Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Rozvoj algoritmického myšlení žáků v prostředí Scratch

Developing students' algorithmic thinking with Scratch

Kateřina Šumová

Vedoucí práce:	PhDr. Jiří Štípek, Ph.D.
Studijní program:	Specializace v pedagogice (B7507)
Studijní obor:	Informační technologie se zaměřením na vzdělávání (7507R040)

2022

Odevzdáním této bakalářské práce na téma Rozvoj algoritmického myšlení žáků v prostředí Scratch 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.

Malotice 1.7.2022

Poděkování

Mé díky patří vedoucímu bakalářské práce PhDr. Jiřímu Štípkovi, Ph.D. za jeho rady, ochotu a trpělivost při vedení této práce. Ráda bych také poděkovala vedení základní školy Šolleho v Kouřimi za možnost ověřit si navržený soubor lekcí na jejich škole. Také děkuji Mgr. Šárce Krombholzové za konzultace a asistenci s vedením lekcí ve třídě 6.A. V neposlední řadě patří můj dík i celé mé rodině a přátelům za podporu ve studiu.

Abstrakt

Bakalářská práce se zabývá tématem rozvoje algoritmického myšlení žáků v prostředí programu Scratch. V teoretické části vysvětluje pojmy informatické a algoritmické myšlení, které v současné společnosti nabývají stále větší důležitosti. Společně s těmito pojmy práce seznámí čtenáře i s jejich částmi, ze kterých se skládají.

Dále je předmětem této práce nalézt aplikaci s úkolově orientovanými aktivitami, která bude sloužit jako příprava na programování v programu Scratch. Aplikace jsou bodově ohodnoceny podle blíže specifikovaných hledisek. Uživatelé díky nim mohou získat základní představu o algoritmických strukturách.

Praktická část práce se zabývá vytvořením souboru na sebe navazujících lekcí v programu Scratch. V tomto souboru lekcí budou prohloubeny žákovské znalosti o algoritmických strukturách společně s jejich praktickým použitím díky programování ve vizuálním programovacím jazyku Scratch.

Soubor lekcí byl ověřen na základní škole ve Středočeském kraji. Soubor obsahuje celkem devět lekcí.

Bakalářská práce také obsahuje výstup ověřování souboru lekcí, který může být inspirací pro učitele, kteří by rádi pomohli rozvoji algoritmického myšlení u svých žáků.

Klíčová slova

Algoritmické myšlení, algoritmické struktury, dětské programovací jazyky, úkolově orientovaná aplikace napomáhající rozvoji algoritmického myšlení, blokové programování

Abstract

The bachelor thesis deals with the development of algorithmic thinking of students in the environment of Scratch. In the theoretical part it explains the concepts of computer and algorithmic thinking, which are becoming more and more important in contemporary society. Together with these concepts, the work introduces the reader to the parts of which they consist.

Furthermore, the object of this thesis is to find an application with task-oriented activities that will serve as a preparation for programming in Scratch. The applications are scored according to specified aspects. Through them, users can get a basic idea of algorithmic structures.

The practical part of the thesis deals with the creation of a set of successive lessons in Scratch. In this set of lessons, students' knowledge of algorithmic structures will be deepened along with their practical application through programming in the visual programming language Scratch.

The set of lessons has been tested in an elementary school in the Central Bohemia region. The set contains a total of nine lessons.

The bachelor thesis also contains the output of the validation of the set of lessons, which can be an inspiration for teachers who would like to help the development of algorithmic thinking in their students.

Keywords

Algorithmic thinking, algorithmic structures, children's programming languages, task-oriented applications helping to develop algorithmic thinking, block programming

Obsah

Část I

Teoretická část

1. Úvod

Současnost by se také dala nazvat věkem informace, stále více si uvědomujeme, že žijeme v informační společnosti. Každým dnem ve zprávách čteme o nových technologiích a staré technologie se modernizují, téměř ve většině případů pomocí počítačů a počítačových programů. Generaci, která v tomto prostředí vyrůstá již nestačí naučit pasivně ovládat již vytvořené programy, na pracovním trhu není dostačující to co platilo léta – ovládat kancelářské programy jako je Excel a Word. Aby byla i v budoucnosti Česká republika konkurenceschopná co se týče moderních technologií, je nutné aby již na základní škole byly probírány základy programování a základní algoritmické struktury. Důležitost ovládat informační a komunikační technologie na vyšší úrovni než pouhý uživatel zdůrazňuje i RVP (rámcový vzdělávací program). Právě v něm se dozvíme o nutnosti seznámení žáků s pojmy jako je digitální gramotnost a informatické myšlení. Ačkoliv nevíme, jak bude budoucnost vypadat, je možné se domnívat, že tyto schopnosti budou mít ve vzdělávání stejnou důležitost jako znalost anglického jazyka.

Cílem této práce je příprava souboru výukových lekcí pro rozvoj algoritmického myšlení u žáků z pátých a šestých tříd základních škol. V lekcích bude využíván populární a školami využívaný program Scratch.

2. Úkoly a postup řešení práce

Za účelem splnění hlavního cíle práce, vytvoření souboru lekcí pro rozvoj algoritmického myšlení u žáků z vybrané věkové kategorie, je nutné zpracovat dílčí cíle.

Seznámit se s pojmem algoritmické myšlení a možnostmi jeho rozvoje s pomocí Scratch u vybrané věkové skupiny dětí školního věku

Seznámení s pojmem algoritmické myšlení bude probíhat pomocí studia odborných zdrojů a zdrojů zaměřených na učitele. Kromě algoritmického myšlení bude nutné se seznámit i s pojmy, které se týkají informatického myšlení – těmi se budeme zabývat v kapitole 3 (kapitola Informatické myšlení, strana 11). Výsledné poznatky týkající se algoritmického myšlení budou zpracovány v kapitole 4 (kapitola Algoritmické myšlení, strana 15).

Zmapovat soudobé prostředky a nástroje pro rozvoj algoritmického myšlení dětí a určit ty, které by bylo možné, resp. vhodné zařadit před výukou v prostředí Scratch (pro vybranou skupinu dětí školního věku)

Vyhledané aplikace budou rozděleny do kategorií. Bude zde vybrána kategorie, která je nejvhodnější a postoupí do další fáze. Aplikace, které postoupí do další fáze budou všechny z kategorie vybrané v předchozí fázi. Tyto aplikace poté ohodnotíme pomocí hledisek blíže popsaných v kapitole 6 (kapitola Aplikace pro přípravu na programování, strana 24).

Pro zvolenou věkovou skupinu připravit ucelený soubor aktivit orientovaných na rozvoj algoritmického myšlení

Budou sestrojeny na sebe navazující lekce, kde žáky postupně seznámíme s algoritmickými strukturami a algoritmickým myšlením samotným. Tímto tématem se bude zabývat kapitola 7 (kapitola Soubor lekcí, strana 53).

Ověřit navržený soubor aktivit v praxi

Připravený soubor lekcí bude ověřen v praxi na základní škole. Zde se zjistí, jaké změny je potřeba učinit, jak lekce upravit a jaké chyby se v nich nacházejí. Zjištění z ověření budeme reflektovat v kapitole 7.27 (kapitola Ověření navrženého souboru lekcí ve třídě 6.A, strana 139).

3. Informatické myšlení

Informatické myšlení je kognitivní či myšlenkový proces, který zahrnuje logické uvažování, díky kterému nalézáme řešení problémů. Použitím informatického myšlení také lépe porozumíme artefaktům. Artefakty v tomto kontextu jsou systémy, procesy, objekty, algoritmy, problémy, řešení, abstrakce a kolekce dat či informací. Informatické myšlení zahrnuje následující schopnosti:

- schopnost uvažovat logicky
- schopnost uvažovat v rámci dekompozice
- schopnost myslet v generalizacích, schopnost identifikovat a využívat vzorce
- schopnost přemýšlet v abstrakcích, schopnost vybírat vhodné znázornění
- schopnost hodnotit[1]

Pouhá existence počítače nám nezaručuje, že problém, jehož řešení hledáme budeme schopni vyřešit. Počítače zdaleka nejsou na takové úrovni, aby sami hledali řešení, odpověď neznají a lidé musí počítačům přesně a jasně říct, co po nich požadujeme – k tomu nám slouží informatické myšlení, proces, kdy víme jak formulovat požadavek tak, aby mu porozuměli lidé a/nebo stroje. Informatické myšlení se v závislosti na zdroji, ze kterého čerpáme, skládá z proměnlivého počtu hlavních částí, vždy ovšem obsahuje následující části: dekompozice problému, hledání vzoru, abstrakce a algoritmické myšlení.[2]

3.1. Dekompozice problému

Dekompozice problému je způsob přemýšlení o artefaktech v rámci částí, ze kterých se skládají. Těmto částem můžeme po oddělení od celku porozumět, ověřit je a vytvářet je. Díky dekompozici je jednodušší řešit komplexní problémy, lépe díky ní rozumíme novým situacím a je jednodušší navrhovat rozsáhlé systémy.[3]

---

[1] Concepts of computational thinking. CSIZMADIA, Andrew, Prof. Paul CURZON, Mark DORLING a et al. *Computational thinking: A guide for teachers*. Computing At School, 2015, s. 6.
[2] Introduction to computational thinking. In: *BBC* [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1
[3] CSIZMADIA et al, Decomposition, s.8

Na začátku máme komplexní problém. Tento problém musíme rozdělit na menší části, které bude jednodušší rozebírat do detailů. Příkladem z reálného života může být například fungování lidského těla, které může na první pohled fungovat jako perfektně sehraný celek, jehož fungování je na první pohled záhadou. Pokud si ovšem tělo rozdělíme na jednotlivé aparáty a následně i na systémy jako trávicí, dýchací a podobně, vše je přehlednější.

Jiným příkladem aplikace dekompozice na jednoduchou lidskou činnost může být příprava čaje ze sáčku. Položíme si následující otázky:

- *Jaký čaj vlastně chceme, černý, ovocný nebo bylinný?*
- *Jaký hrnek ze skříně vybereme?*
- *S jakou teplotou vody je potřeba ho zalít?*
- *Jak dlouho se daný čaj louhuje?*
- *Přídáme do něj cukr?*

Odpovědi na tyto otázky se zodpovídají mnohem jednodušeji a jednoznačněji než obecný problém s názvem „příprava čaje“.[4]

3.2. Generalizace (hledání vzorů)

Generalizace je spojována s identifikací vzorů, podobností, spojitostí a jejich využití. Je to způsob rychlého nalezení řešení nových problémů na základě předchozích nalezených řešení, využíváme předchozí zkušenosti.[5]

Hledání vzorů je jeden ze čtyř pilířů informatického myšlení. Jedná se o nalezení podobností či vzorů mezi menšími problémy, které vznikly dekompozicí komplexního problému. Díky nalezení těchto podobností jsme schopni řešit komplexní problémy efektivněji – pokud jsme našli řešení na podobný problém, který je dostatečně podobný tomu,který řešíme nyní, řešení vlastně již máme.

[4] Decomposition. In: *BBC* [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zqqfyrd/revision/1
[5] CSIZMADIA et al, Generalisation (Patterns), s.8

Co je to vlastně vzor? Představme si, že máme nakreslit sérii žab. Všechny žáby mají hlavu, tělo a celkem čtyři končetiny. Přední končetiny jsou slabší a kratší. Zadní končetiny jsou větší, širší a silnější, uzpůsobené pro skákání. Pokud se blíže podíváme na hlavu žab, všimneme si jejich velkých vypouklých očí. Na hlavě rovněž mají i tlamu. Žáby patří do skupiny obratlovců s názvem obojživelníci. Jejich charakteristický zvuk nazýváme kvákáním a živí se hmyzem. Jejich kůže je na dotek vlhká. V informatickém myšlení bychom tento popis nazvali vzorem žáby. Pokud máme tento obecný popis, jsme schopni rozpoznat i žábu (nebo druh žáby), kterou jsme nikdy neviděli..

Údaje, ve kterých se budou popisované žáby lišit nazýváme specifické informace. Jako příklad si zde uvedeme popis pralesničky azurové a ropuchy obecné.

Pralesnička kváká stále stejným hlasem několikrát za sebou. Má tmavé oči, její kůže má barvu od světle modré po tmavě modrou s černými fleky. Dorůstá pouze necelých pět centimetrů.

Oproti tomu kvákání ropuchy začne poměrně hlubokým tonem, který se postupně zvyšuje, čím déle kvákání trvá. Ropucha obecná má měděné až zlatavé oči a hrbolatou kůži zeleno-hnědé barvy. Samičky dorůstají délky až dvanáct centimetrů.[6]

3.3. Abstrakce

Abstrakce je proces zjednodušování artefaktu pomocí odstraňování nepotřebných detailů. Zásadní věcí v abstrakci je vybrat si, jaký detail skryjeme, aby se problém stal jednodušším bez toho, abychom ztratili důležité informace[7]

Další zdroje popisují abstrakci jako proces filtrování informací získaných ze vzorů. Pokud se jedná o informaci specifickou či není důležitá pro vyřešení problému, tak ji ignorujeme.
Z informací, které filtrem projdou vytvoříme reprezentaci řešení problému.

[6] Pattern recognition. In: *BBC* [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zxxbgk7/revision/1
[7] CSIZMADIA et al, Abstraction, s.7

Pokud bychom ovšem chtěli někomu popsat žábu za účelem jejího namalování, nejspíše bychom použili pouze minimum charakteristik, které jsme našli v hledání vzorů. Žába je zvíře, které má celkem čtyři končetiny. Přední končetiny jsou slabší a kratší, zadní končetiny jsou větší a širší. Nemylnou součástí hlavy žáby jsou velké vypouklé oči a tlama. Tělo žáby je pokryté kůží. Již by nás nezajímalo čím se tito obojživelníci živí a jaký zvuk dělají. Tímto jsme vytvořili model žáby. I když zde nepopisujeme detaily žáby, jako jsou například přesné proporce předních a zadních nohou, tento příklad postačí pro ilustraci abstrakce.

Model reprezentuje obecnou ideu problému, který se snažíme vyřešit.[8]

3.4. Logické uvažování

Logické uvažování žákům umožňuje čerpat z vlastních zkušeností a interních modelů za účelem vytváření předpokladů a vyvozování závěrů. Žáci jej hojně využívají když testují a opravují algoritmy a nalézají v nich chyby.[9]

3.5. Vyhodnocení

Vyhodnocení je proces ujišťění se, že řešení, ač už jde o algoritmus, systém či proces, je vyhovující. Ptáme se zde na otázky jako: Je řešení správné? Je řešení dostatečně rychlé? Je jeho použití pro uživatele jednoduché? Řešení je zřídka ideální pro všechny situace, kde ho chceme použít.[10]

[8] Abstraction. In: *BBC* [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zttrcdm/revision/1
[9] CSIZMADIA et al, Logical reasoning, s.6
[10] CSIZMADIA et al, Evaluation, s.7

4. Algoritmické myšlení

Algoritmické myšlení je způsob získání řešení skrze jednoznačnou definici po sobě jdoucích kroků. Některé problémy vyžadují řešení, které bude řešit pouze jediný problém, u jiných problémů bude možné využít již dříve vymyšlené řešení.

Algoritmickém myšlení je také možno definovat jako schopnost přemýšlet v rámci sekvencí a pravidel za účelem vyřešení problémů či pochopení situací.[11]

G.A. Zharkova a L.N. Polyakova definují algoritmické myšlení jako schopnost rozdělit úkoly do po sobě jdoucích propojených bloků.[12]

Další zdroje definují algoritmické myšlení jako proces vývoje kódu a počítačových aplikací. Je odvozeno z informatiky. Při použití přístupu s využitím algoritmického myšlení zautomatizujeme proces řešení problému a vytvoříme systematické a logické kroky vedoucí k nalezení řešení. Při hledání rovněž musíme brát v potaz určitý počet vstupů a na nich závislý určitý počet výstupů.

Použitím algoritmického myšlení tedy nenalezneme určitou odpověď, nýbrž sestavíme sekvenční, kompletní a opakovatelný proces, který má jasně určený koncový bod.

Algoritmické myšlení také můžeme definovat jako proces vytvoření algoritmu.[13]

[11] CSIZMADIA et al, Algorithmic thinking, s.7

[12] SADYKOVA, O.V. a G.G. IL'BAHTIN. The Definition of Algorithmic Thinking. In: *Proceedings of the International Session on Factors of Regional Extensive Development (FRED 2019)* [online]. Paris, France: Atlantis Press, 2020, 2020, s. - [cit. 2022-06-14]. ISBN 978-94-6252-882-6. Dostupné z: doi:10.2991/fred-19.2020.85

[13] MC-WEIGH MURPHY, Anna. The One About Algorithmic Thinking in Computational Thinking. In: *Equip Learning* [online]. [cit. 2022-06-01]. Dostupné z: https://equip.learning.com/algorithmic-thinking-computational-thinking/

4.1. Algoritmus

Podle H. Rogerse je algoritmus deterministická procedura, kterou můžeme aplikovat na jakoukoliv třídu symbolických vstupů. Pro každý vstup je v této metodě korespondující symbolický výstup.[14]

V knize Algoritmizácia a úvod do programovania je algoritmus definovaný jako návod na vykonání činnosti, který nás od měnitelných vstupních údajů přivede v konečném čase k výsledku. Autor zde rovněž používá termín výpočet, který definuje jako vykonání činnosti na základě algoritmu.[15]

Další zdroje definují algoritmus jako postup, set instrukcí, které mají pevně dané pořadí ve kterém se mají vykonat. Pokud si umíte vyčistit zuby či zavázat tkaničky, vlastně už víte, jak algoritmus provést a nejspíš již tušíte, že se nacházejí všude kolem nás, aniž bychom si to uvědomovali. Nejčastěji spojujeme pojem algoritmus s počítači a informatikou. Počítačový program je pouze tak dobrý, jak dobrý je algoritmus, který mu předáme.

Každý algoritmus musí mít počáteční stav, konečný stav a set instrukcí, které se mezi těmito dvěma stavy musí provést. Je napsaný buď pro lidi, pro počítače nebo je srozumitelný oboum. Před napsáním algoritmu je nutné problém správně a dostatečně rozložit pomocí dekompozice.[16]

[14] SADYKOVA, O.V a G.G IL'BAHTIN, s. 419-420
[15] Algoritmické štruktúry. SKALKA, Ján, Martin CÁPAY, Gabriela LOVÁSZOVÁ a et al. *Algoritmizácia a úvod do programovania*. Nitra: UKF v Nitre, 2007, s. 11. ISBN 978-80-8094-217-5.
[16] Algorithms. In: *BBC* [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zpp49j6/revision/1

4.2. Vlastnosti algoritmu

V knize Algoritmizácia a úvod do programovania jsou definované následující vlastnosti algoritmu:

- Elementárnost – Postup je složený z jednotlivých kroků, které jsou pro vykonavatele srozumitelné. Vykonavatelem může být buď člověk nebo počítač.
- Determinovanost – Postup je sestavený tak, aby bylo v každém momentě jeho vykonávání jednoznačně určené, jaká činnost má následovat či zda postup skončil.
- Rezultativnost – Výpočet dává v konečném počtu kroků výsledek.
- Konečnost – Výpočet vždy skončí po určitém počtu kroků.
- Hromadnost – Algoritmus je použitelný na celou třídu přípustných vstupních údajů.
- Efektivita – Výpočet se uskutečňuje v co nejkratším čase a s využitím co nejmenšího množství prostředků.[15]

Další zdroje definují následující vlastnosti algoritmu:

- *Elementárnost – Instrukce musí být jednoduché a srozumitelné. Příkladem špatně napsané instrukce by bylo: Zapleť vánočku. Způsobu, jak zaplést vánočku je neskutečně mnoho, tato instrukce není dostatečně elementární.*
- *Determinovanost – Kroky v algoritmu a jejich pořadí jsou jednoznačně určeny. V každé situaci je naprosto jasné, jaký krok se má provést.*
- *Univerzálnost – Vstupy mají definované množiny hodnot, kterých mohou nabývat. Algoritmus lze použít na celou množinu vstupních dat.*
- *Rezultativnost –Jednotlivé kroky algoritmu vedou k výsledku v konečném počtu kroků.*
- *Obecnost – Algoritmus neřeší pouze jeden určitý problém, ale je možné ho použít na obecnou třídu obdobných problémů. Například neřešíme pouze sčítání čísla dva a pět, ale sčítání celých čísel.*
- *Efektivita – K realizaci algoritmu potřebujeme co nejméně prostředků a samotná realizace by měla zabrat co nejkratší čas.*
- *Konečnost – Algoritmus musí skončit v konečném počtu kroků, nemůže být nekonečná smyčka.*

Příkladem algoritmu přímo z oblasti informačních technologií může být fungování vyhledávání pomocí Google. Po zadání hledaného termínu se nám zobrazí jako první ty stránky, které jsou nejvíce citované jako zdroje na stránkách ostatních (pokud se nejedná o reklamu). Příkladem z reálného života může být dříve zmiňované čištění zubů. V algoritmu pro tuto činnost bychom přesně a jednoznačně popsali jakým způsobem s kartáčkem pohybovat a kde ho použít (elementárnost), v jakém pořadí kroků je nutné na kartáček vymačkat zubní pastu (determinovanost), jak dlouho zuby v určité části pusy čistit (konečnost). Také zde nezáleží na typu kartáčku a značce zubní pasty (obecnost). [21,17,18,19]

[17] Algoritmus. In: *GJŠ Zlín* [online]. [cit. 2022-06-01]. Dostupné z: https://www.gjszlin.cz/ivt/esf/algoritmizace/algoritmus.php

[18] Vlastnosti algoritmu. elementárnost. determinovanost. rezultativnost. konečnost. hromadnost. efektivnost. In: *DOCPLAYER* [online]. [cit. 2022-06-01]. Dostupné z: https://docplayer.cz/134660472-Vlastnosti-algoritmu-elementarnost-determinovanost-rezultativnost-konecnost-hromadnost-efektivnost.html

[19] Algorithms (Characteristics, Guidelines & Advantages). In: *CodeSansar* [online]. [cit. 2022-06-01]. Dostupné z: https://www.codesansar.com/computer-basics/algorithms.htm

4.3. Algoritmické struktury

Abychom mohli pomocí programovacího jazyka komunikovat, potřebujeme mít stanovené příkazy, díky kterým procesoru sdělíme, co přesně po něm požadujeme vykonat. Podle Skalky příkaz představuje elementární činnost, kterou je schopen vykonavatel algoritmu realizovat. Pokud umístíme několik příkazů pod sebe do pořadí, v jakém chceme, aby byly vykonány, jedná se už o *sekvenci*.[20]

Díky *příkazům vstupu* může uživatel vkládat do algoritmu různé vstupní údaje. S těmito údaji se realizují předepsané operace a výsledek je vypsán či zobrazen na výstupu pomocí *příkazů výstupu*.[21]

Proměnná je místo v paměti, kam se během běhu programu ukládají údaje. Její hodnota se během běhu programu mění. Každá proměnná se musí nějak jmenovat a je vhodné, aby z jejího názvu bylo hned jasné, k čemu slouží.[22]

Pokud v kódu potřebujeme vykonat určité příkazy pouze za splnění určitých dopředu definovaných podmínek, využijeme takzvané *větvení*. V závislosti na splnění či nesplnění se postup větví na různé větve. Pokud je podmínka splněna, vykonají se příkazy v kladné větvi. Pokud podmínka splněna není, dějí se příkazy v záporné větvi.[23]

Běžně se v programování setkáme se situací, kdy potřebujeme určitý příkaz provést více než jednou. Pro tyto případy použijeme *cyklus*. Při jeho použití je nutné vědět, co se má opakovat a do kdy je nutné opakování provádět. Opakovanou činnost nazýváme *tělo cyklu* a část, která určuje, kolikrát činnost opakujeme se nazývá *podmínka cyklu*.[24]

[20] SKALKA et al, Príkazy vstupu a výstupu, s.18
[21] SKALKA et al, Príkazy vstupu a výstupu, s.19
[22] SKALKA et al, Premenná, s.19
[23] SKALKA et al, Vetvenie, s.23
[24] SKALKA et al, Cyklus, s.31

Existují tři typy cyklů. První je *cyklus se známým počtem opakování*, kdy vyjádříme počet opakování před jeho odstartováním a operace v těle na něj nemají žádný účinek.[25] Druhý typ je *cyklus s podmínkou na začátku*, kdy se podmínka umístěná nad tělem stará o ukončení cyklu. Pokud je podmínka cyklu splněná, vykonají se příkazy v těle cyklu a opět se otestuje, zda podmínka platí.[26] Třetím typem je *cyklus s podmínkou na konci*. V tomto typu cyklu se jako první vykoná tělo cyklu a až poté se kontroluje platnost podmínky cyklu. Pokud je podmínky splněná, cyklus se dále neprovádí. Pokud je podmínka splněna již při spuštění kódu proběhnou příkazy uvnitř těla i v tomto případě jedenkrát.[27] Pro začátečníky je nejvhodnější skládat algoritmy pomocí grafických bloků.[28]

[25] SKALKA et al, Cyklus so známym počtom opakovaní, s.31
[26] SKALKA et al, Cyklus s podmienkou na začiatku, s.34
[27] SKALKA et al, Cyklus s podmienkou na konci, s.36
[28] VAIS, Jan. Rozvoj algoritmického myšlení žáků základních škol. *Digitální repozitář Univerzity Karlovy* [online]. [cit. 2022-06-01]. Dostupné z: https://dspace.cuni.cz/handle/20.500.11956/125706

5. Scratch

Scratch je název vizuálního programovacího jazyka, který v roce 2005 vyvinul Mitchel Rosnick. Totožně se jmenuje i program, kde pomocí tohoto programovacího jazyka tvoříme scénáře, to znamená programy napsané právě v tomto programovacím jazyce.

Scratch je kompletně zdarma, neexistuje zde nic jako premiové verze či cokoliv podobného. Existují verze programu pro offline a online použití. Co se týče offline verzí, z oficiálního webu je možné si stáhnout verzi pro Windows, Mac OS a Linux. Pro školy možná jednodušší ovšem bude Scratch používat v online módu. Nemusíme nic instalovat a stačí nám pouze internetový prohlížeč, kterým se připojíme na oficiální stránky a levým tlačítkem myši stiskneme položku v menu nazvanou „tvořit“.[29]

V rámci projektu Podpora rozvoje informatického myšlení také vzniklo několik učebnic, které jsou zdarma k dispozici nejen učitelům na stránkách imysleni.cz.[30] Tyto učebnice jsou určené pro žáky od pátého ročníku až pro pokročilé žáky z druhého stupně základních škol.[31]

Největší výhodou Scratche je jeho téměř kompletní jazyková lokalizace v češtině, co se týká webu i offline programu. Jednou z opravdu malého množství anglických textů na české verzi stránek je níže citovaný dokument Scratch . Teacher Accounts.

[29] Scratch. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-05-22]. Dostupné z: https://cs.wikipedia.org/wiki/Scratch
[30] *IMyšlení* [online]. [cit. 2022-05-22]. Dostupné z: https://www.imysleni.cz/
[31] PODPORA ROZVOJE INFORMATICKÉHO MYŠLENÍ. In: *RVP* [online]. [cit. 2022-05-22]. Dostupné z: https://digiskola.rvp.cz/dvz/detail-projektu?id=4

Nutnost registrace a možnosti uložení scénářů

Pro použití Scratche není nutné mít založený účet, jakožto nepřihlášený uživatel si dokonce můžeme stáhnout vytvořený scénář na svůj počítač a poté ho znovu otevřít v prohlížeči. Pro učitele je ovšem vhodnější založit si přímo učitelský účet, zde založit třídu a žáky do ní přidat. Můžeme žáky přidávat jednotlivě, ale mnohem efektivnější je nasdílet žákům odkaz, na který se připojí a založí si účet s vlastním jménem a heslem. Další možností je nahrát hesla a jména žáků jako csv soubor.

Program Scratch nám rovněž umožňuje práce žáků kontrolovat. V rámci třídy založíme nové třídní studio a sami buď do něj scénáře vytvořené žáky přidáme či v nastavení studia povolíme přidávání scénářů.[32]

Design

Podle typu pohybu, který chceme aby postava vykonala, nám program Scratch nabídne jak postavy, které jsou z profilu i postavy z pohledu. Postavy z profilu ovšem vypadají podivně, pokud je naprogramujeme na pohyb nahoru a dolů. Na výběr máme z pestrého druhu postav, od realistických vyříznutých z fotek po postavy nakreslené rozmanitým kresleným stylem. Kromě postav si můžeme vybrat i pozadí, které se bude k danému scénáři hodit – postavy i pozadí jsou rozdělené do tématických celků. Pokud jsme přesto nenašli vhodného herce do našeho scénáře, máme možnost si vlastní postavu nebo pozadí namalovat ¨přímo v programu Scratch či nahrát existující Obrázek z našeho počítače.

Bloky v programu Scratch

Bloky v programu Scratch jsou přehledně rozdělené do barevně odlišených skupin. Většina skupin, snad kromě Ovládání, se jmenuje dostatečně jednoznačně na to, aby i laik tušil, kde najít blok, který potřebuje použít. To samé platí pro bloky samotné, které jsou opět pojmenovány skoro doslovným způsobem, žáci rychle pochopí, jaká je činnost daného bloku.

[32] Scratch - Teacher Accounts. In: *Scratch* [online]. [cit. 2022-05-22]. Dostupné z: https://resources.scratch.mit.edu/www/guides/en/scratch-teacher-accounts-guide.pdf

Jako příklad můžeme uvést blok Dopředu o 10 kroků z trefně pojmenované skupiny bloků Pohyb. Celkově bloky hodně připomínají pseudokód.

Jako příklad jediného, lehce nejednoznačného, bloku můžeme uvést bloky související s kostýmy. Kostým ve Scratchi znamená převlek postavy, kterou lze v kontextu programu přirovnat k hercovi.

Pokud ve Scratchi pracujeme s blokem, kde máme na výběr z omezeného množství možností (oproti například bloku Dopředu o 10 kroků, zde můžeme desítku přepsat na skoro libovolně velké číslo) program nám sám tyto možnosti nabídne a my pouze vybereme z nabídky. Ručně tedy zapisujeme absolutní minimum. Příkladem takového bloku je například blok Změň kostým na ze skupiny bloků Vzhled. Z menu kostýmů, které má postava k dispozici tedy pouze vybereme požadovaný.

Scratch a algoritmické myšlení

Program Scratch obsahuje všechny hlavní algoritmické struktury. Najdeme zde větvení ve formě bloků Když..tak..jinak a Když..tak, cykly s daným počtem opakováním se zde nacházejí v blocích Opakuj stále a Opakuj x krát.Rovněž zde nechybí cykly s podmínkou na začátku a to v bloku Opakuj dokud nenastane. Jediný typ cyklu, který ve Scratchi nenajdeme je cyklus s podmínkou na konci. Poslední z pilířů algoritmického myšlení, sekvence, žák použije již při prvním setkání s tímto programem.

6. Aplikace pro přípravu na programování

Předmětem zkoumání teoretické části je vybrat aplikaci, která by byla vhodná pro uvedení žáků do problematiky algoritmizace. S touto aplikací by žáci pracovali v předcházející jedné či dvou hodinách před samotnou výukou základů programování ve Scratchi.

Budeme tedy vybírat jak z aplikací počítačových tak i z aplikací, které můžeme nainstalovat na zařízení s operačním systémem Android z obchodu Google play či na zařízení Apple z obchodu App Store.

Výběr byl proveden z aplikací, které byly nalezeny na učitelských portálech či na zdrojích týkajících se informatiky. Výběr bude mít dvě fáze. V první rozdělíme dostupné aplikace do kategorií a vybereme kategorii, která bude vyhovující pro uvedení žáků do problematiky algoritmizace. V rámci této fáze také vyřadíme z vybrané kategorie aplikace, které se z různých důvodů nehodí. Ve druhé fázi porovnáme aplikace podle porovnávaných kritérií. Na základě kriteriálního porovnání aplikací zformulujeme doporučení pro výběr aplikace.

6.1. Přípravná fáze

V prvním kole jsme zmapovali kategorie aplikací týkající se výuky programování, které jsou dostupné na dostupných zařízeních, tedy na počítačích a tabletech. Každou z kategorií zde popíšeme a zmíníme jednu aplikaci jako jejího představitele.

Kurzy pro samouky

Tento typ aplikace je orientovaný na sebevzdělávání a nevyhovuje způsobu výuky ve třídě. Příkladem z této kategorie je například aplikace Enki, kde si uživatel každý den projde část učiva z oblasti technického oboru, která ho zajímá.

Tvůrčí blokové programování

Aplikace, které kvalifikujeme jako tvůrčí blokové programování umožňují uživateli volně tvořit vlastní scénáře, skládáme v nich algoritmus z bloků. V této bakalářské práci již tvoříme scénáře ve Scratchi, který také patří do této kategorie. Není proto potřeba se zabývat aplikacemi tohoto typu, jako příprava na programování by nebyly vhodné. Příkladem takové aplikace je například Pocket Code.

Aplikace pro ovládání robotů

Tento typ aplikace je vlastně jen doplněk k robotovi, je tedy nutností mít jak tablet tak i další hardwarové zařízení. Většinou mu pomocí aplikace programujeme například cestu, kterou se má vydat. Příkladem aplikace je Photon Coding, kde žáci programují robota jménem Photon.

Hlavolamy a úkolově orientované aktivity

Tento typ aplikací často obsahuje postavičku, která se chce dostat k určitému cíli a její cestu musíme naprogramovat. Samotné programování probíhá různými způsoby - v některých aplikacích se tak děje pomocí psaných příkazů, jinde se používají příkazové bloky a v aplikacích zaměřených na mladší děti se používají bloky s ikonickým vyjádřením programové instrukce. Právě hlavolamy, jak je budeme v této práci nazývat, nás budou zajímat. Je to nejvhodnější typ aplikace na první setkání s programováním.

Na závěr první fáze jsme z aplikací z kategorie hlavolamy vyřadili z různých důvodů nevyhovující aplikace. Mezi důvody patřila například přítomnost prvků nesouvisejících s výukou, infantilnost (některé aplikace si dali za úkol naučit kromě programování i základy počítání jako sčítání a odečítání, to žáci v páté a šesté třídě dávno umí) či nutnost si koupit dopředu předplatné programu bez toho, aby byl pořádně otestován.

6.2. Vyhodnocení aplikací

Aplikace, které postoupili do třetího kola budou posuzovány podle následujících hledisek. Ideální aplikace bude obsahovat základní použití programátorských konceptů jako je cyklus a podmínky.

6.2.1 Bodově nehodnocená hlediska aplikace

Následující hlediska jsou pouze informační.

Operační systém a zařízení

Některé aplikace jsou přístupné z jakéhokoliv zařízení – může se jednat o webové aplikace či mohou být naprogramovány pouze pro jeden typ operačního systému. Nejčastěji půjde o operační systém Android, Mac OS, Windows nebo Linux.

Typ programování

Používá aplikace bloky s příkazy, zadávání příkazů v textové podobě či bloky s ikonickým vyjádřením programoví instrukce, např. kolo s číslem uvnitř jako zástupný znak cyklu?

6.2.2 Bodově hodnocená hlediska aplikace

Každou aplikaci zhodnotíme pomocí následujících kritérií. Aplikace s největším počtem bodů bude vybrána jako nejvhodnější. Maximum získaných bodů je 34.

Podpora češtiny (max. 5 bodů)

I když žáci angličtině rozumí, programátorský žargon je lepší jim představit jako první v češtině. V některých hlavolamech jsou instrukce pro žáky rozsáhlé, dlouhé až několik řádků a to už by v angličtině mohl být problém.

Nutnost registrace (max. 2 body)

V této části hodnocení se budeme zaobírat otázkou, zda je pro plné využití aplikace nutné si zakládat účet či ne. Účet si nejčastěji musíme zakládat kvůli uložení postupu, který by se nám například po zavření prohlížeče bez registrace ztratil.

Nutnost instalace (max. 2 body)

Je aplikace webovou aplikací či je nutné si ji nainstalovat a tím pádem řešit její instalaci na všechna zařízení ve školní učebně například se školním it adminem?

Srozumitelnost zadání (max. 5 bodů)

Do jaké míry je aplikace pochopitelná pro laika, někoho kdo v životě neprogramoval? Jak moc nás, lidově řečeno, „vede za ručičku“ a vysvětluje co nám zadává za úkol? Toto hledisko se týká funkčností jako nápovědy v případě kdy se setkáme poprvé s cyklem či novým typem příkazu.

Vhodnost pro věkovou kategorii (max. 5 bodů)

Teoretická část této bakalářské práce počítá s dětmi, které chodí do pátého až šestého ročníku základní školy, nebylo by tedy vhodné po nich chtít řešit úkoly vhodné pro předškoláky.

Uživatelská přívětivost (max. 5 bodů)

V hledisku srozumitelnost zadání porovnáváme aplikaci co se týče jednoznačnosti úkolů, které po nás žádá vyřešit, ale v tomto porovnávaném hledisku se budeme zaobíratsamotným uživatelským prostředím. Je zde v případě nutnosti účtu jasné, že je uživatel přihlášen? Když potřebujeme něco najít, zabere nám maximálně tři kliknutí? Na tyto a podobné otázky si zodpovíme v této části hodnocení.

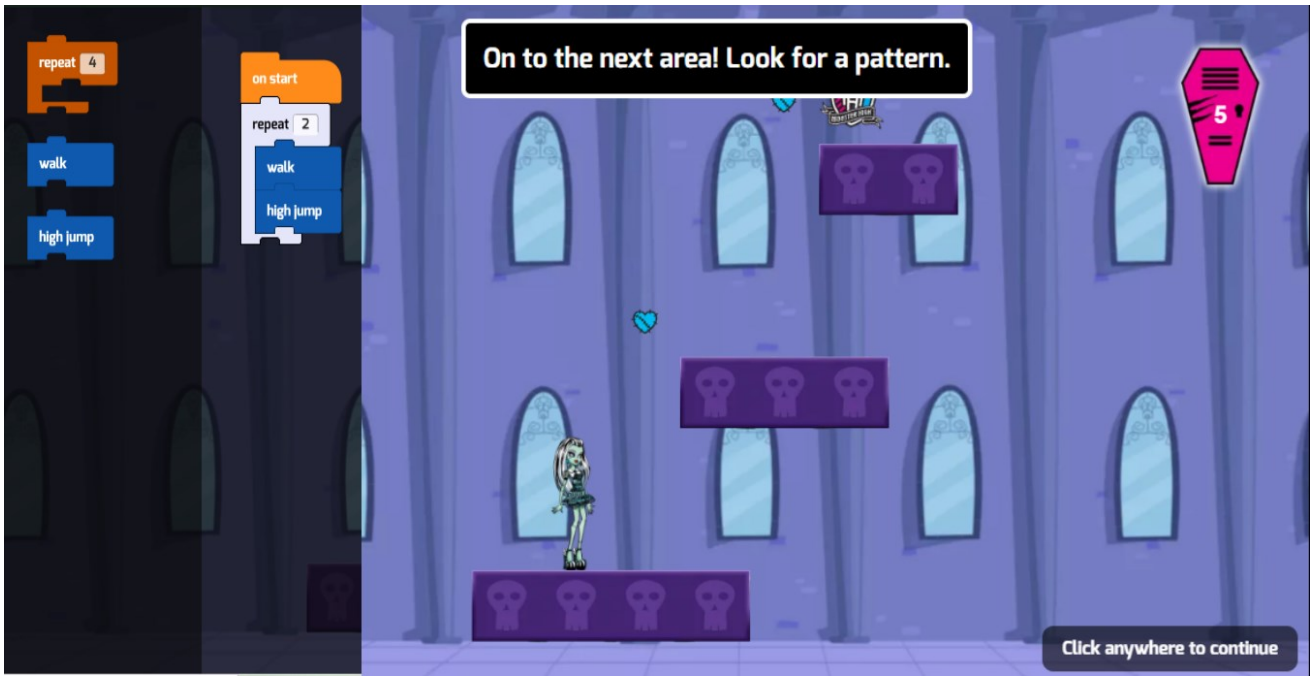
Design (max. 5 bodů)

Aplikace, kterou vybereme jako vítěze by měla kromě funkčnosti i dobře vypadat – chceme jejím použitím koneckonců oslovit ty největší kritiky, děti. Toto hledisko bude z větší části posuzováno ze subjektivního pohledu.

Probírané programátorské koncepty (max 5.bodů)

Aplikace by měla projít minimálně cykly a podmínky. Tempo těchto nově nabývaných vědomostí by mělo být úměrně rychlé, nepříliš zbrklé a také nepříliš pomalé. Příkladem zbrklosti by byla aplikace, která po nás v již čtvrtém cvičení chce řešit spojení cyklu a podmínky, příkladem pomalé aplikace by bylo probírat programátorský koncept příkazu již po patnácté za sebou.

6.2.3 Monster High: Scavenger hunt



Obrázek 1 Vzhled aplikace Monster High: Scavenger Hunt

Operační systém

Monster High: Scavenger hunt je webová aplikace na webu tynker.com, je tedy možné ji spustit na většině zařízení kromě mobilních telefonů – aplikace se zde hůře ovládá.

Typ programování

V aplikaci se používá blokové programování.

Podpora češtiny

Hodnocená aplikace nemá podporu češtiny.

0/5 bodů

Nutnost registrace

Hodnocená aplikace ukládá postup žáka i bez registrace. Problém by ovšem nastal, kdyby dva různí žáci chtěli na stejném počítači a prohlížeči hrát tento hlavolam každý jinou hodinu – uložil by se postup prvního žáka a druhý žák by mu postup přepsal. Z tohoto důvodu by bylo lepší, aby každý žák měl vlastní účet, který funguje nejenom pro hlavolam Monster High: Scavenger Hunt, ale i pro další hlavolamy a hry na webu tynker.com. Jsou zde k dispozici tři typy účtů, učitelský, kde můžeme vytvořit a spravovat virtuální třídu, studentský, kde se k vytvořené třídě můžou přidat žáci a rodičovský. Z rodičovského či učitelského účtu můžeme založit účet studenta, kterým se poté přihlásí.

1/2 bodů

Nutnost instalace

Hodnocená aplikace je webová aplikace, nemusíme ji instalovat.

2/2 bodů

Srozumitelnost zadání

Zadání úkolů je v aplikaci velmi dobře vysvětleno. Když po uživateli chce aplikace použít nový blok, ještě před samotným řešením představí na čem budeme v dané lekci pracovat. Pokud kód sestavíme špatně, aplikace navrhne možné opravy. Pokud už jsme uplně ztraceni, nad okénkem aplikace je k dispozici tlačítko s nápisem „hint“. Zde vidíme nápovědu více rozepsanou a konkrétnější, občas nám odhalí i kompletní část řešení.

5/5 bodů

Vhodnost pro věkovou kategorii

Téma celé aplikace, panenky a seriál Monster High, je pro dívky z páté a šesté třídy ideální. Slovy autora by se dali popsat jako „rebelské barbie“, s nadpřirozenými prvky jako jsou upíři a vlkodlaci v typickém prostředí americké střední školy. Aplikaci srážím jeden bod, protože toto téma nejspíše hochy moc nezaujme.

4/5 bodů

Uživatelská přívětivost

V aplikaci je jednoduché najít výběr lekce – nachází se v liště nad okénkem aplikace, společně s dalším ovládáním, které je viditelné po stisknutí tří teček. Zde můžeme vypnout a zapnout zvuk, zobrazit nápovědu a znovu spustit lekci.

5/5 bodů

Design

Design aplikace sedí do světa kresleného seriálu, ze kterého čerpá inspiraci. Nejde tedy o vzhled klasické dětské aplikace, ale třeba právě proto by mohl nějaké žáky více zaujmout.

5/5 bodů

Probírané programátorské koncepty

Velkou výhodou aplikace je množství programátorských konceptů, které si uživatel projde - podmínky, cykly, vložené podmínky, smyčky „while“. Vše postupuje ideální rychlostí, dokud nedojdeme do pokročilejších lekcí, kde se rychlost dá považovat za zbrklejší.

4/5 bodů

Celkový počet bodů 26/34

Aplikace obsahuje celkem dvacet lekcí. V rámci každé lekce programujeme pohyb postavy přes platformy směrem k cíli, kterým je barevný erb. Ačkoliv je cíl lekce pokaždé totožný a dá se ho dovtípit i bez znalosti angličtiny na vyšší úrovni než je A1, aplikace obsahuje poměrně velký počet textu v angličtině. Jsou v ní vysvětlovány instrukce a nápovědy, dokonce i nápověda poté, co kód špatně sestavíme a spustíme. Samotné programovací bloky používají jednoduchou angličtinu. Z tohoto důvodu by se tato aplikace hodila spíše jako příprava na programování pro základní školy s rozšířenou výukou jazyků či pro základní mezinárodní školy.

6.2.4 Codemonkey

Obrázek 2 Vzhled aplikace Codemonkey

Operační systém

Hodnocená aplikace je webovou aplikací.

Typ programování

Hodnocená aplikace využívá kombinace blokového programování a psaných příkazů – stisknutím tlačítek pod kódem do psaného kódu vložíme příkaz a klávesnicí napíšeme pouze čísla, jako například počet kroků, které musí postava ujít či o kolik stupňů se má otočit. Pokud chceme, můžeme příkazy psát celé pomocí klávesnice.

Podpora čeština

Hodnocená aplikace nemá podporu češtiny.

0/5 bodů

Nutnost registrace

Hodnocená aplikace při prvním použití nenutí uživatele k registraci, ale neukládá jeho postup. Pokud se pokusíme použít stejnou aplikaci na tom samém zařízení podruhé, již budeme k registraci přinuceni. Opět je zde několik typů uživatelských účtů, fungují a jmenují se totožně jako uživatelské účty na webu Tynker, viz. hodnocená aplikace Monster High: Scavenger Hunt a hodnocené stanovisko „nutnost registrace“. Společně s registrací se ukáže i možnost předplatného, v rámci kterého se žáci dostanou k pokročilejším konceptům programování. Neplatícím uživatelům jsou k dispozici pouze úvodní oblast (která vysvětluje co je to příkaz), oblast zabývající se objekty a cykly.

0/2 bodů

Nutnost instalace

Hodnocená aplikace je webová aplikace, nemusíme ji instalovat.

2/2 bodů

Srozumitelnost zadání

Hodnocená aplikace využívá známou formuli zadávání úkolů – postava opice sbírá banány a uživatel programuje její cestu. Nápověda, které se objeví pokud spustíme špatně napsaný kód se dostatečně nepřizpůsobuje chybě v kódu a při různých typech chyby zůstává stejná. Mezi výhodu aplikace, kterou ocení hlavně učitelé zdatně ovládající angličtinu, patří možnost video tutoriálů s vysvětlením teorie pro všechny oblasti. Tyto videotutoriály najdeme pokud klikneme na postavu opice a poté klikem levého tlačítky myši vybereme ikonu play.

3/5 bodů

Vhodnost pro věkovou kategorii

Hodnocená aplikace se zdá pro žáky páté a šesté třídy základních škol ideální, není přehnaně infantilní.

5/5 bodů

Uživatelská přívětivost

Hodnocená aplikace má dobře zpracované uživatelské prostředí, vše je přehledné a lehce dohledatelné.

5/5 bodů

Design

Design hodnocené aplikace je příjemný, vypadá jako namalovaný rukou. Není příliš dětinský, takže pro věkovou skupinu žáků, se kterou tato práce počítá, ideální.

5/5 bodů

Probírané programátorské koncepty

V neplacené verzi aplikace máme přístupné tři oblasti aplikace – příkazy, objekty a cykly. Každá oblast má jiné téma a uvnitř oblasti se nachází deset až dvacet cvičení zabývající se danou problematikou. V placené části aplikace se nachází oblasti s následující tématikou: proměnné, pole, smyčka for. V případě, že by škola byla ochotná si aplikaci zakoupit by získala poměrně komplexně zpracovanou aplikaci, po které by žáci byli připraveni nejenom na Scratch, ale i pro programování například v jazyce Python či C. Další plus má aplikace za využívání číslování dle zvyklostí v programování, tedy od nuly. Mezi její mínusy bohužel patří přehnaně velké množství cvičení, což by zabralo minimálně tři hodiny.

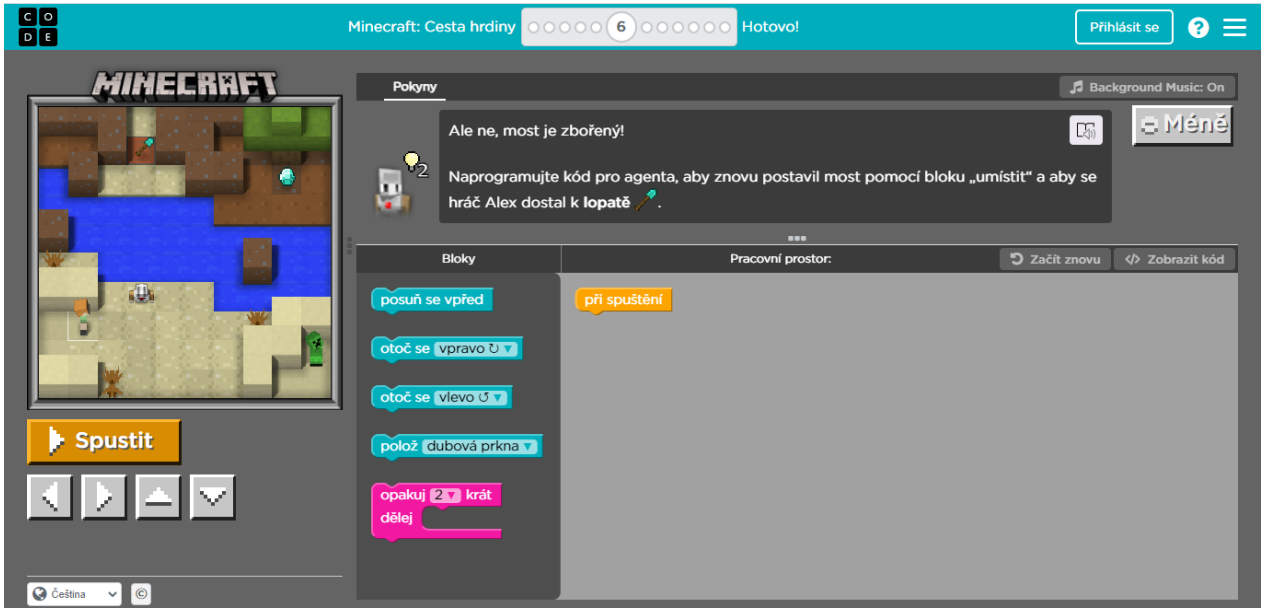
3/5 bodů

Celkový počet bodů 23/34

Velkou výhodou hodnocené aplikace je množství programátorských konceptů a zvyklostí, které žákům představí, i tak zde ale bohužel chybí oblast zaměřující se na podmínky. Jako většina hlavolamů není v češtině, což ale nemusí být nutně na škodu – angličtina zde používaná je natolik jednoduchá, že je možné výuku pomocí této aplikace brát jako ideální nástroj pro rozvoj mezipředmětových vztahů. Dalším plusem aplikace jsou lekce, kde se poprvé představuje nějaký programátorský koncept – v těchto lekcích uživatel nic neprogramuje, pouze pozoruje, jak bloky fungují na jednoduchém příkladu.

Pro jednoduché zjištění vzdálenosti postavy opice od banánu či překážky se v hodnocené aplikaci nachází pravítko. Klikneme na něj levým tlačítkem myši, poté totožným tlačítkem klikneme na oba dva předměty, jejichž vzdálenost chceme změřit a v aplikaci se nám zobrazí vzdálenost. Pravítko rovněž měří i úhly.

6.2.5 Minecraft: Cesta hrdiny



Obrázek 3 Vzhled aplikace Minecraft: Cesta hrdiny

Operační systém

Hodnocená aplikace je webová aplikace, spustíme ji tedy na většině zařízení.

Typ programování

V hodnocené aplikaci programujeme pomocí blokového programování velmi podobnému Scratchi.

Podpora češtiny

Hodnocená aplikace je první nalezená aplikace, která má bloky a instrukce v češtině, úvodní videa k lekcím jsou v angličtině s českými titulky. Do češtiny aplikaci přepneme pomocí tlačítka s názvem jazyka, ve kterém aplikace momentálně funguje. Toto tlačítko se nachází na dolní liště aplikace.

5/5 bodů

Nutnost registrace

Registrace na webu Hour of Code není nutná pro běh aplikace.

2/2 bodů

Nutnost instalace

Hodnocená aplikace je webová aplikace, není nutné ji instalovat.

2/2 bodů

Srozumitelnost zadání

Součástí aplikace jsou úvodní krátká jedno až tříminutová videa, kde známí hráči vysvětlí probírané koncepty zajímavým způsobem, například rozhovorem s programátorkou, která pracuje přímo pro Minecraft. V aplikaci není nijak zpracovaná nápověda. Zadání se kromě ve videu dozvíme i z paragrafu nazvaného Pokyny,který se nachází nad prostorem, kam umisťujeme bloky.

4/5 bodů

Vhodnost pro věkovou kategorii

Tématem aplikace je videohra minecraft, která je dětmi velmi oblíbená. Na hře dokonce spolupracovali (v době vzniku hry) anglicky mluvící známí hráči této hry. Aplikace vznikla ve spolupráci s Microsoftem, jako příprava na programování edukační verze Minecraftu, který Minecraft koupil.

5/5 bodů

Uživatelská přívětivost

Uživatelské rozhraní aplikace je jednoduché na pochopení. Nemusíme se zdlouhavě proklikávat různými menu, abychom našli to, co hledáme.

5/5 bodů

Design

Design hodnocené aplikace je v barvách proslulé videohry Minecraft, která se proslavila právě svým specifickým stylem, který dnes již skoro každý rozezná.

5/5 bodů

Probírané programátorské koncepty

V aplikaci si uživatelé projdou programátorské koncepty jako cyklus a funkce, bohužel se zde nenachází podmínka. Menší počet lekcí umožňuje úměrnou rychlost představování nových konceptů.

3/5 bodů

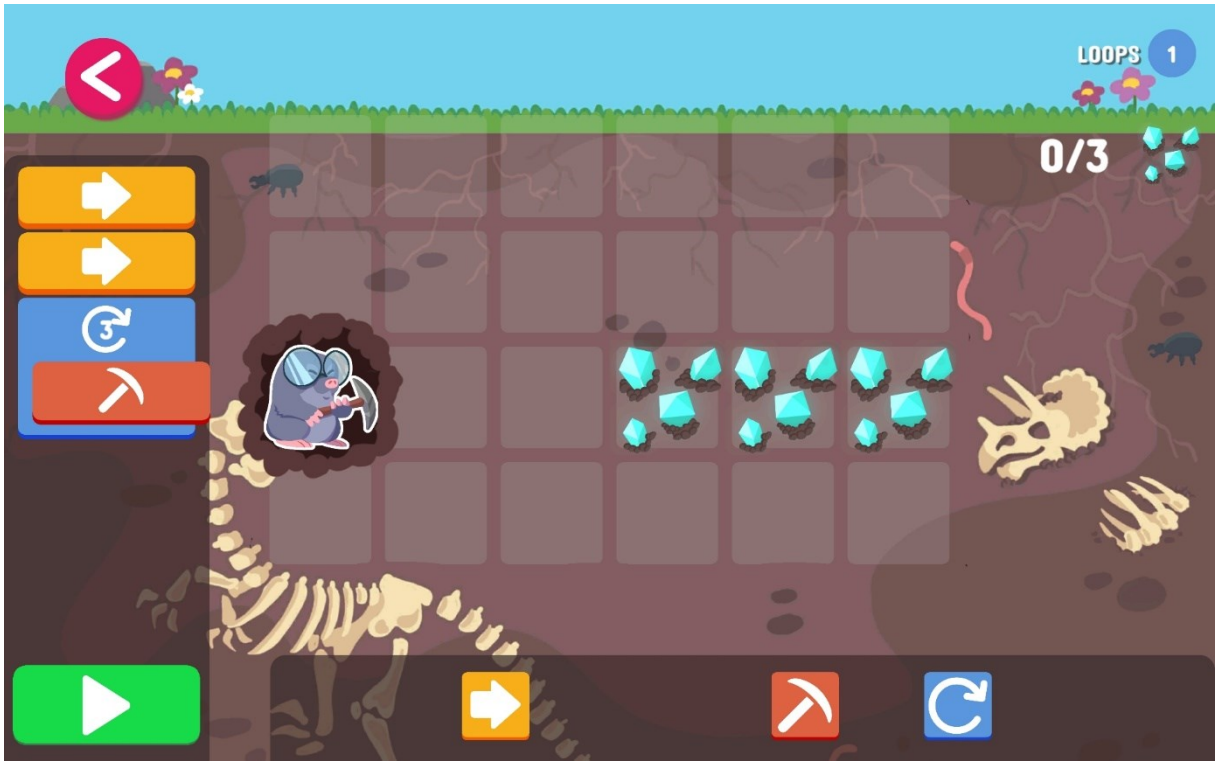
Celkový počet bodů 31/34

Hodnocená aplikace má celkem dvanáct lekcí, pokud počítáme i lekci poslední, kde děti programují pomocí naučených příkazů bez zadaného úkolu. V hlavolamu se nacházejí dvě postavy – postava robota, kterého ovládáme pomocí příkazových bloků a postava hráče, kterou uživatelé ovládají pomocí šipek na klávesnici. Robot je později představen uživatelům jako softwarový agent, kterého programátoři využívají pro často opakovanou činnost.

Lekce, ve kterých aplikace představuje nový programátorský koncept začínají krátkým videem, kde je tento koncept jednoduše vysvětlen. Cílem každé lekce je dostat postavu přes různé překážky k určité věci z videohry Minecraft, například k sekeře. Pro nadanější žáky je zde k dispozici i úkol navíc, který spočívá v sebrání diamantu. Lekce není schopna vyřešit postava hráče sama, robot musí například položit kolejnice, rozbít bloky nebo stát na určitých tlačítkách, které otevírají hráčovi zamčené dveře.

Aplikace je velmi podobná Scratchi, nacházejí se zde i podobné oblasti aplikace – scénář, prostor scénáře, bloky a sekvence bloků.

6.2.6 Kids Coding Skills



Obrázek 4 Vzhled aplikace Kids Coding Skills

Operační systém

Hodnocená aplikace se nachází na obchodu Play na zařízeních Android.

Typ programování

V hodnocené aplikace programujeme pomocí bloků s ikonickým vyjádřením programové instrukce.

Podpora češtiny

Aplikace nemá českou podporu, její ovládání je řešeno názornými bloky s ikonickým vyjádřením programové instrukce místo vysvětlování v jakémkoliv jazyce.

3/5 bodů

Nutnost registrace

Hodnocená aplikace nevyžaduje registraci, k jejímu používání potřebujeme pouze Google účet, který ve většině případů žáci základních škol již od školy zařízený.

2/2 bodů

Nutnost instalace

Aplikaci je nutno nainstalovat na zařízení Android.

0/2 bodů

Srozumitelnost zadání

V hodnocené aplikaci neexistuje nápověda ve smyslu představování nového programátorského konceptu. V každé úrovni přibyde nový blok s ikonickým vyjádřením programové instrukce, který používáme jako příkaz a uživatelé musí na jeho funkčnost přijít sami. Jediné co jim aplikace poskytne je ikonka ruky, která uživatelům ukáže kam vložit blok s ikonickým vyjádřením programové instrukce. V případě cyklu aplikace předá uživateli informaci navíc – kliknutím na blok s ikonickým vyjádřením programové instrukce cyklu se zvýší počet jeho opakování.

Aplikace dá jasně najevo, co přesně má postava sbírat, na pravé straně obrazovky se nachází ikonka sbírané věci společně s počtem nasbíraných věcí stejného typu.

2/5 bodů

Vhodnost pro věkovou kategorii

Hodnocená aplikace je vhodná pro žáky páté i šesté třídy základní školy, dokonce i pro mladší žáky kvůli používání příkazů ve formě bloků s ikonickým vyjádřením programové instrukce.

5/5 bodů

Uživatelská přívětivost

Hodnocená aplikace má jednoduché ovládání.

5/5 bodů

Design

Velkou výhodou aplikace je výběr prostředí a postavy, jaká se nám zamlouvá – v každé úrovni máme na výběr z dvou různých designů té samé aktivity, například u funkcí si vybíráme z programování pohybu žáby nebo ninjy.

5/5 bodů

Probírané programátorské koncepty

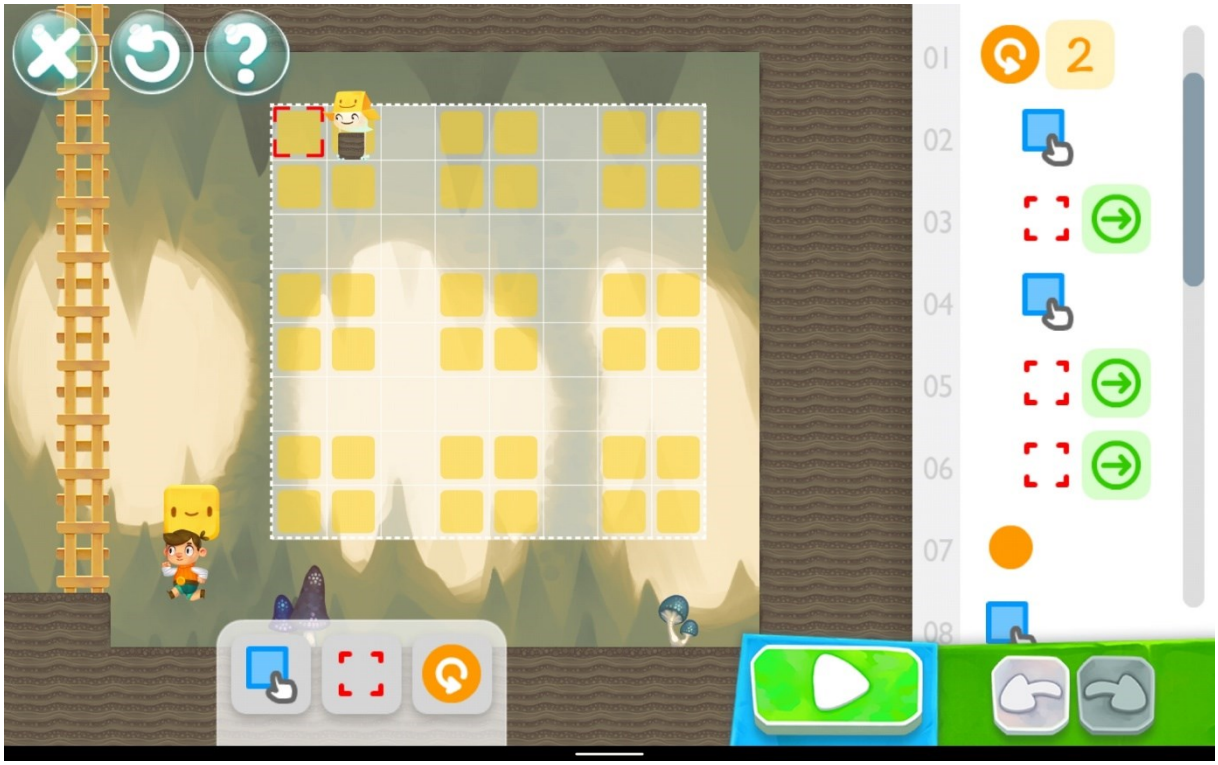
Aplikace je rozdělena na čtyři úrovně, každá obsahuje deset cvičení. V první pouze stavíme cestu pro postavičku, pro starší žáky je tedy možné tuto oblast uplně přeskočit či z ní udělat minimum cvičení. Tématem druhé úrovně jsou sekvence příkazů, ve třetí úrovni aplikace představuje cykly a ve čtvrté funkce. Narozdíl od výše zmiňované aplikace Codemonkey řešení cvičení rychle ubíhá a je zvládnutelné za jednu školní hodinu.

3/5 bodů

Celkový počet bodů 26/34

Výhodou hodnocené aplikace je, že funguje i offline. Pokud ji spustíme a zároveň je na tabletu zapnuta wifi, budou nám průběh hodiny přerušovat reklamy, které se po vypnutí wifi přestanou zobrazovat. Všechny úrovně jsou již na startu aplikace otevřené, takže si můžeme vybrat přímo téma, které nás zajímá.

6.2.7 SpriteBox



Obrázek 5 Vzhled aplikace SpriteBox

Operační systém

Hodnocenou aplikaci si lze stáhnout na tablety s operačními systémy Android a Mac OS.

Typ programování

Hodnocená aplikace používá programování pomocí bloků s ikonickým vyjádřením programové instrukce.

Podpora češtiny

Hodnocená aplikace nemá podporu češtiny.

0/5 bodů

Nutnost registrace

Hodnocená aplikace nevyžaduje registraci, k jejímu používání potřebujeme pouze Google účet, který je ve většině případů žáci základních škol již od školy zařízený.

2/2 bodů

Nutnost instalace

Hodnocená aplikace se nachází na Appstore/Google play obchodu, je nutné ji nainstalovat na zařízení.

0/2 bodů

Srozumitelnost zadání

Ačkoliv bloky s ikonickým vyjádřením programové instrukce nepoužívají jakýkoliv jazyk, instrukce jsou v angličtině na úrovni A2 a kolikrát jsou i docela dlouhé. V hodnocené aplikaci se nanachází jakákoliv forma nápovědy či korektury napsaného kódu. I když se uvnitř jednotlivých světů můžeme pohybovat skoro stylem jako ve videohře s otevřeným světem, uživatel se zde neztratí.

2/5 bodů

Vhodnost pro věkovou kategorii

Hodnocená aplikace je vhodná pro cílovou skupinu žáků, neboť je na takové úrovni, jakou zvládnou žáci pochopit. Plusem této aplikace je to, že se jedná o zajímavou, chytlavou a zábavnou formu učení se.

5/5 bodů

Uživatelská přívětivost

Ovládání aplikace je jednoduché, na úvodní obrazovce se nachází pouze tlačítko pro zapnutí a vymazání postupu uživatele. V aplikaci se nacházejí čtyři světy podobné klasickým hrám typu „skákačky“, které reprezentují oblasti, kde se probírá určitý programátorský koncept. Uživatel ovládá postavu, jejíž vzhled na začátku hry vybere. S postavou poté přejde ke krabicím, které reprezentují portál do jednotlivých světů a zmáčkne šipku nahoru – nyní se postava přesunula do vybraného světa a opět s ní může hráč pohybovat pomocí šipek na displeji tabletu, postava také může skákat.

5/5 bodů

Design

Design hodnocené aplikace je velmi povedený a příjemný, využívá zářivých barev a krásně nakreslených postav. Celkově aplikace svým designem předčí i některé videohry. Každý ze světů, které jsou uživateli k dispozici na prozkoumání má dokonce svůj specifický styl.

5/5 bodů

Probírané programátorské koncepty

Hodnocená aplikace bohužel projde poměrně malý počet programátorských konceptů – příkaz, sekvence příkazů, cyklus a vnořený cyklus, ačkoliv uvnitř aplikace řešíme poměrně velké množství cvičení.

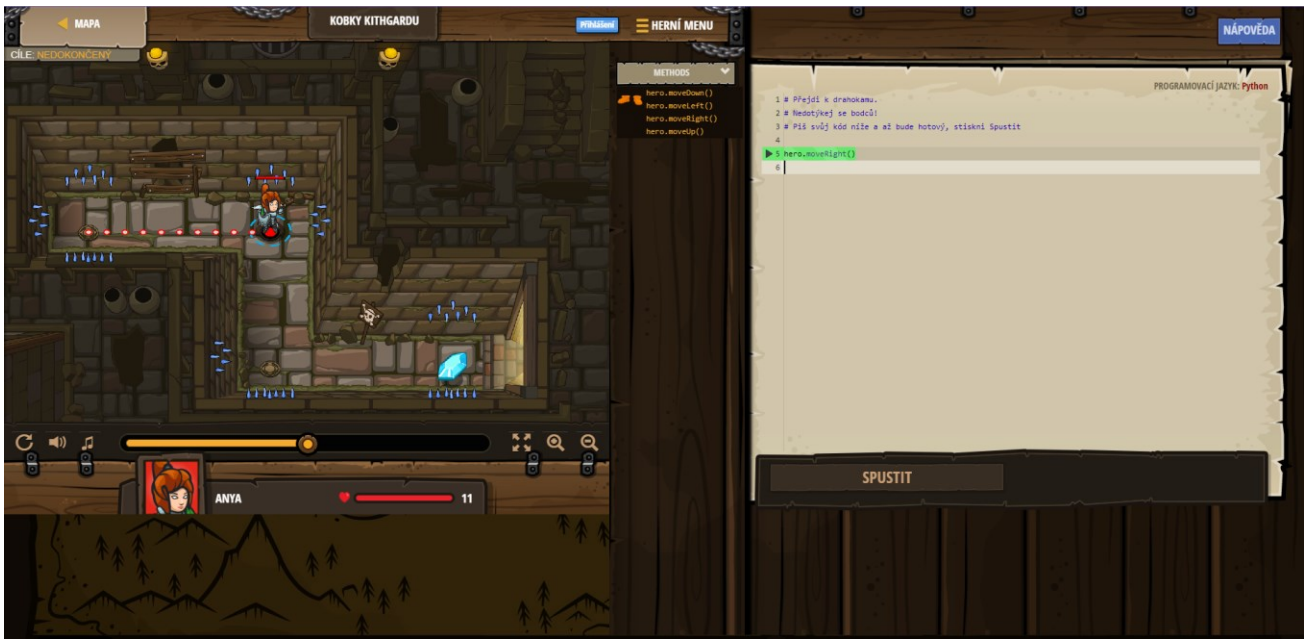
3/5 bodů

Celkový počet 22/34

V hodnocené aplikaci se nacházejí dvě postavy - postava hráče, kterou ovládáme pomocí šipek a ikony skoku na obrazovce zařízení a robot, kterého programujeme pomocí bloků s ikonickým vyjádřením programové instrukce. Postava cestuje skrze jeden ze čtyř možných světů a cestou sbírá hvězdičky a osvobozuje robotovi kamarády. Na ně většinou hráč nedoskočí ani k nim není možné dojít, takže musíme naprogramovat robota aby hráčovi postavil cestu a správně umístil či odstranil bloky.Konec úrovně či oblasti, vzhledem k hodnocené aplikaci lépe nazývaný světem, je označem vlaječkou. Pokud hráč sesbírá dostatečné množství hvězdiček a zachrání daný počet robotových kamarádů, otevře se další svět.

Pokud je potřeba na určitém místě v aplikaci využít robotových služeb, hodnocená aplikace nám to dá najevo specifickou ikonkou, u které pouze stačí pokud se jí hráč dotkne. V tu chvíli se část aplikace uvnitř které budeme umisťovat/odstraňovat bloky změní na robotův pohled a ukáže se nám samozřejmě i lišta, kam umisťujeme bloky s ikonickým vyjádřením programové instrukce.

6.2.8 CodeCombat



Obrázek 6 Vzhled aplikace CodeCombat

Operační systém

Hodnocená aplikace je webová aplikace, spustíme ji tedy na většině zařízení.

Typ programování

V hodnocené aplikaci máme možnost buď psát příkazy na klávesnici celé či napsat pouze jejich část a poté vybrat vhodný příkaz z nabídky našeptávače.

Podpora češtiny

Většina hodnocené aplikace je přeložená do češtiny, ale občas se zde objevuje podivná směs nepřeloženého anglického textu s náhodně umístěným českým slovíčkem. Tato situace nastane například pokud nás aplikace bude nutit si koupit předplatné. Příkazy, pomocí kterých kód tvoříme jsou v jednoduché angličtině a používá slova, které většina žáků již zná z videoher.

4/5 bodů

Nutnost registrace

Registrace uživatele je nutná v hodnocené aplikaci pouze pokud chceme uložit jeho postup.

1/2 bodů

Nutnost instalace

Hodnocená aplikace je webová aplikace, není potřeba ji instalovat.

2/2 bodů

Srozumitelnost zadání

Hodnocená aplikace zadá uživateli lehce srozumitelný úkol, opět se jedná o pohyb postavy. Nápověda je celkem dobře zpracovaná, dokonce kontroluje kód i předtím než ho spustíme. Jediným případem, kdy se neosvědčí je v případě, že uděláme chybu v syntaxu příkazu, například napíšeme špatně anglické slovíčko. Další nevýhodou co se týče zadání je nejasná informace, jak dlouhý je v aplikaci jeden krok postavy. Nejsou zde jasně nakreslené bloky, takže délku pohybu musí uživatel uhádnout.

3/5 bodů

Vhodnost pro věkovou kategorii

Hodnocená aplikace je vhodná pro cílovou věkovou kategorii žáků, má vhodné téma, které zaujme většinu žáků základních škol. Dokonce by se dala použít i pro starší žáky – pro ně by se mohla aplikace celá přepnout do angličtiny.

5/5 bodů

Uživatelská přívětivost

V hodnocené aplikaci není těžké se vyznat, nejtěžší výzvou může být nalezení spuštění hlavolamu na webu, kam se uživatel přihlásí - v tomto případě pouze stačí nalézt odkaz či tlačítko s nápisem „play“.

5/5 bodů

Design

Hodnocená aplikace je inspirován vzhledem her na hrdiny s prvky fantasy, konkrétně té kategorie her na hrdiny, které se odehrávají ve středověku. Její design připomíná dětmi stále oblíbenou online hru Shakes and Fidget.

5/5 bodů

Probírané programátorské koncepty

Hodnocená aplikace žákům představí zejména objektové programování. Pro přípravu na programování nám budou stačit první dvě oblasti, s názvy Kobka Kithgardu a Les zapadákov. V oblasti Kobka Kithgardu si uživatel projde témata jako syntaxe, metody, parametry, řetezce, cykly a proměnné. V oblasti Les zapadákov aplikace probírá témata jako podmínky, relační operátory, vlastnosti objektů a zpracování vstupu od uživatele. V obou oblastech se dohromady nachází 166 cvičení. Celkově by řešení těchto dvou oblastí zabralo dle informace na webových stránkách aplikace tři až devět hodin, což je pro potřeby přípravy na programování příliš dlouho.[33]

3/5 bodů

Celkový počet bodů 28/34

V hodnocené aplikace si uživatel vybere hrdinu, za kterého bude hrát, má možnost vybírat z hrdinů základních a hrdinů, kteří jsou součástí předplatného. Již zde hodnocená aplikace naznačuje,že předplatné je v podstatě nutností, aby fungovala. Tento výběr ovlivní jak moc složitá hra bude, neboť postavy se liší co se týče síly, počtu životů, zda používají magii a podobně. Výběrem postavy se rovněž změní příkazy pro útok, jaké používá. Opět se zde využívá klasického typu hlavolamu: programujeme pohyb postavy, přičemž se určitých věcí dotknout má a jiným se má vyhnout, na některé postavy může dokonce zaútočit.

[33] CodeCombat - výběr oblasti. In: *CodeCombat* [online]. [cit. 2022-05-17]. Dostupné z: https://codecombat.com/play

Aplikace se skládá z oblastí, každá se zabývá jiným programátorským tématem. Pro potřeby přípravy na programování budou stačit dvě výše zmíněné, které se jmenují Kobka Kithardu a Les zapadákov. Bohužel i v tomto případě, kdy chceme využít minimální potenciál hodnocené aplikace, budeme jakožto uživatelé donuceni koupit si předplatné minimálně na měsíc - CodeCombat neplatící uživatele nepustí dále než do pátého cvičení oblasti Kobka Kithardu.

6.2.9 Závěrečné doporučení

Aplikací s nejvíce body je Minecraft: Cesta hrdiny. Hned po ní na druhém místě, se umístily aplikace Kids Coding Skills a Monster High: Scavenger Hunt. Právě z těchto aplikací budeme tvořit možné kombinace pro přípravu na programování.

V přípravě na programování trvající pouze jednu školní hodinu je vhodné použít buď aplikaci Minecraft: Cesta hrdiny. V případě, že škola vlastní tablety je doporučeno použít aplikaci Kids Coding Skills. Pokud se rozhodneme použít aplikaci na tablety pracujte s zařízeními v offline módu, nezapínejte na nich wi-fi.

V případě na přípravy na programování trvající dvě školní hodiny je doporučeno použít opět buď aplikaci Minecraft: Cesta hrdiny či Kids Coding Skills. Projít si tyto aplikace by mělo trvat zhruba čtyřicet minut a po přestávce žákům představíme aplikaci Monster High: Scavenger Hunt. I když je tato aplikace vzhledem orientovaná spíše na dívčí část třídy, naučí žáky pracovat s odlišným prostředím a představí jim další programátorské koncepty, které se v dalších hodnocených aplikací nenacházejí – kromě již probraných cyklů má v repertoáru i podmínky a smyčky „while“. Právě tato její vlastnost a fakt, že v předchozí hodině si již žáci prošli aplikaci s totožným cílem, naprogramovat pohyb postavy a cestou sbírat věci či naopak se určitým věcem vyhýbat, z ní dělá vhodného kandidáta na aplikaci probíranou druhou hodinu přípravy na programování i navzdory použití pokročilejší angličtiny.

Část II

Praktická část

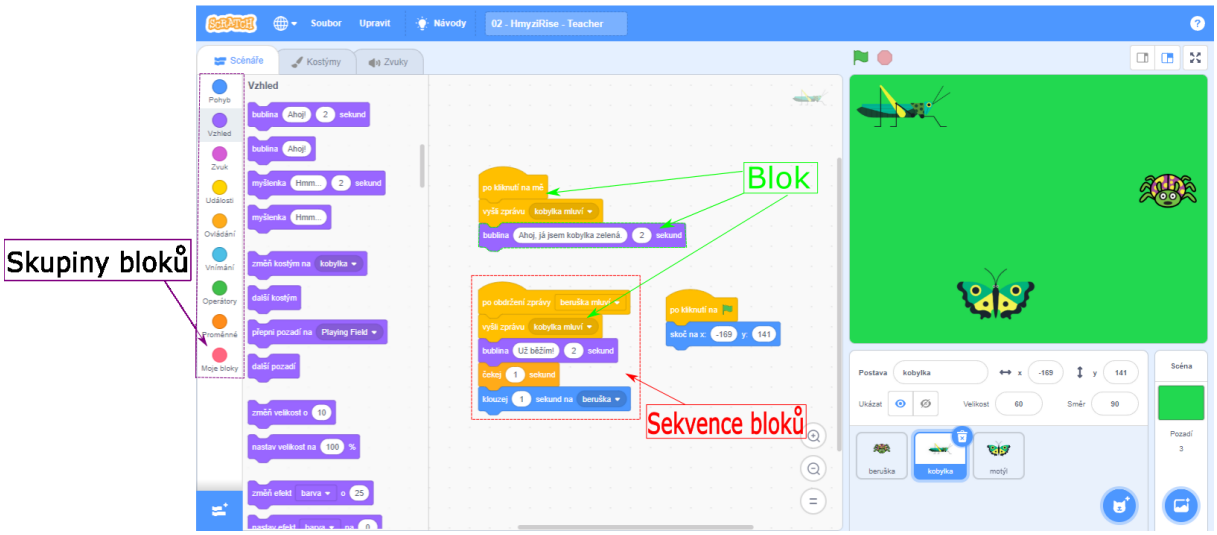
7. Soubor lekcí

7.1. Úvod do praktické části

Soubor lekcí je možné si prohlédnout v přiloženém souboru formátu zip, jmenuje se „souborLekci.zip“. Najdeme zde devět lekcí, ve kterých se nachází dvacet čtyři scénářů. Stejný scénář je zde dvakrát, jednou jako výchozí scénář se kterým žáci začínají a jednou jako dokončený scénář pro učitele.

Scénáře pro učitele se jmenují „čísloLekce - názevScénaře – Teacher.sb3“. Scénáře pro žáky se nazývají „čísloLekce – názevScénaře – Student.sb3“.

7.2. Slovníček pojmů



Obrázek 7 Poloha použitých pojmů týkajících se bloků v programu Scratch

Blok

Bloky jsou základní stavební kameny blokového programování nejen v programu Scratch, ale ve všech programovacích jazycích, které mají blokovou strukturu. Uvnitř bloků se nacházejí příkazy či deklarace, bloky mohou obsahovat i vnořené bloky. Místo psaní příkazů pomocí klávesnice uživatel umístí bloky pomocí myši na předem vymezenou plochu programu (v této práci plochu nazýváme prostor scénáře).[34]

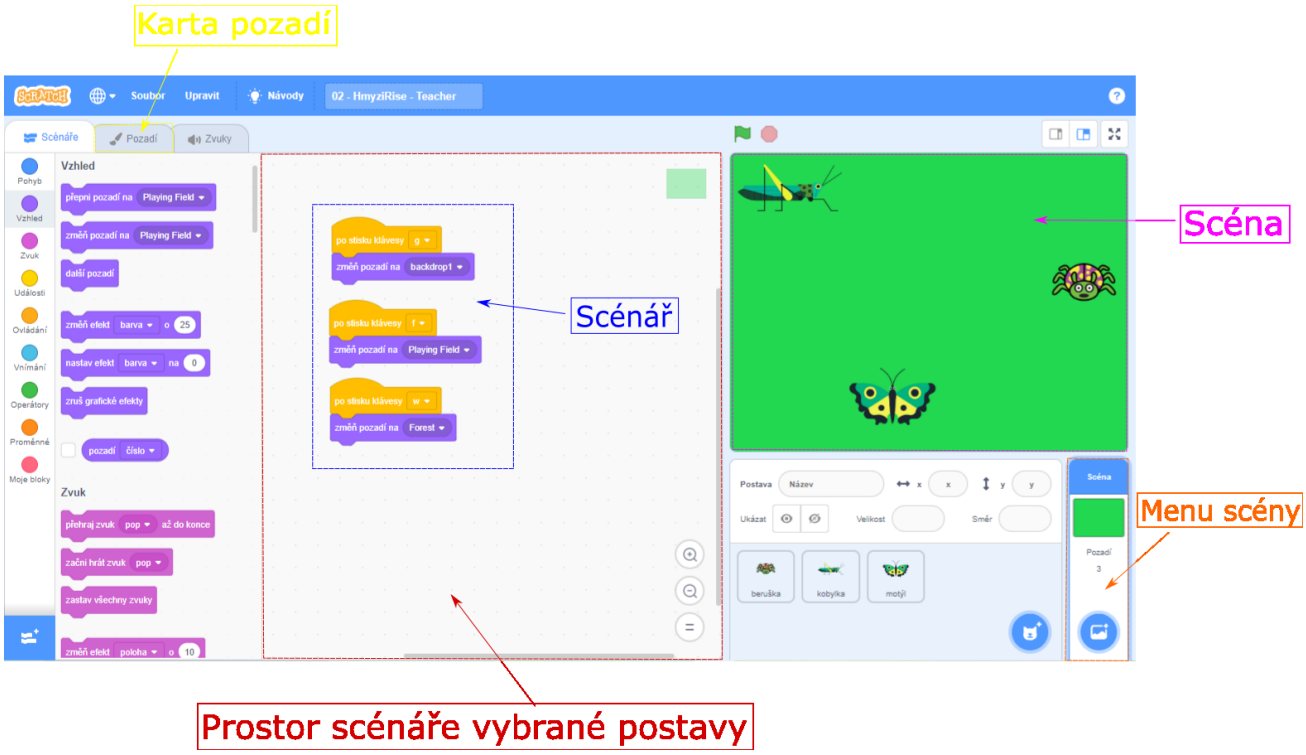
Sekvence bloků

Pokud umístíme několik bloků pod sebe, vytvořili jsme sekvenci bloků. V pořadí v jakém jsou bloky umístěné za sebou se zpracují.

Skupina bloků

Pro lepší orientaci v programu Scratch jsou bloky tématicky rozdělené do skupin. Každá skupina a její členské bloky jsou barevně odlišeny.

[34] *Block* [online]. [cit. 11.5.2022]. Dostupný na WWW: https://www.techopedia.com/definition/17978/block-programming



Obrázek 8 Poloha použitých pojmů týkajících se scény a scénáře v programu Scratch

Prostor scénáře vybrané postavy

Pokud máme z menu postavy vybranou určitou postavu, v prostoru scénáře uvidíme bloky, které k ní patří. V případě, kdy kliknutím levého tlačítka myši vybereme menu scény se zde zobrazí bloky patřící ke scéně.

Scénář

Scénář jsou jakékoliv bloky, které se nacházejí v prostoru scénáře, ať už u postav nebo se zobrazí po kliknutí na menu scény.

Menu scény

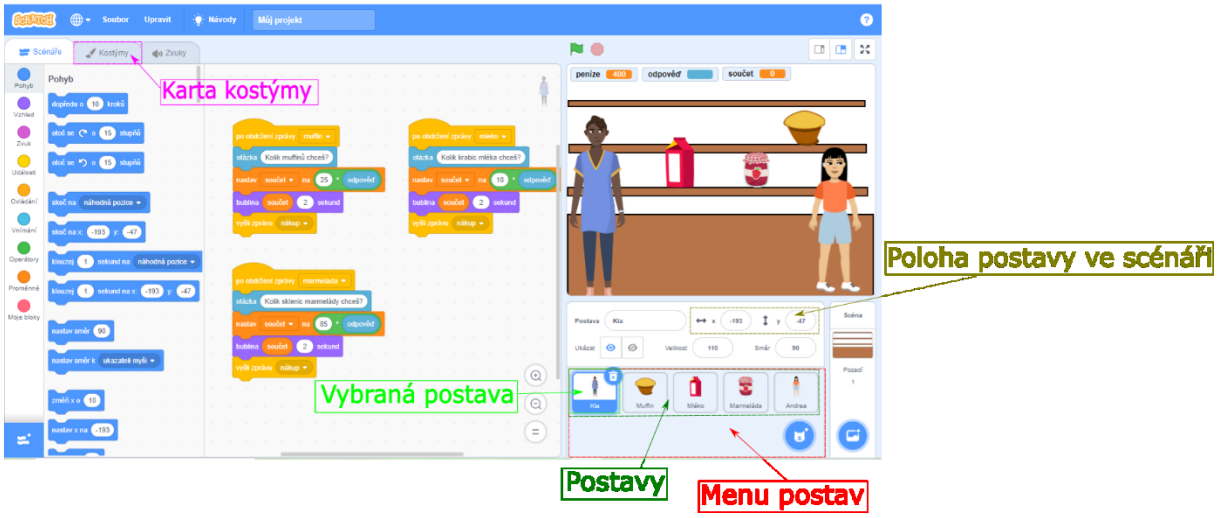
Pokud kliknutím levého tlačítka myši vybereme menu scény, v prostoru scénáře uvidíme prostor scénáře scény. Mohou se zde například nacházet bloky, které uvedou scénář do základního nastavení.

Karta pozadí

Po kliknutí na tlačítko s nápisem Pozadí s logem štětce se nám zobrazí všechna pozadí, které má scénář k dispozici. Můžeme zde také pozadí přidat či namalovat vlastní.

Scéna

Ve scéně vidíme výstup programu.



Obrázek 9 Poloha použitých pojmů týkajících se postav v programu Scratch

Postava

Postavy jsou „herci“ ve scénáři, mohou vnímat a pohybovat se v rámci scénáře.

Vybraná postava

Programovat můžeme vždy pouze jednu postavu či scénu. Kliknutím levého tlačítka myši ji vybereme a v prostoru scénáře se nám zobrazí bloky a sekvence bloků, které k ní patří. Vybraná postava má v programu Scratch modrý rámeček.

Menu postav

V menu postav vidíme všechny postavy, které se nacházejí ve scénáři a také zde máme možnost postavy přidat pomocí modrého tlačítka.

Poloha postavy ve scénáři

Postava se ve scénáři nachází vždy na určitých souřadnicích kartézské soustavy scénáře.

Karta kostýmy

Jedna postava může mít více kostýmů, podobně jako herci v divadle. Pokud kliknutím levého tlačítka myši vybereme tuto kartu a máme zároveň vybranou postavu, zobrazí se nám všechny kostýmy, které má postava k dispozici. Opět je zde možnost přidat další kostým pomocí modrého tlačítka.

7.3. Obecné zásady pro učitele

1. Na začátku hodiny na interaktivní tabuli či plátno promítněte hotový program. Popište, co se v programu děje a co dělá uživatel – pokud v něm budeme například pohybovat s želvou, vysvětlete že po stisknutí klávesy šipka vlevo se želva otočí. Pokud má scénář několik možných situací, které mohou nastat (například u Obchodu – máme dostatek peněz nebo nemáme, či Draka a rytíře – zvítězit mohou oba), ukažte obě situace. Instrukce pro žáky tak budou dávat větší smysl.
2. Instrukce pro žáky je vhodné budˇ vytisknout a žákům rozdat či v případě dvou projekčních zařízeních ve třídě zobrazit na zařízení číslo jedna a na druhém zadání řešit.
3. Pokud je v učitelském návodu napsáno že řešení bodu úlohy máme ukázat na tabuli, použijte výše zmíněný projektor či interaktivní tabuli. V textu se nacházejí i obrázky týkající se teorie, ty je možné nakreslit na klasickou či interaktivní tabuli.

7.4. První lekce – ŽelvaV1 instrukce pro žáky

V dnešní lekci si rozpohybujeme želvu.

1. Blok s vlaječkou ponechejte v programu a nemažte ho.
2. Po stisknutí klávesy šipka vpravo ->želva se otočí vpravo.
3. Obdobně naprogramujte otáčení želvy pro ostatní směry (vlevo, dopředu, dozadu).
4. Jako každá správná želva by se i ta naše měla pohybovat. Stane se tak po stisknutí klávesy w, naprogramujte si řešení a vyzkoušejte.

Nepočítačový úkol: Jak by vypadal program pro nakreslení trojúhelníku?

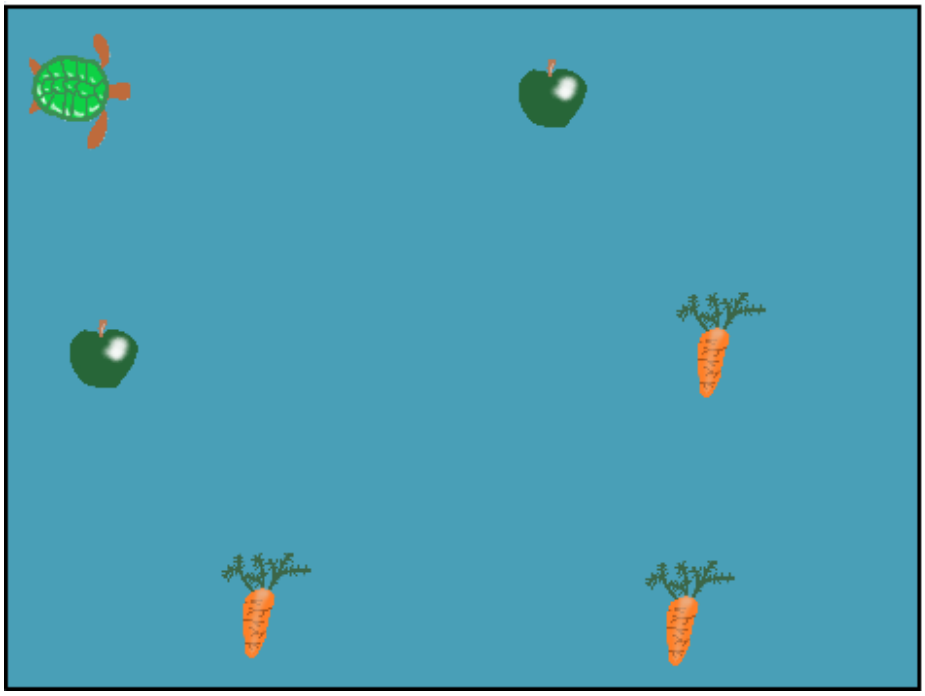
Úkol navíc: Zkuste si naprogramovat želvu, aby se po stisku šipky nejen natočila odpovídajícím směrem, ale navíc ještě popošla 10 kroků.

Použité skupiny bloků: **Pohyb**, **Události**

7.5. První lekce – ŽelvaV1 (učitelský návod)

Popis scénáře

V této lekci si ukážeme ovládání želvy pomocí kláves.



Obrázek 10 Vzhled scénáře ŽelvaV1

Použité programátorské koncepty

Příkaz – nový

Události - nový

Nově představené bloky z programu Scratch

Události - Po stisku klávesy

Pohyb – Nastav směr

Pohyb – Dopředu o x kroků

Využité bloky

Bloky týkající se pohybu

Použitý projekt

01 – ŽelvaV1 – Teacher.sb3	soubor pro učitele
01 – ŽelvaV1 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Poté, co žáci otevřou 01 – ŽelvaV1 – Student.sb3, uvidí ve scénáři sekvenci bloků. Nyní bychom měli vysvětlit k čemu přesně tato sekvence a jí podobné slouží – její účel je umístění želvičky na začátek mapy při spuštění scénáře, jak značí zelená vlaječka.

Na tento důležitý prvek ovládání Scratche můžeme upozornit obkroužením myší jak u bloku 

, tak i u tlačítka s vlaječkou nad scénou. Scénář zachytí kliknutí na zelenou vlaječku a spustí příslušné příkazy, které se nacházejí pod výše zobrazeným blokem. Dále v tomto dokumentu budeme tomuto typu bloku říkat **Po kliknutí na zelenou vlaječku**.

Možné problémy

Během ověřování se stalo, že si děti aktivně tyto předpřipravené bloky vymazali a pak se jim želva ztratila ze scénáře. Je dobré mít na serveru, ze kterého si zadání berou k sobě na plochu, vždy zálohu zadání.

2. Po stisknutí klávesy šipka vpravo ->želva se otočí vpravo.

Metodická poznámka:

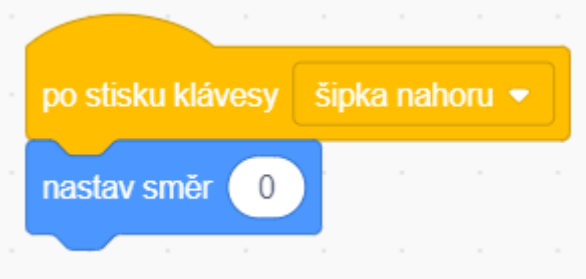
Ještě než se pustíme do samotného programování, je nutné vysvětlit co to vlastně je základní stavební kámen programování – tedy příkaz. Můžeme jej přirovnat například k jednomu kroku výroby nějakého předmětu či k jednomu ze kroků receptu. Program se podobně jako postup skládá z vícero příkazů, je to návod jak někdo nebo něco postupuje. Ve Scratchi program nazýváme scénář a zadáváme jeho použitím želvě (či jakékolic jiné postavě) instrukce, podle kterých se chová. Příkazy se ve Scratchi nacházejí ve formě bloků, budeme je tedy nazývat bloky.

Na tabuli můžeme například začít kreslit první čáru čtverce a říct, že první příkaz pro jeho nakreslení by zněl: jdi dopředu. Můžeme použít libovolnou techniku získávání odpovědi od žáků a postupně si s nimi čtverec „naprogramovat“.

Řešení 2.bodu úlohy

Ze skupiny bloků **Události** vybereme **Po stisku klávesy** a umístíme jej do prostoru scénáře. V menu bloku **Po stisku klávesy** si vybereme na jakou klávesu vlastně bude želva reagovat, vybrat můžeme například šipku nahoru.

Nyní se přepneme do skupiny bloků **Pohyb**. Zde vybereme blok **Nastav směr** a na kruhu nastavíme jakým směrem se želva po stisku klávesy otočí. Obdobně nastavíme pohyb pro ostatní směry, nám stačí pouze 4 základní – nahoru, dolů, vlevo a vpravo.

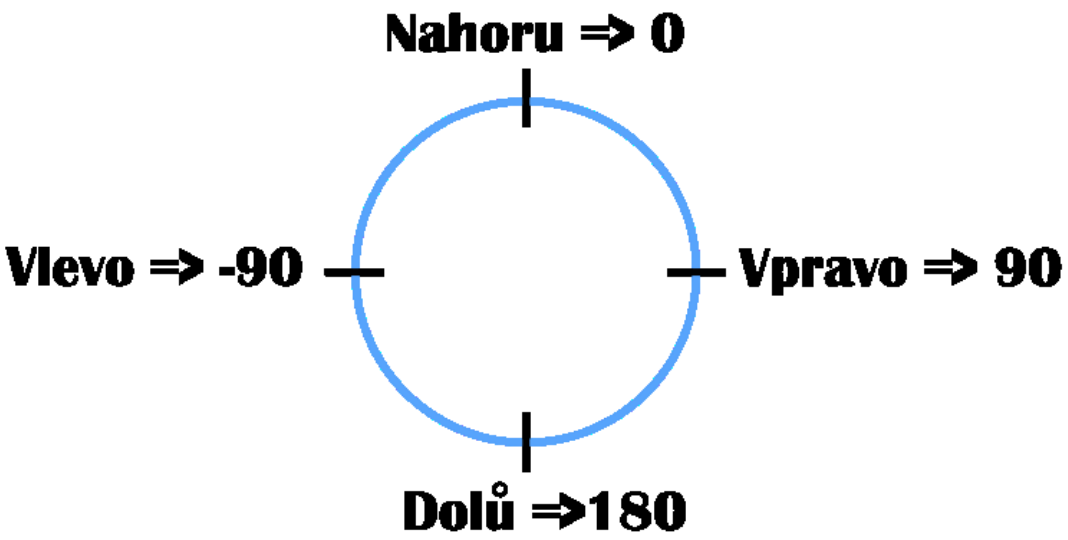


Obrázek 11 Řešení 2.bodu úlohy

Metodická poznámka

Nyní je čas představit žákům nový koncept v programování - událost. Událost je podnět z okolí, na který ve scénáři reagují postavy nebo scéna samotná.Tento podnět je ve většině případů vyvolán uživatelem a reagují na něj pouze postavy, kterých se to týká – tedy postavy, které jsou naprogramované na tento podnět reagovat. Jako případ můžeme použít situaci, kdy někdo na ulici zakřičí naše křestní jméno a my se otočíme, zda toho člověka neznáme. I když se zavolání netýkalo nás, na tuto událost jsme zareagovali otočením.

S událostí jsme se setkali již na začátku hodiny v bloku **Po kliknutí na zelenou vlaječku**. Podnětem zde je kliknutí na zelenou vlaječku. Podobně funguje i blok **Po stisku klávesy**, který zachytává podnět stisku klávesy specifikované uvnitř bloku. Jakmile tuto klávesu uživatel stiskne, scénář reaguje vykonáním bloků, které se nacházejí pod blokem **Po stisku klávesy**.



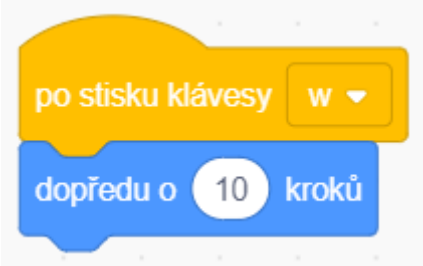
Obrázek 12 Příklad nakresleného příkazu Nastav směr na tabuli

Dále je vhodné nakreslit kruh z výběru **Nastav směr** na tabuli a zakreslit na něj, kde leží námi programované směry. Měli by jsme zmínit, že je dobré si výběr směru v rámci kruhu sám vyzkoušet a pozorovat, jak na to želva reaguje. Příklad, jak by kruh mohl vypadat se nachází nad tímto textem.

3. Jako každá správná želva by se i ta naše měla pohybovat. Stane se tak po stisknutí klávesy w, naprogramujte si řešení a vyzkoušejte.

Řešení 3.bodu úlohy

Obdobně jako jsme naprogramovali otáčení přiřadíme ke klávese „w“ blok **Dopředu o 10 kroků** z **Pohybu**. Celý scénář si spustíme znovu pomocí stisku zelené vlaječky, tím resetujeme pozici želvy a můžeme vyzkoušet novou naprogramovanou funkcionalitu scénáře.



Obrázek 13 Řešení 3.bodu úlohy

Metodická poznámka

S touto částí zadání většinou moc problémů nebývá. Během konstrukce celého scénáře je možné že si žáci zvětší želvu či jinou postavu do obřích rozměrů, případně si je vymažou. Měli bychom zdůraznit důležitost pravidelného ukládání, ukázat postup na interaktivní tabuli či projektoru a klidně jej několikrát za hodinu připomenout – i když na školním serveru můžeme mít původní verzi počátečního stavu scénáře, žáci by vymazáním postavy či snad neuložením celého scénáře ztratili výsledek celého svého snažení a mohlo by je to demotivovat. Uložení provedeme kliknutím na tlačítko Soubor ze základního menu Scratche, dále vybereme Ulož do svého počítače a vhodné umístění. Soubor lze přepsat i když jej samotný máme v programu Scratch otevřený, stačí pouze vybrat jakožto místo uložení právě otevřený scénář v jeho příslušném adresáři.

Závěr

Žáci by měli nyní chápat, jak fungují události na základě stisknuté klávesy a jak používat základní příkazy týkající se pohybu postavy.

7.6. Druhá lekce – Hmyzí říše instrukce pro žáky

V dnešní lekci si naprogramujeme rozhovor mezi členy hmyzí říše.

A. Počáteční bloky s vlaječkou ponechejte v programu a nemažte je.
B. V programu se nacházejí tři postavy hmyzu. Naprogramujte je podle tabulky. Ještě než začnete pracovat na řešení, vložte si do programu bloky z událostí, se kterými budeme pracovat.

Beruška (po kliknutí na ni)	1. Změní kostým na beruska-mluvici. 2. Řekne, že má zprávu pro kobylku. 3. Počká jednu sekundu. 4. Změní kostým na beruska-klid. 5. Vyšle zprávu „beruška mluví“..
Kobylka (po kliknutí na ni)	6. Vyšle zprávu „kobylka mluví“. 7. Představí se.
Kobylka (po obdržení zprávy „beruška mluví“)	8. Vyšle zprávu „kobylka mluví“. 9. Řekne, že už běží. 10. Počká jednu sekundu. 11. Posune se k berušce.
Motýl (po obdržení zprávy „kobylka mluví“)	12. Otočí se lehce nalevo. Zkuste si kolik stupňů vám přijde ideálních. 13. Počká jednu sekundu. 14. Otočí se zpátky do počáteční polohy.
Motýl (po obdržení zprávy „beruška mluví“)	To samé jako v situaci kdy mluví kobylka, ale v prvním kroku se otáčí doprava.

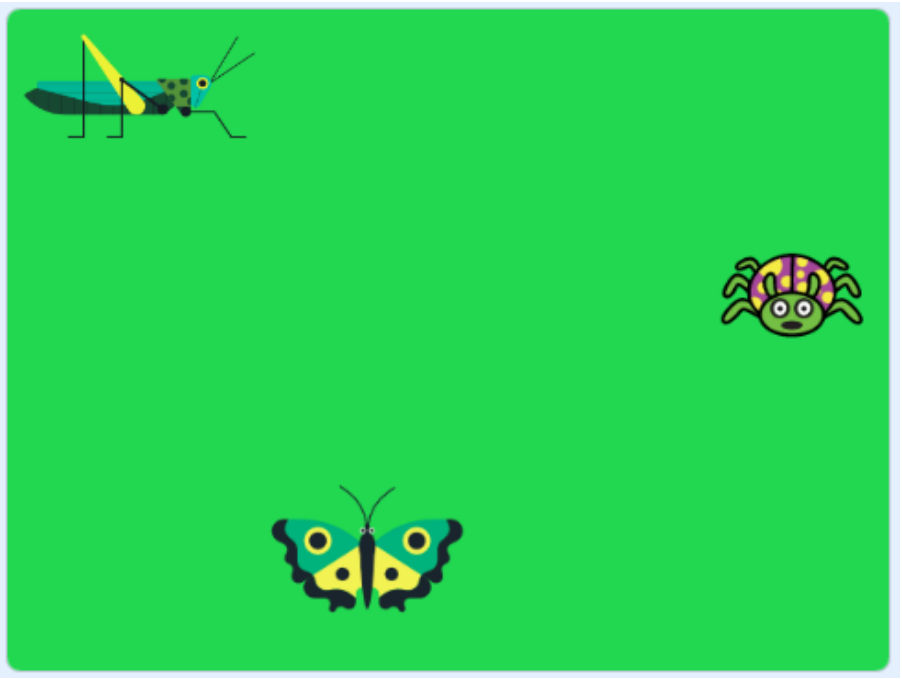
Úkol navíc: Zkuste si přidat čtvrtou postavu do scénáře. Tato postava se po kliknutí na ni představí a poté se posune k jiné z postav ve scénáři.

Použité skupiny bloků: **Pohyb**, **Události**, **Vzhled**, **Ovládání**

7.7. Druhá lekce – Hmyzí říše (učitelský návod)

Popis scénáře

V této lekci si vyzkoušíme další příkazy ze skupiny bloků **Pohyb** a naprogramujeme jednoduchý rozhovor postav.



Obrázek 14 Vzhled scénáře Hmyzí Říše

Použité programátorské koncepty

Příkaz

Události

Zprávy - nový

Nově představené bloky z programu Scratch

Události – Po kliknutí na mě

Události – Vyšli zprávu

Události – Po obdržení zprávy

Vzhled – Změň kostým na

Vzhled – Bublina po dobu x sekund

Pohyb - Klouzej x sekund na

Pohyb – Otoč se o x stupňů doleva/doprava

Využité bloky

Bloky týkající se zpráv, dialogů postav a kostýmů, bloky zabývající se pohybem

Použitý projekt

02 – HmyziRise – Teacher.sb3	soubor pro učitele
02 – HmyziRise – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Pro začátek bude důležité vysvětlit způsob, jakým jsou úkoly zadávány. Levou stranu tabulky si rozdělíme na dvě části, před závorkou a obsah závorky. Před závorkou je název postavy, se kterou budeme pracovat. Uvnitř závorky je blok ze skupiny bloků **Události**. Tento příkaz značí událost, na kterou scénář čeká a jakmile tato událost nastane, budou následovat příkazy na levé straně tabulky, viz. minulá lekce a zachytávání událostí zde zmíněné. Tento styl zadávání úkolů je vhodné třídě přiblížit. Ze začátku je používán, jelikož se žáci s událostmi setkávají poprvé a tento způsob zápisu je velmi názorný. Později je od něho odpuštěno, protože se předpokládá znalost principů událostí.

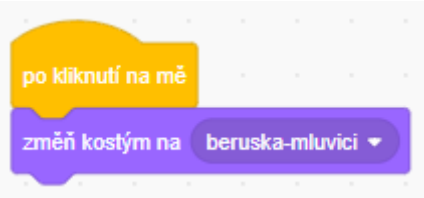
1. Změní kostým na beruska-mluvici.

Metodická poznámka

V každém scénáři se nacházejí herci, to jsou v našem případě postavy jako v minulé lekci želva či v nynějším hmyz. Postavy ovšem nemusejí mít pouze jednu podobu. Pokud kliknutím vybereme jednu z postav, zpřístupní se nám nad skupinami bloků karta Kostýmy, jako logo má Obrázek štětce. Po kliknutí na ni vidíme všechny kostýmy, které má vybraná postava k dispozici. Kostýmy v tomto podmenu můžeme přidávat, kreslit na ně a podobně, a i proto je lepší žákům pouze ukázat, jaké kostýmy jsou k dispozici a více se o tuto kartu nyní nezajímat. Žáci na to stejně přijdou sami a zdlouhavé vybírání postav či dokonca nakreslení vlastní by narušilo hodinu.

Řešení 1.bodu úlohy

Ze skupiny bloků **Vzhled** vybereme příkazový blok **Změň kostým na**. V nabídce jsou jasně pojmenované kostýmy.



Obrázek 15 Řešení 1.bodu úlohy

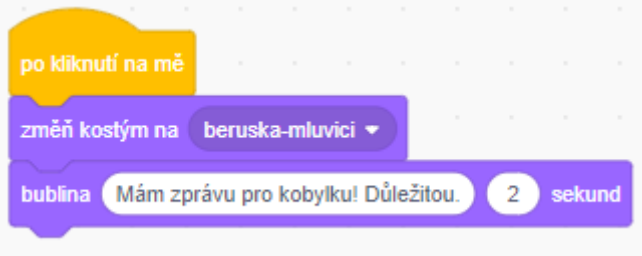
Metodická poznámka

Další připomínka směrem k žákům může být ohledně důležitosti jednoznačného pojmenování kostýmů a postav. Tato praktika je v programování důležitá kvůli přehlednosti. Pokud budeme mít u postavy například šest různých kostýmů a budeme mezi nimi při různých událostech chtít přepínat, pouze číselné názvy nám nepomohou v jednoduchém rozhodování, jaký kostým u menu bloku **Změň kostým na** použít. Přejmenování jednoho z kostýmů by si měli žáci vyzkoušet, stejně jako přejmenování postavy.

2. Řekne, že má zprávu pro kobylku.

Řešení 2.bodu úlohy

Ze **Vzhledu** vybereme **Bublina „“ x sekund**. Do první mezery napíšeme text a do druhé čas jeho setrvání na obrazovce. Ideální je jedna až dvě vteřiny.

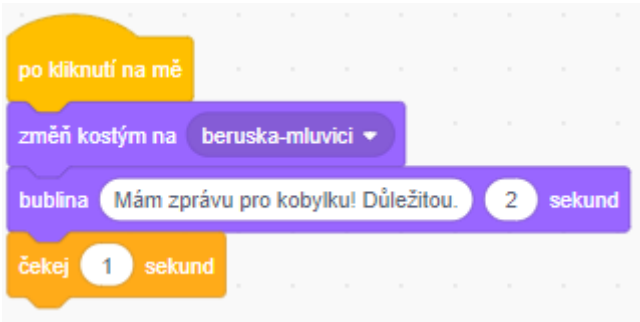


Obrázek 16 Řešení 2.bodu úlohy

3. Počká jednu sekundu.

Řešení 3.bodu úlohy

V skupině bloků **Ovládání** vybereme **Čekej x sekund**.



Obrázek 17 Řešení 4.bodu úlohy

Metodická poznámka

Řešení 4.bodu úlohy je žákům již známé a měli by si ho zkusit sami.

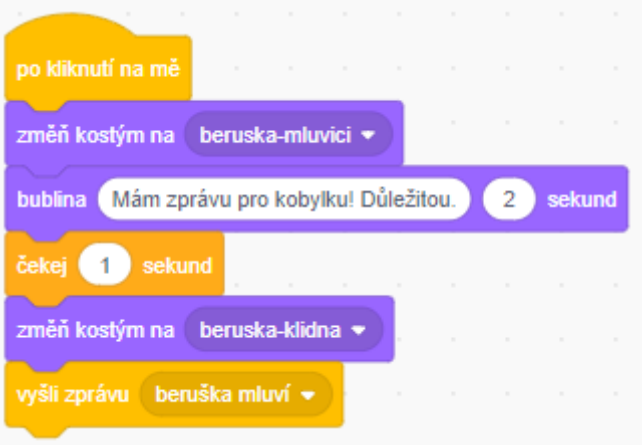
5. Pošle zprávu „Beruška mluví“.

Metodická poznámka

Nyní se dostáváme k hlavní náplni hodiny. Ještě než bude možné pracovat s programátorským konceptem zpráv, je nutné žákům tento koncept přiblížit v realitě. Mezi dobré příklady patří například barvy semaforu na křižovatce. Zprávou v tomto případě je barva světla, jaké právě svítí. Po přijetí zprávy, když si světla všimneme a zaregistrujeme jeho barvu, řidič buď v případě červeného světla zastaví, v případě oranžového zpozorní a v případě zeleného auto pokračuje v cestě. Zde je možné nechat žáky mezi sebou chvilku diskutovat ohledně toho, co je dobrým příkladem podobného chování v reálném životě. Třeba se právě od nich naučíme lepší vysvětlení než výše uvedené.

Řešení 3.bodu úlohy

Z **Událostí** vybereme blok **Vyšli zprávu**. Do mezery napíšeme, co přesně chceme poslat.



Obrázek 18 Řešení 3.bodu úlohy

Metodická poznámka

Nyní se vrátíme ke konceptu zpráv. V kartě **Události** vybereme **Po obdržení zprávy**. Do prázdného políčka vybereme tu zprávu, na kterou bude postava reagovat. Toto ukážeme i na tabuli. Necháme žáky samostatně dodělat všechny úlohy až do bodu 11.

11. Posune se k berušce.

Řešení 11.bodu úlohy

Z **Pohybu** vybereme **Klouzej x sekund na beruška**. V nabídce jsou všechny dostupné postavy i kurzor.

Obrázek 19 Řešení úlohy u kobylky

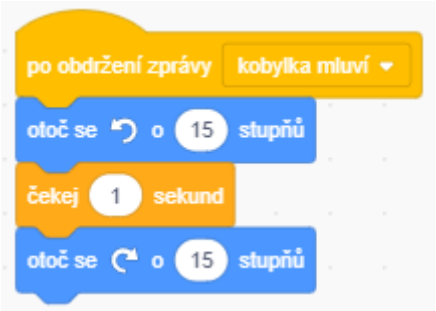
12. Otočí se lehce nalevo. Zkuste si kolik stupňů vám přijde ideálních.

Metodická poznámka

Vybereme si nějaké stupně, o které budeme postavu motýlka otáčet. Když ho budeme natáčet ke kobylce, otočí se o vybraný počet stupňů nalevo a poté zpátky do startovní polohy. Musíme poté tedy motýlka otočit o stejný počet stupňů na opačnou stranu.

Řešení 12.bodu úlohy

Pod blok s přijmutím zprávy o promluvy kobylky umístíme ze skupiny bloků **Pohyb** **Otoč se o x stupňů doleva**. Pod něj umístíme nám již známý blok s počkáním a poté následuje blok **Otoč se o x stupňů doprava**.



Obrázek 20 Řešení otáčení u motýla

Závěr

Žáci by nyní měli chápat koncept zpráv v programování a ovládat některé z příkazů týkající se vzhledu.

7.8. Třetí lekce – ŽelvaV2 instrukce pro žáky

V dnešní lekci si rozšíříme program želvy o ovoce a zeleninu, kterou bude želva jíst. Jako u každé správné hry, jídlo po snězení želvou zmizí.

1. Počáteční bloky s vlaječkou ponechejte ve scénáři a nemažte je.
2. Nyní si naprogramujeme krmení želvy – poté co se dotkne jídla, tak jablko či mrkve zmizí. Následující bloky přidáme k postavám, které mají kostým jídla.

Jídlo (po spuštění)	• Ukáže se. • Čeká, dokud se ho želva nedotkne. • Pokud se ho dotkne, tak se skryje.

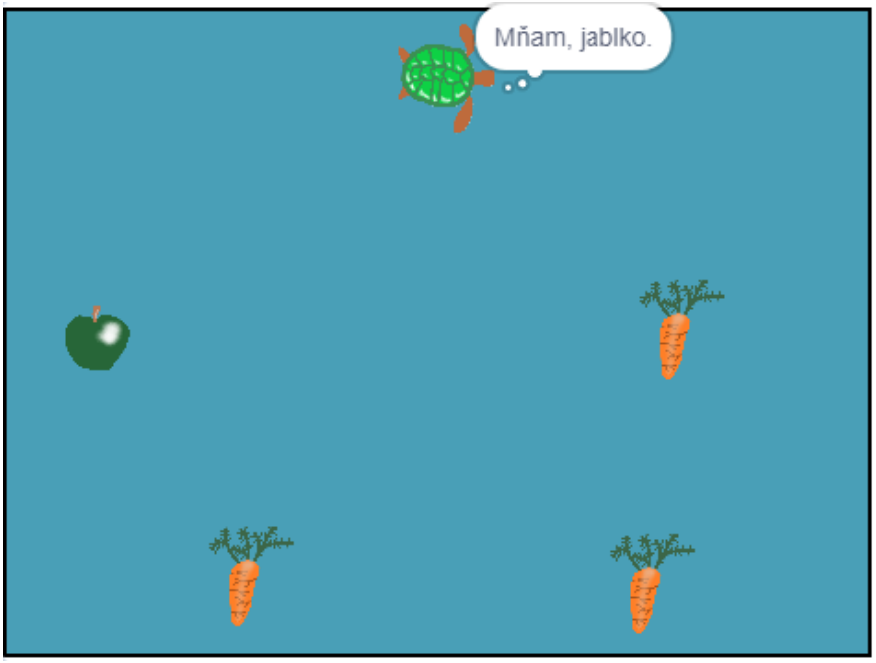
3. Pokud se želva dotkne jablka, řekne: Mňam jablko.
4. Pokud se želva dotkne mrkve, řekne: Mňam mrkev.

Použité skupiny bloků: Ovládání, Vzhled, Události, Vnímání

7.9. Třetí lekce – ŽelvaV2 (učitelský návod)

Popis scénáře

V této lekci si přidáme k již rozpohybované želvě interakci s prostředím – ovoce a zeleninu, které se dotkne, sní.



Obrázek 21 Scénář ŽelvaV2 s ukázkou nově naprogramované funkčnosti

Použité programátorské koncepty

Příkaz

Události

Zprávy

Nově představené bloky z programu Scratch

Vzhled – Myšlenka po dobu x sekund

Vzhled – Ukaž se

Vzhled – Skryj se

Vnímání – Dotýkáš se

Události – Po kliknutí na zelenou vlaječku

Ovládání – Čekej dokud nenastane

Využité bloky

Bloky týkající se zpráv, viditelnosti postavy, blok týkající se smyčky „while“, bloky vnímání

Použitý projekt

03 – ŽelvaV2 – Teacher.sb3	soubor pro učitele
03 – ŽelvaV2 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

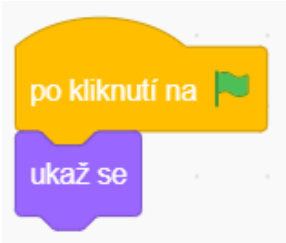
Nyní můžeme připomenout blok **Po kliknutí na zelenou vlaječku**. Ten se spustí poté, co zachytíme specifikovanou událost, v tomto případě kliknutí na zelenou vlaječku. Předtím jsme o tomto bloku pouze mluvili, nyní ho i použijeme. Zelená vlaječka v programu Scratch je vlastně virtuální klávesa, jediná v celém Scratchi, a proto má narozdíl od kláves na klávesnici vlastní blok.

Mezi klasické použití bloku **Po kliknutí na zelenou vlaječku** patří například reset pozic postav či obecně uvedení scénáře do počátečního stavu. Úplně stejně můžeme použít jakoukoliv jinou klávesu, jde tady spíše o zvyklost.

Ukáže se.

Řešení 2.bodu úlohy

Ze **Vzhledu** vybereme **Ukaž se** a přetáhneme na plochu pod blok s vlaječkou.



Obrázek 22 Řešení 1.bodu úlohy

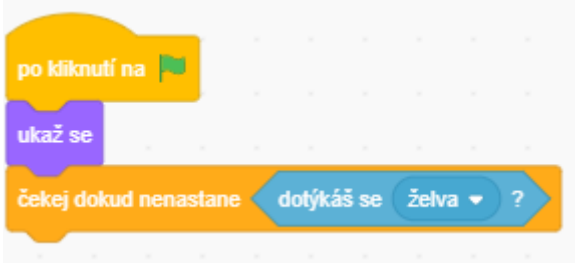
Čeká, dokud se ho želva nedotkne.

Metodická poznámka

Toto je první seznámení žáků s blokem **Čekej dokud nenastane** ze skupiny příkazů **Ovládání**. Jak již název napovídá, tento blok čeká, dokud nenastane programátorem zvolená událost. Pokud tato událost nastane, budou se konat příkazy pod tímto blokem.

Řešení 2.bodu úlohy

Nyní budeme pracovat se dvěmi novými ještě neprobranými skupinami bloků – **Ovládání** a **Vnímání**. Z **Ovládání** zvolíme výše zmíněný **Čekej dokud nenastane**. Do kosočtverce, kam zadáváme podmínku, přetáhneme z **Vnímání** blok **Dotýkáš se** a do prázdného místa vybereme název pro postavu želvy.



Obrázek 23 Řešení 2.bodu úlohy

Metodická poznámka

Nyní můžeme využít techniky read-write-share. Na tabuli můžeme napsat blok Čekej dokud nenastane a vymyslet vzorovou situaci, například kdybychom měli hru typu Space invaders. Můžeme krátce vysvětlit jak tato hra funguje – hráč, který ovládá vesmírnou loď střílí po návštěvnících z kosmu a snaží se je zlikvidovat. Máme tři životy, pokud se nás mimozemštané třikrát dotknou, hra končí. Pokud trefíme určité objekty ve hře, přibyde nám počet lodí z počáteční jedné na tři, pokud se zbavíme určitého počtu mimozemských návštěvníků, tak dostaneme lepší loď a podobně.

Nyní můžeme nabádat žáky, ať individuálně přemýšlí nad tím, jak by použili blok Čekej dokud nenastane právě v případě hry tohoto typu. Jako příklad můžeme napsat na tabuli k již existujícímu textu: trefíš mimozemštana. Jako další příklad můžeme uvést: Čekej dokud nenastane <trefení mimozemské lodi>.

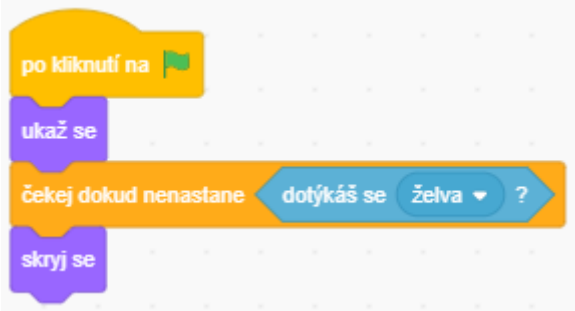
Žáci si po chvilce zamyšlení se nad touto situací svoje řešení zapíší a poté porovnají se sousedem. Jakožto skupinka si dvojice vybere mluvčího a až učitel řekne, ten třídě představí své řešení. Učitel si mluvčí vyslechne a až uplně na konci poví, jaká jsou špatná a jaká správná řešení.

Žáci jsou v tomto věku obeznámeni s videohrami a tak by neměl být problém si podobnou situaci, ke které vymýšlejí použití bloku **Čekej dokud nenastane**, představit. Pouze stačí přistoupit k ní z programátorského úhlu pohledu.

Pokud se ho dotkne, tak se skryje.

Řešení 2.bodu úlohy

Z **Vzhledu** vybeme **Skryj se** a umístíme tento blok pod blok **Čekej dokud nenastane**.

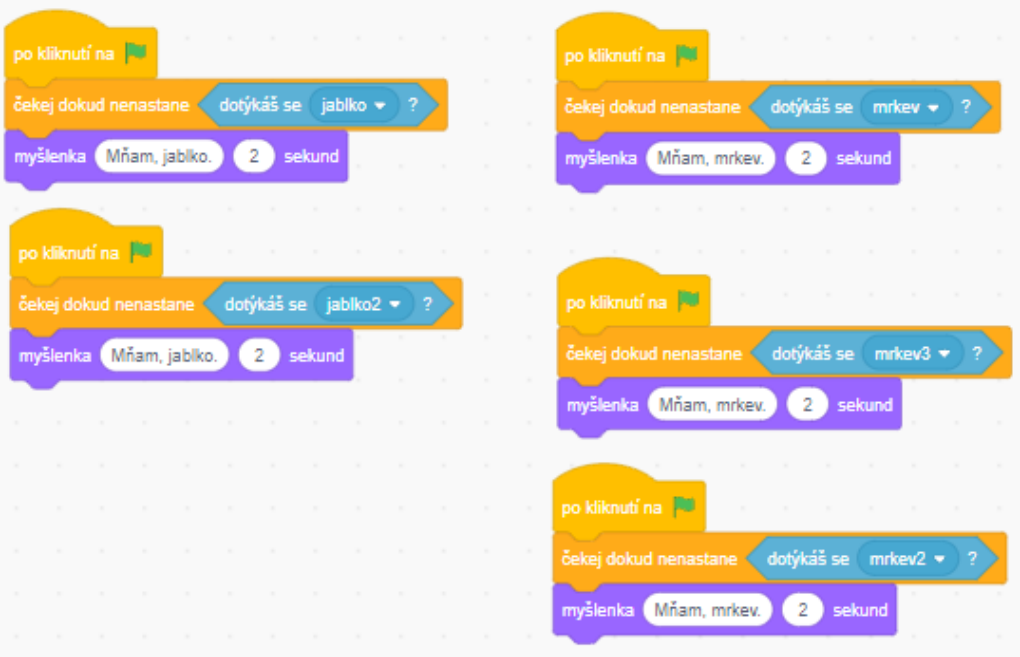


Obrázek 24 Kompletní řešení 2.bodu úlohy

Metodická poznámka

Pokud žáci netuší, jak vyřešit 3. a 4.bod úlohy, můžeme jim připomenout koncept představený v minulé hodině – zprávy. Pomocí nich může postava jídlo odeslat zprávu v případě, že se ho želva dotkne.Tu následovně želva přijme a pronese informaci o tom, o jaký druh jídla šlo.

Obdobně je možné úlohu vyřešit detekcí druhu jídla u želvy. Jediné, na co si musíme při použití tohoto způsobu dát pozor je dát každý z bloků **Čekej dokud nenastane** pod vlastní blok **Po kliknutí na zelenou vlaječku**.



Obrázek 25 Alternativní řešení s detekcí jídla u želvy

Závěr

Žáci by nyní měli umět naprogramovat mizení předmětů a použít blok **Čekej dokud nenastane**.

7.10. Třetí lekce – Auto instrukce pro žáky

Dnes si naprogramujeme závodní auto. Naše auto bude po cestě sbírat hvězdičky.

1. Počáteční bloky s vlaječkou ponechejte v programu a nemažte je.
2. Postupujte podle tabulky. K vyřešení vám může být užitečné vzpomenout si na zprávy z minulé hodiny.

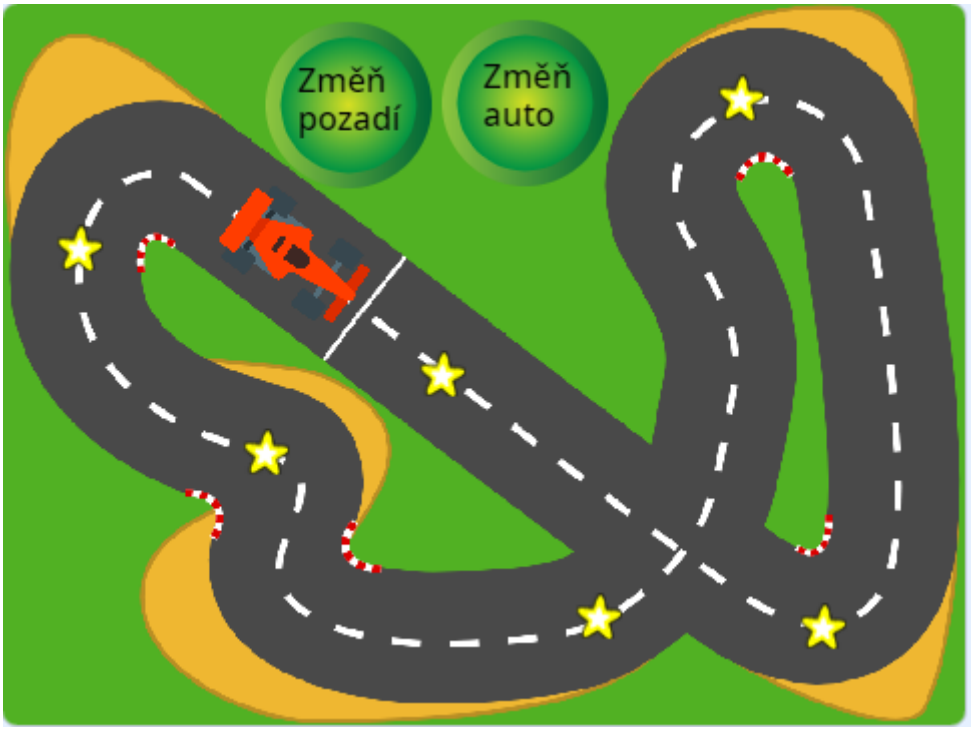
První člen dvojice	• Naprogramuj mizení hvězd poté, co se k nim přiblíží auto. • Po kliknutí na tlačítko s nápisem Změň auto se přepne postava auta na další kostým.
Druhý člen dvojice	• Naprogramuj pohyb auta podle šipek. • Přidej si dvě další pozadí z nabídky Scratche. • Po kliknutí na tlačítko s nápisem Změň pozadí se přepne pozadí na další.

Použité skupiny bloků: Ovládání, Vzhled, Události, Vnímání, Pohyb

7.11. Třetí lekce – Auto (učitelský návod)

Popis scénáře

V této lekci si zopakujeme znalosti z minulých lekcí týkající se zpráv a pohybu postavy. Využijeme u něho práci ve dvojicích.



Obrázek 26 Vzhled scénáře Auto

Použité programátorské koncepty

Příkaz

Události

Zprávy

Nově představené bloky z programu Scratch

Vzhled – Další kostým

Vzhled – Další pozadí

Využité bloky

Bloky týkající se zpráv, viditelnosti postavy, blok týkající se smyčky „while“, bloky vnímání, Bloky týkající se pohybu

Použitý projekt

03 – Auto – Teacher.sb3	soubor pro učitele
03 – Auto – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Právě protože v tomto úkolu jde primárně o zopakování již naučených znalostí, můžeme u něho využít práci ve dvojicích. Žáky můžeme rozdělit například podle toho jak již sedí ve třídě do párů (případně podle jakéhokoliv způsobu, který uznáme za vhodný). V instrukcích máme dvě skupiny úkolů, rozdělené aby každý člen dvojice programoval část scénáře.

Po kliknutí na tlačítko s nápisem Změň auto se přepne postavu auta na další kostým.

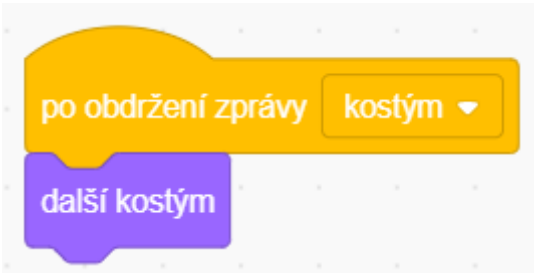
Metodická poznámka

Tento úkol, ačkoliv jde o něco nového, je formulován tak, že by mělo být jasné, kde hledaný blok nalézt a jak se bude jmenovat. Necháme žáky pracovat chvíli ve dvojicích, obejdeme si je a zjistíme, jak na tom jsou. Pokud po čase potřebném pro dokončení úkolu většina třídy stále neví, ukážeme postup na tabuli.

Všem dvojicím řekneme, že nastane-li situace, kdy jeden z dvojice ví jak toto naprogramovat a druhý ne, mají si zkusit princip bloku mezi sebou vysvětlit. V této lekci celkově nefungujeme jako koordinátor výuky, spíše jako pomocník, který zná řešení a může žákům pomoci, pokud je potřeba. Oproti ostatním lekcím, kdy režii hodiny máme na starosti my jakožto učitelé, nyní si ji z velké části vedou dvojice samy.

Řešení 2. bodu úlohy

Pod blok **Po kliknutí na mě** z **Událostí** umístíme blok s odesláním zprávy o změně kostýmu. Tato událost se děje v rámci postavy tlačítka. Poté, co zprávu postava auta přijme, pod stejnojmenný blok umístíme **Další kostým**, tentokrát ze skupiny bloků **Vzhled**.



Obrázek 27 Řešení 2.bodu úlohy u auta

Přidej si dvě další pozadí z nabídky Scratche.

Metodická poznámka

Nyní se můžeme zeptat, zda někdo již vyzkoušel, jak přidat pozadí. Z mé zkušenosti na to žáci přišli celkem rychle i sami od sebe, ale pokud se tak nestalo ukážeme přidání pozadí na tabuli.

Řešení 2.bodu úlohy

Kliknutím na menu scény se nám nad skupinami bloků zpřístupnila karta Pozadí. Pokud ji klikem vybereme, zobrazí se nám všechny pozadí, jaké zatím scénář obsahuje. Nové pozadí vyberem najetím myší na kulaté tlačíko a kliknutím na Obrázek lupy. Tento úkon nám zpřístupnil již hotová pozadí, která můžeme v programu Scratch použít.

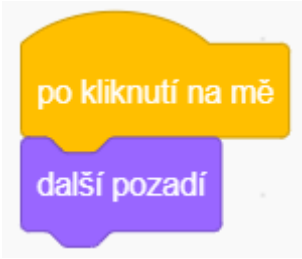
Po kliknutí na tlačítko s nápisem Změň pozadí se přepne pozadí na další.

Metodická poznámka

Zde platí to samé, co u minulých zadání v této hodině - opět se zeptáme zda někdo potřebuje pomoci. Pokud se tací ve třídě najdou, ukážeme postup na tabuli.

Řešení 2.bodu úlohy

Ze skupiny bloků **Události** vybereme blok **Po kliknutí na mě**. Pod něj umístíme **Další pozadí**, blok, který najdeme ve skupině bloků **Vzhled**.



Obrázek 28 Řešení 2.bodu úlohy u tlačítka

Závěr

Žáci si zopakovali výklad z minulých hodin a do jejich programovacího repertoáru přibyly další příkazy týkající se změny vzhledu.

7.12. Čtvrtá lekce – AutoV2 instrukce pro žáky

V dnešní lekci si k autu z minulé hodiny přidáme hudební efekty a vyzkoušíme další způsob pohybu postavy.

1. Pomocí cyklu Opakuj stále naprogramuj pohyb auta dopředu. Auto se začne pohybovat, jakmile zmáčkneme šipku dopředu.
2. Pokud klikneme na tlačítko s nápisem Změň auto, přehraje se zvuk „auto“ až do konce.
3. Pokud se auto dotkne některé z hvězd, začne hrát zvuk s názvem sebrat.

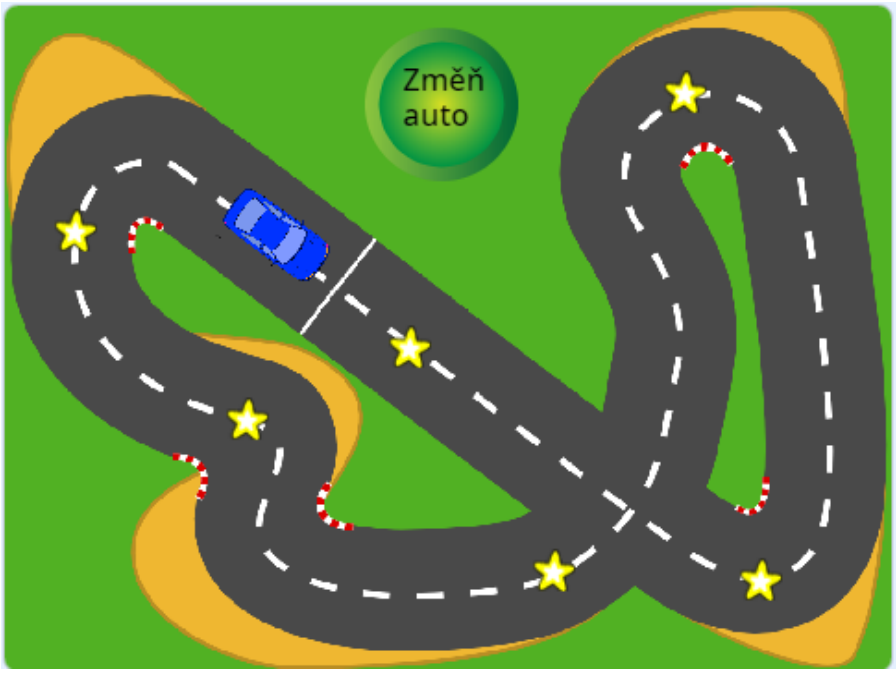
Úkol navíc: Zkuste si po kliknutí na tlačítko Změň auto odebrat **Přehraj zvuk až do konce** a vyměňte ho za **Začni hrát zvuk**. Jaký je mezi nimi rozdíl?

Použité skupiny bloků: Zvuk, Ovládání, Události, Pohyb

7.13. Čtvrtá lekce – AutoV2 (učitelský návod)

Popis scénáře

V této lekci si přidáme zvukové efekty ke scénáři z minulé hodiny a naučíme žáky naprogramování pohybu postavy s využitím cyklu.



Obrázek 29 Vzhled programu AutoV2

Použité programátorské koncepty

Příkaz

Události

Zprávy

Cyklus - nové

Nově představené příkazy z programu Scratch

Zvuk – Přehraj zvuk až do konce

Zvuk – Začni hrát zvuk

Ovládání – Opakuj stále

Využité bloky

Bloky týkající se zvuku, pohybu, bloky týkající se cyklů

Použitý projekt

04 – AutoV2 – Teacher.sb3	soubor pro učitele
04 – AutoV2 – Student.sb3	soubor pro žáky

Praktická část

1. Pomocí cyklu Opakuj stále naprogramuj pohyb auta dopředu. Auto se začne pohybovat, jakmile zmáčkneme šipku dopředu.

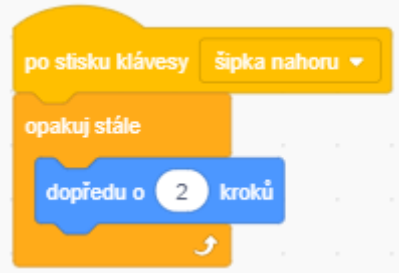
Metodická poznámka

V souboru 04 – AutoV2 – Student.sb3 se nachází scénář velmi podobný minulému. Dnešní úkol se bude trochu lišit, a to v ovládání pohybu. Jakmile zmáčkneme šipku dopředu, auto se bude chovat jako by mělo cihlu na plynu a pojede dopředu, dokud scénář nerestartujeme či nestopneme.

K naprogramování tohoto scénáře bude nutné žákům vysvětlit co je to cyklus. Jak již název napovídá, cyklus je sekvence příkazů, která se opakuje tolikrát, kolik specifikuje programátor. V případě Scratche můžeme cyklus buď opakovat donekonečna, to je blok **Opakuj stále** ze skupiny bloků **Ovládání** či opakovat pouze zadané množství opakování. K tomu slouží blok **Opakuj 10 krát** ze stejné skupiny příkazů, kdy desítku můžeme přepsat na požadované množství opakování.

Řešení 1.bodu úlohy

Ze skupiny bloků **Události** vybereme blok **Po stisku klávesy** a umístíme ho do prostoru scénáře. Z nabídky kláves vybereme šipku nahoru. Pod něj umístíme blok **Opakuj stále** ze skupiny bloků **Ovládání**. Dovnitř tohoto bloku umístíme nám již známý blok **Dopředu o 2 kroků** ze skupiny bloků **Pohyb**.



Obrázek 30 Řešení 1.bodu úlohy

2. Pokud klikneme na tlačítko s nápisem Změň auto, začne hrát zvuk s názvem auto.

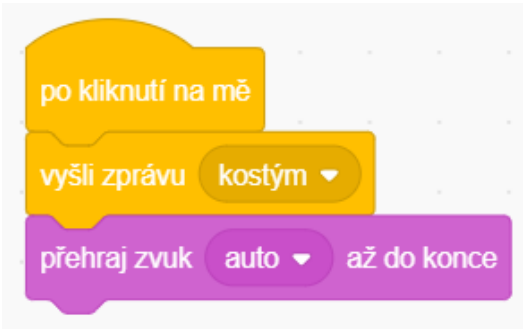
Metodická poznámka

Před programováním scénáře AutoV2 je dobré v předchozí hodině zmínit, aby si žáci přinesli vlastní sluchátka (pokud jimi škola nedisponuje).

Klikneme na jednu z postav a upozorníme žáky na kartu Zvuky. Tato karta se nachází nad skupinami bloků vedle karty Kostýmy a má jako logo vypovídající Obrázek reproduktoru. Nabádáme je k proklikání jednotlivých postav a poslechu zvuků, které se u nich nacházejí. Poté ukážeme, jak sestrojit druhý bod úlohy.

Řešení 2.bodu úlohy

Ze skupiny bloků **Zvuk** vybereme **Přehraj zvuk až do konce** a umístíme ho pod existující bloky. Do prázdného místa vybereme zvuk auto.

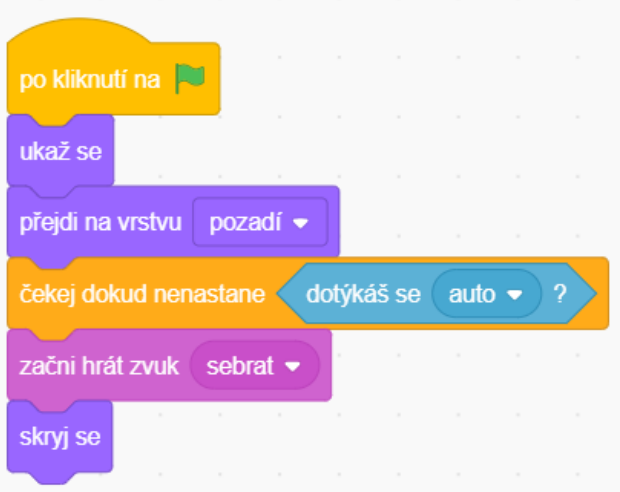


Obrázek 31 Řešení 2.bodu úlohy

3. Pokud se auto dotkne některé z hvězd, začne hrát zvuk s názvem sebrat.

Řešení 3.bodu úlohy

Mezi blok **Čekej dokud nenastane** a **Skryj se** umístíme **Začni hrát zvuk** ze skupiny bloků **Zvuk.** Do prázdného místa vybereme zvuk sebrat.



Obrázek 32 Řešení 3.bodu úlohy

Závěr

Žáci by nyní měli rozumět základním příkazům týkajících se zvuku v programu Scratch a umět použít blok **Opakuj stále**.

7.14. Pátá lekce – Dopravní značky instrukce pro žáky

V dnešní lekci si naprogramujeme pohyb postavy a auta v závislosti na barvě na semaforu pro chodce a pro auta.

1. Naprogramujte, aby se po zapnutí scénáře na semaforu pro chodce objevil červený panáček.

2. Naprogramujte, aby se po zapnutí scénáře na semaforu svítilo červené světlo.

3.1 U semaforu pro chodce se donekonečna bude kontrolovat, zda jsou níže zmíněné klávesy stisknuté. Naprogramujte:

3.2 Pokud stiskneme klávesu „j“ jako jdi, objeví se na značce zelený panáček.

3.3 Pokud stiskneme klávesu „s“ jako stop, objeví se na značce červený panáček.

4.1 U semaforu se bude také donekonečna kontrolovat, zda jsou stisknuty klávesy zmíněné v následujících instrukcích bodu čtyři. Naprogramujte podobně jako u předchozího bodu úlohy:

4.2 Pokud stiskneme klávesu „r“ jako „red“, na semaforu se objeví červená.

4.3 Pokud stiskneme klávesu „y“ jako „yellow“, na semaforu se objeví žlutá.

4.4 Pokud stiskneme klávesu „g“ jako „green“, na semaforu se objeví zelená.

5. Když se změní semafor pro chodce na zeleného panáčka, chodec přejde silnici.

6. Když se na semaforu objeví zelená, auto přejede silnici.

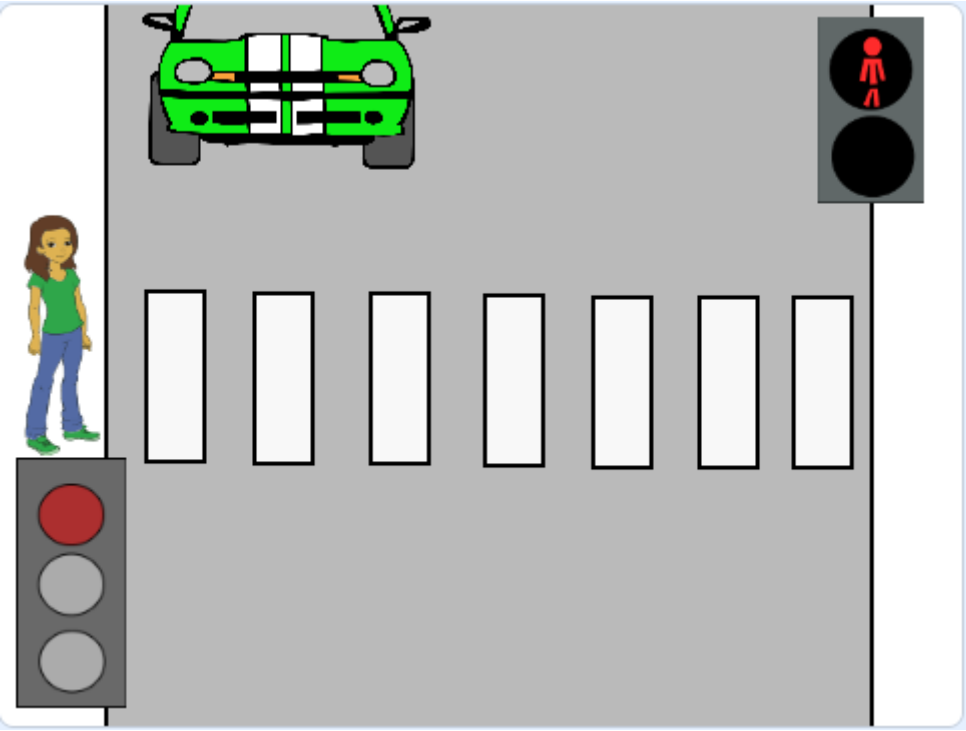
Nápověda: Pro vyřešení bodu 5 a 6 si vzpomeňte na zprávy.

Použité skupiny bloků: Vzhled, Vnímání, Události, Ovládání

7.15. Pátá lekce – Dopravní značky (učitelský návod)

Popis scénáře

V této lekci si představíme programátorský koncept podmínky.



Obrázek 33 Vzhled scénáře Dopravní značky

Použité programátorské koncepty

Příkaz

Události

Zprávy

Cyklus

Podmínky – nové

Nově představené příkazy z programu Scratch

Ovládání - Když..tak

Vnímání – Klávesa stisknuta?

Pohyb – Klouzej x sekund na x: y:

Využité bloky

Bloky týkající se cyklů a podmínek, bloky týkající se zpráv a kostýmů

Použitý projekt

05 – DopravniZnacky – Teacher.sb3	soubor pro učitele
05 – DopravniZnacky – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

V této lekci se poprvé potkáváme v instrukcích pro žáky s body zadání s tečkou. Konkrétně v tomto příkladu jsou to body zadání tři a čtyři, které se skládají z několika dílčích kroků. Tento druh číslování jsem zvolila aby bylo jasné, že bodem 3.1 se začne sekvence bloků a další bloky budeme dodávat postupně pod bloky, které jsme přidali v bodě 3.1. Podobně učiníme i u čtvrtého bodu úlohy a dalších úloh, které budou zadány tímto způsobem.

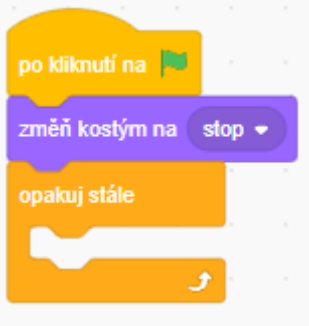
To však není jediná novinka s kterou se potkáváme - také tu poprvé použijeme podmínky. Podmínky jsou základní stavební kameny programování a fungují podobně jako podmínky v češtinářském významu. Na začátku podmínky je kouzelné slovíčko když či pokud, poté následuje zápis děje, který tímto programátorským konceptem zachytíme. V těle podmínky, v případě Scratche podmínkového bloku **Když..tak** ze skupiny bloků **Ovládání**, se nachází příkazy, které se provedou, pokud zachytávaný děj proběhne. Příkladem děje, který můžeme v programu Scratch zachytávat je například stisknutí určité klávesy.

Tělem podmínky se v programu Scratch rozumí prázdné místo uvnitř bloků jako **Opakuj stále**, **Když…tak** a některých dalších bloků ze skupiny bloků **Ovládání**.

3.1 U semaforu pro chodce se donekonečna bude kontrolovat, zda jsou níže zmíněné klávesy stisknuté. Naprogramujte:

Řešení 3.bodu úlohy

Jelikož budeme stisknutí kláves donekonečna kontrolovat, je nutné pod bloky, které jsme přidali v 1.bode řešení úlohy umístit blok **Opakuj stále** ze skupiny bloků **Ovládání.**. Do něj budeme umísťovat podmínky z bodu úlohy 3.2 a 3.3. Toto je nám již známý koncept cyklu, nyní je nově představen ve spojení s podmínkami.



Obrázek 34 Řešení bodu úlohy 3.1

3.2 Pokud stiskneme klávesu „j“ jako jdi, objeví se na značce zelený panáček.

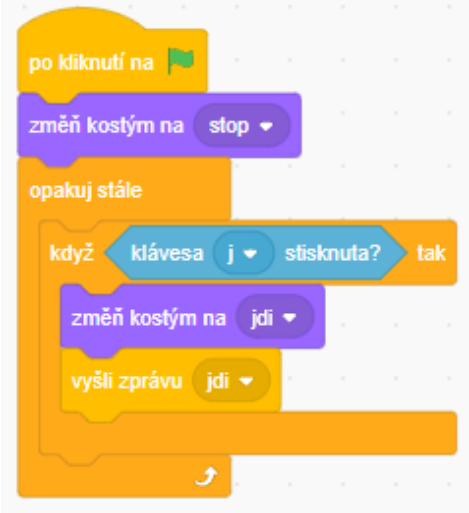
Metodická poznámka

Nyní můžeme na tabuli ukázat řešení podmínky v bodě úlohy 3.2, následující podmínku v bodě úlohy 3.3 a celý bod úlohy čtyři mohou žáci již naprogramovat sami – řešení mají obdobné.

Řešení bodu úlohy 3.2

Ze skupiny bloků **Ovládání** vybereme blok **Když..tak** a umístíme ho do prostoru scénáře. Do prázdného místa ve tvaru kosočtverce umístíme naši podmínku – v tomto případě blok **Klávesa stisknuta** ze skupiny bloků **Vnímání**. Z rozbalovacího menu bloku **Klávesa stisknuta** vybereme „j“.

Do těla podmínky vložíme bloky, které odpovídají zadání.

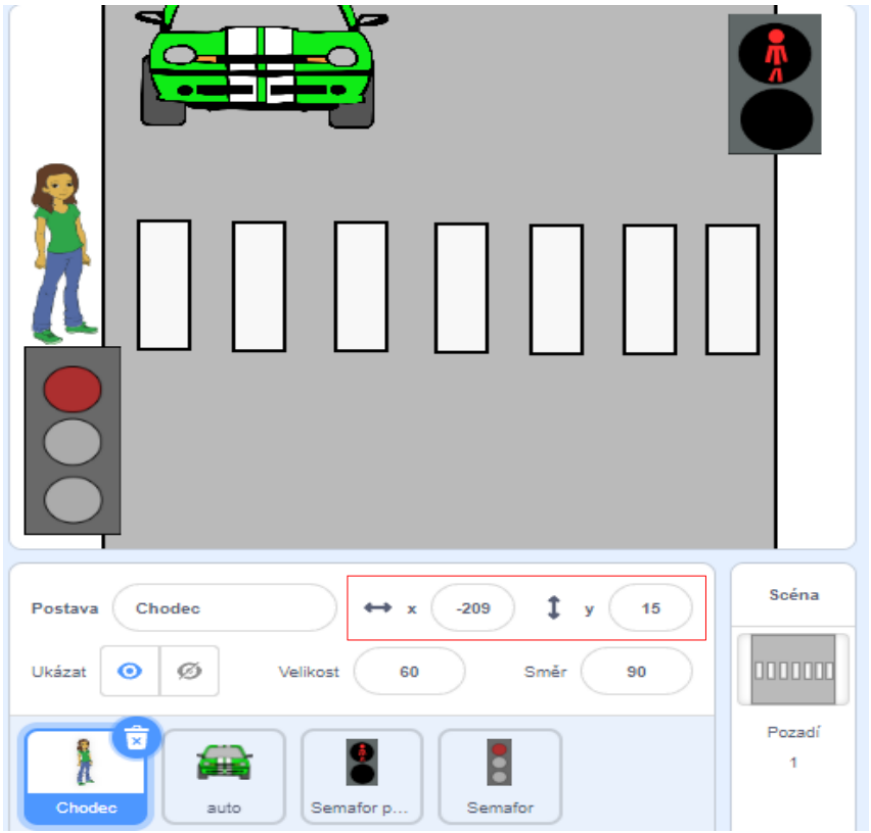


Obrázek 35 Řešení .bodu úlohy 3.2

5. Když se změní semafor pro chodce na zeleného panáčka, chodec přejde silnici.

Metodická poznámka

V pátém bodu úlohy se podíváme na způsob, jakým program Scratch říká, kde se právě nachází námi vybraná postava. Kliknutím vybereme postavu, u které tuto informaci chceme zjistit. Pod scénou se nachází dvě řady ikonek a informací. Souřadnice polohy postavy najdeme na prvním řádku na levé straně, u nápisu x a y. Postavy ve scéně se pohybují uvnitř kartézské soustavy.



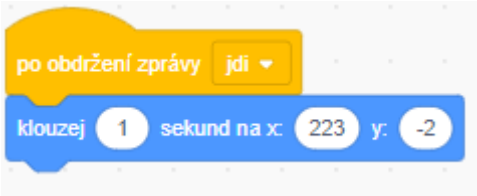
Obrázek 36 Souřadnice polohy postavy

Scratch a v něm obsažená skupina bloků **Pohyb** obsahuje několik bloků, kde se pohyb v sytému kartézské souřadnice využívá a my si ukážeme jeden z nich.

Řešení 5.bodu úlohy

Nyní budeme pracovat s postavou chodce. Začneme přetažením bloku **Po přijetí zprávy**, který oznamuje změnu semaforu pro chodce na zeleného panáčka do prostoru scénáře. Vybereme postavu chodce pomocí levého tlačítka myši a tlačítko nepouštíme. Následovně postavu přesuneme na pravou stranu silnice a až nyní levé tlačítko myši pustíme. Poté kliknutím na myši podržíme postavu a postuneme ji uvnitř scénáře tak, aby „přešla“ silnici. Dále se přepneme do skupiny bloků **Pohyb**. Všechny bloky, které pracují s kartézskými souřadnicemi x a y nám nyní ukazují aktuální polohu postavy.

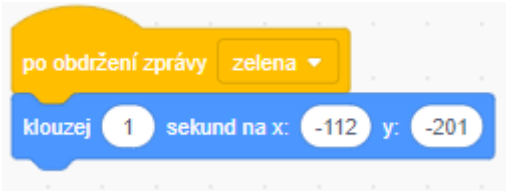
Do prostoru scénáře, pod blok o přijetí zprávy o zeleném panáčkovi, umístíme blok **Klouzej x sekund na x: y:** ze skupiny bloků **Pohyb**. Pohyb auta se vyřešíme obdobně.



Obrázek 37 Řešení 5.bodu úlohy

Metodická poznámka

Podobně jako jsme vyřešili pátý bod úlohy vyřešíme i šestý. Opět zde budeme hýbat s postavou uvnitř scénáře a pracovat se zprávami.



Obrázek 38 Řešení 6.bodu úlohy

Závěr

Žáci by nyní měli rozumět programátorskému konceptu podmínek.

7.16. Šestá lekce – Hudební nástroje instrukce pro žáky

V dnešní lekci si naprogramujeme hudební nástroje, které budou hrát pouze pokud se jich postava dotkne a zároveň bude stisknut mezerník.

1. Po stisknutí šipky vlevo Holly popojde vlevo a přepne se na další kostým. Naprogramuj.
2. Po stisknutí šipky vpravo Holly popojde vpravo a přepne se na další kostým. Naprogramuj.
3. Pokud se bude Holly dotýkat piána a zároveň bude stisknuta klávesa mezerník, tak se přehraje zvuk s názvem piano. Naprogramuj.
4. Pokud se bude Holly dotýkat kytary a zároveň bude stisknuta klávesa mezerník, tak se přehraje zvuk s názvem kytara. Naprogramuj.
5. Po kliknutí na tlačítko se zastaví všechny zvuky.

Úkol navíc: Ve scénáři zbývá ještě jeden hudební nástroj. Naprogramujte ho podobně jako předcházející.

Použité skupiny bloků: **Ovládání**, **Vzhled**, **Události**, **Vnímání**, **Zvuk**, **Pohyb**, **Operátory**

7.17. Šestá lekce – Hudební nástroje (učitelský návod)

Popis scénáře

V této lekci si zopakujeme pohyb postavy a vysvětlíme operátory a jejich použití v programování.

Obrázek 39 Vzhled scénáře Hudební nástroje

Použité programátorské koncepty

Příkaz

Události

Cyklus

Podmínky

Operátory - nové

Nově představené příkazy z programu Scratch

Operátory – A

Zvuk – Zastav všechny zvuky

Využité bloky

Bloky týkající se pohybu, bloky týkající se zvuku, podmínkové bloky, bloky cyklu

Použitý projekt

06 – Hudební nástroje – Teacher.sb3 soubor pro učitele

06 – Hudební nástroje – Student.sb3 soubor pro žáky

Praktická část

Metodická poznámka

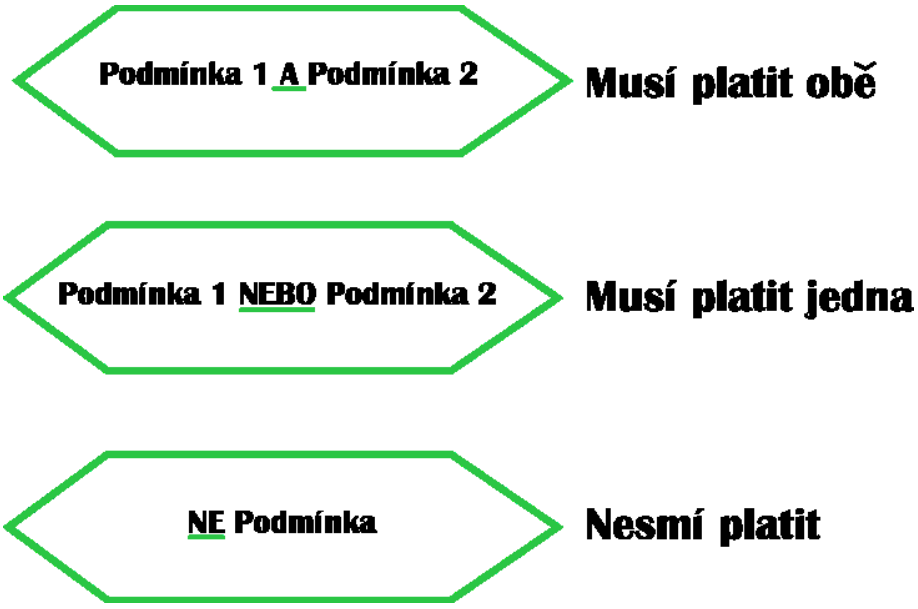
První dva body úlohy můžeme žáky nechat řešit samostatně, jedná se o opakování pohybu.

3. Pokud se bude Holly dotýkat piána a zároveň bude stisknuta klávesa mezerník, tak se přehraje zvuk s názvem piano. Naprogramuj.

Metodická poznámka

Před řešením třetího bodu úlohy si připomeneme podmínku v češtinářské podobě. Typicky se skládá z výroku, který má na začátku spojku pokud nebo když. Po ní následuje situace, která musí nastat, aby se stalo pokračování věty – to často začíná spojkou tak.

Podmínka v programování je velmi podobná. Příklad podmínky v programování může být například: pokud je zmáčknutá klávesa „k“, přepni pozadí na další. Kromě případu s jednou podmínkou můžeme mít podmínek více a to v různých vztazích. Opět se zde jedná o spojky. Lépe se to vysvětlí obrázkem a následovnou prací žáků, kde si promyslí své nápady, napíší si je a poté proberou se sousedem.



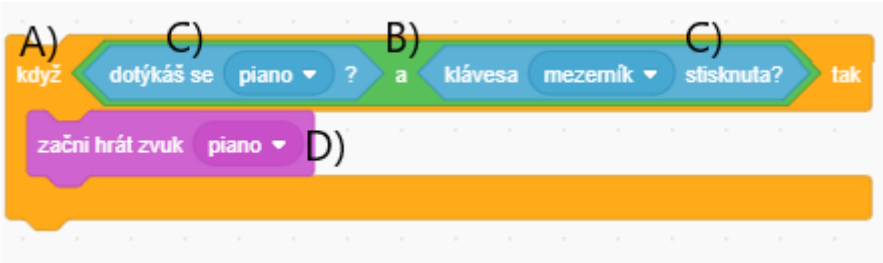
Obrázek 40 Obrázek popisující podmínky

Nyní by si žáci měli napsat dva příklady ke každé z podmínek, jeden z reálného života a druhý, který by využili v programu Scratch nebo v obecném pseudo - programovacím jazyku. Poté co budou s psaním hotovi a učitel tak určí, proberou své doměnky se sousedem.Učitel poté vybere, pokud možno náhodně, jednoho ze dvojice a ten za oba její členy řekne nejpovedenější podmínku z reálného života a programování.

Řešení 3.bodu úlohy

Ještě než umístíme samotné podmínky, je nutné je obalit v cyklu, který bude platit po celý běh scénáře. Tento cyklus již známe – jmenuje se **Opakuj stále** a najdeme ho ve skupině bloků **Ovládání**.

Toto řešení je lepší zobrazit obrázkem, který si blíže popíšeme.



Obrázek 41 Řešení podmínky

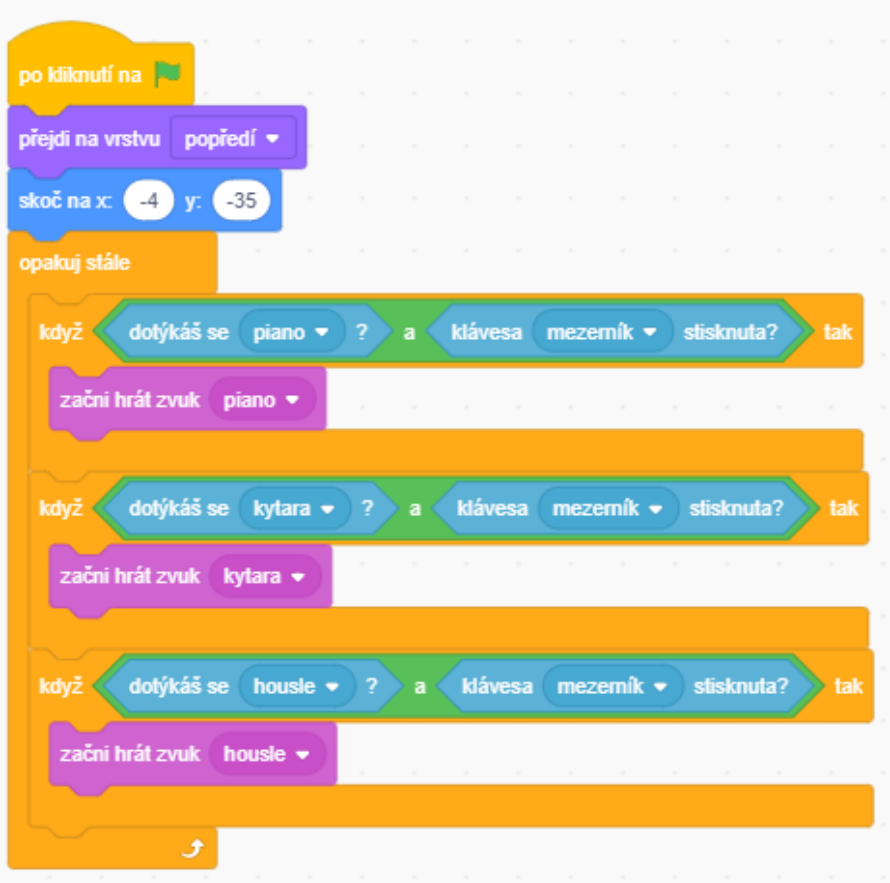
A) Jako první do cyklu **Opakuj stále** umístíme ze stejné skupiny bloků, **Ovládání**, podmínkový blok **Když..tak**.

B) Pracujeme se dvěma podmínkami, takže nám místo nebude stačit. Z **Operátorů** vybereme blok se dvěma prázdnými místy a spojkou „a“ mezi nimi. Tyto podmínky musí platit obě, aby proběhly příkazy uvnitř.

C) Přepneme se do skupiny bloků **Vnímání** a vybereme **Klávesa stisknuta**. Tu umístíme do libovolného prázdného místa, na pořadí nezáleží. Klávesu, jejíž stisknutí kontrolujeme, můžeme vybrat v nabídce. Do druhého dáme **Dotýkáš se** a vybereme název pro postavu piana.

D) Vevnitř podmínky se nachází blok, který se stane pouze pokud jsou obě výše zmíněné podmínky splněny.

Nyní se podíváme na kompletní řešení cyklu s podmínkami.



Obrázek 42 Kompletní řešení cyklu s podmínkami

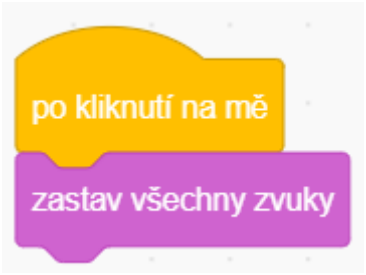
Metodická poznámka

Po přidání bloku **Začni hrát zvuk** do podmínky se občas stane, že při zkoušení zvuk nebude spustitelný. Pro vyřešení této situace stačí stisknout zelenou vlaječku nad scénářem.

4. Po kliknutí na tlačítko se zastaví všechny zvuky.

Řešení 4.bodu úlohy

Ze **Zvuku** vybereme blok s názvem **Zastav všechny zvuky** s názvem nástroje, kterého se podmínka týká.



Obrázek 43 Řešení 4.bodu úlohy

Závěr

Žáci by nyní měli rozumět základním typům operátorů a jak ovlivňují podmínky, u kterých se nacházejí.

7.18. Šestá lekce – Hudební nástrojeV2 instrukce pro žáky

Kromě postav lidí a zvířat můžou v programu Scratch vnímat v lidských očí i „neživé“ předměty. Naprogramujte podmínky podobně jako u předchozího úkolu, ale s tím rozdílem, že nyní budou vnímat hudební nástroje. Podmínky tedy budou napsané u nich, z jejich „pohledu“.

1. Piáno: Když se ho dotýká postava a zároveň je stisknut mezerník, začne hrát zvuk piano.
2. Kytara: když se jí dotýká postava a zároveň je stisknut mezerník, začne hrát zvuk kytara.

Úkol navíc: Podobně naprogramujte i housle.

Použité skupiny bloků: Ovládání, Vnímání, Zvuk, Operátory

7.19. Šestá lekce – Hudební nástrojeV2 (učitelský návod)

Popis scénáře

Nyní si ukážeme, že ve Scratchi i „neživé“ předměty mohou vnímat a vskutku nezáleží, co postava zobrazuje.

Použité programátorské koncepty

Příkaz

Události

Cyklus

Podmínky

Operátory

Nově představené příkazy z programu Scratch

V této lekci žádné nové příkazy neukazujeme.

Využité bloky

Bloky týkající se pohybu, bloky týkající se zvuku, podmínkové bloky, bloky cyklu

Použitý projekt

06 – Hudební nástrojeV2 – Teacher.sb3	soubor pro učitele
06 – Hudební nástrojeV2 – Student.sb3	soubor pro žáky

Praktická část

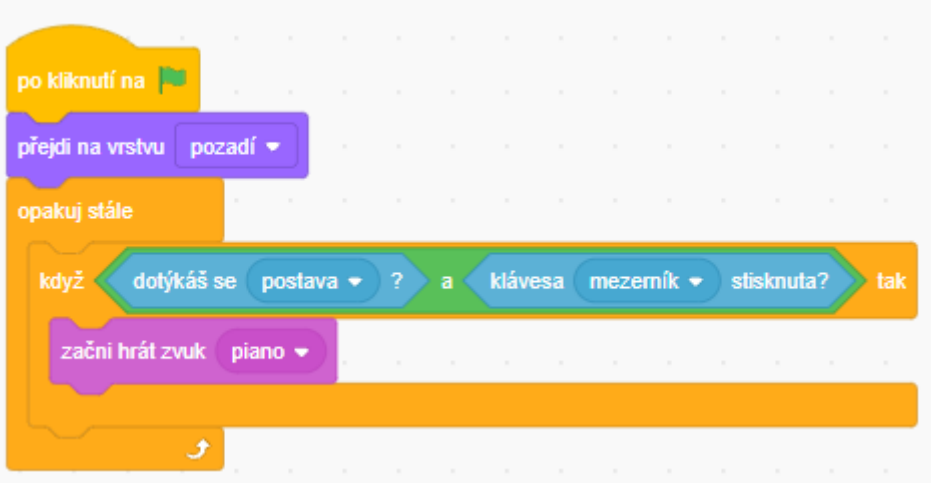
Metodická poznámka

Když se na program podíváme lidským pohledem, vnímat bude vždy postava člověka nebo zvířete – ta je toho schopná i v reálném životě. V programování ale mohou vnímat i předměty, postavy neživých věcí a tato lekce je přesně stavěné na pochopení tohoto problému.

Pokud žáci z minulé hodiny zapomněli, můžeme jim řešení prvního úkolu ukázat na tabuli. Opět je nutné podmínku vložit do cyklu **Opakuj stále**, řešení je obdobné jako minule.

1. Piáno: Když se ho dotýká postava a zároveň je stisknut mezerník, začne hrát zvuk piano.

Řešení 1.bodu úlohy



Obrázek 44 Řešení podmínky u piána

Závěr

Žáci nyní chápou, že v programování lze vnímat i skrze předměty, ačkoliv lidská mysl funguje jinak.

7.20. Sedmá lekce – ŽelvaV3 instrukce pro žáky

V dnešní lekci si ze scénáře s želvou uděláme hru, jak se patří – bude sbírat body.

1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny postavy ve scénáři a pojmenujte ji „body“.
2. U jaké postavy se nachází zvuk? Jak se tento zvuk jmenuje?
3. Pokud se želva dotkne jablka, zvednou se body o 1. Také se přehraje zvuk, který jste našli v předchozím bodě úlohy.
4. Pokud se želva dotkne mrkve, zvednou se body o 1. Přehraje se ten samý zvuk jako u jablka.
5. Co se stane s proměnnou pokud scénář párkrát spustíme? Je možné tento děj ošetřit a zabránit mu?

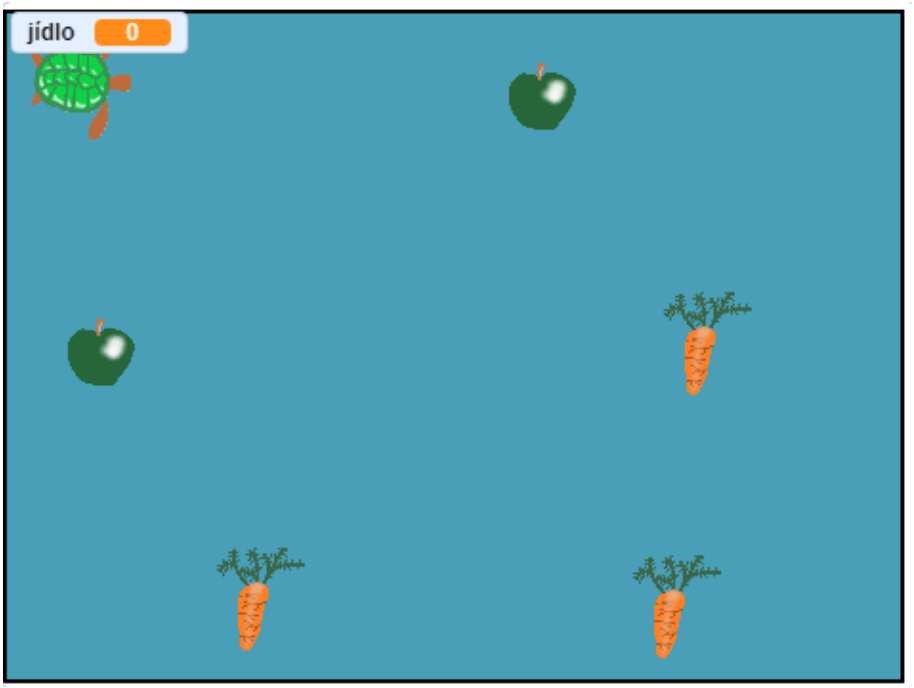
Nápověda: Pro řešení pátého bodu úlohy si prohlédněte příkazy ve skupině bloků **Proměnné**.

Použité skupiny bloků: Zvuk, Proměnné

7.21. Sedmá lekce – ŽelvaV3 (učitelský návod)

Popis scénáře

V této lekci si vysvětlíme, co je to proměnná a jak s ní manipulovat pomocí příkazů v programu Scratch.



Obrázek 45 Vzhled scénáře Želva V3

Použité programátorské koncepty

Příkaz

Události

Zprávy

Proměnné - nové

Nově představené příkazy z programu Scratch

Proměnné – Změň proměnnou o

Proměnné – Nastav proměnnou na

Využité bloky

Bloky pro práci s proměnnou, bloky týkající se zvuku

Použitý projekt

07 – ŽelvaV3 – Teacher.sb3	soubor pro učitele
07 – ŽelvaV3 – Student.sb3	soubor pro žáky

Praktická část

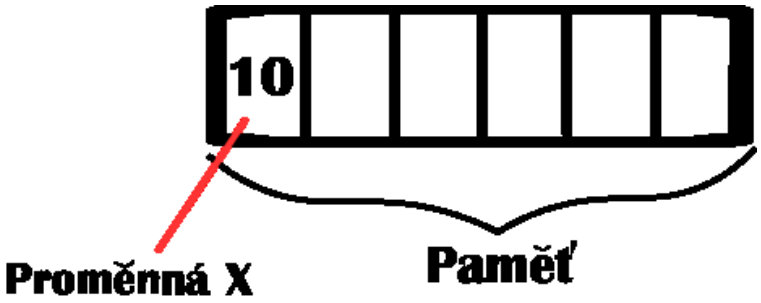
1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny postavy ve scénáři a pojmenujte ji „body“.

Metodická poznámka

Nyní budeme vysvětlovat, co to vlastně proměnná je. Aby program mohli řešit nějaký úkol potřebuje si určité informace zapamatovat. Právě k tomu účelu slouží proměnná. Část paměti počítače si pojmenujeme určitým jménem a dále na ni tímto jménem odkazujeme.[35] Každá proměnná by měla mít jedinečný název a například program Scratch nám ani nedovolí pojmenovat novou proměnnou totožně jako již existující.

Ukažte na tabuli/projektoru celý postup řešení prvního kroku úlohy. Také můžeme pro lepší představu namalovat na tabuli čtvereček znamenající místo v paměti. Paměť má místo pro více proměnných, takže můžeme podobných čtverečků zakreslit více. Poté vymýšlejte jednoduché příkazy typu: Přičti 2 a přepisujte ten samý čtvereček, aby bylo jasné, že jde o jedno a to samé místo v paměti. Můžete se inspirovat obrázkem pod textem, který můžeme namalovat na tabuli nebo do programu malování, podle vybavení třídy.

[35] Proměnné. In: *Úvod do programování* [online]. [cit. 2022-05-06]. Dostupné z: https://mrlvsb.github.io/upr-skripta/c/promenne/promenne.html



1) Odečti 2 od X
2) Přičti 10 k X
3) zdvojnásob X

Obrázek 46 Vysvětlení proměnné na tabuli

Existují dva typy proměnných – globální a lokální. Můžeme se zeptat třídy, co si pod pojmem globální v kontextu programování představuje a udělat podobné cvičení jako v minulé lekci. K sobě si každý napíše co podle něho tyto dva výrazy znamenají a poté se o svůj názor podělí se sousedem. Učitel se po nějaké době zeptá jednoho z dvojice či si dvojice sama vybere mluvčího za skupinu. Teprve až se všechny dvojice vyjádří řekneme správnou odpověď, do té doby by jsme ji neměli prozradit.

Lokální proměnná platí pouze pro určitou část programu, v případě Scratch to znamená pouze pro jednu z postav. Globální, jak už název napovídá, platí pro celý program a může s ní manipulovat každá z postav.

Pokud je proměnná v programu Scratch zaškrtnutá, její hodnota a název bude zobrazený na scéně. V případě, že proměnná je lokální, zobrazí se před názvem proměnné jméno postavy, ke které patří.

Řešení 1.bodu úlohy

Vyberte skupinu bloků s názvem **Proměnné**. Zde vidíme přednastavenou proměnnou s názvem Moje proměnná. Klikneme na ni pravým tlačítkem myši a smažeme ji.

Novou proměnnou založíme kliknutím na tlačítko Vytvoř proměnnou, stále ve skupině bloků **Proměnné**. Vybereme možnost „pro všechny postavy“ a pojmenujeme ji body. Volbu potvrdíme.

2. U jaké postavy se nachází zvuk? Jak se tento zvuk jmenuje?

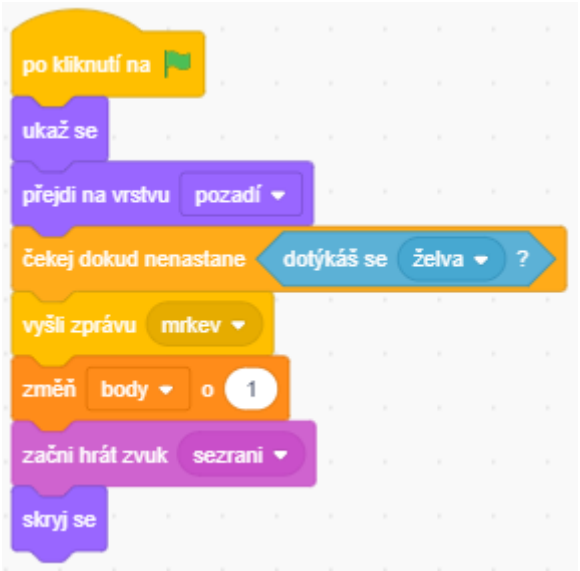
Metodická poznámka

Pokud zjistíme, kde se hledaný zvuk nachází, můžeme s ním v rámci této postavy pracovat. Zvuky vždy patří k určité postavě a jiná postava jej nemůže spustit.

3. Pokud se želva dotkne jablka, zvednou se body o 1. Také se přehraje zvuk, který jste našli v předchozím bodě úlohy.

Řešení 3.bodu úlohy

Přepneme se na postavu jablka. Mezi bloky **Vyšli zprávu** a **Skryj se** umístíme **Změň proměnnou** o ze skupiny bloků **Proměnné**. Opět pracujeme s tou samou globální proměnnou jménem body. Po snězení jablka se hodnota bodů navýší o jedna, takže do prázdného místa u bloku Změň proměnnou o napíšeme jedničku.



Obrázek 47 Řešení 3.bodu úlohy u jablka, u mrkve je totožné

Metodická poznámka

Tento postup můžeme ukázat na interaktivní tabuli či přes projektor. Úkol číslo čtyři má obdobné řešení a necháme ho na žácích.

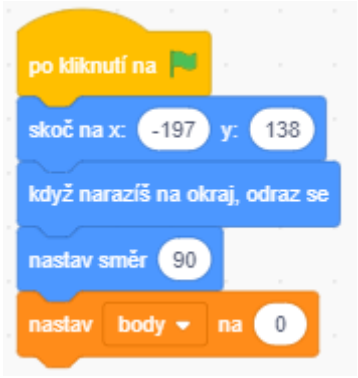
5. Co se stane s proměnnou pokud scénář párkrát spustíme? Je možné tento děj ošetřit a zabránit mu?

Metodická poznámka

Když želva sní nějaké ovoce, body se nasčítají přesně tak, jak jsme naprogramovali, ale při dalším spuštění tento počet bodů zůstane stejný. Podle zvyklostí ve videohrách, kde sbíráme body, by se správně měly body po opětovném spuštění vynulovat. Chvíli můžeme nechat žáky na řešení tohoto bodu úlohy přijít sami, ale nakonec bychom měli řešení opět ukázat na tabuli.

Řešení 5.bodu úlohy

Pod blok **Po kliknutí na zelenou vlaječku**, pod existující bloky, umístíme blok **Nastav proměnnou na**. V menu proměnných je na výběr pouze proměnná s názvem body – jediná proměnná, která v tomto scénáři existuje. Do prázdného místa napíšeme 0.



Obrázek 48 Řešení 5.bodu úlohy

Závěr

Žáci nyní umí základní příkazy pro práci s proměnnými.

7.22. Osmá lekce – Drak a rytíř instrukce pro žáky

V dnešní lekci si naprogramujeme boj mezi drakem rytířem.

1. Zjisti si, jaké kostýmy mají postavy k dispozici a jaká pozadí jsou v programu dostupná. V dalších bodech úlohy s nimi budeme pracovat.

2.1 Naprogramuj, aby rytíř měl na začátku scénáře 100 životů.

2.2 Ve scénáři se bude donekonečna kontrolovat, zda je rytíř naživu. Jaké číslo musíme dát do podmínky? Doprogramuj si další kroky bodu úlohy dva do těla podmínky a vyzkoušej.

2.3 Rytíř je poražen, leží na zemi a nastává konec hry. Naprogramuj.

3. Po kliknutí na meč, zaujme rytíř bojovnou pozu a poté se vrátí do klidové polohy. Naprogramuj.

4. Naprogramuj, aby se po útoku draka životy rytíře zmenšili o 25.

5. Po kliknutí na lektvar přibyde rytíři 20 životů. Jak naprogramuješ, aby počet jeho životů nepřekročil 100?

6.1 Naprogramuj, aby drak měl na začátku scénáře 120 životů. Můžeš použít stejně pojmenovanou proměnnou jako u rytíře?

6.2 Ve scénáři se bude donekonečna kontrolovat, zda je drak naživu. Dále naprogramuj do těla podmínky následující kroky bodu úlohy tři.

6.3 Drak je poražen a změní se na hvězdičku s počtem bodů.

7. Po kliknutí na tlačítko s nápisem „drak útočí“ drak plivne oheň a poté se vrátí do původní polohy. Naprogramuj.

8. Po útoku rytíře na draka se životy draka zmenší o 30.

9. Pokud vyhraje souboj drak, nastane konec hry, všechny postavy zmizí a bude viditelné pouze pozadí s nápisem konec hry. Co se stane když nyní znovu zapneš scénář? Co je potřeba udělat, aby byla hra opět hratelná?

10. Co se stane s životy rytíře pokud bude poražen? Jak naprogramuješ řešení, aby k tomu nedocházelo a kam blok umístíš?

11. Podobně se stane i s drakem, jak to vyřešíš u něho?

Úkol navíc: Vytvoř další pozadí, tentokrát znamenající vítězství. Pokud bude drak zabit, všechny postavy se skryjí a bude vidět pouze pozadí. Řešení je obdobné jako u prohry.

Použité skupiny bloků: Pohyb, Vzhled, Události, Ovládání, Operátory, Proměnné

7.23. Osmá lekce – Drak a rytíř (učitelský návod)

Popis scénáře

V tomto scénáři si zopakujeme práci s podmínkami a proměnnými a seznámíme se s další kategorií operátorů.

Obrázek 49 Vzhled scénáře Drak a rytíř

Použité programátorské koncepty

Příkaz

Události

Zprávy

Cyklus

Podmínky

Operátory

Proměnné

Nově představené příkazy z programu Scratch

Operátory – Operátor <

Operátory – Operátor >

Pohyb - Změň y o

Vzhled – Přepni pozadí na

Využité bloky

Bloky pro práci s proměnnými, podmínkové bloky, bloky týkající se vzhledu

Použitý projekt

08 – Drak a rytíř – Teacher.sb3	soubor pro učitele
08 – Drak a rytíř – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Necháme žáky udělat úkoly hlavně samostatnou prací, koneckonců jde o opakování, do bodu 2.2. Nyní je čas vysvětlit něco málo z teorie, neboť se zde poprvé potkáme s novým typem operátoru. Konkrétně se jedná o symboly pro menší než „<“ a větší než „>“, které se využívají hlavně pro práci s proměnnými. Na tabuli můžeme tyto symboly namalovat a vysvětlit na konkrétní dvojici čísel, jak fungují. Poté můžeme zmínit, že místo čísel si zde pro účely programování představíme proměnné. Například můžeme uvést podmínku, která se stane platnou v případě, že postavě ve hře klesnou životy na menší než 1 – hra tedy končí, postava umírá.

Jako pomůcku pro pochopení symbolů větší než a menší než mohu doporučit říkanku: Krokodýl chce sníst větší číslo.

2.2 Ve scénáři se bude donekonečna kontrolovat, zda je rytíř naživu. Jaké číslo musíme dát do podmínky? Doprogramuj si další kroky bodu úlohy dva do těla podmínky a vyzkoušej.

Řešení bodu úlohy 2.2

Většina řešení je již známá, zde jde spíše o kombinaci cyklu, podmínky a operátorů, kterou bychom měli zobrazit na tabuli. Z **Ovládání** vybereme cyklus **Opakuj stále**, ten umístíme pod bloky, které jsme přidali v bodě pět.

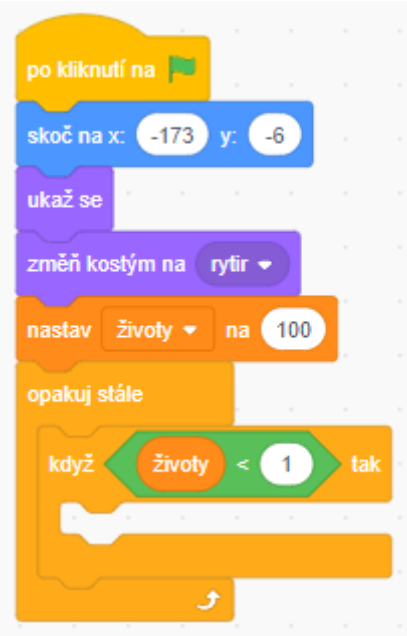
Do těla, to znamená „vnitřek“ cyklu umístíme blok **Když...tak**. Překlikneme se na skupinu bloků **Operátory** a do prázdného místa v podmínce vložíme zelený **blok se znamínkem menší než**, tedy **<**.

Ze skupiny bloků **Proměnné** přetáhneme proměnnou **životy** do levého prázdného místa **bloku se znaménkem menší než**. Do pravého místa zapíšeme jedničku. Teprve teď přidáváme příkazy v tabulce do těla této podmínky. První dva příkazy nejsou ničím novým, problém může nastat až u třetího.

Metodická poznámka

Řešení bodů 2.1 až 2.4 můžeme ukázat na tabuli. Při zkoušení, jaké číslo bude do podmínky sedět nejlépe se může stát, že nám logičtější přijde nula spíše než jednička. Při odzkoušení podmínky s nulou ovšem nastane situace, kdy rytíř, ačkoliv má nula životů stále stojí na nohou. Proto do bloku napíšeme jedničku.

Také je zde potřeba vysvětlit, co znamená výraz „tělo podmínky“ – jedná se o prostor uvnitř podmínky, kam umisťujeme příkazy, které se uskuteční pokud je podmínka platná. Ve Scratchi je tento prostor ohraničen podmínkovým blokem, v programování je hranice značena speciálními znaky.



Obrázek 50 Řešení bodu úlohy 2.2

2.3 Rytíř je poražen, leží na zemi a nastává konec hry. Naprogramuj.

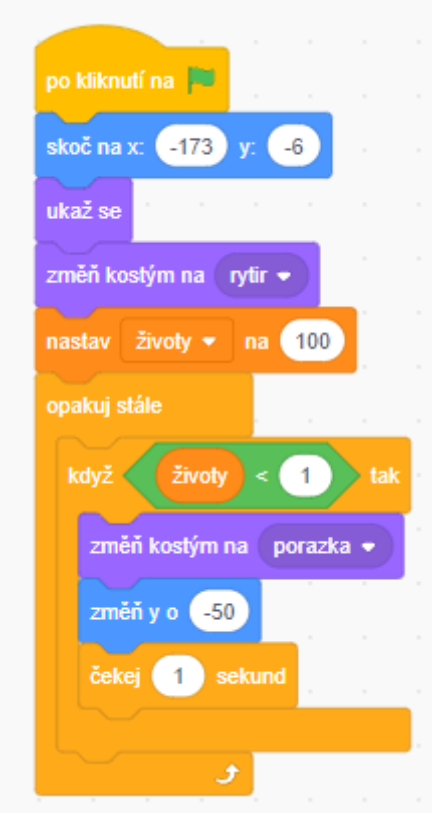
Metodická poznámka

Nyní můžeme vysvětlit, jak se řeší umístění prvků v programu Scratch. Jsou zde dvě souřadnice, x a y. Každá má kladnou a zápornou část a v rámci těchto souřadnic pohybujeme postavami. Také můžeme ukázat, že po vybrání jedné z postav vidíme pod scénářem čísla, která udávají jeho přesné umístění právě v rámci těchto souřadnic.

Na tabuli nakreslíme jednoduchou kresbu této kartézské souřadnice společně s několika příklady umístění, kde se nachází kladná část této souřadnice, kde záporná, jaká z čas je x a jaká y. Nemusíme jít nijak do hloubky, stačí pouze základní znalost, že nějaký takový systém existuje.

Řešení bodu úlohy 2.3

Jako první naprogramujeme změnu kostýmu. Ze skupiny bloků **Pohyb** vybereme příkaz **Změň y o** a umístíme ho pod předchozí blok. Do prázdného pole napíšeme -50, Scratch umí pracovat i s zápornými hodnotami. Tímto jsme naprogramovali pád rytíře na zem.



Obrázek 51 Řešení bodu 2.3

Metodická poznámka

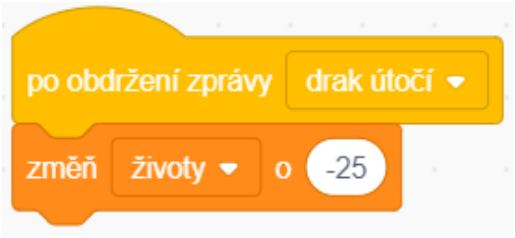
Nyní jsme pouze posunuli postavu o určitou hodnotu vůči jeho minulé pozici, narozdíl od přímo zadání polohy, které je taky možné naprogramovat. Příklad takového příkazu je například hned první pod blokem, který určuje, co se děje po spuštění scénáře. Tuto informaci je vhodné sdělit i žákům.

Bod úlohy číslo tři necháme udělat žáky samostatně.

4. Naprogramuj, aby se po útoku draka životy rytíře zmenšili o 25.

Řešení 4.bodu úlohy

Pod blok s přijatou zprávou umístíme **Změň proměnnou o** ze skupiny bloků **Proměnné**. Budeme zde měnit jedinou přítomnou proměnou, která náleží k postavě rytíře, tedy životy. Pokud chceme odečítat, stačí napsat před požadovaný počet mínus. Zapíšeme tedy: -25.



Obrázek 52 Řešení 4.bodu úlohy

Metodická poznámka

Zbytek dílčích kroků řešení je obdobný či jsme ho již probírali v minulých lekcích. Jediné, na co bychom si měli dát pozor, je přepínání mezi postavami a přidávání bloků k té postavě, k jaké má patřit. Tato úloha nejspíše zabere dvě, možná i tři hodiny, ale to není vzhledem k její komplexnosti nic zvláštního.

5. Po kliknutí na lektvar přibyde rytíři 20 životů. Jak naprogramuješ, aby počet jeho životů nepřekročil 100?

Řešení 5.bodu úlohy

Pomocí zpráv se postava rytíře dozví o kliknutí na letvar. Životy navýšíme podobným způsobem jako v lekci ŽelvaV3. Nyní naprogramujeme podmínku.

Přesuneme se do skupiny bloků **Ovládání**. Zde vybereme blok **Když..tak**. Do prázdného místa, kam umisťujeme podmínku vložíme blok **>** ze skupiny bloků **Operátory**.

Nyní nám zbývá tuto podmínku naprogramovat. Na levou stranu umístíme název proměnné **životy** ze stejnojmenné skupiny bloků a do pravého prázdného místa zapíšeme 100.

Obrázek 53 Řešení 5.bodu úlohy

6.1 Naprogramuj, aby drak měl na začátku scénáře 120 životů. Můžeš použít stejně pojmenovanou proměnnou jako u rytíře?

Metodická poznámka

Scratch se sám ozve a nově vytvořenou proměnnou jménem již existující proměnné pojmenovat nedovolí. Necháme třídu pracovat na úloze až do bodu devět.

9. Pokud vyhraje souboj drak, nastane konec hry, všechny postavy zmizí a bude viditelné pouze pozadí s nápisem konec hry. Co se stane když nyní znovu zapneš scénář? Co je potřeba udělat, aby byla hra opět hratelná?

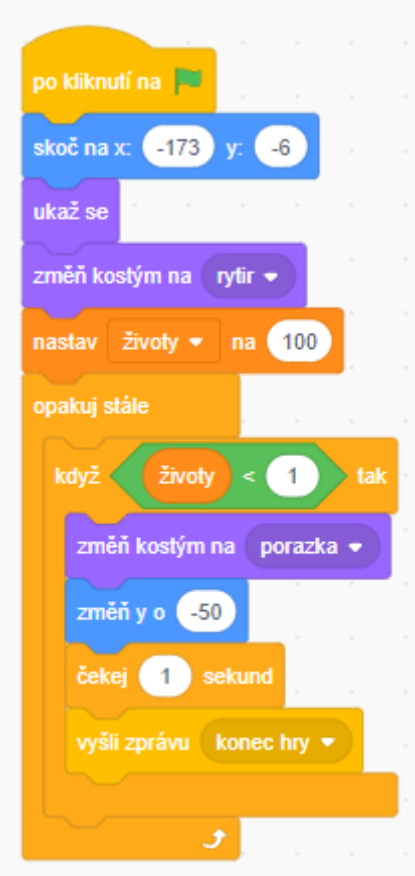
Metodická poznámka

Pokud naprogramujeme pouze mizení, postavy zůstanou schované, dokud nenaprogramujeme při zapnutí scénáře jejich ukázání se.

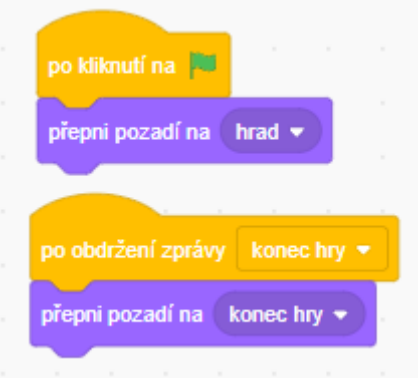
Řešení 9.bodu úlohy

Pokud rytíř souboj prohraje, vyšle zprávu ohledně konce hry. Poté, co tuto zprávu přijmou všechny postavy ve scénáři (včetně draka) tak se schovají pomocí bloku **Ukaž se/Skryj se** ze skupiny bloků **Vzhled**. Tento blok se použije i u opětovného objevení postav po zapnutí scénáře.

Vyřešit nyní zbývá pouze změna pozadí. Vybereme menu scény a do prostoru scénáře vložíme pod blok o konci hry blok s názvem **Přepni pozadí na** ze skupiny bloků **Vzhled**.
V menu tohoto bloku vybereme pozadí s názvem „konec hry“. Obdobně naprogramujeme přepnutí pozadí na pozadí s názvem „hrad“ po zapnutí scénáře.



Obrázek 54 Posílání zprávy u bodu úlohy 9



Obrázek 55 Řešení 9.bodu úlohy

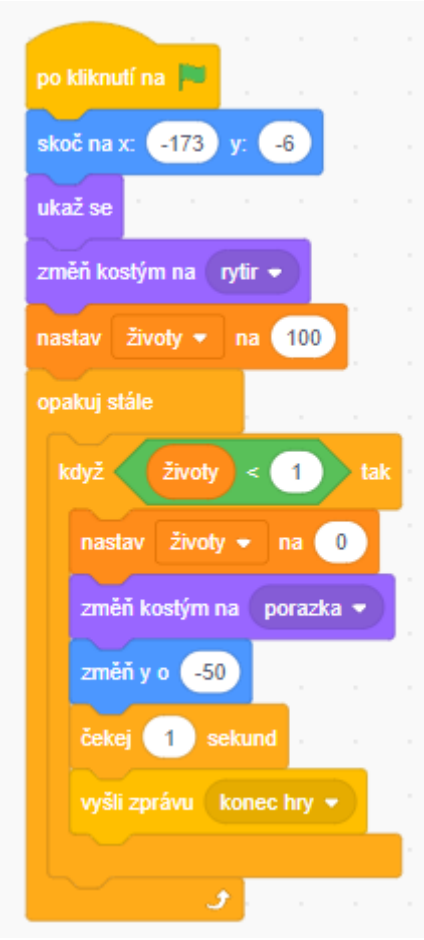
10. Co se stane s životy rytíře pokud bude poražen? Jak naprogramuješ řešení, aby k tomu nedocházelo a kam blok umístíš?

Metodická poznámka

Životy se budou donekonečna odečítat a budou nám vznikat záporné hodnoty.

Řešení 10.bodu úlohy

U rytíře pod blokem **Po kliknutí na zelenou vlaječku** do podmínky umístíme nad blok se změnou kostýmu blok **Nastav proměnnou na** ze skupiny bloků **Proměnné**. Do místa s názvem proměnné vybereme proměnnou značící rytířovy životy a do prázdného místa napíšeme 0.



Obrázek 56 Řešení 10.bodu úlohy

Metodická poznámka

U draka nastane stejná situace s nekonečným odečítáním, opět stačí umístit blok **Nastav proměnnou na** s vybranou proměnnou značící drakovy životy nad jediný existující blok v podmínce. Do prázdného místa opět napíšeme nulu.

Závěr

Žáci si vyzkoušeli použití dalšího z operátorů a zopakovali práci s proměnnými.

7.24. Devátá lekce – ObchodV1 instrukce pro žáky

V dnešní lekci se podíváme do obchodu a nakoupíme si svačinu.

1. Pokud klikneme na marmeládu, muffin a mléko, dají tyto postavy o kliknutí vědět ostatním postavám. Jak to provést?

2. Založ si dvě proměnné: jednu jménem „peníze“ a další má název „součet“. Zvaž, zda bude lepší, aby byly lokální nebo globální. Na začátku nákupu začneme s 400 korunami.

3.1 Jako první si naprogramujeme koupi muffinu. Pokud si muffin vybereme kliknutím, prodavačka se nás zeptá: Kolik muffinů chceš?

3.2 Jeden muffin stojí 25 kč. Do proměnné součet si uložíme celkovou cenu nákupu. Jak ji vypočítáme?

3.3 Prodavačka řekne cenu za nákup a dá vědět ostatním postavám ve scénáři, že došlo k nákupu.

4. Naprogramuj podobně i nákup marmelády za 80 kč.

5. Mléko stojí 18 kč, naprogramujte obdobně jako u marmelády a muffinu.

6.1 Andrea se dozví, že má nyní za muffiny zaplatit a ráda by tak učinila. Zjistí si, zda je cena vyšší než počet peněz, které má u sebe.

6.2 Pokud je cena vyšší, tak řekne: Bohužel nemám dost peněz.

6.3 Pokud je peněz dostatek na zaplacení nákupu, tak se cena nákupu odečte od peněz a Andrea poděkuje a rozloučí se.

Nápověda: Na řešení 6. bodu úlohy použij blok **Když…tak..jinak**.

Použité skupiny bloků: Vzhled, Události, Ovládání, Vnímání, Operátory, Proměnné

7.25. Devátá lekce –ObchodV1 (učitelský návod)

Popis scénáře

V této lekci budeme více pracovat s proměnnými a také se seznámíme s použitím uživatelského vstupu v programu Scratch.

Obrázek 57 Vzhled scénáře ObchodV1

Použité programátorské koncepty

Příkaz

Události

Zprávy

Podmínky

Operátory

Proměnné

Uživatelský input – nové

Nově představené příkazy z programu Scratch

Vnímání – Otázka

Vnímání – Odpověď

Operátory – Operátor *

Když..tak..jinak - Ovládání

Využité bloky

Bloky týkající se vstupu uživatele, podmínkové bloky, bloky týkající se vzhledu, bloky pro práci s proměnnými

Použitý projekt

09 – ObchodV1 – Teacher.sb3	soubor pro učitele
09 – ObchodV1 – Student.sb3	soubor pro žáky

Praktická část

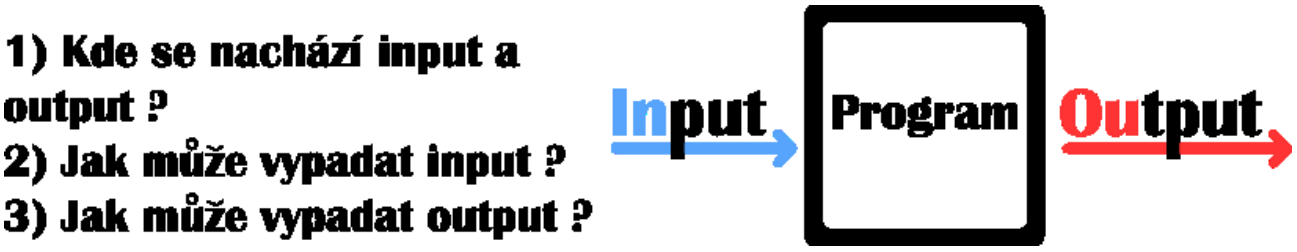
Metodická poznámka

Necháme žáky dodělat úlohu až do třetího bodu řešení. Nyní žákům vysvětlíme, jak vstup od uživatele funguje.

Vstup, nebo také input, uživatele má dvě části. Jako první položíme otázku pomocí stejnojmenného příkazu ve Scratchi. S odpovědí zacházíme jako s jakoukoliv jinou proměnnou. Pokud ji zaškrtneme, tak její hodnotu uvidíme zobrazenou ve scénáři.

Důležitým předpokladem pro schopnost programovat je znalost angličtiny, takže můžeme zadat samostatnou práci. Na tabuli nakreslíme čtverec s šipkou dovnitř a šipkou ven. Chvíli necháme přemýšlet jednotlivce nad tím, kde se bude nacházet output a kde input. Co to vlastně takový output je? Jak se může projevit a jak vypadá? Jak vypadá input?

Poté, co si jednotlivci utřídí své představy je mohou sdělit sousedovi. Necháme chvíli dvojice diskutovat a vyslechneme si mluvčího z každé dvojice a jejich odpovědi. Neříkáme to je špatně, to dobře, správné řešení vysvětlíme až u konce diskuze.



Obrázek 58 Řešení společně s otázkami ohledně konceptu input/output

Input je jakákoliv informace, kterou jsme předali programu ke zpracování. Je uskutečňován pomocí vstupního zařízení, v případě počítače například pomocí klávesnice a stisknutí určitého tlačítka.[36] Příklad z reálného života je například kalkulačka, uživatel tam napíše čísla, operaci, kterou chce s čísly provést a zbytek nechá na programu uvnitř přístroje.

Output je informace zpracovaná počítačem. Nejčastěji ji vidíme na displeji počítače, ale může mít i podobu audio souboru nebo videa.[37] Stále se držíme příkladu kalkulačky, takže output je výsledek, který se nám zobrazí na obrazovce.

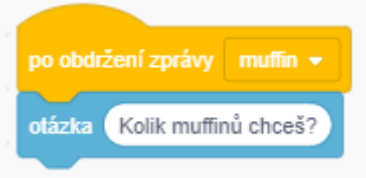
3.1 Jako první si naprogramujeme koupi muffinu. Pokud si muffin vybereme kliknutím, prodavačka se nás zeptá: Kolik muffinů chceš?

Řešení bodu úlohy 3.1

O kliknutí na muffin se dozvíme pomocí zprávy, to je žákům již známý koncept. Následující bloky se tudíž nacházejí pod blokem **Po obdržení zprávy** u postavy prodavačky.

Pod blok **Po obdržení zprávy** ze skupiny bloků **Vnímání** vybereme a vložíme blok **Otázka**. Do prázdného místa napíšeme: Kolik muffinů chceš?

[36] Input. In: *Computer Hope* [online]. [cit. 2022-05-06]. Dostupné z: https://www.computerhope.com/jargon/i/input.htm
[37] Output. In: *Computer Hope* [online]. [cit. 2022-05-06]. Dostupné z: https://www.computerhope.com/jargon/o/output.htm



Obrázek 59 Řešení bodu úlohy 3.1

3.2 Jeden muffin stojí 25 kč. Do proměnné součet si uložíme celkovou cenu nákupu. Jak ji vypočítáme?

Metodická poznámka

Cenu za celý nákup, tedy proměnnou součet vypočítáme vynásobením ceny za jeden kus muffinu krát počet kusů, který se dozvíme z inputu uživatele – ve Scratchi tedy z odpovědi.

Řešení bodu úlohy 3.2

Dalším krokem je práce s takto získanou proměnnou. Z **Proměnných** vybereme blok **Nastav proměnnou na**. Do prázdného místa vložíme blok ***** z **Operátorů**. Proměnnou, které chceme nastavovat hodnotu je součet.

Do levé strany tohoto zeleného bloku vložíme cenu jednoho muffinu, tedy „25“ a do pravé přetáhneme **Odpověď** z **Vnímání**, získanou proměnnou z uživatelského vstupu.

Obrázek 60 Řešení bodu úlohy 3.2

3.3 Prodavačka řekne cenu za nákup a dá vědět ostatním postavám ve scénáři, že došlo k nákupu.

Řešení bodu úlohy 3.3

V bublině lze zobrazovat kromě textu i hodnoty uložené v proměnných v paměti. Toho docílíme, když z **Vzhledu** vybereme blok **Bublina**. Do prázdného místa přetáhneme proměnnou **součet** z **Proměnné**. Ostatním postavám dáme vědět o nákupu pomocí zprávy.

Obrázek 61 Řešení bodu úlohy 3.3

Metodická poznámka

Nyní necháme třídu naprogramovat samostatně podobný kód i pro ostatní potraviny, tedy pro marmeládu a mléko. Řešení bodu úlohy čtyři si ukážeme na tabuli. Vybereme v menu postav postavu jménem Andrea a u ní si představíme nový typ podmínky, který si vysvětlíme pomocí bloků v programu Scratch.

Do prostoru scénáře umístíme blok **Když..tak..jinak** ze skupiny bloků **Ovládání**. V části bloku „Když..tak“ můžeme například kontrolovat, zda se kontrolované číslo rovná tomu, které chceme na vstupu, zda je číslo nenulové, zda je větší či menší než nějaká hodnota s kterým ho porovnáváme – využíváme operátory větší než, menší než a rovná se. Pokud kontrolujeme pouze jednu podmínku, můžeme pro jakoukoliv jinou možnost využít část podmínky „jinak“. Takto učiníme i při řešení bodu úlohy 6.2 (pokud je podmínka platná) a 6.3 (možnost jinak – podmínka platná není).

6.1 Andrea se dozví, že má nyní zaplatit a ráda by tak učinila. Zjistí si, zda je cena vyšší než počet peněz, které má u sebe.

Řešení bodu úlohy 6.1

Jako první si vložíme pod blok se zprávou podmínku, se kterou budeme pracovat. Poslouží nám k tomu výše vysvětlený blok **Když..tak..jinak** ze skupiny bloků **Ovládání**.

Do prázdného místa podmínky vložíme blok **<** z **Operátorů**. Do levé strany, tedy té menší veličiny, přetáhneme **peníze** ze skupiny bloků **Proměnné** a do pravé strany s větší hodnotou přetáhneme **součet**. Vlastně je jedno, jaký z bloků z **Operátorů** použijeme, zda ten co má otevřený zobáček na levou stranu nebo na pravou. Důležité je zde dodržet, že kontrolujeme, zda jsou peníze menší než součet a správně umístit proměnné.

Obrázek 62 Řešení bodu úlohy 6.1

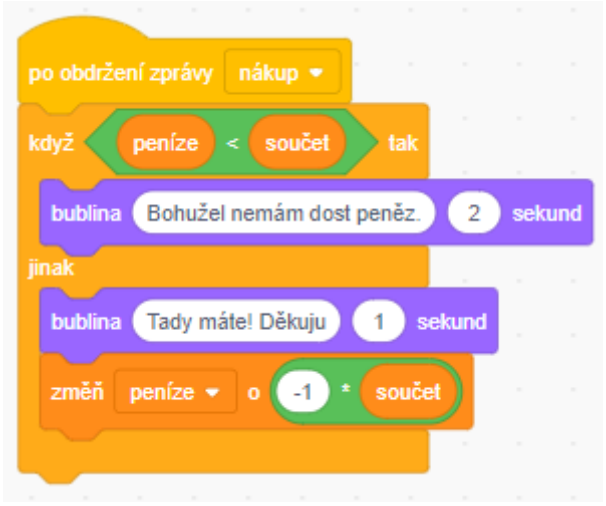
Metodická poznámka

Přidání bubliny můžeme ukázat na tabuli, ale jde pouze o již známou věc. Pokračujme tedy na případ, kdy se podmínky nesplní a uživatel má dostatek peněz na zaplacení nákupu.

6.3 Pokud je peněz dostatek na zaplacení nákupu, tak se cena nákupu odečte od peněz a Andrea poděkuje a rozloučí se.

Řešení bodu úlohy 6.3

Odečítání hodnot proměnných od jiných proměnných bohužel není ve Scratchi tak jednoduché, není to ale neřešitelné. Ze skupiny bloků **Proměnné** vybereme blok **Změň proměnnou o**. Odečítáme od peněz, vybereme z nabídky proměnných tedy **peníze**. Do prázdného místa vložíme blok z **Operátorů** se znamínkem *****. Do levé strany napíšeme -1, tím způsobíme, že číslo bude záporné a bude tedy odečítáno. Do pravé strany vložíme **součet** z **Proměnných**.



Obrázek 63 Řešení bodu úlohy 6.3

Závěr

Žáci se seznámili s dalším typem podmínky a s programátorskými koncepty vstupu a výstupu.

7.26. Devátá lekce – ObchodV2 instrukce pro žáky

V dnešní lekci si upravíme scénář s obchodem z minulé hodiny a naprogramujeme si vystavení účtenky.

1. Uprav zprávu, která se odešle po kliknutí na muffin, mléko a marmeládu takovým způsobem, aby v ní bylo jasně napsáno, jaká potravina byla vybrána.
2. Pokud klikneme na muffin, proběhne dialog, nastaví se proměnná součet a prodavačka řekne cenu. Poté dá postava prodavačky vědět ostatním postavám, že proběhl nákup muffinu. Naprogramuj.
3. Podobně jako u muffinu se dozvědí ostatní postavy také o nákupu mléka. . Naprogramuj.
4. Naprogramuj, aby se podobně jako u muffinu a mléka dozvěděli o nákupu marmelády i ostatní postavy.
5. Pokud proběhne nákup muffinu v pořádku, do účtenky se zapíše slovo muffin a cena za všechny zakoupené muffiny. Naprogramujte.
6. Podobně jako u muffinu se do účtenky zapíše stejným způsobem i mléko. Naprogramujte.
7. Naprogramujte zapsání marmelády do účtenky, podobně jako u mléka a muffinu.
8. Co se stane se s účtenkou pokud scénář několikrát spustíme a pokaždé si něco koupíme? Je možné tomu zabránit a tento děj ošetřit?

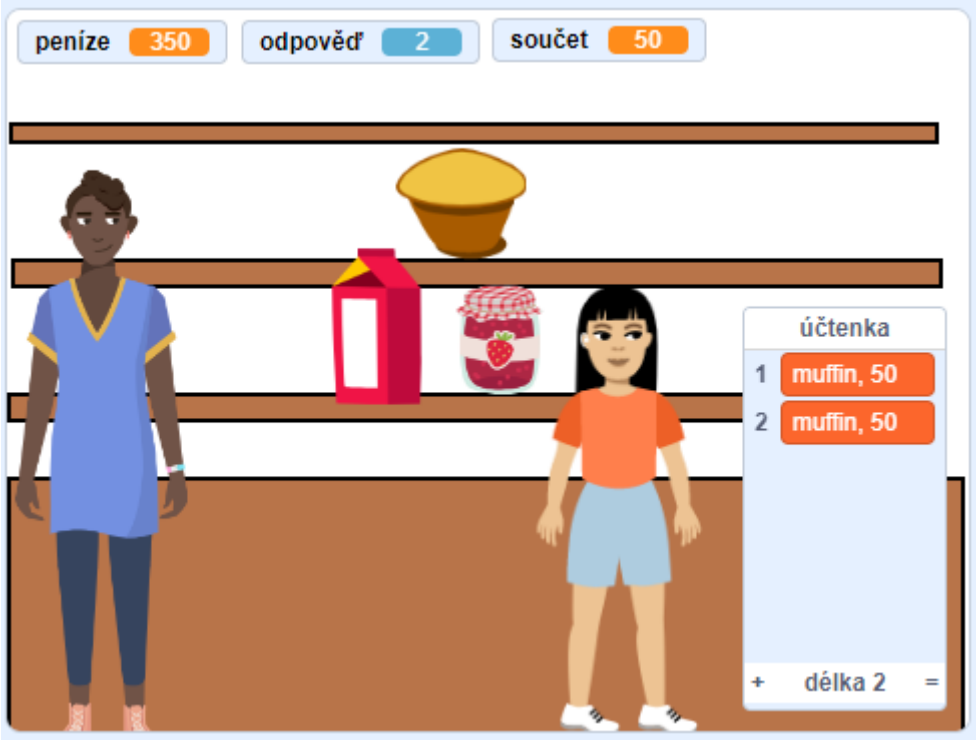
Úkol navíc: Do sešitu si pomocí kombinace kreslení a psaní znázorni, jak takový seznam funguje. Porovnej se sousedem.

Použité skupiny bloků: **Operátory**, **Proměnné**

7.27. Devátá lekce –ObchodV2 (učitelský návod)

Popis scénáře

Nyní si upravíme předchozí verzi scénáře a ukážeme si jak přidávat položky do seznamu v programu Scratch.



Obrázek 64 Vzhled scénáře ObchodV2

Použité programátorské koncepty

Příkaz

Zprávy

Události

Podmínky

Operátory

Proměnné

Uživatelský input

Seznam – nové

Nově představené příkazy z programu Scratch

Operátory – Spoj

Proměnné – Smaž všechno ze seznamu

Proměnné – Přidej k seznamu

Využité bloky

Bloky týkající se práce se seznamem

Použitý projekt

09 – ObchodV2 – Teacher.sb3	soubor pro učitele
09 – ObchodV2 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

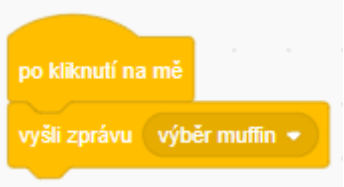
V této lekci si ukážeme zápis slov a číselné hodnoty proměnné do seznamu a také promazání celého seznamu. Seznam s názvem „účtenka“ je pro potřeby prvotního setkání s tímto konceptem v souboru pro studenty založený.

Budeme pracovat se scénářem z minulé hodiny, který si upravíme pro zápis do seznamu. Tento proces bude zahrnovat změnu znění zpráv, aby bylo možné rozlišit zda se jedná o výběr či nákup mléka, marmelády nebo muffinu.

1. Uprav zprávu, která se odešle po kliknutí na muffin, mléko a marmeládu takovým způsobem, aby v ní bylo jasně napsáno, jaká potravina byla vybrána.

Řešení 1.bodu úlohy

Z menu postav vybereme muffin. Pod blokem **Po kliknutí na mě** přepíšeme zprávu uvnitř bloku **Vyšli zprávu** na „výběr muffinu“. Podobně učiníme u postavy mléka a marmelády, samozřejmě budeme odesílat zprávy s příslušnými názvy postav.



Obrázek 65 Řešení 1.bodu úlohy

Metodická poznámka

V této lekci je obzvláště důležité rozlišit v jakých případech odesíláme zprávy o nákupu a výběru.

2. Pokud klikneme na muffin, proběhne dialog, nastaví se proměnná součet a prodavačka řekne cenu. Poté dá postava prodavačky vědět ostatním postavám, že proběhl nákup muffinu. Naprogramuj.

Řešení 2.bodu úlohy

Nyní z menu postav vybereme postavu prodavačky. V bloku **Po obdržení zprávy**, který se zabývá muffinem, změníme přijatou zprávu na nově vzniklou „výběr muffin“. Nyní se v té samé sekvenci bloků přesuneme uplně na konec. Změníme pouze zprávu, která se odešle na konci sekvence bloků pod **Po obdržení zprávy „výběr muffin“**. Její text může například znít „nákup muffin“.

Obrázek 66 Řešení 2.bodu úlohy

Metodická poznámka

Stejne učiníme i u bloků **Po obdržení zprávy** s výběrem mléka a marmelády, třídu necháme do čtvrtého bodu úlohy pracovat samostatně.

5. Pokud proběhne nákup muffinu v pořádku, do účtenky se zapíše slovo muffin a cena za všechny zakoupené muffiny. Naprogramujte.

Řešení 5.bodu úlohy

V menu postav se přepneme na postavu jménem Andrea. Nachází se tu jediná sekvence bloků, která začíná blokem **Po obdržení zprávy „nákup“**. Tento blok nyní upravíme, aby se zabýval zprávou o nákupu muffinu. Z menu uvnitř bloku **Po obdržení zprávy** vybereme zprávu „nákup muffin“.Nyní pouze přidáme blok na konec této sekvence bloků.

Ze skupiny bloků **Proměnné** vybereme **Přidej věc k seznamu**. Vymažeme zástupný název věc a vložíme do prázdného místa **Spoj jablko banán** z **Operátorů**. Opět je k dispozici pouze jeden seznam a to ten s názvem „účtenka“, tak ho vybereme.

Místo jablka napíšeme název kupované věci, v případě muffinu je to samozřejmě muffin. Můžeme zde vyzkoušet mezery a čárky. Jedna z možností zápisu je název koupené věci, čárka a mezera.

Místo banánu vložíme do prázdného místa proměnnou **součet** ze stejnojmenné skupiny bloků.

Obrázek 67 Řešení 5.bodu úlohy

Místo vytváření téměř totožných sekvencí bloků pro řešení bodů úlohy šest a sedm bude stačit když myší najedeme na blok **Po obdržení zprávy „nákup muffin“**, klikneme pravým tlačítkem myši a z menu, které se nyní objevilo vybereme levým tlačítkem myši možnost kopírovat. Nyní máme „v ruce“ zkopírovanou sekvenci bloků, kterou umístíme do volného prostoru scénáře. Tím jsme získali sekvenci bloků pro šestý bod úlohy, to znamená pro mléko.

Celý proces zopakujeme ještě jednou, abychom měli sekvenci i pro marmeládu. Nyní nám zbývá pouze sekvence příslušně upravit, vybrat příslušné zprávy v menu bloku **Po obdržení zprávy** a v bloku **Spoj** přepsat jaký typ potraviny přidáváme k účtence. Žáci nejspíše tuto možnost kopírování bloků již znají, ale pro některé to může být novinka.

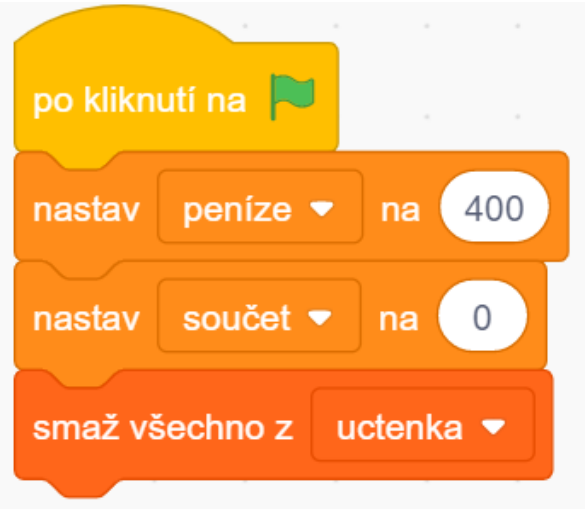
8. Co se stane se s účtenkou pokud scénář několikrát spustíme a pokaždé si něco koupíme? Je možné tomu zabránit a tento děj ošetřit?

Metodická poznámka

Podobně jako se u scénáře ŽelvaV3 hromadily body a bylo nutné je vynulovat, tak i u tohoto scénáře se bude po několika nákupech a spuštěních počet položek zvyšovat. To ovšem nedává smysl, protože v reálném světě nám při nákupu vystaví vždy novou účtenku. Bude tedy nutné seznam s názvem „účtenka“ po stisknutí zelené vlaječky vynulovat.

Řešení 8.bodu úlohy

Ve skupině bloků **Proměnné** nyní vidíme nové možnosti týkající se seznamu. Vybereme **Smaž všechno z** a vložíme příkaz pod některý z bloků se zelenou vlaječkou. Jeden je u pozadí a další u postavy, nezáleží na tom, kam příkaz umístíme – účtenka je globální. Účtenka je rovněž jediný ve scénáři existující seznam a tak je volba jasná.



Obrázek 68 Řešení 1.bodu úlohy

Závěr

Žáci nyní umí seznam vynulovat a přidávat do něj položky.

7.28. Ověření navrženého souboru lekcí ve třídě 6.A

Soubor lekcí jsme ověřovali s šestou třídou ze základní školy Šolleho v Kouřimi. Jako pokus jsme jedné polovině třídy jako úvodní lekci týkající se programu Scratch dali lekci ŽelvaV1 (první lekce, strana 59). To se osvědčilo velice dobře, programování pohybu želvy žáky bavilo. Dokonce jsme se dostali i k naprogramování snědení ovoce želvou. Scénář byli schopní naprogramovat jak rychlejší tak i pomalejší z žáků.

Druhé polovině třídy jsme dali jako úvodní lekci Broadcast (příloha 1, strana 149). Tímto způsobem jsme testovali vhodnost úvodní lekce, kde probíráme programátorský koncept zpráv. Broadcast se ukázal jako nevhodný, v lekci chyběla dynamika a žákům činilo problém pochopit, co po nich vlastně požadujeme. Na jeho místo nyní nastoupila lekce Hmyzí říše (druhá lekce, strana 65), které v podstatě má za úkol to samé – přiblížit žákům koncept odesílání a přijímání zpráv. Právě tato lekce pomohla žákům pochopit jak zprávy v programu Scratch vlastně fungují. Se zprávami zde nakládáme jako s doslovnými zprávami, částmi konverzace.

Z ověření těchto lekcí jsem získala zkušenosti, které jsme s učitelem na základní škole uplatnili na další lekce. Pokud je zadání složitější, je nutné ho žákům vytisknout, již nestačí, že je obvykle po omezenou dobu zobrazené na tabuli. S učitelem jsme rovněž zadání diskutovali ve sborovně.

Lekci Auto (třetí lekce, strana 80) bylo nutné přepracovat, práce ve dvojicích byla podle učitele nesmyslně rozdělena. Jeden žák v ní například programoval pohyb auta vlevo a vpravo a další dopředu a dozadu. Původní verzi s nevhodně rozdělenými úkoly můžeme vidět v příloze (příloha 2, strana 152).

Konzultace s učitelem také ukázala nutnost vytvořit lekci, která by žákům jednoduše vysvětlila programátorský koncept podmínky. Jedná se o lekci Dopravní značky (pátá lekce, strana 90). V jedné z prvních verzí souboru lekcí se tato lekce vůbec nenacházela a místo toho jsme žákům podmínku představovali v kombinaci s operátorem „a“ v lekci Hudební nástroje. Starou verzi této lekce si můžete prohlédnout v příloze (příloha 3, strana 156). Přepracovaná verze lekce Hudební nástroje (šestá lekce, strana 97) byla později zařazena za lekci Dopravní značky.

Další změnou v lekci Hudební nástroje (šestá lekce, strana 97) a Hudební nástrojeV2 (šestá lekce, strana 103) bylo zkrácení audiosouboru, který se přehraje, pokud se dotýkáme hudebního nástroje a zmáčkneme mezerník. Původně šlo o krátkou písničku, nyní se přehraje pouze akord.

V původní verzi lekce ObchodV1 (příloha 4, strana 235) a ObchodV2 (příloha 4, strana 241) se nacházela logická chyba. Postava prodavačky věděla, kolik peněz má zákazník a podle toho se rozhodla buď zboží vydat či oznámit zákazníkovi, že na zaplacení nemá dostatek peněz. Tato chyba je opravena v přepracovaných verzí lekcí ObchodV1 (devátá lekce, strana 124) a Obchod V2 (devátá lekce, strana 132). Přibyla zde postava zákazníka, která si sama zjistí, zda má na zaplacení zboží dost peněz a celý scénář dává větší smysl.

Bylo nutné upravit lekce, kde představujeme jakékoliv ukládání dat do scénáře, tedy lekce kde představujeme žákům proměnné a seznam. Tato úprava se týká lekcí ŽelvaV3 (sedmá lekce, strana 106) a ObchodV2 (devátá lekce, strana 132). Původní verze lekce ŽelvaV3 můžeme vidět v příloze (příloha 5, strana 247), v příloze 4 můžeme vidět původní lekci ObchodV2 (příloha 4, strana 241). V původním zadání bylo žákům zadáno přímo vynulovat proměnnou či seznam, bez toho, aby na tento důležitý krok přišli sami. V lekci ŽelvaV3 bylo také nutné přejmenovat proměnnou na obecnější název – z původně pojmenované proměnné jídlo se staly body.

Ve třídě se nacházeli velice schopní žáci, kteří v programu Scratch již pracovali, či ho velmi rychle pochopili - tací byli schopni vymýšlet i alternativní řešení úkolů, než jaké byly ukázané v instrukcích pro učitele. Rovněž se ve třídě nacházeli žáci, kteří neměli s počítači zkušenosti, s těmi bylo nutné trávit u řešení více času a více je postrkovat. Skoro se zdálo, jakoby se báli, že program rozbijí, pokud něco udělají špatně. Takovým žákům jsem společně s učitelkou vysvětlila, že je možné kdykoliv začít od začátku, není tedy třeba se obávat možné chyby.

Žáci velmi rychle přišli na další možnosti v rámci programu Scratch, například možnost nakreslení vlastní postavy, zvětšování postav a manipulace s nimi. Občas se stalo, že si scénář upravili natolik, že bylo nutné ho naprogramovat znovu celý. Je tedy nutností mít výchozí scénář pro žáky uložený na školním serveru či jiném místě, odkud si ho mohou žáci v případě potřeby stáhnout a načíst do programu Scratch. Ve třídě jsme pracovali s online verzí bez přihlášení a žáci neměli potřebu dělat na počítačích cokoliv jiného co by je mohlo rozptylovat, jako například online hry či sociální sítě. Pokud se odchýlili od zadání, jednalo se ve většině případů o manipulaci s postavami ve scénáři a jejich úpravy v kartě kostýmy.

Největším problémem se ukázalo uložení scénáře do počítače, někteří z žáků ihned pochopili jak ho provést a poté ukazovali postup svým spolužákům. I tak tento pro dospělé jednoduchý úkol zabral poměrně dost času.

8. Závěr

Hlavním cílem práce bylo vytvoření souboru lekcí zaměřených na rozvoj algoritmického myšlení u žáků z pátých a šestých tříd základních škol. Jako první bylo nutné prostudovat odborné texty týkající se algoritmického myšlení. Druhým krokem byl výběr aplikace, která by sloužila jako příprava na programování v programu Scratch. Dalším logickým krokem bylo samotné vytvoření lekcí, které následně byly ověřeny a podle získaných poznatků z praxe upravovány. Finální verze je v práci, původní verze upravovaných lekcích se nacházejí v přílohách. Podrobně se o proběhlých úpravách dočtete v kapitole Ověření navrženého souboru lekcí ve třídě 6.A.

Seznam použitých informačních zdrojů

Odborná literatura a články

CSIZMADIA, Andrew, Prof. Paul CURZON, Mark DORLING a et al. *Computational thinking: A guide for teachers* [online]. Computing At School, 2015. (cit. na s. 11, 12, 13, 14, 15)

SADYKOVA, O.V. a G.G. IL'BAHTIN. The Definition of Algorithmic Thinking. In: Proceedings of the International Session on Factors of Regional Extensive Development (FRED 2019) [online]. Paris, France: Atlantis Press, 2020, 2020, s. - [cit. 2022-06-14]. ISBN 978-94-6252-882-6. Dostupné z: doi:10.2991/fred-19.2020.85 (cit. na s. 15, 16)

SKALKA, Ján, Martin CÁPAY, Gabriela LOVÁSZOVÁ a et al. *Algoritmizácia a úvod do programovania*. Nitra: UKF v Nitre, 2007. ISBN 978-80-8094-217-5. (cit. na s. 16, 19, 20)

Elektronické zdroje

Abstraction. In: BBC [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zttrcdm/revision/1 (cit. na s. 14)

Algorithms (Characteristics, Guidelines & Advantages). In: CodeSansar [online]. [cit. 2022-06-01]. Dostupné z: https://www.codesansar.com/computer-basics/algorithms.htm (cit. na s. 18)

Algorithms. In: BBC [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zpp49j6/revision/1 (cit. na s. 16)

Algoritmus. In: GJŠ Zlín [online]. [cit. 2022-06-01]. Dostupné z: https://www.gjszlin.cz/ivt/esf/algoritmizace/algoritmus.php (cit. na s. 18)

Block: What Does Block Mean?. In: Techopedia [online]. [cit. 2022-06-29]. Dostupné z: https://www.techopedia.com/definition/17978/block-programming (cit. na s. 54)

CodeCombat - výběr oblasti. In: CodeCombat [online]. [cit. 2022-05-17]. Dostupné z: https://codecombat.com/play (cit. na s. 49)

Decomposition. In: BBC [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zqqfyrd/revision/1 (cit. na s. 12)

IMyšlení [online]. [cit. 2022-05-22]. Dostupné z: https://www.imysleni.cz/ (cit. na s. 21)

Input. In: Computer Hope [online]. [cit. 2022-05-06]. Dostupné z: https://www.computerhope.com/jargon/i/input.htm (cit. na s. 127)

Introduction to computational thinking. In: BBC [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1 (cit. na s. 11)

MC-WEIGH MURPHY, Anna. The One About Algorithmic Thinking in Computational Thinking. In: Equip Learning [online]. [cit. 2022-06-01]. Dostupné z: https://equip.learning.com/algorithmic-thinking-computational-thinking/ (cit. na s. 15)

Output. In: *Computer Hope* [online]. [cit. 2022-05-06]. Dostupné z: https://www.computerhope.com/jargon/o/output.htm (cit na s. 127)

PODPORA ROZVOJE INFORMATICKÉHO MYŠLENÍ. In: RVP [online]. [cit. 2022-05-22]. Dostupné z: https://digiskola.rvp.cz/dvz/detail-projektu?id=4 (cit. na s. 21)

Pattern recognition. In: BBC [online]. [cit. 2022-06-01]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zxxbgk7/revision/1 (cit. na s. 13)

Proměnné. In: Úvod do programování [online]. [cit. 2022-05-06]. Dostupné z: https://mrlvsb.github.io/upr-skripta/c/promenne/promenne.html (cit. na s. 108)

Scratch - Teacher Accounts. In: Scratch [online]. [cit. 2022-05-22]. Dostupné z: https://resources.scratch.mit.edu/www/guides/en/scratch-teacher-accounts-guide.pdf (cit. na s. 22)

Scratch. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-05-22]. Dostupné z: https://cs.wikipedia.org/wiki/Scratch (cit. na s. 21)

VAIS, Jan. Rozvoj algoritmického myšlení žáků základních škol. Digitální repozitář Univerzity Karlovy [online]. [cit. 2022-06-01]. Dostupné z: https://dspace.cuni.cz/handle/20.500.11956/125706 (cit. na s. 20)

Vlastnosti algoritmu. elementárnost. determinovanost. rezultativnost. konečnost. hromadnost. efektivnost. In: DOCPLAYER [online]. [cit. 2022-06-01]. Dostupné z: https://docplayer.cz/134660472-Vlastnosti-algoritmu-elementarnost-determinovanost-rezultativnost-konecnost-hromadnost-efektivnost.html (cit. na s. 18)

Seznam příloh

Seznam obrázků

Příloha 1

Broadcast

Projekt broadcast

Oblast Scratche: vzhled, události

Používané příkazy:

Vzhled – další kostým, další pozadí, změň pozadí na x, změň kostým na x

Ovládání – vyšli zprávu „x“

V tomto projektu si ukážeme jak funguje vysílání zpráv. Máme předpřipravené grafické podklady, knoflíky s nápisy, které určují jejich budoucí funkci a ikony měsíce, slunce a mraku, stejně jako noční a denní oblohu. Je vhodné žáky upozornit, s jakými částmi Scratche mají pracovat.

Úkol:

- Nastavte knoflíky, aby po kliknutí vyslali zprávu, která odpovídá jednoznačně jejich popisku, například knoflík noc vyšle zprávu noc.
- Nastavte ikonku mraku, aby po přijmutí zprávy mračno či jasno zobrazila či nezobrazila mrak.
- Nastavte ikonku slunce, aby po přijmutí zpráv den změnilo kostým na slunce a pozadí na modrou oblohu, to samé ale s měsícem a noční oblohou udělejte u obdržení zprávy noc.
- Po obdržení zprávy den/noc, tedy přepínače, nastavte přepnutí na další kostým.

Řešení:

1. Klikneme na knoflík s nápisem den, který se nachází pod okénkem programu. Ze žluté nabídky vybereme blok **Po kliknutí na mě**. Z nabídky té samé barvy vyberte blok **vyšli zprávu** a **Nová zpráva**. Zde napíšeme, jakou zprávu pro všechny posluchače chceme vyslat. Tento mechanismus funguje podobně jako čekárna u doktora, kde skupina lidí vyčkává, dokud nebude řečeno jejich jméno, tedy zpráva pro ně a můžou na ni zareagovat. Zde by bylo vhodné nakreslit Obrázek na tabuli a ukázat žákům tento proces v programu Scratch.
2. Obdobně to uděláme i u ostatních knoflíků, pouze s obměněnými zprávami.

3. Nyní budeme pracovat s ikonkou mraku. K tomu budeme potřebovat část příkazů v části vzhled. Zelenou vlaječkou program zapneme, takže po jejím stisknutí (opět příkaz, který najdeme v ovládání) nastane základní nastavení. Mrak má být po zapnutí skryt, takže z nabídky je vybráno **Skryj se**.
4. Z událostí vybereme **Po obdržení zprávy jasno** a **Po obdržení zprávy mračno** a podle názvu vybereme z menu vzhledu bud **ukaž se** nebo **skryj se.** Nyní pracujeme s zobrazením či nezobrazením obrázku, na který jsme klikli, tedy mraku.
5. Teď nastal čas se soustředit na ikonku slunce. Opět ho vybereme a z událostí vybereme **Po obdržení zprávy den** a **Po obdržení zprávy noc**. Žákům ukážeme jak nastavit pouze jeden z nich a druhý by měli zvládnout sami. V kartičce vzhled vybereme **změň kostým na slunce** a **přepni pozadí na modré nebe**. Podobně nastavíme po obdržení zprávy noc kostým na měsíc a pozadí na noční nebe.
6. Nyní nastavíme knoflík den/noc. Ten bude fungovat jako přepínač. Jeho funkčnosti docílíme tak, že **po obdržení zprávy den/noc** přidáme z vzhledu bloky **další kostým** a **další pozadí**.

Příloha 2

Původní verze lekce Auto

Třetí lekce – Auto učitelský návod

Popis scénáře

V tomto cvičení si zopakujeme znalosti z minulých lekcí týkající se zpráv a pohybu avataru. Využijeme u něho práci ve dvojicích.

Použité programátorské koncepty

Příkaz

Zprávy

Použité oblasti Scratche

Pohyb

Vzhled

Události

Ovládání

Vnímání

Využité bloky

Bloky týkající se zpráv, po stisku klávesy, přejdi na vrstvu, ukaž se/skryj se, čekej dokud nenastane, dotýkáš se avataru, bloky týkající se pohybu

Použitý projekt

03 – Auto – Teacher.sb3	soubor pro učitele
03 – Auto – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Právě protože v tomto úkolu jde primárně o zopakování již naučených znalostí, můžeme u něho využít práci ve dvojicích. Žáky můžeme rozdělit například podle toho jak již sedí ve třídě do párů (případně podle jakéhokoliv způsobu uznáme za vhodné). V instrukcích máme dvě skupiny úkolů, jeden se týká pohybu a další mizení hvězdiček. Ty si zopakují oba dva zpárovaní a společně přijdou na řešení třetího úkolu změny vzhledu avataru auta.

3. Po kliknutí na knoflík s nápisem Změň auto se přepne avatar auta na další kostým. Na tomto úkolu pracujte společně.

Metodická poznámka

Tento úkol, ačkoliv jde o něco nového, je formulován tak, že by mělo být jasné, kde hledaný blok nalézt a jak se bude jmenovat. Necháme žáky pracovat chvíli ve dvojicích, obejdeme si je a zjistíme, jak na tom jsou. Pokud po čase potřebném pro dokončení úkolu většina třídy stále neví, ukážeme postup na tabuli.

Všem dvojicím řekněme, že nastane-li situace, kdy jeden z dvojice ví jak toto naprogramovat a druhý ne, mají si zkusit princip příkazu mezi sebou vysvětlit. V tomto cvičení celkově nefungujeme jako hlavní vedoucí hodiny, spíše jako pomocník, který zná řešení a může žákům pomoci, pokud je potřeba.

Řešení 3. bodu úlohy

Pod blok **Po kliknutí na mě** z **Událostí** umístíme blok **Další kostým**, tentokrát z oblasti **Vzhled**.

Přidejte na obrazovku další knoflík, který tentokrát bude klikem přepínat pozadí.

Metodická poznámka

Tuto část by jsme měli ukázat na tabuli, ačkoliv se jedná o úkol navíc. Ta samá situace nastává u přidání pozadí. Měli by jsme ovšem ukázat pouze zkopírování a otevření okénka pro editaci avataru. Také by jsme se měli zmínit, že společně s vzhledem jsme zkopírovali i bloky, které k němu náleží.

Řešení úkolu navíc

Vybereme ikonu knoflíku pod okénkem, kde vidíme spuštěný program. Pravým tlačítkem na ni klikneme a vybereme zkopírovat. S avatarem můžeme různě hýbat a je možné i přepsat, co je na knoflíku napsané překliknutím se do okna kostýmy a výběrem písma na editovaném obrázku. Kromě vzhledu jsme zkopírovali i příkazy, které u sebe měl originál. Ty editujeme jako obyčejně.

Pozadí budou tři – původní a dvě pozadí, každé vytvořené jedním ze členů skupiny.

Metodická poznámka

Opět ukážeme pouze jak pozadí přidat a nabádáme k vyzkoušení jednotlivých možností.

Řešení úkolu navíc

Pod okénkem programu klikneme na obdélník s nadpisem scéna. Tímto jsme si zpřístupnili kartu pozadí, kde po přejetí myší po modrém kolečku máme na výběr z nahrání obrázku z počítače, náhodného pozadí, nakreslení vlastního motivu či vyhledání pozadí v databázi Scratche.

Závěr

Žáci si zopakovali výklad z minulých hodin a do jejich programovacího repertoáru přibyly další příkazy týkající se změny vzhledu.

Příloha 3

Původní verze Hudební nástrojeV1

Čtvrtá lekce – Hudební nástroje učitelský návod

Popis scénáře

V tomto cvičení si zopakujeme pohyb avataru a vysvětlíme koncept podmínky v programování.

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky - nové

Použité oblasti Scratche

Pohyb

Vzhled

Události

Ovládání

Zvuk

Vnímání

Operátory - nové

Využité bloky

Po stisku klávesy, přejdi na vrstvu, dotýkáš se avataru, bloky týkající se pohybu, bloky týkající se zvuku, nastav směr, nastav otáčení, klávesa stisknuta

Použitý projekt

04 – Hudební nástroje – Teacher.sb3 soubor pro učitele

04 – Hudební nástroje – Student.sb3 soubor pro žáky

Praktická část

Metodická poznámka

Při plnění bodu číslo dva můžeme žáky upozornit na kostýmy postavy, které již tvoří iluzi pohybu. Proto je zde přepínání pouze pomocí další kostým.

Až do čtvrtého bodu by nemělo dělat řešení úloh žákům žádný problém. Je dobré vysvětlit a popřípadě i nakreslit, dva případy postav – můžeme na ně mít pohled shora či ze strany. U druhého případu je nutné nastavit otáčení vlevo-vpravo.

Nastavíme otáčení vlevo-vpravo.

Řešení 1.bodu úlohy

Z Pohybu vybereme Nastav otáčení. U něj vybereme možnost vlevo-vpravo.

Jaký avatar v programu bude v pozadí a popředí? Krátce popřemýšlej a poté naprogramuj pod blok signalizující zapnutí programu u všech avatarů.

Metodická poznámka

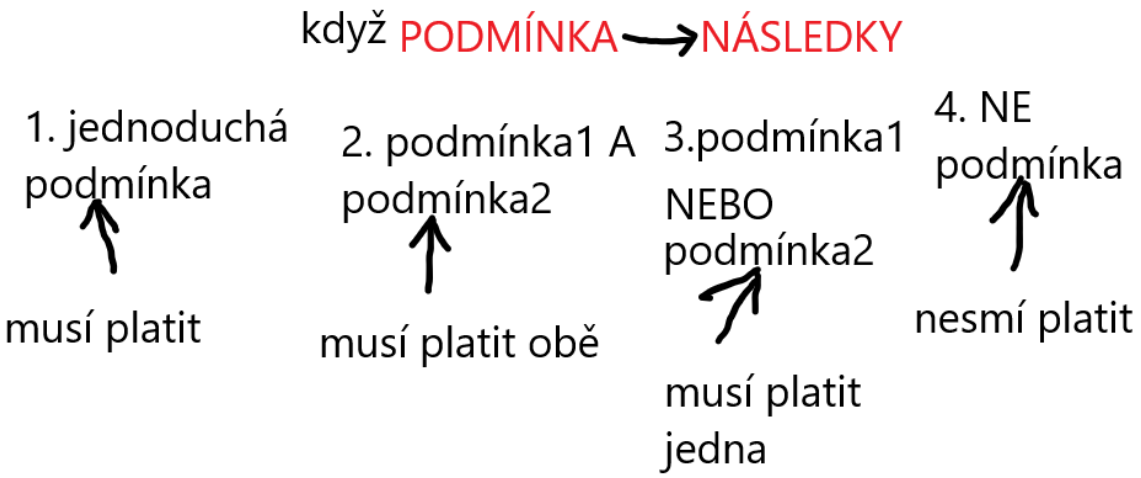
Jako jsme si říkali již dříve, většinou je avatar postavy v popředí a předměty, s kterými má interakce jsou v pozadí. Řešení je jasné, ale v tomto případě musíme postavě přiřadit blok, který tuto funkci plní – narozdíl od předchozích cvičení kde stačilo dát tento blok pouze k předmětu interakce.

Když se postava...	**Tak udělej..**
dotýká piana a klávesa mezerník stisknuta	zahraj zvuk piano
dotýká se kytary a klávesa mezerník stisknuta	zahraj zvuk kytara

Metodická poznámka

Ve třetí části cvičení si připomeneme podmínku v češtinářské podobě. Typicky se skládá z výroku, který má na začátku spojku pokud nebo když. Po ní následuje situace, která musí nastat, aby se stalo pokračování věty – to často začíná spojkou tak.

Podmínka v programování je velmi podobná. Příklad podmínky v programování může být například: pokud je zmáčknutá klávesa k, přepni pozadí na další. Kromě případu s jednou podmínkou můžeme mít podmínek více a to v různých vztazích. Opět se zde jedná o spojky. Lépe se to vysvětlí obrázkem a následovnou prací žáků, kde si promyslí své nápady, napíší si je a poté proberou se sousedem.



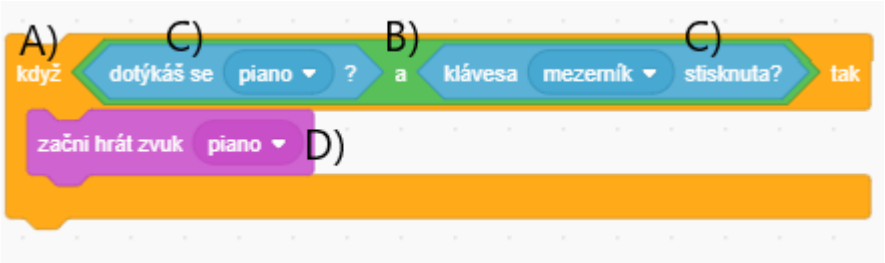
Obrázek 69 Obrázek popisující podmínky

Nyní by si žáci měli napsat dva příklady ke každé z podmínek, jeden z reálného života a druhý, který by využili v programu Scratch nebo v obecném pseudo - programovacím jazyku. Poté co budou s psaním hotovi a učitel tak určí, proberou své doměnky se sousedem.Učitel poté vybere, pokud možno náhodně, jednoho ze dvojice a ten za oba její členy řekne nejpovedenější podmínku z reálného života a programování.

Řešení 3.bodu úlohy

Ještě než umístíme samotné podmínky, je nutné je obalit v cyklu, který bude platit po celý běh programu. Tento blok se jmenuje **Opakuj stále** a najdeme ho v **Ovládání**. Více o cyklech si řekneme v dalších hodinách.

Toto řešení je lepší zobrazit obrázkem, který si blíže popíšeme.



Obrázek 70 Řešení podmínky

A) Jako první do cyklu **Opakuj stále** umístíme ze stejné oblasti, **Ovládání**, podmínkový blok **Když..tak**.

B) Pracujeme se dvěma podmínkami, takže nám místo nebude stačit. Z **Operátorů** vybereme blok se dvěma prázdnými místy a spojkou „a“ mezi nimi. Tyto podmínky musí platit obě, aby proběhly příkazy uvnitř.

C) Přepneme se do oblasti **Vnímání** a vybereme **Klávesa stisknuta**. Tu umístíme do libovolného prázdného místa, na pořadí nezáleží. Klávesu, jejíž stisknutí kontrolujeme, můžeme vybrat v nabídce. Do druhého dáme **Dotýkáš se** a vybereme název pro avatar piana.

D) Vevnitř podmínky se nachází blok, který se stane pouze pokud jsou obě výše zmíněné podmínky splněny.

Metodická poznámka

Je dobré aspoň v krátkosti vysvětlit, že vše, co se nachází uvnitř tohoto cyklu bude opakováno po dobu celého programu – to znamená, že se bude od zapnutí po vypnutí programu kontrolovat zda se podmínky splnily.Ke konci ukazování můžeme opět připomenout, že se zvuk piana přehraje pouze tehdy, pokud obě podmínky platí. Ukažte první podmínku a zbytek nechejte na žácích.

4. Po kliknutí na knoflík se zastaví všechny zvuky.

Řešení 4.bodu úlohy

Ze **Zvuku** vybereme blok s názvem **Začni hrát zvuk** s názvem nástroje, kterého se podmínka týká.

Čtvrtá lekce – Hudební nástroje instrukce pro žáky

1. Naprogramuj podle tabulky pohyb postavy.

Po stisku klávesy w	• Pohne se dopředu. • Přepne se na další kostým.
Po stisku klávesy šipka vlevo	• Otočí se vlevo.
Po stisku klávesy šipka vpravo	• Nastavíme otáčení vlevo-vpravo. • Otočí se vpravo.

2. Jaký avatar v programu bude v pozadí a popředí? Krátce popřemýšlej a poté naprogramuj pod blok signalizující zapnutí programu u všech avatarů.

3. Naprogramuj podmínky podle následující tabulky.

Když se postava...	**Tak udělej..**
dotýká piana a klávesa mezerník stisknuta	zahraj zvuk piano
dotýká se kytary a klávesa mezerník stisknuta	zahraj zvuk kytara

Úkol navíc: V programu zbývá ještě jeden hudební nástroj. Naprogramujte ho podobně jako předcházející.

4. Po kliknutí na knoflík se zastaví všechny zvuky.

Použité oblasti Scratche: Ovládání, Vzhled, Události, Vnímání, Zvuk, Pohyb, Operátory

Příloha 4

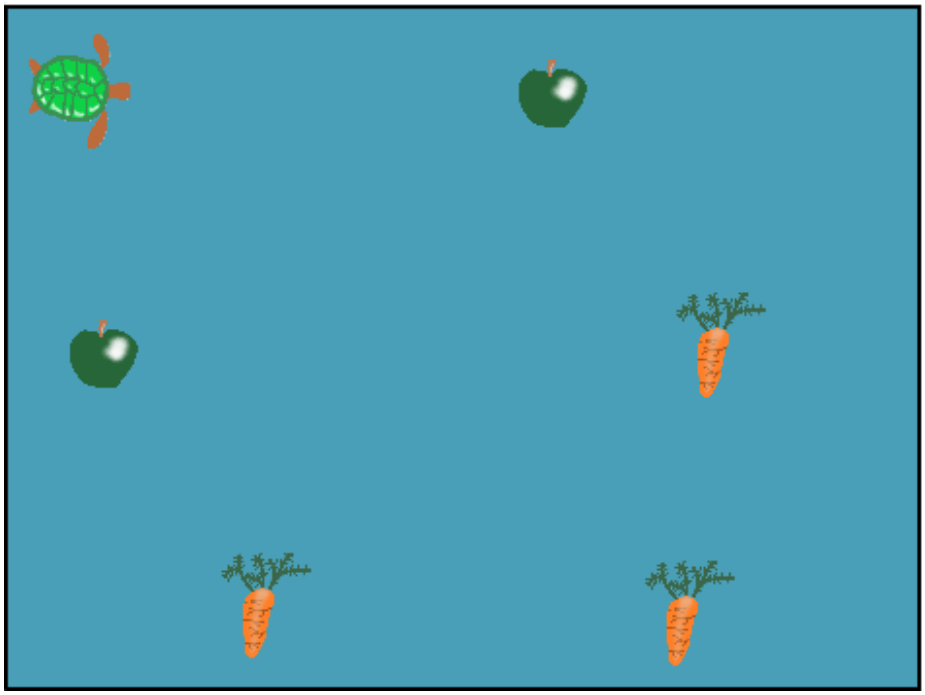
Původní verze souboru lekcí

Původní soubor lekcí

První lekce – ŽelvaV1 učitelský návod

Popis scénáře

V tomto cvičení si ukážeme ovládání želvy pomocí kláves.



Obrázek 71 Vzhled scénáře ŽelvaV1

Použité programátorské koncepty

Příkaz – nový

Nově představené bloky z programu Scratch

Události - Po stisku klávesy

Pohyb – Nastav směr

Pohyb – Dopředu o x kroků

Využité bloky

Bloky týkající se pohybu

Použitý projekt

01 – ŽelvaV1 – Teacher.sb3	soubor pro učitele
01 – ŽelvaV1 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Poté, co žáci otevřou 01 – ŽelvaV1 – Student.sb3, uvidí ve scénáři sekvenci bloků. Nyní bychom měli vysvětlit k čemu přesně tato sekvence a jí podobné slouží – její účel je umístění želvičky na začátek mapy při spuštění scénáře, jak značí zelená vlaječka.

Na tento důležitý prvek ovládání Scratche můžeme upozornit obkroužením myší jak u počátečního bloku 

, tak i u tlačítka s vlaječkou nad scénou. Kliknutím na toto tlačítko námi napsaný scénář spouštíme. Dále v tomto dokumentu budeme tomuto bloku říkat **Po stisknutí zelené vlaječky**.

Možné problémy

Během ověřování se stalo, že si děti aktivně tyto předpřipravené bloky vymazali a pak se jim želva ztratila ze scénáře. Je dobré mít na serveru, ze kterého si zadání berou k sobě na plochu, vždy zálohu zadání.

2. Naprogramujte si otáčení želvy podle šipek.

Metodická poznámka:

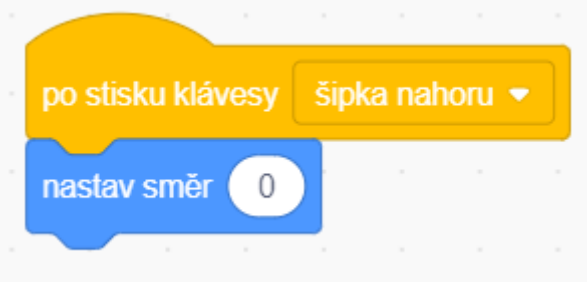
Ještě než se pustíme do samotného programování, je nutné vysvětlit co to vlastně je základní stavební kámen programování – tedy příkaz. Můžeme jej přirovnat například k jednomu kroku výroby nějakého předmětu či k jednomu ze kroků receptu. Program se podobně jako postup skládá z vícero příkazů, je to návod jak někdo nebo něco postupuje. Ve Scratchi program nazýváme scénář a zadáváme jeho použitím želvě (či jakékolic jiné postavě) instrukce, podle kterých se chová. Příkazy se ve Scratchi nacházejí ve formě bloků, budeme je tedy nazývat bloky.

Na tabuli můžeme například začít kreslit první čáru čtverce a říct, že první příkaz pro jeho nakreslení by zněl: jdi dopředu. Můžeme použít libovolnou techniku získávání odpovědi od žáků a postupně si s nimi čtverec „naprogramovat“.

Řešení 2.bodu úlohy

Z skupiny bloků **Události** vybereme **Po stisku klávesy** a umístíme jej do prostoru scénáře. V menu bloku **Po stisku klávesy** si vybereme na jakou klávesu vlastně bude želva reagovat, vybrat můžeme například šipku nahoru.

Nyní se přepneme do skupiny bloků **Pohyb**. Zde vybereme blok **Nastav směr** a na kruhu nastavíme jakým směrem se želva po stisku klávesy otočí. Obdobně nastavíme pohyb pro ostatní směry, nám stačí pouze 4 základní – nahoru, dolů, vlevo a vpravo.

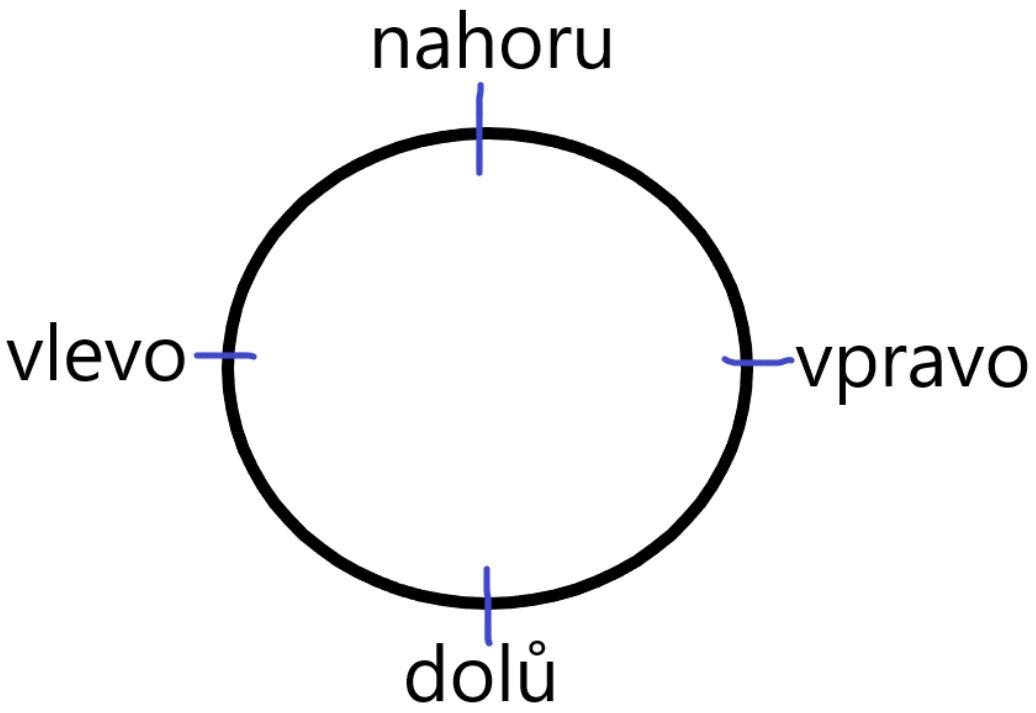


Obrázek 72 Řešení 2.bodu úlohy

Metodická poznámka

Událost jsme vlastně viděli již v prvním bodě, zelená vlaječka je událost po spuštění. Událost je vlastně akce, která vyvolá reakci scénáře, podobně jako když zazvoní poslední zvonek ve škole v pátek, žáci se mají chuť rozutéct po škole a již chtějí být doma.

Blok po stisku klávesy pracuje s zachytáváním událostí – blok čeká například na stisknutí určité klávesy, třeba právě šipky nahoru. Jakmile tato událost nastane, začnou se provádět další bloky, které se nachází pod blokem **Po stisku klávesy**.



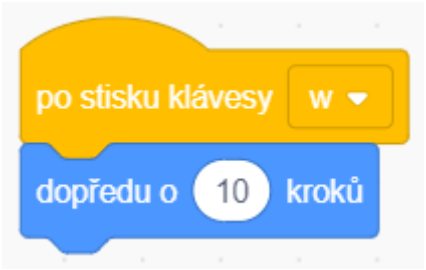
Obrázek 73 Příklad nakresleného příkazu Nastav směr na tabuli

Dále je vhodné nakreslit kruh z výběru **Nastav směr** na tabuli a zakreslit na něj, kde leží námi programované směry. Měli by jsme zmínit, že je dobré si výběr směru v rámci kruhu sám vyzkoušet a pozorovat, jak na to želva reaguje. Příklad, jak by kruh mohl vypadat se nachází nad tímto textem.

3. Jako každá správná želva by se i ta naše měla pohybovat. Stane se tak po stisknutí klávesy w, naprogramujte si řešení a vyzkoušejte.

Řešení 3.bodu úlohy

Obdobně jako jsme naprogramovali otáčení přiřadíme ke klávese „w“ blok **Dopředu o 10 kroků** z **Pohybu**. Celý scénář si pro jistotu restartujeme pomocí stisku zelené vlaječky a můžeme vyzkoušet, jak funguje.



Obrázek 74 Řešení 3.bodu úlohy

Metodická poznámka

S touto částí zadání většinou moc problémů nebývá. Během konstrukce celého cvičení je možné že si žáci zvětší želvu či jinou postavu do obřích rozměrů, případně si je vymažou. Měli bychom zdůraznit důležitost pravidelného ukládání, ukázat postup na interaktivní tabuli či projektoru a klidně jej několikrát za hodinu připomenout – i když na školním serveru můžeme mít původní verzi počátečního stavu scénáře, žáci by vymazáním postavy či snad neuložením celého scénáře ztratili výsledek celého svého snažení a mohlo by je to demotivovat. Uložení provedeme kliknutím na tlačítko Soubor ze základního menu Scratche, dále vybereme Ulož do svého počítače a vhodné umístění. Soubor lze přepsat i když jej samotný máme v programu Scratch otevřený, stačí pouze vybrat jakožto místo uložení právě otevřený scénář v jeho příslušném adresáři.

Závěr

Žáci by měli nyní chápat, jak fungují události na základě stisknuté klávesy a jak používat základní příkazy týkající se pohybu postavy.

První lekce – ŽelvaV1 instrukce pro žáky

1. Počáteční bloky s vlaječkou ponechejte v programu a nemažte ho.
2. Naprogramujte si otáčení želvy podle šipek.
3. Jako každá správná želva by se i ta naše měla pohybovat. Stane se tak po stisknutí klávesy w, naprogramujte si řešení a vyzkoušejte.

Nepočítačový úkol: Jak by vypadal program pro nakreslení trojúhelníku?

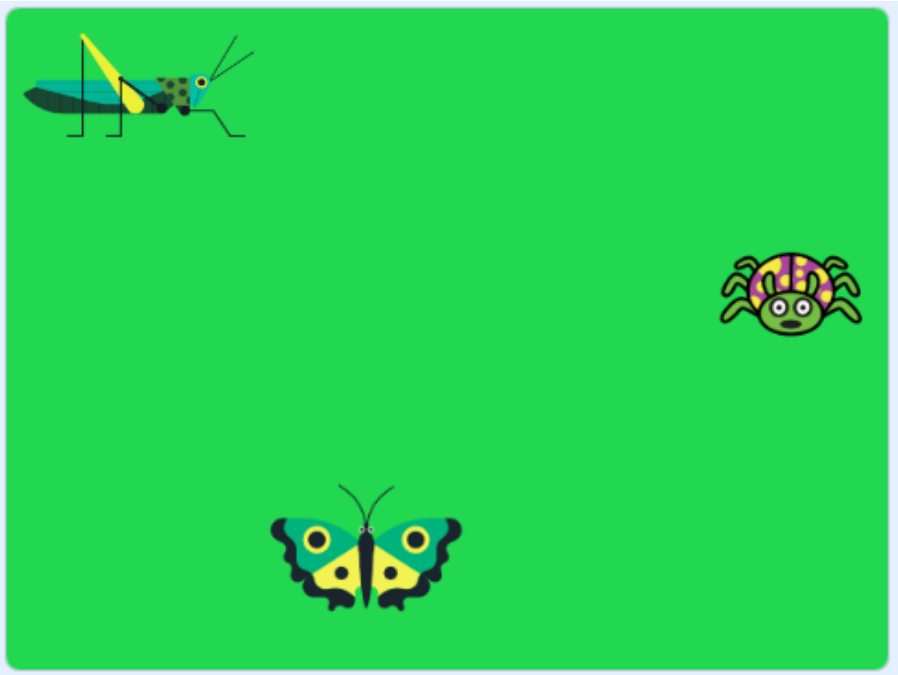
Úkol navíc: Zkuste si naprogramovat želvu, aby se po stisku šipky natočila i posunula daným směrem.

Použité skupiny bloků: **Pohyb**, **Události**

Druhá lekce – Hmyzí říše učitelský návod

Popis scénáře

V tomto cvičení si vyzkoušíme další příkazy z skupiny bloků **Pohyb** a naprogramujeme jednoduchý rozhovor postav.



Obrázek 75 Vzhled scénáře Hmyzí Říše

Použité programátorské koncepty

Příkaz

Zprávy - nový

Nově představené bloky z programu Scratch

Události – Po kliknutí na mě

Události – Vyšli zprávu

Události – Po obdržení zprávy

Vzhled – Změň kostým na

Vzhled – Bublina po dobu x sekund

Pohyb - Klouzej x sekund na

Pohyb – Otoč se p x stupňů

Využité bloky

Bloky týkající se zpráv, dialogů postav a kostýmů, bloky zabývající se pohybem

Použitý projekt

02 – HmyziRise – Teacher.sb3	soubor pro učitele
02 – HmyziRise – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Pro začátek bude důležité vysvětlit způsob, jakým jsou úkoly zadávány. Levou stranu tabulky si rozdělíme na dvě části, před závorkou a obsah závorky. Před závorkou je název postavy, se kterou budeme pracovat. Uvnitř závorky je blok z skupiny bloků **Události**. Tento příkaz značí událost, na kterou scénář čeká a jakmile tato událost nastane, budou následovat příkazy na levé straně tabulky, viz. minulá lekce a zachytávání událostí zde zmíněné. Tento styl zadávání úkolů je vhodné třídě přiblížit, v následujících lekcích je hojně využíván.

V pokročilejších lekcích je tento styl zápisu používán i pro podmínky. Pokud tomu tak je, je to nad tabulkou s instrukcemi napsáno.

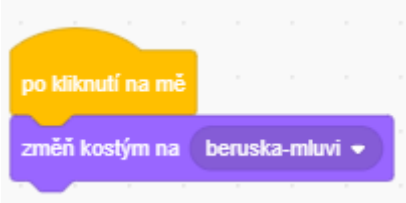
1. Změní kostým na beruska-mluvi.

Metodická poznámka

V každém scénáři se nacházejí herci, to jsou v našem případě postavy jako v minulém cvičení želva či v nynějším hmyz. Postavy ovšem nemusejí mít pouze jednu podobu. Pokud kliknutím vybereme jednu z postav, zpřístupní se nám nad skupinami bloků karta Kostýmy, jako logo má Obrázek štětce. Po kliknutí na ni vidíme všechny kostýmy, které má vybraná postava k dispozici. Kostýmy v tomto podmenu můžeme přidávat, kreslit na ně a podobně, a i proto je lepší žákům pouze ukázat, jaké kostýmy jsou k dispozici a více se touto kartou nyní nezaobírat. Žáci na to stejně přijdou sami a zdlouhavé vybírání postav či dokonca nakreslení vlastní by narušilo hodinu.

Řešení 1.bodu úlohy

Z skupiny bloků **Vzhled** vybereme příkazový blok **Změň kostým na**. V nabídce jsou jasně pojmenované kostýmy.



Obrázek 76 Řešení 1.bodu úlohy

Metodická poznámka

Další připomínka směrem k žákům může být ohledně důležitosti jasného pojmenování kostýmů a postav. Tato praktika je v programování důležitá kvůli přehlednosti. Pokud budeme mít u postavy například šest různých kostýmů a budeme mezi nimi při různých událostech chtít přepínat, pouze číselné názvy nám nepomohou v jednoduchém rozhodování, jaký kostým u menu bloku **Změň kostým na** použít. Přejmenování jednoho z kostýmů by si měli žáci vyzkoušet, stejně jako přejmenování postavy.

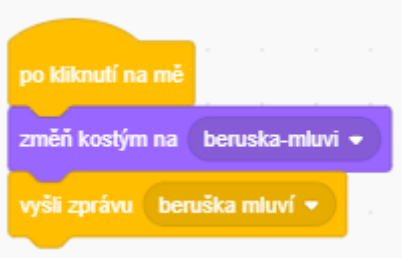
2. Pošle zprávu „Beruška mluví“.

Metodická poznámka

Nyní se dostáváme k hlavní náplni hodiny. Ještě než bude možné pracovat s programátorským konceptem zpráv, je nutné žákům tento koncept přiblížit v realitě. Mezi dobré příklady patří například zpráva od kamaráda, že chce jít ven. My reagujeme na zprávu například odepsáním že nám není dobře, či převléknutím do venkovního oblečení a odepsáním že už jdeme. Zde je možné nechat žáky mezi sebou chvilku diskutovat ohledně toho, co je dobrým příkladem podobného chování v reálném životě. Třeba se právě od nich naučíme lepší vysvětlení než výše jmenované.

Řešení 3.bodu úlohy

Z **Událostí** vybereme blok **Vyšli zprávu**. Do mezery napíšeme, co přesně chceme poslat.

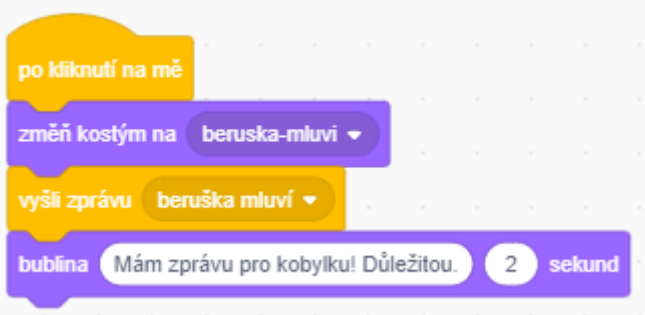


Obrázek 77 Řešení 3.bodu úlohy

3. Řekne, že má zprávu pro kobylku.

Řešení 3.bodu úlohy

Ze **Vzhledu** vybereme **Bublina „“ x sekund**. Do první mezery napíšeme text a do druhé čas jeho setrvání na obrazovce. Ideální je jedna až dvě vteřiny.

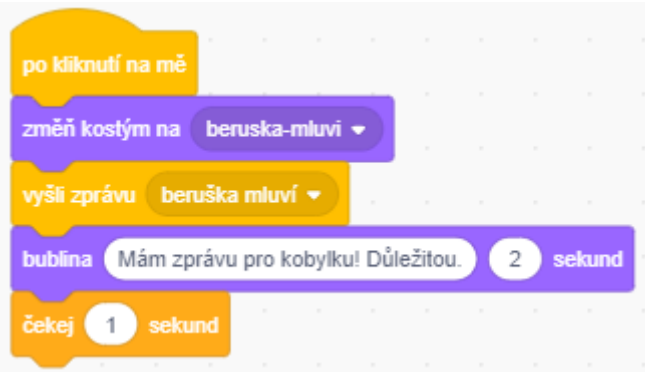


Obrázek 78 Řešení 2.bodu úlohy

4. Počká jednu sekundu.

Řešení 4.bodu úlohy

V skupině bloků **Ovládání** vybereme **Čekej x sekund**.



Obrázek 79 Řešení 4.bodu úlohy

Metodická poznámka

Nyní se vrátíme ke konceptu zpráv. V kartě **Události** vybereme **Po obdržení zprávy**. Do prázdného políčka vybereme tu zprávu, na kterou bude postava reagovat. Toto ukážeme i na tabuli. Necháme žáky samostatně dodělat všechny úlohy až do bodu 11.

11. Posune se k berušce.

Řešení 11.bodu úlohy

Z **Pohybu** vybereme **Klouzej x sekund na beruška**. V nabídce jsou všechny dostupné postavy i kurzor.

Obrázek 80 Řešení úlohy u kobylky

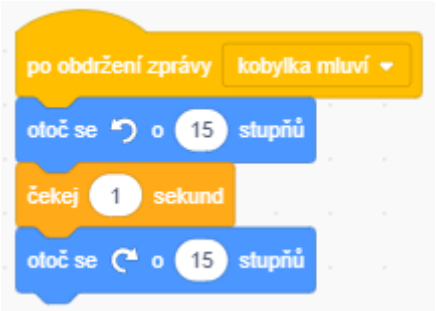
12. Otočí se lehce nalevo. Zkuste si kolik stupňů vám přijde ideálních.

Metodická poznámka

Vybereme si nějaké stupně, o které budeme postavu motýlka otáčet. Když ho budeme natáčet ke kobylce, otočí se o vybraný počet stupňů nalevo a poté zpátky do startovní polohy. Musíme poté tedy motýlka otočit o stejný počet stupňů na opačnou stranu.

Řešení 12.bodu úlohy

Pod blok s přijmutím zprávy o promluvy kobylky umístíme z skupiny bloků **Pohyb** **Otoč se o x stupňů doleva**. Pod něj umístíme nám již známý blok s počkáním a poté následuje blok **Otoč se o x stupňů doprava**.



Obrázek 81 Řešení otáčení u motýla

Závěr

Žáci by nyní měli chápat koncept zpráv v programování a ovládat některé z příkazů týkající se vzhledu.

Druhá lekce – Hmyzí říše instrukce pro žáky

1. Počáteční bloky s vlaječkou ponechejte v programu a nemažte je.
2. V programu se nacházejí tři postavy hmyzu. Naprogramujte je podle tabulky. Ještě než začnete pracovat na řešení, vložte si do programu bloky z událostí, se kterými budeme pracovat.

Beruška (po kliknutí na ni)	3. Změní kostým na beruska-mluvi. 4. Vyšle zprávu „Beruška mluví“. 5. Řekne, že má zprávu pro kobylku. 6. Počká jednu sekundu. 7. Změní kostým na beruska-klid.
Kobylka (po kliknutí na ni)	8. Vyšle zprávu „kobylka mluví“. 9. Představí se.
Kobylka (po obdržení zprávy „beruška mluví“)	10. Vyšle zprávu „kobylka mluví“. 11. Řekne, že už běží. 12. Počká jednu sekundu. 13. Posune se k berušce.
Motýl (po obdržení zprávy „kobylka mluví“)	14. Otočí se lehce nalevo. Zkuste si kolik stupňů vám přijde ideálních. 15. Počká jednu sekundu. 16. Otočí se zpátky do počáteční polohy.
Motýl (po obdržení zprávy „beruška mluví“)	To samé jako v situaci kdy mluví kobylka, ale v prvním kroku se otáčí doprava.

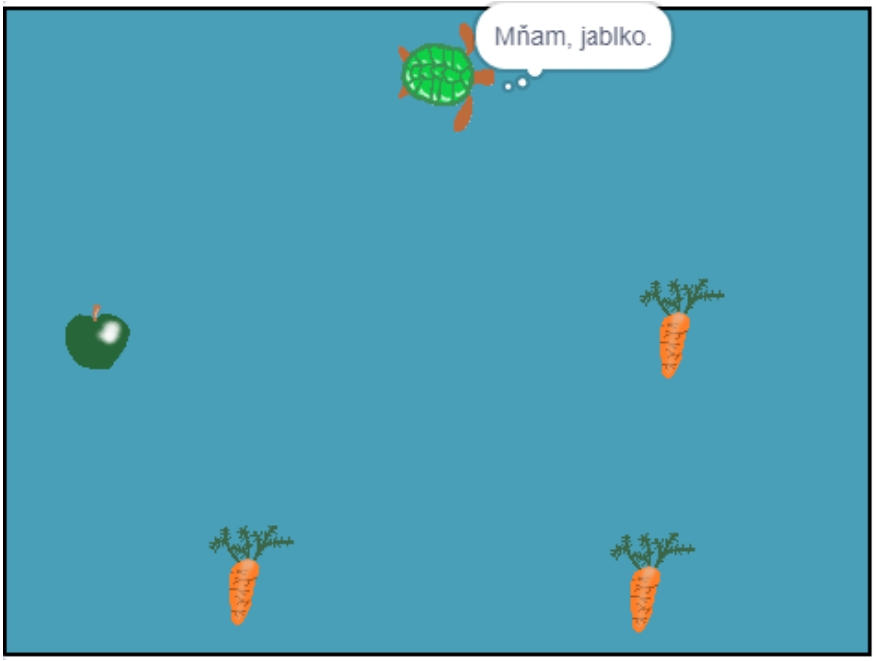
Úkol navíc: Zkuste si přidat čtvrtou postavu do scénáře. Kliknutím na něj se představí a poté se posune k jiné z postav ve scénáři.

Použité skupiny bloků: **Pohyb**, **Události**, **Vzhled**, **Ovládání**

Třetí lekce – ŽelvaV2 učitelský návod

Popis scénáře

V tomto cvičení si přidáme k již rozpohybované želvě interakci s prostředím – ovoce a zeleninu, které se dotkne, sní.



Obrázek 82 Scénář ŽelvaV2 s ukázkou nově naprogramované funkčnosti

Použité programátorské koncepty

Příkaz

Zprávy

Nově představené bloky z programu Scratch

Vzhled – Myšlenka po dobu x sekund

Vzhled – Ukaž se

Vzhled – Skryj se

Vnímání – Dotýkáš se

Události – Po kliknutí na zelenou vlaječku

Ovládání – Čekej dokud nenastane

Využité bloky

Bloky týkající se zpráv, viditelnosti postavy, blok týkající se smyčky „while“, bloky vnímání

Použitý projekt

03 – ŽelvaV2 – Teacher.sb3	soubor pro učitele
03 – ŽelvaV2 – Student.sb3	soubor pro žáky

Praktická část

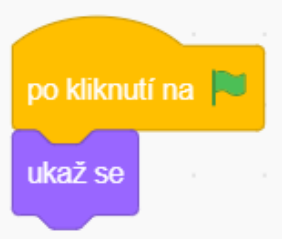
Metodická poznámka

Nyní můžeme připomenout blok **Po kliknutí na zelenou vlaječku**. Ten se spustí poté, co zachytíme specifikovanou událost, v tomto případě zapnutí scénáře. Předtím jsme o tomto bloku pouze mluvili, nyní ho i použijeme.

Ukáže se.

Řešení 2.bodu úlohy

Ze **Vzhledu** vybereme **Ukaž se** a přetáhneme na plochu pod blok s vlaječkou.



Obrázek 83 Řešení 1.bodu úlohy

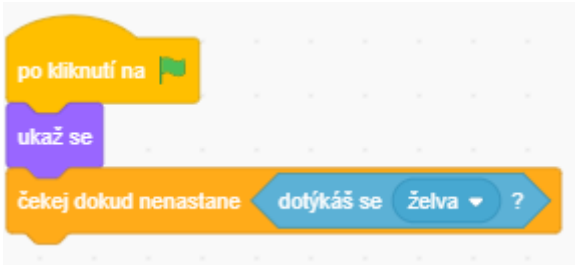
Čeká, dokud se ho želva nedotkne.

Metodická poznámka

Toto je první seznámení žáků s blokem **Čekej dokud nenastane** ze skupiny příkazů **Ovládání**. Jak již název napovídá, tento blok čeká, dokud nenastane programátorem zvolená událost. Pokud tato událost nastane, budou se konat příkazy pod tímto blokem.

Řešení 2.bodu úlohy

Nyní budeme pracovat se dvěmi novými ještě neprobranými skupinami bloků – **Události**, **Ovládání** a minimálně i **Vnímání**. To si opět lépe probereme později. Z **Ovládání** zvolíme výše zmíněný **Čekej dokud nenastane**. Do kosočtverce, kam zadáváme podmínku, přetáhneme z **Vnímání** blok **Dotýkáš se** a do prázdného místa vybereme název pro postavu želvy.



Obrázek 84 Řešení 2.bodu úlohy

Metodická poznámka

Nyní můžeme využít techniky read-write-share. Na tabuli můžeme napsat blok Čekej dokud nenastane a vymyslet vzorovou situaci, například kdybychom měli hru typu Space invaders. Můžeme krátce vysvětlit jak tato hra funguje – hráč, který ovládá vesmírnou loď střílí po návštěvnících z kosmu a snaží se je zlikvidovat. Máme tři životy, pokud se nás mimozemštané třikrát dotknou, hra končí. Pokud trefíme určité objekty ve hře, přibyde nám počet lodí z počáteční jedné na tři, pokud se zbavíme určitého počtu mimozemských návštěvníků, tak dostaneme lepší loď a podobně.

Nyní můžeme nabádat žáky, ať individuálně přemýšlí nad tím, jak by použili blok Čekej dokud nenastane právě v případě hry tohoto typu. Jako příklad můžeme napsat na tabuli k již existujícímu textu: trefíš mimozemštana. Následující akce, kterou tento blok zachytí bude: počet bodů se zvýší o jedna.

Žáci si po chvilce zamyšlení se nad touto situací svoje řešení zapíší a poté porovnají se sousedem. Jakožto skupinka si dvojice vybere mluvčího a až učitel řekne, ten třídě představí své řešení. Učitel si mluvčí vyslechne a až uplně na konci poví, jaká jsou špatná a jaká správná řešení.

Žáci jsou v tomto věku obeznámeni s videohrami a tak dle mě není problém si podobnou situaci, ke které vymýšlejí použití bloku Čekej dokud nenastane, představit. Pouze stačí přistoupit k ní z programátorského úhlu pohledu.

Pokud se ho dotkne, tak se skryje.

Řešení 2.bodu úlohy

Z **Vzhledu** vybeme **Skryj se** a umístíme tento blok pod cyklus **Čekej dokud nenastane**. Tímto se podmínka v cyklusu splnila.

Obrázek 85 Kompletní řešení 2.bodu úlohy

Metodická poznámka

Pokud žáci netuší, jak vyřešit 3. a 4.bod úlohy, můžeme jim připomenout koncept představený v minulé hodině – zprávy. Pomocí nich může jídlo v případě, že se ho želva dotkne poslat zprávu. Tu následovně želva přijme a pronese informaci o tom, o jaký druh jídla šlo.

Závěr

Žáci by nyní měli umět naprogramovat mizení předmětů a použít blok **Čekej dokud nenastane**.

Třetí lekce – ŽelvaV2 instrukce pro žáky

1. Počáteční bloky s vlaječkou ponechejte ve scénáři a nemažte je.
2. Nyní si naprogramujeme krmení želvy – poté co se dotkne jídla, tak jablko či mrkve zmizí. Následující bloky přidáme k postavám, které mají kostým jídla.

Jídlo (po spuštění)	• Ukáže se. • Čeká, dokud se ho želva nedotkne. • Pokud se ho dotkne, tak se skryje.

3. Pokud se želva dotkne jablka, řekne: Mňam jablko.
4. Pokud se želva dotkne mrkve, řekne: Mňam mrkev.

Použité skupiny bloků: Ovládání, Vzhled, Události, Vnímání

Třetí lekce – Auto učitelský návod

Popis scénáře

V tomto cvičení si zopakujeme znalosti z minulých lekcí týkající se zpráv a pohybu postavy. Využijeme u něho práci ve dvojicích.

Obrázek 86 Vzhled scénáře Auto

Použité programátorské koncepty

Příkaz

Zprávy

Nově představené bloky z programu Scratch

Vzhled – Další kostým

Vzhled – Další pozadí

Využité bloky

Bloky týkající se zpráv, viditelnosti postavy, blok týkající se smyčky „while“, bloky vnímání, Bloky týkající se pohybu

Použitý projekt

03 – Auto – Teacher.sb3	soubor pro učitele
03 – Auto – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Právě protože v tomto úkolu jde primárně o zopakování již naučených znalostí, můžeme u něho využít práci ve dvojicích. Žáky můžeme rozdělit například podle toho jak již sedí ve třídě do párů (případně podle jakéhokoliv způsobu, který uznáme za vhodný). V instrukcích máme dvě skupiny úkolů, rozdělené aby každý člen dvojice programoval část cvičení.

Po kliknutí na tlačítko s nápisem Změň auto se přepne postavu auta na další kostým.

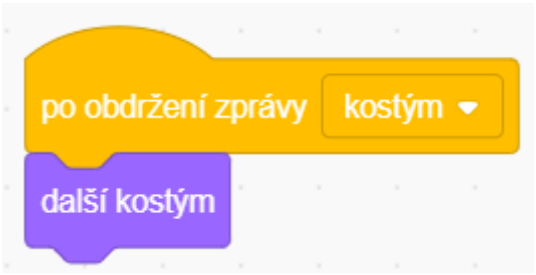
Metodická poznámka

Tento úkol, ačkoliv jde o něco nového, je formulován tak, že by mělo být jasné, kde hledaný blok nalézt a jak se bude jmenovat. Necháme žáky pracovat chvíli ve dvojicích, obejdeme si je a zjistíme, jak na tom jsou. Pokud po čase potřebném pro dokončení úkolu většina třídy stále neví, ukážeme postup na tabuli.

Všem dvojicím řekněme, že nastane-li situace, kdy jeden z dvojice ví jak toto naprogramovat a druhý ne, mají si zkusit princip bloku mezi sebou vysvětlit. V tomto cvičení celkově nefungujeme jako koordinátor výuky, spíše jako pomocník, který zná řešení a může žákům pomoci, pokud je potřeba. Oproti ostatním cvičením, kdy režii hodiny máme na starosti my jakožto učitelé, nyní si ji z velké části vedou dvojice samy.

Řešení 2. bodu úlohy

Pod blok **Po kliknutí na mě** z **Událostí** umístíme blok s odesláním zprávy o změně kostýmu. Tato událost se děje v rámci postavy tlačítka. Poté, co zprávu postava auta přijme, pod stejnojmenný blok umístíme **Další kostým**, tentokrát z skupiny bloků **Vzhled**.



Obrázek 87 Řešení 2.bodu úlohy u auta

Přidej si dvě další pozadí z nabídky Scratche.

Metodická poznámka

Nyní se můžeme zeptat, zda někdo již vyzkoušel, jak přidat pozadí. Z mé zkušenosti na to žáci přišli celkem rychle i sami od sebe, ale pokud se tak nestalo ukážeme přidání pozadí na tabuli.

Řešení 2.bodu úlohy

Kliknutím na menu scény se nám nad skupinami bloků zpřístupnila karta Pozadí. Pokud ji klikem vybereme, zobrazí se nám všechny pozadí, jaké zatím scénář obsahuje. Nové pozadí vyberem najetím myší na kulaté tlačíko a kliknutím na Obrázek lupy. Tento úkon nám zpřístupnil již hotová pozadí, která můžeme v programu Scratch použít.

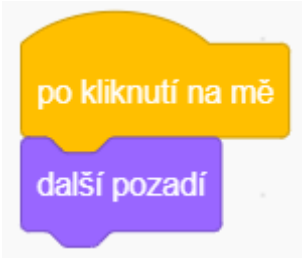
Po kliknutí na tlačítko s nápisem Změň pozadí se přepne pozadí na další.

Metodická poznámka

Zde platí to samé, co u minulých zadání v této hodině - opět se zeptáme zda někdo potřebuje pomoci. Pokud se tací ve třídě najdou, ukážeme postup na tabuli.

Řešení 2.bodu úlohy

Ze skupiny bloků **Události** vybereme blok **Po kliknutí na mě**. Pod něj umístíme **Další pozadí**, blok, který najdeme ve skupině bloků **Vzhled**.



Obrázek 88 Řešení 2.bodu úlohy u tlačítka

Závěr

Žáci si zopakovali výklad z minulých hodin a do jejich programovacího repertoáru přibyly další příkazy týkající se změny vzhledu.

Třetí lekce – Auto instrukce pro žáky

1. Počáteční bloky s vlaječkou ponechejte v programu a nemažte je.
2. Postupujte podle tabulky.

První člen dvojice	• Naprogramuj otáčení auta nalevo a napravo podle zmáčknuté šipky. • Naprogramuj mizení hvězdy poté, co se k ní přiblíží auto, u tří prvních hvězd.
Druhý člen dvojice	• Naprogramuj pohyb auta vpřed a vzad podle zmáčknuté šipky. • Naprogramuj mizení hvězdy poté, co se k ní přiblíží auto u zbývajících hvězd.

3. Po kliknutí na knoflík s nápisem Změň auto se přepne avatar auta na další kostým. Na tomto úkolu pracujte společně. K vyřešení vám může být užitečné vzpomenout si na zprávy z minulé hodiny.

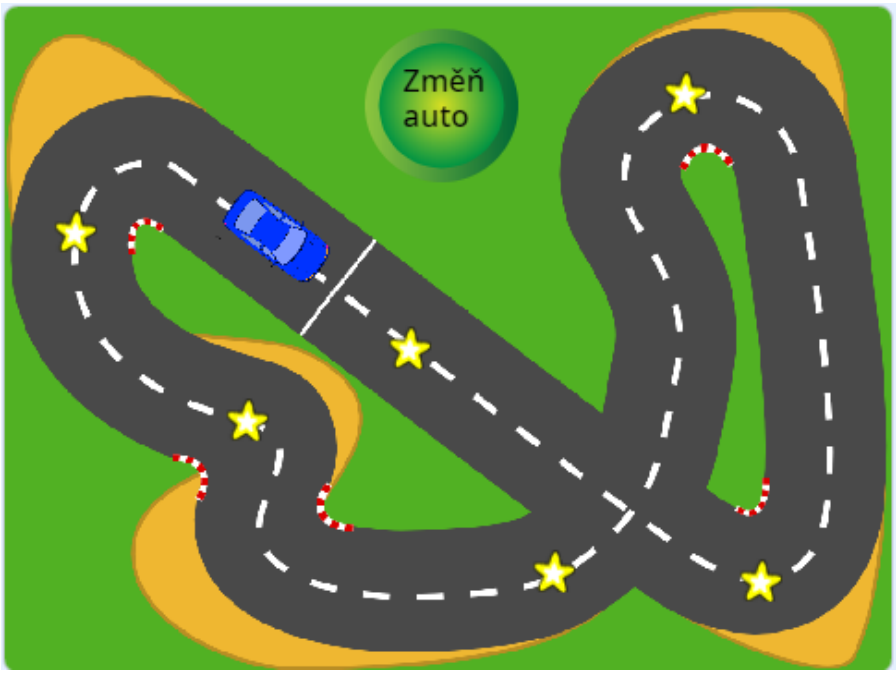
Úkol navíc: Přidejte na obrazovku další knoflík, který tentokrát bude klikem přepínat pozadí. Pozadí budou tři – původní a dvě pozadí, každé vytvořené jedním ze členů skupiny.

Použité oblasti Scratche: Ovládání, Vzhled, Události, Vnímání

Čtvrtá lekce – AutoV2 učitelský návod

Popis scénáře

V tomto cvičení si přidáme zvuk ke cvičení z minulé hodiny.



Obrázek 89 Vzhled programu AutoV2

Použité programátorské koncepty

Příkaz

Zprávy

Cyklus - nové

Nově představené příkazy z programu Scratch

Zvuk – Přehraj zvuk až do konce

Zvuk – Začni hrát zvuk

Využité bloky

Bloky týkající se zvuku, pohybu, bloky týkající se cyklů

Použitý projekt

04 – AutoV2 – Teacher.sb3	soubor pro učitele
04 – AutoV2 – Student.sb3	soubor pro žáky

Praktická část

1. Pomocí cyklu Opakuj stále naprogramuj pohyb auta dopředu. Auto se začne pohybovat, jakmile zmáčkneme šipku dopředu.

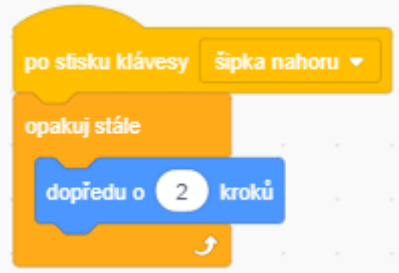
Metodická poznámka

V souboru 04 – AutoV2 – Student.sb3 se nachází scénář velmi podobný minulému. Dnešní úkol se bude trochu lišit, a to v ovládání pohybu. Jakmile zmáčkneme šipku dopředu, auto se bude chovat jako by mělo cihlu na plynu a pojede dopředu, dokud scénář nerestartujeme či nestopneme.

K naprogramování tohoto scénáře bude nutné žákům vysvětlit co je to cyklus. Jak již název napovídá, cyklus je sekvence příkazů, která se opakuje tolikrát, kolik specifikuje programátor. V případě Scratche můžeme cyklus buď opakovat donekonečna, to je blok **Opakuj stále** ze skupiny bloků **Ovládání** či opakovat pouze zadané množství opakování. K tomu slouží blok **Opakuj 10 krát** ze stejné skupiny příkazů, kdy desítku můžeme přepsat na požadované množství opakování.

Řešení 1.bodu úlohy

Ze skupiny bloků **Události** vybereme blok **Po stisku klávesy** a umístíme ho do prostoru scénáře. Z nabídky kláves vybereme šipku nahoru. Pod něj umístíme blok **Opakuj stále** ze skupiny bloků **Ovládání**. Dovnitř tohoto bloku umístíme nám již známý blok **Dopředu o 2 kroků** ze skupiny bloků **Pohyb**.



Obrázek 90 Řešení 1.bodu úlohy

2. Pokud klikneme na knoflík s nápisem Změň auto, začne hrát zvuk s názvem auto.

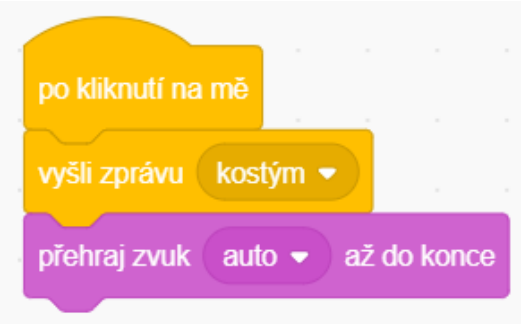
Metodická poznámka

Před tímto cvičením je dobré v minulé hodině zmínit, aby si žáci přinesli vlastní sluchátka (pokud jimi škola nedisponuje).

Klikneme na jednu z postav a upozorníme žáky na kartu Zvuky. Tato karta se nachází nad skupinami bloků vedle karty Kostýmy a má jako logo vypovídající Obrázek reproduktoru. Nabádáme je k proklikání jednotlivých postav a poslechu zvuků, které se u nich nacházejí. Poté ukážeme, jak sestrojit druhý bod úlohy.

Řešení 2.bodu úlohy

Ze skupiny bloků **Zvuk** vybereme **Přehraj zvuk až do konce** a umístíme ho pod existující bloky. Do prázdného místa vybereme zvuk auto.

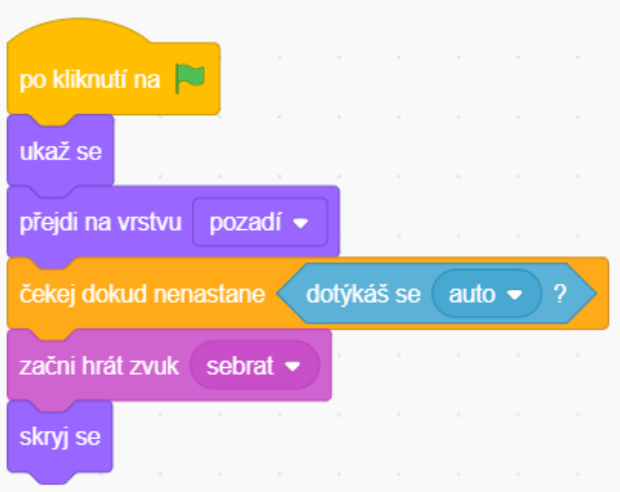


Obrázek 91 Řešení 2.bodu úlohy

3. Pokud se auto dotkne některé z hvězd, začne hrát zvuk s názvem sebrat.

Řešení 3.bodu úlohy

Mezi blok **Čekej dokud nenastane** a **Skryj se** umístíme **Začni hrát zvuk** ze skupiny bloků **Zvuk.** Do prázdného místa vybereme zvuk sebrat.



Obrázek 92 Řešení 3.bodu úlohy

Závěr

Žáci by nyní měli rozumět základním příkazům týkajících se zvuku v programu Scratch a umět použít blok **Opakuj stále**.

Čtvrtá lekce – AutoV2 instrukce pro žáky

1. Pomocí cyklu Opakuj stále naprogramuj pohyb auta dopředu. Auto se začne pohybovat, jakmile zmáčkneme šipku dopředu.
2. Pokud klikneme na knoflík s nápisem Změň auto, přehraje se zvuk „auto“ až do konce.
3. Pokud se auto dotkne některé z hvězd, začne hrát zvuk s názvem sebrat.

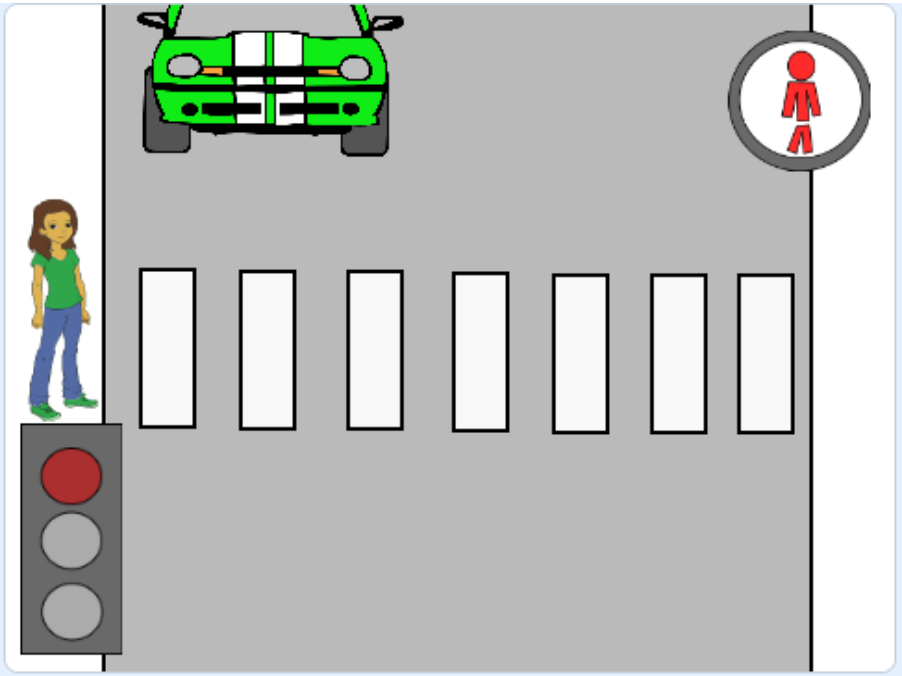
Úkol navíc: Zkuste si po kliknutí na knoflík Změň auto odebrat **Přehraj zvuk až do konce** a vyměňte ho za **Začni hrát zvuk**. Jaký je mezi nimi rozdíl?

Použité skupiny bloků: Zvuk, Ovládání, Události, Pohyb

Čtvrtá lekce – Dopravní značky učitelský návod

Popis scénáře

V tomto cvičení si představíme programátorský koncept podmínky.



Obrázek 93 Vzhled scénáře Dopravní značky

Použité programátorské koncepty

Příkaz

Zprávy

Cyklus

Podmínky - nové

Nově představené příkazy z programu Scratch

Ovládání - Když..tak

Vnímání – Klávesa stisknuta?

Pohyb – Klouzej x sekund na x: y:

Využité bloky

Bloky týkající se cyklů a podmínek, bloky týkající se zpráv a kostýmů

Použitý projekt

04 – DopravniZnacky – Teacher.sb3	soubor pro učitele
04 – DopravniZnacky – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

V tomto cvičení se poprvé potkáváme v instrukcích pro žáky s body zadání s tečkou. Konkrétně v tomto příkladu jsou to body zadání tři a čtyři, které se skládají z několika částí sestrojení. Tento druh číslování jsem zvolila aby bylo jasné, že bodem 3.1 se začne sekvence bloků a další bloky budeme dodávat postupně pod bloky, které jsme přidali v bodě 3.1. Podobně učiníme i u čtvrtého bodu úlohy a dalších úloh, které budou zadány tímto způsobem.

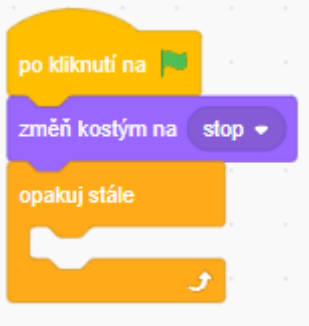
To však není jediná novinka s kterou se potkáváme - také tu poprvé použijeme podmínky. Podmínky jsou základní stavební kameny programování a fungují podobně jako podmínky v češtinářském významu. Na začátku podmínky je kouzelné slovíčko když či pokud, poté následuje zápis děje, který tímto programátorským konceptem zachytíme. V těle podmínky, v případě Scratche podmínkového bloku **Když..tak** ze skupiny bloků **Ovládání**, se nachází příkazy, které se provedou, pokud zachytávaný děj proběhne. Příkladem děje, který můžeme v programu Scratch zachytávat je například stisknutí určité klávesy.

Tělem podmínky se v programu Scratch rozumí prázdné místo uvnitř bloků jako **Opakuj stále**, **Když…tak** a některých dalších bloků ze skupiny bloků **Ovládání**.

3.1 U značky pro chodce se donekonečna bude kontrolovat, zda jsou níže zmíněné klávesy stisknuté. Naprogramujte:

Řešení 3.bodu úlohy

Jelikož budeme stisknutí kláves donekonečna kontrolovat, je nutné pod bloky, které jsme přidali v 1.bode řešení úlohy umístit blok **Opakuj stále** ze skupiny bloků **Ovládání.**. Do něj budeme umísťovat podmínky z bodu úlohy 3.2 a 3.3. Toto je nám již známý koncept cyklu, nyní je nově představen ve spojení s podmínkami.



Obrázek 94 Řešení bodu úlohy 3.1

3.2 Pokud stiskneme klávesu „j“ jako jdi, objeví se na značce zelený panáček.

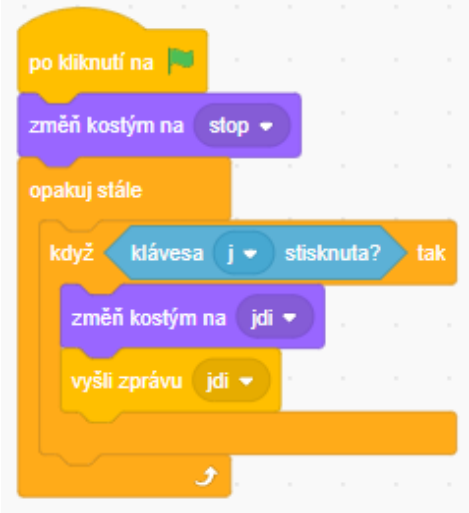
Metodická poznámka

Nyní můžeme na tabuli ukázat řešení podmínky v bodě úlohy 3.2, následující podmínku v bodě úlohy 3.3 a celý bod úlohy čtyři mohou žáci již naprogramovat sami – řešení mají obdobné.

Řešení bodu úlohy 3.2

Ze skupiny bloků **Ovládání** vybereme blok **Když..tak** a umístíme ho do prostoru scénáře. Do prázdného místa ve tvaru kosočtverce umístíme naši podmínku – v tomto případě blok **Klávesa stisknuta** ze skupiny bloků **Vnímání**. Z rozbalovacího menu bloku **Klávesa stisknuta** vybereme „j“.

Do těla podmínky vložíme bloky, které odpovídají zadání.

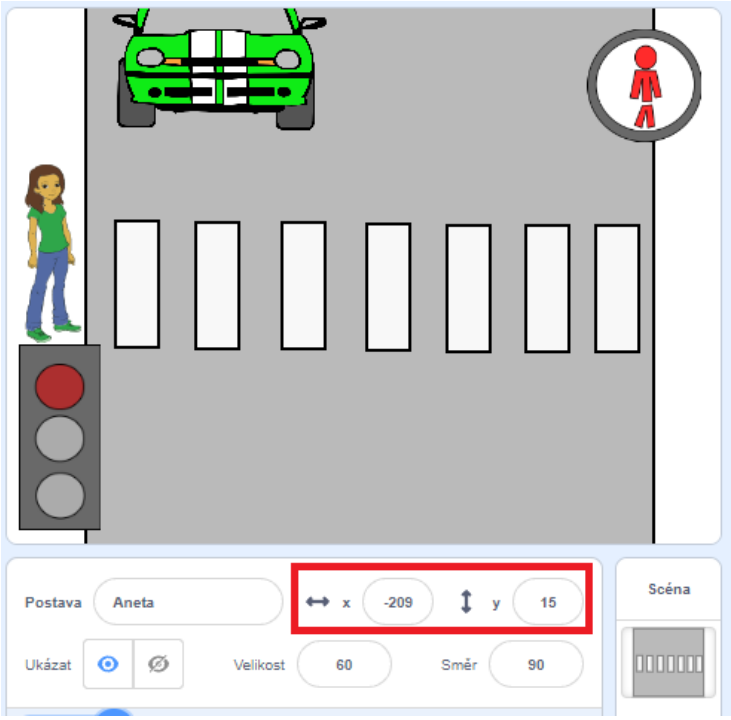


Obrázek 95 Řešení .bodu úlohy 3.2

5. Když se změní značka pro chodce na zeleného panáčka, chodec přejde silnici.

Metodická poznámka

V pátém bodu úlohy se podíváme na způsob, jakým program Scratch říká, kde se právě nachází námi postava. Kliknutím vybereme postavu, u které tuto informaci chceme zjistit. Pod scénou se nachází dvě řady ikonek a informací. Souřadnice polohy postavy najdeme na prvním řádku na levé straně, u nápisu x a y. Postavy ve scéně se pohybují po kartézské souřadnici.



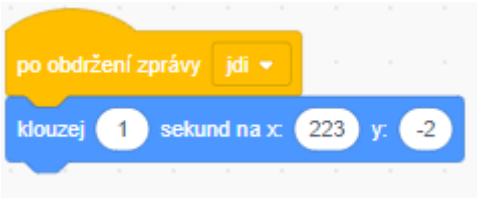
Obrázek 96 Souřadnice polohy postavy

Scratch a v něm obsažená skupina bloků **Pohyb** obsahuje několik bloků, kde se pohyb v sytému kartézské souřadnice využívá a my si ukážeme jeden z nich.

Řešení 5.bodu úlohy

Nyní budeme pracovat s postavou chodce. Začneme přetažením bloku **Po přijetí zprávy**, který oznamuje změnu značky pro chodce na zeleného panáčka do prostoru scénáře. Poté pohneme postavou tak, aby „přešla“ silnici. Dále se přepneme do skupiny bloků **Pohyb**. Všechny bloky, které pracují s kartézskými souřadnicemi x a y nám nyní ukazují aktuální polohu postavy.

Do prostoru scénáře, pod blok o přijetí zprávy o zeleném panáčkovi, umístíme blok **Klouzej x sekund na x: y:** ze skupiny bloků **Pohyb**. Pohyb auta se vyřešíme obdobně.



Obrázek 97 Řešení 5.bodu úlohy

Závěr

Žáci by nyní měli rozumět programátorskému konceptu podmínek.

Čtvrtá lekce – Dopravní značky instrukce pro žáky

1. Po kliknutí na zelenou vlaječku u značky pro chodce se zobrazí červený panáček.

2. Po kliknutí na zelenou vlaječku u semaforu se zobrazí kostým s červenou barvou.

3.1 U značky pro chodce se donekonečna bude kontrolovat, zda jsou níže zmíněné klávesy stisknuté. Naprogramujte:

3.2 Pokud stiskneme klávesu „j“ jako jdi, objeví se na značce zelený panáček.

3.3 Pokud stiskneme klávesu „s“ jako stop, objeví se na značce červený panáček.

4.1 U semaforu se bude také donekonečna kontrolovat, zda jsou stisknuty klávesy zmíněné v následujících instrukcích bodu čtyři. Naprogramujte podobně jako u předchozího bodu úlohy:

4.2 Pokud stiskneme klávesu „r“ jako „red“, na semaforu se objeví červená.

4.3 Pokud stiskneme klávesu „y“ jako „yellow“, na semaforu se objeví červená.

4.4 Pokud stiskneme klávesu „g“ jako „green“, na semaforu se objeví zelená.

5. Když se změní značka pro chodce na zeleného panáčka, chodec přejde silnici.

6. Když se na semaforu objeví zelená, auto přejede silnici.

Nápověda: Pro vyřešení bodu 5 a 6 si vzpomeňte na zprávy.

Použité skupiny bloků: Vzhled, Vnímání, Události, Ovládání

Pátá lekce – Hudební nástroje učitelský návod

Popis scénáře

V tomto cvičení si zopakujeme pohyb avataru a vysvětlíme koncept podmínky v programování.

Obrázek 98 Vzhled scénáře Hudební nástroje

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky - nové

Nově představené příkazy z programu Scratch

Operátory – A

Zvuk – Zastav všechny zvuky

Vzhled – Přejdi na vrstvu

Využité bloky

Po stisku klávesy, přejdi na vrstvu, dotýkáš se postavy, bloky týkající se pohybu, bloky týkající se zvuku, nastav směr, nastav otáčení, klávesa stisknuta, podmínkové bloky, bloky týkající se operátorů, blok s nekonečným cyklem

Použitý projekt

05 – Hudební nástroje – Teacher.sb3	soubor pro učitele
05 – Hudební nástroje – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Při plnění bodu číslo dva můžeme žáky upozornit na kostýmy postavy, které již tvoří iluzi pohybu. Proto je zde přepínání pouze pomocí další kostým.

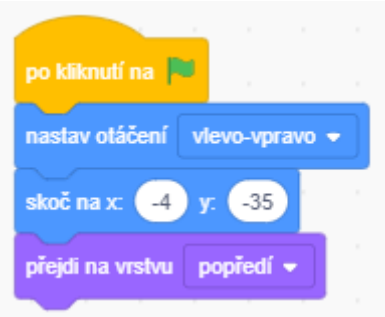
2. Jaká postava ve scénáři bude v pozadí a popředí? Krátce popřemýšlej a poté naprogramuj pod blok signalizující zapnutí scénáře u všech postav.

Metodická poznámka

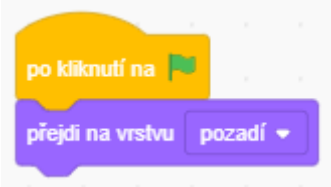
V tomto cvičení, narozdíl od předchozích, bude záležet na tom, zda se postavy nachází v pozadí či popředí. Hlavní hrdinkou tohoto scénáře je Holly, takže ji umístíme pomocí bloků, které jsou ve Scratchi dostupné, do popředí. Ostatní postavy, hudební nástroje, se budou nacházet v pozadí.

Řešení 2.bodu úlohy

Pod blok **Po kliknutí na zelenou vlaječku** pod bloky, které se zde již nachází, umístíme blok **Přejdi na vrstvu** ze skupiny bloků **Vzhled**. Z menu u bloku **Přejdi na vrstvu** vybereme v případě nástrojů pozadí. U postavy s názvem Holly vybereme z menu popředí.



Obrázek 99 Řešení 2.bodu úlohy u postavy Holly



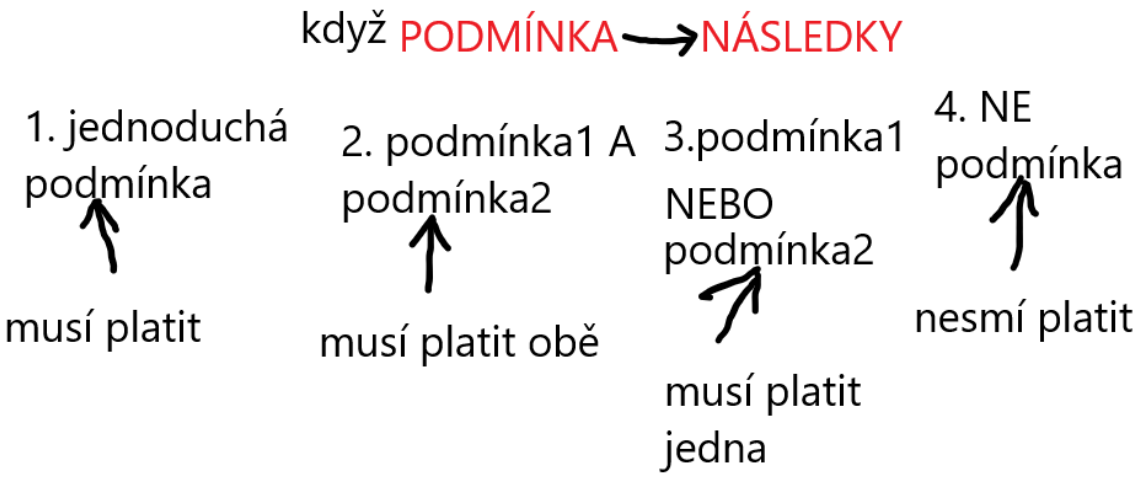
Obrázek 100 Řešení 2.bodu úlohy u hudebních nástrojů

Když se postava...	**Tak udělej..**
dotýká piana **a** klávesa mezerník stisknuta	zahraj zvuk piano
dotýká se kytary **a** klávesa mezerník stisknuta	zahraj zvuk kytara

Metodická poznámka

Ve třetí části cvičení si připomeneme podmínku v češtinářské podobě. Typicky se skládá z výroku, který má na začátku spojku pokud nebo když. Po ní následuje situace, která musí nastat, aby se stalo pokračování věty – to často začíná spojkou tak.

Podmínka v programování je velmi podobná. Příklad podmínky v programování může být například: pokud je zmáčknutá klávesa k, přepni pozadí na další. Kromě případu s jednou podmínkou můžeme mít podmínek více a to v různých vztazích. Opět se zde jedná o spojky. Lépe se to vysvětlí obrázkem a následovnou prací žáků, kde si promyslí své nápady, napíší si je a poté proberou se sousedem.



Obrázek 101 Obrázek popisující podmínky

Nyní by si žáci měli napsat dva příklady ke každé z podmínek, jeden z reálného života a druhý, který by využili v programu Scratch nebo v obecném pseudo - programovacím jazyku. Poté co budou s psaním hotovi a učitel tak určí, proberou své doměnky se sousedem.Učitel poté vybere, pokud možno náhodně, jednoho ze dvojice a ten za oba její členy řekne nejpovedenější podmínku z reálného života a programování.

Řešení 3.bodu úlohy

Ještě než umístíme samotné podmínky, je nutné je obalit v cyklu, který bude platit po celý běh scénáře. Tento blok se jmenuje **Opakuj stále** a najdeme ho v **Ovládání**. Více o cyklech si řekneme v dalších hodinách.

Toto řešení je lepší zobrazit obrázkem, který si blíže popíšeme.

Obrázek 102 Řešení podmínky

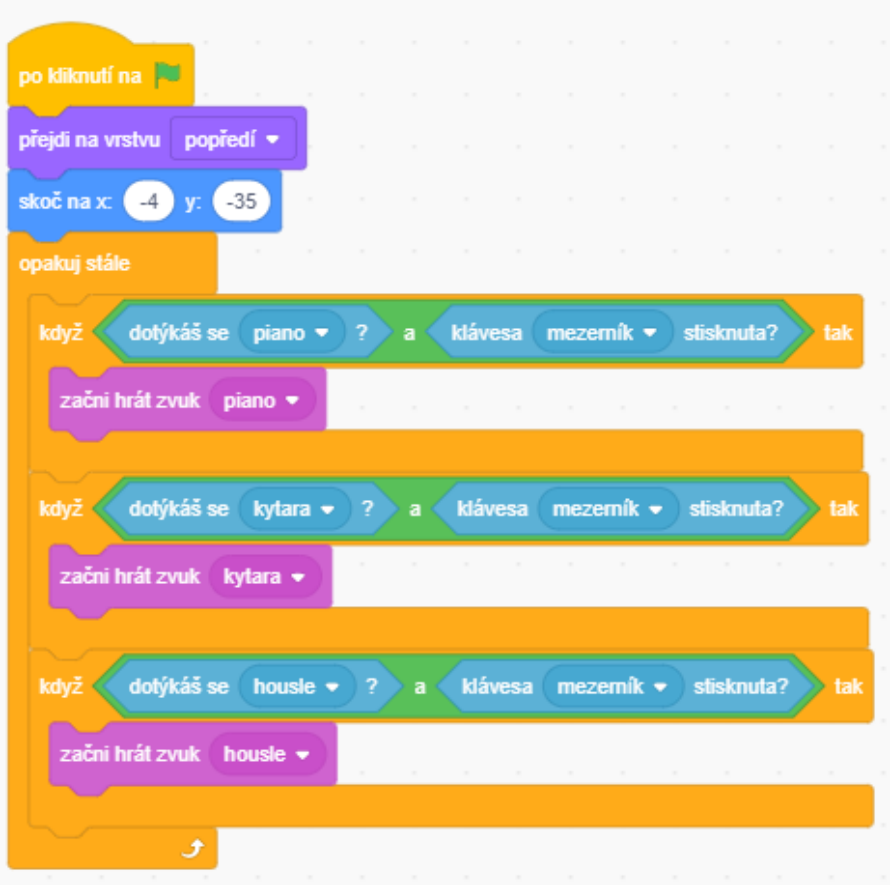
A) Jako první do cyklu **Opakuj stále** umístíme ze stejné skupiny bloků, **Ovládání**, podmínkový blok **Když..tak**.

B) Pracujeme se dvěma podmínkami, takže nám místo nebude stačit. Z **Operátorů** vybereme blok se dvěma prázdnými místy a spojkou „a“ mezi nimi. Tyto podmínky musí platit obě, aby proběhly příkazy uvnitř.

C) Přepneme se do skupiny bloků **Vnímání** a vybereme **Klávesa stisknuta**. Tu umístíme do libovolného prázdného místa, na pořadí nezáleží. Klávesu, jejíž stisknutí kontrolujeme, můžeme vybrat v nabídce. Do druhého dáme **Dotýkáš se** a vybereme název pro postavu piana.

D) Vevnitř podmínky se nachází blok, který se stane pouze pokud jsou obě výše zmíněné podmínky splněny.

Nyní se podíváme na kompletní řešení cyklu s podmínkami.



Obrázek 103 Kompletní řešení cyklu s podmínkami

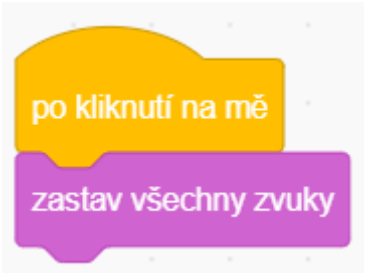
Metodická poznámka

Je dobré aspoň v krátkosti vysvětlit, že vše, co se nachází uvnitř tohoto cyklu bude opakováno po dobu běhu celého scénáře – to znamená, že se bude od zapnutí po vypnutí scénáře kontrolovat zda se podmínky splnily.Ke konci ukazování můžeme opět připomenout, že se zvuk piana přehraje pouze tehdy, pokud obě podmínky platí. Ukažte první podmínku a zbytek nechejte na žácích.

4. Po kliknutí na knoflík se zastaví všechny zvuky.

Řešení 4.bodu úlohy

Ze **Zvuku** vybereme blok s názvem **Zastav všechny zvuky** s názvem nástroje, kterého se podmínka týká.



Obrázek 104 Řešení 4.bodu úlohy

Závěr

Žáci by nyní měli rozumět základům podmínek a vztahům mezi více podmínkami, které diktují jak se bude scénář chovat.

Pátá lekce – Hudební nástroje instrukce pro žáky

1. Naprogramuj podle tabulky pohyb postavy.

Po stisku klávesy šipka vlevo	• Popojde vlevo • Přepne se na další kostým.
Po stisku klávesy šipka vpravo	• Popojde vpravo. • Přepne se na další kostým

2. Jaká postava v programu bude v pozadí a popředí? Krátce popřemýšlej a poté naprogramuj pod blok signalizující zapnutí programu u všech postav.

3. Naprogramuj podmínky podle následující tabulky.

Když se postava...	**Tak udělej..**
dotýká piana **a** klávesa mezerník stisknuta	zahraj zvuk piano
dotýká se kytary **a** klávesa mezerník stisknuta	zahraj zvuk kytara

Úkol navíc: Ve scénáři zbývá ještě jeden hudební nástroj. Naprogramujte ho podobně jako předcházející.

4. Po kliknutí na knoflík se zastaví všechny zvuky.

Použité skupiny bloků: **Ovládání**, **Vzhled**, **Události**, **Vnímání**, **Zvuk**, **Pohyb**, **Operátory**

Pátá lekce – Hudební nástrojeV2 učitelský návod

Popis scénáře

Nyní si ukážeme, že i neživé předměty mohou vnímat a vskutku nezáleží, co postava zobrazuje.

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky

Nově představené příkazy z programu Scratch

V tomto cvičení žádné nové příkazy neukazujeme.

Využité bloky

Dotýkáš se postavy, bloky týkající se zvuku, bloky týkající se operátorů, podmínkové bloky, blok s nekonečným cyklem

Použitý projekt

05 – Hudební nástrojeV2 – Teacher.sb3	soubor pro učitele
05 – Hudební nástrojeV2 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Když se na program podíváme lidským pohledem, vnímat bude vždy postava – ta je toho schopná i v reálném životě. V programování ale mohou vnímat i předměty, postavy neživých věcí a toto cvičení je přesně stavěné na pochopení tohoto problému.

Pokud žáci z minulé hodiny zapomněli, můžeme jim řešení prvního úkolu ukázat na tabuli. Opět je nutné podmínku vložit do cyklu **Opakuj stále**, řešení je obdobné jako minule.

Závěr

Žáci nyní chápou, že v programování lze vnímat i skrze předměty, ačkoliv lidská mysl funguje jinak.

Pátá lekce – Hudební nástrojeV2 instrukce pro žáky

Naprogramujte podmínky podobně jako u předchozího úkolu, ale s tím rozdílem, že nyní budou vnímat hudební nástroje. Podmínky tedy budou napsané u nich, z jejich „pohledu“.

1. Piáno: Když se ho dotýká postava a zároveň je stisknut mezerník, začne hrát zvuk piano.

2. Kytara: když se jí dotýká postava a zároveň je stisknut mezerník, začne hrát zvuk kytara.

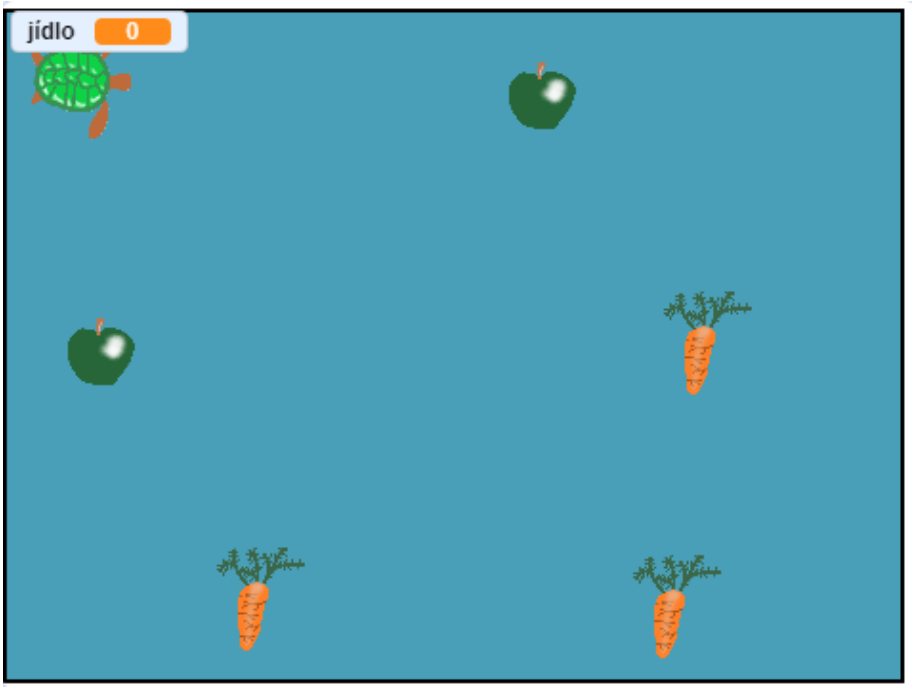
Úkol navíc: Podobně naprogramujte i housle.

Použité skupiny bloků: Ovládání, Vnímání, Zvuk, Operátory

Šestá lekce – ŽelvaV3 učitelský návod

Popis scénáře

V tomto cvičení si vysvětlíme, co je to proměnná a jak s ní manipulovat pomocí příkazů v programu Scratch.



Obrázek 105 Vzhled scénáře Želva V3

Použité programátorské koncepty

Příkaz

Zprávy

Cyklus

Podmínky

Operátory

Proměnné – nové

Nově představené příkazy z programu Scratch

Proměnné – Změň proměnnou o

Proměnné – Nastav proměnnou na

Využité bloky

Bloky pro práci s proměnnou, bloky týkající se zvuku

Použitý projekt

06 – ŽelvaV3 – Teacher.sb3	soubor pro učitele
06 – ŽelvaV3 – Student.sb3	soubor pro žáky

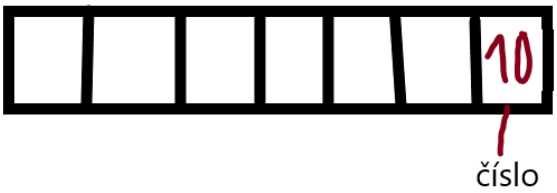
Praktická část

1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny postavy ve scénáři a pojmenujte ji „body“.

Metodická poznámka

Nyní budeme vysvětlovat, co to vlastně proměnná je. Jedná se o místo v paměti, kam se mohou ukládat různé hodnoty – ať už číselné nebo slovní. Tuto hodnotu můžeme přepisovat, navyšovat ho či z ní odečítat pokud se jedná o číslo. Každá proměnná má název, který by se neměl opakovat, aby ve scénáři nebyl zmatek.

Ukažte na tabuli/projektoru celý postup řešení prvního kroku úlohy. Také můžeme pro lepší představu namalovat na tabuli čtvereček znamenající místo v paměti. Paměť má místo pro více proměnných, takže můžeme podobných čtverečků zakreslit více. Poté vymýšlejte jednoduché příkazy typu: Přičti 2 a přepisujte ten samý čtvereček, aby bylo jasné, že jde o jedno a to samé místo v paměti. Můžete se inspirovat obrázkem pod textem, který můžeme namalovat na tabuli nebo do programu malování, podle vybavení třídy.



1) odečti 2 od čísla

2) přičti 10 k číslu

3) zdvojnásob číslo

Obrázek 106 Vysvětlení proměnné na tabuli

Existují dva typy proměnných – globální a lokální. Můžeme se zeptat třídy, co si pod pojmem globální v kontextu programování představuje a udělat podobné cvičení jako v minulé lekci. K sobě si každý napíše co podle něho tyto dva výrazy znamenají a poté se o svůj názor podělí se sousedem. Učitel se po nějaké době zeptá jednoho z dvojice či si dvojice sama vybere mluvčího za skupinu. Teprve až se všechny dvojice vyjádří řekneme správné řešení, do té doby by jsme ho neměli prozradit.

Lokální proměnná platí pouze pro určitou část programu, v případě Scratch to znamená pouze pro jednu z postav. Globální, jak už název napovídá, platí pro celý program a může s ní manipulovat každá z postav.

Pokud je proměnná v programu Scratch zaškrtnutá, její hodnota a název bude zobrazený na scéně. V případě, že proměnná je lokální, zobrazí se před názvem proměnné jméno postavy, ke které patří.

Řešení 1.bodu úlohy

Vyberte skupinu bloků s názvem **Proměnné**. Zde vidíme přednastavenou proměnnou s názvem Moje proměnná. Klikneme na ni pravým tlačítkem myši a smažeme ji.

Novou proměnnou založíme kliknutím na knoflík Vytvoř proměnnou, stále ve skupině bloků **Proměnné**. Vybereme možnost „pro všechny proměnné“ a pojmenujeme ji body. Volbu potvrdíme.

2. U jaké postavy se nachází zvuk? Jak se tento zvuk jmenuje?

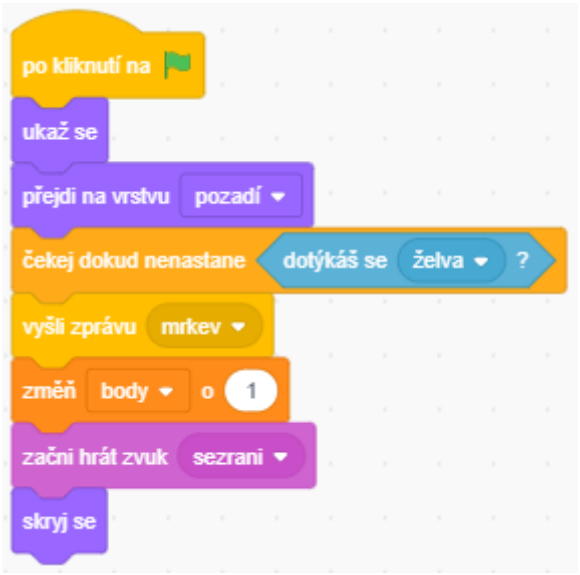
Metodická poznámka

Pokud zjistíme, kde se hledaný zvuk nachází, můžeme s ním v rámci této postavy pracovat. Zvuky vždy patří k určité postavě a jiná postava jej nemůže spustit.

3. Pokud se želva dotkne jablka, zvednou se body o 1. Také se přehraje zvuk, který jste našli v předchozím bodě úlohy.

Řešení 3.bodu úlohy

Přepneme se na postavu jablka. Mezi bloky **Vyšli zprávu** a **Skryj se** umístíme **Změň proměnnou** o ze skupiny bloků **Proměnné**. Opět pracujeme s tou samou globální proměnnou jménem body. Po snězení jablka se hodnota bodů navýší o jedna, takže do prázdného místa u bloku Změň proměnnou o napíšeme jedničku.



Obrázek 107 Řešení 3.bodu úlohy u jablka, u mrkve je totožné

Metodická poznámka

Tento postup můžeme ukázat na interaktivní tabuli či přes projektor. Úkol číslo čtyři má obdobné řešení a necháme ho na žácích.

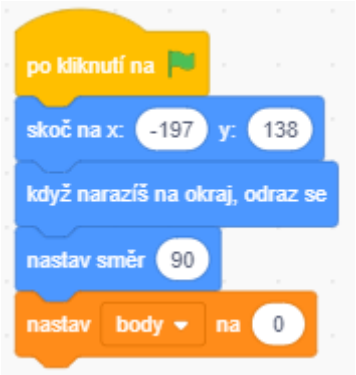
5. Co se stane s proměnnou pokud scénář párkrát spustíme? Je možné tento děj ošetřit a zabránit mu?

Metodická poznámka

Když želva sní nějaké ovoce, body se nasčítají přesně tak, jak jsme naprogramovali, ale při dalším spuštění tento počet bodů zůstane stejný. Podle zvyklostí ve videohrách, kde sbíráme body, by se správně měly body po opětovném spuštění vynulovat. Chvíli můžeme nechat žáky na řešení tohoto bodu úlohy přijít sami, ale nakonec bychom měli řešení opět ukázat na tabuli.

Řešení 5.bodu úlohy

Pod blok **Po kliknutí na zelenou vlaječku**, pod existující bloky, umístíme blok **Nastav proměnnou na**. V menu proměnných je na výběr pouze proměnná s názvem body – jediná proměnná, která v tomto scénáři existuje. Do prázdného místa napíšeme 0.



Obrázek 108 Řešení 5.bodu úlohy

Závěr

Žáci nyní umí základní příkazy pro práci s proměnnými.

Šestá lekce – ŽelvaV3 instrukce pro žáky

1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny postavy ve scénáři a pojmenujte ji „body“.

2. U jaké postavy se nachází zvuk? Jak se tento zvuk jmenuje?

3. Pokud se želva dotkne jablka, zvednou se body o 1. Také se přehraje zvuk, který jste našli v předchozím bodě úlohy.

4. Pokud se želva dotkne mrkve, zvednou se body o 1. Přehraje se ten samý zvuk jako u jablka.

5. Co se stane s proměnnou pokud scénář párkrát spustíme? Je možné tento děj ošetřit a zabránit mu?

Nápověda: Pro řešení pátého bodu úlohy si prohlédněte příkazy ve skupině bloků **Proměnné**.

Použité skupiny bloků: Zvuk, Proměnné

Šestá lekce – Drak a rytíř učitelský návod

Popis scénáře

V tomto scénáři si zopakujeme práci s podmínkami a proměnnými a seznámíme se s další kategorií operátorů.

Obrázek 109 Vzhled scénáře Drak a rytíř

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky

Proměnné

Cyklus – nové

Nově představené příkazy z programu Scratch

Operátory – Operátor <

Operátory – Operátor >

Pohyb - Změň y o

Vzhled – Přepni pozadí na

Využité bloky

Bloky pro práci s proměnnými, podmínkové bloky, bloky týkající se vzhledu

Použitý projekt

06 – Drak a rytíř – Teacher.sb3	soubor pro učitele
06 – Drak a rytíř – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Necháme žáky udělat úkoly hlavně samostatnou prací, koneckonců jde o opakování, do bodu 2.2. Nyní je čas vysvětlit něco málo z teorie, neboť se zde poprvé potkáme s novým typem operátoru. Konkrétně se jedná o symboly pro menší než „<“ a větší než „>“, které se využívají hlavně pro práci s proměnnými. Na tabuli můžeme tyto symboly namalovat a vysvětlit na konkrétní dvojici čísel, jak fungují. Poté můžeme zmínit, že místo čísel si zde pro účely programování představíme proměnné. Například můžeme uvést podmínku, která se stane platnou v případě, že postavě ve hře klesnou životy na menší než 1 – hra tedy končí, postava umírá.

Jako pomůcku pro pochopení symbolů větší a menší než mohu doporučit říkanku: Krokódýl chce sníst větší číslo.

2.2 Ve scénáři se bude donekonečna kontrolovat, zda je rytíř naživu. Jaké číslo musíme dát do podmínky? Doprogramuj si další kroky bodu úlohy dva do těla podmínky a vyzkoušej.

Řešení bodu úlohy 2.2

Většina řešení je již známá, zde jde spíše o kombinaci cyklu, podmínky a operátorů, kterou bychom měli zobrazit na tabuli. Z **Ovládání** vybereme cyklus **Opakuj stále**, ten umístíme pod bloky, které jsme přidali v bodě pět.

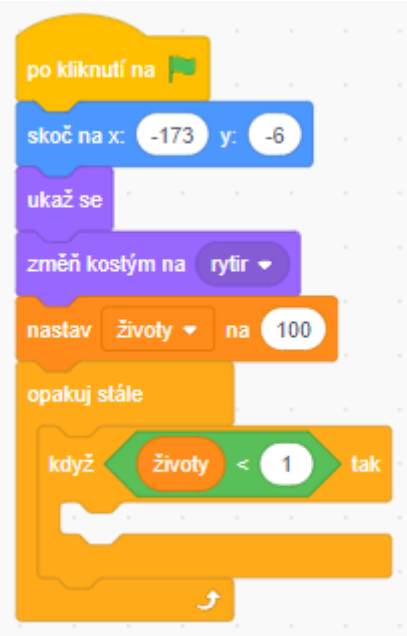
Do těla, to znamená „vnitřek“ cyklu umístíme blok **Když...tak**. Překlikneme se na skupinu bloků **Operátory** a do prázdného místa v podmínce vložíme zelený **blok se znamínkem menší než**, tedy **<**.

Ze skupiny bloků **Proměnné** přetáhneme proměnnou **životy** do levého prázdného místa **bloku se znaménkem menší než**. Do pravého místa zapíšeme jedničku. Teprve teď přidáváme příkazy v tabulce do těla této podmínky. První dva příkazy nejsou ničím novým, problém může nastat až u třetího.

Metodická poznámka

Řešení bodů 2.1 až 2.4 můžeme ukázat na tabuli. Při zkoušení, jaké číslo bude do podmínky sedět nejlépe se může stát, že nám logičtější přijde nula spíše než jednička. Při odzkoušení podmínky s nulou ovšem nastane situace, kdy rytíř, ačkoliv má nula životů stále stojí na nohou. Proto do bloku napíšeme jedničku.

Také je zde potřeba vysvětlit, co znamená výraz „tělo podmínky“ – jedná se o prostor uvnitř podmínky, kam umisťujeme příkazy, které se uskuteční pokud je podmínka platná. Ve Scratchi je tento prostor ohraničen podmínkovým blokem, v programování je hranice značena speciálními znaky.



Obrázek 110 Řešení bodu úlohy 2.2

2.3 Rytíř je poražen, leží na zemi a nastává konec hry. Naprogramuj.

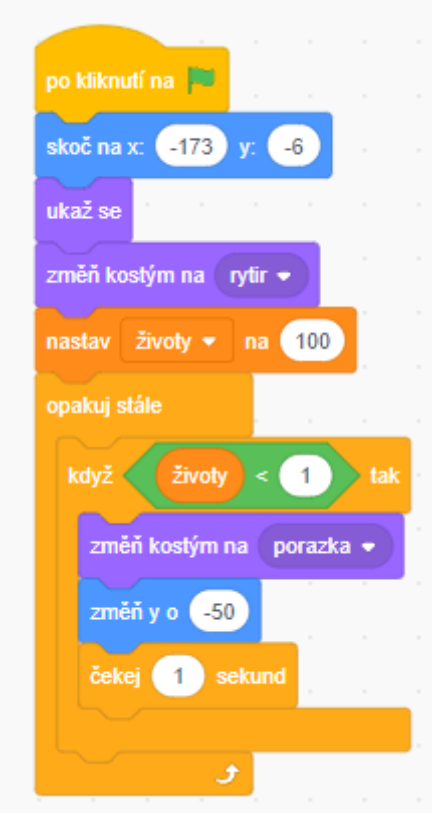
Metodická poznámka

Nyní můžeme vysvětlit, jak se řeší umístění prvků v programu Scratch. Jsou zde dvě souřadnice, x a y. Každá má kladnou a zápornou část a v rámci těchto souřadnic pohybujeme postavami. Také můžeme ukázat, že po vybrání jedné z postav vidíme pod scénářem čísla, která udávají jeho přesné umístění právě v rámci těchto souřadnic.

Na tabuli nakreslíme jednoduchou kresbu této kartézské souřadnice společně s několika příklady umístění, kde se nachází kladná část této souřadnice, kde záporná, jaká z čas je x a jaká y. Nemusíme jít nijak do hloubky, stačí pouze základní znalost, že nějaký takový systém existuje.

Řešení bodu úlohy 2.3

Jako první naprogramujeme změnu kostýmu. Ze skupiny bloků **Pohyb** vybereme příkaz **Změň y o** a umístíme ho pod předchozí blok. Do prázdného pole napíšeme -50, Scratch umí pracovat i s zápornými hodnotami. Tímto jsme naprogramovali pád rytíře na zem.



Obrázek 111 Řešení bodu 2.3

Metodická poznámka

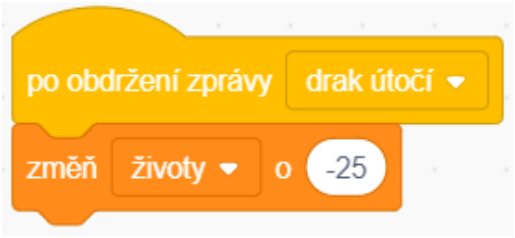
Nyní jsme pouze posunuli postavu o určitou hodnotu vůči jeho minulé pozici, narozdíl od přímo zadání polohy, které je taky možné naprogramovat. Příklad takového příkazu je například hned první pod blokem, který určuje, co se děje po spuštění scénáře. Tuto informaci je vhodné sdělit i žákům.

Bod úlohy číslo tři necháme udělat žáky samostatně.

4. Naprogramuj, aby se po útoku draka životy rytíře zmenšili o 25.

Řešení 4.bodu úlohy

Pod blok s přijatou zprávou umístíme **Změň proměnnou o** ze skupiny bloků **Proměnné**. Budeme zde měnit jedinou přítomnou proměnou, která náleží k postavě rytíře, tedy životy. Pokud chceme odečítat, stačí napsat před požadovaný počet mínus. Zapíšeme tedy: -25.



Obrázek 112 Řešení 4.bodu úlohy

Metodická poznámka

Zbytek bodů k sestrojení cvičení jsou obdobná či jsme je již probírali v minulých lekcích. Jediné, na co bychom si měli dát pozor, je přepínání mezi postavami a psaní příkazů k této postavě, k jakému má přináležet. Tato úloha nejspíše zabere dvě, možná i tři hodiny, ale to není vzhledem k její komplexnosti nic zvláštního.

5. Po kliknutí na lektvar přibyde rytíři 20 životů. Jak naprogramuješ, aby počet jeho životů nepřekročil 100?

Řešení 5.bodu úlohy

Pomocí zpráv se postava rytíře dozví o kliknutí na letvar. Životy navýšíme podobným způsobem jako v cvičení ŽelvaV3 v této lekci. Nyní naprogramujeme podmínku.

Přesuneme se do skupiny bloků **Ovládání**. Zde vybereme blok **Když..tak**. Do prázdného místa, kam umisťujeme podmínku vložíme blok **>** ze skupiny bloků **Operátory**.

Nyní nám zbývá tuto podmínku naprogramovat. Na levou stranu umístíme název proměnné **životy** ze stejnojmenné skupiny bloků a do pravého prázdného místa zapíšeme 100.

Obrázek 113 Řešení 5.bodu úlohy

6.1 Naprogramuj, aby drak měl na začátku scénáře 120 životů. Můžeš použít stejně pojmenovanou proměnnou jako u rytíře?

Metodická poznámka

Scratch se sám ozve a nově vytvořenou proměnnou jménem již existující proměnné pojmenovat nedovolí. Necháme třídu pracovat na úloze až do bodu devět.

9. Pokud vyhraje souboj drak, nastane konec hry, všechny postavy zmizí a bude viditelné pouze pozadí s nápisem konec hry. Co se stane když nyní znovu zapneš scénář? Co je potřeba udělat, aby byla hra opět hratelná?

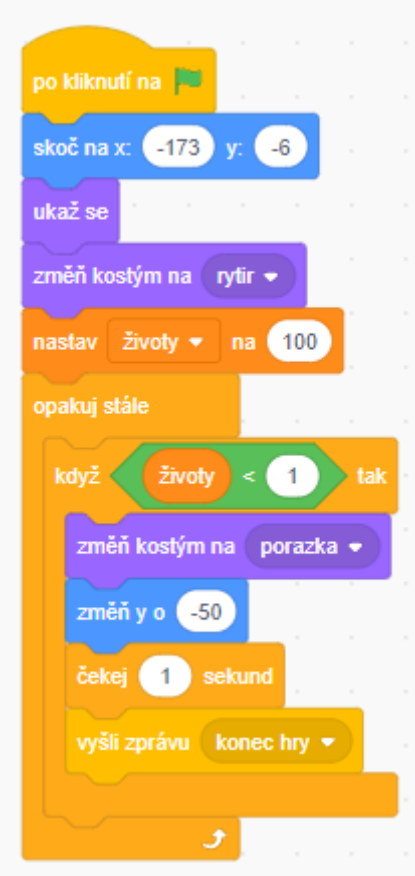
Metodická poznámka

Pokud naprogramujeme pouze mizení, postavy zůstanou schované, dokud nenaprogramujeme při zapnutí scénáře jejich ukázání se.

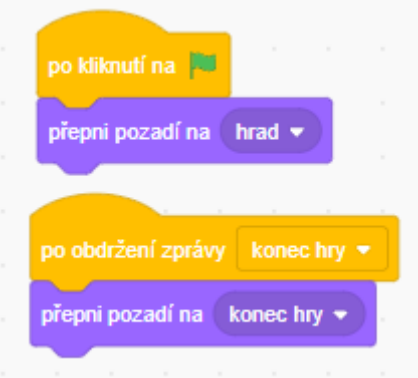
Řešení 9.bodu úlohy

Pokud rytíř souboj prohraje, vyšle zprávu ohledně konce hry. Poté, co tuto zprávu přijmou všechny postavy ve scénáři (včetně draka) tak se schovají pomocí bloku **Ukaž se/Skryj se** ze skupiny bloků **Vzhled**. Tento blok se použije i u opětovného objevení postav po zapnutí scénáře.

Vyřešit nyní zbývá pouze změna pozadí. Vybereme menu scény a do prostoru scénáře vložíme pod blok o konci hry blok s názvem **Přepni pozadí na** ze skupiny bloků **Vzhled**. V menu tohoto bloku vybereme pozadí s názvem „konec hry“. Obdobně naprogramujeme přepnutí pozadí na pozadí s názvem „hrad“ po zapnutí scénáře.



Obrázek 114 Posílání zprávy u bodu úlohy 9



Obrázek 115 Řešení 9.bodu úlohy

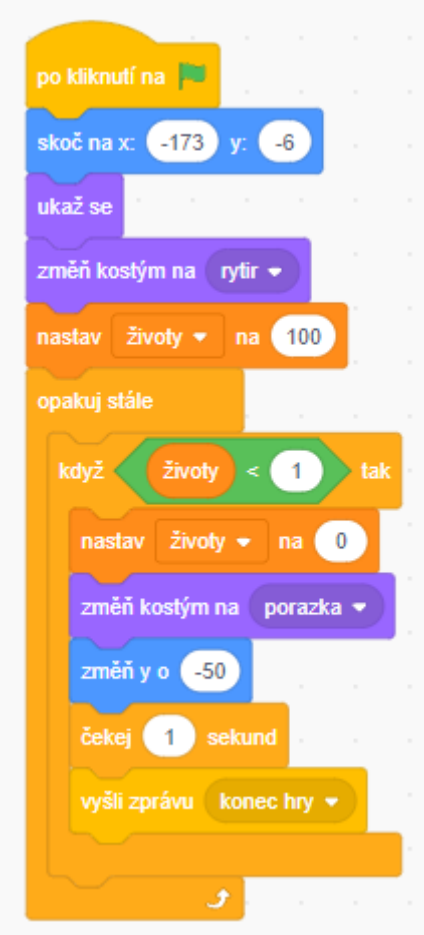
10. Co se stane s životy rytíře pokud bude poražen? Jak naprogramuješ řešení, aby k tomu nedocházelo a kam blok umístíš?

Metodická poznámka

Životy se budou donekonečna odečítat a budou nám vznikat záporné hodnoty.

Řešení 10.bodu úlohy

U rytíře pod blokem **Po stisknutí zelené vlaječky** do podmínky umístíme nad blok se změnou kostýmu blok **Nastav proměnnou na** ze skupiny bloků **Proměnné**. Do místa s názvem proměnné vybereme proměnnou značící rytířovy životy a do prázdného místa napíšeme 0.



Obrázek 116 Řešení 10.bodu úlohy

Metodická poznámka

U draka nastane stejná situace s nekonečným odečítáním, opět stačí umístit blok **Nastav proměnnou na** s vybranou proměnnou značící drakovy životy nad jediný existující blok v podmínce. Do prázdného místa opět napíšeme nulu.

Závěr

Žáci si vyzkoušeli použití dalšího z operátorů a zopakovali práci s proměnnými.

Šestá lekce – Drak a rytíř instrukce pro žáky

1. Zjisti si, jaké kostýmy mají postavy k dispozici a jaká pozadí jsou v programu dostupná. V dalších bodech úlohy s nimi budeme pracovat.

2.1 Naprogramuj, aby rytíř měl na začátku scénáře 100 životů.

2.2 Ve scénáři se bude donekonečna kontrolovat, zda je rytíř naživu. Jaké číslo musíme dát do podmínky? Doprogramuj si další kroky bodu úlohy dva do těla podmínky a vyzkoušej.

2.3 Rytíř je poražen, leží na zemi a nastává konec hry. Naprogramuj.

3 Po kliknutí na meč, zaujme rytíř bojovnou pozu a poté se vrátí do klidové polohy. Naprogramuj.

4. Naprogramuj, aby se po útoku draka životy rytíře zmenšili o 25.

5. Po kliknutí na lektvar přibyde rytíři 20 životů. Jak naprogramuješ, aby počet jeho životů nepřekročil 100?

6.1 Naprogramuj, aby drak měl na začátku scénáře 120 životů. Můžeš použít stejně pojmenovanou proměnnou jako u rytíře?

6.2 Ve scénáři se bude donekonečna kontrolovat, zda je drak naživu. Dále naprogramuj do těla podmínky následující kroky bodu úlohy tři.

6.3 Drak je poražen a změní se na hvězdičku s počtem bodů.

7. Po kliknutí na tlačítko s nápisem „drak útočí“ drak plivne oheň a poté se vrátí do původní polohy. Naprogramuj.

8. Po útoku rytíře na draka se životy draka zmenší o 30.

9. Pokud vyhraje souboj drak, nastane konec hry, všechny postavy zmizí a bude viditelné pouze pozadí s nápisem konec hry. Co se stane když nyní znovu zapneš scénář? Co je potřeba udělat, aby byla hra opět hratelná?

10 Co se stane s životy rytíře pokud bude poražen? Jak naprogramuješ řešení, aby k tomu nedocházelo a kam blok umístíš?

11. Podobně se stane i s drakem, jak to vyřešíš u něho?

Úkol navíc: Vytvoř další pozadí, tentokrát znamenající vítězství. Pokud bude drak zabit, všechny postavy se skryjí a bude vidět pouze pozadí. Řešení je obdobné jako u prohry.

Použité skupiny bloků: Pohyb, Vzhled, Události, Ovládání, Operátory, Proměnné

Sedmá lekce –ObchodV1 učitelský návod

Popis scénáře

V tomto cvičení budeme více pracovat s proměnnými a také se seznámíme s použitím uživatelského vstupu v programu Scratch.

Obrázek 117 Vzhled scénáře ObchodV1

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky

Proměnné

Uživatelský input - nové

Nově představené příkazy z programu Scratch

Vnímání – Otázka

Vnímání – Odpověď

Vzhled – Bublina

Operátory – Operátor *

Využité bloky

Bloky týkající se vstupu uživatele, podmínkové bloky, bloky týkající se vzhledu, bloky pro práci s proměnnými

Použitý projekt

07 – ObchodV1 – Teacher.sb3	soubor pro učitele
07 – ObchodV1 – Student.sb3	soubor pro žáky

Praktická část

Metodická poznámka

Necháme žáky dodělat úlohu až do třetího bodu sestrojení. Nyní žákům vysvětlíme, jak vstup od uživatele funguje.

Vstup, nebo také input, uživatele má dvě části. Jako první položíme otázku pomocí stejnojmenného příkazu ve Scratchi. Zaškrtneme odpověď, neboť s ní chceme pracovat a dále s ní zacházíme jako s jakoukoliv jinou proměnnou.

Důležitým předpokladem pro um programování je znalost angličtiny, takže můžeme zadat samostatnou práci. Na tabuli nakreslíme čtverec s šipkou dovnitř a šipkou ven. Chvíli necháme přemýšlet jednotlivce nad tím, kde se bude nacházet output a kde input. Co to vlastně takový output je? Jak se může projevit a jak vypadá? Jak vypadá input?

Poté, co si jednotlivci utřídí své představy je mohou sdělit sousedovi. Necháme chvíli dvojice diskutovat a vyslechneme si mluvčího z každé dvojice a jejich odpovědi. Neříkáme to je špatně, to dobře, správné řešení vysvětlíme až u konce diskuze.

1. Kde se nachází input a output?

2. Co to je output? Jak může vypadat?

3. Jak může vypadat input?

Obrázek 118 Řešení společně s otázkami ohledně konceptu input/output

Input je cokoliv, co dáme jako vstupní informaci do programu, cokoliv co do programu vejde a program s tím nějak pracuje. Může to být právě odpověď na otázku jako v programu Scratch, může to být kliknutí na nějakou postavu. Zkrátka to je nějaký úkon uživatele, s kterým program poté pracuje. Příklad z reálného života je například kalkulačka, uživatel tam napíše čísla, operaci, kterou chce s čísly provést a zbytek nechá na programu uvnitř přístroje.

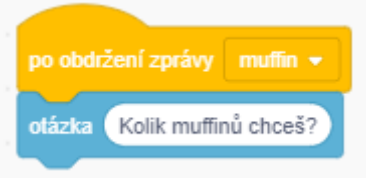
Output je výsledek nějakého úkonu systému, program si dle svého upravil a spočítal co mohl a nyní ukáže výsledek. Stále se držíme kalkulačky, takže output je výsledek, který se nám zobrazí na obrazovce.

3.1 Jako první si naprogramujeme koupi muffinu. Pokud si muffin vybereme kliknutím, prodavačka se nás zeptá: Kolik muffinů chceš?

Řešení bodu úlohy 3.1

O kliknutí na muffin se dozvíme pomocí zprávy, to je žákům již známý koncept. Následující bloky se tudíž nacházejí pod blokem **Po obdržení zprávy** u postavy prodavačky.

Pod blok **Po obdržení zprávy** ze skupiny bloků **Vnímání** vybereme a vložíme blok **Otázka**. Do prázdného místa napíšeme: Kolik muffinů chceš?



Obrázek 119 Řešení bodu úlohy 3.1

3.2 Jeden muffin stojí 25 kč. Do proměnné součet si uložíme celkovou cenu nákupu. Jak ji vypočítáme?

Metodická poznámka

Cenu za celý nákup, tedy proměnnou součet vypočítáme vynásobením ceny za jeden kus muffinu krát počet kusů, který se dozvíme z inputu uživatele – ve Scratchi tedy z odpovědi.

Řešení bodu úlohy 3.2

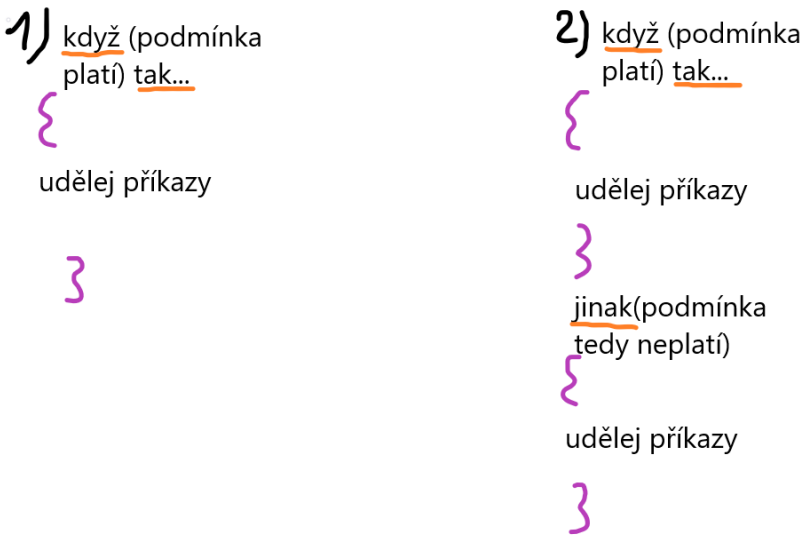
Dalším krokem je práce s takto získanou proměnnou. Z **Proměnných** vybereme blok **Nastav proměnnou na**. Do prázdného místa vložíme blok * z **Operátorů**. Proměnnou, které chceme nastavovat hodnotu je součet.

Do levé strany tohoto zeleného bloku vložíme cenu jednoho muffinu, tedy „25“ a do pravé přetáhneme **Odpověď** z **Vnímání**, získanou proměnnou z uživatelského vstupu.

Obrázek 120 Řešení bodu úlohy 3.2

Metodická poznámka

Nyní třídě ukážeme řešení u muffinu, obdobně se řeší zbytek cvičení. Setkáme se tu s novým stylem podmínky, kterou můžeme i nakreslit pro lepší porozumění.



Obrázek 121 Příklad vysvětlení podmínek dostupných ve Scratchi

Ve výuce můžeme slovně popsat tento Obrázek. Můžeme například kontrolovat, zda se kontrolované číslo rovná tomu, které chceme na vstupu, zda je číslo nenulové, zda je větší či menší než nějaká hodnota s kterým ho porovnáváme – využíváme operátory větší než, menší než a rovná se. Pokud kontrolujeme pouze jednu podmínku, můžeme pro jakoukoliv jinou možnost využít část podmínky „jinak“. Takto učiníme i při řešení bodu úlohy 3.3 (pokud je podmínka platná) a 3.4 (možnost jinak – podmínka platná není).

Stále se nacházíme pod stejným blokem **Po obdržení zprávy „muffin“**.

3.3 Pokud je cena nákupu větší než počet peněz, které máme u sebe, prodavačka řekne: Nemáš dostatek peněz.

Řešení bodu úlohy 3.3

Jako první si vložíme podmínku, se kterou budeme pracovat. Poslouží nám k tomu výše vysvětlený blok **Když..tak..jinak** ze skupiny bloků **Ovládání**.

Do prázdného místa podmínky vložíme blok **<** z **Operátorů**. Do levé strany, tedy té menší veličiny, přetáhneme **peníze** ze skupiny bloků **Proměnné** a do pravé strany s větší hodnotou přetáhneme **součet**. Vlastně je jedno, jaký z bloků z **Operátorů** použijeme, zda ten co má otevřený zobáček na levou stranu nebo na pravou. Důležité je zde dodržet, že kontrolujeme, zda jsou peníze menší než součet a správně umístit proměnné.

Obrázek 122 Řešení .bodu úlohy 3.3

Metodická poznámka

Přidání bubliny můžeme ukázat na tabuli, ale jde pouze o již známou věc. Pokračujme tedy na případ, kdy se podmínky nesplní a uživatel má dostatek peněz na zaplacení nákupu. Zvednutí ruky je řešeno změnou kostýmu, novinkou je pro žáky zobrazení hodnoty proměnné v bublině a odečtení ceny nákupu od peněz, s kterými jsme do obchodu přišli.

3.4 Pokud je peněz dostatek na zaplacení nákupu, prodavačka zvedne ruku a řekne cenu za nákup.

Řešení bodu úlohy 3.4

V bublině lze zobrazovat kromě textu i hodnoty uložené v proměnných v paměti. Toho docílíme, když z **Vzhledu** vybereme blok **Bublina**. Do prázdného místa přetáhneme proměnnou **součet** z **Proměnné**.

Obrázek 123 Řešení bodu úlohy 3.4

3.5 Cena nákupu se odečte od peněz, prodavačka se postaví do počáteční pozice a poděkuje za nákup.

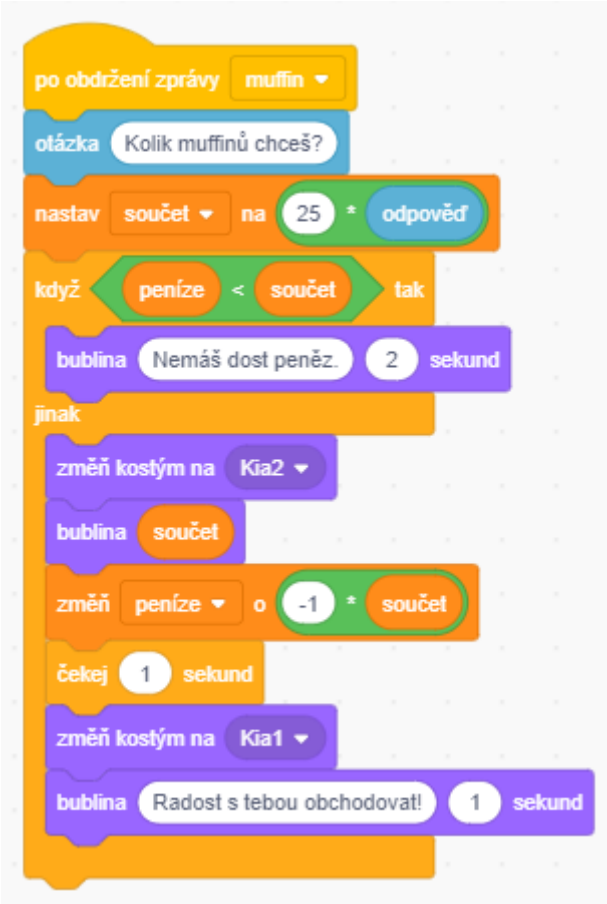
Řešení bodu úlohy 3.5

Odečítání hodnot proměnných od jiných proměnných bohužel není ve Scratchi tak jednoduché, není to ale neřešitelné. Ze skupiny bloků **Proměnné** vybereme blok **Změň proměnnou o**. Odečítáme od peněz, vybereme z nabídky proměnných tedy **peníze**. Do prázdného místa vložíme blok z **Operátorů** se znamínkem *. Do levé strany napíšeme -1, tím způsobíme, že číslo bude záporné a bude tedy odečítáno. Do pravé strany vložíme **součet** z **Proměnných**.

Obrázek 124 Řešení odečtení od proměnné peníze u bodu úlohy 3.5

Metodická poznámka

Zbývající příkazy pro dokončení bodu úlohy 3.5 jsou již známé, takže je na nás, zda je ukážeme či ne.



Obrázek 125 Kompletní řešení bodu úlohy 3.5

Metodická poznámka

Nákup mléka a marmelády je řešen obdobně, ten již můžeme nechat řešit žáky samotné.

Závěr

Žáci se seznámili s dalším typem podmínky a s programátorskými koncepty vstupu a výstupu.

Sedmá lekce – ObchodV1 instrukce pro žáky

1. Pokud klikneme na marmeládu, muffin a mléko, dají tyto postavy o kliknutí vědět ostatním postavám. Jak to provést?

2. Založ si dvě proměnné: jednu jménem „peníze“ a další má název „součet“. Zvaž, zda bude lepší, aby byly lokální nebo globální. Na začátku nákupu začneme s 400 korunami.

3.1 Jako první si naprogramujeme koupi muffinu. Pokud si muffin vybereme kliknutím, prodavačka se nás zeptá: Kolik muffinů chceš?

3.2 Jeden muffin stojí 25 kč. Do proměnné součet si uložíme celkovou cenu nákupu. Jak ji vypočítáme?

3.3 Pokud je cena nákupu větší než počet peněz, které máme u sebe, prodavačka řekne: Nemáš dostatek peněz.

3.4 Pokud je peněz dostatek na zaplacení nákupu, prodavačka zvedne ruku a řekne cenu za nákup.

3.5 Cena nákupu se odečte od peněz, prodavačka se postaví do počáteční pozice a poděkuje za nákup.

Nápověda: Na řešení 3. - 5. bodu úlohy použij blok **Když…tak..jinak**.

4. Naprogramuj podobně i nákup marmelády za 80 kč.

5. Mléko stojí 18 kč, naprogramujte obdobně jako u marmelády a muffinu.

Použité skupiny bloků: Vzhled, Události, Ovládání, Vnímání, Operátory, Proměnné

Sedmá lekce –ObchodV2 učitelský návod

Popis scénáře

Nyní si na minulém cvičení ukážeme, jak funguje programátorský koncept seznamů.

Obrázek 126 Vzhled scénáře ObchodV2

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky

Proměnné

Uživatelský input

Seznam – nové

Nově představené příkazy z programu Scratch

Operátory – Spoj

Proměnné – Smaž všechno ze seznamu

Proměnné – Přidej k seznamu

Využité bloky

Bloky týkající se práce se seznamem

Použitý projekt

07 – ObchodV2 – Teacher.sb3	soubor pro učitele
07 – ObchodV2 – Student.sb3	soubor pro žáky

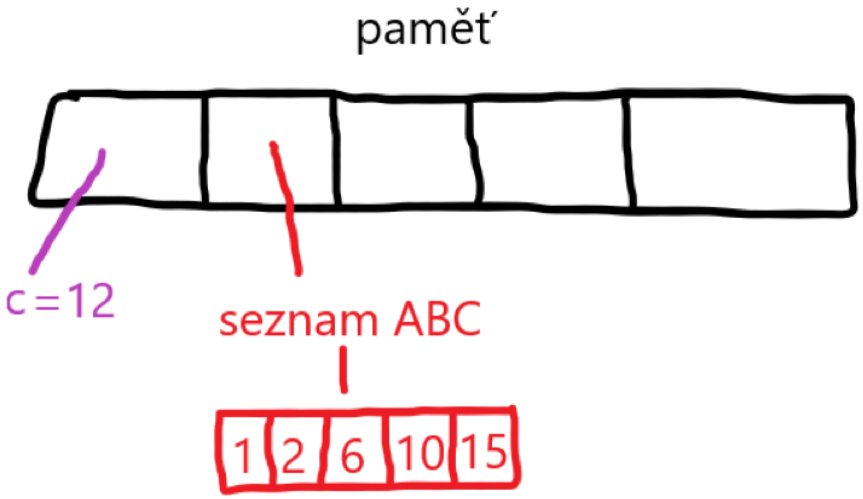
Praktická část

Metodická poznámka

Nyní se na v přechozí hodině sestrojeném cvičení seznámíme s konceptem seznamu. Je to místo v paměti programu. Narozdíl od obyčejné proměnné, která je pouze jediný údaj, seznam má údajů několik. Tyto údaje spolu nějak souvisí, například označují stejnou proměnnou a její proměny v čase. S seznamy se zachází uplně stejně jako s jakoukoliv jinou proměnnou, můžeme ji mazat, přidávat do ní hodnoty. Seznamy se ovšem liší jejich schopností pracovat s pořadími svých prvků a možností změny hodnoty/vymazání/vložení prvku, který je v pořadí na určitém místě, například na šestém.

Stejně jako proměnné může seznam být globální, pro všechny postavy, či lokální, pouze pro ten u kterého je vytvořen.

Na obrázku vidíme proměnou c, obyčejnou proměnnou a seznam ABC, který obsahuje pět prvků.



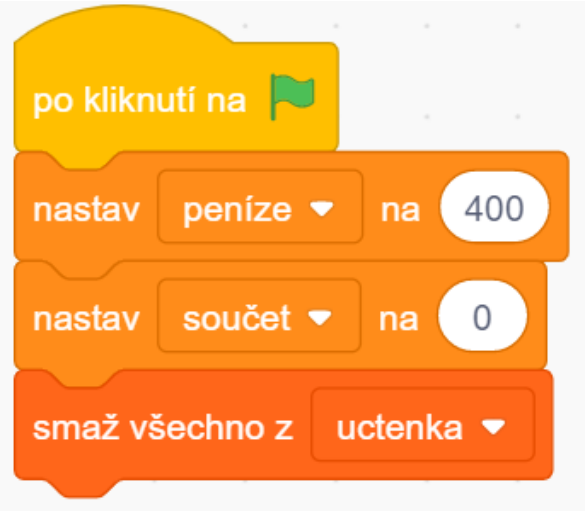
Obrázek 127 Znázornění seznamu na tabuli

V tomto cvičení si ukážeme zápis slov a číselné hodnoty proměnné do seznamu a také promazání celého seznamu. Seznam s názvem „uctenka“ je pro potřeby prvotního setkání s tímto konceptem v souboru pro studenty založený.

1. Naprogramujte vymazání účtenky po zapnutí scénáře.

Řešení 1.bodu úlohy

Ve skupině bloků **Proměnné** nyní vidíme nové možnosti týkající se seznamu. Vybereme **Smaž všechno z** a vložíme příkaz pod některý z bloků se zelenou vlaječkou. Jeden je u pozadí a další u postavy, nezáleží na tom, kam příkaz umístíme – uctenka je globální. Uctenka je rovněž jediný ve scénáři existující a tak je volba jasná.



Obrázek 128 Řešení 1.bodu úlohy

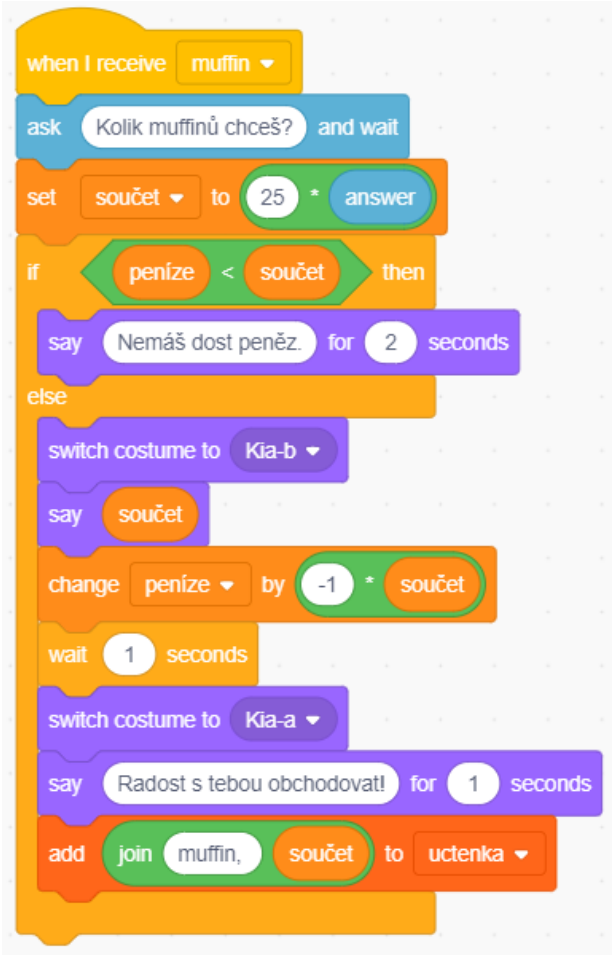
2. Naprogramujte, aby se do účtenky zapsalo po koupi muffinu slovo muffin a cena za všechny zakoupené muffiny.

Řešení 2.bodu úlohy

Ze skupiny bloků **Proměnné** vybereme **Přidej věc k seznamu**. Vymažeme zástupný název věc z prázdného místa a vložíme do něj **Spoj jablko banán** z **Operátorů**. Opět je k dispozici pouze jeden seznam a to ten s názvem uctenka, tak ho vybereme.

Místo jablka napíšeme název kupované věci, v případě muffinu je to samozřejmě muffin. Můžeme zde vyzkoušet mezery a čárky. Jedna z možností zápisu je název koupené věci, čárka a mezera.

Místo banánu vložíme do prázdného místa proměnnou **součet** ze stejnojmenné skupiny bloků.



Obrázek 129 Řešení 2.bodu úlohy

Závěr

Žáci jsou nyní obeznámeni s programátorským konceptem seznamu.

Sedmá lekce – ObchodV2 instrukce pro žáky

1. Naprogramujte vymazání účtenky po zapnutí scénáře.

2. Naprogramujte, aby se do účtenky zapsalo po koupi muffinu slovo muffin a cena za všechny zakoupené muffiny.

3. Podobně jako u muffinu se do účtenky zapíše stejným způsobem i mléko. Naprogramujte.

4. Naprogramujte zapsání marmelády do účtenky, podobně jako u mléka a muffinu.

Úkol navíc: Do sešitu si pomocí kombinace kreslení a psaní znázorni, jak takový seznam funguje. Porovnej se sousedem.

Použité skupiny bloků: **Operátory**, **Proměnné**

Příloha 5

ŽelvaV3

Pátá lekce – ŽelvaV3 učitelský návod

Popis scénáře

V tomto cvičení si vysvětlíme, co je to proměnná a jak s ní manipulovat pomocí příkazů v programu Scratch.

Použité programátorské koncepty

Příkaz

Zprávy

Podmínky

Proměnné - nové

Použité oblasti Scratche

Zvuk

Proměnné - nové

Využité bloky

Začni hrát zvuk, nastav proměnnou na, změň proměnnou o

Použitý projekt

04 – ŽelvaV2 – Teacher.sb3	soubor pro učitele
04 – ŽelvaV2 – Student.sb3	soubor pro žáky

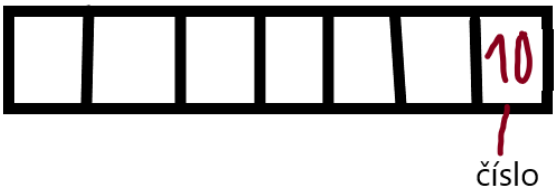
Praktická část

1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny avatary v programu a pojmenujte ji „jídlo“. Má u sebe některý z avatarů zvuk? Jak se jmenuje?

Metodická poznámka

Nyní budeme vysvětlovat, co to vlastně proměnná je. Jedná se o místo v paměti, kam se mohou ukládat různé hodnoty – ať už číselné nebo slovní. Toto místo můžeme přepisovat, navyšovat ho či z něj odečítat pokud se jedná o číslo. Každá proměnná má název, který by se neměl opakovat, aby v programu nebyl zmatek

Ukažte na tabuli/projektoru celý postup řešení prvního kroku úlohy. Také můžeme pro lepší představu namalovat na tabuli čtvereček znamenající místo v paměti. Poté vymýšlejte jednoduché příkazy typu: Přičti 2 a přepisujte ten samý čtvereček, aby bylo jasné, že jde o jedno a to samé místo v paměti. Můžete se inspirovat obrázkem pod textem, který můžeme bude namalovat na tabuli nebo do programu malování, podle vybavení třídy.



1) odečti 2 od čísla

2) přičti 10 k číslu

3) zdvojnásob číslo

Obrázek 130 vysvětlení proměnné na tabul

Existují dva typy proměnných – globální a lokální. Můžeme se zeptat třídy, co si pod pojmem globální v kontextu programování představuje a udělat podobné cvičení jako v minulé lekci. K sobě si každý napíše co podle něho tyto dva výrazy znamenají a poté se o svůj názor podělí se sousedem. Učitel se po nějaké době zeptá jednoho z dvojice či si dvojice sama vybere mluvčího za skupinu. Teprve až se všechny dvojice vyjádří řekneme správné řešení, do té doby by jsme ho neměli prozradit.

Lokální proměnná platí pouze pro určitou část programu, v případě Scratch to znamená pouze pro jeden z avatarů. Globální, jak už název napovídá, platí pro celý program a může ji upravovat každý z avatarů.

Řešení 1.bodu úlohy

Vyberte oblast s názvem **Proměnné**. Zde vidíme přednastavenou proměnnou s názvem Moje proměnná. Klikneme na ni pravým tlačítkem myši a smažeme ji.

Novou proměnnou založíme kliknutím na knoflík Vytvoř proměnnou, stále v oblasti **Proměnné**. Vybereme možnost „pro všechny proměnné“ a pojmenujeme ji jídlo. Volbu potvrdíme.

2. Po zapnutí programu se „jídlo“ nastaví na 0.

Řešení 2.bodu úlohy

Z oblasti **Proměnné** vybereme **Nastav proměnnou na**. Do pole s názvem zvolíme naši jedinou proměnnou, se kterou v tomto programu pracujeme. Do prázdného pole napíšeme 0. Tento blok umístíme pod počáteční blok programu – ten s vlaječkou.

3. Pokud se želva dotkne jablka, zvedne se jídlo o 2. Také se přehraje zvuk, jehož název už znáte.

Přepneme se na avatar jablka. Mezi bloky **Vyšli zprávu** a **Skryj se** umístíme **Změň proměnnou o**. Opět pracujeme s tou samou globální proměnnou. Hodnota jablka je 2, takže do prázdného políčka umístíme právě toto číslo.

Metodická poznámka

Tento postup ukážeme na tabuli. Úkol číslo čtyři má obdobné řešení a necháme ho na žácích.

Pátá lekce – ŽelvaV3 instrukce pro žáky

1. Vymažte proměnnou s názvem „moje proměnná“. Založte novou, viditelnou pro všechny avatary v programu a pojmenujte ji „jídlo“. Má u sebe některý z avatarů zvuk? Jak se jmenuje?

2. Po zapnutí programu se „jídlo“ nastaví na 0.

3. Pokud se želva dotkne jablka, zvedne se jídlo o 2. Také se přehraje zvuk, jehož název už znáte.

4. Pokud se želva dotkne mrkve, zvedne se jídlo o 1. Přehraje se ten samý zvuk jako u jablka.

Použité oblasti Scratche: Zvuk, Proměnné