UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

David Květoň

Sada úloh pro výuku programování ve Scratchi pro ZŠ

Olomouc 2023

vedoucí práce: doc. RNDr. Petr ŠALOUN, Ph.D.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a uvedl jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použil.

V Olomouci dne: …….

David Květoň

Poděkování

Chtěl bych poděkovat panu docentu Šalounovi za cenné rady a ochotu při spolupráci.

David Květoň

Anotace

Bakalářská práce je zaměřena na tvorbu sady úloh, které se zabývají vizuálním programovacím jazykem Scratch. Sada úloh může sloužit jako pomocný materiál pro výuku programování na druhém stupni základních škol v předmětu Informatika. V práci se nachází úlohy rozdělené do tří kategorií dle obtížnosti. V každé kategorii se nachází šest komentovaných úloh s řešením a jedna úloha bez řešení, která slouží jako úvod do dané kategorie. Celkem je tedy připraveno 21 úloh.

Klíčová slova

vizuální programovací jazyk, Scratch, úlohy, programování, informatické myšlení

Abstract

The bachelor's thesis focuses on creating a set of tasks centered around visual programming language Scratch. This set of tasks can serve as a supporting material for teaching programming at the secondary level of primary schools in the subject of IT. The thesis contains tasks divided into three categories based on difficulty. In each category, there are six annotated tasks with solutions and one task without a solution, which serves as an introduction to that particular category. In total, there are 21 tasks prepared.

Key words

visual programming language, Scratch, tasks, programming, computational thinking

OBSAH

ÚVOD

Vizuální programovací jazyk je skvělý nástroj pro oblast vzdělávání na základních školách v předmětu Informatika. Pomáhá žákům rozvíjet informatické myšlení a učí je základním principům programování v jednoduché formě vizuálních bloků, které se snadno manipulují a pomocí nichž lze vytvářet zajímavé interaktivní příběhy, animace, či hry. Z tohoto důvodu se tato bakalářská práce zaměřuje na tvorbu sady úloh, konkrétně ve vizuálním programovacím jazyce zvaném Scratch.

V teoretické části mé práce nastíníme pojem informatické myšlení a řekneme si jaké má využití v reálném životě. Dále si řekneme něco o digitálním vzdělávání, jeho strategii a novém pojetí Informatiky na ZŠ. V poslední řadě si pak představíme, co je vizuální programovací jazyk a jaké má využití. Ukážeme si příklady různých programovacích jazyků a nástroje pro možnost konverze do vyšších programovacích jazyků jako jsou JavaScript, Python, či C.

Praktická část bakalářské práce se věnuje tvorbě sady úloh ve Scratchi. Tato sada může sloužit jako pomocný materiál pro výuku programování na druhém stupni ZŠ, kterou mohou učitelé volně využívat, či se inspirovat. Kvůli odlišnosti znalostí a dovedností žáka je sada úloh rozdělena do tří obtížnostních kategorií. Tyto kategorie na sebe postupně navazují. Učitel se tedy může rozhodnout, zdali bude sadu úloh procházet chronologicky, anebo dle individuálních potřeb a schopnostech žáka přidělí úlohy v různém pořadí. První kategorie s názvem Elév se zaměřuje na orientaci v uživatelském rozhraní jazyka Scratch. Ve druhé kategorii s názvem Kadet se potom už žáci naučí pracovat s cyklem, podmínkou a proměnnou. V poslední kategorii Profík se již vytváří komplexnější projekty, většinou ve formě her. Každá kategorie obsahuje celkem šest úloh, které mají své zadání, doplňující úkoly a podrobný popis řešení a jednu úlohu bez řešení, která slouží jako motivační úvod do dané kategorie. Obsahem poslední kapitoly je potom samotné umístění a přístupnost sady úloh.

Cílem celé práce je vytvořit pomocný materiál pro výuku programovaní na ZŠ, který bude volně přístupný ke stažení a využití.

1 INFORMATICKÉ MYŠLENÍ

Myšlení je poznávací proces, který je charakteristický hned několika body. Skládá se z vnitřních, implicitních myšlenkových operací. Probíhá jednak na vědomé, kontrolované a řízené úrovni (myšlení logické, induktivní, deduktivní), jednak na neuvědomované úrovni (myšlení intuitivní). Obvykle se dá usměrňovat vůlí (myšlení volné). Může však probíhat bez volního úsilí (myšlení asociativní), dokonce i proti volnímu úsilí (myšlení vtíravé). [1]

1.1 Vymezení a definice pojmu informatické myšlení

Spojení informatické myšlení by se zjednodušeně dalo vysvětlit jako schopnost myslet jako informatik při řešení problémů. Computational thinking teacher resources uvádí relativně dobře srozumitelnou a konkrétní definici: Informatické myšlení je postup řešení problému, který zahrnuje mimo jiné následující charakteristiky: [15]

- Formulovat problémy způsobem, který umožňuje jejich strojové řešení.
- Logicky uspořádat a zkoumat data.
- Reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace.
- Automatizovat řešení pomocí algoritmického myšlení (jako posloupnost kroků).
- Odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů.
- Zobecňovat a přenášet tento postup řešení problémů do nejrůznějších dalších oblastí.

Všechny tyto dovednosti podporuje další nezbytná součást informatického myšlení, a tou jsou předpoklady a postoje. Jedinec by měl být sebejistý tváří v tvář složitosti, vytrvalý při řešení obtížného problému. Měl by také snášet nejednoznačnosti a být schopný vypořádat se s otevřenými problémy. A v neposlední řadě je důležité, aby se jedinec dokázal dorozumět a také spolupracovat s ostatními, a tak dosáhnout společného cíle. [6]

Do českého vzdělávání se požadavek na rozvoj informatického myšlení žáků dostává v roce 2014 prostřednictvím vládního dokumentu Strategie digitálního vzdělávání do roku 2020, který řadí rozvoj informatického myšlení žáků mezi své tři prioritní cíle. [7]

Selby a Woolard chtěli na diskuzi o informatickém myšlení vrhnout nové světlo. Proto se rozhodli zformulovat jednu ne příliš širokou definici a shrnout používanou terminologii. Pro tyto účely zanalyzovali celkem 22 různých definic pojmu „computational thinking“. Součástí výstupu jejich výzkumu je tabulka s přehledem pojmů, které bývají s informatickým myšlením spojovány či dokonce ztotožňovány, včetně vyjádření, zda příslušný pojem může být zahrnut do definice informatického myšlení či nikoliv: [9]

Tabulka 1: *Terminologie spojená s definicí informatického myšlení*

Pojem	**Lze zahrnout?**	**Zdůvodnění**
Myšlenkový proces	ano	Literární zdroje se shodují.
Abstrakce	ano	Literární zdroje se shodují.
Dekompozice (rozklad)	ano	Literární zdroje se shodují.
Logické myšlení	ne	Příliš široký pojem, nedostatečně definovaný.
Algoritmické myšlení	ano	Dobře definovaný napříč různými obory.
Řešení problémů	ne	Příliš široký pojem, je důkazem použití IM, nerozvíjí ho.
Hodnocení (evaluace)	ano	Dobře definovaný napříč různými obory.
Zobecnění (generalizace)	ano	Dobře definovaný napříč různými obory.
Navrhování systémů	ne	Důkaz použití IM.
Automatizace	ne	Důkaz použití IM.
Informatický obsah	ne	Důkaz použití IM.

Modelování a simulace	ne	Důkaz použití IM

Selby a Woolard na základě svých zjištění navrhují definici tohoto znění: *Informatické myšlení je činnost, typicky orientovaná na výsledek, spojována, ale ne výlučně omezena, na řešení problémů. Jedná se o kognitivní proces, který odráží schopnost:*

- *abstrahovat,*
- *rozkládat problém na podproblémy (dekompozice),*
- *myslet algoritmicky,*
- *hodnotit,*
- *zobecňovat (generalizace).* [9]

Jinými slovy je informatické myšlení dle Selbyho a Woolarda přístup zaměřený na řešení problémů zahrnující myšlenkový proces používající abstrakci, dekompozici, algoritmický přístup, hodnocení a zobecňování.

Je tedy zřejmé, že informatické myšlení nemá téměř nic společného s obsluhou počítače, s uživatelským přístupem k technologiím, protože „takové používání počítače informatické myšlení nerozvíjí“. [2]

E. W. Dijkstra, známý nizozemský informatik, který bývá řazen i mezi průkopníky informatiky, dokonce řekl: „Informatika není o počítačích o nic víc než astronomie o dalekohledech.“ [10]

Tento známý citát zmiňuje i Pelánek a objasňuje ho následovně: „Počítače, stejně jako dalekohledy, jsou jen prostředek. Informatik, jenž se stará jen o svůj počítač, je na tom stejně jako hvězdář, který se pro všechnu starost o svůj dalekohled zapomněl dívat na nebe.“ [3]

1.2 Složky informatického myšlení

Pro rozvoj informatického myšlení žáků je třeba vymezit, jaké složky koncept informatického myšlení zahrnuje. Do velké míry nám s tímto může pomoci definice Selbyho a Woolarda (2013, s. 5), neboť právě schopnost abstrakce, dekompozice, algoritmického myšlení, zobecňování a hodnocení jsou považovány za základní předpoklady informatického myšlení.

Cílem **abstrakce** je problém zjednodušit, a to určením částí problému, které jsou důležité a podstatné, a částí, které jsou naopak nepodstatné. Prostředkem pro znázornění abstrakce může být model, simulace, diagram, abstraktní jazyk apod. Jako příklad použití abstrakce v matematice lze uvést slovní úlohy, u kterých k vyřešení napomáhá vyjádřit si klíčové informace úlohy více abstraktním jazykem, například algebraicky.

Dekompozice je proces, při kterém je problém rozdělen na dílčí podproblémy. Tento přístup má mnoho výhod. Zaprvé z velkých zdánlivě neřešitelných problémů učiní sérii nebo strukturu menších problémů, jejichž řešení je snazší, protože je řešiteli problému například už známé. Zároveň rozklad na podproblémy vytváří vhodné podmínky pro týmovou práci.

Algoritmické myšlení představuje zejména proces tvorby algoritmů a algoritmických řešení, která neslouží jen k řešení jedné úlohy, ale jsou řešením celé skupiny úloh, které se od sebe liší vstupními údaji. Vzhledem k tomu, že je algoritmické myšlení vnímáno jako dominantní složka informatického myšlení.

Generalizace neboli **zevšeobecňování** je přístup založený na rozeznávání vzorů, podobností a spojitostí, které vedou k pochopení podstaty zkoumaného jevu. Generalizace umožňuje rychleji vyřešit nový problém na základě zkušeností z řešení předchozího podobného problému. Neinformatickým příkladem generalizace může být učení se správné výslovnosti anglických slov. Když se žák v rámci určitého slova naučí vyslovovat určitou 15 posloupnost písmen, dokáže ji správně vyslovit i v případě, kdy na ni narazí v jiném dosud neznámém slově. Poslední ze základních složek informatického myšlení je hodnocení. Jeho cílem je ověření toho, že navržené řešení je dobré a účelné. Za tímto účelem je potřeba vždy zhodnotit řešení z různých hledisek jako je správnost fungování, rychlost, efektivita, náročnost použití řešení z pohledu uživatele, řešení nestandardních situací apod. Součástí procesu hodnocení je i tzv. debugging tedy vyhledávání a ladění chyb. [9]

1.3 Využití informatického myšlení v běžném životě

Pokud bychom chtěli přiblížit využití informatického myšlení v běžném životě obyčejného člověka, museli bychom hledat nějaký často opakovaný a jednotvárný proces, při kterém se pracuje s množstvím nějakých položek. Například proces nakupování si můžeme urychlit uspořádáním položek na nákupním seznamu dle rozmístění zboží v konkrétní prodejně. Tím

předejdeme složitému hledání potravin v seznamu. Dalším příkladem by mohl být výběr pokladny v nákupním centru, abychom čekali co možná nejkratší dobu ve frontě, či uspořádání potravin v lednici dle data trvanlivosti. [8]

Díky informatickému myšlení můžeme zachraňovat i lidské životy. Například při řetězové transplantaci ledvin. Právě zapojení informatického myšlení na straně organizátorů dárcovského systému vedlo k rozpoznání a posouzení jeho řešitelnosti. Následná spolupráce s informatiky vedla ke zlepšení situace a nalezení efektivního algoritmu k vyřešení celého problému. Díky sestavenému řetězci je možné zachránit hned několik životů najednou. V Česku probíhají řetězové transplantace již několik let. [11]

2 DIGITÁLNÍ VZDĚLÁVÁNÍ

V současnosti jsme svědky mnoha společenských proměn, které zasahují bez výjimky všechny oblasti lidské činnosti. Abychom byli schopni na tyto proměny reagovat, je třeba na ně adekvátně připravit vzdělávací systém. Vzhledem k významným změnám ve společnosti, způsobeným dynamickým rozvojem, je nutné tomuto vývoji přizpůsobit obsah, metody a formy vzdělávání.

2.1 Vize digitálního vzdělávání

Vizí digitálního vzdělávání je, aby vzdělávací systém vybavil každého jedince bez rozdílu takovými kompetencemi, které mu umožní se uplatnit v informační společnosti a využívat nabídky otevřeného vzdělávání v průběhu celého života. [12]

Koncepce Digitální Česko v. 2.0, kterou v roce 2013 schválila Vláda České republiky, konkrétně uvádí: *„Informační technologie by měly postupovat celým procesem výuky na základních školách, nikoli, jen v předmětech typu Práce s počítačem. Plné zapojení moderních technologií do výuky všech předmětů vnímá stát jako nezbytné v rámci posunu vzdělávacího systému od prostého memorování faktů k důrazu na čtenářskou gramotnost, komunikační dovednosti a logické myšlení.“* Součástí usnesení vlády k této koncepci je soubor opatření, z nichž se jedno opatření týká problematiky vzdělávání a ukládá MPSV ve spolupráci s MŠMT vypracovat strategii pro zvýšení digitální gramotnosti a rozvoj elektronických dovedností občanů Cílem strategie je nastavit podmínky a procesy ve vzdělávání, které toto digitální vzdělávání umožní realizovat. [13]

2.2 Strategie digitálního vzdělávání v ČR

Vláda České republiky reaguje na přetrvávající rychlý vývoj v oblasti digitálních technologií a je si vědoma nutnosti a potřeby implementovat moderní technologie do výuky. V souvislosti s rychle postupující digitalizací společnosti je bezpochyby žádoucí, aby vzdělávací systém, s přihlédnutím k dynamice těchto změn, byl dostatečně flexibilní a adekvátně připraven.

Tomuto má napomoci Strategie vzdělávací politiky ČR do roku 2030+. Jedná se o aktuálně platný klíčový dokument, který navazuje na strategický dokument pro oblast vzdělávání vydaný pod názvem Strategie vzdělávací politiky České republiky do roku 2020. Strategie 2030+ je stěžejním dokumentem pro rozvoj vzdělávacího systému v ČR pro nadcházející desetiletí let 2020 – 2030. Zmíněná strategie si klade za cíl zmodernizovat vzdělávací systém tak, aby děti i dospělí obstáli v dynamickém a neustále se měnícím světě 21. století, dále připravit ho na nové výzvy a řešit přetrvávající problému, které v česku panují. Dokument vymezuje dva hlavní strategické cíle a pět strategických linií, které představují cesty a nástroje k realizace těchto cílů. [14]

2.3 Cíle strategie vzdělávací politiky ČR do roku 2030+

Dnešní žáci se velmi výrazně liší oproti svým předchozím vrstevníkům. Za společný socializační znak soudobé generace je považováno hlavně využívání digitálních technologií. Předmětná strategie si proto klade následující cíle: [14]

- využívat moderní technologie k dosažení nově definovaných vzdělávacích cílů,
- vytvořit podmínky pro rozvoj digitálního vzdělávání všech žáků a učitelů,
- zvýšit úroveň kompetencí v oblastech užívání digitálních technologií, informatického myšlení a digitální gramotnosti,
- uzpůsobit vzdělávací systém, aby byl schopen přiměřeně se přizpůsobit dynamickému prostředí a pokroku spojeného s rozvojem nových technologií, digitalizace a internacionalizace,
- snažit se zvýšit úroveň digitálních dovedností a informatického myšlení,
- zahrnout informační a datovou gramotnost, komunikaci a spolupráci, mediální gramotnost, tvorbu digitálního obsahu, bezpečnost v on-line prostředí, ale i řešení problémů a kritické myšlení do procesu vzdělávání.

2.4 Nové pojetí výuky informatiky na ZŠ

Doposud se žáci věnovali většinou práci s textovými, tabulkovými a prezentačními editory. Nové pojetí výuky informatiky kromě základů uživatelských dovedností pro práci se zařízeními a aplikacemi přináší a klade důraz:

- na informatické myšlení,
- strukturovaně přemýšlet (i s využitím počítačů a aplikací),
- žáci by se měli učit pracovat s informacemi,
- žáci by měli umět popsat problém, analyzovat jej a hledat funkční řešení,
- porozumět principům digitálních technologií,
- bezpečně a eticky využívat digitálních technologie a týmovou práci. [14]

Inovovat obsah vzdělávací oblasti informatika s důrazem na rozvoj informatického myšlení žáků výrazně přispěl projekt, který byl realizován v rámci Operačního programu Výzkum, vývoj a vzdělávání. Konkrétně se jedná o projekt „Podpora rozvíjení informatického myšlení“ (dále jen ve zkratce PRIM), na jehož spolufinancování se podílela Evropská unie. Hlavním příjemcem a garantem projektu, jehož realizace byla úspěšně ukončena k datu 30. 11. 2020, byla Jihočeská univerzita v Českých Budějovicích. Mezi další spolupracující partnery se řadí všechny pedagogické fakulty v ČR a Národní ústav pro vzdělávání.

V rámci tohoto projektu se také podařilo úspěšně vytvořit ucelené sady materiálů pro výuku, a to pro všechny stupně škol. Skutečnosti, proč využít zrovna, a právě prostředí vizuálního programovacího jazyka Scratch pro praktickou část mé bakalářské práce přispívá i fakt, že byl zařazen projektem PRIM, jakožto výchozí produkt, ve kterém by se měli začít učit programovat již žáci od 1. stupně ZŠ a plynule v něm navázat a pokračovat na 2. stupni ZŠ (Obrázek 1).

Obrázek 1: *Ukázka ucelené sady výukových materiálů dle projektu PRIM (2021)*[1]

[1] Obrázek 1: převzato z webu imysleni.cz. Dostupné online z: https://www.imysleni.cz/ucebnice

3 VIZUÁLNÍ PROGRAMOVACÍ JAZYK

V této kapitole se budeme zabývat pojmem vizuální programovací jazyk (ve zkratce VPL – anglicky Virtual Programming Language). Jak už název vypovídá, nebude se jednat o klasické konvenční programování, kde je zdrojový kód zapisován pomocí konstruktů datového programovacího jazyka a reprezentován zejména v textové podobě. V případě vizuálního programování je kód reprezentován graficky, pomocí obrázků. Ve výpočetní technice VPL je programovací jazyk, který uživatelům umožňuje vytvářet programy manipulací s programovými prvky graficky na rozdíl od klasického programování, kde se program vytváří pomocí textové syntaxe. [4]

VPL umožňuje programování s vizuálními výrazy, prostorovým uspořádáním textu a grafických symbolů, které se používají buď jako prvky syntaxe nebo sekundární notace. Například mnoho VPL je založeno na myšlence „boxů a šipek“, kde se s boxy nebo jinými objekty obrazovky zachází jako s entitami spojenými šipkami, čarami nebo oblouky, které představují relace (nebo také vztahy) mezi těmito prvky. [16]

3.1 Cíl VPL

Obecným cílem VPL je zpřístupnit programování začátečníkům a podporovat programátory na třech různých úrovních: [17]

1) **Syntax:** VPL využívají ikony/bloky, formuláře a diagramy s cílem snížit nebo dokonce eliminovat možnost syntaktických chyb a pomáhají s uspořádáním programovacích primitiv k vytvoření správně utvořených programů
2) **Sémantika:** VPL mohou poskytovat mechanismy k odhalení významu programovacích primitiv. To může zahrnovat pomocné funkce poskytující dokumentační funkce zabudované do programovacích jazyků
3) **Pragmatika:** VPL podporují studium toho, co programy znamenají v konkrétních situacích. Tato úroveň podpory uživatelům umožňuje umístit artefakty vytvořené pomocí VPL do určitého stavu, aby zkoumali, jak program zareaguje na tento stav. Například s programovacím jazykem pro robota Thymio mohou uživatelé dostat robota do určitého stavu, aby viděli, jak zareaguje, tedy které senzory budou aktivovány.

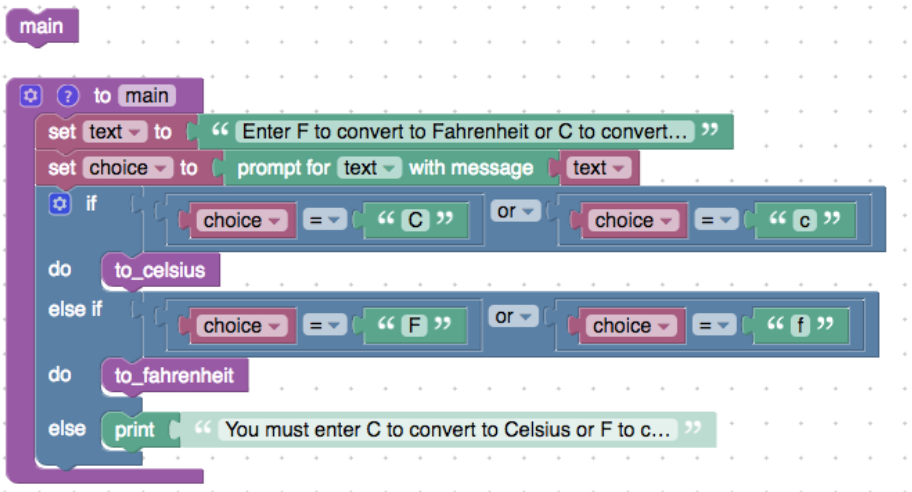
3.2 Příklady vizuálních programovacích jazyků

Za jeden z nejznámějších a nejvíce využívaných vizuálně programovacích jazyků v oblasti vzdělávání se dá považovat Scratch. A jelikož se praktická část mé bakalářské práce zaměřuje právě na tento jazyk, věnuji mu samostatnou podkapitolu a popíšu jej hlouběji. Existuje však mnoho dalších vizuálních programovacích jazyků, které jsou navrženy s různými cíli a zaměřením. Proto jich níže pár uvedu a stručně popíšu jejich využití.

3.2.1 Blockly

Byl vyvinutý společností Google. Jedná se o javascriptovou knihovnu, která je free a open source, tedy může být využívaná širokou veřejností zdarma. Blockly poskytuje vizuální rozhraní pro vytváření vlastních programovacích editorů pro tvorbu specifických typů aplikací, jako jsou vzdělávací nástroje, hry nebo nástroje pro vývoj softwaru. Knihovnu Blockly využívá i Scratch. Scratch byl totiž postaven do roku 2019 na technologii Flash. Po skončení podpory této technologie přešel Scratch právě na technologii Blockly.

Obrázek 2: *Ukázka kódu v Blockly*[2]



[2] Obrázek 2: převzato z webu wikimedia.org [online]. Dostupné online z: https://upload.wikimedia.org/wikipedia/commons/9/95/Blockly_Conditions_main.png

3.2.2 App Inventor

Tento VPL umožňuje uživatelům vytvářet mobilní aplikace pro platformu Android pomocí vizuálního programování. Tento nástroj byl vyvinut na MIT (Massauchusetts Institute of Technology). Je free a open source, tedy přístupný všem zdarma. Díky tomuto nástroji mohou uživatelé bez hlubší znalosti klasického programování vytvářet funkční aplikace. Jelikož je tento nástroj vizuálně založen, usnadňuje porozumění základům tvorby aplikací, a proto je také vhodný pro výuku ve školách a vzdělávacích institucích.

Obrázek 3: *Ukázka uživatelského rozhraní App Inventoru*[3]

[3] Obrázek 3: převzato z webu cdn-learn.adafruit.com [online]. Dostupné online z: https://cdn-learn.adafruit.com/guides/cropped_images/000/001/277/medium640/AppInvent09.PNG?1535067563

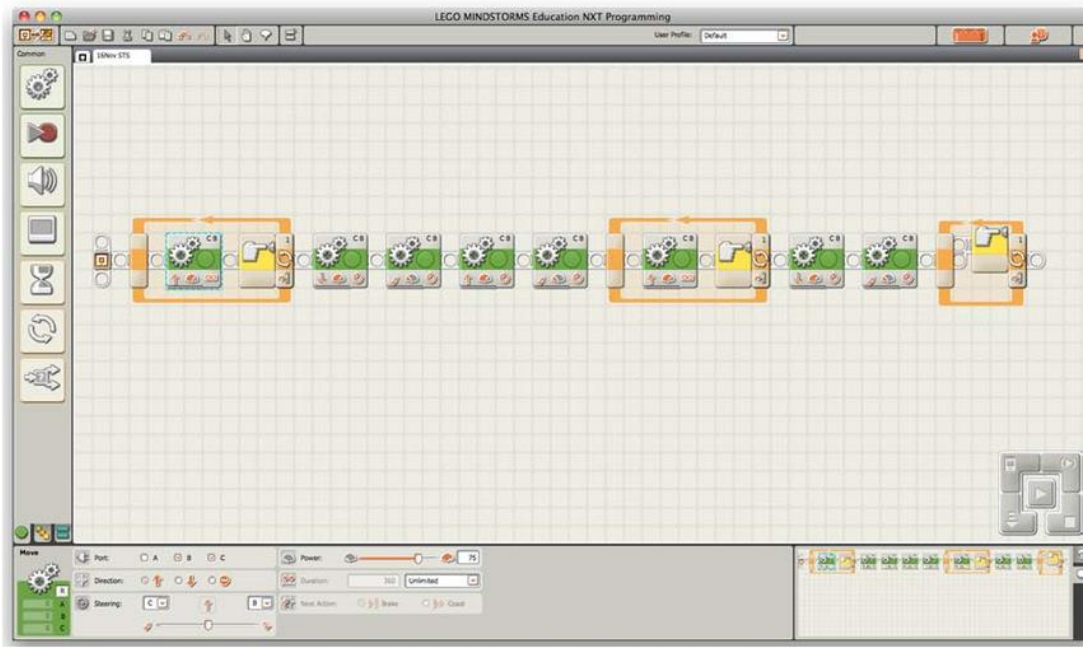
3.2.3 Lego Mindstorms EV3 / NXT Software

Tento software, který byl vyvinutý společností LEGO, je navržený pro ovládání a programování Lego robotických sad Mindstorms pomocí vizuálního programování. Existují dvě hlavní verze:

- **Lego Mindstroms NXT Software**, které umožňuje vytvářen programy pro NXT roboty.
- **Lego Mindstorms EV3 Software,** což je verze pro novější sadu Lego Mindstorms EV3.

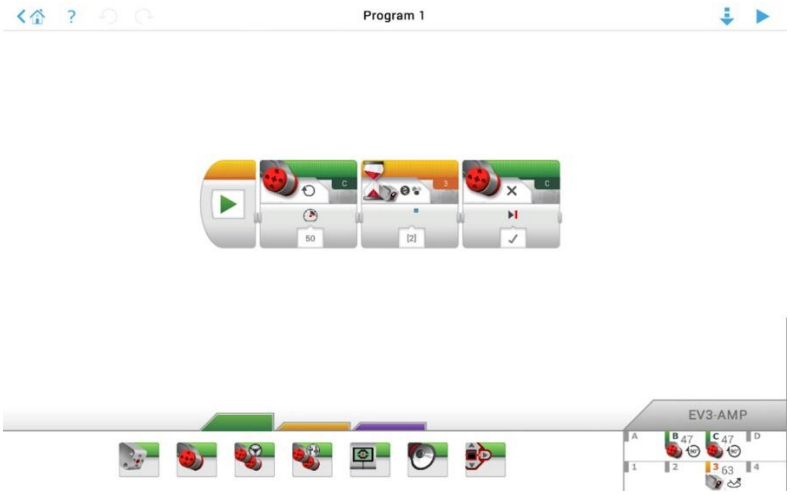
Obě verze softwaru používají VPL založený na principu přetahování bloků do pracovní plochy a jejich propojování pro vytvoření instrukcí pro chování robotů a poskytují uživatelům nástroje k vytváření různých typů robotů, od jednoduchých modelů až po složitější robotické systémy.

Obrázek 4: *Ukázka uživatelského rozhraní Lego Mindstorms NXT Software.*[4]



[4] Obrázek 4: převzato z webu researchgate.net [online]. Dostupné online z: https://www.researchgate.net/figure/Example-of-NXT-robot-program-solution_fig4_258831472

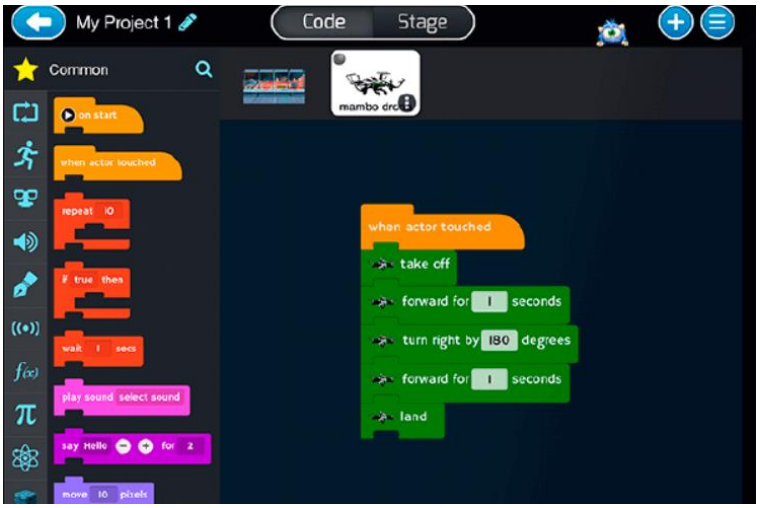
***Obrázek 5**: Ukázka prostředí Lego Mindstorms EV3 Software.*[5]



3.2.4 Tynker

Tynker je platforma určená primárně pro vzdělávání. Skrz svůj design, vzhled a provedení je určená spíše dětem. Jeho vizuální prostředí umožňuje uživatelům vytvářet programy, hry, animace a další aplikace pomocí blokového rozhraní. Tynker je založen na HTML5 a JavaScriptu a lze jej používat v prohlížečích, tabletu, či mobilu. Co se přístupnosti týče, tak se bohužel nejedná o open-source software, nýbrž jde o komerční platformu. V nabídce sice je zdarma verze s omezenými funkcemi, ale pro plnější a rozšířené funkce si musíte pořídit placenou verzi Tynkeru.

***Obrázek 6:** Ukázka uživatelského rozhraní platformy Tynker*[6]



[5] Obrázek 6: převzato z webu chromewebstore.google.com. Dostupné online z: https://chromewebstore.google.com/detail/lego®-mindstorms®-educati/jhnhfnolmcleankdkhfklakpchnccipg?pli=1

[6] Obrázek 7: převzato z webu blog.tcea.org. Dostupné online z: https://blog.tcea.org/computer-literacy-skills/

3.3 Scratch

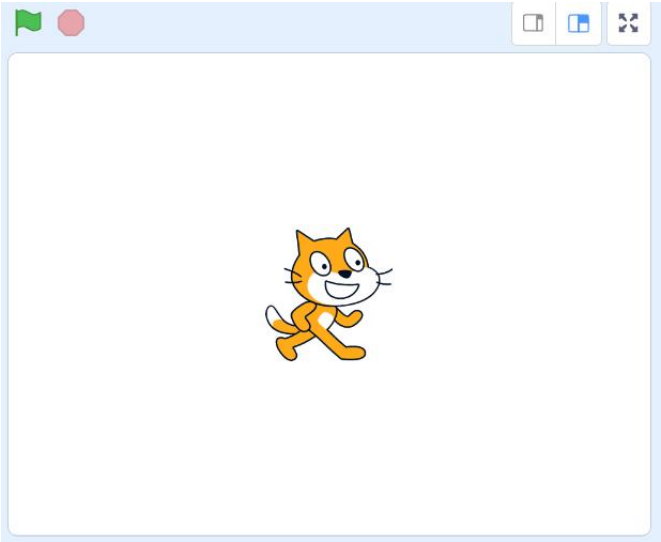
Jedním z předních zástupců vizuálního programování je právě programovací jazyk Scratch. Je to intuitivní prostředek pro vstup do světa programování, navržený primárně pro začátečníky a děti. Byl vytvořen na MIT (Massachusetts Institute of Technology) a v roce 2003 vyšel první prototyp. Scratch umožňuje vytvářet interaktivní příběhy, hry a všemožné animace pomocí přetahování a spojování bloků, které reprezentují programovací příkazy. Jeho vizuální rozhraní a bloková struktura odstraňují potřebu psát kód, což umožňuje uživatelům snadno porozumět základním principům, programování a algoritmům. Jazyk Scratch nejenom podporuje rozvoj programovacích dovedností, ale také kreativitu, logické myšlení a řešení problémů. Díky svému interaktivnímu prostředí umožňuje uživatelům experimentovat s různými nápady a vidět okamžitě výsledky svého úsilí. Tento jazyk se stal oblíbeným nástrojem pro výuku programování ve školách a vzdělávacích institucích po celém světě díky své přístupnosti a schopnosti zaujmout a motivovat nové programátory a tvůrce. Je zdarma všem dostupný na jeho stránkách: https://scratch.mit.edu, kde máte možnost prohlížet si nebo vytvářet nové projekty, které pokud si zde založíte účet, můžete dál sdílet komunitě. Uživatelům je v nabídce i offline verze Scratche, kterou si můžete zdarma stáhnout na svůj desktop.

3.3.1 Uživatelské rozhraní jazyka Scratch

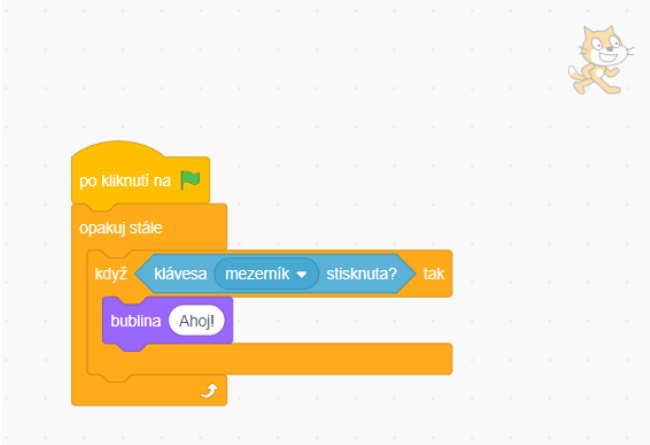
Uživatelské rozhraní Scratche by se dalo rozdělit na tři části:

- První částí je scéna, což je okno, ve kterém probíhá daný program. Většina programů se spouští pomocí zelené vlaječky a pomocí červeného tlačítka vynutíme okamžité zastavení programu (Obrázek 7). V této sekci lze nahrávat, vytvářet, či z volné nabídky Scratche vybírat postavy a pozadí.
- Druhou částí je potom programovací oblast, ve které můžeme manipulovat s bloky a tvořit tak smysluplné scénáře, které jsou konstruktem celého programu (Obrázek 8).
- Třetí částí je paleta bloků, které jsou rozděleny do různých kategorií, dle své funkce (Obrázek 9). Dále jsou zde i editory pro kostýmy a zvuky postavy.

Obrázek 7: *Scéna programu*



Obrázek 8: *Programovací oblast se scénářem*



Obrázek 9: *Paleta bloků*

Následující tabulka popisuje jednotlivé kategorie bloků v programovacím jazyce Scratch:

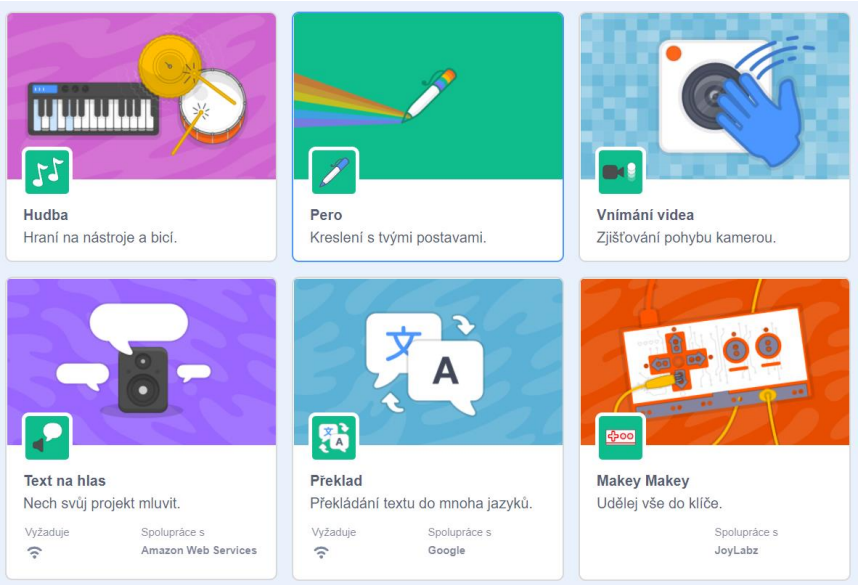
Tabulka 2: *Kategorie bloků ve Scratchi*

Barva	Název kategorie	Popis
	Pohyb	Bloky určené pro pohyb postavy, změna jeho souřadnic a otáčení se.
	Vzhled	Bloky upravující vzhled postavy, změna kostýmů, změna velikosti, komiksové bubliny s textem a změna pozadí.
	Zvuk	Bloky pro přehrávání zvukových souborů.
	Události	Počáteční bloky pro scénáře, které se spustí po určité události (např. po kliknutí na zelenou vlajku).
	Ovládání	Bloky pro cyklus, podmínky a klonování postavy.
	Vnímání	Bloky pro kontakt postavy s okolím, dotýkáš se barvy, jiné postavy, je něco stisknuté? Správa času a otázky.
	Operátory	Aritmetické a logické operátory, základní matematické funkce, generátor náhodných čísel.
	Proměnné	Bloky pro práci s proměnnou a seznamem.
	Moje bloky	Uživatelské procedury.

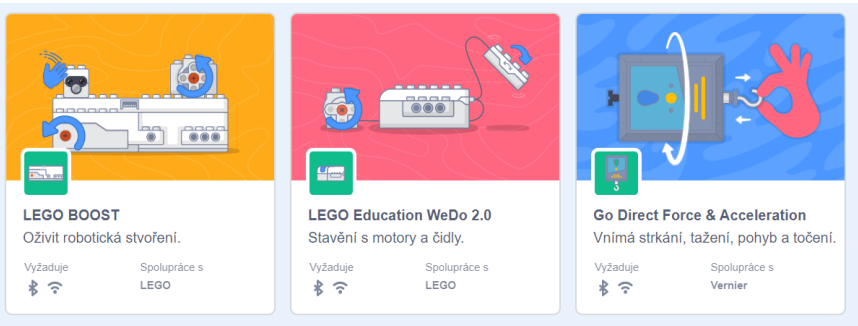
3.3.2 Rozšíření jazyka Scratch

Ve Scratchi jsou k dispozici krom normálních bloků možná rozšíření, které přidávají do programu vlastní bloky s určitou funkcionalitou. Ve verzi Scratch 2.0 byla rozšíření určená pouze pro propojení s nějakým hardwarem. Příkladem je třeba rozšíření LEGO Mindstorms EV3, které kontroluje řízení motorů a přijímá data ze senzorů robota. Ve verzi Scratch 3.0 se však už i přidala softwarová rozšíření, která doplňovala projekt užitečnými funkcemi jako třeba kreslení, hudba, vnímání videa, převod textu na hlas či překladač. Obrázek 10 a Obrázek 11 ukazují možnou nabídku rozšíření jazyka Scratch.

Obrázek 10: *Nabídka rozšíření Scratch*



Obrázek 11: *Nabídka rozšíření Scratch*



3.4 Možnost konverze do vyšších programovacích jazyků

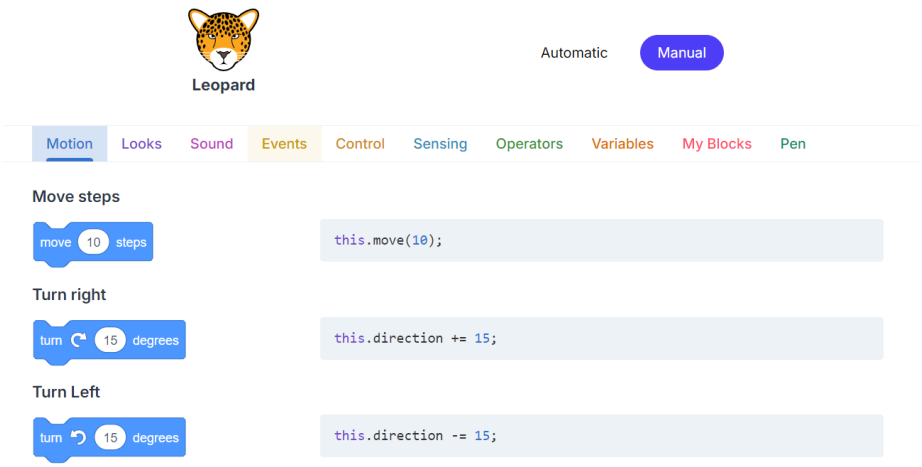
Vyšší programovací jazyk budeme zde vnímat jako programovací jazyk, který využívá tradiční způsob programování v textové podobě kódu. Příkladem mohou být jazyky Java, C, či Python. Vizuální programovací jazyk je dobrá volba pro lidi, kteří s programováním začínají nebo chtějí vytvářet jednoduché projekty bez nutnosti znalosti komplikovaných programovacích konceptů. Každopádně pro více komplexní projekty, které vyžadují větší customizaci a kontrolu kódu, může být vhodnější právě vyšší programovací jazyk.

Proto zde nastává otázka, jestli existuje způsob konverze visuálně zpracovaného kódu do tradiční textové podoby vyšších programovacích jazyků. Odpověď zní ano - pojďme si ukázat nástroje, které tuto konverzi umožňují.

3.4.1 Leopardjs

Leopardjs je nástroj, který umožňuje konvertovat Scratch projekty do programovacího jazyka JavaScript. Funguje jednoduše, stačí nahrát svůj Scratch projekt do okna, které se nachází na hlavní stránce Leopardjs, zde je odkaz: https://leopardjs.com a z vašeho projektu se stane webová stránka, která je již zapsána v textové podobě jazyka JavaScript. Projekty zde můžete vytvářet i manuálně pomocí krásně zpracovaného návodu, který ukazuje, jak zapsat vizuální bloky v programovacím jazyce JavaScript (Obrázek 12).

Obrázek 12: *Ukázka manuálního převodu bloků ze Scratche do JavaScriptu*[7]

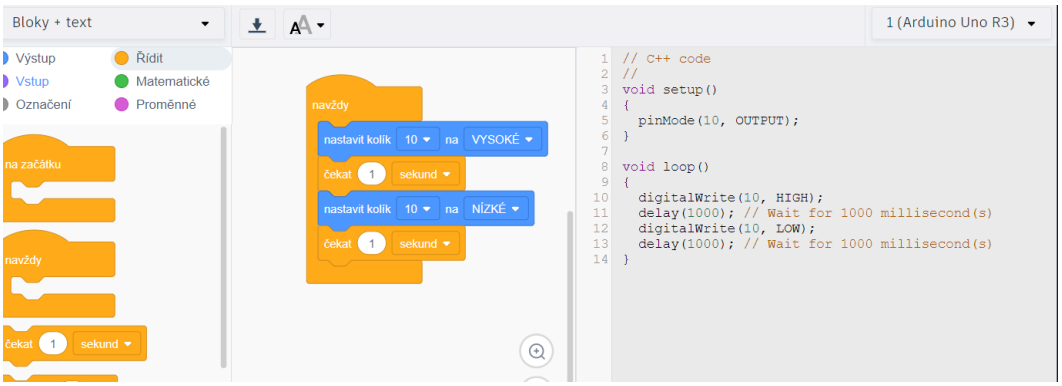


[7] Obrázek 13: Dostupný online z: https://leopardjs.com

3.4.2 TinkerCAD

Dalším nástrojem je webová aplikace TinkerCAD, která poskytuje vizuální prostředí pro 3D modelování, elektronický design a tvorbu prototypů. Tato webová aplikace je často využívaná i ve vzdělávání v oblasti robotiky. Existuje tu totiž simulátor elektronických obvodů. Uživatel si tak může naprogramovat senzor, diodu, či funkci motorů pro robota a pomocí virtuální simulace vidět jejich funkcionalitu. Programování se zde provádí blokově, za využití jazyka Scratch. Je tu však i možnost převést jazyk na textovou podobu programovacího jazyka C++ (Obrázek 13).

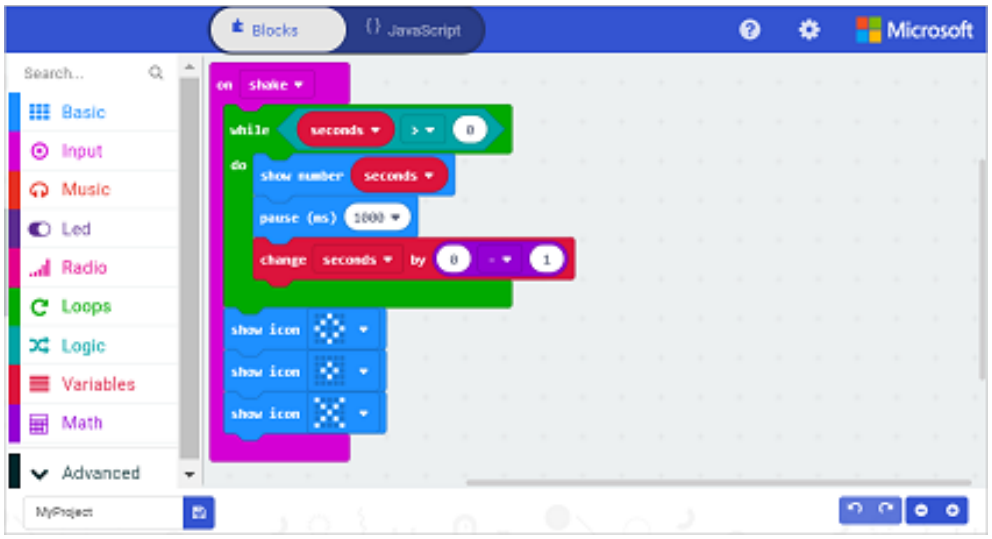
Obrázek 13: *Ukázka konvertace jazyka Scratch do C++, TinkerCAD*



3.4.3 Microsoft MakeCode

MakeCode je open-source interaktivní vizuální platforma vytvořena od společnosti Microsoft. Slouží primárně pro vzdělávání v oblasti programování a elektroniky. Jeho cílem je poskytnout jednoduché prostředí pro výuku a tvorbu elektronických zařízení, her, robotů a dalších interaktivních aplikací. Podporuje několik různých technologií, včetně mikrokontrolérů jako je Micro:bit, LEGO Mindstorms, Arduino a dalších. Pro větší komplexnost projektů tu existuje i jazykový editor, který právě převádí vizuální bloky do textové podoby jazyka JavaScript, ve kterém můžete řešit komplexnější funkce (Obrázek 15).

Obrázek 14: *MakeCode, ukázka blokového programování*[8]



[8] Obrázek 14: převzato z microsoft.com. Dostupné online z: https://makecode.com/about

Obrázek 15: *Jazykový editor v MakeCode, převod bloků do JavaScriptu*[9]

3.4.4 Blockly Code editor

Blockly, který byl již zmíněn výše, poskytuje i nástroj pro konverzi do vyšších programovacích jazyků. Stačí vytvořit nebo exportovat projekt v Blockly a editor vám nabídne možnost konverze do jazyka JavaScript, Python, PHP, Lua, Dart, XML a JSON. Editor naleznete na stránce: https://blockly-demo.appspot.com/static/demos/code/index.html. V Obrázek 16 můžete vidět konverzi z vizuálního programovacího jazyka Blockly do Pythonu.

Obrázek 16: *Konverze z jazyka Blockly do Pythonu*[10]

[9] Obrázek 15: převzato z microsoft.com. Dostupné online z: https://makecode.com/about
[10] Obrázek 17: Dostupné online z: https://blockly-demo.appspot.com/static/demos/code/index.html

4 PRAKTICKÁ ČÁST BAKALÁŘSKÉ PRÁCE

V této části mé bakalářské práce vytvořím sadu úloh v programovacím jazyce Scratch. Sada bude obsahovat celkem 21 úloh v prostředí programovacího jazyka Scratch. Všech 18 úloh bude k dispozici s podrobným popisem zadání, řešením a případným doplňujícím úkolem. Zbylé tři úlohy slouží pouze jako motivační materiál a jsou tedy bez popisu a řešení. Úlohy budou mít za cíl, aby si žák osvojil základní programovací kompetence a dostatečně se tak připravil na programování více komplexních jazyků jako je Python, Java, C# apod. Tato sada je určená pro druhý stupeň ZŠ, tedy pro žáky 5-9 tříd. Sadou úloh bude žáky provázet postavička se jménem „Bajtík". Jedná se o animovaného netopýra a jeho jméno je odvozené od pojmu „Bajt", což je v informatice základní jednotka kapacity počítačové paměti. Ke každé úloze bude taky k dispozici metodický list, který bude sloužit jako pomůcka pro pedagogy.

4.1 Rozdělení úloh

Úlohy budou rozděleny do tří kategorií. V každé kategorii si žák osvojí určité znalosti a bude tak rozvíjet svou znalost nejen v jazyce Scratch, ale taky obecně v programování. Tyto kategorie se budou od sebe odlišovat obtížností, která je uzpůsobená schopnostem žáka. Kategorie na sebe navazují, a to tím způsobem, že v obtížnostně těžší kategorii budou použity znalosti nabité z předchozí kategorie. Každá kategorie bude obsahovat celkem pět úloh + jednu závěrečnou úlohu ve formě většího projektu, který bude sumarizovat danou kategorii a bude tam použito vše, co se žák naučil. Výjimkou je akorát poslední kategorie „Profík", ve které jsou všechny úlohy koncipovány jako „větší projekt". Nyní popíšu jednotlivé kategorie a jejich zaměření.

Kategorie Elév:

V této části jsou úlohy koncipovány tak, aby se žák zejména seznámil s prostředím programovacího jazyka Scratch. Pracuje se zde převážně s jednoduchými cykly, kostýmy, pozadím a zvuky. Žáci se i naučí pracovat s rozšířením pero, které je v rámci Scratche volně přístupné. Řešení takových úloh vychází časově na cca 30 minut.

Kategorie Kadet:

Kategorie Kadet se věnuje složitějším cyklům, podmínkám, práce s proměnou a seznamem. Budou i využity dříve naučené znalosti z kategorie Elév. Řešení takových úloh vychází časově na cca 45 minut, tedy jednu vyučovací jednotku.

Kategorie Profík:

V poslední kategorii Profík bude šest větších projektů, kde budou žáci využívat vše, co se naučili v předchozích kategoriích, a navíc si osvojí práci s klonem. Projekty budou většinou koncipovány ve formě vytvoření interaktivní hry. Řešení takových projektů zabere žákům zhruba 90 minut a mohou sloužit právě jako větší projekty v rámci více výukových jednotek.

5 ÚLOHY KATEGORIE ELÉV

5.1 Létající Bajtík

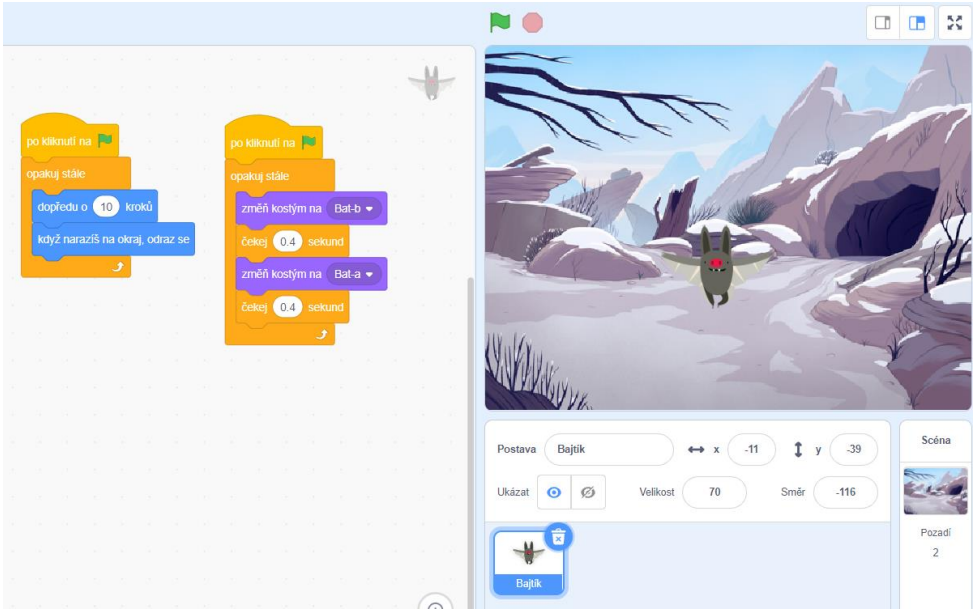
Tabulka 3: *Metodický list - Létající Bajtík*

Metodický list	
Název materiálu:	Letající Bajtík
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	30 minut
Rozsah:	1 pozadí, 1 postava, 2 scénáře, 10 bloků
Klíčová slova:	animace, pohyb, kostým, postava
Anotace materiálu:	Žák vytvoří animaci pohybu postavy Bajtík.

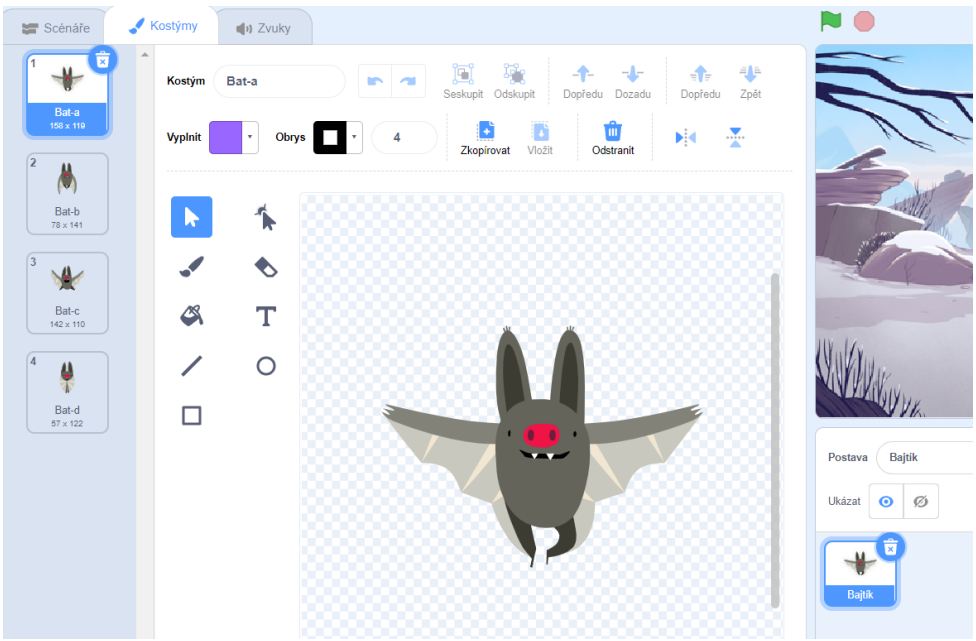
Zadání: Rozpohybuj Bajtíka, aby náhodně létal po celé obrazovce a vytvoř mu animaci, která bude simulovat mávání jeho křídel.

Řešení: Žák bude mít k dispozici kostým postavy Bajtík a pozadí. Postava bude měnit dva kostýmy pomocí cyklu „**opakuj stále**“. Poté v druhém cyklu „**opakuj stále**“ nastavíme, aby se postava posouvala **dopředu o 10 kroků** a nastavíme vlastnost, aby se **po nárazu na okraj odrazila** (Obrázek 17).

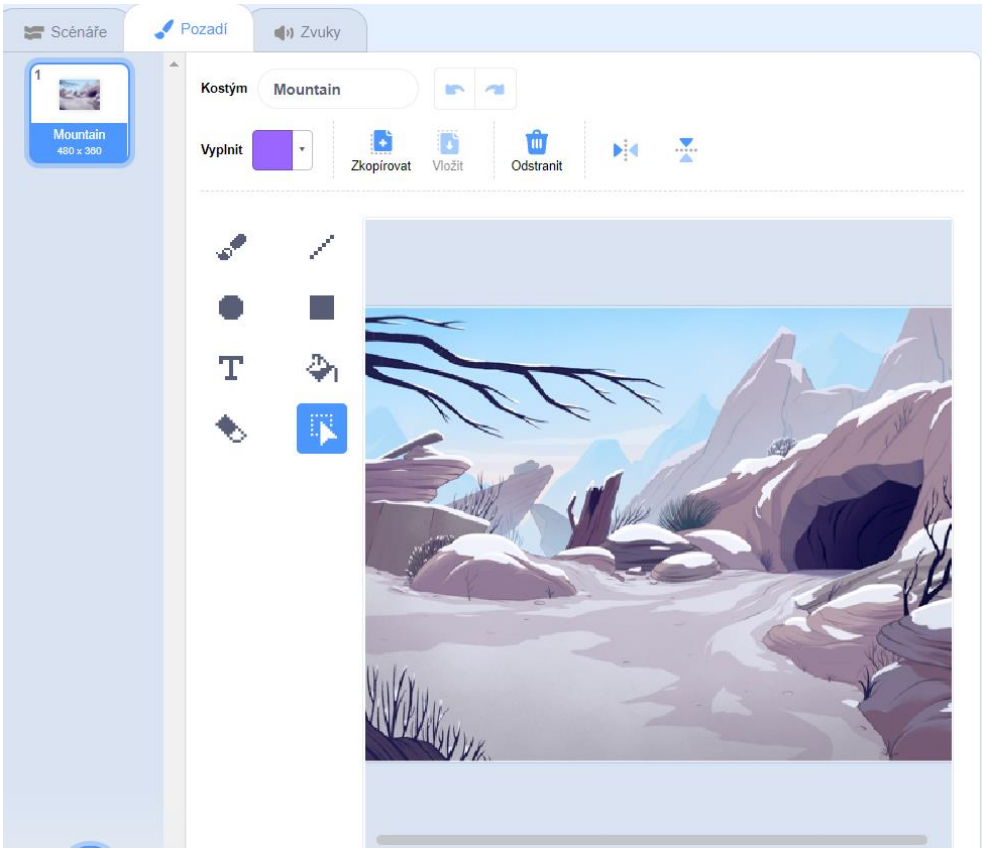
Obrázek 17 *Kód Bajtík*



Obrázek 18 *Postava Bajtík*



Obrázek 19 *Pozadí*



Doplňující úkoly: Změň nebo přidej nové pozadí. Přidej novou postavu a pokus se o animaci pohybu této postavy.

5.2 Pohyb pomocí tlačítek

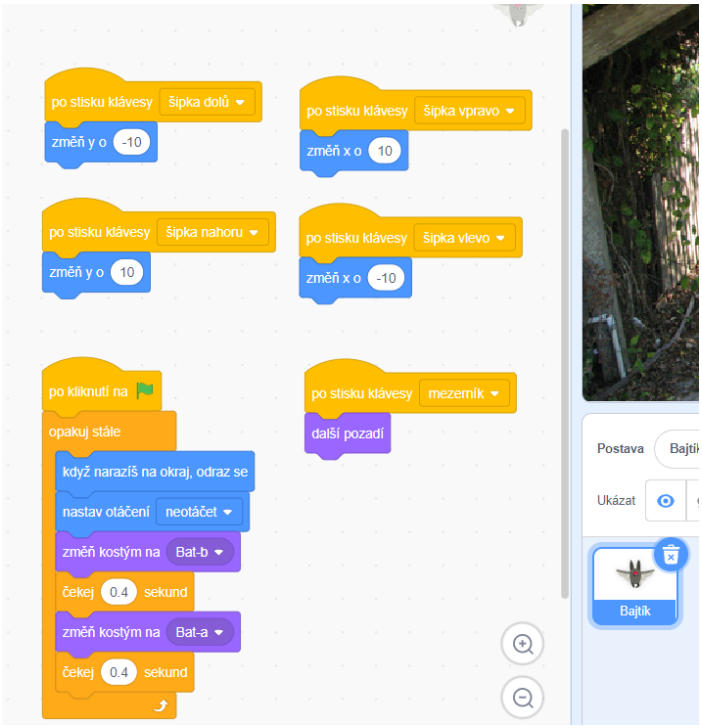
Tabulka 4: *Metodický list - Pohyb pomocí tlačítek*

Metodický list	
Název materiálu:	Pohyb pomocí tlačítek
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	30 minut
Rozsah:	x pozadí, 1 postava, 6 scénářů, 18 bloků
Klíčová slova:	animace, pohyb, tlačítko, pozadí
Anotace materiálu:	Žák naprogramuje pohyb postavy „Bajtík“ pomocí tlačítek a animaci mávání křídel.

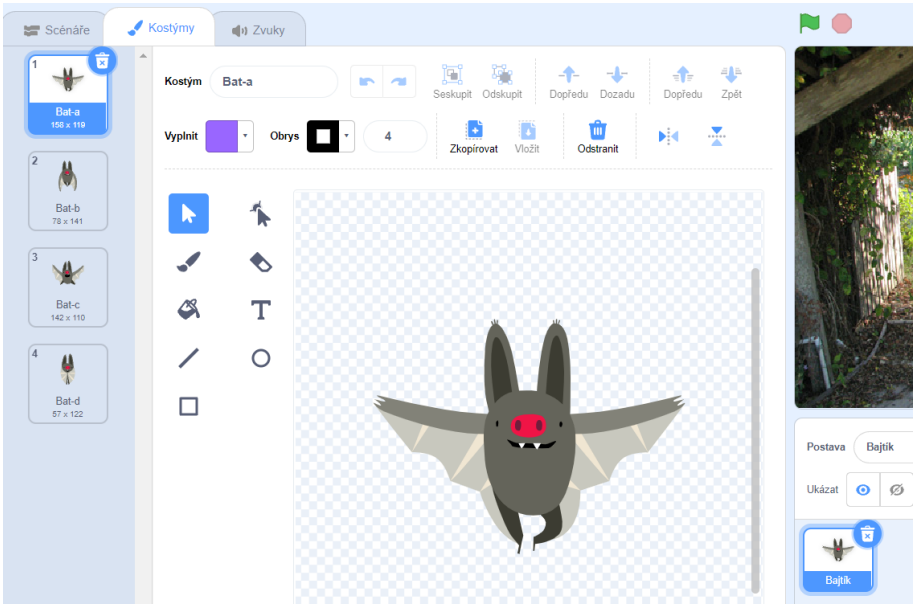
Zadání úlohy: Naprogramuj pohyb Bajtíka pomocí tlačítek na klávesnici, aby mohl létat vpravo, vlevo, nahoru a dolů. Zamez tomu, aby se při změně směru otáčel a zároveň, aby se při nárazu na okraj odrazil. Nakonec rozpohybuj Bajtíka pomocí jednoduché animace mávání křídel.

Řešení úlohy: Pomocí bloků „**po stisku klávesy**“ nastavíme pohyb postavy a přepnutí pozadí. Je dobré seznámit žáky se souřadnicemi. Souřadnice X určuje horizontální pozici postavy (zleva doprava). Souřadnice Y určuje vertikální pozici postavy (nahoru dolů). V hlavním cyklu „**opakuj stále**“ přidáme „**nastav otočení – neotáčet**“ a „**když narazíš na okraj, odraz se**“. Poté už jen nastavíme dříve naučenou animaci pohybu (Obrázek 20).

Obrázek 20: *Kód Bajtík*



Obrázek 21: *Kostým Bajtík*



Doplňující úkoly: Přidej do programu více pozadí a naprogramuj, aby se po stisku klávesy mezerník pozadí změnilo.

5.3 Divadlo

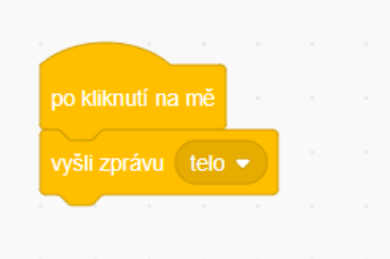
Tabulka 5: *Metodický list - Divadlo*

Metodický list	
Název materiálu:	Divadlo
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 4 postavy, 4 scénářů, 8 bloků
Klíčová slova:	tlačítko, vyslání zprávy, obdržení zprávy, kliknutí na tlačítko
Anotace materiálu:	Žák vytvoří přehlídku kostýmů a pomocí dvou tlačítek bude měnit u postav tělo nebo hlavu.

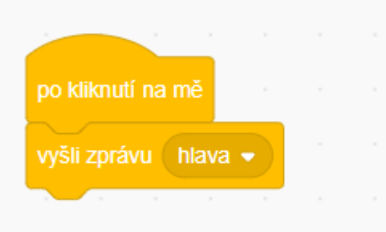
Zadání úlohy: Vytvoř divadlo z kostýmů zvířátek. Vyber si několik zvířecích postav z dostupné nabídky Scratche a uprav si v editoru zvířata tak, abys oddělil hlavu a tělo. Poté naprogramuj postavy „**Tlačítko Hlava**“ a „**Tlačítko Tělo**“, aby se po kliknutí na ně vyslala zpráva, pomocí které se bude měnit příslušná část zvířete (hlava nebo tělo).

Řešení úlohy: Žáci se seznámí s novým blokem „**vyšli zprávu**“. Pomocí něj dokážou po kliknutí na tlačítko vyslat zprávu pro vykonání určité události. V tomto případě, pokud bude kliknuto např. na tlačítko „**Hlava**“ – vyšle se zpráva a v kódu postavy „**Hlava**“ použijeme blok „**po obdržení zprávy**“ a změníme kostým postavy.

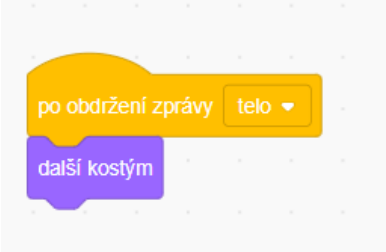
Obrázek 22: *Kód Tlačítko Tělo*



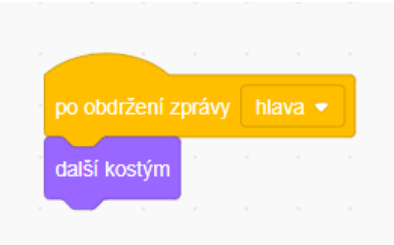
Obrázek 23: *Kód Tlačítko hlava*



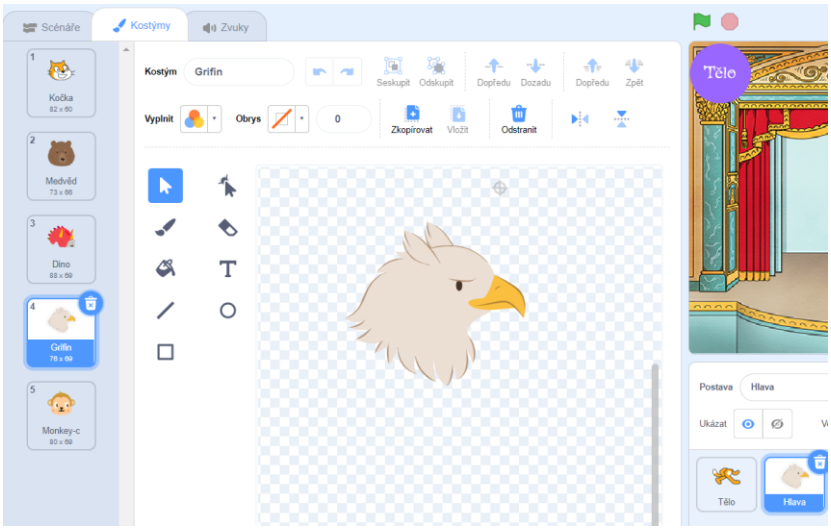
Obrázek 24: *Kód Tělo*



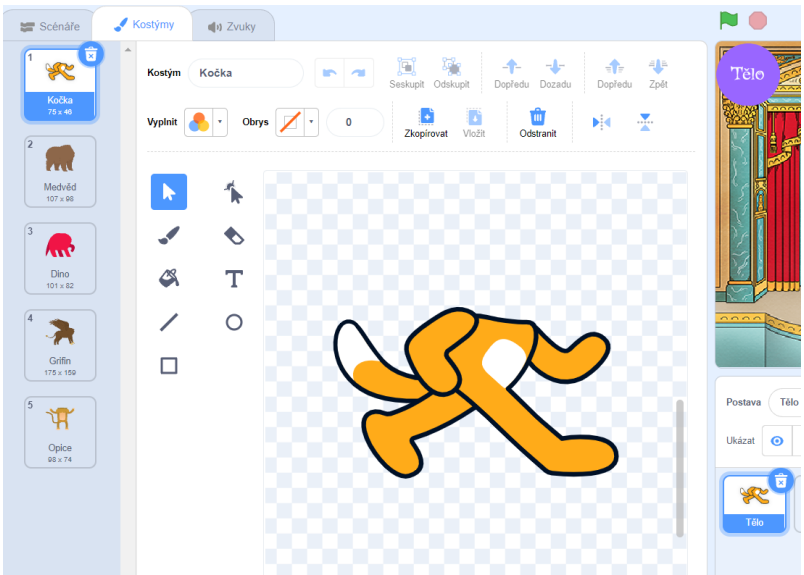
Obrázek 25: *Kód Hlava*



Obrázek 26: Kostýmy Hlava



Obrázek 27: *Kostýmy Tělo*



Doplňující úkoly: Přidej více postav. Zkus změnit hudbu, která hraje v pozadí. Zkus si vytvořit a naprogramovat vlastní tlačítko, které bude například po kliknutí měnit pozadí.

5.4 Mandala

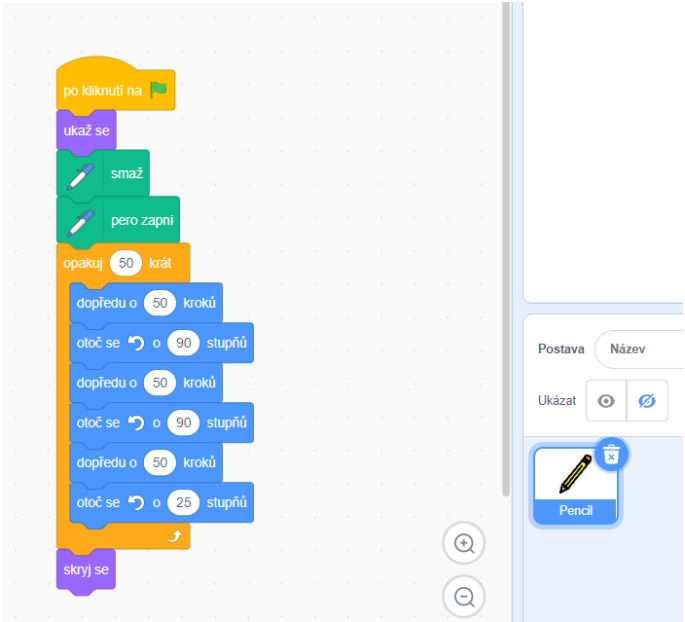
Tabulka 6: *Metodický list - Mandala*

Metodický list	
Název materiálu:	Mandala
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	20 minut
Rozsah:	1 pozadí, 1 postava, 1 scénář, 12 bloků
Klíčová slova:	cyklus, pero, otáčení, stupně, pohyb
Anotace materiálu:	Žák naprogramuje mandalu pomocí rozšíření pero a za využití cyklu.

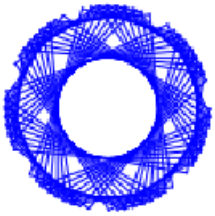
Zadání úlohy: Pokus se nakreslit mandalu pomocí rozšíření jazyka Scratch – Pero a za využití cyklu naprogramuj pohyb pera s určitým vzorem.

Řešení úlohy: Žáci budou seznámeni s rozšířením **Pero**. Za využití bloku „**pero zapni**“ se zvolená barva otiskne na pozadí podle aktuální souřadnice postavy. Mandalu nakreslíme pomocí cyklu „**opakuj x krát**“, kde budeme využívat bloky „**dopředu o x kroků**“ a „**otoč se o x stupňů**". Žáci budou mít svobodu v tom, jaké parametry si nastaví (každému se může vykreslit něco jiného). V Obrázek 28 a Obrázek 29 je kód a výstup programu.

Obrázek 28: *Kód pro vytvoření Mandaly*



Obrázek 29: *Výsledná Mandala*



Doplňující úkoly: Zkus upravit počet opakování nebo stupeň otáčení se, aby si vytvořil jiný zajímavý obrázek.

5.5 DJ pult

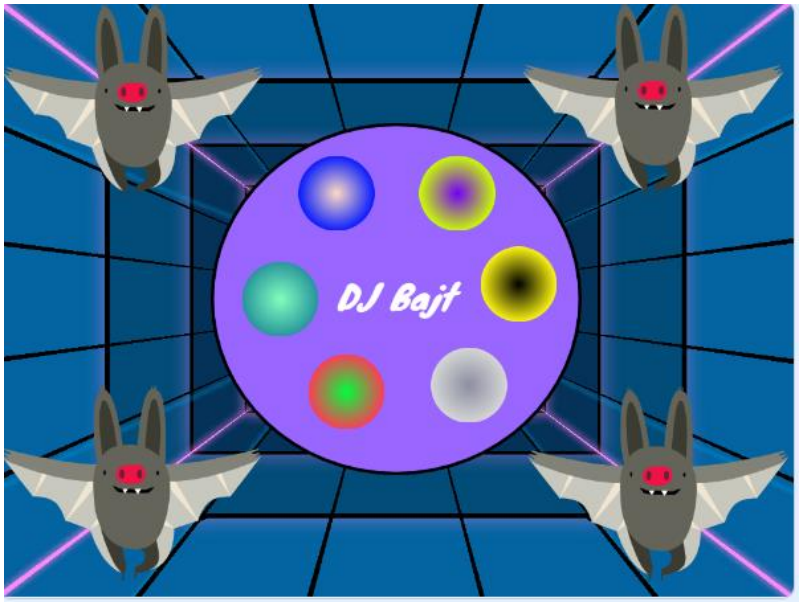
Tabulka 7: *Metodický list - DJ pult*

Metodický list	
Název materiálu:	DJ Pult
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 6 postav, 6 scénářů, 12 bloků
Klíčová slova:	tlačítko, kliknutí na tlačítko, zvuk
Anotace materiálu:	Žák vytvoří šest postav, které si upraví jako tlačítko. Každému tlačítko přidá i zvířecí zvuk, který se přehraje po kliknutí na dané tlačítko.

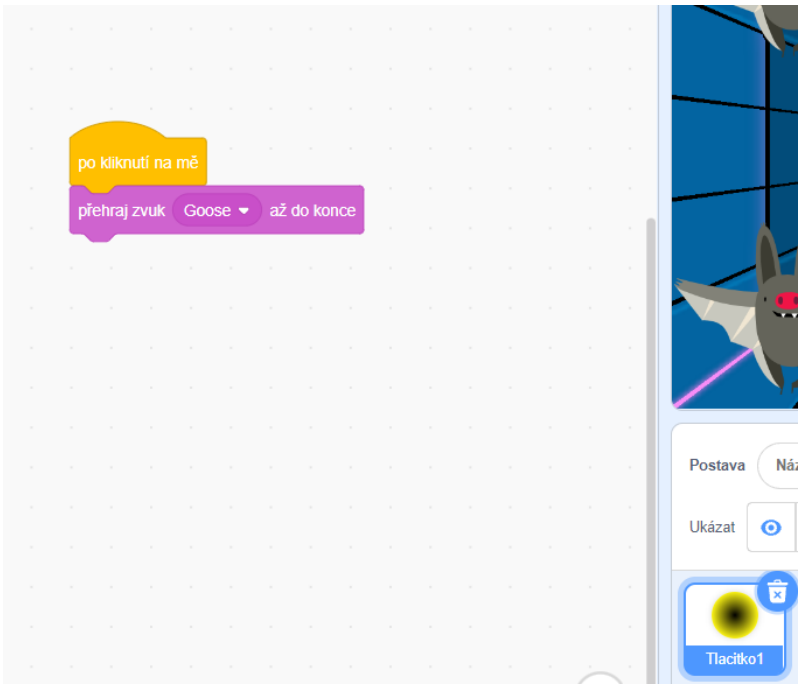
Zadání úlohy: Pomoz DJ Bajtíkovi vytvořit tu největší zvířecí diskotéku. Vytvoř šest postav, které budou sloužit jako tlačítka. Přidej je do DJ pultu a naprogramuj tlačítka tak, aby se po kliknutí na ně přehrál tebou zvolený zvuk zvířete.

Řešení: Žáci si primárně nacvičí úpravu kostýmů. Vytvoří si svůj DJ pult se šesti tlačítky. Využijeme bloku „**Po kliknutí na mě**“, který provede přehrání zvoleného zvuku – žáci si mohou vyhrát s volbou zvuků. V obrázcích níže je ukázka vytvořeného DJ pultu.

Obrázek 30: *6 vytvořených tlačítek, které jsou umístěné na DJ Pultu*



Obrázek 31: *Kód pro tlačítko*



Doplňující úkoly: Přidej a naprogramuj více tlačítek. Zkus změnit hudbu v pozadí.

5.6 Malování

Tabulka 8: *Metodický list - Malování*

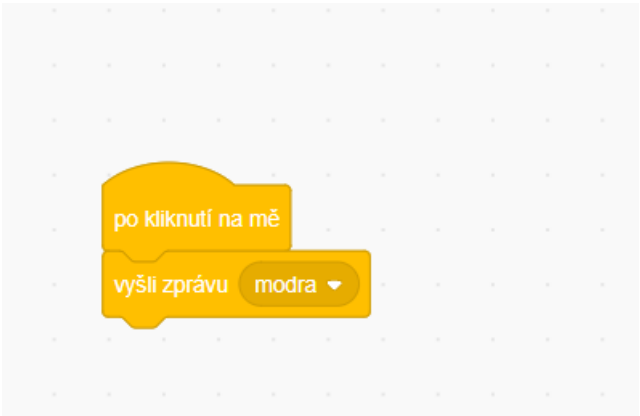
Metodický list	
Název materiálu:	Malování
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	60 minut
Rozsah:	1 pozadí, 5 postav, 11 scénářů, 51 bloků
Klíčová slova:	Pero, Kreslení, Cyklus, Podmínka, Tlačítko, Barva
Anotace materiálu:	Žák naprogramuje ve Scratchi primitivní program malování.

Zadání úlohy: Vytvoř primitivní program Malování. Na předem vytvořenou paletu přidej sebou zvolené barvičky. Naprogramuj barvičky, aby se po kliknutí na danou barvu změnila barva pera. Pero naprogramuj, aby kreslilo, dokud bude stisknuta myš.

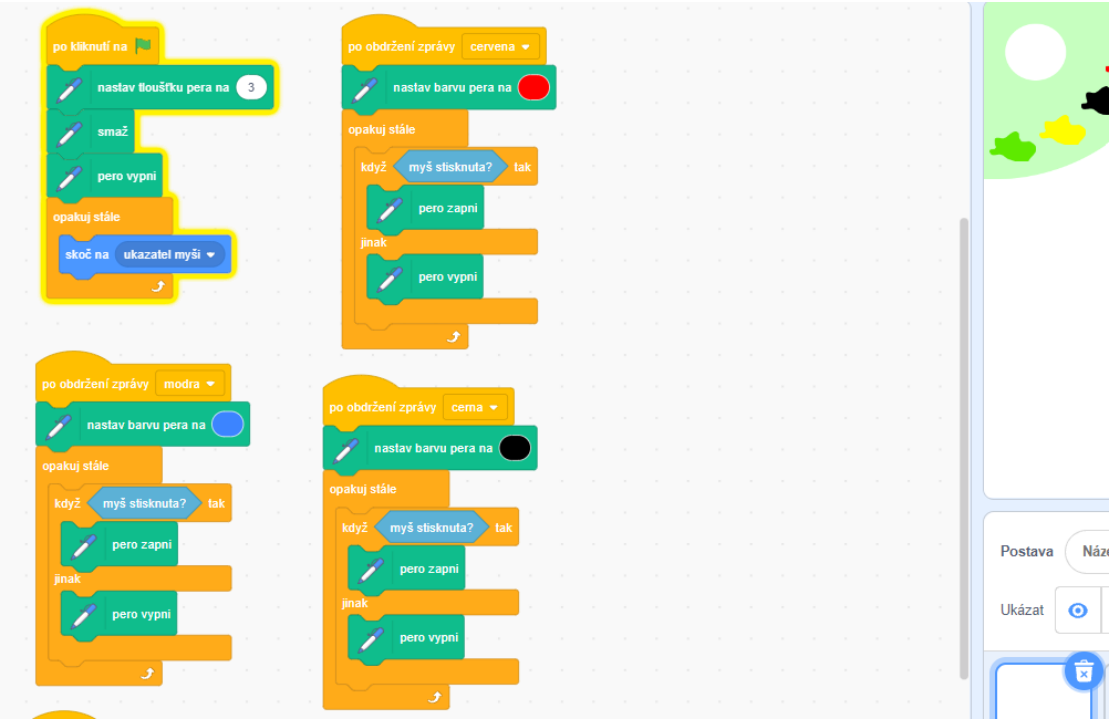
Řešení úlohy: Na začátku projektu bude mít žák k dispozici paletu, která je součástí pozadí (Obrázek 32**:** Přidané barvičky na paletě) a celkem dvě postavy - „**Tuzka**“ a „**Cerna**“. Postava „**Cerna**“ reprezentuje černou skvrnu na paletě. Po kliknutí na skvrnu se vyšle zpráva. (Obrázek 33) Postava „**Tuzka**“ po obdržení zprávy změní barvu pera na danou barvu. Poté pomocí cyklu „**opakuj stále**“ nastaví, aby pero kreslilo po dobu, co uživatel má stisknutou myš. (Obrázek 34) Tímto způsobem žák přidá a naprogramuje i ostatní barvičky.

Obrázek 32: *Přidané barvičky na paletě*

Obrázek 33: *Kód pro barvu Modra*



Obrázek 34: *Kód postavy Tuzka*



Doplňující úkoly: Zkus vytvořit a naprogramovat tlačítko, které bude upravovat tloušťku pera.

6 ÚLOHY KATEGORIE KADET

6.1 Kostka

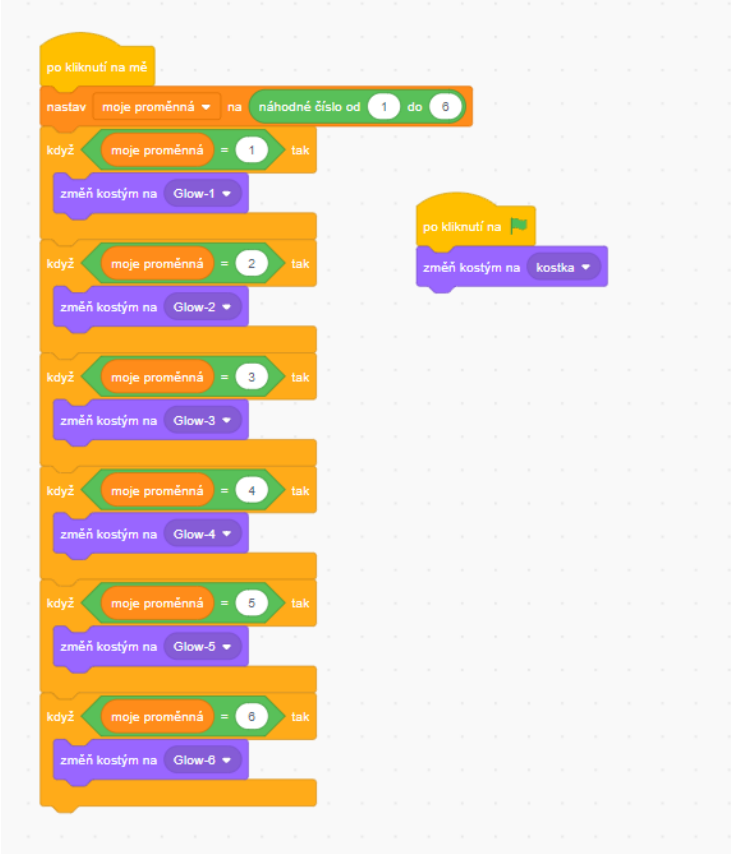
Tabulka 9: *Metodický list - Kostka*

Metodický list	
Název materiálu:	Kostka
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 1 postava, 2 scénáře, 22 bloků
Klíčová slova:	kostka, náhodný generátor čísel, proměnná, podmínka, cyklus
Anotace materiálu:	Žák vytvoří kostku, která bude generovat náhodná čísla od jedné do šesti.

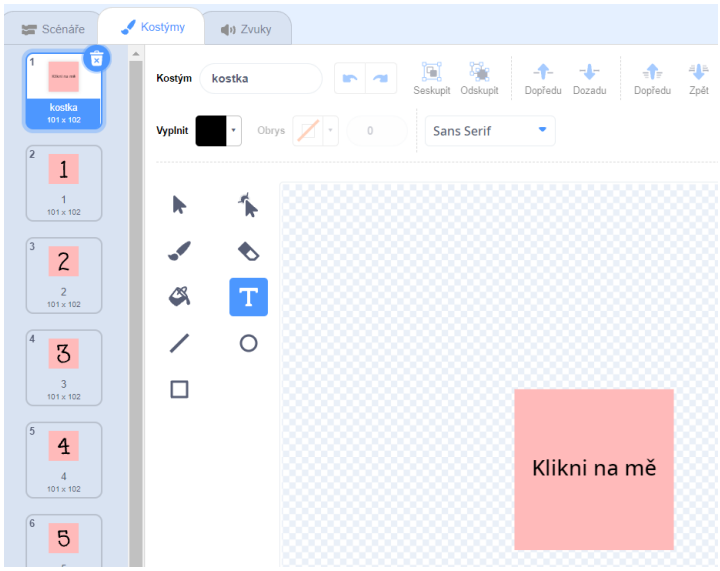
Zadání: Vytvoř kostku, která bude po kliknutí generovat čísla od jedné do šesti.

Řešení: V této úloze se žáci seznámí s **proměnnou**, **podmínkou** a blokem „**náhodné číslo od x do y**“. Je potřeba vysvětlit, jak proměnná funguje. Doporučuji vysvětlit jako krabici, do které něco vložíme/uložíme. V tomhle případě budeme do proměnné ukládat pomocí generátoru náhodných čísel, číslo od jedné do šesti. Poté pomocí podmínky budeme ověřovat jaké číslo je v proměnné uložené. Pokud např. padne číslo šest, změníme kostým kostky, aby se zobrazila šestka. V Obrázek 35 je ukázka kódu.

Obrázek 35: *Kód Kostka*



Obrázek 36: *Kostýmy*



6.2 Numerická paměť

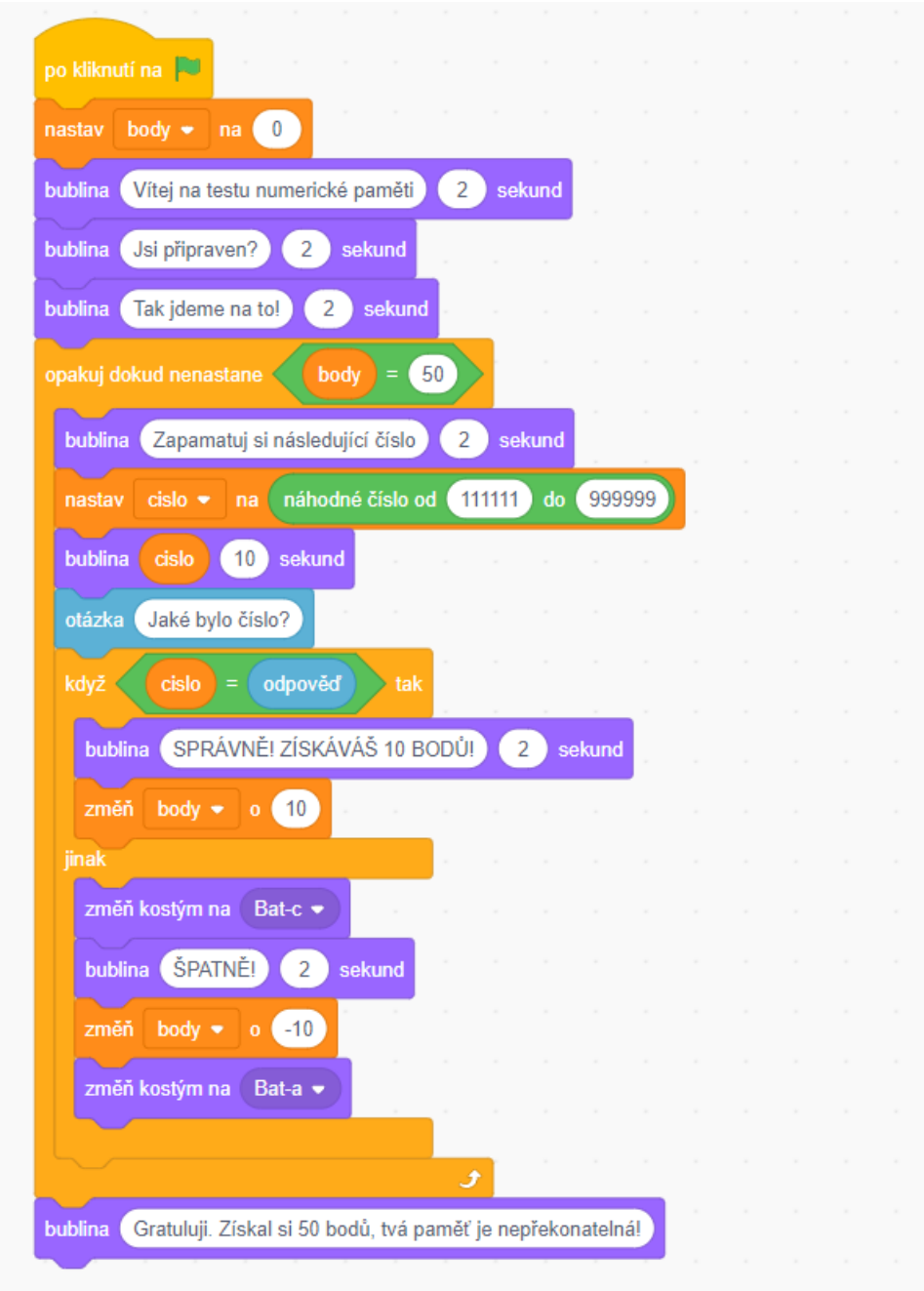
Tabulka 10: *Metodický list - Numerická paměť*

Metodický list	
Název materiálu:	Numerická paměť
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 1 postava, 1 scénář, 18 bloků
Klíčová slova:	náhodný generátor čísel, proměnná, podmínka, cyklus, otázka
Anotace materiálu:	Žák vytvoří test numerické paměti, kde bude generovat náhodná vícerociferná čísla a poté uživateli podá otázku, zdali si pamatuje vygenerované číslo.

Zadání: Vytvoř test numerické paměti. Vygeneruj šesticiferné číslo, které po 10 sekundách zmizí. Poté se uživatele zeptáš, jaké bylo číslo. Pokud uživatel odpoví správně, získá 10 bodů. Pokud špatně, ztrácí 10 bodů. Jakmile uživatel získá bodů 50, ukonči program.

Řešení: Vytvoříme proměnnou „**body**“ a proměnnou „**cislo**“. Hodnotu proměnné „**body**“ budeme buď zvyšovat nebo snižovat dle správnosti odpovědi. Do cyklu „**opakuj dokud nenastane**“ vložíme podmínku, aby se cyklus opakoval, dokud proměnná „**body**“ nebude mít hodnotu 50. V samotném cyklu potom vygenerujeme dané číslo, uložíme ho do proměnné „**cislo**“ a vypíšeme jej uživateli po 10 sekund. Poté číslo zmizí a zeptáme se pomocí bloku „**otázka**“ jaké bylo číslo. Co uživatel napíše se uloží do bloku „**odpověď**“. Srovnáme odpověď s proměnnou „**cislo**“ a pokud se rovná – přičteme 10 bodů, pokud ne – ubereme 10 bodů. V Obrázek 37 můžete vidět řešení.

Obrázek 37: *Kód testu numerické paměti*



Doplňující úkoly: Zkus naprogramovat, aby se cifra vygenerovaného čísla postupně navyšovala.

6.3 Sběr banánů

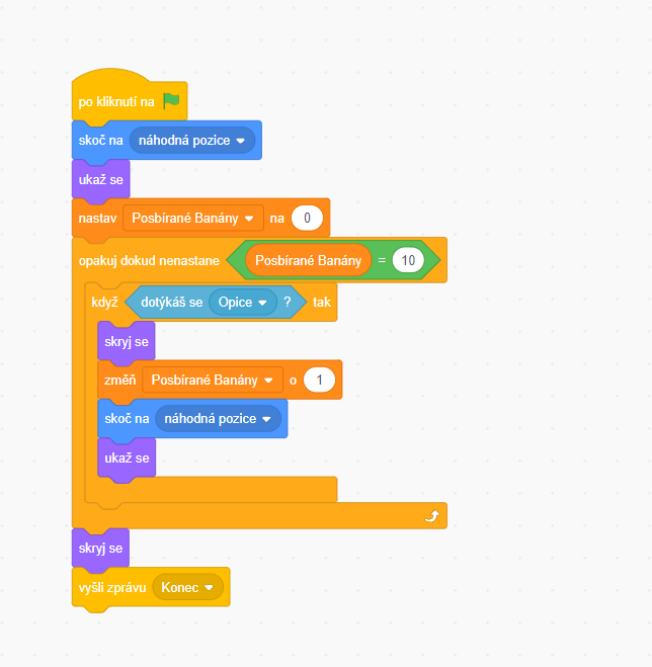
Tabulka 11: *Metodický list - Sběr banánů*

Metodický list	
Název materiálu:	Sběr banánů
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 2 postavy, 6 scénářů, 27 bloků
Klíčová slova:	podmínka, cyklus, dotyk postavy, proměnná, pohyb postavy
Anotace materiálu:	Žák naprogramuje opičku, která bude mít za úkol sbírat banány. Bude zapotřebí naprogramovat i postavu banány, která se bude objevovat a mizet dle potřeby.

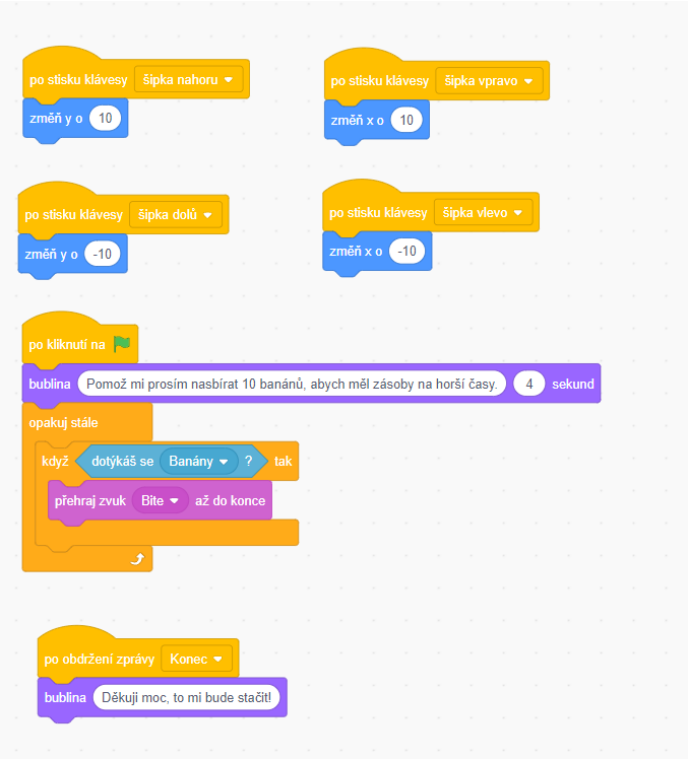
Zadání: Naprogramuj hru, kde opička bude mít za úkol nasbírat 10 banánů. Po spuštění hry se vždy na náhodné pozici zobrazí banán. Jakmile se postava opičky dotkne banánu, banán zmizí a znovu se objeví na jiné náhodné pozici. Až opička nasbírá 10 banánů, ukonči hru.

Řešení: Postava Opice bude mít v sobě kód pro pohyb postavy pomocí bloků „**po stisku klávesy**“. Poté je potřeba přidat cyklus „**opakuj stále**“, kde se pomocí podmínky bude ověřovat, jestli se postava dotýká postavy „**Banány**“ (Obrázek 39). V postavě „**Banány**“ vytvoříme proměnnou „**Posbírané Banány**“, do které budeme ukládat počet posbíraných banánů. Pomocí cyklu „**opakuj dokud nenastane**“, nastavíme, aby se cyklus opakoval, dokud proměnná nebude mít hodnotu 10. V samotném cyklu je potom podmínka „ **když dotýkáš se Opice**“, - „**změň Posbírané Banány o 1**“, skryj postavu „**Banány**“ a přemísti na náhodnou pozici (Obrázek 38).

Obrázek 38: *Kód Banány*



Obrázek 39: *Kód Opice*



Doplňující úkoly: Ozvuč postavy, přidej efekty, pohraj si s pozadím. Pokus se o to, aby se na obrazovce objevilo zároveň více banánů.

6.4 Dostihy

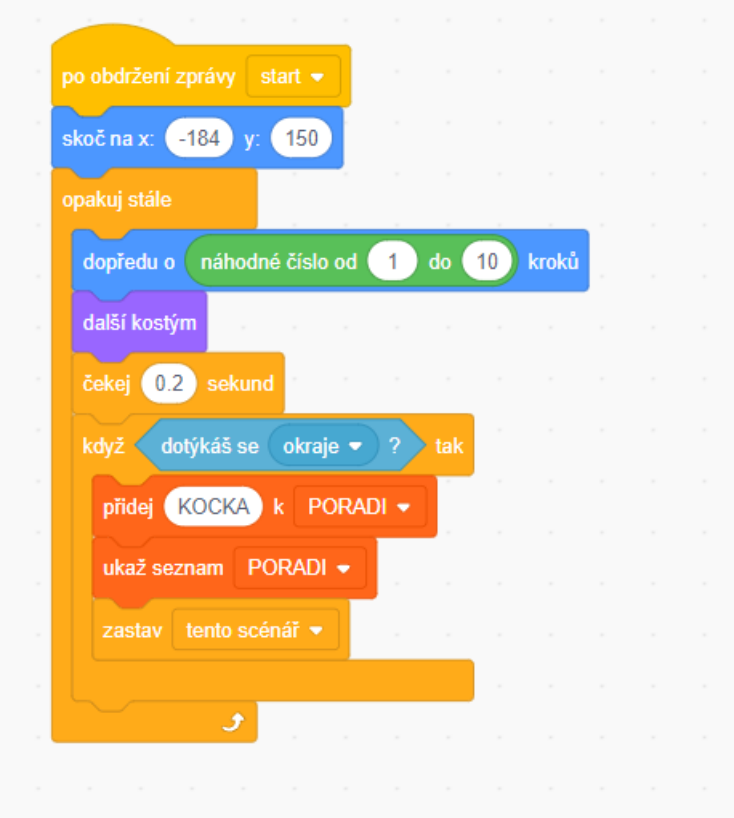
Tabulka 12: *Metodický list - Dostihy*

Metodický list	
Název materiálu:	Sběr banánů
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 5 postav, 5 scénářů, 50 bloků
Klíčová slova:	podmínka, cyklus, dotyk postavy, proměnná, seznam, náhodný generátor čísel
Anotace materiálu:	Žák naprogramuje dostihy. Vytvoří 5 postav a bude upravovat jejich rychlost pomocí náhodného generátoru čísel. Jaká postava se první dotkne cíle, zvítězí. Pořadí se potom vypíše ve vytvořeném seznamu.

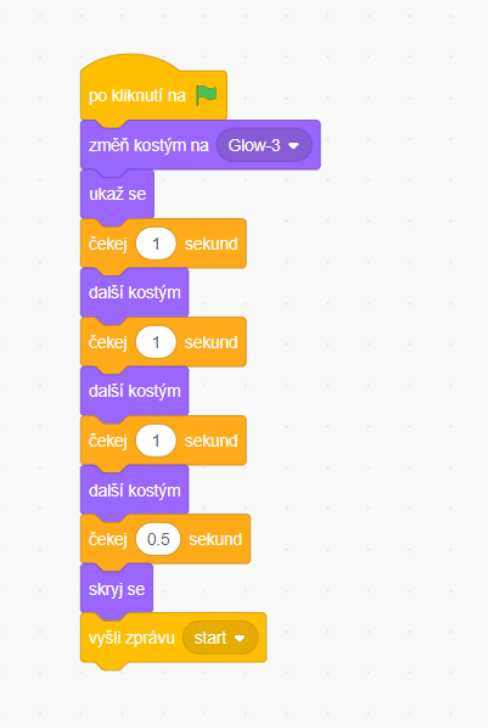
Zadání: Naprogramuj dostihový závod zvířátek. Vytvoř pět zvířecích postav a náhodně upravuj jejich rychlost pohybu. Jakmile se postava dotkne cíle, ulož jí do seznamu a zobraz tak finální pořadí závodníků.

Řešení: V této úloze se žáci naučí pracovat se seznamem. Funguje stejně jako proměnná, akorát se do něj dá uložit více věcí zároveň. Každá postava bude mít téměř stejný kód. V každé bude cyklus „**opakuj stále**“, ve kterém se bude náhodně upravovat rychlost postavy od 1 do 10. Pokud se postava bude dotýkat okraje, znamená to, že je v cíli a název dané postavy se uloží do seznamu „**PORADI**“. Na konci závodu budou tedy v seznamu vypsány všechny postavy v pořadí, ve kterém dorazily do cíle. V Obrázek 40 je ukázka kódu jedné z postav.

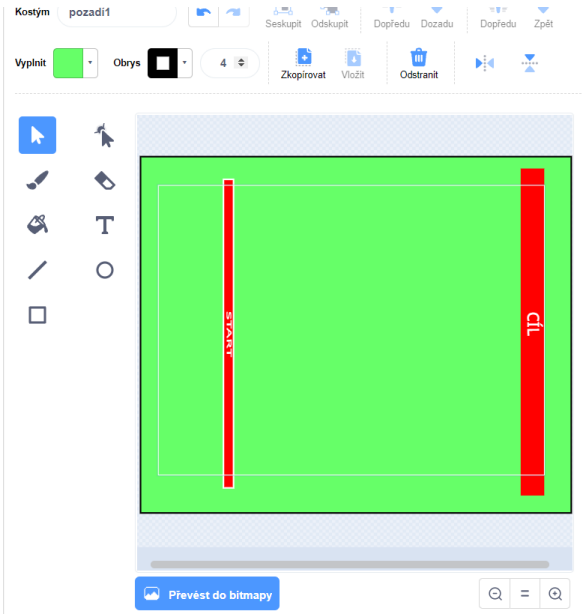
Obrázek 40: *Kód Kočka*



Obrázek 41: *Kód animace startu*



Obrázek 42: *Pozadí*



Doplňující úkoly: Pohraj si s grafickou stránkou programu, přidej animace a zvuk pro každou postavu. Než začne závod, odstartuj ho pomocí jednoduché animace prolínání čísel a zvuku výstřelu.

6.5 Minihra s kuličkou

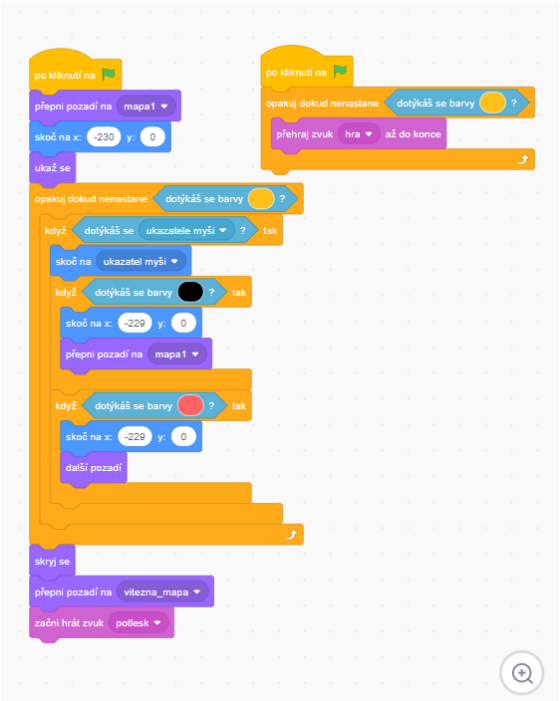
Tabulka 13: *Metodický list - Minihra s kuličkou*

Metodický list	
Název materiálu:	Minihra s kuličkou
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 5 postav, 5 scénářů, 50 bloků
Klíčová slova:	podmínka, cyklus, dotek barvy
Anotace materiálu:	Žák vytvoří minihru, kde bude za úkol přesunout kuličku přes vytvořenou dráhu, aniž by se kulička dotkla hrany vytyčené dráhy.

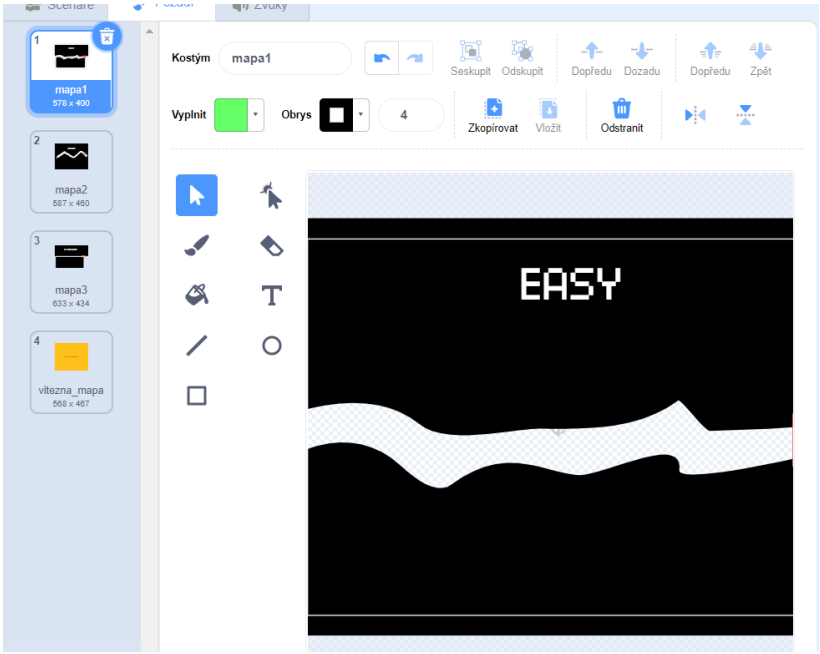
Zadání: Vytvoř minihru pro červenou kuličku. Hra bude obsahovat celkem tři dráhy, které nakreslíš (ideálně, aby se stupňovala náročnost drah). Červená kulička poté bude muset postupně projít všechny dráhy, aniž by se dotkla okraje dráhy.

Řešení: Nejdříve vytvoříme dráhy. Postačí nám dva černé obdélníky, které si upravíme dle své volby. Kulička bude procházet po bílé ploše, a pokud se dotkne černé barvy, skočí na začátek mapy. Ovládání kuličky nastavíme na ukazatel myši. Využijeme bloků „**dotýkáš se barvy**“, „**opakuj dokud nenastane**“ a „**skoč na ukazatel myši**“. V Obrázek 43 můžete vidět řešení takové úlohy.

Obrázek 43: *Kód pro kuličku*



Obrázek 44: *Ukázka dráhy*



Doplňující úkoly: Vytvoř a přidej více map do programu. Pokus se uživateli po úspěšném překonání všech map vypsat jeho počet chyb.

6.6 Lov hmyzu

Tabulka 14: *Metodický list - Lov hmyzu*

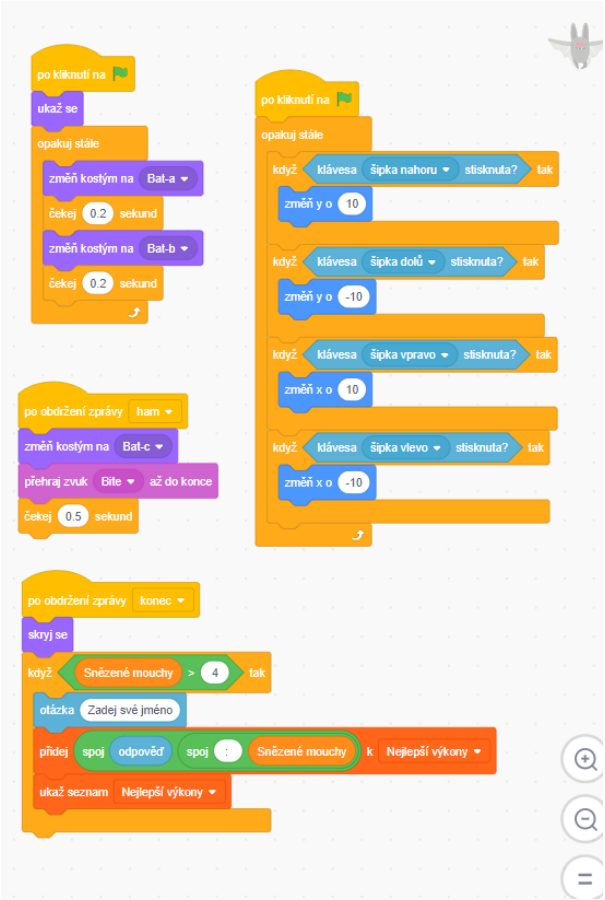
Metodický list	
Název materiálu:	Lov hmyzu
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	90 minut
Rozsah:	1 pozadí, 2 postav, 6 scénářů, 47 bloků
Klíčová slova:	podmínka, cyklus, pohyb postavy, interakce s druhou postavou, seznam, stopky, vyšli zprávu
Anotace materiálu:	Žák naprogramuje hru, ve které netopýr loví mouchy, které náhodně poletují po obrazovce. Hra potrvá minutu, poté bude mít hráč možnost zapsat své skóre do seznamu výsledků.

Zadání úlohy: Vytvoř hru, kde Bajtík bude ovládán pomocí šipek a jeho úkol bude lovit mouchy. Mouchu naprogramuj tak, aby létala náhodně po celé obrazovce. Pokaždé, co se Bajtík dotkne mouchy, započítá se mu bod. Hra potrvá celkem jednu minutu a po konci si hráč bude mít možnost zapsat své skóre na seznam výsledků (ale pouze v tom případě, že nasbíral alespoň 10 much).

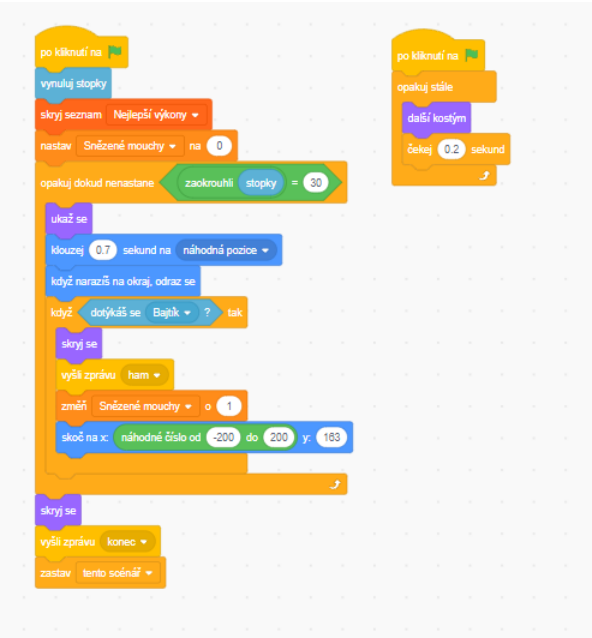
Řešení: Pro postavu Bajtík nám bude stačit, když naprogramuje pohyb – tentokrát pomocí cyklu „**opakuj stále**“ a podmínky s blokem „**klávesa x stisknuta?**“, tento způsob pohybu postavy je plynulejší. Dále tu máme dva scénáře po obdržení zprávy „**ham**“ a „**konec**“. Po obdržení zprávy „**ham**“ přehrajeme zvuk a změníme kostým. Po obdržení zprávy „**konec**“ provedeme uložení jména hráče a jeho skóre do seznamu „**Nejlepší výkony**“ (Obrázek 45). V postavě Moucha je hlavní část programu, kde se kontroluje čas pomocí bloku „**stopky**“, který

měří čas programu od spuštění. Hra je tedy nastavená na 30 sekund (Žáci si mohou nastavit 60). Moucha bude náhodně klouzat po obrazovce, když narazí na okraj, odrazí se. Pokud se dotýká postavy Bajtík, vyšle se zpráva „**ham**“, změníme proměnou „**Snězené Mouchy**“ a skočíme na náhodnou pozici, odkud moucha vylétne. V Obrázek 46 najdete řešení.

Obrázek 45: *Kód postavy Bajtík*



Obrázek 46: *Kód postavy Moucha*



Doplňující úkoly: Přidej do hry zvukové efekty, animace postav, nakresli vlastní pozadí.

7 KATEGORIE PROFÍK

7.1 Trénink přesnosti

Tabulka 15: *Metodický list - Trénink přesnosti*

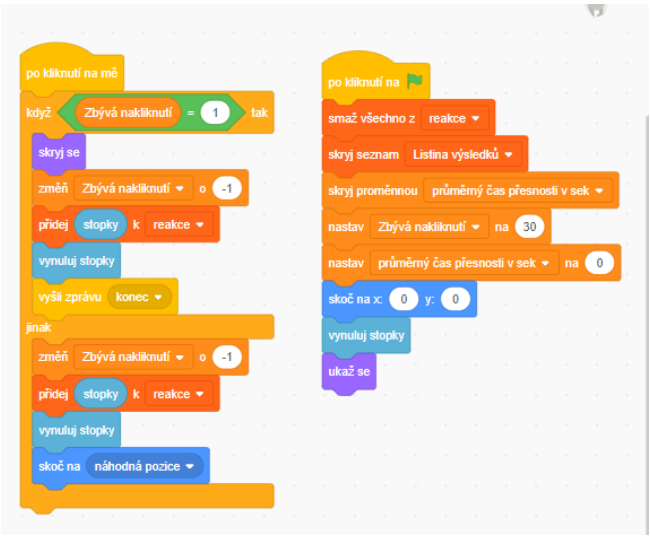
Metodický list	
Název materiálu:	Trénink přesnosti
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	45 minut
Rozsah:	1 pozadí, 1 postava, 3 scénáře, 32 bloků
Klíčová slova:	čas, stopky, seznam, průměr, cyklus
Anotace materiálu:	Žák naprogramuje trénink přesnosti, kde bude za úkol 30x kliknout na postavu, která se po každém kliknutí přemístí na náhodnou pozici. Po každém kliknutí se měří čas reakce, poté se vypočítá průměrný čas reakce.

Zadání úlohy: Vytvoř program, který otestuje tvou rychlost přesnosti v kliknutí na určitý cíl. Test bude spočívat v tom, že uživatel bude muset 30x kliknout na postavu. Při každém kliknutí se postava přemístí na jinou pozici a zaznamenáš čas uživatelovi reakce. Poté vypočítáš průměrný čas jeho reakce, vypíšeš výsledek a dáš uživateli možnost zapsat se do listiny výsledků.

Řešení: Vytvoříme seznam „**reakce**“, do kterého se vždy bude ukládat čas reakce kliknutí na postavu Bajtík. Dále vytvoříme proměnnou „**zbývá nakliknutí**“, která nám bude ukazovat kolik zbývá pokusů a nastavíme ji na hodnotu 30. V samotném programu pak jen za pomocí bloku „**po kliknutí na mě**“ naprogramuje průběh (Obrázek 47). Po obdržení zprávy „**konec**“ se provede finální výpočet. Budeme postupně brát prvky ze seznamu „**reakce**“ a přičítat je

postupně do proměnné „**soucet**“. Musíme však začít od druhého prvku, protože první kliknutí na postavu slouží jako start testu, tedy se nemůže započítávat. Takhle projdeme celý seznam a poté provedeme aritmetický průměr (součet reakcí / počet pokusů) a uložíme jej do proměnné „**výsledek**“, kterou uživateli vypíšeme. Poté se pomocí bloku „**otázka**“ zeptáme na jméno uživatele a uložíme jeho výsledek do seznamu „**Listina výsledků**“ (Obrázek 48).

***Obrázek 47**: Kód postavy Bajtík 1*



***Obrázek 48:** Kód postavy Bajtík 2*

7.2 Dodgeball

Tabulka 16: *Metodický list - Dodgeball*

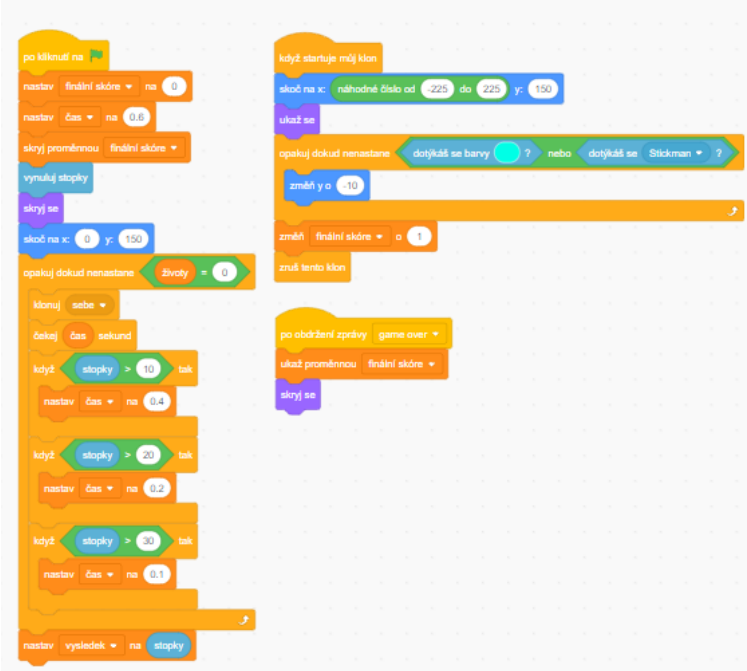
Metodický list	
Název materiálu:	Dodgeball
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	90 minut
Rozsah:	2 pozadí, 2 postava, 6 scénářů, 56 bloků
Klíčová slova:	klon, klonuj sebe, interakce s postavou, čas
Anotace materiálu:	Žák vytvoří hru, kde se postava snaží vyhýbat míčům, které padají shora dolů. Každých deset sekund se bude intenzita padajících míčů zvyšovat.

Zadání úlohy: Vytvoř hru, kde budeš ovládat postavu zleva doprava. Její úkol bude vyhýbat se míčům, které budou padat se shora dolů. Padající míče naprogramuj tak, aby se po deseti sekundách změnila intenzita jejich padání. Postava bude mít tři životy. Za každé uhnutí míče, započítej jeden bod. Za každé trefení míčem uber život. Na konci vypiš skóre.

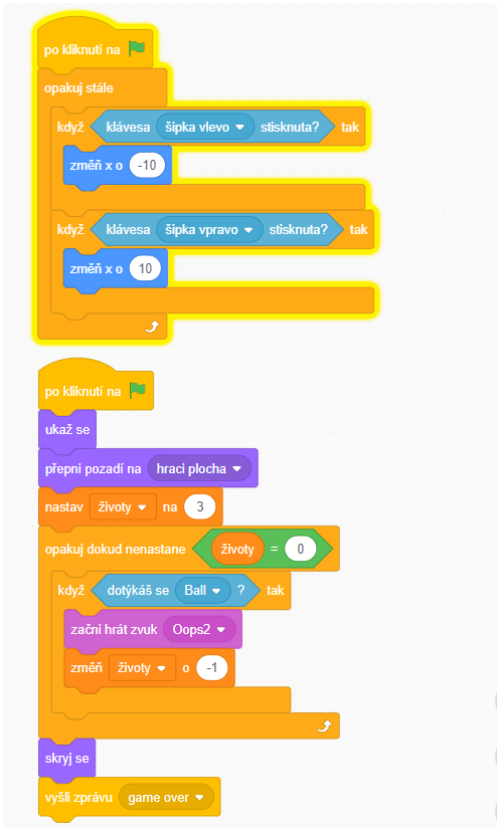
Řešení úlohy: V této úloze poprvé využijeme blok „**klonuj sebe**“. Tento blok vytvoří kopii postavy a pomocí bloku „**když startuje můj klon**“ nastavíme, co se s danou kopií bude dít. V tomto případě nastavíme, aby klon skočil na náhodnou horizontální pozici, vertikálně je nastavený staticky na souřadnici y = 150. Klon se bude měnit svou vertikální souřadnici směrem dolů, tedy bude padat, dokud nenastane, že se dotýká postavy „**Stickman**“ anebo barvy, která je na dolní plošině pod postavou „**Stickman**“, pak se klon zruší a skočí se do hlavního cyklu, který probíhá, dokud má hráč víc jak nula životů. Za využití stopek budeme měřit čas programu. Po uplynutí každých deseti sekund budeme měnit proměnou „**čas**“, která bude určovat, jak dlouho se bude čekat na vytvoření dalšího klonu. V Obrázek 49 můžete vidět řešení takového

programu. V Obrázek 50 je program pro postavu „**Stickman**“, kde je scénář pro pohyb postavy a poté cyklus, který hlídá počet životů postavy.

Obrázek 49: *Kód postavy Ball*



Obrázek 50: *Kód postavy Stickman*



7.3 Tetris

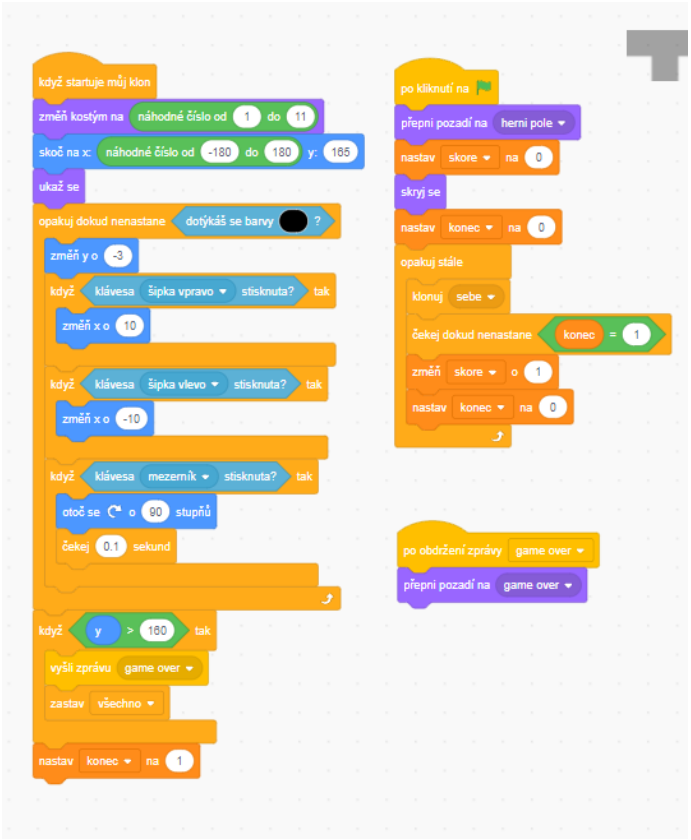
Tabulka 17: *Metodický list - Tetris*

Metodický list	
Název materiálu:	Tetris
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	90 minut
Rozsah:	2 pozadí, 1 postava, 3 scénáře, 38 bloků
Klíčová slova:	klon, klonuj sebe, interakce s postavou, otáčení postavy
Anotace materiálu:	Žák naprogramuje napodobeninu známé hry tetris.

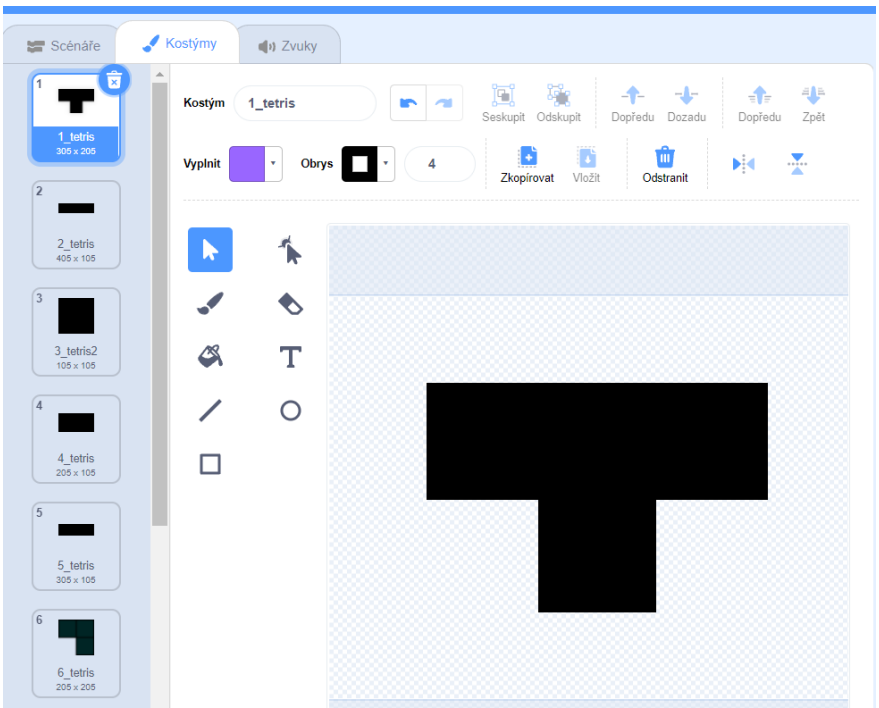
Zadání úlohy: Vytvoř napodobeninu hry Tetris. Jde o padající bloky různých tvarů. Bloky zkus vytvořit v jiném grafickém programu a nahraj je do Scratche.

Řešení úlohy: Bloky můžeme vytvořit např. v jednoduchém programu Malování (popř. pokročilejší programy jako GIMP, Adobe Photoshop, Adobe Illustrator) a nahrajeme je do kostýmů (Obrázek 52). V hlavním cyklu programu budeme využívat bloku „**klonuj sebe**“. Na každý další klon se bude čekat pomocí proměnné „**konec**“. Pro každý klon potom nastavíme, aby se zvolil náhodný kostým bloku a dokud se nedotýká černé barvy (tzn. dokud se nedotýká spodku hracího pole nebo dalšího bloku), bude blok klouzat směrem dolů. Pomocí šipek nastavíme pohyb doleva, doprava a pomocí mezerníku umožníme otáčet s blokem o 90 stupňů. Pokud se klon bude dotýkat vršku hracího pole, znamená to konec a vyšleme zprávu „**game over**“, která změní pozadí hry a je konec. Obrázek 51 ukazuje řešení takového programu.

Obrázek 51: *Kód postavy Blok*



Obrázek 52: *Kostýmy postavy Blok*



7.4 Flappy Bajtík

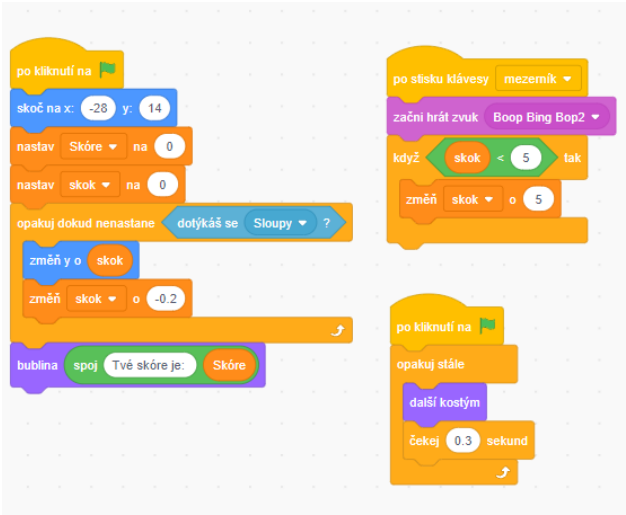
Tabulka 18: *Metodický list - Flappy Bajtík*

Metodický list	
Název materiálu:	Flappy Bajtík
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	90 minut
Rozsah:	1 pozadí, 2 postavy, 4 scénáře, 30 bloků
Klíčová slova:	skákací hra, skok postavy, proměnná, interakce s postavou
Anotace materiálu:	Žák vytvoří hru podobnou známé hře „Flappy Bird“. Jde o skákací hru, kde se postava snaží vyhnout překážkám.

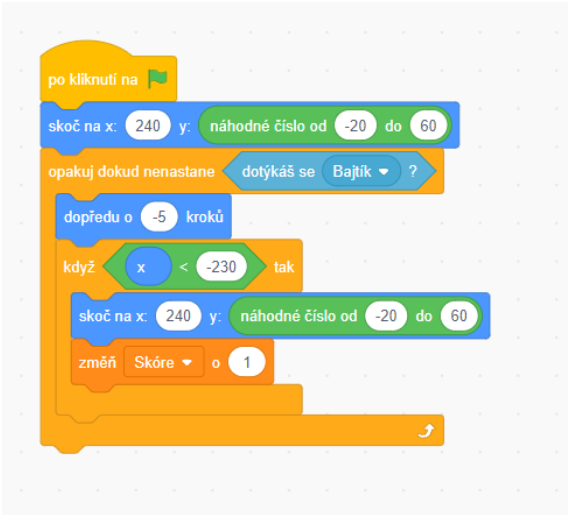
Zadání úlohy: Vytvoř hru, podobnou známé hře „Flappy Bird“. Místo ptáčka ale nahradí náš známý Bajtík. Jeho úkolem bude přelétnout mezi dvěma sloupy. Jediné tlačítko pohybu bude mezerník, po jehož stisknutí postava poskočí. Pokud se postava dotkne sloupu, ukonči hru a vypiš skóre.

Řešení úlohy: Vytvoříme sloupy, které se budou vždy pohybovat proti postavě Bajtík v náhodné výšce. Pokud se nedotknou postavy Bajtík a jejich souřadnice bude „menší jak -230“ (tzn. budou na levém kraji), skočí zpátky doprava na náhodnou souřadnici y (Obrázek 53). Postavu Bajtík naprogramujeme tak, aby dokud se nedotýká postavy „**Sloupy**“, skákala po stisknutí klávesy mezerník. To dosáhneme pomocí proměnné „**skok**“. Na začátku programu je nastavená na nulu a po každém stisknutí mezerníku se změní o pět (pokud je hodnota proměnné menší jak pět). V hlavním cyklu se potom v každém opakování snižuje o hodnotu -0.2. Díky tomu se dá hezky simulovat animace skoku postavy. V Obrázek 54 můžete vidět řešení.

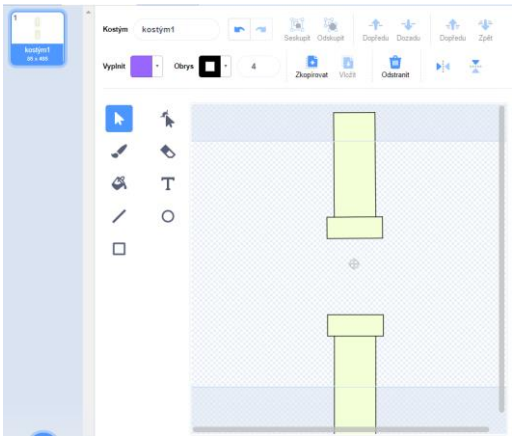
Obrázek 53: *Kód postavy Bajtík*



Obrázek 54: *Kód postavy Sloupy*



Obrázek 55: *Kostým Sloupy*



7.5 Útok asteroidů

Tabulka 19: *Metodický list - Útok asteroidů*

Metodický list	
Název materiálu:	Útok asteroidů
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	120 minut
Rozsah:	1 pozadí, 6 postav, 14 scénářů, 100 bloků
Klíčová slova:	klonuj sebe, cyklus, podmínka, proměnná, vyšli zprávu, výroková logika
Anotace materiálu:	Žák naprogramuje hru, kde vesmírná loď bude mít za úkol odrazit útok asteroidů, které padají směrem na planetu.

Zadání úlohy: Naprogramuj hru, kde vesmírná loď bude bránit planetu od padajících asteroidů. Loď bude mít tři životy a jako zbraň bude mít laser, který může vystřelit po asteroidu a zničit jej. Planeta bude mít 100 životů, po každém poškození asteroidem se planetě ubere 10 životů. To stejné platí pro styk vesmírné lodě s asteroidem – loď ztrácí jeden život. Pokud planeta dokáže přežít, pogratuluj hráčovi. Pokud planeta nepřežije, zmizí a hráčovi se vypíše „Game Over“.

Řešení:

Postava „**vesmírná loď**“ (Obrázek 56): v prvním scénáři postavy „**vesmírná loď**“ je cyklus „**opakuj stále**“, kde nastavíme pohyb postavy pomocí podmínky „**když klávesa šipka vpravo/vlevo stisknuta? tak**“. Druhý scénář se spustí po obdržení zprávy „**START**“. Nastavíme proměnnou „**loď životy**“ na hodnotu 3. V cyklu „**opakuj dokud nenastane“**, který probíhá dokud „**loď životy**“ není nula je podmínka pro zjištění kolize s postavou „**asteroid**“,

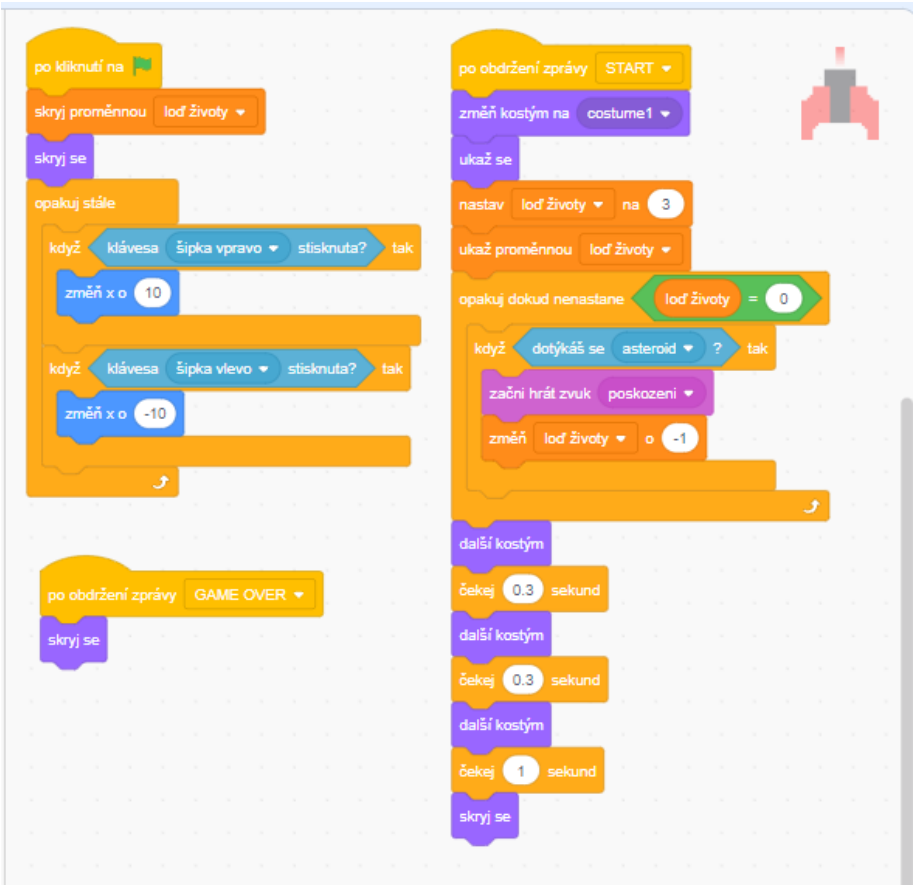
která ubere životy, pokud je pravdivá. Po skončení cyklu se provede animace výbuchu vesmírné lodě.

Postava „**laser**“ (Obrázek 57): První scénář začíná blokem „**po stisku klávesy mezerník**“, kde je podmínka, která kontroluje počet životů postavy „**vesmírná lod'**“, pokud je splněna podmínka, provede se blok „**klonuj sebe**“. V druhém scénáři navazuje blok „**když startuje můj klon**“, kde se řeší chování laseru od momentu odpalu až po samotný náraz na okraj mapy nebo do postavy „**asteroid**“.

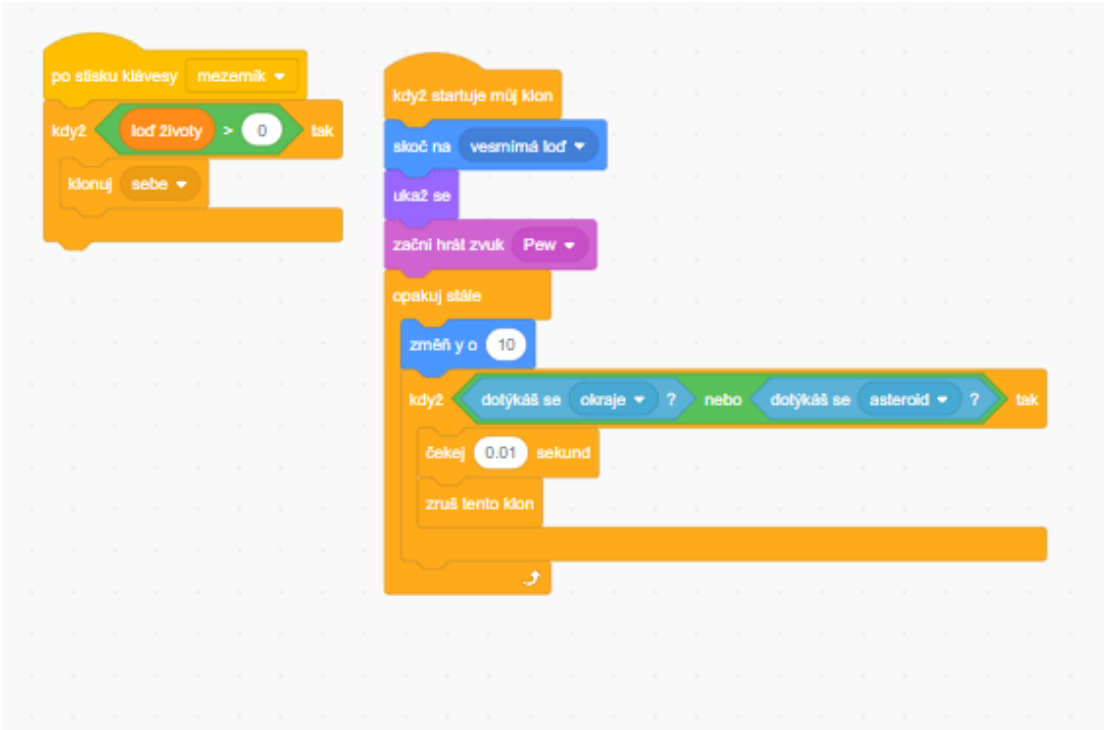
Postava „**asteroid**“ (Obrázek 58): V prvním scénáři s blokem „**po obdržení zprávy START**“ je cyklus „**opakuj 50 krát**“, ve kterém se v každém opakování klonuje postava a po průběhu všech 50 opakování se zkontrolují životy postavy **„planeta“.** V druhém scénáři je potom kód pro každý začínající klon. Každý klon skočí na náhodnou x souřadnici a začne pomalu padat směrem dolů, dokud nenarazí na jednu z postav – v takovém případě zmizí. Vše je řešeno cyklem „**opakuj dokud nenastane**“ a podmínkou „**dotýkáš se?“**.

Postava „**planeta**“ (Obrázek 59): Scénář začíná blokem „**po obdržení zprávy START**“. Je tu proměnná „**planeta životy**“, kterou v základu nastavíme na hodnotu 100. Dále tu je cyklus „**opakuj dokud nenastane**“, který probíhá dokud není proměnná „**planeta životy**“ na nule. V cyklu se potom kontroluje, jestli došlo ke kolizi s postavou „**asteroid**“. Pokud ano, proměnná „**planeta životy**“ se sníží o 10. Po skončení cyklu se vyšle zpráva „**GAME OVER**“, která se zpracovává v postavě „**konec Hry“** (viz Obrázek 60).

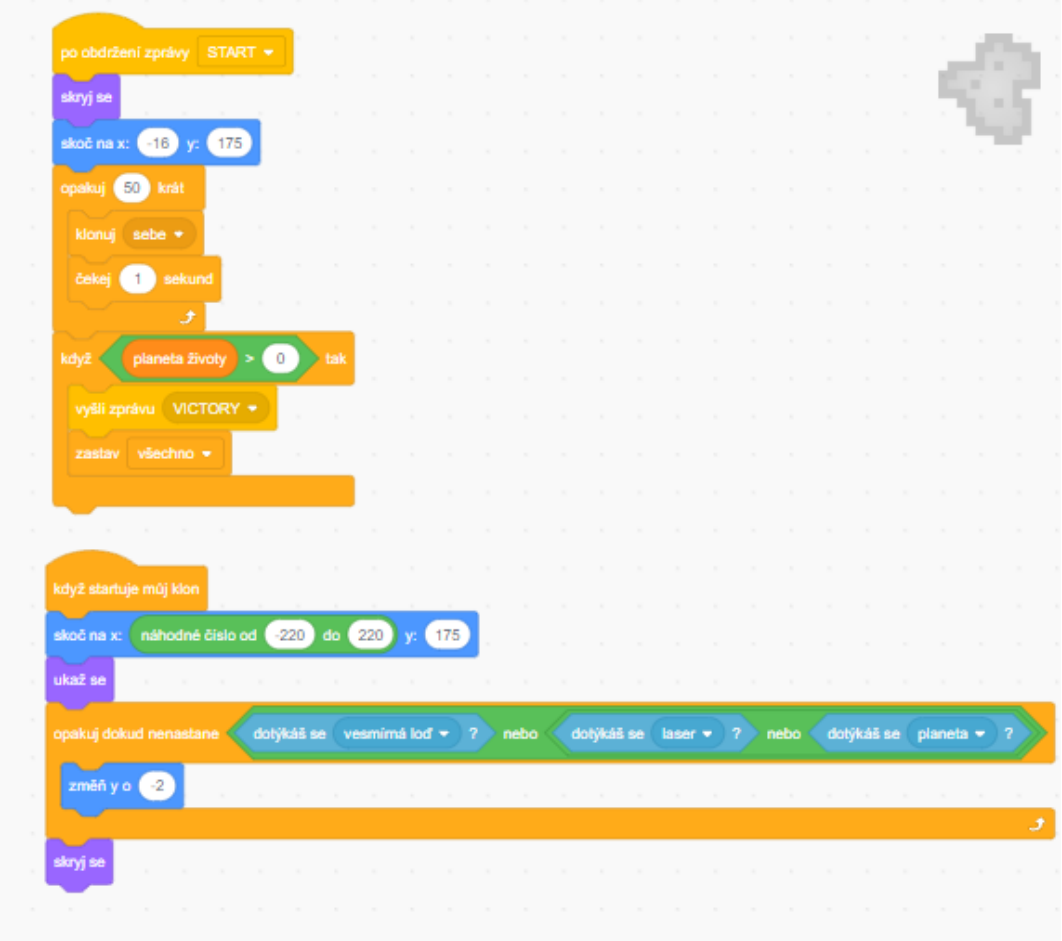
Obrázek 56: *Kód postavy vesmírná loď*



Obrázek 57: *Kód postavy laser*



Obrázek 58: *Kód postavy asteroid*



Obrázek 59: *Kód postavy planeta*

Obrázek 60: *Kód postavy konec hry*

7.6 Had

Tabulka 20: *Metodický list - Had*

Metodický list	
Název materiálu:	Had
Vzdělávací oblast:	Informační a komunikační technologie
Vyučovací předmět:	Informatika
Věkové určení:	2. stupeň ZŠ
Časová dotace:	120 minut
Rozsah:	4 pozadí, 3 postavy, 21 scénářů, 134 bloků
Klíčová slova:	klonuj sebe, cyklus, podmínka, proměnná, vyšli zprávu, výroková logika, interakce s postavou
Anotace materiálu:	Žák naprogramuje hru Had, která bude fungovat pro dva hráče. Had sbírá potravu - po každém snězení potravy se had zvětšuje. Pokud had narazí do druhého hada nebo narazí na kraj hracího pole, vyhrává ten druhý had a naopak.

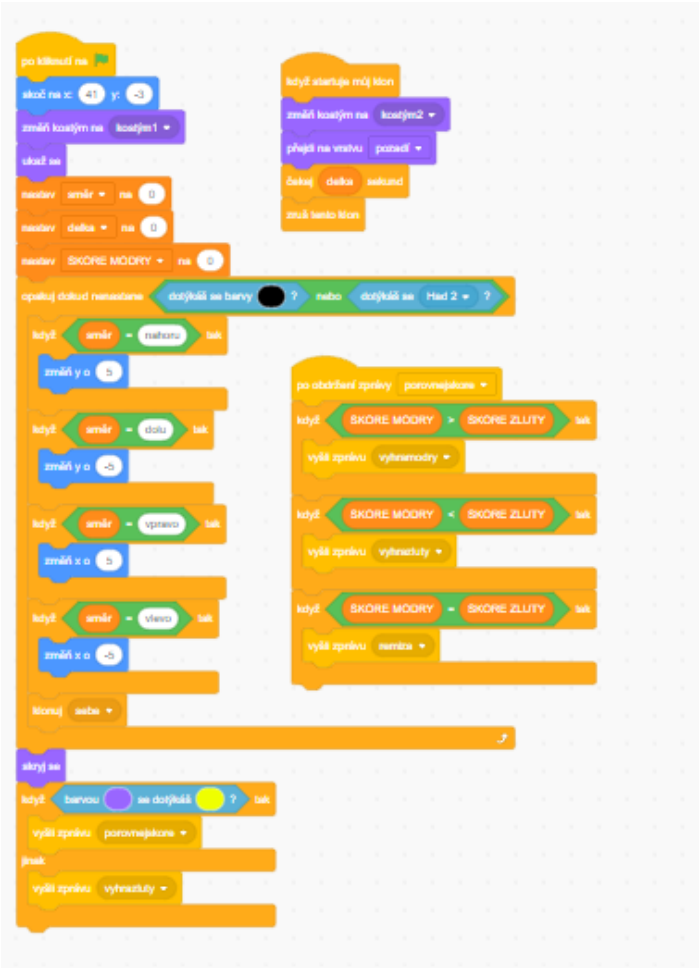
Zadání úlohy: Naprogramuj hru Had pro dva hráče. Pravidla jsou následující:
Pokud Had1 narazí do těla Had2 (stejně i naopak) - vyhrává Had2.
Pokud Had1 narazí do hlavy Had2 (stejně i naopak) – vyhrává ten had, který nasbíral více jídla.
Pokud Had1 nebo Had2 narazí na kraj hracího pole – vyhrává automaticky ten druhý had.

Řešení úlohy: Pohyb Hada nastavíme pomocí proměnné „**směr**“, kterou nastavíme na hodnoty nahoru, dolu, doleva a doprava podle toho jaká klávesa šipky je stisknuta (Obrázek 61). Poté pomocí proměnné „**delka**“ budeme zvyšovat délku hada. Funguje to tak, že pokud se postava „**Had1** nebo **Had2**“ dotkne postavy „**Jidlo**“, proměnná „**delka**“ se změní o 0.1. V hlavním cyklu postavy „**Had** nebo **Had2**“ se potom bude neustále klonovat tělo hada přesně o aktuální nastavené délce. Konec hry je vyřešen tak, že se čeká, dokud se had dotkne buď černé barvy (tzn. okraje mapy), anebo druhého hada, který se odlišuje barvou. Poté se vyšle příslušná

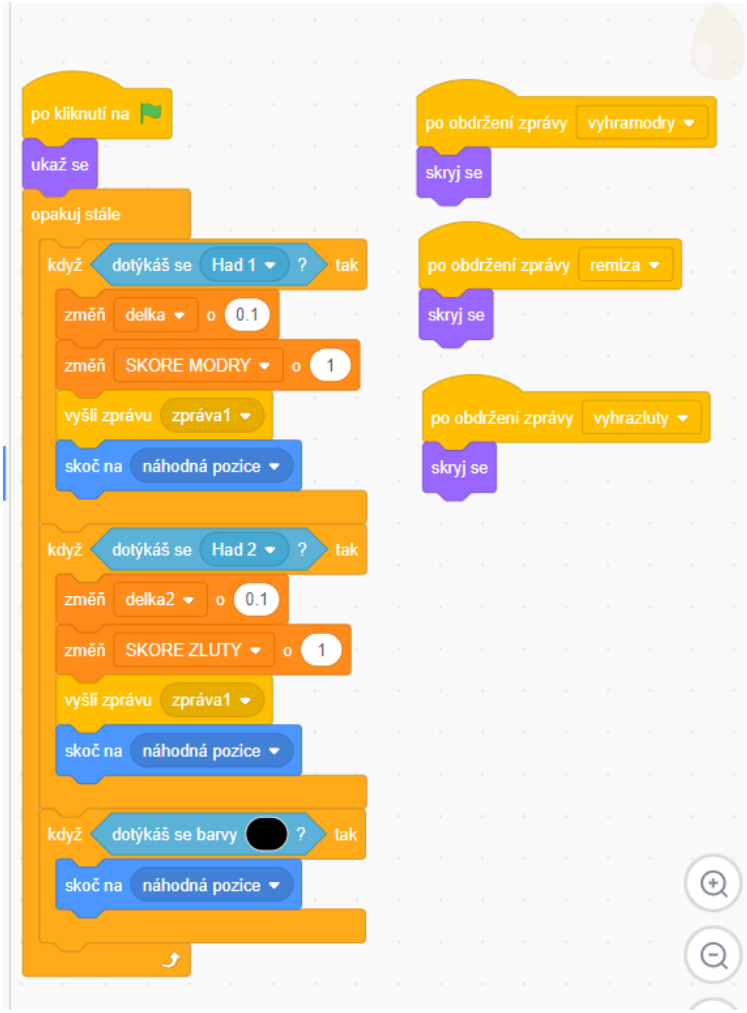
zpráva, kdo vyhrál. Pokud se „**Had1**“ dotkne hlavou postavy „**Had2**“ (nebo naopak), tak se vyhodnotí skóre dvou hadů a opět pomocí „**vyšli zprávu**“ se určí vítěz a ukončí se hra. Obrázek 62 je řešení kódu pro postavu hada. Obrázek 63 ukazuje kód pro postavu jídla, které slouží jako potrava pro hada.

Obrázek 61: *Pohyb postavy Had1*

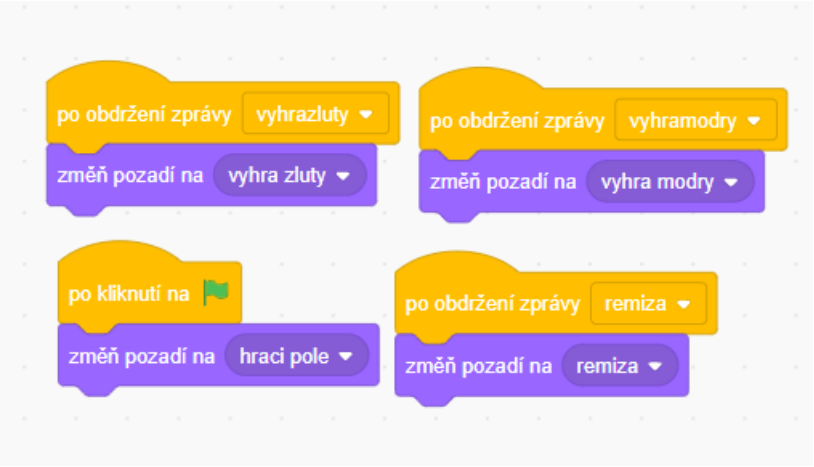
Obrázek 62: *Hlavní kód postavy Had1*



Obrázek 63: *Kód postavy Jídlo*



Obrázek 64: *Kód pozadí*



8 UMÍSTĚNÍ ÚLOH

Sada úloh bude umístěná na dvou místech. Prvním místem bude webová stránka scratch.mid.edu, což je webová platforma určena pro tvorbu interaktivních příběhů, animací, her a dalších multimediálních projektů. Je vyvinuta a provozována MIT Media Lab. Tato platforma je primárně zaměřena na vzdělávání v oblasti programování a počítačového myšlení pro děti a začátečníky. Úlohy s řešením pro učitele jsou dostupné na mém profilu. Zde je odkaz: https://scratch.mit.edu/users/kvetda00/projects/.

Druhým místem je cloudové online úložiště GitHub, což je webová platforma pro správu verzí a správu kódových projektů. Je to jeden z nejpopulárnějších nástrojů pro správu verzí a poskytuje služby pro spolupráci na vývoji softwaru. Umožňuje vývojářům spolupracovat na projektech, sledovat změny ve zdrojovém kódu, navrhovat úpravy, spravovat problémy a mnoho dalšího. GitHub je postaven na systému správy verzí Git, který umožňuje ukládat historii změn v kódu a zpřístupňuje nástroje pro správu těchto změn. Uživatelé mohou vytvářet repozitáře, které obsahují zdrojový kód jejich projektů a umožňují lidem přispívat kódem, navrhovat změny a diskutovat o vylepšeních či chybách. Úlohy s řešením určené pro učitele i úlohy upravené pro žáky budou nahrané a volné ke stažení v repozitáři mého účtu, který najdete na odkaze: https://github.com/davidkveton/kveton_bc.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit sadu úloh, která bude sloužit jako pomocný materiál pro učitele základních škol a kterou mohou využít při výuce programování na druhém stupni ZŠ. Úlohy jsou vytvořené ve vizuálním programovacím jazyce Scratch. Žáci se tak naučí orientovat v programovacím prostředí jazyka Scratch a osvojí si schopnost pracovat s cyklem, podmínkou, proměnnou a dalšími bloky z prostředí tohoto jazyka.

První kapitola bakalářské práce se zaměřovala na informatické myšlení a jeho rozvoj u žáků při výuce Informatiky na základních školách. Vymezil jsem složky a definici tohoto pojmu a popsal jeho využití v běžném životě. Druhá kapitola se zabývala digitálním vzděláváním. Popsal jsem vizi a strategii digitálního vzdělávání v ČR. Vymezil jsem cíle strategie vzdělávací politiky do roku 2030+ a zkusil popsat nové pojetí výuky informatiky na ZŠ. Třetí a poslední kapitola teoretické části se zaměřuje na popis a využití vizuálního programovacího jazyka. Dále jsem popsal různé typy VPL a nejvíce se zaměřil na popis jazyka Scratch, kterému se věnuji v praktické části bakalářské práce. Nakonec jsem ještě uvedl příklady možností konverze VPL do vyšších programovacích jazyků jako jsou Python, JavaScript či C.

Praktická část popisuje rozdělení úloh do tří kategorií a důvod tohoto rozhodnutí. V dalších kapitolách se potom nachází podrobný rozbor jednotlivých úloh. U každé úlohy je obsažen metodický list, zadání úlohy, doplňující úkoly, následné řešení a obrázky ukazující způsob řešení. Poslední kapitola praktické části je věnována umístěním úloh a přístupnosti úloh.

Cíle práce byly naplněny. Všechny tři kategorie obsahují celkem šest popsaných úloh a jednu nepopsanou úlohu, která slouží pouze jako motivační úvod k dané kategorii. Bylo tedy vytvořeno 21 úloh a jejich využití může být jako pomocný materiál pro výuku programování na druhém stupni ZŠ.

SEZNAM POUŽITÝCH ZKRATEK

ČR	Česká republika
IT	Informační technologie
MIT	Massachusettský technologický institut
MPSV	Ministerstvo práce a sociálních věcí
MŠMT	Ministerstvo školství, mládeže a tělovýchovy
např.	například
PRIM	Podpora rozvíjení informatického myšlení
RVP ZV	Rámcový vzdělávací program pro základní vzdělávání
ŠVP	Školní vzdělávací program
tzn.	to znamená
tzv.	takzvaně/ý
VPL	Vizuální programovací jazyk
ZŠ	Základní škola

SEZNAM POUŽITÝCH ZDROJŮ

Knižní zdroje:

[1] PRŮCHA, Jan, WALTEROVÁ, Eliška a MAREŠ, Jiří (2013). Pedagogický slovník. 7., aktualiz. a rozš. vyd. Praha: Portál, ISBN 978-80-262-0403-9.

[2] VANÍČEK, Jiří, ed. Výuka algoritmizace patří především do informatiky. In: ROSECKÝ, Jan. *Počítač ve škole 2016 – sborník příspěvků*. Nové Město na Moravě: Gymnázium Vincence Makovského se sportovními třídami. 2016. ISBN 978- 80-905765-6-8.

[3] PELÁNEK, Radek. *Jak to vyřešit?: logické úlohy a hry*. 1. vyd. Praha: Portál, 2011. ISBN 978-80-7367-872-2.

[4] JOST, KETTERL, BUDDE a LEIMBACH. *Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One?* IEEE, 2014. ISBN 978-1-4799-4311-1.

[5] MAJED, Marji. *Learn to Program with Scratch: A Visual Introduction to Programming with Games, Art, Science, and Math*. No Starch Press, 2014. ISBN 978-1593275433.

Online zdroje:

[6] *International Society for Technology in Education* [online]. Alexandria: ISTE, 2012 [cit. 2023-11-17]. Dostupné z: http://www.assnstrategies.com/pdf/ISTEPositionProfileFinal.pdf

[7] *Strategie digitálního vzdělávání do roku 2020* [online]. Praha: Ministerstvo školství, mládeže a tělovýchovy, 2014 [cit. 2023-11-26]. Dostupné z: https://www.msmt.cz/uploads/DigiStrategie.pdf

[8] *Informatické myšlení: Co je informatické myšlení* [online]. Jihočeská univerzita v Českých Budějovicích, 2018 [cit. 2023-11-17]. Dostupné z: https://www.imysleni.cz/informaticke-mysleni/co-je-informaticke-mysleni

[9] SELBY, Cynthia a John WOOLARD. UNIVERSITY OF SOUTHAMPTON. *Computational Thinking: The Developing Definition*. Southampton, 2013. [online]. [cit. 2023-11-17]. Dostupné z: https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf

[10] COHEN, Avi a Bruria HABERMAN. *Computer Science: A Language of Technology* [online]. ACM SIGCSE Bulletin, 2007 [cit. 2023-11-22]. Dostupné z: doi:10.1145/1345375.1345417

[11] LESSNER, Dan. *Hledání dárců ledvin* [online]. Jihočeská univerzita v Českých Budějovicích, 2018 [cit. 2023-11-17]. Dostupné z: https://www.imysleni.cz/clanky/priklady/23-hledani-darculevin?fbclid=IwAR3wpkcVVV0JxcB_up9tCbKz38dAilA3huMyAysUM6pToVQSBuJ P7PKKQCM

[12] HYLÉN, Jan. *Open educational resources: Opportunities and challenges. Oecd* [online], 2006 [cit. 2023-11-21]. Dostupné z: http://bit.ly/1sFikJX

[13] MŠMT. *Rozpracovaný koncept digitální gramotnosti* [online]. 2018, verze 2.0 [cit. 2023-11-21]. Dostupné z: https://digigram.cz/files/2019/06/VM1.1-Koncept-DG.pdf

[14] Ministerstvo práce a sociálních věcí. *Strategie vzdělávací politiky ČR do roku 2030+* [online]. ©2013-2021, 2020 [cit. 2023-11-21]. Dostupné z: https://www.msmt.cz/uploads/Brozura_S2030_online_CZ.pdf

[15] BARR, HARRISON a CONERY. *Computational Thinking: Digital Age* [online]. International Society for Technology in Education, 2011 [cit. 2023-11-17]. ISSN-1082-5754. Dostupné z: https://files.eric.ed.gov/fulltext/EJ918910.pdf

[16] KUHAIL, FAROOQ, HAMMAD a BAHJA. *Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review*. IEEE, 2021. [online]. ISSN 2169-3536. Dostupné z: https://ieeexplore.ieee.org/document/9320477

[17] REPENNING. *Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets* [online]. Journal of Visual Languages and Sentient Systems [cit. 2023-11-20]. Dostupné z: doi:10.18293/vlss2017-010

[18] *Visual programming vs Traditional Programming: Full Guide* [online]. nanbox [cit. 2023-11-26]. Dostupné z: https://nandbox.com/visual-programming-vs-traditional-programming-full-guide/

SEZNAM OBRÁZKŮ

SEZNAM TABULEK