s balónky.) Tato varianta je vhodnější pro usnadnění dalšího programování.

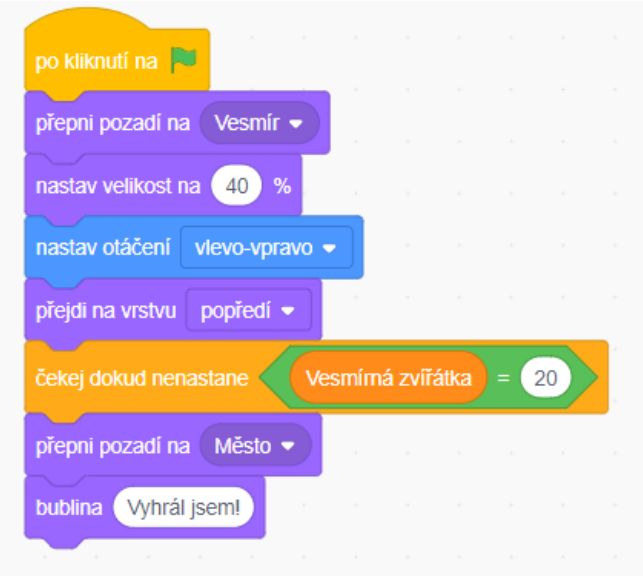
4. Založíme proměnnou počítající, kolik zvířátek už kosmonaut chytil a zajistíme, aby se vždy při chycení zvířátka přičetl bod. Podmínka se nachází u zvířátek z důvodu, abychom mohli zajistit, že zvíře nebude chyceno vícekrát na jeden dotyk, úplně nejjednodušší variantou je chycené zvíře posunout na náhodnou pozici, ale můžeme použít i např. čekání nebo skrytí a následné ukázání. Nicméně skrývat a ukazovat zvířátka příliš nedoporučuji, protože to by nám opět zkomplikovalo situaci při návratu na zem.

   Poznámka: pokud se zvířátka pohybují klouzáním, musí být podmínka pro chycení a přičtení bodů v samostatném nekonečném cyklu, zároveň je vhodné po chycení buď chvíli počkat nebo zvířátko na chvíli skrýt a pak ukázat, aby se nepřičetlo více bodů naráz.



Obrázek 76 - Program vesmírných zvířátek (pohyb a interakce s kosmonautem)

5. Návrat na Zemi: (programujeme ke kosmonautovi)
    a. Přidáme pozadí návratu na Zemi.
    b. Počkáme dokud nenastane počet chycených zvířat = 20 (čímž zavádíme dvě nové věci *čekej dokud nenastane* a *operátor porovnání hodnot*)
    c. Změníme pozadí a kosmonaut prohlásí "Vyhrál jsem". (*bublina*)
    d. U ostatních postav přidáme: *Po změně pozadí skryj se*. a naopak po startu se musí ukázat.
    e. Zároveň musíme po startu hry příkazem *přepni pozadí* definovat, kterým pozadím hra začíná.
    f. Totéž by bylo možné naprogramovat i pomocí nekonečného cyklu s podmínkou. (*Opakuj stále když Vesmírná zvířátka = 20*)

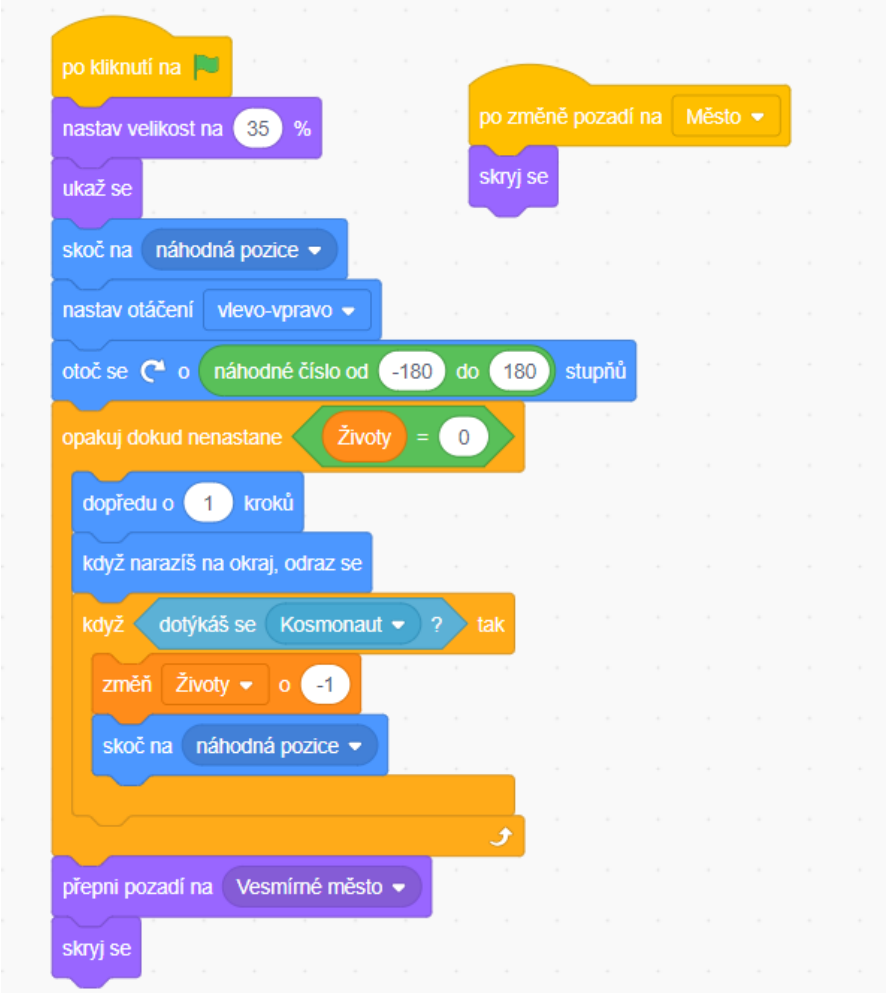


Obrázek 77 - Návrat na zem (výhra kosmonouta)

6. Škodící postava:
    a. Založíme další proměnnou “životy”, kterou nastavíme na jiné číslo než 0. (Poznámka: Tato hodnota udává, kolik životů pro hru budeme mít.) Proměnnou ukážeme.
    b. Přidáme postavu, která bude zlá a bude životy ubírat, její pohyb funguje stejně jako u vesmírných zvířátek. Nicméně místo *opakuj stále* můžeme využít příkaz *opakuj dokud nenastane*, více je popsáno níže. Zároveň nezapomeneme na úpravu velikosti, aby byla po startu vidět, a start na náhodné pozici.
    c. Přidáme pozadí pro prohru.

7. Vyhodnocení prohry: (Pro vyhodnocení prohry existují minimálně 3 možnosti.)

   a. *Opakuj dokud nenastane životy = 0*: místo nekonečného cyklu použijeme *opakuj dokud nenastane*, tento příkaz funguje tak, že dokud není splněna podmínka, děje se činnost uvnitř cyklu, tedy v našem případě pohyb a ubírání životů. Ve chvíli, kdy životy = 0 program z cyklu vyskočí a dějí se příkazy pod ním, což je *změna pozadí a skrytí* postavy. Kosmonaut *po změně scény (prohra)* řekne "Prohrál jsem". a zvířátka se po změně této scény skryjí. Tato varianta mi přijde nejelegantnější a zároveň žáci mají možnost seznámení s dalším příkazem.



Obrázek 78 - Škodící postava a vyhodnocení prohry

b. *Čekej dokud nenastane životy = 0*: cyklus pro pohyb je v tomto případě nekonečný. U škodící postavy nebo u kosmonauta přidáme nové *Po startu (zelená vlaječka) čekej dokud nenastane životy = 0* a opět změníme pozadí, zajistíme, aby zvířátka a „škodič“ byli skrytí a kosmonaut řekl “Prohrál jsem”.

c. *Opakuj stále když životy = 0:* můžeme využít i nekonečný cyklus s podmínkou když.

8. Poznámka: Pokud jsme některé postavy v průběhu hry skrývali, musíme zajistit, aby ještě neproběhlo jejich skrytí a zobrazení v rámci podmínky s dotykem, i když už máme změněnou scénu. Možností je více, nejjednodušší však je po změně scény ještě chvíli počkat a až poté postavu definitivně skrýt.

9. Bonusové úkoly:

   a. Náhodnou rychlost zvířátek zajistíme pomocí *náhodného čísla* umístěného do prostoru určeného pro počet kroků, o které se má zvířátko posunout.

   b. Přidáme další předmět, který bude přidávat životy. Tento předmět je skrytý a zobrazí se vždy jen na krátkou chvíli na náhodném místě. Pokud jej kosmonaut stihne sebrat přičte se mu jeden život.

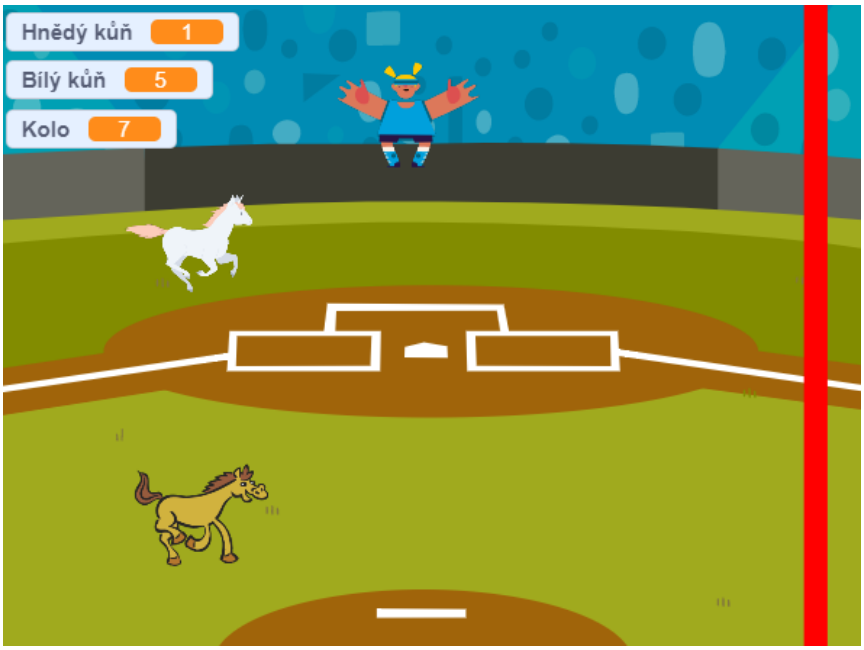
Možné problémy a komplikace

- Postavy se pohybují hlavou vzhůru - není nastaven způsob otáčení vlevo-vpravo.
- Body za chycení zvířátka jsme dostali vícekrát - tato situace není ošetřena například přemístěním zvířátka na náhodnou pozici po chycení nebo chvilkovým vyčkáním.
- Po změně scény zvířátka nebo zlá postava stále běhají - není nastaveno jejich skrytí po změně scény nebo se skrývají a následně ukazují i v průběhu chytání. Tedy se postava po chycení skryla, scéna se změnila, ale postavě uplynul čas pro ukázání. V tomto případě to musíme ošetřit například tím, že po změně scény dáme ještě chvíli prodlevu a až poté postavu skryjeme. (Z tohoto důvodu není doporučeno je skrývat v průběhu.)
- Postavy se po startu neukázaly - chybí příkaz *po startu ukaž se*.
- Program začal s pozadím "Město" nebo "Vesmírné město" - chybí příkaz *po startu změň pozadí na "Vesmír"*.
- Program přidal mnoho bodů/ubral životy hned po startu hry - chybí umístění postav po startu na náhodné pozice.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.

3.6.12 Úkol 12: Dostihy

Příběh hry

Na dostihovém závodišti se běží dostihy. Koně běží celkem 10 kol, který z nich vícekrát zaběhne kolo rychleji a stane se celkovým vítězem závodu?



Obrázek 79 - Dostihy náhled 1

Obrázek 80 - Dostihy náhled 2

Cíl hry

Žák se seznámí s relačním operátorem (je větší/menší), vymyslí logiku určení vítěze a procvičí si cyklus s daným počtem opakování.

Vstupní požadavky na žáka

- **Algoritmizace:** posloupnost příkazů, reakce na vstup uživatele, nekonečný cyklus, cyklus s daným počtem opakování, neúplná podmínka, proměnná, porovnání hodnot, čekání po daný čas, čekání s danou podmínkou
- **Scratch:** události klávesnice, událost po startu, práce s postavami a pozadím, práce s kostýmy, velikost postavy, opakuj stále, opakuj x krát, podmínka když, vyhodnocení dotyku, práce s proměnnou, skrývání a ukazování postav, náhodné číslo, pozicování, mluvící bublina

Nové dovednosti

- **Algoritmizace:** porovnání hodnot (je větší/menší)
- **Scratch:** operátor porovnání (<)

Zadání pro žáky v krocích

1. Vyber vhodné pozadí.
2. Přidej bílého a hnědého koně. Vhodně uprav jejich velikost a umísti je na vhodné startovní pozice.

   Nápověda: Koně by měli být podobně velcí a mít stejnou startovní pozici.
3. Zajisti, aby koně po stisku mezerníku vystartovali a přejeli scénu.

   Nápověda: Jejich rychlost pohybu by měla být náhodná, aby byl dostih napínavý.
4. Přidej cílovou čáru, u která určí vítězného koně daného kola dostihu.

   Nápověda: Poté co jí jeden z koní protne, čára zmizí, aby se bod započítal jen tomu prvnímu.

5. Přidej rozhodčího a vhodně uprav jeho velikost. Po ukončení desátého kola závodu řekne, který z koní je celkovým vítězem závodů.

   Nápověda: Dej si pozor na to, že desáté kolo je odběhnuté až poté co se jeden z koní dostane do cíle. Ošetři i případ remízy.

6. Zajisti, aby se koním pohybovaly nohy.
7. Bonusový úkol:
   a. Vylepši program poté co jeden z koní zvítězí, například zůstane jen vítězný kůň a druhý se skryje, fantazii se však meze nekladou.

Zadání pro žáky ve formě úkolu

- Dva koně běhají dostihy, ten který jako první protne cílovou čáru vyhrává dané kolo dostihu. Oba koně se rozbíhají po stisku mezerníku a jejich rychlosti se liší.
- Dostih má 10 kol, kůň který vyhraje více jednotlivých kol, se stane celkovým vítězem závodů.

  Nápověda: Dej si pozor na to, že desáté kolo je odběhnuté až poté co se jeden z koní dostane do cíle. Ošetři i případ remízy.

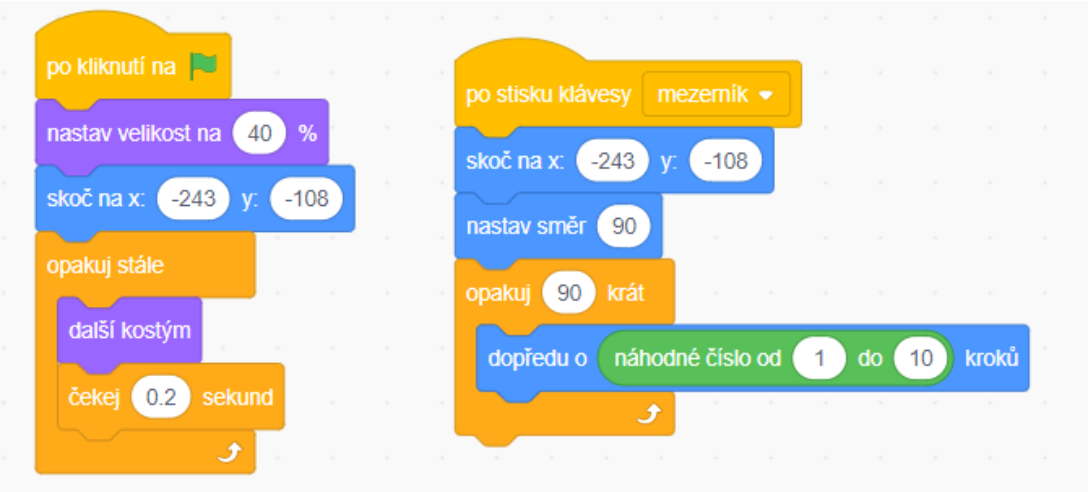
- O celkovém vítězi nás informuje rozhodčí.
- Bonusový úkol:
  - Vylepši program poté co jeden z koní zvítězí, například zůstane jen vítězný kůň a druhý se skryje, fantazii se však meze nekladou.

Program

https://scratch.mit.edu/projects/339967677/

Řešení a metodické poznámky

1. Vybereme vhodné pozadí.
2. Program koní:
    a. Je vhodné si koně pojmenovat (hnědý kůň, bílý kůň).
    b. Je důležité, aby koně měli stejnou výchozí pozici a stejnou velikost. (Poznámka: Střed každého koně je trochu jinde, proto se kůň musí vhodně napozicovat, aby ani jeden neměl náskok.)
    c. Vizualizace běhu (pohyb nohou): přes změnu kostýmů a krátké čekání. Musí se použít samostatný cyklus *opakuj stále*, pokud bychom tyto změny kostýmů vložili do cyklu pro běh, koně by to značně brzdilo a sekalo. (Využijeme příkazy po *stisknutí startu*, *opakuj stále*, *další kostým*, *čekej*.)
    d. *Po stisku mezerníku* se koně rozběhnou. Je důležité koně vždy po stisku mezerníku umístit na jejich startovní pozici (*skoč na pozici X Y*). Ošetříme i směr kterým se vydají (*nastav směr na 90*). Pro běh použijeme cyklus s daným počtem opakování (Poznámka: to kolikrát má cyklus proběhnout musí žáci vyladit, také to záleží na rozmezí počtu kroků.) Rozdílné rychlosti koní zajistíme tím, že se posouvají o *náhodný počet* kroků.



Obrázek 81 - Program koní

3. Program cílové čáry:

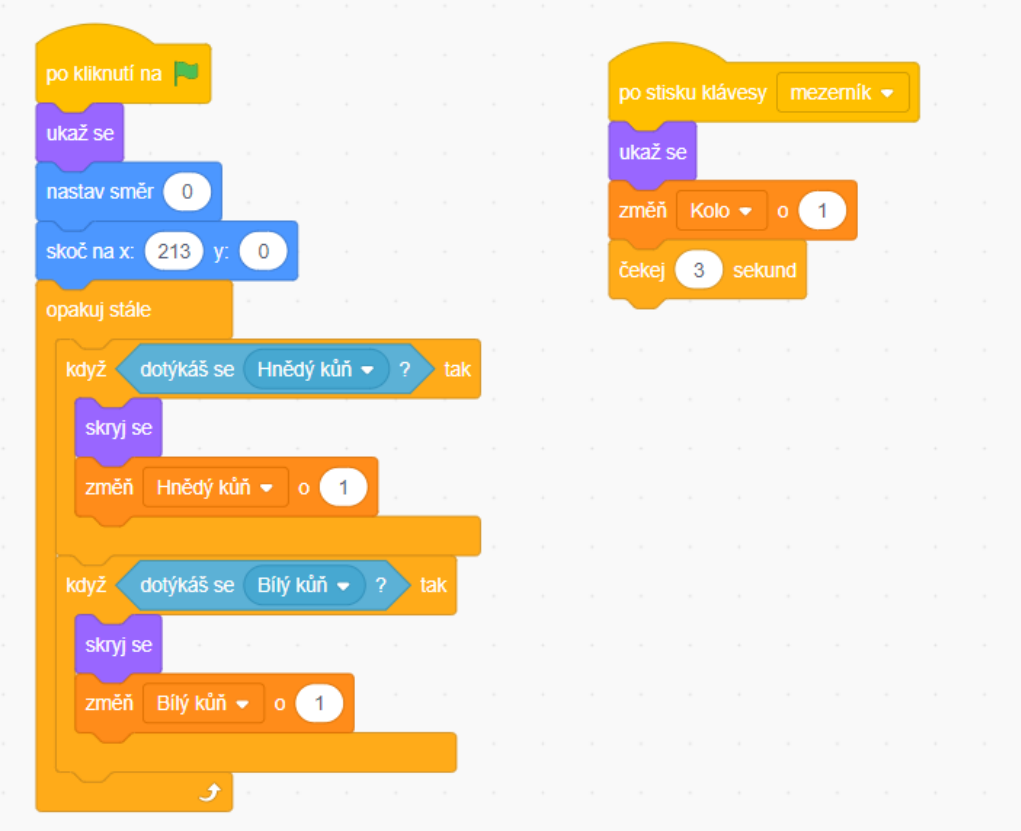
    a. Cílová čára slouží k vyhodnocení, který kůň byl v daném kole rychlejší.

    b. Umístíme ji na vhodnou pozici a *nastavíme ji směr 0*, aby byla svisle.

    c. Pro logiku dotyku koní musíme mít založené proměnné "Hnědý kůň" a "Bílý kůň". Kůň který se startovní čáry dotkne jako první, dostane bod a čára zmizí (*skryj se*), aby se ji nemohl dotknout i druhý kůň.

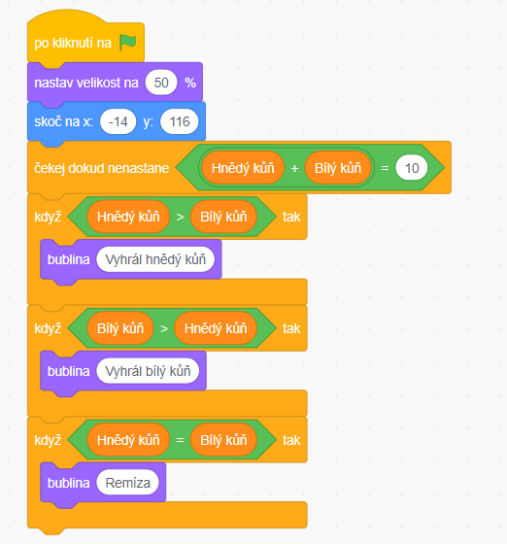
    d. Proměnná "kolo" není pro program v ničem důležitá, má pouze informativní charakter, kolik kol již koně odběhli. Tedy není nezbytně nutné ji zakládat.

    e. Po stisku mezerníku musíme čáru ukázat (protože po dotyku koní je skrytá) a můžeme změnit proměnnou kolo, není však nezbytné ji ve hře vůbec mít. Pokud tuto proměnnou měníme je vhodné přidat chvíli čekání, abychom nemohli během jednoho běhu koní namačkat kol několik.



Obrázek 82- Program cílové čáry

4. Program rozhodčího:
    a. Rozhodčí slouží k tomu, aby nám oznámil celkového vítěze.
    b. Umístíme jej na vhodnou pozici a upravíme jeho velikost.
    c. Rozhodčí *čeká dokud nenastane* situace, že koně doběhli do cíle celkem 10×, zde je více možností jak to zjistit. Musíme si dát pozor na to, že po startu desátého kola ještě chvíli trvá, než jej koně doběhnou.
        i. Elegantním řešením je použít operátor *součet* proměnných hnědý kůň a bílý kůň (*čekej dokud nenastane hnědý kůň + bílý kůň = 10*).
        ii. Méně elegantní variantou, ale pro žáky pravděpodobně více intuitivní je možnost využít proměnnou počet kol, zde však musíme ještě chvíli počkat, než koně proběhnou naposledy cílovou čáru. (*čekej dokud nenastane kolo = 10, čekej 3 sekund*)
    d. Poté co uplynulo desáté kolo pomocí podmínky určíme vítěze, zde používáme *operátory porovnání* (větší, menší, je rovno). Jedná se o matematiku, tedy můžeme využít tři samostatné podmínky *když s operátorem porovnání*. Čímž ošetříme případ vítězství hnědého koně, vítězství bílého koně a remízu. Rozhodčí zahlásí, kdo se stal celkovým vítězem.



Obrázek 83 - Program rozhodčího

5. Bonusový úkol:
    a. Poté co jeden z koní zvítězí můžeme dodat do programu nějaká vylepšení, například zůstane zobrazený jen vítězný kůň, to můžeme udělat přes *změnu pozadí* (každý kůň má své vítězné pozadí) nebo přes *vyslání zprávy*, s tím se však žáci zatím při svém programování nesetkali.

Možné problémy a komplikace

- Koně neběhají správným směrem - není nastaven směr 90.
- Čára není svisle - není nastaven směr 0.
- Koně nedoběhnou až k čáře - je nastaven malý počet opakování nebo je rychlost (počet kroků) koní příliš malá.
- Koně sice pohybují nohama, ale brzdí to jejich pohyb do cíle - pohyb nohama a pohyb do cíle musí být v oddělených cyklech.
- Postavy se neustále zmenšují nebo zvětšují - není upravena procentuální velikost, ale použit příkaz *změň velikost o* a kvůli tomu jsou postavy po každém startu větší nebo menší.
- Žádané proměnné nejsou v seznamu – proměnné nejsou deklarovány.
- Jeden z koní stále vyhrává - jeho startovní pozice je ve výhodě nebo není dobře nastavený počet kroků.
- Čára zmizí a už se neukáže - po stisku mezerníku chybí příkaz *ukaž se*.
- Oba koně protínají čáru a dostávají body - chybí skrytí čáry po protnutí koněm.
- Rozhodčí špatně nebo vůbec hlásí vítěze - chyba v podmínkách vítězství, tato logika je pro poměrně náročná, proto v ní žáci mohou snadno udělat chybu.

3.7 Reflexe úloh

Všech 12 úloh bylo otestováno v rámci zájmového kroužku, který navštěvují žáci 4.– 6. třídy. Jak již bylo popsáno výše, testovací skupina čítala 8 žáků. Ze začátku všichni dostávali zadání v krocích a zároveň jsem jim ukázala finální hru a doplnila ji o slovní popis příběhem. Postupně se ve skupině objevili žáci vhodní na vyzkoušení zadání pouze formou úkolu. Snažila jsem se do práce žáků zasahovat co nejméně, aby si na řešení přišli samostatně, nicméně občas bylo potřeba některým žákům trochu pomoci.

Úkol 1: Všudypřítomná kočička

Na úvod došlo k našemu seznámení v rámci skupiny, řekli jsme si něco o sobě a zároveň jsem zjistila, zda žáci mají s programováním nějaké zkušenosti, čtyři žáci se již s nějakým výukovým nástrojem na rozvoj algoritmického myšlení setkali a jeden žák znal přímo Scratch. Následně jsme společně založili uživatelské účty ve Scratchi.

Jedná se o úvodní lekci, kdy se žáci seznamují se Scratchem. Nejprve měli čas si prostředí prohlédnout, aby se v něm alespoň trošku zorientovali a následně dostali k řešení úkol.

Řešení úkolu se žákům dařilo bez větších problémů, někteří obzvláště z počátku bojovali se spojováním bloků, tento problém jsem zaznamenala zejména u žáků mladších (4. třída), starší žáci (6. třída) s tím neměli žádný problém. Obzvláště na začátku plnění úkolu jsem také zaznamenala drobný problém v orientaci, kde najít jaký příkaz, na základě čehož jsem toto zadání doplnila o nápovědu v podobě barev. Tato nápověda spočívá v obarvení částí zadání stejnými barvami jako mají příkazy ve Scratchi, které budou žáci používat, což jim hledání vhodných příkazů usnadní (viz Zadání pro žáky v krocích úloha 1: Všudypřítomná kočička). Další úlohy již takto obarvené nejsou, protože to by, dle mého názoru, vedlo k přílišné trivialitě a u dalších úloh už žáci s orientací, kde který příkaz najdou, problém neměli. Všichni žáci zvládli základní zadání a všichni se pustili i do plnění bonusových úkolů, někteří dokončili všechny.

Úkol 2: Denní a noční život

V druhé úloze se žáci seznámili s možností volby kostýmů a pozadí, což je velice zaujalo, někteří strávili poměrně dost času nad výběrem ideální postavy a scény pro svou hru, jiní se tím příliš nezaobírali, protože je mnohem více lákala tvorba kódu.

Drobná komplikace nastala u dvou žáků při programování pohybu postavy pomocí šipek, dokázali si spojit, že potřebují příkazy *po stisknutí klávesy a posun dopředu o kroky*, ale nedařilo se jim bez mé nápovědy přijít na to, že je nutné nastavit postavám směr, nicméně ostatní žáci to zvládli vymyslet samostatně, tedy jsem se rozhodla, že to v zadání takto doslovně napovídat nebudu. Také někteří žáci bojovali s rychlostí pohyby postav, ale rychle je napadlo, že tento problém vyřeší úprava počtu kroků.

Základní zadání zvládli všichni, pouze dva zmínění potřebovali radu ohledně ovládání postavy klávesami. Do bonusového úkolu se pustilo celkem pět žáků, z toho jeden žák dokonce stihl osm kombinací pozadí + postava.

Úkol 3: Ples

V této úloze se poprvé objevilo pozicování. Úloha byla pro žáky vcelku snadná. Vzhledem k tomu, že je v zadání se nachází rada, že novou pozici je nutné definovat příkazem *skoč na souřadnice x y*, tak to všichni zvládli, sami přišli na to, že čísla pod náhledem hry znamenají polohu postavy.

Jeden žák si musel osvěžit, jak zajistit pohyb postavy pomocí šipek nahlédnutím do předchozího úkolu s motýlem. Právě v tomto spatřuji výhodu vytvoření uživatelských účtů, žáci mají možnost kdykoliv a kdekoliv se podívat na své předchozí projekty.

Vždy když ve Scratchi dojde na zvuky, tak to budí u žáků ohromné nadšení, ani zde tomu nebylo jinak.

Jak jsem zmínila výše, tato úloha nebyla nikterak komplikovaná a všichni žáci měli krom základního zadání minimálně o jeden hudební nástroj a o jeden pár tanečníků navíc.

Úkol 4: Módní salón

V této úloze jde hlavně o pozicování předmětů. Zde jsem poprvé zkusila jednomu žákovi dát zadání pouze formou úkolu (zbytek měl zadání v krocích) a tento žák se práce zhostil výborně.

Jediný problém, který se zde vyskytl byl, že ne vždy postavě oblečení sedělo. Což bylo zapříčiněno tím, že žák neupravil velikost oblečení stejně, jako velikost postavy, nicméně sám přišel na to, co je potřeba udělat, aby nastalý problém vyřešil. Tento bod (aby byla velikost postavy i oblečení stejná) jsem doplnila i do zadání, jako nápovědu.

Základních pět kusů oblečení zvládli všichni. Někteří žáci měli opravdu rozsáhlou nabídku oblečení, které postavě mohli obléknout..

Úkol 5: Upovídaná čarodějnice

V této úloze se žáci poprvé setkali s cykly a podmínkami, proto jsem u všech zvolila zadání v krocích, abych jim toto první seznámení příliš nekomplikovala. Interakce postav byla pro žáky velice efektní.

Všichni samostatně přišli na to, že potřebují, aby program stále vyhodnocoval, zda se postavy dotýkají. Ne všichni žáci dbali na doporučení, že by bylo vhodné si postavy přejmenovat, domnívám se, že hlavním důvodem byla jejich lenost a pak bylo znát, že se jim podmínky pro dotyk vymýšlí trochu hůře, než těm, kteří si postavy na základě doporučení uvedeného v zadání přejmenovali.

Opět se zde zejména u mladších žáků (stejných jako při úkolu 1) objevil malý problém při spojování *opakuj, když a co se má stát*, přeci jen je to komplikovanější než skládat příkazy za sebe, ale po drobných komplikacích to zvládli.

Při zadávání této úlohy potřeba velmi dobře promyslet, zda žákům dovolíme zvuky. Já jsem se rozhodla jim udělat radost a využití zvuků jim povolila. Nicméně musím říct, že toho nijak nezneužívali. Už se mi stalo (dříve v jiné skupině), že zvuky žáci pojali jako formu narušování hodiny a vyrušování ostatních.

Základní zadání zvládli všichni žáci, někteří stihli pouze přidání dalšího zvířete a do pohyblivých zvířat se úspěšně pustili tři žáci.

Úkol 6: Žralok

Tuto úlohu opět nepovažuji za náročnou a zároveň s ní nepřichází příliš nových prvků, proto jsem tentokrát již u tří žáků zvolila zadání formou úkolu. Dva to zvládli výborně, u třetího jsem nakonec poskytla i zadání v krocích, protože zřejmě ještě na náročnější formu zadání nebyl zcela připraven a zbytečně se trápil.

Jinak tato úloha žákům nečinila problémy, jen je důležité zde pomyslet na krátké vyčkání, než se ryba skryje, aby i žralok stihl vyhodnotit společnou interakci, což nemusí být příliš intuitivní, proto je v zadání uvedena nápověda.

Snad v každé skupině se objeví někdo, kdo po skrytí postavy zapomene, že jí po startu musí zase ukázat pro případ, že bychom hru vypnuli zrovna ve stavu, kdy je postava skrytá. Ani v této skupině tomu nebylo jinak, tuto drobnou chybu udělali celkem tři žáci.

Základní zadání zvládli všichni žáci a zároveň se všichni pustili do vylepšování hry dle bonusových úkolů.

Úkol 7: Poslušný pejsek

Zde se žáci poprvé setkali s cyklem s daným počtem opakování a s podmínkou if/else. Zadání ve formě úkolu dostali dva žáci, zbytek měl zadání formou postupu po krocích.

Zde nám nastaly celkem dva problémy, někteří žáci trochu bojovali s vyladěním počtu opakování a počtu kroků při posouvání míče, zde je potřeba si počet kroků a opakování otestovat a upravit. Druhým problémem bylo, že poměrně dost žáků (čtyři) od posledního úkolu se žralokem zapomněli na záležitost, s krátkým čekáním, než se míč skryje, tedy jeho chycení pes občas nestihl vyhodnotit a zareagovat na chycení míče.

V zadání je uvedeno, že při chycení míče má pes za pomoci bubliny říct „Haf“. Ze strany žáků se objevil požadavek, zda by pes místo bubliny nemohl vydat zvuk.

Tentokrát úplně všichni žáci stihli bonusový úkol, jeden žák si dokonce vymyslel i svůj vlastní bonusový úkol v podobě dalšího zvířete pohybujícího se na klávesy WASD.

Úkol 8: Predátor a kořist

Tato hra je zejména o kombinacích pozadí, predátora a kořisti. Má i jistý přesah do přírodovědy, kdy považuji za vhodné po žácích požadovat smysluplné kombinace (pozadí, predátor, kořist). Opět jsem zkusila dát zadání formou úkolu třem žákům (stejným jako u úkolu 6 se žralokem) a tentokrát i třetí žák neměl problém úkol zvládnout.

Podstatné bylo vhodně vyladit velikost postav a rychlost jejich pohybu, pokud žáci měli rychlosti v nepoměru nebo postavy příliš velké, tak jim scény přeskakovaly příliš rychle.

Nicméně tato úloha žákům nečinila žádné problémy a všichni ji zvládli bez komplikací. Všichni žáci nakonec vytvořili více než pět kombinací (pozadí, predátor, kořist).

Úkol 9: Opice

V této úloze se žáci poprvé setkávají s proměnnou, z toho důvodu jsem zvolila zadání ve formě kroků u všech žáků, aby pro ně bylo snazší se s touto problematikou seznámit a pochopit ji.

Ve vzorovém řešení úkolu jídlo na pozicích příliš dlouho nečeká a rychle skáče jinam, což by se mohlo zdát jako chyba v programu, ale není tomu tak. Pokud by jídlo dlouho čekalo na jedné pozici, tak by hra byla příliš snadná.

Zvládnout problematiku proměnných nebyl pro žáky žádný problém. Všichni pomocí dostupné nápovědy v zadání proměnnou založili i bez jakékoliv pomoci přišli na to, že ji musí do programu zavést a také ukázat. Její následné použití při interakci opice s jídlem pro ně bylo zcela automatické.

U této hry všichni žáci zvládli i bonusové úkoly, a ještě nám zbyl čas na to, abychom udělali krátkou soutěž v tom, kdo nasbírá během časového úseku více bodů.

Úkol 10: Balóny

V této úloze je zavedena jiná možnost samovolného pohybu objektů než klouzáním, také se zde žáci seznámili s náhodným číslem. Zadání formou úkolu dostali opět tři žáci.

Nápady, jak by se dal zajistit samovolný pohyb jinak, než klouzáním měli všichni, až na jednoho žáka, který si urputně stál za tím, že se mu klouzání líbí a že žádnou novou metodu zkoušet nechce. Setkala jsem se s problémem, že náhodný směr žák vložil přímo do cyklu, poté balón létal hodně zvláštně. Zároveň u dvou žáků nastala situace, že zapomněli na odrážení od okraje.

Přičítání bodů za chycení balónů a jejich zmenšování nebyl sebemenší problém. Tento úkol nebyl pro žáky časově příliš náročný a úplně všichni zvládli vyřešit i bonusový úkol.

Úkol 11: Vesmírná mise

V této úloze se žáci poprvé seznamují s řešením výhry a prohry. Řešení je cíleno přes změnu pozadí, protože je to méně abstraktní než rozesílání zpráv, se kterým se zároveň žáci zatím nesetkali. Tentokrát zadání formou úkolu dostali jen dva žáci z důvodu, že třetí žák, který toto zadání i v minulých úlohách dostával, nebyl na této lekci přítomen.

Vyřešit pohyb zvířátek bylo možné dvěma způsoby, ač byl doporučený stejný způsob jako s balónky, taky si dva žáci zvolili klouzání, což jim později mírně zkomplikovalo situaci při přičítání bodů za chycení postavy (body se jim přičítali obrovskou rychlostí, než přišli na to, jak problém vyřešit).

Přičítání bodů a vyřešení výhry nečinilo žákům žádné závažné problémy. Některé trošku potrápila postava „škodiče“ a prohra, z toho důvodu jsem do zadání přidala další doporučení, že by žáky mohl zajímat příkaz *opakuj dokud nenastane*. Samozřejmě jak je popsáno v řešení, je možné problematiku prohry realizovat minimálně třemi způsoby, nicméně mým cílem je navést žáky na ten, který mi přijde nejlepší. Poměrně obvyklým nápadem bylo využití *čekej dokud nenastane* stejně jako v případě výhry. Nicméně celkem dva žáci z pěti, kteří toto řešení chtěli použít narazili na špatné zapojení příkazu, umístili jej za stejné *po startu*, jako čekání na výhru, tedy se na něj nikdy nedostalo. Po mé drobné nápovědě, přišli na to, v čem je problém.

Další komplikace, která se u úlohy objevila byla viditelnost postav vesmírných zvířátek po výhře či prohře (v této fázi hry už by všechny postavy, kromě kosmonauta, měly být skryté). Tento problém nastal z důvodu, že po chycení vesmírného zvířátka, se někteří žáci rozhodli zvířátko na chvíli skrýt a poté ukázat. Pokud v tom momentě došlo k ukončení hry (dosažení potřebného počtu bodů nebo ztráta posledního života), změnilo se pozadí a všem postavám zvířátek bylo naprogramováno, že se mají skrýt, nicméně zároveň u nich běžel proces *skryj se, čekej, ukaž se* (který měli naprogramovaný po dotyku s kosmonautem). Všichni žáci, které tento problém potkal, jej vyřešili samostatně. Někteří zrušili skrývání vesmírných zvířátek po chycení kosmonautem a jiní zařadili po změně pozadí ještě krátké čekání a až poté zvířátka skryli.

I zde jsme se neobešli bez komplikace, že po spuštění hry se postavy neukáží, protože konkrétně jeden žák zapomněl na to, že když definitivně ve hře něco skryje, musí to po startu zase zobrazit.

Základní zadání zvládli všichni žáci, ale k řešení bonusového úkolu s prvkem, který přidává životy se dostali jen dva. U některých bylo důvodem to, že sotva stihli základní zadání a u jiných, že raději dali přednost tomu hru hrát, než programovat dál. Jejich rozhodnutí nepovažuji za špatné.

Úkol 12: Dostihy

Poslední úkol sbírky je zaměřen na matematické porovnávání dvou hodnot. Zde jsem se rozhodla dát všem žákům zadání v krocích, protože úlohu považuji za poměrně dost náročnou. Ukázalo se, že tato volba byla správná a ani pro tři nejšikovnější žáky řešení nebylo úplně jednoduché.

Samotný pohyb koní a přičítání bodů za protnutí cílové čáry nečinilo žákům větší problémy. Nejkomplikovanější částí byla logika rozhodčího, která zejména pro dva mladší žáky byla náročná, tito dva žáci potřebovali mou pomoc. Ostatní logiku rozhodčího zvládli vyřešit samostatně.

Původně jsem měla jako bonusový úkol třetího koně (nebo jiné zvíře), to se však ukázalo jako extrémně těžké a na řešení logiky, který z koní vyhrál žádný z žáků nedokázal přijít samostatně. Se třemi žáky jsme logiku určení vítěze vyřešili společnými silami, další dva řešení třetího koně po chvíli vzdali. Z toho důvodu jsem se rozhodla tento bonusový úkol zrušit a nahradit jej jednodušším, který dva žáci i vyzkoušeli, když jsem jim ho v hodině navrhla.

3.8 Zhodnocení ověřování úloh

Na základě pozorování úlohy zhodnotím v následujících aspektech: orientace v prostředí Scratch, samostatná práce žáků na úlohách, práce se zadáním a časová náročnost úloh.

Orientace v prostředí Scratch

V první lekci (Úloha číslo 1: Všudypřítomná kočička) se žáci seznamovali s programem Scratch a mimo algoritmické struktury posloupnosti příkazů se museli seznámit i s nástroji typickými pro dětský programovací jazyk Scratch. V daší úloze (Úloha číslo 2: Denní a noční život) již většině žáků orientace v nástrojích Scratche nečinila problém.

Tedy k orientaci v prostředí dětského programovacího jazyka Scratch většině žáků stačila jedna lekce. Ve třetí lekci (Úloha číslo 3: Ples) již byli plně zorientovaní všichni žáci.

Samostatná práce žáků na úlohách

Žáci pracovali na úlohách samostatně. Při jejich práci jsem jim byla k dispozici pro řešení komplikací, které jim nastanou, případně poskytnutí nápovědy.

U většiny úloh žáci nepotřebovali pomoci. Větší potřebu pomoci či nápovědy jsem pozorovala u žáků mladších, tedy navštěvujících 4. třídu, oproti tomu žáci starší, navštěvující 6. třídu, nepotřebovali v rámci základního zadání pomoc vůbec žádnou. Žáci navštěvující 5. třídu se množstvím otázek a samostatností řešení úloh přibližovali spíše žákům z 6. třídy.

Nejvíce otázek ze strany žáků se objevilo u úlohy číslo 5: Upovídaná čarodějnice, v této úloze se žáci seznamují s cykly. Další náročnou úlohou, opět hlavně pro mladší žáky, byla úloha číslo 12: Dostihy, která vyžaduje vhodné použití porovnání dvou hodnot.

Práce se zadáním

Primárně jsem využívala zadání v krocích, zejména u úloh, kdy docházelo k seznamování s novými algoritmickými strukturami (například zavedení cyklů).

V rámci pozorování práce žáků bylo po několika lekcích patrné, že někteří žáci jsou vhodnými adepty pro zadání formou úkolu. U jednoho žáka jsem toto zadání použila již u úlohy číslo 4: Módní salón. Nejvyšší počet žáků, kteří dostali toto složitější zadání formou

úkolu, byli tři žáci (2× 6. třída, 1× 5. třída). Žáci navštěvující 4. třídu vždy dostávali zadání v krocích.

Časová náročnost úloh

Vzhledem k časové dotaci kroužku, která činí 90 minut, po odečtení úvodní agendy a krátké přestávky v polovině kroužku je k dispozici 75 minut čistého času. Tato časová dotace je na vytvořené úlohy zcela dostačující. Průměrná doba potřebná k vypracování základního zadání činí 60 minut.

Všichni žáci v dostupném čase zvládli vyřešit základní zadání úloh a vždy minimálně polovina řešila i bonusové úkoly, některým žákům zbyl i čas na vymýšlení vlastních vylepšení her. Zde je však nutné přihlédnout ke dvou aspektům: 1. velikost skupiny (takto malá skupina umožňuje rychlejší práci, protože učitel není příliš vytížen a žáci na konzultaci či kontrolu úlohy nemusí dlouho čekat) 2. složení skupiny (skupina byla složena se žáků, kteří se do kroužku dobrovolně přihlásili, pravděpodobně z důvodu, že je jim problematika programování blízká).

Druh zadání, které žáci měli k dispozici neměl na rychlost řešení úloh vliv. Žáky však v rychlosti řešení ovlivňoval věk, zatímco žáci navštěvující 6. třídu byli hotovi mezi prvními, i když u některých úloh dostávali zadání formou úkolu, žákům navštěvujícím 4. třídu řešení úloh trvalo déle. Žáci navštěvující 5. třídu byli v rychlosti řešení úloh průměrní.

Nejvíce časově náročné úlohy byly Úloha číslo 2: Denní a noční život, Úloha číslo 11: Vesmírná mise a Úloha číslo 12: Dostihy, jejichž řešení základního zadání trvalo žákům průměrně cca 70 minut. Nejméně časově náročné úlohy byly Úloha číslo 3: Ples a Úloha číslo 10: Balóny, jejichž řešení základního zadání trvalo žákům průměrně cca 45 minut.

U ostatních úloh řešení základního zadání žákům průměrně trvalo 55–60 minut. Velký přínos jsem pozorovala právě v bonusových úkolech, kdy žáci, kteří byli hotoví dříve mohli řešit další úkoly v rámci dané úlohy. Nebránila jsem však žákům ani v tom, vymyslet si svoje vlastní vylepšení.

Závěr

Rozvoji programování a algoritmického myšlení není v českém školství přikládán příliš velký význam. Existuje mnoho nástrojů a možností jakým způsobem u žáků rozvíjet algoritmické myšlení. I na základní škole je možné využít takzvané profesionální programátorské jazyky, ne vždy jsou na to však žáci dostatečně vyspělí nebo motivovaní a nemusí to pro ně být zcela ideální přístup, jak se s programováním a algoritmickým myšlením seznámit. Z toho důvodu je k dispozici stále více nástrojů určených přímo pro využití ve výuce. Tyto nástroje učiteli i žákům usnadňují rozvoj algoritmického myšlení a výuku programování. Je možné využít robotické programovatelné hračky, robotické stavebnice, předpřipravené hry nebo dětské programovací jazyky, které mají z výše jmenovaných kategorií nejméně omezení a umožňují žákovi největší možnosti vlastní tvorby.

Cílem práce bylo sestavit ucelenou sbírku úloh pro rozvoj algoritmického myšlení žáků na základní škole. Jako nástroj pro realizaci byl zvolen dětský programovací jazyk Scratch (https://scratch.mit.edu/). A vznikla sada úloh (http://scratch.milichovska.cz/) určená pro výuku programování a rozvoj algoritmického myšlení. Cílovou věkovou skupinou je 4. –6. třída základní školy.

Sbírka obsahuje celkem 12 úkolů, jejichž obtížnost postupně graduje. V prvních několika úlohách se žáci seznamují s posloupnostmi příkazů a také funkcionalitou a nástroji samotného dětského programovacího jazyka Scratch. Postupně jsou přidávány cykly a podmínky a na závěr jsou žáci seznámeni s proměnnými a jejich využitím. Všechny úlohy jsou koncipovány tak, aby žáci byli schopni samostatného řešení. Také všechny úlohy mají charakter hry.

Každá úloha obsahuje dvě verze zadání, které může učitel žákům poskytnout. První z nich je zadání v krocích, které odpovídá pracovnímu postupu, neznamená to však, že je žákům krok za krokem nadiktováno, jaký příkaz mají kam přesunout. Druhá varianta zadání je formou úkolu, jedná se o zběžné popsání úkolu, kde žák musí zapojit mnohem více vlastní invence. Obě zadání mimo hlavních úkolů obsahují i takzvané bonusové úkoly určené pro rychlejší žáky, aby se nenudili. Zároveň všechny úlohy obsahují přesný postup řešení,

který je určen pro učitele a také sadu možných problémů a komplikací, které mohou u žáků při řešení úloh nastat, což usnadní učiteli jejich dohledání a vyřešení.

Všechny úlohy byly ověřeny se skupinou žáků navštěvujícími zájmový kroužek programování. Žáci řešení úloh zvládli ve většině případů zcela samostatně, někteří maximálně s drobnou dopomocí. Vždy minimálně polovina žáků stihla vyřešit i bonusové úkoly.

Během ověřování bylo patrné, že se žáci v řešení úloh zlepšují. Vzhledem ke gradující obtížnosti úloh musí žáci využívat již získané znalosti a dovednosti a zároveň nabalovat nové. Při předložení úloh žákům bylo dbáno na to, aby na jejich řešení pracovali samostatně s minimální mírou pomoci.

Na základě pozorování žáků řešících úlohy jsem dospěla k závěru, že na rychlost a samostatnost řešení má hlavní vliv věk žáků. Starší žáci (6. třída) byli vždy rychlejší a dokázali pracovat zcela samostatně, oproti tomu mladší žáci (4. třída) pracovali pomaleji a častěji potřebovali konzultaci.

Žáci se ve vybraném dětském programovacím jazyce Scratch zvládli zcela zorientovat během dvou lekcí a byli schopni úlohy vypracovat samostatně v dostupném čase. Vzhledem k tomu lze považovat vybraný nástroj a navržený soubor úloh za adekvátní kombinaci pro danou věkovou skupinu.

Seznam použitých informačních zdrojů

BRDIČKA, Bořivoj (1995). *Učení s počítačem: Vyhodnocování výukových programů* [online]. In. 1995 [cit. 2020- 01-16]. Dostupné z: http://it.pedf.cuni.cz/~bobr/ucspoc/

ČERVENÝ, Martin (2016). *Průvodce hlavními funkcemi programu Stencyl*, 2016[online]. Plzeň, 2016 [cit. 2020-01-16]. Dostupné z: https://dspace5.zcu.cz/handle/11025/24346. Bakalářská práce. ZÁPADOČESKÁ UNIVERZITA V PLZNI. Vedoucí práce Mgr. Tomáš Přibáň, Ph.D.

FUTSCHEK, Gerald (2006). *Algorithmic Thinking: The Key for Understanding Computer Science*. In. Lecture Notes in Computer Science 4226, Springer

HORVÁTHOVÁ, Dana a Jaroslav KNĚŽNÍK (2019). *Blokové programovanie v rámci informatického krúžku*. In: DidInfo 2019. Banská Bystrica, 2019

ISTE a CSTA (2011), *Operational Definition of Computational Thinking for K–12 Education* [online]. In. 2011, [cit. 2020-01-17]. Dostupné z: http://www.iste.org/docs/ct-documents/computationalthinking-operational-definition-flyer.pdf?sfvrsn=2

KALAŠ, Ivan (2017). *ScratchMaths: vzdelávací obsah a princípy tvorby*. In: DidInfo&DidactIG 2017. Banska Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko

KANÁLIOVÁ, Alžběta. (2015). *Programovanie, alebo hra?*. In: DidInfo 2015. Banská Bystrica: Univerzita Mateja Bela, Fakulta prírodných vied v Banskej Bystrici, Slovensko, 2015

KLEMENT, Milan (2002). Základy programování v jazyce Visual Basic. 1. vyd. Olomouc: Univerzita Palackého, 2002.

KLEMENT, Milan (2005). *Možnosti evaluace výukových programů*. In: Trendy technického vzdělávání 2005 [online]. 2005, [cit. 2020-01-16]. Dostupné z: https://www.researchgate.net/publication/292983376_MOZNOSTI_EVALUACE_VYUKOVYCH_PRO GRAMU

KLIMOVÁ, Nika a Dana HORVÁTHOVÁ (2017). *Súťaž KODU CUP ako podporný prostriedok algoritmického myslenia*. In: DidInfo&DidactIG 2017. Banska Bystrica

KOZÁK, Petr (2015). *Nástroje pro výuku programování* [online]. In: 2015, 25. 05. 2015 [cit. 2019-11-8]. Dostupné z: https://spomocnik.rvp.cz/clanek/20019/NASTROJE-PRO-VYUKU-PROGRAMOVANI.html

KREJSA, Jan (2014). *Výuka základů programování v prostředí Scratch.* České Budějovice, 2014. Diplomová práce. Jihočeská univerzita v Českých Budějovicích. Vedoucí práce Doc. PaedDr. Jiří Vaníček, Ph. D.

KUNCOVÁ, Lucie (2019). *V hlavní roli kyslík: návrh a ověření badalské aktivity*. Praha, 2019. Diplomová práce. Univerzita Karlova, Pedagogická fakulta, Katedra chemie a didaktiky chemie. Vedoucí práce Rusek, Martin.

KVAPILOVÁ, Kateřina (2016). Lego výuka děti baví [online]. In. 2016 [cit. 2020-01-14]. Dostupné z: https://www.veskole.cz/clanky/lego-vyuka-deti-bavi

LE, Dustin. Edudemic: The Three Best Free Coding Websites for Kids [online]. In. 2015 [cit. 2020-01-16]. Dostupné z: http://www.edudemic.com/the-three-best-free-coding-websites-for-kids/

LESSNER, Daniel (2014). Code Week: Další kampaň na podporu informatiky [online]. In: 2014 [cit. 2020-01-19]. Dostupné z: http://ucime-informatiku.blogspot.com/2014/10/code-week-dalsi-kampan-na-podporu.html

LESSNER, Daniel. (2015). How Do We Translate Computional Tjinking Into Czech?: Jak si přeložíme „Computional thinking?“ [online]. In. 2015, 10 [cit. 2020-01-17]. Dostupné z: http://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf

MANĚNOVÁ, Martina (2012). Vliv ICT na práci učitele 1. stupně základních škol. Praha: ExtraSystem, 2012. ISBN 978-80-87570-09-8.

MŠMT (2016). *Stovky českých škol zařadily do své výuky programování*, Tisková zpráva [online]. In: 2016 [cit. 2020-01-19]. Dostupné z: http://www.msmt.cz/ministerstvo/novinar/stovky-ceskych-skol-zaradily-do-vyuky-programovani

MUSÍLEK, Michal (2012). *Dětské programovací jazyky* [online]. In. 2012. Hradec Králové [cit. 2019-11-13]. Dostupné z: https://docplayer.cz/41023168-2-programovaci-jazyk-logo.html#show_full_text. 2012

MUSÍLEK, Michal (2013). *PROJEKT SCRATCH*. Journal of Technology and Information Education, 5(1), 102-106. doi: 10.5507/jtie.2013.015.

Národní ústav pro vzdělávání (2018). *Návrh revizí rámcových vzdělávacích programů v oblasti informatiky a informačních a komunikačních technologií*. [online]. In. Praha: NÚV, 2018. [cit. 2020-01-19]. Dostupné z: http://www.nuv.cz/file/3362_1_1/

NEUMAJER, Ondřej (2017). *Být digitálně gramotný už neznamená jen ovládat počítač* [online]. In. 2017, 20.3.2017 [cit. 2019-10-12]. Dostupné z: https://spomocnik.rvp.cz/clanek/21311/BYT-DIGITALNE-GRAMOTNY-UZ-NEZNAMENA-JEN-OVLADAT-POCITAC.html

PECINOVSKÝ, Rudolf (2001). *Zásady správné výuky programování*. Česká škola [online]. In. 2001 [cit. 2020-01-14]. Dostupné z: http://www.ceskaskola.cz/2001/09/rudolf-pecinovsky-zasady-spravne-vyuky.html

PIAGET, Jean a Bärbel INHELDER (2014). Psychologie dítěte. Praha: Portál, 2014. Klasici. ISBN 978-80-262-0691-0.

PITNER, Tomáš. (2000). *Výuka programování na základní a střední škole*. [online]. In. 2000, [cit. 2019-11-05]. Dostupný z: http://www.fi.muni.cz/~tomp/semuc/text_pitner.html

RVP ZV (2017). *Rámcový vzdělávací program pro základní vzdělávání*. Praha: MŠMT, 2017, ročník 2017, 82/2015 Sb. Dostupné z: http://www.msmt.cz/file/43792/

SMÉKAL, Vladimír (2006). *Individuální přístup jako podmínka kvality života žáků*. In: 2. konference ŠKOLA a ZDRAVÍ 21 [online]. Brno, 2006 [cit. 2020-01-14]. Dostupné z: http://www.ped.muni.cz/z21/puv/sbornik_06/pdf/032.pdf

SVOBODA, Milan (2018). *Rozvoj algoritmického myšlení žáků ZŠ ve výuce informaticky zaměřených předmětů s využitím Scratch*. Praha, 2018. Diplomová práce. Pedagogická fakulta Univerzita Karlova. Vedoucí práce RNDr. Miroslava Černochová, CSc.

TOCHÁČEK, Daniel a Jakub LAPEŠ (2012). *Edukační robotika*. Praha: Univerzita Karlova, Pedagogická fakulta, 2012. ISBN 978-80-7290-577-5.

VANÍČEK, Jiří (2004). Kritéria evaluace výukových programů pro vyučování matematiky pomocí počítače [online]. In. Jihočeská univerzita, Pedagogická fakulta, 2004 [cit. 2020-01-16]. Dostupné z: http://amper.ped.muni.cz/~svobodka/useful/kriteria_sw.pdf

VANÍČEK, Jiří. (2016). Výuka algoritmizace patří především do informatiky. In: Počítač ve škole 2016 [online].2016 [cit. 2020-01-18]. Dostupné z: https://docplayer.cz/17569962-Vyuka-algoritmizace-patri-predevsim-do-informatiky.html

VAŇKOVÁ, Petra a Lenka PÍTROVÁ. *Robotické programovatelné hračky ve výuce*. In: Počítač ve škole 2018–celostátní konference učitelů základních a středních škol [online]. In. Nové Město na Moravě, 2018 [cit. 2020-01-14]. Dostupné z: https://docplayer.cz/105870436-Roboticke-programovatelne-hracky-ve-vyuce.html

WING, Jeannette (2010). *Computational Thinking: What and Why?* [online]. In: 2010 [cit. 2019-10-09]. Dostupné z: https://www.cs.cmu.edu/~CompThink/papers/Th eLinkWing.pdf

ZOUNEK, J., ŠEDOVÁ, K. 2009. Učitelé a technologie. Mezi tradičním a moderním pojetím. Brno : Paido, 2009. ISBN 978-80-7315-187-4.

Seznam obrázků