TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní
a pedagogická

Využití programovacího jazyka Scratch pro tvorbu her jako součást výuky programování

Diplomová práce

Studijní program:	N7503 Učitelství pro základní školy
Studijní obory:	Učitelství anglického jazyka pro 2. stupeň základní školy Učitelství informatiky pro 2. stupeň základní školy
Autor práce:	**Bc. Andrea Dufková**
Vedoucí práce:	Ing. Jindra Drábková, Ph.D. Katedra aplikované matematiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní
a pedagogická

Zadání diplomové práce

Využití programovacího jazyka Scratch pro tvorbu her jako součást výuky programování

Jméno a příjmení:	**Bc. Andrea Dufková**
Osobní číslo:	P19000336
Studijní program:	N7503 Učitelství pro základní školy
Studijní obory:	Učitelství anglického jazyka pro 2. stupeň základní školy Učitelství informatiky pro 2. stupeň základní školy
Zadávající katedra:	Katedra aplikované matematiky
Akademický rok:	**2017/2018**

Zásady pro vypracování:

Cílem diplomové práce je zhodnotit, zda je možné využít programovací jazyk Scratch pro vytvoření složitějšího projektu se základem v již existující deskové či počítačové hře. Postup tvorby bude popsán a doplněn o poznámky tak, aby bylo možné tuto práci využít jako inspiraci pro rozšiřující zadání úloh pro nadané žáky na druhém stupni základní školy.

1. Studentka provede rešerši již existujících složitějších her (projektů) vytvořených pomocí jazyka Scratch.
2. Studentka s pomocí prostudované literatury stanoví a popíše jednotlivé etapy tvorby zvolené hry.
3. Studentka se zaměří na problémy, se kterými se při programování projektu setkala, a popíše jejich řešení.
4. Studentka popíše, jakým způsobem lze využít vytvořený projekt u nadaných žáků.

Rozsah grafických prací:
Rozsah pracovní zprávy:
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

VANÍČEK, Jiří, Ingrid NAGYOVÁ a Monika TOMCSÁNYIOVÁ, 2019. *Programování ve Scratch pro 2. stupeň základní školy* [online]. [cit. 2020-04-17]. Dostupné z: https://imysleni.cz/ucebnice/programovani-ve-scratchi-pro-2-stupen-zakladni-skoly/
ČERNOCHOVÁ, Miroslava, Jiří ŠTÍPEK a Petra VAŇKOVÁ, 2019. *Programování ve Scratch II (projekty pro 2. stupeň základní školy)* [online]. [cit. 2020-04-17]. Dostupné z: https://imysleni.cz/ucebnice/programovani-ve-scratchi-ii-projekty-pro-2-stupen-zakladni-skoly/
PECINOVSKÝ, Rudolf. *Současné trendy v metodice výuky programování*. Počítač ve škole 2006, 2006.
FREEMAN, Elisabeth, Kathy SIERRA a Bert BATES, 2004. *Head first design patterns*. Sebastopol: O'Reilly. ISBN 05-960-0712-4.
GIBSON, Jeremy, 2015. *Introduction to Game Design, Prototyping, and Development*. Boston: Addison-Wesley. ISBN 978-0-321-93316-4.
SALEN, Katie a Eric ZIMMERMAN, 2004. *Rules of Play – Game Design Fundamentals* [online]. Massachusetts: MIT Press [cit. 2020-04-17]. ISBN 0-262-24045-9. Dostupné z: https://is.muni.cz/el/phil/jaro2016/IM090/um/rules.of.play.game.design.fundamentals.pdf

Vedoucí práce: Ing. Jindra Drábková, Ph.D.
Katedra aplikované matematiky

Datum zadání práce: 7. listopadu 2017
Předpokládaný termín odevzdání: 20. prosince 2018

L.S.

prof. RNDr. Jan Picek, CSc.
děkan

doc. RNDr. Miroslav Koucký, CSc.
vedoucí katedry

V Liberci dne 12. prosince 2017

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracovala samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědoma toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědoma následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

11. července 2021 Bc. Andrea Dufková

Poděkování

Ráda bych poděkovala vedoucí mé diplomové práce paní Ing. Jindře Drábkové, Ph.D. za cenné rady, věcné připomínky, vstřícnost a podporu při tvorbě této diplomové práce.

Anotace

Tato diplomová práce se zaměřuje na tvorbu rozšiřujících materiálů pro výuku programování na druhém stupni základní školy. Součástí práce je představení prostředí Scratch a některých jeho alternativ jako je Kodu Game Lab, Code.org a Tynker. Výstupem práce je série pracovních listů doplněných o videa s ukázkou průběhu každé hry a hotový projekt, který má v sobě zapracovaná i rozšiřující zadání. Pracovní listy jsou primárně určeny pro nadané žáky, ale mohou sloužit i jako podpůrný materiál pro práci s celou třídou. Jednotlivé pracovní listy jsou tematicky zaměřené a mají svůj základ v již existujících hrách.

Klíčová slova

Informatické myšlení, informatika, pracovní listy, programování, Scratch, výukové materiály

Annotation

This diploma thesis focuses on the creation of extension materials for teaching computer programming at lower secondary schools. An introduction of the Scratch programming interface is a part of this thesis, as well as a brief description of alternative interfaces such as Kodu Game Lab, Code.org or Tynker. The outcome of this thesis is in the form of a series of worksheets supplemented with illustrative gameplay videos and finished projects which have all the extra tasks incorporated. The worksheets are primarily designed for talented pupils, but they can also be used for work with the whole class. The individual worksheets are all thematically focused and based on already existing games.

Keywords

Computational thinking, computer programming, Informatics, Scratch, teaching materials, worksheets

Obsah

Seznam obrázků

Úvod

Pro svou diplomovou práci jsem si zvolila tvorbu rozšiřujících materiálů pro výuku programování v prostředí Scratch. V současné době je na trhu značné množství učebnic a jiných materiálů, které žáka s tímto prostředím seznamují a učí ho základním principům programování. Běžně dostupné materiály jsou však orientovány spíše na jednotlivé koncepty než na komplexnější projekty a hry. Společně s nově přicházející reformou rámcového vzdělávacího programu se právě výuka programování stává čím dál více relevantním tématem pro širokou skupinu učitelů. S vyšší dotací hodin v informatice a s větším důrazem na informatické myšlení a programování tak můžeme očekávat také vyšší zájem o rozšiřující materiály ze strany učitelů, rodičů či samotných žáků, než jak tomu bylo doposud.

S prostředím Scratch jsem se seznámila v rámci svého bakalářského studia, kdy jsem navštívila Scratch konferenci v Budapešti a rok vedla kroužek programování ve Scratch v rámci Dětské univerzity. Již během takto krátké doby jsem se setkala s několika dětmi, které vyčnívaly svým talentem nad zbytek skupiny. Často jsem však narážela na problém, kdy jsem nemohla těmto nadaným dětem věnovat dostatečný prostor pro rozvoj jejich potenciálu. Mnohokrát mi chyběly materiály, s jejichž pomocí by bylo možné poskytnout podporu žákům nadaným a spolu s tím mít dostatek času pro práci s ostatními žáky. S podobnými obtížemi jsem se potýkala i později při výuce programování na základní škole Stanton Harcourt C of E Primary School či při vedení dalších zájmových kroužků se zaměřením na programování.

Jedním z cílů této diplomové práce je zhodnotit, zda je možné prostředí Scratch využít pro tvorbu rozsáhlejších projektů hodících se jako rozšiřující zadání pro nadané žáky. Dalším cílem je identifikace jednotlivých stádií tvorby projektu, jejich popis a identifikace obtíží, se kterými se mohou žáci při tvorbě podobných projektů setkat. Výstupem

práce by měl být metodický materiál, který doplní již existující kolekci materiálů o složitější a komplexnější zadání vhodná pro práci s nadanými žáky. Navržená rozšiřující zadání, využijí kombinaci různých konceptů osvojených tvorbou zmíněných jednodušších projektů. Zadání vytvořená jako součást této diplomové práce by mohla být inspirací pro učitele, jejichž žáci projeví nadání pro programování a kteří mají zájem se v něm dále zlepšovat.

1 Výuka programování a informatického myšlení

Výuka programování a informatického myšlení bývá v mnoha zdrojích spojována s konstruktivistickým přístupem. Velmi často se také objevuje jméno Seymour Papert, na kterého se velké množství autorů odkazuje právě ve spojitosti s konstruktivismem. Ve své knize *Mindstorms: children, computers, and powerful ideas* Papert kritizuje tehdejší počítačem podporovanou výuku, která je v jeho očích využívána pro tzv. programování dítěte. Počítač podle něho byl ve většině případů využíván jako nástroj pro poskytování cvičení přiměřené obtížnosti, zpětné vazby a jako zdroj informací [1, s. 19]. Pro Paperta měl však počítač ve vztahu k dítěti mnohem větší potenciál. Spolu s týmem ze společnosti Bolt, Beranek and Newman pracoval Papert na vývoji programovací jazyka Logo. Nejpopulárnější Logo prostředí obsahovalo želvu, původně robotické zařízení, které bylo umístěno na zemi a jehož pohyb byl ovládaný příkazy zadanými do počítače. Velmi záhy se tato robotická želva přenesla do virtuálního prostředí počítačové obrazovky, kde programátor s její pomocí kreslil různé tvary a obrázky [2].

V prostředí Logo mělo najednou dítě zcela nové postavení. Papert [1, s. 19–37] uvádí, že dítě je zcela záměrně již od předškolního věku postaveno do role programátora, který programuje počítač. Žáci tak učí počítač myslet, čímž rozvíjí i své vlastní myšlenkové procesy a utvářejí si své vlastní struktury. Pomocí změny pozice dítěte, které do té doby bylo převážně pouhým příjemcem informací, chce Papert dosáhnout vyšší efektivity učení, které bude pro žáka mimo jiné i atraktivnější. Papert se také odkazuje na jistou rigiditu systému, kdy se společnost snaží zachovat principy, které nemají žádné logické vysvětlení. Potýká se vlastně s podobnými problémy, které jsou často řešeny i nyní, když věci zůstávají takové, jaké byly, jen proto, že tomu tak bylo v minulosti. Papert [1, s. 95–105] se v knize *Mindstorms* ve velkém zabýval použitím programovacího jazyka Logo

pro výuku matematických představ u žáků, ale také pro rozvoj tzv. informatického myšlení.

Pojem informatické myšlení v návaznosti na Paperta popularizovala Jeannette M. Wing ve svém článku *Computational Thinking*. Wing uvádí, že informatické myšlení je jedna za základních dovedností, který by měl ovládat každý člověk [3, s. 1]. Dále pak informatické myšlení definuje pomocí několika klíčových vlastností [3, s. 2–3]:

- vytváření jednotlivých konceptů,
- fundamentální, nerutinní činnost,
- způsob, jakým lidé řeší problémy (nikoliv počítače),
- doplnění a provázání matematického a inženýrského myšlení,
- nejen jednotlivé artefakty, ale také nápady a myšlenky,
- pro všechny a všude.

Kolektiv autorů učebnice *Programování ve Scratch II* doplňuje definici informatického myšlení o činnosti, které jsou s ním spojené tak, jak je ve své prezentaci definoval Gerald Futschek [4 s. 3]:

- definovat, formulovat problém tak, aby byl řešitelný pomocí počítače, případně s použitím dalších technologických zařízení (hardwarových komponent, aj.),
- logicky uspořádat, organizovat a analyzovat data,
- (re)prezentovat data prostřednictvím abstraktních postupů (modelování, simulace),
- rozložit problém na podproblémy a automatizovat postupy řešení problému s použitím algoritmického popisu (řady uspořádaných kroků),
- identifikovat, analyzovat a implementovat možná řešení s cílem dosáhnout co možná nejefektivnější a nejúčinnější kombinace kroků, postupů a zdrojů,
- zobecnit a zajistit transfer procesu řešení problému na další problémy.

Důležitost informatického myšlení popisuje Miles Berry ve svém článku *Computational Thinking* pro Primary Schools Partnership March Newsletter, který vydává Roehamptonská Univerzita v Londýně. Berry ve svém článku [5] uvádí, že informatické myšlení a programování spolu úzce souvisí a v některých případech také využívá stejných strategií a konceptů. Mezi tyto koncepty řadíme například dekompozici, abstrakci, užití algoritmů a rozpoznávání vzorců či jejich zobecňování. Můžeme sledovat určitou podobnost mezi koncepty, které uvádí Berry [5], a činnostmi, které jsou zmíněny v učebnici *Programování ve Scratch II* [4 s. 3]. Berry [5] dále ve svém článku uvádí, že mnoho konceptů, které žáci využívají v programování, mají své využití i v dalších předmětech či životních situacích.

Rudolf Pecinovský [6] se shoduje s Papertem [1 s. 13] v názoru, že děti se s programováním mohou setkávat již ve velmi raném věku. Pecinovský [6] také podobně jako Berry [5] uvádí, že programování nás provází celý život. Toto tvrzení podkládá využitím příkladu postupu, který použije batole, aby dosáhlo na hračku s pomocí dostupného nástroje. Pecinovský také uvádí, že se spolu s rozvojem úrovně myšlení jedince zvyšuje také složitost „programů“, které v životě provádíme. Po jednoduchých postupech se přidávají procedury (prosba sourozence o pomoc s úkonem, který sám nezvládnu), funkce (dotázání se kolemjdoucího na čas), podmínky, proměnné apod.

Jako jeden z důležitých bodů při výuce programování a rozvoji informatického myšlení můžeme napříč zdroji identifikovat vlastní aktivitu žáka a jeho vytváření vlastních teorií a domněnek, které pak na základě experimentu potvrzuje či vyvrací. Papert [1, s. 131–134] se ve spojitosti s konstruktivistickým přístupem ke vzdělávání odkazuje na práci Jeana Piageta. Dále v této části také upozorňuje na to, že mylné i správné domněnky žáků jsou stejně důležité. Umožnění žákům experimentovat můžeme také nalézt mezi doporučeními pro práci se žáky v učebnici *Programování ve Scratch II* [4, s. 9].

Dalším důležitým faktorem při výuce programování je pro Paperta [1, s. 95–105] a Pecinovského [6] jednoduchá syntaxe zvoleného programovacího jazyka. Zejména pokud žáci začínají s programováním, měl by být programovací jazyk co nejjednodušší. Jednoduchý jazyk pak žákům umožní soustředit se na tvorbu programu místo na správnou syntaxi. Zároveň by žáky také mohla složitá syntaxe rychle odradit a oni by tak ztratili touhu řešit problémy, které jim v daném jazyce předkládáme.

2 Jednotlivé pojmy použité v práci

Blok – Kód tvořený v jazyce Scratch je složen z mnoha jednotlivých částí tvaru připomínajícího puzzle. Blok označuje nejmenší stavební jednotku kódu. Ve standardním programování můžeme blok přirovnat k příkazu.

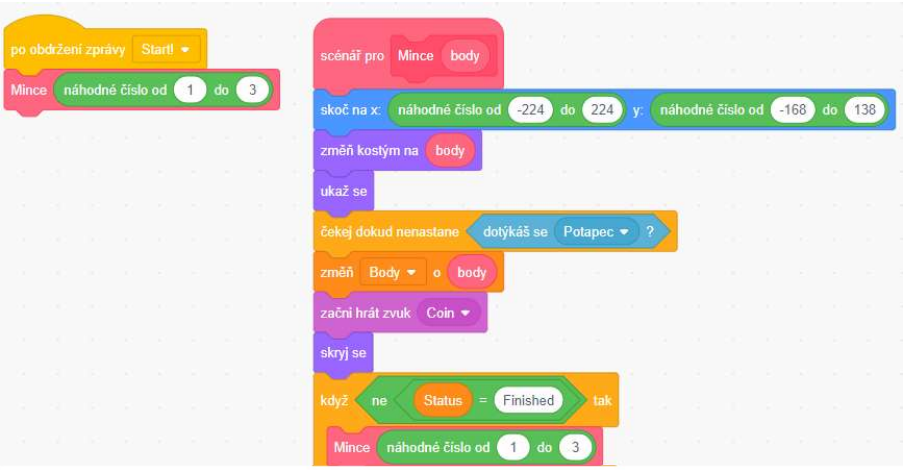
Koncept – Během své práce ve Scratch se žáci postupně seznamují s různými funkcemi jednotlivých bloků, ale také celých scénářů. Určité sekvence bloků se pravidelně objevují v programech a plní určitou funkci. Tyto pravidelně se objevující sekvence bloků můžeme označit jako koncepty. Tyto koncepty by si žáci měli v průběhu vlastní aktivity osvojit a následně být schopni je aplikovat i v jiných projektech či ve změněných podmínkách.

Scénář – Program ve Scratch je složen z jednotlivých scénářů. Jedná se o spojené sekvence bloků, které společně tvoří část programu. Zpravidla je projekt složen z více scénářů. Existují však i přístupy, kdy se autor snaží vměstnat všechny části programu do jednoho scénáře, což však často vede k jeho nepřehlednosti a častějšímu chybování. Scénář můžeme považovat za alternativu algoritmu.

Program – Program jsou všechny scénáře v projektu.

Projekt – Projekt je produkt vytvořený v prostředí Scratch. Jedná se o shrnující pojem pro všechny postavy, pozadí a scénáře využité pro daný produkt. Může se jednat o hru, animaci, hudební či výtvarné dílo apod.

Vlastní blok – Vlastní blok je speciální druh scénáře. Často se využívá pro zpřehlednění kódu či zamezení opakování jeho části v rámci jedné postavy. Můžeme ho přirovnat k definici metody. Pod hlavičkou vlastního bloku je umístěn scénář, který je v případě potřeby volán svým vlastním volacím blokem. Součástí definice a následného volání vlastního bloku může být také předání parametrů pro provedení daného scénáře. Praktické využití vlastního bloku s předáním parametru je znázorněno na obrázku 1.



Obrázek 1 Vlastní blok s předáním parametru

3 Scratch a jeho využití

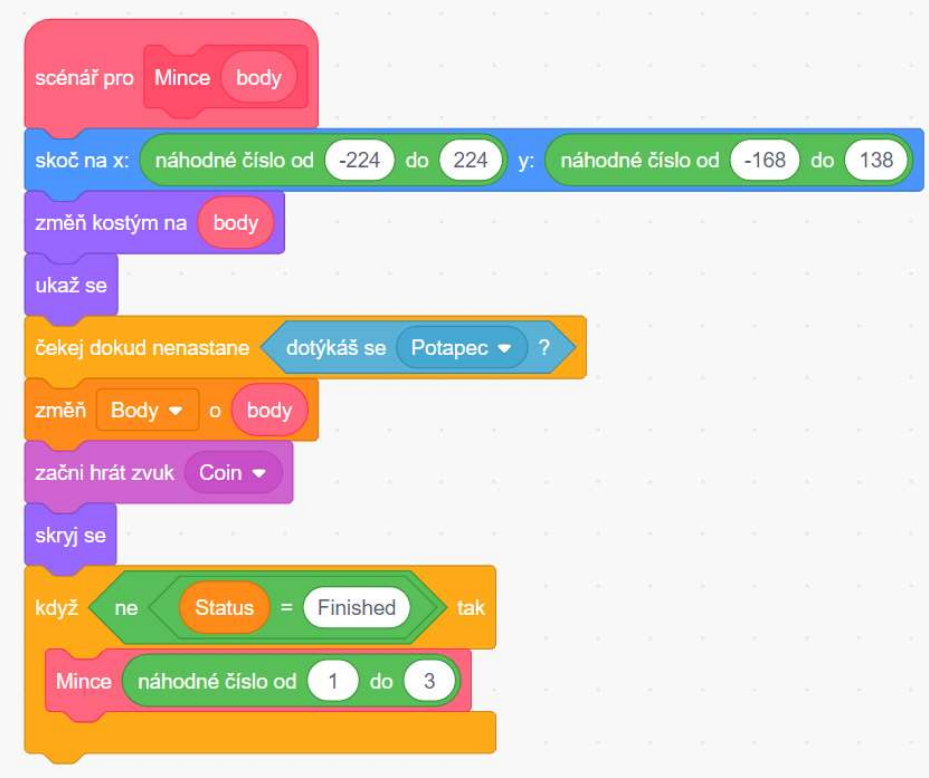
Tato kapitola se věnuje prostředí Scratch jako takovému. Stručně zde popisuji, co Scratch je a k čemu je možné ho využít. Zejména se zaměřuji na vlastnosti jednotlivých bloků, ze který je výsledný kód skládán.

3.1 Co je to Scratch?

Scratch je výukový blokový programovací jazyk určený především pro děti a mládež [7]. Díky různobarevnému prostředí, které disponuje velmi intuitivním ovládáním, se v něm děti často začnou učit i samy. Pro děti, které ještě neumí číst, existuje varianta Scratch Junior. Tato verze Scratch má omezené funkce, které jsou reprezentovány obrázky, což umožňuje i malým dětem vytvářet své první programy. Tato diplomová práce se bude zabývat šířeji známou verzí Scratch pro větší děti, které již číst umějí. Programování v tomto prostředí nesestává z psaní komplikovaných konstrukcí, ale ze skládání jednotlivých bloků.

Pro lepší představu o tom, jak programování ve Scratch probíhá, je na obrázku 2 vybraný scénář obsahující velké množství různých bloků. Z obrázku je patrné, že v tomto prostředí skutečně není při programování třeba psát dlouhé texty, jelikož celý kód je vytvářen skládáním jednotlivých bloků k sobě. Díky této vlastnosti je minimalizována možnost překlepů a žáci se mohou plně soustředit na tvorbu kódu jako takového.

Dále na tomto obrázku můžeme pozorovat, že různé bloky mají odlišné barvy. Tyto barvy jsou blokům přiřazeny podle kategorie, do které patří. O těchto kategoriích uvedu více informací v následující podkapitole Vlastnosti prostředí Scratch.



Obrázek 2 Ukázkový scénář v prostředí Scratch

Scratch je dobrým nástrojem pro tvorbu jednoduchých her, animací či příběhů. Díky své přehlednosti, široké nabídce funkcí a možnosti žáků okamžitě pozorovat výsledky své práce, vytváří velmi motivující prostředí pro výuku programování. K podobnému závěru dochází i učitel gymnázia Petr Němec, který se o svou zkušenost podělil ve formě článku na webu rvp.cz [8].

3.1.1 Vlastnosti prostředí Scratch

Jedna z předností prostředí Scratch je to, že je možné ho využívat v prohlížeči a není ho třeba instalovat. Existuje však i desktopová verze. Obě verze jsou k dispozici se všemi svými funkcemi a rozšířeními zcela zdarma. Celé prostředí Scratch bylo také přeloženo do více než 50 jazyků (včetně jazyka českého) a další jazyky prostupně přibývají [9].

Celé prostředí působí celkem přehledně i přes velké množství funkcí, které nabízí. Pro lepší přehlednost využívá Scratch různých vizuálních dělení. Jedno z těchto dělení je

rozdělení bloků do devíti základních kategorií: Pohyb, Vzhled, Zvuk, Události, Ovládání, Vnímání, Operátory, Proměnné a Moje bloky. K těmto kategoriím je možné snadno doplnit další, které v nově vytvořeném projektu nejsou zpočátku zobrazené, jako je Hudba, Pero, Vnímání videa, Text na hlas a Překlad. Dále je možné Scratch využít pro vytváření programů ovládající další programovatelná zařízení a stavebnice jako je micro:bit, Lego Mindstorms, Lego WeDo či Makey (obdoba stavebnice Adruino). Je také možné využít zmíněná zařízení pro ovládání postav v projektu. Každá kategorie má přirazenou vlastní barvu, podle které si žáci snadno zapamatují, k čemu který blok slouží.

Samotné bloky mají kromě barvy také rozdílné tvary, které indikují jejich správné použití. Podle tvaru rozlišujeme bloky na pět kategorií: startovací blok, základní stavební blok, koncový stavební blok, návratový blok a logický blok. Na obrázku 3 jsou zobrazeny tvary jednotlivých kategorií bloků v pořadí, v jakém jsou zmíněny výše.

Obrázek 3 Jednotlivé tvary bloků

Startovací blok, jak již jeho název napovídá, se objevuje pouze na počátku scénáře, který zahajuje. Žák má možnost si zvolit různé akce, které daný scénář spustí. V nabídce je například reakce na kliknutí na zelenou vlajku, která je implementována jako základní způsob spouštění celého programu, dále také reakce na stisk různých kláves, obdržení zprávy, změnu pozadí apod.

Základní stavební blok má tvar připomínající puzzle. Na horní straně je lehce vykrojen, což značí jeho připojitelnost k již existujícím blokům či scénářům, a na spodní straně má výstupek, na který je možné připojit další stavební bloky. Většina bloků v prostředí Scratch je právě tohoto typu. S výjimkou operátorů můžeme tento typ bloku najít v každé kategorii bloků.

Koncový stavební blok je na spodní hraně rovný bez výstupku, jelikož na něj již není možné přímo připojit žádný další blok. Nemusí to však nutně znamenat, že tento blok bude vždy posledním blokem scénáře. Na obrázku 4 jsou zobrazeny obě možné pozice koncového stavebního bloku. Koncový stavební blok se může nacházet i ve středu scénáře, pokud je umístěn například na konci podmínkového bloku.

Obrázek 4 Příklady umístění koncového bloku

Návratový blok v sobě obsahuje hodnotu, která se může lišit v závislosti na tom, co do nich žák vloží. Návratové bloky jsou dvojího typu: první typ je **návratový blok operátoru**. Důležitou vlastností návratových bloků je jejich oválný tvar, který žákům jasně značí, že je možné tyto bloky vložit do oválných otvorů v ostatních blocích. Do těchto bílých oválných prostorů v různých blocích je možné, vkládat jak návratové bloky, tak přímo dopisovat hodnoty. Hodnotu návratového bloku operátoru není možné využít na více místech v projektu s odkazem na tento blok a slouží pouze pro lokální použití, Pro možnost využití jedné hodnoty na více místech je třeba vytvořit **proměnnou** (tmavě oranžová kategorie), která je druhým typem návratového bloku. Do vytvořené proměnné je možné mimo jiné vložit i hodnotu z návratového bloku operátoru. Na obrázku 5 můžeme vidět návratový blok operátoru, do kterého je vložen další návratový blok obsahující

proměnnou. Tato vlastnost nám umožňuje vkládat návratové bloky do sebe a vytvářet tak složitější konstrukce v rámci jednoho řádku kódu.

Obrázek 5 Návratový blok operátoru s vloženou proměnnou

Posledním druhem bloku je **logický blok**, jenž může nabývat hodnot 1 a 0 (True/False). Tento blok je opět charakteristický svým tvarem a je možné ho vkládat do šestiúhelníkových otvorů v blocích s podmínkou. Na obrázku 6 je zobrazen logický blok se dvěma vstupy. V některých případech je možné do sebe vkládat i jednotlivé logické bloky. Jedná se o situace, kdy potřebujeme řetězit jednotlivé logické bloky do jednoho řádku či jejich výstup negovat. Příklad takového využití je vyobrazen na obrázku 7.

Obrázek 6 Logický blok se dvěma vstupy

Obrázek 7 Zřetězení logických bloků

4 Již existující Scratch Projekty

Prostředí Scratch je díky své dostupnosti velmi populární a v současné chvíli obsahuje obrovské množství projektů. Nyní se jedná o necelých 74,5 milionu sdílených projektů [10]. Mezi sdílenými projekty je velké množství na první pohled slibně vypadajících projektů, které však po otevření nefungují.

Podařilo se mi však nalézt i velké množství funkčních projektů inspirovaných různými již existujícími hrami. Velmi populární jsou v tomto ohledu hry Minecraft, Super Mario či Fortnite. Dále se často objevují projekty inspirované hrami pro mobilní telefony jako je například Geometry Dash, Solitaire, či různé logické hry.

Téměř pro každou existující hru je možné najít variantu na Scratch. Hru Osadníci z Katanu, která byla mým původně zamýšleným projektem pro tuto diplomovou práci, jsem si zvolila právě z důvodu, že na ni existuje velké množství adaptací ve formě projektů, ale žádná z nich není plně funkční. Jak však vysvětluji níže, nebylo ani pro mne možné tuto hru vytvořit.

Vzhledem k velkému množství her na trhu je obtížné vymyslet hru s úplně novým a neotřelým konceptem. Srovnatelně obtížné je pak takovou hru nalézt ve zdánlivě nekončícím seznamu sdílených projektů v prostředí Scratch. Toto prostředí bohužel nenabízí žádnou formu dělení projektů podle tématu, žánru či jiné vlastnosti a je možné vyhledávat pouze podle názvu projektu či jména jeho autora.

Kromě posledního pracovního listu, který je do značné míry silně individuální, a není tedy možné předem znát povahu žákem zvolené hry, existuje ke všem ostatním zadaným projektům velké množství projektů podobných. Žáci se tedy mohou inspirovat již vytvořenými projekty, mohou je s těmi svými porovnávat a využít toto porovnání pro vylepšení vlastních her.

Pro rešerši již vytvořených her jsem volila takové hry, které jsou z mého pohledu možné ve výuce použít. Jedná se například o hry, které jsou svým konceptem nějak zajímavé, a je možné na nich tento koncept žákům ukázat. Snažila jsem se volit hry na takové úrovni, aby je žáci mohli zvládnout naprogramovat sami. Záměrně jsem nevolila hry, které jsou velmi komplexní, protože pro žáka na základní škole je pak jejich fungování složité na pochopení a vývoj takové hry by byl velmi zdlouhavý. Je však možné tyto hry použít jako motivační ukázky. Žáci mohou nahlédnout do kódů uvnitř takto složitých her, aby si vytvořili představu, kam až je možné své dovednosti rozvinout a jak působivý pak může být výsledek jejich práce. Ačkoliv některé mnou zvolené projekty obsahují celou řadu programátorských chyb a nepřesností, je možné je právě pro tyto chyby ve výuce velmi dobře využít.

4.1 Super Mario Brothers

Jedním z mnoha populárních témat pro tvorbu her v prostředí Scratch je Mario. Postava Maria se poprvé objevila v legendární arkádě ze studia Nintendo s názvem *Donkey Kong* a navzdory očekávání se stala velmi populární. Nyní je Mario jednou z nejznámějších animovaných postav vůbec a hry a předměty s Mario tematikou jsou jedny z nejprodávanějších na světě [11].

Na webu Scratch se nachází tisíce her s touto ikonickou postavou, což je jeden z důvodů, proč jsem se rozhodla tuto hru ve své diplomové práci uvést. Projekt, který budu dále rozebírat, je dostupný na adrese https://scratch.mit.edu/projects/2176968/.

Autor tohoto projektu zvolil relativně netradiční postup při jeho tvorbě. Po otevření editačního prostředí je zřejmé, že pro celý projekt využil pouze jednu postavu, do které vložil jeden dlouhý scénář. Tento scénář má na starost celý běh hry, díky čemuž je značně nepřehledný. Hra má celkem 16 úrovní, kterými hráč postupně prochází.

Při standardním přístupu ke tvorbě projektu jsou pro interakci postav často využívány bloky detekce doteku postav. Vzhledem k tomu, že autor využil pouze jednu postavu, jsou tyto bloky nahrazeny detekcí doteku barvy (viz obrázek 8). Další zajímavý koncept, který autor používá, se týká změny pozadí. Místo klasické změny pozadí jsou jednotlivá pozadí uložena jako kostýmy postavy, které jsou otiskovány na plochu hry.

Obrázek 8 Super Mario Brothers – ukázka negované detekce doteku barvy

Projekty tohoto typu by mohly být velmi zajímavé pro žáky k analýze. Jelikož úkon tohoto typu již vyžaduje vysokou míru abstrakce, hodil by se spíše pro žáky starší. Na projektu můžeme demonstrovat právě již zmíněné atypické přístupy k řešení dílčích problémů. Žáci se také mohou pokusit o navržení jiného řešení, které by využívalo více scénářů, s cílem zjednodušit strukturu kódu. Domnívám se, že vést žáky k tomuto stylu programování není zcela šťastným krokem a považovala bych za velmi důležité při práci s tímto projektem, aby bylo žákům zdůrazněno, že se nejedná o typický přístup k programování. Tento přístup žáky nevede k dělení problémů na dílčí podproblémy, což vnímám jako velmi důležitou část programování a informatického myšlení obecně.

Z programátorského hlediska je projekt zpracován relativně dobře. Nachází se zde však několik drobných nedostatků, které pravděpodobně pramení právě z přístupu jedné postavy a jednoho scénáře. Jako jeden z hlavních problémů bych označila problematické skákání. V okamžiku, kdy postavička Maria vyskočí a dotkne se boční hrany překážky, na kterou se snaží vyskočit, stává se, že se místo spadnutí dolů postava postupně přesune na horní hranu dané překážky.

Na obrázku 9 je zobrazena část kódu, která je pravděpodobně zodpovědná za nelogické šplhání postavy po bočních hranách překážek. Vzhledem k přístupu „jedna postava, jeden scénář“ nelze bez konzultace s autorem jednoznačně určit, zda je tato vlastnost v některých případech žádoucí či nikoliv, a zda je tedy možné ji z kódu jednoduše odstranit.

Obrázek 9 Super Mario Brothers – potenciálně problematická část kódu

Dále se také postava objevuje levitující nad zemí, což by mohlo být označeno jako pouhá estetická vada. Tento drobný nedostatek však hře ubírá na celkovém dojmu. Problém levitující postavy by však bylo velmi jednoduché opravit. Vzhledem k tomu, že každá úroveň má stejný počáteční bod (levý dolní roh obrazovky), stačilo by v kódu na obrázku 10 změnit souřadnici y z -96 například na hodnotu -112.

Obrázek 10 Super Mario Brothers – kód pro nastavení úvodní pozice postavy

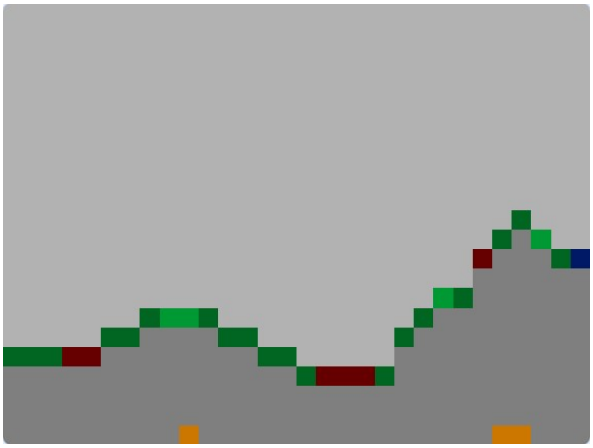
4.2 Minecraft Platformer

Další populární námět pro tvorbu projektů je hra Minecraft. Zatímco původní hra Super Mario, která byla předlohou pro první projekt, je vytvořena ve 2D prostředí, Minecraft je v prostředí 3D. V tomto projektu dochází ke změně originální perspektivy, jelikož projekt vyobrazuje herní prostředí pouze ve dvou dimenzích. Hotový projekt je dostupný na adrese https://scratch.mit.edu/projects/118428609/. Na rozdíl od předchozího projektu se autor nesnaží o kompaktní řešení jedné postavy a jednoho scénáře.

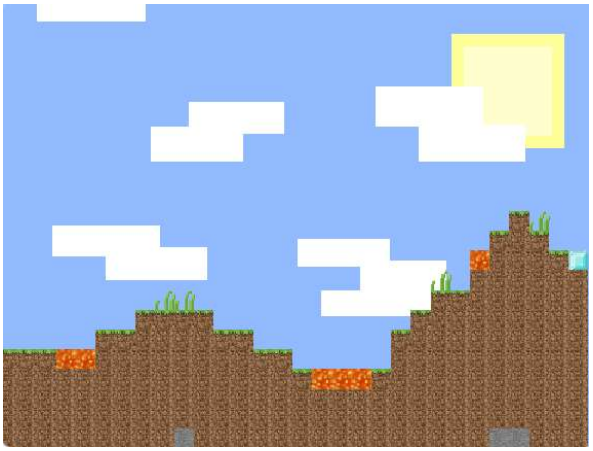
Stejně jako u předchozího projektu se jedná o tzv. skákačku neboli hru, kdy se hráč snaží dostat se z jednoho místa na obrazovce na druhé a při tom překonat různé překážky. Hra Minecraft jako taková však skákačkou není, a tak je tento projekt originální hrou pouze tematicky inspirovaný. Na Scratch existuje i několik projektů, které se snaží imitovat Minecraft v originální podobě, či využívají stejný koncept ve 2D prostředí. Tyto projekty jsou však velmi komplikované a příliš se nehodí pro demonstraci jednotlivých konceptů.

Tento projekt je do výuky zajímavý z několika důvodů. Prvním zajímavým prvkem je zvolený postup autora při generaci terénu pro každou úroveň. Autor nemá připravené úrovně ve finální podobě, která by obnášela velké množství kopírování jednotlivých bloků, ze kterých je hra vytvořena. Pro zjednodušení si autor připravil podkladový vzor, který později při generaci úrovně prochází a otiskuje na něj jednotlivé textury automaticky. Pro každou texturu má autor nastavenou jinou barvu, podle které pak hra pozná, jaký blok na dané místo má otisknout. Na obrázku 11 můžeme vidět předlohu pro generaci textury první úrovně.

Na obrázku 12 pak můžeme vidět tutéž úroveň po generaci textur. Zajímavé jsou zejména bloky značené světle zelenou barvou, které na rozdíl od všech ostatních na sobě mají stébla trávy, a nejsou tak omezené pouze na rozměr vlastní kostky. Na těchto prvcích můžeme žákům ukázat, že je možné relativně jednoduše jejich projekty ozvláštnit a udělat je tak vizuálně atraktivnějšími.

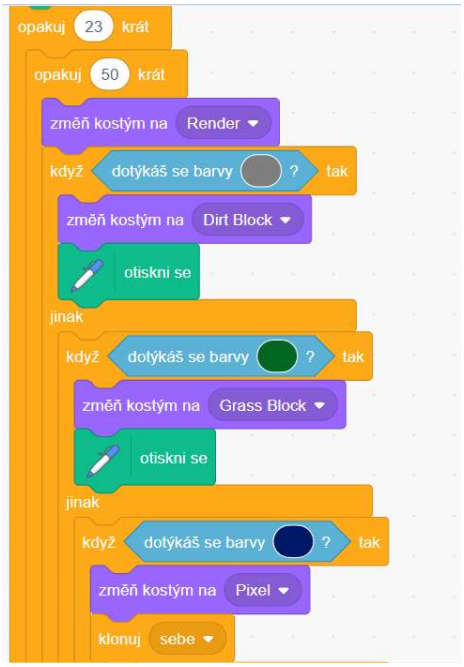


Obrázek 11 Minecraft Platformer – předloha pro generaci textury úrovně



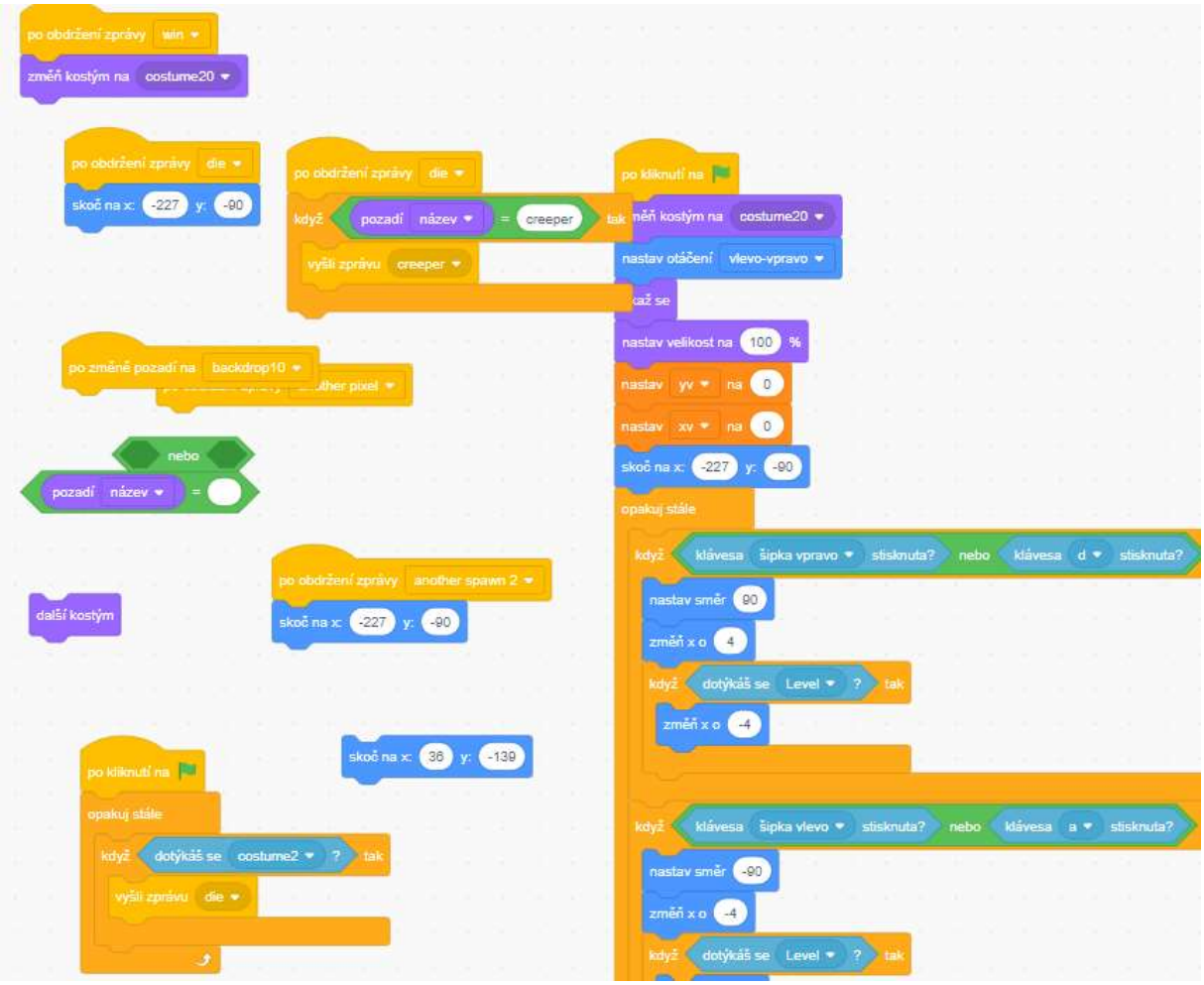
Obrázek 12 Minecraft Platformer – finální podoba vygenerovaného pozadí úrovně

Na obrázku 13 je zobrazena část kódu užitá pro generaci jednotlivých úrovní. Z počtů opakování vnořených cyklu je patrné, že jedna úroveň je rozdělena na síť čtverců, která je 23 čtverců vysoká a 50 čtverců široká. Jeden čtverec má rozměr 16 × 16 pixelů (což je patrné na konci tohoto scénáře, který zde již není zobrazen). Zajímavostí je, že ve hře Minecraft, která je pro tento projekt předlohou, mají všechny bloky také tuto velikost (16 × 16 pixelů). Dále zde můžeme pozorovat, že postava, která generuje terén úrovně, má definované kostýmy pro jednotlivé barvy v předloze. Určité barvy (zde například tmavě modrá) reprezentují interaktivní bloky a na svou pozici se místo pouhého otisknutí postava klonuje.



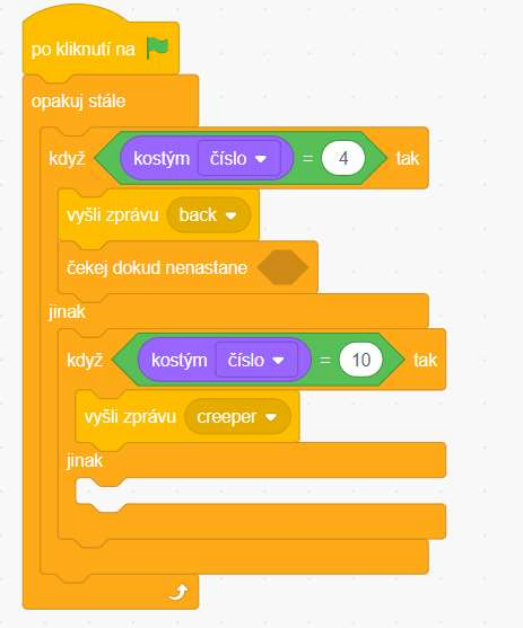
Obrázek 13 Minecraft Platformer – část kódu pro generaci textur

Tento projekt je ovšem také dobrý materiál pro ukázkou výhod uspořádaného kódu. Napříč jednotlivými postavami můžeme v kódu nalézt různě se překrývající scénáře a také bloky, které nejsou použity. Tyto prvky dělají projekt nepřehledným. Při tvorbě složitějšího projektu je však přehlednost kódu jednou z klíčových vlastností pro úspěšné dokončení projektu. Můžeme tedy tuto příležitost využít pro to, aby se žáci pokusili projekt zpřehlednit a zafixovali si tak postupy, které sami budou při tvorbě vlastních projektů potřebovat. Dále můžeme na obrázku 14 vidět, že autor nepojmenoval všechny kostýmy tak, aby věděl, co na daném kostýmu je a využívá názvy typu kostým1 apod. Opět se zde nabízí možnost žáky nechat vymyslet vhodné názvy pro dané kostýmy a zvýšit tím přehlednost projektu.



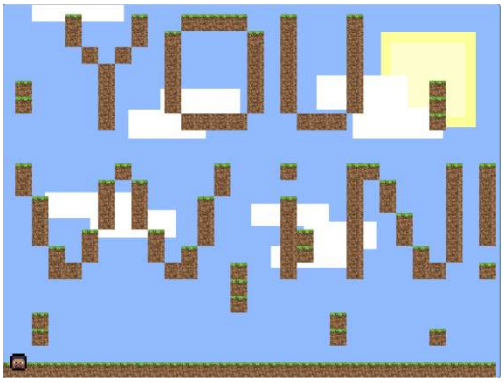
Obrázek 14 Minecraft Platformer – neuspořádaný kód a zbytečné nevyužité bloky

Přestože hra funguje, můžeme v projektu na mnoha místech najít nadbytečné či nevhodně zvolené bloky. Na obrázku 15 můžeme vidět nevhodně použitý podmínkový blok typu E. V případě, že autor nepotřebuje individualizovanou reakci na více možných situací, může použít obyčejnou jednoduchou podmínku „když“. V tomto případě je třeba poslat zprávu *back*, když má kostým dané postavy číslo 4, a zprávu *creeper*, když má kostým dané postavy číslo 10. Z kódu projektu však není zcela jasné, zda je vůbec nutné obě tyto zprávy odesílat, a je tedy možné, že se jedná o celý nadbytečný scénář. Na obrázku také můžeme vidět nadbytečný blok čekání na splnění podmínky. V případě nevyplnění podmínky, je hodnota bloku stále „pravda“ a nemá tedy ve scénáři žádný význam. Zároveň by bylo vhodné, kdyby autor eliminoval použití nekonečných cyklů, které taktéž můžeme vidět na obrázcích 14 i 15.



Obrázek 15 Minecraft Platformer – nevhodně využitý podmínkový blok typu E

Posledním drobným ale důležitým nedostatkem tohoto projektu je málo rozpracovaný konec hry. Když hráč dokončí všechny úrovně, objeví se mu pouze obrazovka s textem „You win!“ (v překladu „Vyhrál jsi!“). Snímek této obrazovky je možné vidět na obrázku 16. V předchozím projektu se po úspěšném dokončení hry hráč dozví, jak dlouho mu hra trvala, a vidí animaci princezny děkující Mariovi za záchranu. Pro zvýraznění úspěšného dokončení hry by bylo dobré, aby byl i tento projekt zakončen zajímavě. Vzhledem k tomu, že se jedná o třetí a finální hru ze tří, které autor na téma Minecraftu vytvořil, nabízí se například možnost vytvoření animace zabití draka, což je prvním a základním cílem hry, kterou byl tento projekt původně inspirován.



Obrázek 16 Minecraft Platformer – závěrečná obrazovka

4.3 Tiles Logic Game

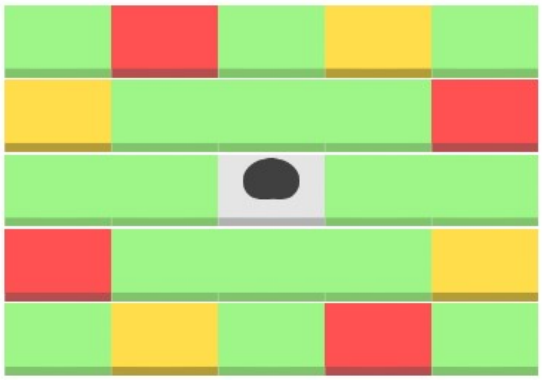
Další projekt, který by mohl být vhodný pro použití ve výuce, nese název Tiles Logic Game. Tento projekt je dostupný na adrese https://scratch.mit.edu/projects/270685846/. Jedná se o logickou hru, ve které musí hráč s kuličkou postupně a ve správném pořadí skočit na všechna určená políčka. Každé políčko má předem stanovené, kolikrát na něj hráč může skočit, a je tak nutné, aby hráč předem rozmýšlel, kudy je nejlepší se vydat.

Tato hra je mírně komplikovanější než předchozí dvě, ale stále by ji mohl nadaný žák zvládnout naprogramovat. Daný koncept by bylo možné výrazně zjednodušit, aby hra nebyla tak náročná. Autorka se ve velké míře soustředila na estetickou stránku projektu, což je na výsledném produktu velmi znát. Všechna tlačítka ve hře reagují na kurzor myši zvětšením či pohybem. Mizející políčka využívají efekt *ghost* (v překladu *duch*), který nastavuje dané postavě průhlednost, a společně s drobným pohybem dolů vytvářejí dojem klesajícího políčka. Právě tyto estetické prvky, které však projektu přidávají na složitosti, by bylo možné omezit na minimum a snížit tak jeho obtížnost.

Již na první pohled je zřejmé, že hra je programátorsky velmi dobře zvládnutá. Na rozdíl od ukázky s tematikou Minecraftu jsou všechny scénáře v postavách srovnané a nepřekrývají se. V projektu se nachází pouze jeden nevyužitý blok, který autorka pravděpodobně zapomněla odstranit. Využité proměnné a seznamy mají logické pojmenování stejně jako odesílané zprávy, které jsou použity pro ovládání hry. Z programovacího hlediska jistě stojí za zmínku autorčina práce se seznamem pro generování jednotlivých obtížností propojeným s klonováním jedné postavy políčka.

Způsob, jakým autorka hru naprogramovala, umožňuje velmi jednoduché přidávání nových úrovní hry. Pro každý typ políčka si autorka zvolila hodnotu od 1 do 4, přičemž se každá úroveň skládá z 25 políček uspořádaných v mřížce 5 × 5. V závislosti na hodnotě v sekvenci čísel pro každou úroveň je políčku přiřazena správná pozice v mřížce a barva, která hráči značí kolikrát na dané políčko může skočit. Například sekvence „4243434442441442444343424“ se pomocí kódu z obrázku 17 převede na úroveň vyobrazenou na obrázku 18.

Obrázek 17 Tiles Logic Game – kód pro generaci úrovně



Obrázek 18 Tiles Logic Game – ukázka vygenerované úrovně

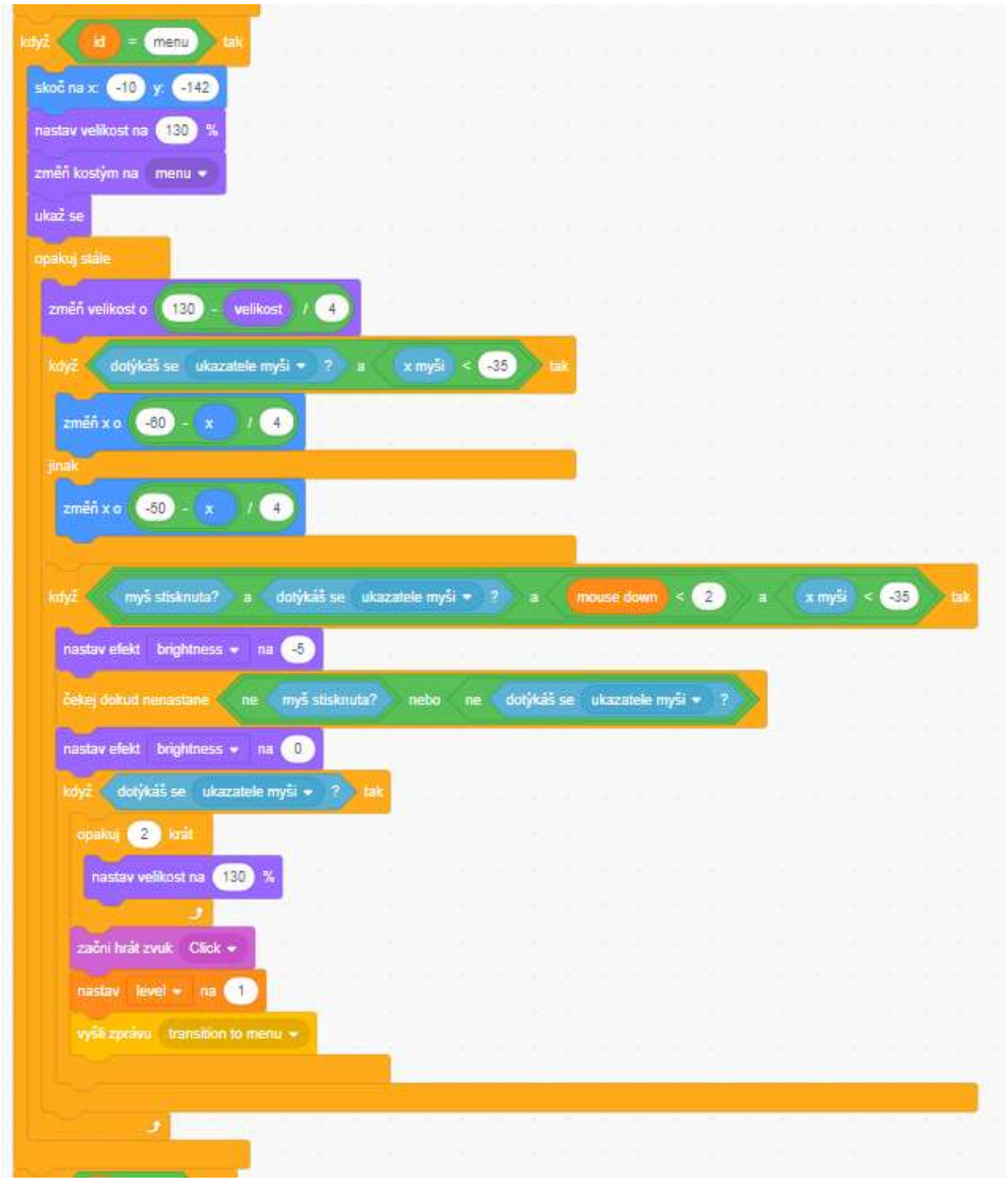
Tento projekt je vhodným materiálem pro demonstraci práce s klony. Autorka zde klony využívá nejen pro jednotlivá políčka ve hře, jak již bylo zmíněno, ale také pro všechna tlačítka. Efektivní využití klonování umožňuje v projektu využít menší množství postav a také celkově menší množství použitých scénářů. Velmi zajímavé je právě využití klonů pro práci s ovládacími tlačítky. Autorka zvolila velmi decentní minimalistický vzor, který je vyobrazen na obrázku 19.

Obrázek 19 Tiles Logic Game – úvodní ovládací panel tlačítek

Všechna tři tlačítka na obrázku mají při klonování přiřazené ID, podle kterého je později nastavena jejich pozice a také jejich chování při kontaktu s kurzorem myši. Celkem využívá autorka osm tlačítek v jedné postavě. Pokud hráč určitou úroveň zkazí, objeví se podobně rozložená nabídka tlačítek (viz obrázek 20). Důležité je zde ovládání zvuku, které je z neznámého důvodu zobrazeno na opačné straně než na obrazovce úvodní.

Obrázek 20 Tiles Logic Game – ovládací panel tlačítek v průběhu hry

Na obrázku 21 je vyobrazen kód pro tlačítko po startu klonu, jehož ID je *menu*. Graficky je tlačítko *menu* zastoupeno symbolem domečku (vlevo na obrázku 20). Jedná se o část scénáře, který sdílí všechny klony tlačítek. Při celkové délce téměř 100 řádků se jedná o velmi dlouhý scénář pro relativně jednoduchý projekt. Polovina kódu v tomto scénáři je však pro všechna tlačítka téměř shodná. Jedná se především o části kódu, které mají za úkol zajistit reakci tlačítek na kontakt s myší. Liší se pouze orientace symbolů pro ostrou nerovnost (např. na řádku 8) a znaménka některých přímo vepsaných hodnot. Díky velkému množství opakovaných sekvencí bloků by bylo vhodné projekt zjednodušit.

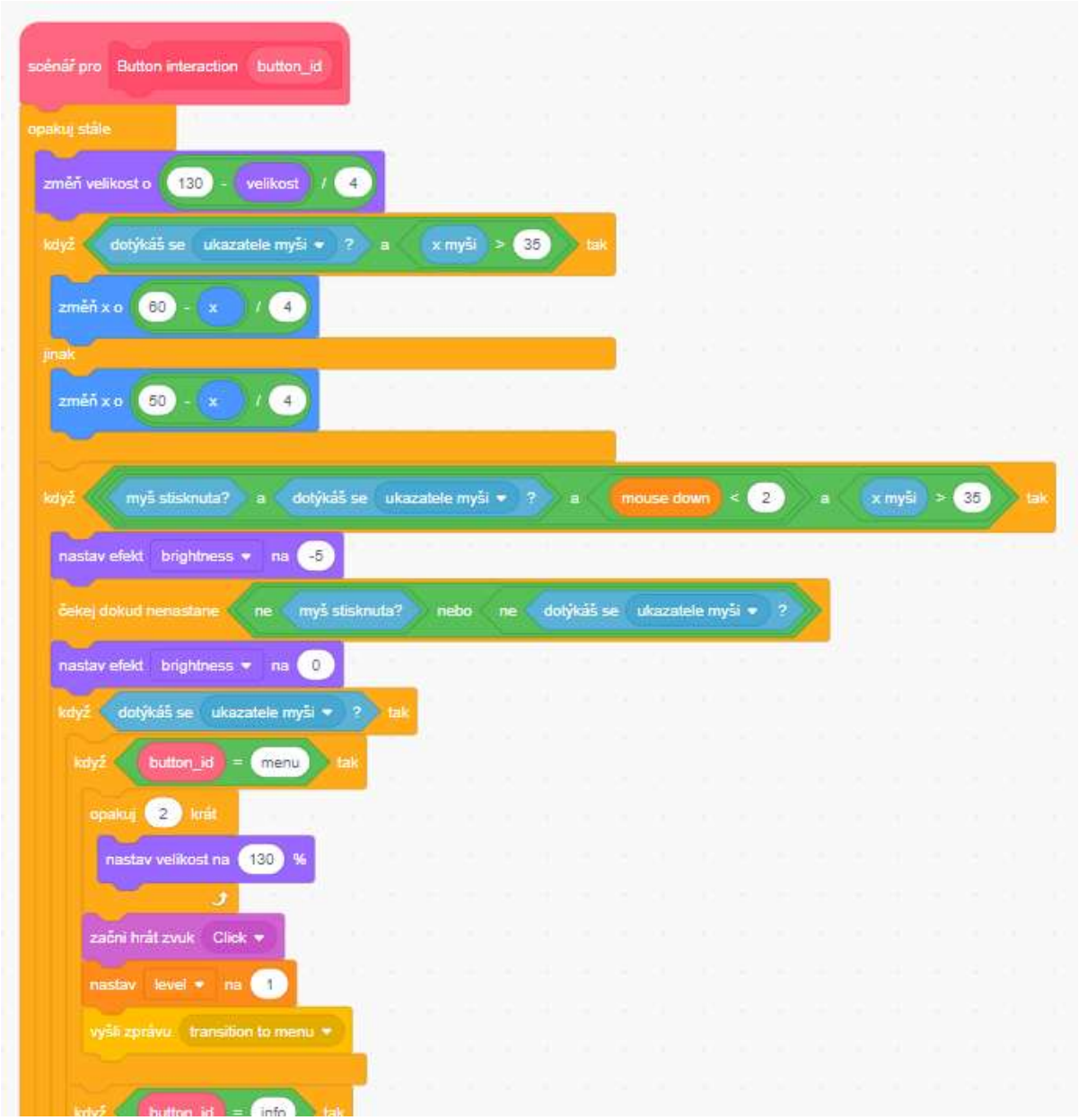


Obrázek 21 Tiles Logic Game – scénář pro tlačítko menu

Jako část řešení se nabízí možnost sloučení kódů pro tlačítka *menu* a *info*. Vzhledem k tomu, že celá sestava tlačítek má svůj střed na souřadnici x = 0 a tlačítka jsou umístěna souměrně podle osy y, nepředstavuje změna znaménka u přímo vepsaných hodnot problém. Problematická je však změna orientace ostré nerovnosti v logických blocích. Při použití současného rozložení tlačítek tak není možné kód zjednodušit. Pokud by však autorka zachovala umístění tlačítka pro ovládání zvuku vlevo a vpravo by se objevovalo buď tlačítko *menu* či tlačítko *info*, bylo by možné kód efektivně zjednodušit. Kód pro zobrazení a interakci tlačítek pro ovládání hlasitosti by zůstal zachován pouze ve své levé orientaci, čímž by zároveň došlo ke snížení počtu tlačítek. (V současném návrhu používá autorka odlišná tlačítka pro levé a pravé zobrazení.) Dále bychom pro zefektivnění kódu potřebovali nahradit kód z obrázku 21 kódem v obrázku 22, ve kterém bychom po umístění tlačítka na správné místo zavolali vlastní blok pro interakci hráče a tlačítka. V tomto vlastním bloku by bylo třeba přenést parametr *button_id*, jelikož je třeba odlišit akci, kterou tlačítko *menu* a *info* provede po kliknutí.

Obrázek 22 Tiles Logic Game – upravený scénář pro tlačítko menu

Kód pro interakci tlačítek je vyobrazen na obrázku 23. Na jeho konci můžeme vidět specifický kód pro interakci tlačítka *menu*. Analogicky je dále ve scénáři umístěn specifický kód pro interakci tlačítka *info*. Navržené změny jsem úspěšně implementovala do původního projektu pro otestování zachování plné funkčnosti hry.



Obrázek 23 Tiles Logic Game – upravený scénář pro interakci hráče s tlačítky

Podobně jako autor hry inspirované hrou Super Mario, autorka tohoto projektu ukládá počet hráčů, kteří úspěšně dokončili všechny úrovně hry bez jejich přeskakování. Můžeme tedy tento projekt také využít pro praktickou ukázku práce s cloudovými proměnnými. V tomto projektu se však z blíže neurčených důvodů hodnota dokončení hry přestala aktualizovat a zastavila se na hodnotě 437. Při rozboru projektu se mi nepodařilo odhalit příčinu tohoto problému.

5 Jednotlivá stádia procesu tvorby hry

Tvorbu hry jsem rozdělila na dvě základní fáze. První fází je teoretická příprava hry, kde se žák rozhoduje, jakou hru naprogramuje. Zároveň by si v této fázi měl žák rozmyslet, jaká budou ve hře pravidla, jaký bude mít cíl. Po stanovení těchto základních vlastností projektu se může žák přesunout do druhé fáze, která již obsahuje vlastní programování.

5.1 Teoretická příprava hry

Prvním krokem před vlastní tvorbou hry by měla být teoretická příprava hry. Žák by si měl zvolit, jaký druh hry chce programovat. Dále by si měl také stanovit, jaká pravidla ve hře budou nastavena, jaký bude její cíl, kolik bude mít hráčů apod. Pro začátek je vhodné se také inspirovat již existujícími hrami, aby si žák utvořil představu o konečném produktu, který zamýšlí naprogramovat. Vhodné je také provést například tzv. testování na papíře. Jeremy Gibson ve své knize *Introduction to Game Design, Prototyping, and Development* uvádí, že i přes dostupnost moderních technologií se velké množství herních vývojářů stále spoléhá v úvodních fázích tvorby hry na papír a tužku. Podle Gibsona má testování na papíře mnoho výhod, mezi které patří například: rychlá příprava a možnost jednoduše provádět úpravy herního konceptu. Na tvorbě papírového prototypu se může podílet i technicky či výtvarně méně zdatný člen týmu a vzhledem k vědomí, že se jedná pouze o papírový prototyp, se může jedinec plně soustředit na vývoj herního konceptu, aniž by byl zatížen detaily, jako je např. grafická stránka věci [12, s. 126].

5.1.1 Smysluplná hra

Před započetím programování prochází žáci velmi důležitým krokem, ve kterém si volí, jakou hru chtějí naprogramovat. S použitím navržených pracovních listů, je tento krok již za žáky proveden, jelikož dostávají zadanou konkrétní hru či herní koncept. Bez

využití podpůrných materiálů hrozí, že žák nesprávně uchopí zvolený herní princip, či že výsledek jeho snažení nebude tzv. smysluplnou hrou. Koncept smysluplné hry, a především naučení se takovou hru vytvořit, Salen a Zimmerman ve své knize *Rules of Play* pokládají za jeden z nejdůležitějších cílů herního designu [13 s. 33]. Smysluplná hra vzniká díky vztahu mezi akcí, kterou hráč provede, a reakcí, které se mu dostane ze strany hry. Tyto akce a reakce jsou jasně rozpoznatelné a zasazené do širšího kontextu hry [13 s. 34]. Z této definice vyplývá, že aby hra byla smysluplná, je třeba, aby vhodně a viditelně reagovala na svého hráče a aby byl hráč prostřednictvím svých akcí schopen měnit průběh hry.

5.1.2 Pravidla a cíl hry

Pravidla hry jsou velmi důležitým faktorem při tvorbě a následném hraní jakékoliv hry. Scott Rogers, autor knihy *Level Up!*, definuje hru pomocí tří klíčových vlastností [14, s. 3]:

- má alespoň jednoho hráče,
- má jasně stanovená pravidla,
- má podmínku zajišťující vítězství.

Hry bez pravidel často postrádají smysl a hráče často velmi rychle omrzí. Stejně tak důležitý je cíl hry, ke kterému se má hráč dostat. V průběhu procesu tvorby hry je možné cíl měnit či rozvíjet, ale je vhodné, aby již na začátku programování měl žák nějaký cíl rozmyšlený, aby k němu mohlo jeho programování směřovat. Z pohledu hráče je cíl hry neméně důležitý. Žák, který hru programuje musí tedy myslet mimo jiné i na to, aby pro hráče byl cíl hry dostatečně zřejmý a snadno uchopitelný [14, s. 3]. Shoda Rogersovy definice hry [14 s. 3] a definice smysluplné hry od Salenové a Zimmermana [13, s. 34] naznačuje, že se skutečně jedná o důležité prvky, na které by se žáci v úvodní části tvorby své vlastní hry měli zaměřit.

5.2 Vlastní programování

Během programování projektů z pracovních listů jsem identifikovala několik stádií vlastního programování, ve kterých jsem se soustředila na určitý úkon. Jedná se o tato stádia:

- úvodní příprava základních prvků,
- návrh možného řešení,
- implementace zvoleného řešení,
- testování a oprava programu a
- ladění programu.

Tato stádia se různě prolínají a opakují. Jelikož je obtížné celou hru naprogramovat na první pokus bez průběžného testování, opakují se některá stádia v průběhu tvorby hry mnohokrát. Žáci se mohou libovolně přesouvat mezi jednotlivými fázemi podle potřeby. Dále také záleží na individuálním stylu programování jednotlivých žáků. Někteří žáci preferují soustavnou práci na jedné postavě před tím, než se přesunou na druhou. Jiní žáci naopak potřebují neustále přeskakovat mezi jednotlivými postavami, aby si udrželi v hlavě představu toho, jaký kód v sobě jednotlivé postavy mají, a mohli je tak všechny současně programovat a vylepšovat. Pro co nejlepší finální výsledek je dobré žáky motivovat k tomu, aby prošli všechny zmíněné fáze.

5.2.1 Úvodní příprava základních prvků

Před zahájením programování jednotlivých postav je vhodné si urovnat myšlenky a připravit si základní strukturu postav a pozadí, která budeme v projektu potřebovat. Ve své podstatě se jedná o fázi, kdy si žák vytvoří jednotlivé postavy a pozadí, které bude dále programovat. U jednodušších projektů se toto stádium vyskytuje pouze na počátku programovací fáze tvorby hry. U složitějších projektů se však tato fáze může mnohokrát

objevovat v redukované formě, kdy žák přidává pouze jednu či dvě nové postavy či pozadí, na která předtím zapomněl, či o kterých nevěděl, že je bude potřebovat. Součástí této fáze je také pojmenovávání jednotlivých postav a pozadí, kresba kostýmů či jejich volba z nabídky, a také import či volba zvuků, které mají jednotlivé postavy posléze vydávat. Pro žáky by mohla být komplexní příprava všech postav příliš komplikovaná, a tak by mohli zvolit postup, kdy postavy přidávají a programují postupně. Většinou se však nevyhnou nutnosti interakce více postav najednou a budou nuceni tyto postavy také najednou přidávat či programovat.

5.2.2 Návrh možného řešení

Po přípravě základního rozvržení prvků přichází na řadu přípravná programovací fáze. V této fázi jsem rozmýšlela, jak dané postavy naprogramuji, jaké bloky k tomu budu potřebovat a podobně. Zejména u mladších žáků tato fáze může chybět či se překrývat s fází následující, jelikož nemusejí mít dostatečně rozvinuté abstraktní myšlení. Tento myšlenkový proces návrhu řešení může být nahrazen zkoušením jednotlivých možností. Přímé zkoušení jednotlivých řešení daných podproblémů může být velmi zdlouhavé a alespoň rámcové promyšlení postupu před započetím programování by žákům mohlo velmi urychlit následnou tvorbu vlastního kódu.

V této fázi se opět můžeme odkázat na Gibsonovo testování na papíře [12, s. 126]. S využitím papírového prototypu hry, si mohou žáci snáze představit, jaké úkony bude daná postava muset vykonat a jaké bloky k tomu budou potřebovat použít.

Fáze návrhu možného řešení může předcházet úvodní přípravě základních prvků. Volba pořadí těchto dvou fází je individuální a závislá na preferenci a stylu práce každého žáka. Zároveň se tyto fáze mohou prolínat, jelikož v závislosti na návrhu možného řešení může žák přidávat další prvky do projektu, které pak opět ovlivní finální zvolené řešení.

5.2.3 Implementace zvoleného řešení

Při implementaci zvoleného řešení žáci aktivně tvoří jednotlivé scénáře. Jedná se o čistě programovací fázi, kdy je tvořen nový kód. Tato fáze se může ve velké míře prolínat s fázemi testování a ladění. Jelikož Scratch umožňuje neustálou kontrolu výsledků naší práce, mohou žáci postupovat v programování velmi malými krůčky. Tento postup jim však umožňuje eliminaci opakovaného opravování velkých částí kódu.

Právě v této fázi žáci často objeví chybějící postavu, kostým či pozadí a musejí se vracet zpět do první fáze, aby chybějící prvek doplnili. Poté se opět přesunou do fáze rozmýšlení a návrhu řešení či budou rovnou implementovat další část již navrženého řešení.

5.2.4 Testování a oprava programu

Testovací část procesu tvorby hry je velmi důležitá a je třeba ji nezanedbávat. Žáci mohou svůj pokrok kontrolovat opakovaným spouštěním programu a nacházením chyb. Do této fáze také patří oprava nalezených chyb či úprava řešení v závislosti na odhalené chybě.

Při tvorbě her se žákům může stát, že budou stále dokola hrát již funkční úsek své hry a zapomenou, že je třeba hru ještě dokončit. Toto může být jeden z důvodů, proč zabírají testovací a opravná fáze velkou část celkového času stráveného tvorbou projektu.

Zároveň v této fázi musejí žáci opětovně procházet vlastní kód s cílem odhalit chybu. Pro co nejsnazší odhalení chyb je vhodné, aby žáci všechny prvky smysluplně pojmenovávali a umisťovali. Specificky se jedná o různé zprávy odesílané napříč projektem, názvy a obsah proměnných, ale také názvy postavy, jejich kostýmů a jednotlivých pozadí.

Po opuštění této fáze mohou žáci přejít do libovolné další fáze v závislosti na stavu projektu. Pokud žák objeví, že mu chybí postava, přechází do první fáze. Pokud žák objeví a vyřeší chybu, může se přesouvat buď do fáze tvorby nového kódu či do fáze ladění.

5.2.5 Ladění programu

Jako ladění programu je označena aktivita, při které žák mění již funkční části kódu tak, aby byly efektivnější, jednodušší, úspornější či přehlednější. Jedná se často využití vlastních bloků a jejich volání, případně tako o náhradu více stejných postav jednou klonující se postavou apod.

Narozdíl od testování a opravy programu zde žák nehledá chybu, ale možnost zlepšení. Tato fáze není pro fungování programu nezbytná, ale je vhodné, abychom žáky vedli k tomu, aby svou práci kriticky zhodnotili a snažili se ji co nejvíce zlepšit. Zároveň se již také nejedná o fázi, kde by žáci přidávali nové funkcionality do projektu. Můžou se však do fáze implementace nového řešení plynule přesunout, pokud je v souvislosti s vylepšováním funkce stávajícího kódu napadne i nové celkové vylepšení projektu.

6 Základní programovací mechanismy

Při tvorbě ukázkových projektů se mi podařilo identifikovat některé scénáře, které se v projektech často opakují. Je možné tedy tyto scénáře považovat za základní mechanismy, které žák dříve či později bude potřebovat využít. V následujících podkapitolách jsou jednotlivé druhy scénářů popsány a jejich popisy jsou doplněny o příklady jejich použití či možnosti modifikace.

6.1 Pohyb ovládaný klávesami

Jedním z prvních interaktivních scénářů, které žák během své tvorby v prostředí Scratch používá, je pohyb postavy ovládaný klávesami na klávesnici. Tato forma interakce mezi hráčem a hrou je využívána velmi často. Důležitým krokem je volba kláves, které žák pro ovládání postavy využije. Nejčastějšími volbami pro pohyb jsou šipky, či klávesy W, A, S, D. Obě tyto varianty jsou využívány v počítačových hrách právě pro ovládání pohybu postavy a jsou tak většině žáků důvěrně známé. Scratch umožňuje individuálně zaznamenávat vstup ze všech kláves anglické abecedy, mezerníku a také z čísel na numerické klávesnici. Vstup z ostatních kláves je možné zaznamenávat pouze po zvolení varianty „libovolná“. Tato varianta umožňuje zaznamenat stisk libovolné klávesy, ale neidentifikuje, která klávesa byla stisknuta.

Pro každou klávesu, kterou chce žák zaznamenávat, je nejvhodnější vytvořit samostatný scénář. To umožňuje nastavit postavu tak, že se bude pohybovat pouze po jedné ose či do všech čtyř směrů. Vytvořením jednotlivých scénářů pro individuální klávesy,

které mají sloužit pro ovládání pohybu, také minimalizujeme nutnost využití nekonečných cyklů, které by jinak musely stisk zvolených kláves detekovat. Na obrázku 24 jsou zobrazeny základní scénáře pro pohyb do čtyř směrů.

Obrázek 24 Pohyb čtyřmi směry ovládaný šipkami

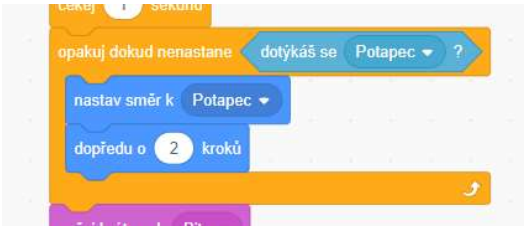
Pohyb ovládaný šipkami je možné také doplnit o blok „dotýkáš se“. Tento logický blok je možné využít pro detekci kolize postav, ale také pro kontrolu barev, kterých se aktivní postava dotýká. Doplnění o tento blok je užitečné zejména v případě, že žák programuje bludiště, kdy postava nesmí překročit hranici cesty, či různé honičky, kdy je třeba kontrolovat, zda se postavy dotýkají. Ve třech z pěti ukázkových projektů musí žák využít určitou formu ovládání pohybu postavy pomocí kláves. Na obrázcích 25 a 26 jsou zobrazeny scénáře pro barevnou detekci a detekci dotyku postav.

Obrázek 25 Detekce dotyku barvy

Obrázek 26 Detekce dotyku postavy

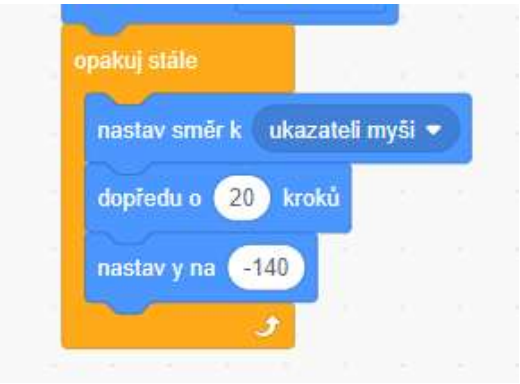
6.2 Semi-automatický pohyb

Semi-automatický pohyb postavy může do jisté míry nahradit ovládání šipkami. Jedná se o užitečný scénář, který se v projektech také často vyskytuje. Pro tento scénář žák využívá možnost nastavení orientace postavy určitým směrem či k určitému cíli. Pro hry, které jsou založeny na honičce mezi postavami, je možné využít orientaci postavy na postavu. Toto využití je zobrazeno na obrázku 27.



Obrázek 27 Scénář pro orientaci na postavu

Další variantou semi-automatického pohybu je následování kurzoru myši (viz obrázky 28 a 29). Stejně jako u ovládání pohybu klávesami na klávesnici je možné využít pohyb pouze po jedné ose či ve všech směrech. Na rozdíl od pohybu ovládaného šipkami se postava následující kurzor myši či jinou postavu může pohybovat skutečně všemi směry, nikoliv pouze horizontálně a vertikálně jako je tomu u pohybu ovládaného klávesami.



Obrázek 28 Scénář pro orientaci na kurzor myši

Obrázek 29 Scénář pro orientaci na kurzor myši s variabilní rychlostí pohybu

6.3 Skok na náhodnou pozici

Skok na náhodnou pozici je ve své podstatě relativně jednoduchý scénář. Pro celý tento úkon obsahuje Scratch v kategorii pohybu jeden blok. Alternativně může žák využít skoku na náhodnou pozici na jedné ose. Tato možnost je zobrazena na obrázku 30. Obě tyto variace jsou využity v navržených pracovních listech. Skok na kompletně náhodnou pozici je využit například ve hře *Potop se!* pro dvě postavy. Stejný koncept používáme pro generaci mincí, které hráč sbírá, a pro generaci bublin kyslíku, které může žák přidat do hry jako jedno z navrhovaných rozšíření. Skok na náhodnou pozici na ose x je využit ve hře *Bezedný košík?!*, kdy potřebujeme, aby se jablka generovala na náhodné pozici vždy u horního okraje obrazovky.

Obrázek 30 Skok na náhodnou pozici na jedné ose

6.4 Skok na náhodnou pozici z pevného seznamu

Využití seznamu pozic, na které může postava skočit, je již mírně náročnější. Praktické využití tohoto scénáře můžeme vidět například v ukázkovém projektu pro poslední pracovní list. Hra *Kouzelné bludiště*, která je využita pro poslední pracovní list obsahuje fixní seznam 15 pozic rozmístěných v bludišti. Tento seznam je využit pro více postav jako je generace náhodného efektu pro postavu, doplňující život, který se objevuje při snížené hodnotě životů základní postavy, a také pro teleportaci základní postavy na náhodné místo.

Pro co nejefektivnější a nejjednodušší využití seznamu pozic, je třeba vytvořit seznam pro každou souřadnici zvlášť. Poté můžeme využít například proměnnou, pomocí které zvolíme číslo náhodné pozice, na kterou se poté daná postava přemístí. Jedno z možných využití můžeme vidět na obrázku 31. Tento koncept najde své využití především v projektech, kde žák z nějakého důvodu potřebuje mít kontrolu nad tím, kde se daná postava smí objevit. Může se jednat právě o hry založené na principu bludiště, kde je třeba, aby se postava objevila na vytyčené cestě, a ne mimo ni.

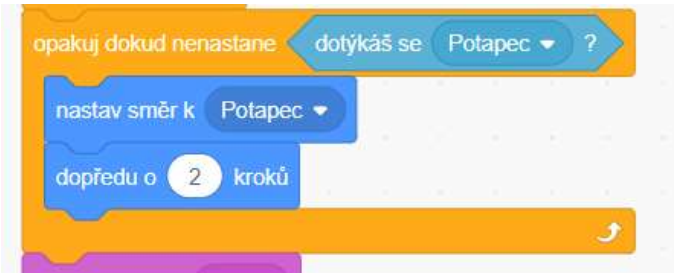
Obrázek 31 Skok na náhodnou pozici ze seznamu

Seznam pozic, na kterých se může postava objevit, je také využit v projektu Pexeso. Zde je však využití tohoto seznamu odlišné, jelikož přiřazujeme každé postavě jednu pozici tak, aby byla každá postava na jiném místě ve stejnou chvíli. Pro úspěšné využití seznamu pozic pro zamíchání kartiček pexesa je třeba využít ještě třetí seznam, který bude náhodně naplněn čísly podle počtu kartiček, například od 1 do 12 (když je počet kartiček 12). Další možnost využití scénáře skoku na náhodnou pozici ze seznamu můžeme vidět na obrázku 32.

Obrázek 32 Skok na náhodnou pozici s využitím tří seznamů

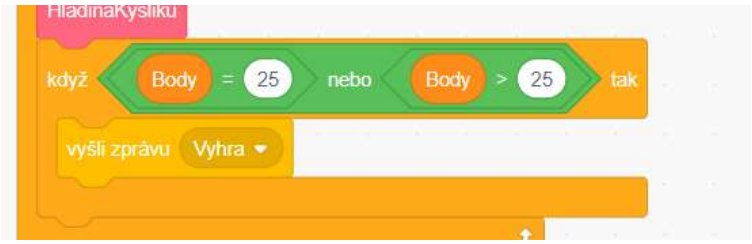
6.5 Ověřování splnění podmínek

Scénáře pro ověření splnění podmínek mohou mít mnoho podob. Jedná se o velmi obecný koncept, který je možné využít pro velké množství situací. Zpravidla pro ověřování splnění podmínky využíváme kombinaci dvou bloků, z nichž jeden je logický blok. První z možností je využití cyklu s podmínkou, do kterého je poté logický blok vložen. Obrázek 33 ukazuje využití cyklu s podmínkou, kde se daná činnost provádí tak dlouho, dokud není splněna podmínka uvedená v logickém bloku.



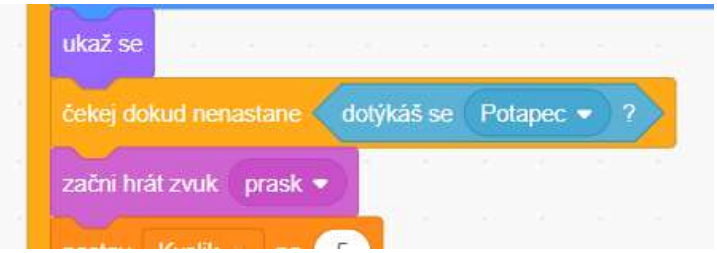
Obrázek 33 Využití cyklu s podmínkou

Obdobným způsobem můžeme využít podmínkový blok typu C („když, tak“) či typu blok E („když, tak nebo jinak“), které v sobě obsahují prostor pro vložení libovolného logického bloku. Na obrázku 34 je zobrazené využití logického bloku v kombinaci s podmínkovým C blokem. C blok využíváme v okamžiku, kdy potřebujeme nastavit pouze reakci na splnění podmínky. V případě, kdy potřebujeme naprogramovat více možností reakce v závislosti na tom, zda byla podmínka splněna či nikoliv, můžeme využít podmínkový blok typu E.



Obrázek 34 Využití logického bloku spolu s podmínkovým blokem C

Poslední variantou ověření splnění podmínky je využití čekajícího bloku s podmínkou, kdy jednoduše celý scénář stojí, dokud není zvolená podmínka splněna. Toto využití je zobrazena na obrázku 35. Použitý logický blok může ve všech případech patřit do více kategorií. Využít můžeme logické bloky z kategorie vnímání či z kategorie operátorů, které lze, jak již bylo řečeno výše, v případě potřeby vložit do sebe a zřetězit více podmínek do jedné.



Obrázek 35 Využití čekajícího bloku s podmínkou

7 Osadníci z Katanu

Mým prvním plánem bylo otestovat, zda je možné v prostředí Scratch vytvořit složitější hru. Pro tento úkol jsem si zvolila hru Osadníci z Katanu. Pří výběru hry pro tuto práci jsem se částečně řídila také tím, které hry jsou již ve hratelné podobě na platformě Scratch dostupné, a snažila jsem se zvolit takovou, která dostupná není. Po průzkumu již existujících variant této hry ve Scratch jsem došla k závěru, že nejspíš žádný z již existujících projektů nefunguje. Dále jsem také zjistila, že pod českým názvem hry neexistuje projekt žádný, a tak jsem procházela projekty tvořené v anglickém jazyce. Několik projektů již mělo zpracovanou grafickou stránku, kde většinou autoři využívali grafické provedení shodné se stolní verzí hry či její zjednodušenou obdobu. Při procházení projektů jsem také objevila, že velké množství autorů vychází ze tří nejpropracovanějších projektů a vytváří pouze remixy těchto projektů, ve kterých však příliš mnoho prvků nepřidávají.

7.1 Teoretická příprava

Před zahájením programování jsem prostudovala pravidla hry a pokusila se rozvrhnout, jaké postavy budou v projektu potřeba a jaká bude jejich funkce. Na papír jsem si rozkreslila poznámky pro skupiny postav, které v projektu budou potřeba, jejich základní funkce apod. V této fázi jsem měla největší problém s vytvořením závislostí jednotlivých políček se surovinami a jejich distribucí mezi hráče. Tento graf jsem několikrát překreslovala a k papírovému návrhu jsem se vracela i při vlastním programování. Nejproblematičtější bylo vytvoření vhodného systému, který by kontroloval při každém hodu kostkami, zda má nějaký hráč vesnici či město postavenou u pole s daným číslem, a zároveň by hráčům získané suroviny přiděloval.

7.1.1 Pravidla hry Osadníci z Katanu

Hra Osadníci z Katanu je strategická společenská hra založená na budování vlastní sítě vesnic a měst. Cílem hry je jako první nasbírat 10 nebo více bodů. Primárním zdrojem bodů je stavba cest, vesnic a měst na herní ploše, která je vyobrazena na obrázku 36. Různobarevná pole na herním plánu představují zdroje surovin (žlutá – obilí, zelená – ovce, hnědá – dřevo, oranžová – cihly, šedá – kámen). Aby mohl hráč budovat, musí získat určený počet surovin pro každou stavbu. Na začátku hry každý hráč umístí dvě vesnice a k nim přiléhající cesty na libovolné neobsazené místo na herním plánu (žluté křižovatky kolem jednotlivých surovinových polí).

V každém kole hráč, který je právě na tahu, hází dvěma kostkami a součet bodů na kostce označuje pole, ze kterého může hráč s přilehlou vesnicí či městem získat suroviny. Pokud se daného pole žádná vesnice či město nedotýká, žádný hráč nezískává žádné suroviny. Pokud padne číslo 7, přichází do hry zloděj (černobílá postava v masce, která je v základním rozložení umístěna na středovém poli hrací plochy, která představuje poušť). Hráč, který je na tahu a hodí číslo 7, volí libovolné jiné pole, na které zloděje umístí. Zde zloděj zůstává, dokud někdo opět nehodí číslo 7, a po tuto dobu z daného pole hráči nemohou získávat suroviny, ani když jim padne číslo daného pole.

Během hry mohou hráči stavět cesty, vesnice a města, bourat své budovy a vyměňovat suroviny s bankou (v kurzu 1:4) nebo smlouvat s ostatními hráči. V deskové verzi jsou suroviny získávány ve formě herních karet. Hráči mohou také získávat jiné karty, díky kterým dostanou body navíc či různé výhody. V okamžiku, kdy hráč, který je na tahu, získá libovolným způsobem 10 nebo více bodů, hra končí a daný hráč je vítězem.

7.2 Úvodní příprava základních prvků

Po několika dnech přerušované práce na návrhu hry na papíře jsem se přesunula do prostředí Scratch a začala připravovat projekt jako takový. Nejprve jsem začala pracovat na grafické stránce projektu a připravila postavy pro jednotlivá pole (pastviny, pole, lesy, skály a jíly), dále také postavu zloděje, tlačítka pro ovládání apod. Také jsem připravila instrukce na pozadí i ve formě postav. Dlouhou dobu jsem strávila nad kresbou jednotlivých polí, ale nakonec jsem se rozhodla pro obyčejnou jednolitou barvu představující danou surovinu. Finální vzhled polí i dalších prvků implementovaných do projektu je vyobrazen na obrázku 36. Všechny postavy mají originální kostým, což značně prodloužilo délku přípravné fáze. Pro většinu prvků však v nabídce Scratch nejsou vhodné postavy k dispozici.

Obrázek 36 Osadníci z Katanu – Výsledný vzhled hrací plochy

7.3 Návrh možného řešení

Při plánovaní vlastního programování jsem ve velké míře čerpala z teoretické přípravy, v rámci které jsem si již částečně řešení připravila. Nejobtížnější na návrh a následnou implementaci se ukázalo být vytvoření spolehlivého systému provázání náhodně

přidělených čísel k jednotlivým políčkům a seznamů surovin, které z daných polí každý hráč získával. Další relativně komplikovaná funkce byla náhodná, ale i základní generace hracího pole. Bylo nutné rozhodnout, jak se budou polím přiřazovat pozice a také jakým způsobem se k již umístěným polím přiřadí čísla.

U jednodušších prvků, jako jsou tlačítka, volba počtu hráčů apod., nebyla tato navrhovací fáze příliš nutná. Jelikož se jedná o prvky, které se ve hrách objevují relativně často a jejich konstrukce jsou velmi jednoduché, mohla jsem z velké části přejít rovnou do implementační fáze.

7.4 Implementace zvoleného řešení

Samotné programování jsem zahájila scénářem pro generaci herního plánu. V návodu ke stolní verzi hry Osadníci z Katanu je navrženo rozložení hracího plánu pro začátečníky. Toto rozložení jsem se rozhodla do hry zahrnout spolu s možností vygenerovat mapu náhodně. Po rozmístění polí tak, aby mohly být libovolně promíchané, jsem si poznačila souřadnice do poznámkového bloku a poté také do seznamu s názvem *Souřadnice*. Nakonec jsem se rozhodla použít pouze jeden seznam, ve kterém byly uloženy souřadnice x i y. Souřadnice jednotlivých polí byly v seznamu uloženy ve formátu xxxx&yyyy. Maximální počet číslic v jedné souřadnici je tři. První pozice každého čísla je rezervovaná pro případné využití znaménka mínus. Z důvodu zvolení pouze jednoho seznamu místo dvou jsem musela poté využívat spojování jednotlivých číslic, což bylo velmi zdlouhavé (viz obrázek 37).

Obrázek 37 Osadníci z Katanu – Systém přiřazování pozice pro jednotlivá pole

Stejný problém jsem pak musela řešit pro číslice, které se náhodně přiřazují k jednotlivým polím. Komplikovanější však bylo propojit čísla polí, podle kterých se hráčům přiřazují suroviny, s typem pole na hrací desce. Několikrát jsem se vracela k papírovému nákresu a hledala, kde jsem udělala chybu. Domnívám se, že by finální řešení, které je vidět jako poslední řádek na obrázku výše, mělo být funkční. Nepodařilo se mi však dovést hru do takové fáze, abych mohla tento postup ověřit při opravdové hře.

Po vyřešení potíží s přiřazováním čísel k polím se zdála implementace postavy zloděje do hry relativně jednoduchá. Postava má nastavenou kontrolu po každém vrhu kostkou. V případě že padne číslo 7, zvolí hráč, který je zrovna na tahu pole, na které je zloděj nově umístěn. Postava zloděje následuje kurzor myši a po kliknutí na zvolené pole je zloději nastavena stejná pozice ze seznamu souřadnic, jako má pole, na které hráč klikl. Celý scénář pro relokaci zloděje je zobrazen na obrázku 38.

Do hry jsem také implementovala jednodušší věci jako volbu počtu hráčů (tři nebo čtyři) a grafické znázornění hodu kostkou, přípravu na změnu módu (budování, bourání či změnu vesnice na město). Dále jsem hře také přidala kompletní ovládání pomocí tlačítek a úvodní menu s volbou rozložení herního pole.

Obrázek 38 Osadníci z Katanu – scénář pro přemístění zloděje

Během vytváření nového kódu jsem průběžně již naprogramované funkce testovala a opravovala. Vzhledem k tomu, že jsem téměř u každého nového kusu kódu okamžitě ověřovala jeho funkčnost, není možné přesně oddělit jednotlivé fáze či identifikovat, která z nich byla časově náročnější. U některých složitějších konstrukcí se zdála bát testovací a opravná část náročnější, jelikož bylo třeba pouze pozměnit malé části již existujícího kódu tak, aby se nenarušila žádná jiná funkce.

7.5 Závěrečné obtíže

Během procesu programování funkce přidávání vesnic, cest a měst se však hra přestala ukládat. Došla jsem k závěru, že projekt byl příliš komplexní a platforma již nezvládala takový projekt zpracovat. Jedním z důvodů pro tento problém může být velké množství postav v projektu (154) a komplexnost některých scénářů. Bohužel se mi nepodařilo identifikovat přesný zdroj těchto problémů a následně ho odstranit. Rozdělaný projekt je k dispozici na adrese https://scratch.mit.edu/projects/386280851/.

Bez možnosti ukládání postupu jsem na projektu nemohla dále pokračovat a musela jsem směr své diplomové práce upravit. Místo jednoho komplikovanějšího projektu jsem proto vytvořila několik jednodušších her, které mají základ v již existujících počítačových či stolních hrách. Celkem jsem se pro účely této diplomové práce rozhodla připravit pět ukázkových projektů doplněných o pracovní listy.

7.6 Retrospektiva

Po vytvoření zmíněných pracovních listů, jsem opět procházela svůj prvotní projekt a našla jsem mnoho věcí, které jsem bývala mohla udělat jinak a lépe. Fázi ladění projektu zařazuji až úplně na konec tvorby projektu a z tohoto důvodu jsem se při tvorbě hry Osadníci z Katanu do fáze ladění nedostala. Je možné, že by se mi ve fázi ladění bývalo podařilo některé problémy vyřešit, jelikož jsem posléze objevila určité chyby, které mohly být příčinou mého prvotního neúspěchu.

První zásadní chybou bylo rozhodnutí, že budu místo dvou seznamů pro souřadnice používat pouze jeden. Toto rozhodnutí mi značně zkomplikovalo nastavování souřadnic u všech postav. Nejspíš se však nejednalo o chybu, která by zapříčinila selhání projektu jako takového.

Další a pravděpodobně nejzávažnější chybou bylo nevyužití možnosti klonování postav. V projektu bylo několik míst, kde by bývalo bylo vhodné klonování využít, a já se místo toho rozhodla vytvářet jednotlivé postavy. Klonování polí jsem nevyužila z důvodu, že jsem v té době nepřišla na jednoduchý způsob, jakým jednotlivá naklonovaná pole odlišit a provázat s dalšími prvky, které na polích jsou závislé. U možnosti klonovat vesnice a cesty jsem již dále pokračovala cestou odmítající klonování a neviděla jsem benefity toho, proč ho použít. Toto rozhodnutí však způsobilo, že projekt obsahoval velké množství postav, což pravděpodobně zapříčinilo nemožnost projekt ukládat.

Při zpětném pohledu na projekt bych zcela jistě pro potřeby vesnic a cest využila klonování. Ve všech postavách pro vesnice a cesty je, kromě úvodních souřadnic, které je možné přenést při zahájení klonování, naprosto shodný kód, a tudíž by klonování mohlo mít pouze kladný přínos. Pro úpravu projektu by bylo třeba si poznamenat souřadnice a orientace jednotlivých vesnic a cest do jednotlivých seznamů. Dále bych také rozdělila všechny souřadnice, včetně souřadnic pro jednotlivá pole, do dvou seznamů, abych usnadnila jejich používání.

8 Pracovní listy

Celou sérii pracovních listů jsem se rozhodla pojmenovat *ScratchIT*. Na obrázku 39 je zobrazeno logo, které jsem pro vytvořené pracovní listy použila. Logo pracovních listů se skládá z loga programu Scratch doplněného o písmena IT. Název *ScratchIT* je možné interpretovat dvěma způsoby:

• Scratch it – *it* v anglickém jazyce znamená to. Volně bychom tedy mohli název přeložit jako „Scratchni to“ ve smyslu naprogramuj to ve Scratch. Termín „scratching“ v informatice označuje opakované používání částí kódu, které je možné snadno kombinovat, sdílet a upravovat [15, s. 1]. Tuto ideologii Scratch následuje například v možnosti „remixování“ sdílených projektů.

• Scratch IT – druhá interpretace se opírá o zkratku IT – information technology, což do českého jazyka překládáme jako informační technologie. Tento pojem bývá používán ve spojitosti s výukou informatiky na školách. (U nás využíváme převážně zkratku ICT/IKT – informační a komunikační technologie.)

Obrázek 39 Logo pracovních listů

Každý pracovní list obsahuje několik prvků, které se objevují v celé sérii. Jedná se o tyto prvky:

• označení obtížnosti daného pracovního listu,

• název projektu,

• úvodní motivační odstavec,

• shrnutí základních prvků, které se v projektu objevují,

• jednotlivé kroky vedoucí k úspěšnému řešení zadání,

• potenciálně obtížné části projektu,

• tipy pro snazší vypracování projektu,

• možná vylepšení a nástavby výsledné hry a

• odkaz na video se záznamem průběhu hry z ukázkového projektu.

Každý projekt má přiřazenou obtížnost, která je znázorněna počtem hvězd v rozmezí jedné až pěti. Rostoucí počet hvězd znázorňuje rostoucí obtížnost projektu. Jednotlivé prvky jsou záměrně barevně odlišeny pro lepší orientaci žáka v pracovním listu. Důležité jsou zejména texty ohraničené červeně (potenciálně problematické části projektu), žlutě (obecné tipy pro snazší vypracování projektu) a zeleně (možná vylepšení projektu). Série pracovních listů je doplněná o průvodní list (viz příloha 1), který uživatele seznamuje s pracovními listy, jejich koncepcí a také s významem jednotlivých rámečků v závislosti na jejich barevném ohraničení. Součástí každého pracovního listu je také video zaznamenávající celý průběh hry, které žáci mohou využít jako zdroj inspirace pro jejich vlastní projekt.

Do pracovních listů jsem také přidala chlapce jménem Ignác, který nikde není vyobrazen, ale slouží jako virtuální spojnice mezi projekty a také jako vrstevník, se kterým se může žák ztotožnit. Ignác je v průvodním listu popsán jako chlapec, kterého ze všeho nejvíc baví převádět zkušenosti z každodenního života na počítačové hry. Z toho důvodu je také vždy v úvodním motivačním textu uveden příklad, kdy se mohou žáci s daným tématem či situací v životě setkat.

Projekty, které byly využity pro jednotlivé pracovní listy jsou hry, jsem v průběhu posledních let vytvářela spolu s dětmi z různých věkových kategorií od žáků šestiletých

po žáky čtrnáctileté. Zkušenosti získané vedením zájmového kroužku v rámci Dětské univerzity a posléze také z vedení několika kroužků programování a výuky na základní škole v rámci mého ročního pobytu ve Velké Británii jsem využila při tvorbě zmíněných pracovních listů tak, aby co nejvíce odpovídaly průběhu tvorby jednotlivých projektů a aby reflektovaly jednotlivé úrovně zpracování podle schopností žáků. Pracovní listy by tak měly pokrýt široké spektrum úrovně schopností jednotlivých žáků a být tak oporou co největšího množství učitelů či žáků samotných.

Napříč jednotlivými projekty můžeme pozorovat určité chyby, které se objevují nezávisle na zadání. Jedna z nich se objevuje velmi často v okamžiku, kdy si žáci kreslí vlastní kostýmy pro jednotlivé postavy. Při práci v grafickém editoru, který je součástí prostředí Scratch, umístí žáci svou kresbu na libovolné místo vyhrazeného prostoru pro kresbu. V případě, že se jedná o postavu, která ve výsledném projektu bude statická, nezpůsobí nám umístění kresby žádný problém. V momentě, kdy však žák potřebuje s danou postavou pohybovat, je třeba, aby střed kresby ležel ve středu prostoru vytyčeného pro úpravu kostýmu postavy. V opačném případě se pak postava otáčí kolem středu a může dojít k jejímu vychýlení z trasy či k žákem nepředvídanému chování při skoku na pozici. Důvodem tohoto problému je to, že střed kresby se neutváří automaticky, ale je předem daný a vyznačený. Postava se pak nezávisle na obsahu otáčí kolem stanoveného středu celého prostoru pro kresbu kostýmu, nikoliv kolem středu vlastní kresby.

Další obecnou chybou, se kterou jsem setkala v žákovských pracích, je práce s proměnnými. Zde se jedná již o pokročilejší koncept, než je například ovládání pohybu. Z množství chyb vyplývá, že žáci více přemýšlí o funkci proměnných než o jejich nastavení. Jsou pak schopni je správně použít v podmínkových konstrukcích a podobných situacích, ale zapomínají na jejich úvodní nastavení. Ve hrách, které jsou založeny na

získání dostatečného počtu bodů, se pak hráč dostává do situace, že po opakovaném spuštění hry již má na počátku dostatečný počet bodů a hra končí. Pro tento typ chyby je velmi jednoduše implementovatelná náprava – do scénáře, který se spouští jako první, musí žák přidat bloky pro nastavení jednotlivých proměnných na jejich výchozí hodnotu. (Nejčastěji se jedná o hodnotu 0.)

8.1 Bezedný košík?!

První a nejsnazší pracovní list v sérii *ScratchIT* nese název *Bezedný košík?!* a celý tento pracovní list je k nahlédnutí v příloze 2. Tento pracovní list je založen na principu padajících objektů, které hráč chytá do zvolené nádoby. Za každý chycený objekt hráč získává body. Pokud hráč nasbírá určený počet bodů, vyhrává. Pokud však některý z padajících objektů nechytí, hra končí a hráč prohrává či případně s každým nechyceným objektem postupně ztrácí životy tak dlouho, až hra nakonec také skončí.

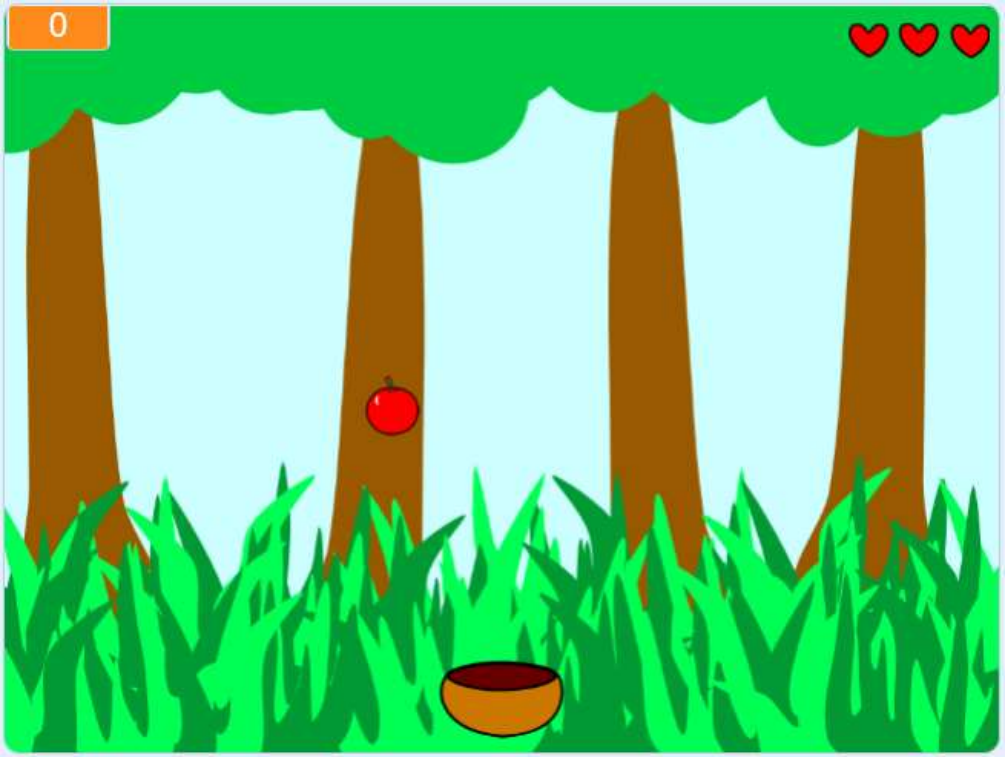
Tomuto pracovnímu listu jsem přiřadila obtížnost 1, jelikož není konceptuálně příliš složitý. Tento projekt také vyžaduje využití malého množství postav s relativně snadnými scénáři. Vytvořený pracovní list je možné využít i v běžné výuce například pro seznámení žáků se způsobem, jak pracovat s ostatními, náročnějšími pracovními listy. Na dalších zadáních posléze mohou žáci pracovat samostatně. Základní zadání, podle kterého je určována celková obtížnost pracovního listu, se skládá z postavy košíku a postavy padajícího jablka. (Žák si může zvolit postavy vlastní.) Košík je ovládán šipkami na klávesnici a pohybuje se pouze horizontálně. Jablka se objevují na náhodné pozici u horního okraje obrazovky a padají směrem dolů.

Již pro splnění základního zadání, musí žák využít několik konceptů jako je pohyb ovládaný šipkami, ověřování splněné podmínky a detekce dotyku okraje obrazovky. V rozšířeném zadání žák dále pracuje s hodnotou proměnné, podle níž dále mění různé atributy hry.

Nejobtížnější část kódu, která je v pracovním listu popsaná v červeném rámečku, sestává ze dvou cyklů s podmínkou vnořených do sebe. Toto řešení, ač je velmi efektivní, může být pro žáky komplikované a hůře uchopitelné. Z toho důvodu je tento scénář do pracovního listu vložen, aby si žák mohl snáze představit, jak vypadá vnoření cyklů, a následně ho správně použil.

8.1.1 Praktické otestování zadání

Projekt *Bezedný košík?!* Je z hlediska časového i obtížnostního nejméně náročný. Dokončení základního zadání zabralo žákům v průměru cca 40 minut souvislé práce. Z tohoto zjištění vyplývá, že by mohlo být vhodné zařadit tento pracovní list do klasické výuky, protože žáci by měli šanci jej vypracovat například v průběhu dvou na sebe navazujících vyučovacích hodin. Učitel by pak měl možnost se žáky projít celý pracovní list, poukázat na jeho klíčové vlastnosti a představit způsob, jakým by žáci mohli s pracovním listem pracovat. Náhled hry je vyobrazen na obrázku 40.



Obrázek 40 Bezedný košík?! – Náhled hry

Jednou z často opakujících se chyb u žáků je, že naprogramují jablko (či jinou postavu) tak, že padá pořád ze stejného místa. Je třeba žáky poté navést na myšlenku, že se jablko musí objevovat pokaždé jinde, aby byla hra zábavnější. Řešení této chyby je v pro-

Obrázek 41 Bezedný košík?! – náhodná generace u horního okraje obrazovky

středí Scratch jednoduché. Stačí v bloku skoč na pozici nahradit fixní hodnotu osy x blokem generujícím náhodné číslo ve stanoveném intervalu (viz obrázek 41). Občas se také u žáků objevuje řešení, kdy nastavují jednotlivé souřadnice odděleně. Oba tyto způsoby jsou naprosto v pořádku a případná korekce generace náhodné pozice na ose x je řešena stejným způsobem popsaným výše.

Další problém nastává při tvorbě cyklu pro padání a znovuobjevování postavy u horního okraje obrazovky, kdy je třeba rozlišit, kdy se dotkla košíku a kdy země. Tento problém je však v pracovním listu vyřešen snímkem potřebného kódu a popisem daného problému.

Žáci často zapomínají na to, že hra má mít také konec, a nevyužívají proměnnou pro kontrolu počtu sesbíraných objektů. Občas bylo třeba jim tuto drobnost připomenout. Pro vyřešení tohoto úkolu může žák využít například scénář pro ověření podmínek zobrazený na obrázku 42.

Obrázek 42 Bezedný košík?! – ověření sesbírání dostatečného počtu bodů

8.2 Potop se!

Druhým pracovním listen v sérii *ScratchIT* je klasická honička dvou postav. Jelikož v tomto projektu musí žák využít jak pohyb postavy ovládaný hráčem, tak také postavu pohybující se automaticky, byla tomuto listu přiřazena obtížnost 2. Základní princip hry spočívá ve snaze uniknout před pronásledující postavou a při tom nasbírat určený počet bodů.

V tomto projektu žák využívá hned několik scénářů, které jsem popisovala výše v kapitole *Základní programovací mechanismy*. Konkrétně se jedná o dvojité využití automatického pohybu spolu se skokem na náhodnou pozici a kontrolou splnění zadané podmínky. Stejně jako v předchozím listu se obtížnost projektu zvyšuje implementací rozšiřujících úkolů. Pro usnadnění splnění základního zadání je v pracovním listu ukázána možná implementace automatického pohybu a kontroly splnění zadané podmínky. Také je žák opět vybízen k použití vlastních bloků pro zjednodušení a zpřehlednění kódu. Celý pracovní list je přiložen k této práci jako příloha 3.

Tento projekt má vyšší obtížnost než projekt s padajícími jablky také proto, že žák musí myslet na více věcí najednou a také jich musí více ohlídat. V průběhu hry je třeba

kontrolovat, zda se žralok dotkl potápěče, zda se potápěč dotkl mince, či kolik bodů již hráč posbíral.

Při plnění rozšiřujících zadání musí žák již prokázat vyšší míru samostatného logického myšlení, jelikož je třeba při nastavování hladiny kyslíku počítat s automatickým snižováním jeho hladiny. Dále je třeba měřit dobu, kdy je mince zobrazena, a ve správný okamžik ji opět přemístit. V neposlední řadě je také třeba implementovat ukazatel kyslíku. V ukázkovém videu je také patrné, že potápěč s nízkou hladinou kyslíku plave pomaleji a žralok ho snáze dostihne. Tato vlastnost však v pracovním listu záměrně zmíněna není. Podobná vlastnost je již využita v rozšiřujícím zadání prvního pracovního listu, kdy se spolu s rostoucím množstvím získaných bodů zvyšuje také rychlost, jakou jablka padají.

8.2.1 Praktické otestování zadání

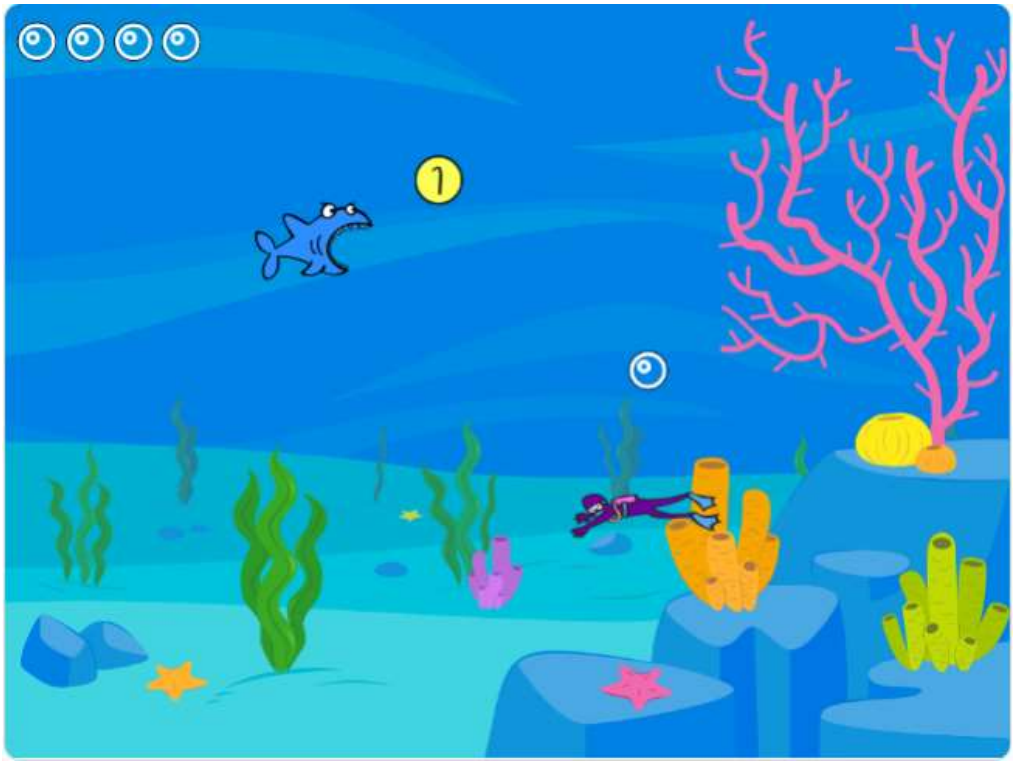
V tomto typu projektu můžeme nalézt několik chyb, které se objevují relativně často, ale zároveň nejsou obtížné na nápravu. Jednou z nejčastějších chyb je neukončení některých částí programu v okamžiku, kdy je postava chycena. V ukázkovém projektu se jedná o chvíli, kdy žralok dohoní potápěče. V tuto chvíli se mohou stát například tyto dvě věci v závislosti na chybě, kterou žák udělal:

• automaticky se pohybující postava se na místě začne divoce otáčet a převracet, jelikož se již dotýká svého cíle,

• postava, která originálně pronásledovala kurzor myši, bude pokračovat ve svém pohybu.

Ve videu mohou žáci vidět, že po chycení postavy potápěče se žralok přesouvá do středu obrazovky a hráč se dozvídá své výsledné skóre. Pokud je hráč úspěšný, je ukázkový projekt naprogramován tak, aby se objevila podobná obrazovka jako při prohře, ale hráč je pogratulováno k vítězství a je mu nabídnuta možnost hrát znovu. Stejnou možnost

hráč dostává, i když prohraje. Závěrečná obrazovka není zahrnuta v zadání pracovního listu, ovládání pomocí tlačítka (včetně tlačítka pro znovuspuštění hry) však již zahrnuto je. Na obrázku 43 je zobrazen snímek průběhu hry.



Obrázek 43 Potop se! – náhled hry

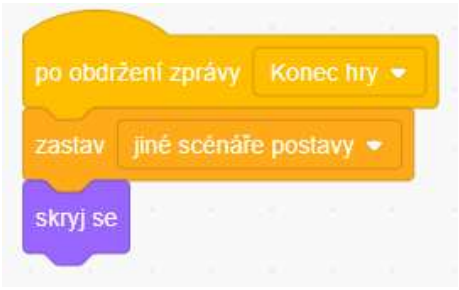
Situaci konce hry lze řešit lokálně či globálně v závislosti na struktuře celého kódu. Žák může do každého scénáře, který by neměl po splnění určité podmínky pokračovat vložit blok pro ukončení daného scénáře (lokální). Další možností řešení takovýchto chyb je využití bloku pro ukončení všech scénářů či všech ostatních scénářů. Tato volba záleží na tom, zda chce žák, aby se ještě něco stalo po ukončení hry či nikoliv. (Například by se mohla objevit závěrečná obrazovka s výsledným počtem získaných bodů apod.) Na ob-

rázku 44 je zobrazena část scénáře postavy žraloka, která zajišťuje, aby po chycení potápěče byla odeslána zpráva Konec hry, na kterou poté reagují všechny potřebné postavy podobně jako postava potápěče (viz obrázek 45).

Často se v tomto typu zadání negativně projevuje nedostatečné nastavení úvodní polohy jednotlivých postav. Stává se pak to, že při druhém spuštění hry zůstávají postavy na poslední poloze z předchozího kola, a tím je hra ihned ukončena, protože se postavy dotýkají. Je třeba tedy zdůraznit, aby žáci nezapomněli na úvodní herní nastavení.

Nastavení rychlosti pohybu jednotlivých postav nelze zcela jednoznačně označit za chybu, ale nevhodná volba rychlosti pohybu obou postav může značně omezit hratelnost výsledné hry. U postavy, která následuje kurzor myši je obecně vhodnější nastavit lehce vyšší rychlost než u postavy, která ji automaticky pronásleduje. Pokud jsou rychlosti stejné, či pokud má „honící“ postava vyšší rychlost, stává se velice obtížné až nemožné hru úspěšně dokončit, což může být pro hráče demotivující.

Obrázek 44 Potop se! – scénář pro odeslání zprávy Konec hry z postavy žraloka



Obrázek 45 Potop se! – scénář s reakcí na zprávu Konec hry z postavy potápěče

8.3 Pexeso

Tento pracovní list je označen obtížností 3, přestože se jedná konceptuálně o velmi jednoduchou hru, se kterou se žáci setkávají již v předškolním věku. Cílem hráče je postupně otáčet kartičky s obrázky a zapamatovat si, kde leží shodné páry. Tyto páry hráč pak postupně odstraňuje. Cílem hry je odstranit všechny shodné páry kartiček. Tento pracovní list je zaměřen především na práci s proměnnými a se seznamy a je přiložen k této práci jako příloha 4.

Na rozdíl od ostatních projektů z této série žák nepotřebuje programovat téměř žádný pohyb postav. Na počátku hry jsou kartičky umístěny na hrací pole a poté už se pouze mění jejich vzhled tak, aby byla vidět rubová či lícová strana. Nejkomplikovanějším prvkem tohoto projektu je míchání kartiček. Polohy jednotlivých kartiček jsou náhodně přiřazovány z předem připraveného seznamu. Dále musí žák ošetřit, aby hráč mohl otočit pouze dvě kartičky, než se opět otočí lícem dolů.

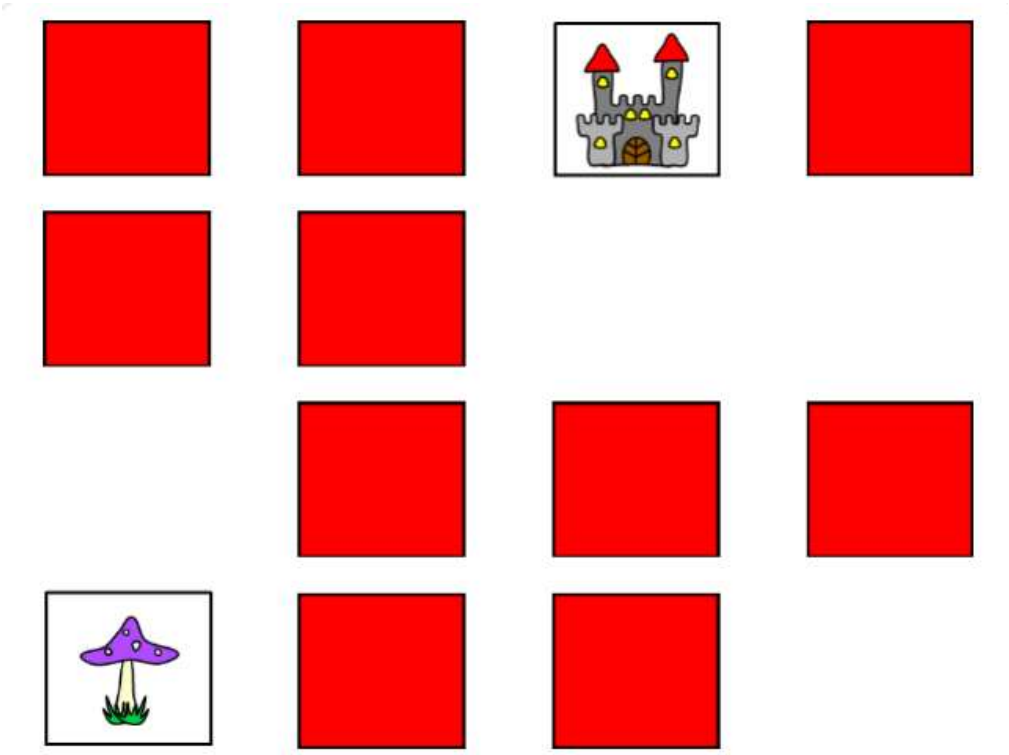
Tento projekt umožňuje žákům se výtvarně projevit, jelikož si mohou nakreslit vlastní kartičky. Hráči, kteří jsou více orientovaní na programování, a výtvarná stránka projektu je tolik nezajímá, mohou využít již připravené postavy v galerii.

Projekt je realizován ve dvou fázích. V první fázi žák naprogramuje systém otáčení kartiček se všemi opatřeními proti otočení více kartiček apod. Výsledkem první fáze tak je již hotová hra pexesa, kde jsou však kartičky stále na stejném místě a opakovaná hra se tak stává nudnou, jelikož si hráč rychle zapamatuje pozici jednotlivých párů kartiček. Ve druhé fázi projektu tak žák programuje samotný systém pro míchání kartiček, aby byla hra celkově zajímavější a atraktivnější. Z tohoto důvodu má pracovní list více stránek než předchozí pracovní listy.

Přiřazování náhodných poloh k jednotlivým kartičkám může být pro žáky z počátku komplikované. Pro snazší vypracování projektu jsou klíčové části kódu zobrazeny v pracovním listu.

8.3.1 Praktické otestování zadání

Projekt založený na hře pexeso již patří mezi náročnější práce. Jeho obtížnost však spočívá především v kontrole otočení správného páru kartiček a v jejich míchání na začátku každé hry. Prvním úkolem pro žáky je připravit jednotlivé páry kartiček a rozmístit je na hrací plochu (viz obrázek 46). Posléze přichází na řadu obtížnější úkoly jako navržení systému kontroly otočení kartiček a míchání kartiček na začátku hry.

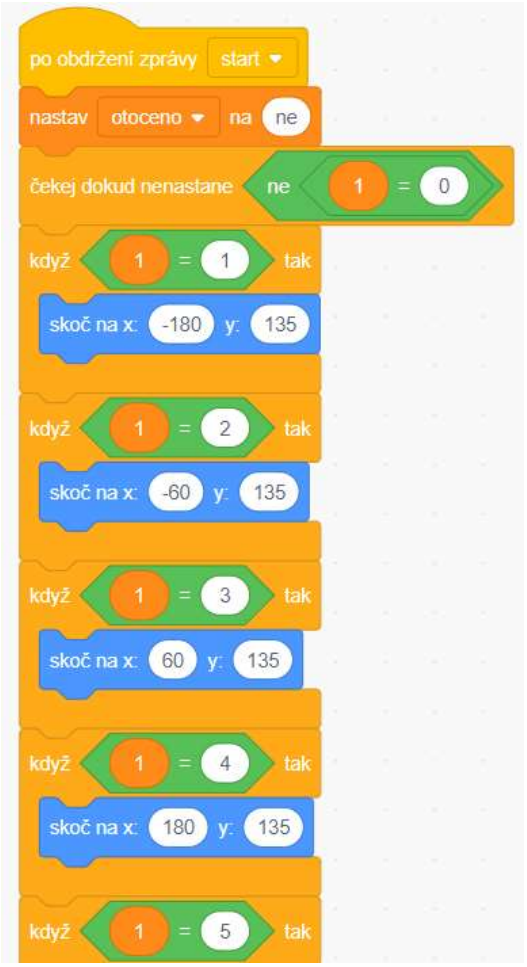


Obrázek 46 Pexeso – náhled hry

Velká část žáků snahu o vytvoření funkčního mechanismu pro míchání kartiček vzdala. Při tvorbě vlastního scénáře pro míchání kartiček jsem nejprve i já zvolila velmi složitý postup, který mi však v daném okamžiku připadal nejrozumnější. Na obrázku 47 je vyobrazena původní část kódu pro přiřazování pozice jednotlivým kartičkám, která byla umístěna v postavě určené k ovládání hry. Dále bylo v postavě každé kartičky třeba nastavit pozici podle přiřazeného čísla (viz obrázek 48). Tento způsob však nebyl ekonomický a ani příliš přehledný.

Při ladění projektu jsem však přišla na lepší řešení, které ve větší míře využívá seznamy a mnohem efektivněji a přehledněji zajistí stejné úkony jako předchozí řešení. Toto řešení je k vidění na obrázku 49. Přiřazování se nově odehrává pouze v jednotlivých kartičkách a v řídící postavě zůstal jen scénář pro generaci seznamu použitých čísel.

Obrázek 47 Pexeso – původní způsob přiřazování pozic; kód v řídící postavě



Obrázek 48 Pexeso – původní způsob přiřazování pozic; kód v každé kartičce

Obrázek 49 Pexeso – nový způsob přiřazování pozic

Podobný problém jsem řešila i u propojení jednotlivých párů kartiček. V původním řešení jsem využívala proměnné pro uložení, která kartička byla otočena první, která druhá a zda mají obě stejné číslo. Toto řešení, leč funkční, bylo taktéž velmi složité na provedení i pro následné popsání v pracovním listu. Později jsem systém změnila a pro kontrolu otočení správnosti jsem našla mnohem jednodušší řešení založené na čísle kostýmu jednotlivých kartiček. Scratch umožňuje všem postavám číst číslo/název kostýmu ostatních postav. Tato funkce samotná umožňuje snadnou kontrolu správnosti otočeného

páru. Každá kartička může mít dva kostýmy – lícem dolů (1) a lícem nahoru (2). Po otočení dvou kartiček hra kontroluje, zda jsou lícem nahoru otočeny kartičky z jednoho páru. Pokud má odpovídající kartička číslo kostýmu 2, znamená to, že hráč našel pár, přičítá se mu bod a kartičky se skrývají. Původní a vylepšené řešení pro kontrolu správnosti páru jsou k porovnání na obrázcích 50 a 51. Obě nová a jednodušší řešení jsem zapracovala do výsledného pracovního listu, který je součástí této práce.

Obrázek 50 Pexeso – původní kontrola otočení páru

Obrázek 51 Pexeso – nové řešení kontroly otočení páru

Kromě těchto dvou komplikovaných úkolů v projektech nebyly žádné výrazné nedostatky. Někteří žáci se rozhodli udělat různé tvary kartiček, což by v praxi nefungovalo, jelikož by se tvary po zamíchání překrývaly. Tito žáci však s projektem nedošli tak daleko, aby tento problém odhalili.

8.4 Brick Breakers

Předposlední hrou v pořadí je velmi známá hra založená na odrážení kuličky od platformy tak, aby zničila všechny cihly v horní části obrazovky. Tato hra opět staví na základech z předchozích pracovních listů a přidává další koncepty. V tomto pracovním listu se konkrétně jedná o klonování objektů. Původní inspirací pro tento projekt je hra založená pouze na míčku, který se volně pohybuje po herní ploše a ve spodní části herní obrazovky je platforma, která míčku musí zabránit s kontaktem se spodním okrajem obrazovky. Tento herní koncept byl však pro rozšiřující zadání příliš snadný, a tak k němu bylo přidáno právě ničení cihliček, což celkovou obtížnost projektu výrazně zvýšilo.

S ohledem na velké množství konceptů, které jsou v této hře použity má tento pracovní list již obtížnost 4 a je svou obtížností vhodný zejména pro žáky starší. Při tvorbě hry musí žák opět použít různé druhy pohybu a detekci splněných podmínek. Dále nastavuje klonování cihly tak, aby vykreslila požadovaný obrazec. Důležitou částí je také pohyb kuličky a její interakce s jednotlivými klony cihly.

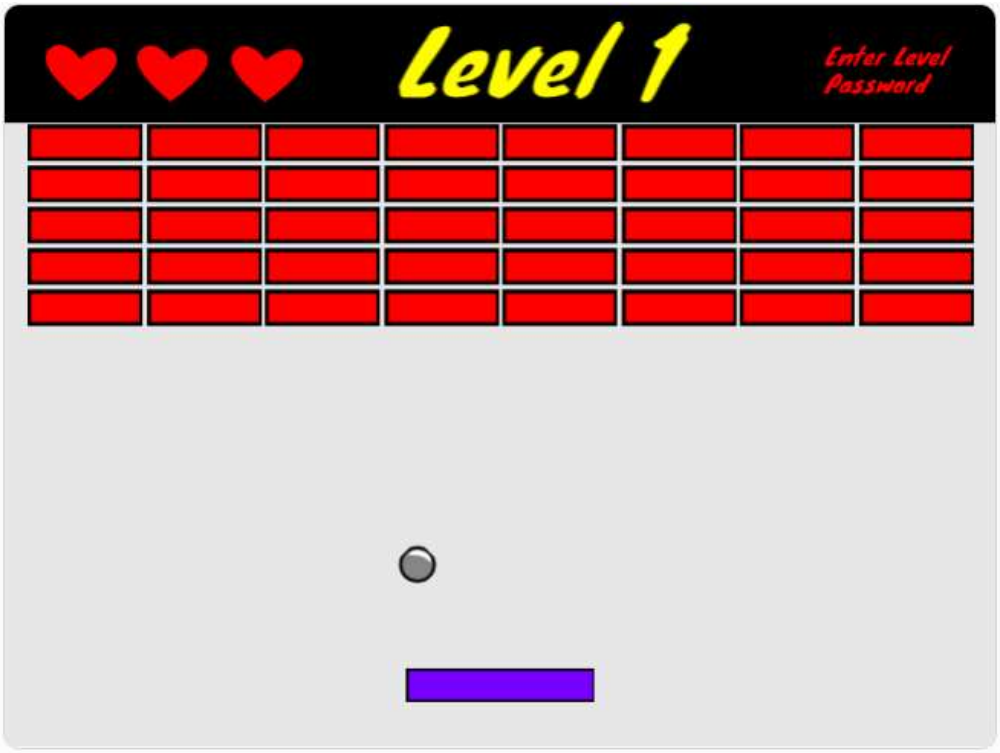
Právě klonování cihly by mohlo být pro žáky nejkomplikovanější, a tak jsou některé části kódu přímo ukázány v pracovním listu. Další problematickou částí je směr, ve kterém se kulička od jednotlivých cihel odráží. Na rozdíl od odrážení se od krajů obrazovky musí žák toto odrážení nastavit sám. Ani v ukázkovém videu není odraz kuličky zcela správný, jelikož by bylo třeba ošetřit kromě směru, ze kterého kulička letí, také hranu cihly, které se dotkne apod. Tato problematika je stručně osvětlena v pracovním listu, aby se žák zbytečně netrápil s problematickým odrážením kuličky a mohl se soustředit na ostatní části kódu.

V rozšiřujícím zadání dostává žák možnost se také výtvarně projevit a vytvořit zajímavé vzory pomocí cihel a také navrhnout různé vzhledy platformy a kuličky, ze kterých si hráč může vybrat. Pro umožnění odemykání jednotlivých vzhledů například po

splnění dané úrovně je třeba rozšířit kód o detekci splnění podmínek pro odemčení jednotlivých vzhledů, což projektu opět přidává na obtížnosti. Další prvek, který byl zařazen do rozšiřujících zadání je tvorba více úrovní a možnost vracet se do již splněných úrovní pomocí hesla. Celý pracovní list je přiložen k práci jako příloha 5.

8.4.1 Praktické otestování zadání

Pracovní list *Brick Breakers* je nejsložitějším zadáním ze sady řízených projektů. Jeho obtížnost spočívá především v generaci jednotlivých úrovní a v nastavení mechanismu odrážení kuličky. V porovnání s ukázkovým projektem je však možné celý projekt velmi zjednodušit. V základním zadání žáci programují pouze jednu úroveň hry a hráč má pouze jeden život na její dokončení. Při vypracovávání této varianty zadání bude stále zapotřebí, aby žáci dopředu uvažovali o umístění jednotlivých cihel a o jejich velikosti. Nejjednodušší rozložení cihliček v první úrovni je vyobrazeno na obrázku 52.



Obrázek 52 Brick Breakers – náhled hry

Základem pro úspěšné rozvržení cihel je nutné, aby si žáci předem připravili cihlu správné velikosti tak, aby se jimi zvolený počet cihel vešel do řádku na hrací plochu bez překrývání. Nejsnazší je vytvořit si zpočátku stejných cihliček více a ručně je na plochu rozložit. Poté žáci naprogramují klonovací scénář tak, aby se cihly objevovaly na požadovaných místech automaticky. Pro nejvyšší přehlednost programu je vhodné pro jednotlivé typy akcí využít různé vlastní bloky. V ukázkovém projektu jsou využity celkem tři různé typy akcí – pohyb s klonováním, pohyb bez klonování a skok na začátek dalšího řádku. Všechny tři vlastní bloky jsou na obrázku 53. Ukázkové využití je poté zdokumentováno na obrázku 54. Při volání každé akce využíváme parametry pro určení počtu opakování (klonování a pohyb bez klonování) a pozice (úvodní pozice při skoku na další řádek). Právě tvorba klonovacího scénáře může být pro žáky tím nejobtížnějším krokem tvorby celého projektu, a proto jsou všechny tři vlastní bloky pro akce generace úrovní součástí pracovního listu.

Obrázek 53 Brick Breakers – vlastní bloky pro jednotlivé akce generace úrovní

Další relativně komplikovaný úkon je správné nastavení odrážení míčku od jednotlivých cihliček. Jelikož směr odrazu míčku závisí také na tom, které strany cihly se dotkne a z jakého směru letí, což by v praxi znamenalo velké množství různých kombinací, není pro úspěšné vypracování projektu nezbytné, aby hra přesně odpovídala zákonům fyziky. Pro absolutní správnost odrazu by bylo třeba hrany každé cihly od sebe odlišit. Nabízí se například řešení, kdy se horní, dolní, pravá a levá hrana obarví na různé barvy. Žák by

Obrázek 54 Brick Breakers – příklad generace jedné úrovně

pak musel vzít v úvahu směr, kterým se míček pohybuje, a zároveň barvu, které se při nárazu míčku do cihly dotkl. Dále je také třeba zmínit, že Scratch implicitně neumí vnímat dotek klonu jinou postavou a je tak třeba pokaždé míčku posílat zprávu, že k doteku došlo.

Rozšíření projektu je možné z mnoha úhlů pohledu. V pracovním listu je navržena například implementace více úrovní, různých vzorů, které jsou vytvářeny cihličkami různých barev, více životů, a různé vzhledy platformy a míčku. Všechna tato rozšíření přidávají projektu na složitosti a samostatně se jedná o více i méně náročné úkony. Nejsnazším rozšířením je pravděpodobně přidání více životů, kdy žák pouze okamžité ukončení hry nahradí scénářem snižujícím počet životů o jeden vždy, když se míček dotkne spodního okraje obrazovky. Toto snižování pak probíhá tak dlouho, dokud se počet životů nesníží na 0. Toto rozšíření je také vhodné doplnit vizualizací počtu životů, jejíž příklad je možné vidět na obrázku 55.

Obrázek 55 Brick Breakers – Hlavička hry s ukazatelem počtu životů

8.5 Kouzelné bludiště

Pracovní list Kouzelné bludiště je závěrečným projektem ze série *ScratchIT*. Tento pracovní list je silně individuální a vzhledem k velkému množství podprogramů a konceptů je mu přiřazena nejvyšší obtížnost 5 a celý pracovní list je k práci připojen jako příloha 6. Projekt, který však při práci s tímto pracovním listem vznikne, nemusí být nutně obtížnější než například projekty vytvářené podle předcházejících pracovních listů. Obtížnost výsledného projektu je závislá na schopnostech a ambicích každého žáka.

Ukázkový projekt pro tento pracovní list je založen na myšlence bludiště, ve kterém hráč postupně plní různé menší úkoly tzv. minihry. Po splněních všech miniher a odhalení všech tajemství v bludišti se hráč dostává z bludiště ven a hru vyhrává. Narozdíl od ostatních pracovních listů není možné zde zcela obsáhnout možnosti, jak lze projekt vylepšit. Tento pracovní list již počítá s tím, že žák má vlastní touhu a motivaci aktivně vymýšlet hry a výzvy pro svůj výsledný projekt. Při zpracování tohoto finálního zadání si většinu úkolů stanovuje sám žák s pomocí návrhů uvedených v pracovním listu.

V ukázkovém projektu jsou využity scénáře pro automatizovaný a ovládaný pohyb, skok na náhodnou pozici ze seznamu, ověřování splněných podmínek a různé další menší scénáře. Pro lepší dokreslení představy o projektu slouží v pracovním listu inspirační video a také odkaz na hotový projekt. Do ukázkového projektu byla zahrnuta pouze jedna minihra inspirovaná únikovými místnostmi, kde hráč musí interagovat s objekty okolo sebe za účelem úniku z místnosti. V ukázkovém projektu je cílem sesbírat 10 mrkví do košíku, jelikož hráč je v daném okamžiku přeměněn na králíka.

8.5.1 Praktické otestování zadání

Poslední pracovní list je velmi obecný. Cílem tohoto pracovního listu je představit žákům ukázkový projekt a motivovat je ke tvorbě projektu vlastního. Žáci by měli využít znalosti a dovednosti, které získali při práci s ostatními pracovními listy a použít se

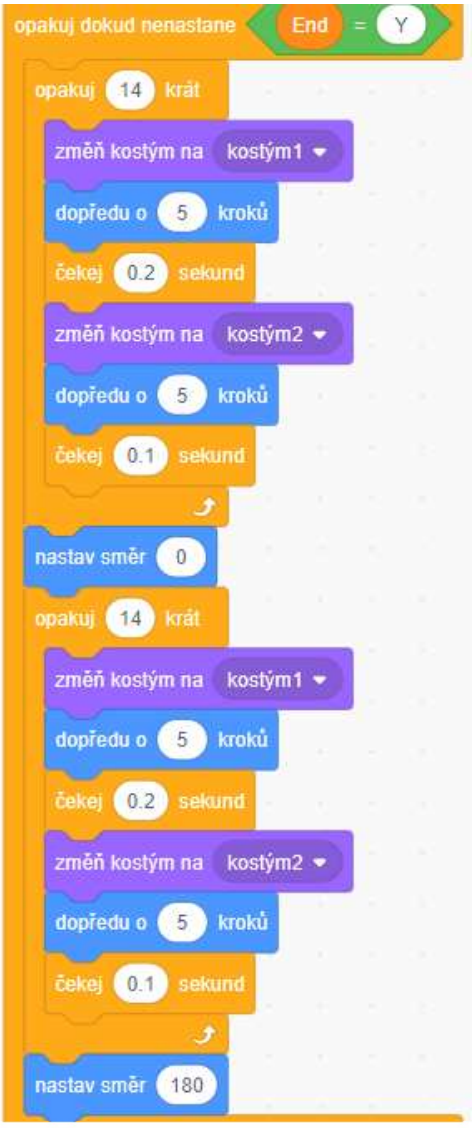
z chyb, které v předchozích projektech učinili. Z těchto důvodů je velmi obtížné obecně odhadnout, jaké problémy mohou žáci při tvorbě tohoto projektu řešit.

Tvorba závěrečného projektu byla časově nejnáročnější. Celý projekt byl vytvářen v průběhu jednoho a půl roku a není zcela možné odhadnout, přesný počet hodin, kolik jeho příprava zabrala, jelikož jsem na něm pracovala nepravidelně a pokaždé různě dlouhou dobu. Množství času, který žáci na tvorbu tohoto projektu budou potřebovat, se bude individuálně velmi lišit v závislosti na komplexitě a rozsahu zvolené hry. Pravděpodobně se však bude jednat o dobu výrazně kratší, než jaká byla věnována tvorbě ukázkového projektu.

Základní myšlenka projektu je relativně jednoduchá a zakládá na konceptech využívaných již v prvním pracovním listu. Základem hry je postava v bludišti, která je ovládána hráčem. Při procházení bludištěm však naráží hráč na různé překážky, které je možné vidět v náhledu hry na obrázku 56. Jednou z nejsnáze naprogramovatelných interaktivních překážek je statická modrá květina či pohybující se troll.

Obrázek 56 Kouzelné bludiště – náhled hry

Kód pro pohyb trolla je vyobrazen na obrázku 57. Detekce dotyku postavy čarodějnice a trolla je však vložena do cyklu, který se opakuje po celou dobu trvání hry a po doteku postav vyšle zprávu „troll“. Analogicky je řešena také detekce doteku čarodějnice a květiny.



Obrázek 57 Kouzelné bludiště – pohyb trolla

Před oběma těmito překážkami jsou umístěny informační otazníky, které se objevují v okamžiku, kdy se hráč dostatečně přiblíží k daným překážkám. (viz obrázky 58 a 59). Dalšími překážkami v bludišti jsou plot a dřevěný poklop, pro jejich odstranění/otevření musí hráč splnit určitý úkol. U všech těchto prvků se může žák odkázat například do pracovního listu případně vlastního projektu *Potop se!*, kde využíval vzájemnou detekci blízkosti či dotyku pohybujících se postav.

Obrázek 58 Kouzelné bludiště – informační otazník pro květinu

Obrázek 59 Kouzelné bludiště – informační otazník pro trolla

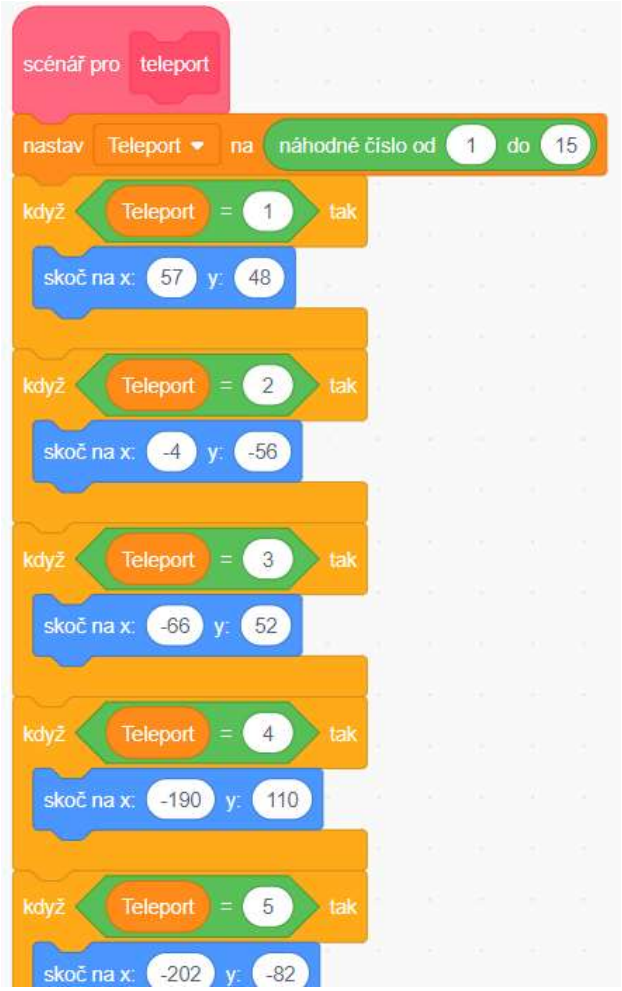
Dalším interaktivním prvkem v bludišti jsou schopnosti, které se může postava hráče v průběhu hry naučit. Konkrétně se jedná o schopnost proměnit se v myš či králíka a také objevení kouzla pro odstranění modré květiny. Stejným způsobem fungují také dva klíče, které hráč potřebuje získat za účelem otevření poklopu, kterým se dostane z bludiště ven. Všechny tyto prvky se hráči ukládají do inventáře, který je možné zobrazit pomocí stisku klávesy I na klávesnici. Umisťování předmětů do inventáře může být pro žáky do určité míry komplikované, jelikož je může hráč sesbírat v různém pořadí. Pro zachování jednoduchosti má každý předmět přiřazenou svou polohu v inventáři a po jeho sebrání se vždy objevuje na stejné pozici. Pro úspěšné dokončení hry musí hráč sesbírat a použít všechny předměty. Sbírání předmětů si žáci také vyzkouší ve druhém pracovním listu, kde je stejný koncept využit pro sbírání mincí a kyslíkových bublin.

V bludišti se také periodicky objevují pasti (jednou za 5 vteřin). Tyto pasti se objevují na 15 různých pozicích. Před aktivací pasti se nejprve na dané pozici objeví postupně se zvětšující symbol X. Pokud se hráč dotýká pasti v okamžiku, kdy se již zobrazí, o jakou past se jedná, je na něj aplikován daný efekt. Past může hráči ubrat život, může ho teleportovat na libovolnou pozici ze svého seznamu či na start. Jednou ze strategií tedy může být úmyslné stoupání na pasti s tím, že je hráč snadno přemístěn do jiné části bludiště. Při takovémto postupu však hráči hrozí, že ztratí život, či rovnou hru prohraje (pokud mu již

zbývá život jen jeden). Na stejných místech jako pasti se také objevují malá srdíčka, která mohou hráči doplnit chybějící životy. Tato srdíčka se začínají objevovat až v okamžiku, kdy hráč nemá plný počet životů, a na jednom místě jsou vždy pouze 10 sekund. Po vypršení této doby se opět skryjí a za 20 vteřin se objeví na další pozici. Jednotlivé pozice jsou všem prvkům přiřazovány náhodně. Po určení náhodné pozice skočí daný prvek na správné místo, kde se poté objeví. Pozice jsou pro všechny prvky uloženy ve dvou společných seznamech pro pozici x a y, které jsou vyobrazeny na obrázku 60.

Obrázek 60 Kouzelné bludiště – seznamy pozic

Původní řešení (viz řešení teleportu postavy na obrázku 61), kdy byly pozice pevně vepsány do jednotlivých podmínkových C bloků, bylo velmi neefektivní a projekt byl z tohoto důvodu předělán. Nové řešení na obrázku 62 je pro porovnání umístěno ihned pod řešením původním. Použití seznamů pozic a skok na pozici ze seznamu budou žáci znát ze třetího projektu s názvem Pexeso.



Obrázek 62 Kouzelné bludiště – původní kód skoku na pozici ze seznamu

Obrázek 61 Kouzelné bludiště – nové řešení s použitím seznamů souřadnic

Ukázkový projekt má dvě úrovně. Po získání schopnosti přeměny na králíka se však hráč může dostat do úrovně druhé – králičí nory. Do nory se hráč nedostane v lidské ani myší podobě. Náhled herního prostředí ve druhé úrovni ukázkového projektu je vyobrazen na obrázku 63. V noře musí hráč klikat na různé předměty a získat stanovený počet

mrkví. Koncepčně se zde jedná o prostou reakci na kliknutí či dotek postav, které si žáci průběžně budou procvičovat ve všech pracovních listech (ovládání hry pomocí tlačítek přímo ve hře namísto ovládání stiskem zelené vlajky). Na pořadí, v jakém hráč mrkve sesbírá, nezáleží. Po sesbírání dostatečného počtu mrkví se hráč vrací opět do bludiště, kde je odstraněn plot a hráč tak může hru úspěšně dokončit.

Obrázek 63 Kouzelné bludiště – náhled druhé úrovně hry

Závěrečné projekty žáků se velmi liší v kvalitě i komplexitě provedení. Jedná se však o kreativní práce, kde každý žák vytvořil hru podle vlastního nápadu. Ve výsledných projektech se objevují podobné chyby jako v projektech předchozích a vzhledem k rozmanitosti není možné všechny chyby obsáhnout. Obecně je však znát, že v závěrečných projektech žáci dělají méně chyb než v projektech předchozích, což nasvědčuje tomu, že se ze svých chyb v určitých aspektech poučili.

8.6 Vize využití pracovních listů a jejich rozšiřování

Výsledná série pracovních listů a ukázkových projektů může sloužit pro podporu studentů přímo ve výuce či jako doplňující materiál pro práci mimo výuku. Není nezbytně

nutné, aby byly listy použity právě s prostředím Scratch, ale mohou po lehké adaptaci posloužit i pro podporu práce v jiných prostředích. Příklady možných alternativních prostředí jsou popsány v následující kapitole. Každý projekt popisovaný v těchto pracovních listech je možné v závislosti na úrovni dovedností jednotlivých žáků různě rozšiřovat, díky čemuž se učiteli dostane velké množství různorodých projektů založených na stejné myšlence. Tyto projekty je dále možné použít pro demonstraci různých provedení či pro rozbor a následnou opravu nedostatků.

Je pouze na učiteli, zda se rozhodne žákům pracovní list doplnit o vlastní výklad, či nikoliv. Jednotlivé projekty zpracované ve formě pracovních listů by mohly také sloužit pro vytyčení cíle části výuky věnované programování. Celá práce v rámci hodin informatiky zaměřených na toto téma by tak směřovala ke splnění jednoho či více pracovních listů.

Tuto sérii pracovních listů by mohly předcházet také listy vysvětlující funkce jednotlivých bloků či případně listy obsahující jednodušší projekty na procvičování použití jednotlivých konceptů. Kompletní sada pracovních listů by pak fungovala jako ucelený materiál na podporu výuky programování, který by učiteli umožňoval individualizovat tempo práce jednotlivých žáků podle jejich možností a schopností.

V pracovních listech záměrně nejsou uvedeny přesné bloky, které žáci mají při tvorbě projektu využít, jelikož programování je do velké míry kreativní práce. Žáci tak dostávají pouze základní kostru postupu a musejí zapojit vlastní fantazii k tomu, aby mohli projekt dokončit. Ze stejného důvodu na řízených pracovních listech není uveden odkaz na hotový projekt, ale pouze videozáznam dokumentující celý průběh dané hry. Žáci tak mají inspiraci, od které se mohou odrazit, ale jejich řešení konkrétních dílčích problémů není ovlivněno cizími konstrukcemi kódu. Hotové projekty jsou však součástí této práce a měly by být k dispozici učitelům, kteří by se potenciálně rozhodli pracovní

listy využít. Každý by pak měl možnost žákům například i v hodině ukázat průběh hry či v případě obtíží ukázat část kódu, která by pro daného žáky byla příliš komplikovaná a znemožnila by jim posun vpřed.

V ideálním případě by mnou navržené pracovní listy neměly u žáků rozvíjet pouze schopnost programovat a informaticky myslet, ale zároveň by se měly podílet na rozvoji jejich kreativity a svobodného myšlení.

9 Různé alternativy k prostředí Scratch

V této kategorii popisuji další tři prostředí, která jsou určená pro dětské programátory. Pro co největší dostupnost jsem se zaměřila pouze na prostředí, která je možné využívat zdarma. Všechna tato prostředí je možné využít pro výuku programování. S určitými modifikacemi by bylo možné využít mnou navržené pracovní listy i pro práci v jiném prostředí, než je Scratch. Pro Kodu Game Lab by byla modifikace nejkomplikovanější, jelikož se na rozdíl od všech ostatních prostředí jedná o 3D vývojové prostředí, a tak musejí žáci vzít v potaz také třetí rozměr aplikace.

9.1 Kodu Game Lab

Kodu je určeno primárně pro tvorbu her. Velmi atraktivní je 3D prostředí, ve kterém je hra tvořena. Stejně jako Scratch, Kodu nevyžaduje žádné dlouhé psaní kódu a programování sestává ze skládání jednotlivých bloků typu „když to, tak udělej“. Význam bloků je reprezentován obrázky a doplněn o textová vysvětlení.

Z mého pohledu je prostředí obtížnější pro navigaci a vyžaduje z počátku více vysvětlování ze strany učitele před tím, než může žák samostatně začít kód tvořit. Tento nedostatek je však vyvážen právě svým 3D zpracováním, které je pro žáky velmi motivující. V okamžiku, kdy však žák pochopí základní principy ovládání tohoto prostředí, otevírá se mu svět téměř neomezených možností.

Dalším rozdílem od prostředí Scratch je nutnost Kodu Game Lab instalovat do počítače. Kodu nemá verzi spustitelnou v prohlížeči. Zároveň také nepodporuje systémy Android, tudíž žáci nemohou programovat na telefonu či tabletu.

9.2 Code.org

Code.org je velmi dobrá platforma pro úvod do programování. Kromě prostředí přímo určeného pro vlastní tvorbu her a dalších projektů, Code.org také nabízí širokou

nabídku kurzů, které slouží k uvedení žáků do problematiky programování. Všechny kurzy jsou tematicky populárně laděné. Žák si může zvolit kurz s tématikou hry Minecraft či Angry Birds, dále je v nabídce programování ve stylu Disney Frozen, Flappy Bird či Star Wars. Jazyk u jednotlivých kurzů lze ve velké většině nastavit na český. Videa, která doprovázejí jednotlivé kurzy, pak mají české titulky.

Pro absolvování jednotlivých kurzů není třeba registrace. Pro vlastní tvorbu projektů je již však nutné mít vlastní účet. Samotné prostředí působí relativně jednoduše a je podobné prostředí Scratch. Jednotlivé bloky kódu jsou čistě textové, bez obrázkových vysvětlivek. Ke každému bloku nabízí prostředí příklad jeho využití, což je velmi užitečná vlastnost, která ve Scratch chybí.

Velkou výhodou Code.org je možnost psát kód místo jeho skládání z bloků a případné přepínání mezi těmito dvěma režimy. Přestože je k dispozici v tomto prostředí menší množství kategorií příkazů, je zde možné vytvořit podobné projekty jako v prostředí Scratch. Code.org navíc také nabízí prostředí pro tvorbu aplikací či webů. Všechna tato prostředí jsou od sebe přehledně oddělená a nabízí tak žákům velmi širokou nabídku aktivit spojených s programováním. Ačkoliv se toto prostředí zdá být ideální, má jeden velký nedostatek. Jednotlivá studia na vlastní tvorbu není možné přepnout do češtiny. Prostředí je tak vhodné spíše pro starší žáky, kteří si s příkazy v anglickém jazyce snáze poradí.

9.3 Tynker

Tynker opět nabízí rozmanitou paletu aktivit pro žáky. Kromě blokového programování mají žáci možnost programovat v jazyce Python či JavaScript, dále také mohou vytvářet HTML a CSS dokumenty. Mimo jiné Tynker také nabízí prostředí pro editaci textur a tvorbu nových entit či modifikací pro hru Minecraft.

V blokovém programování opět nacházím určitou podobnost s prostředím Code.org a Scratch. Na rozdíl od předchozích prostředí však vůbec nepodporuje český jazyk. Z tohoto důvodu bych ho doporučila pro starší žáky, kteří již s jazykovou bariérou nebudou mít takové obtíže jako žáci mladší.

Jako hlavní výhodu tohoto prostředí spatřuji v možnosti volby programovacího jazyka. Zejména starší žáci, kteří již touží po realističtější zkušenosti s programováním, mohou vnímat možnost kód napsat místo pouhého sestavení z bloků jako velmi motivující.

Samotné programovací prostředí je současně navrženo jako hra. Žáci zvyšují svoji úroveň programováním, tvorbou her, absolvováním lekcí či řešením různých zadaných výzev. V závislosti na jeho úrovni jsou danému žáku zpřístupňovány nové postavy a možnosti programování. Tato vlastnost prostředí Tynker je velmi zajímavá a může působit jako velká motivace pro žáky všeho věku. Také z tohoto důvodu je velká škoda, že prostředí zatím není přeloženo do češtiny.

10 Závěr

Cílem mé diplomové práce bylo zhodnotit použitelnost prostředí Scratch pro tvorbu složitějších projektů, které by sloužily jako rozšiřující zadání pro nadané žáky. V tomto ohledu musím zkonstatovat, že použitelnost prostředí pro komplexní projekty do velké míry závisí na struktuře daného projektu. Jak uvádím v kapitole 7.6 Retrospektiva, po vytvoření projektu s vysokým počtem postav se projekt přestal ukládat a bez ukládání nebylo možné pokračovat v programování. Pro správné fungování projektu je zřejmě třeba maximálně zefektivnit scénáře a minimalizovat počet postav v projektu. Po těchto úpravách by však projekt pravděpodobně mohl být dokončen a prostředí Scratch by mohlo pro tento účel být použito. Tento problém je třeba vzít v úvahu při plánování a zadávání komplexnějších úkolů, aby bylo možné předejít nutnosti projekt opustit nedokončený.

V závislosti na obtížích spojených s tvorbou prvotně zamýšleného projektu byl výstup mé diplomové práce lehce pozměněn a místo jednoho metodického materiálu jsem vytvořila pět pracovních listů různé obtížnosti. Při tvorbě původního projektu Osadníci z Katanu a následně při tvorbě ukázkových projektů pro jednotlivé pracovní listy jsem identifikovala dvě základní fáze tvorby projektu: fázi teoretické přípravy a fázi vlastního programování. Druhá fáze se posléze dělí na několik dalších fází, mezi které se řadí: fáze přípravy základních prvků, fáze návrhu možného řešení, implementace zvoleného řešení, testování a oprava programu a fáze jeho ladění. Mezi těmito fázemi se žáci mohou libovolně přesouvat v závislosti na konkrétních činnostech, které v souvislosti s vývojem programu provádějí.

Ke každému pracovnímu listu jsem vytvořila krátké video zachycující průběh hry, které má sloužit pro žáky jako úvodní inspirace bez ukázky hotového kódu. Práce je také doplněna o hotové ukázkové projekty, které obsahují nejen základní zadání z pracovních

listů, ale také všechna zmíněná rozšíření. Na závěr jsem krátce zhodnotila alternativní prostředí, která by pro práci s pracovními listy mohla být využita. Zatímco Kodu Game Lab by vyžadovalo velkou míru úprav původních pracovních listů, prostředí Code.org a Tynker by bylo možné využít s pracovními listy téměř beze změny.

11 Reference

1. PAPERT, Seymour, 1980. Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books. ISBN 0-465-04627-4.
2. Logo History. Logo Foundation [online]. 2015 [cit. 2021-6-22]. Dostupné z: https://el.media.mit.edu/logo-foundation/what_is_logo/history.html
3. WING, Jeannette M., 2006. Computational thinking. Communications of the ACM. 49(3), 33-35. ISSN 0001-0782. doi:10.1145/1118178.1118215
4. ČERNOCHOVÁ, Miroslava, Jiří ŠTÍPEK a Petra VAŇKOVÁ, 2019. Programování ve Scratch II (projekty pro 2. stupeň základní školy) [online]. [cit. 2020-4-17]. Dostupné z: https://imysleni.cz/ucebnice/programovani-ve-scratchi-ii-projekty-pro-2-stupen-zakladni-skoly/
5. BERRY, Miles, 2021. Computational thinking. Primary Schools Partnership March Newsletter [online]. 3/2021, 13-14 [cit. 2021-5-1]. Dostupné z: https://www.roehampton.ac.uk/globalassets/documents/education/primary-partnerships/march-newsletter.pdf
6. PECINOVSKÝ, Rudolf. Proč učit programování na základní škole. In: Česká škola [online]. 10/09/2001 [cit. 2021-5-1]. ISSN 1213-6018. Dostupné z: http://www.ceskaskola.cz/2001/09/rudolf-pecinovsky-proc-ucit.html
7. About scratch. Scratch [online]. [cit. 2021-04-03]. Dostupné z: https://scratch.mit.edu/about
8. NĚMEC, Petr. Scratch – univerzální nástroj ICT na ZŠ I. Metodický portál: Články [online]. 27. 06. 2018, [cit. 2021-04-24]. Dostupný z: https://clanky.rvp.cz/clanek/c/Z/21690/SCRATCH---UNIVERZALNI-NASTROJ-ICT-NA-ZS-I.html. ISSN 1802-4785.

9. Scratch in Many Languages, 2020. Scratch Wiki [online]. [cit. 2021-02-22]. Dostupné z: https://en.scratch-wiki.info/wiki/Scratch_in_Many_Languages
10. Scratch Statistics, 2021. Scratch [online]. [cit. 2021-04-03]. Dostupné z: https://scratch.mit.edu/statistics/
11. Nintendo's shining star. Gamecubicle.com [online]. [cit. 2021-04-15]. Dostupné z: http://www.gamecubicle.com/features-mario-nintendo_shining_star.htm
12. GIBSON, Jeremy, 2015. Introduction to Game Design, Prototyping, and Development. Crawfordsville: Pearson Education. ISBN 978-0-321-93316-4.
13. SALEN, Katie a Eric ZIMMERMAN, 2004. Rules of Play – Game Design Fundamentals. Massachusetts: MIT Press. ISBN 0262240459.
14. ROGERS, Scott, 2010. Level Up!: The guide to great vide game design. New Jersey, USA: John Wiley. ISBN 978-0-470-68867-0.
15. LAMB, Annette a Larry JOHNSON. Scratch: Computer programming for 21st century learners [online]. Bowie: E L Kurdyla Publishing LLC, 2011. 64-68,75 s. Copyright - Copyright E L Kurdyla Publishing LLC Apr 2011; Další obsah dokumentu - Photographs; Poslední aktualizace - 2017-11-18.
16. Rámcový vzdělávací program pro základní vzdělávání, 2017. In: Národní ústav pro vzdělávání [online]. Praha: MŠMT [cit. 2020-4-9]. Dostupné z: http://www.msmt.cz/file/43792/
17. Rámcový vzdělávací program pro základní vzdělávání, 2021. In: Národní ústav pro vzdělávání [online]. Praha: MŠMT [cit. 2021-5-10]. Dostupné z: http://www.nuv.cz/file/4982/
18. DUFKOVÁ, Andrea. Osadníci z Katanu. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/projects/386280851/

19. DUFKOVÁ, Andrea. Studio projektů pro DP Dufková. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/27834444/
20. Eynsham coding club Friday, 2018. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/5897455/
21. Eynsham coding club Monday, 2018. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/5897453/
22. Stanton Harcourt Coding Classes, 2019. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/5896845/
23. Úvod do algoritmizace – závěrečné práce letní semestr, 2018. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/5085993/
24. Úvod do Algoritmizace – závěrečné práce za zimní semestr, 2017. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/4521086/
25. Úvod do algoritmizace, 2017. In: Scratch [online]. [cit. 2021-6-24]. Dostupné z: https://scratch.mit.edu/studios/4367156/

Seznamy příloh

Přílohy vázané v práci

Příloha 1 – Průvodní list k sérii pracovních listů

Příloha 2 – Pracovní list k projektu Bezedný košík?!

Příloha 3 – Pracovní list k projektu Potop se!

Příloha 4 – Pracovní list k projektu Pexeso

Příloha 5 – Pracovní list k projektu Brick Breakers

Příloha 6 – Pracovní list Vytvoř si vlastní hru! (projekt Kouzelné bludiště)

Volné přílohy

Ukázkový projekt Bezedný košík?! – BezednyKosik.sb3

Ukázkový projekt Potop se! – PotopSe.sb3

Ukázkový projekt Pexeso – Pexeso.sb3

Ukázkový projekt Brick Breakers – BrickBreakers.3sb

Ukázkový projekt Kouzelné bludiště – KouzelneBludiste.3sb

SCRATCH IT

Právě se Vám dostal do rukou pracovní list ze série „Scratch IT"! Tato série se zaměřuje na výuku programování v programovacím jazyce Scratch a pracovní listy slouží jako podklad pro učitele či pro samotné žáky.

Průvodce každým listem je chlapec Ignác, zkráceně Iggy, kterého ze všeho nejvíc baví převádět zkušenosti z každodenního života na počítačové hry.

Každý pracovní list je označený podle obtížnosti různým počtem hvězdiček. Čím více hvězdiček má, tím je obtížnější.

Každý pracovní list obsahuje seznam kroků, které pro vypracování projektu jsou potřeba. Každý list také uvádí jednotlivé oblasti dovedností nezbytné pro dokončení projektu. V návrzích však mohou být i další.

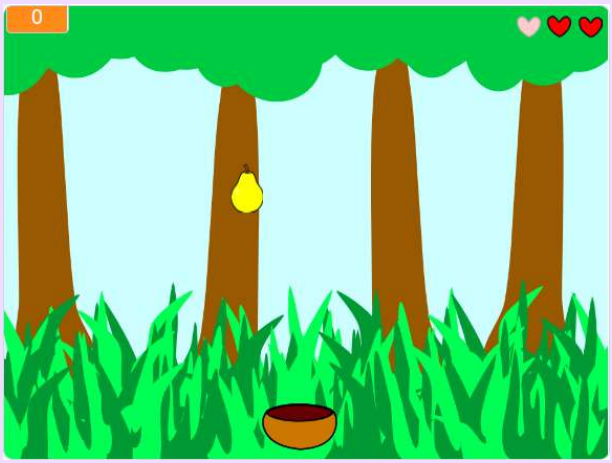
Problematické sekce jsou popsané v červeném rámečku.

V zeleném rámečku jsou uvedené možnosti, jak projekt vylepšit, či co do něj přidat.

Zlaté rámečky jsou místem pro různé tipy, které mohou žákům pomoci s vypracováním projektu. Také se zde objevují obecné rady pro tvorbu projektů v prostředí Scratch.

Bezedný košík!?

Iggy spolu s rodiči sklízel jablka, a tak ho to bavilo, že se rozhodl naprogramovat si hru ***Bezedný košík***. Zkus si takovou hru naprogramovat spolu s ním. Co budeš do košíku chytat je úplně na tobě. Iggy si vybral jablka a hrušky. Co si vybereš ty?



Co budeš v tomto projektu potřebovat?

- **cykly** pro opakované padání předmětů
- pohyb košíku **pomocí šipek**
- **proměnné** pro uchovávání skóre hráče

Jak na to?

1. Vyber si, co budeš chytat a do čeho.
2. Připrav si pozadí hry.
3. Naprogramuj košík tak, aby se pohyboval jen doleva a doprava.
4. Naprogramuj padající předmět tak, aby se objevoval na náhodné pozici na horním okraji obrazovky a padal postupně dolů.
5. Když se padající objekt dotkne košíku, musí se vrátit zpět nahoru na obrazovku a začít padat znova. Nezapomeň, že hráč by měl dostat nějaké body za chycení objetu.
6. Rozhodni, kdy hra skončí – kolik musí hráč posbírat objektů nebo kolik musí získat bodů. Skončí hra, pokud hráč nechytí padající objekt?

Zkus použít **VLASTNÍ BLOK** pro scénář padání objektů.

Když vnoříš dva cykly do sebe, můžeš jednoduše nastavit chytání objektů a kontrolu, zda nespadly mimo košík.

Nedovol košíku utéct z obrazovky!

Místo spouštění hry jen zelenou šipkou naprogramuj tlačítko pro start hry. Můžeš také přidat tlačítko „Hrát znovu" pro opětovné spuštění hry, když skončí.

Neboj se vytvořit více úrovní, kdy bude hráč sbírat jiné předměty!

Dej hráči životy. Ubírej mu pomocí kostýmů třeba srdíčka.

Přidej předměty, které budou hráči body ubírat, třeba špinavé ponožky nebo červivá jablka. Také může každý předmět dávat jiný počet bodů.

Upravuj rychlost pádu předmětů podle počtu bodů – čím více bodů, tím rychleji ovoce padá.

Tady najdeš inspiraci, jak hra může vypadat: https://youtu.be/MwcfA9f3BeY

Potop se!

Na dovolené u moře se Iggy potápěl a hledal v písku poklady. Když se vrátil domů, chtěl si potápění užít znova, a tak si naprogramoval hru ***Potop se***. Dokážeš také naprogramovat honičku se žraloky? Vyzkoušej to spolu s Iggym.

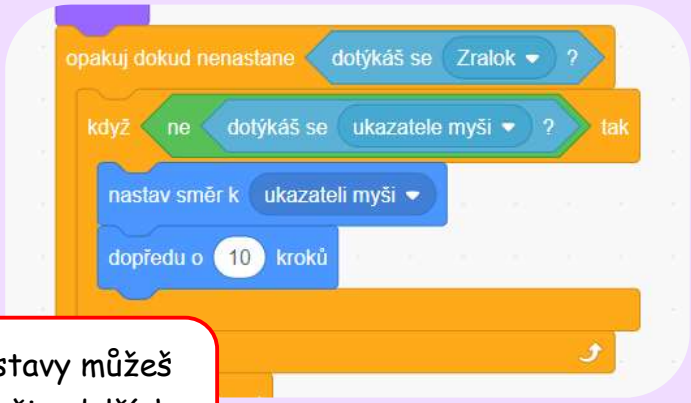
Co budeš v tomto projektu potřebovat?

- **cykly** pro objevování mincí
- pohyb potápěče sledující **kurzor myši**
- pohyb žraloka, který **následuje potápěče**
- **proměnné** pro uchovávání mincí

Jak na to?

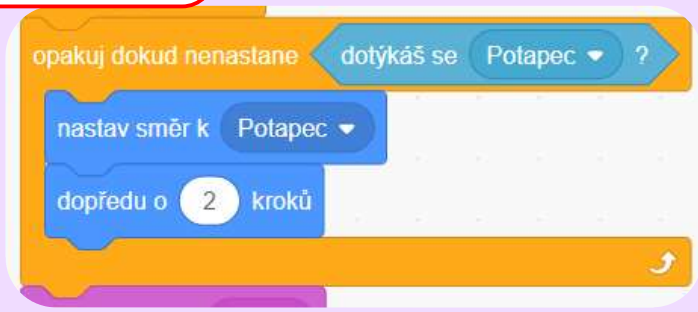
1. Připrav si postavy pro potápěče, žraloka a minci.
2. Připrav si pozadí hry.
3. Naprogramuj potápěče tak, aby po spuštění hry následoval kurzor myši.
4. Naprogramuj žraloka tak, aby s malým zpožděním následoval potápěče.
5. Nastav hru tak, aby hra skončila, když se žralok dotkne potápěče.
6. Naprogramuj minci tak, aby se objevovala na náhodné pozici. Když se potápěč dotkne mince, měl by dostat body, a mince by se měla přemístit na další náhodné místo.
7. Nezapomeň hru nastavit tak, aby měla konec. Hráč prohrává, když se dotkne žraloka, a naopak vyhrává, když nasbírá dostatečný počet bodů/mincí.

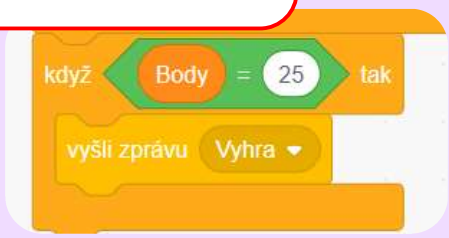
Zkus použít **VLASTNÍ BLOK** pro objevování mince.



Při nastavování orientace postavy můžeš také využít pozici kurzoru myši a dalších postav.

Nezapomeň nastavit počet bodů, které hráč musí nasbírat, aby vyhrál.





Místo spouštění hry jen zelenou šipkou naprogramuj tlačítko pro start hry. Můžeš také přidat tlačítko „Jak hrát?“ nebo tlačítko pro opětovné spuštění hry, když skončí.

Nastav mince tak, aby se po určité době přemístily jinam, ikdyž je potápěč nesebere.

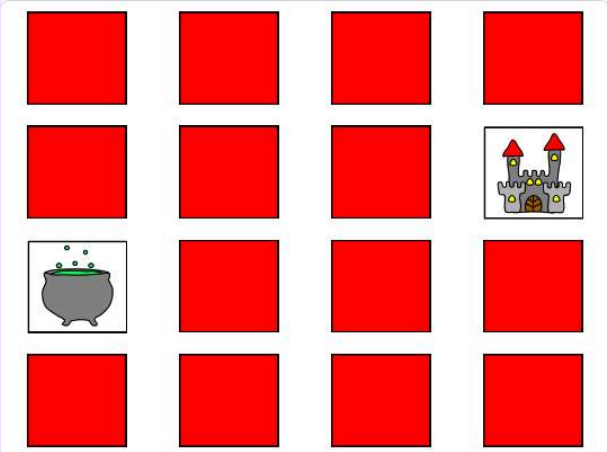
Přidej mincím různé hodnoty a kostýmy.

Přidej do hry hladinu kyslíku potápěče. Stejně jako mince se mohou objevovat bublinky kyslíku, které potápěč musí sbírat, aby se neutopil. Musíš také vytvořit novou proměnnou pro kyslík, která se bude automaticky s časem snižovat.

Tady najdeš inspiraci, jak hra může vypadat: https://youtu.be/7mEABSTi258

Pexeso

Jako malý Iggy miloval hru Pexeso. Často se mu ale některé kartičky poztrácely, a tak hru nemohl dohrát. Občas si tuto hru rád zahraje i nyní a aby se nemusel bát, že nebude mít všechny kartičky, pexeso si naprogramoval. Zvládneš to také?



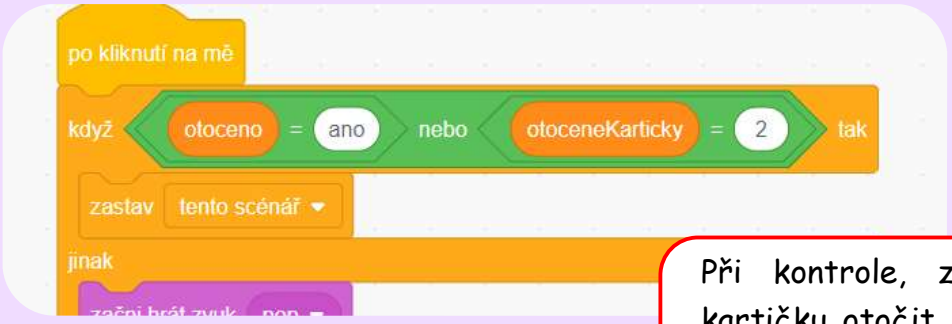
Co budeš v tomto projektu potřebovat?

- několik **lokálních a globálních proměnných** pro ovládání hry
- **operátory** pro porovnávání hodnot proměnných
- **seznamy** jednotlivých poloh kartiček
- **cykly** a **podmínkové bloky**
- **odesílání a přijímání** zpráv

Jak na to? Část první

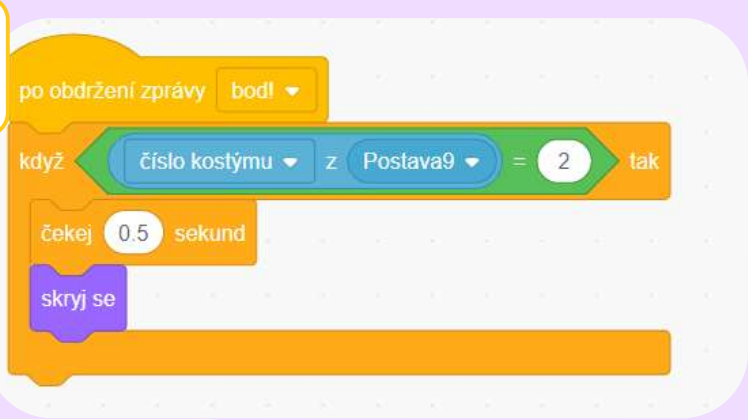
1. Připrav si dvě kartičky, které budou tvořit pár.
2. Připrav si pozadí hry.
3. Naprogramuj kartičky tak, aby každá měla vlastní (lokální) proměnnou pro stav otočení a po kliknutí na ně se změnil jejich kostým a stav otočení na otočeno.
4. Přidej proměnnou, která bude počítat počet aktuálně otočených kartiček a tuto vlastnost kartičkám přidej.
5. Když je splněná podmínka pouze dvou otočených kartiček, nezapomeň kartičkám nastavit kontrolu toho, zda jsou otočené kartičky, které tvoří pár.

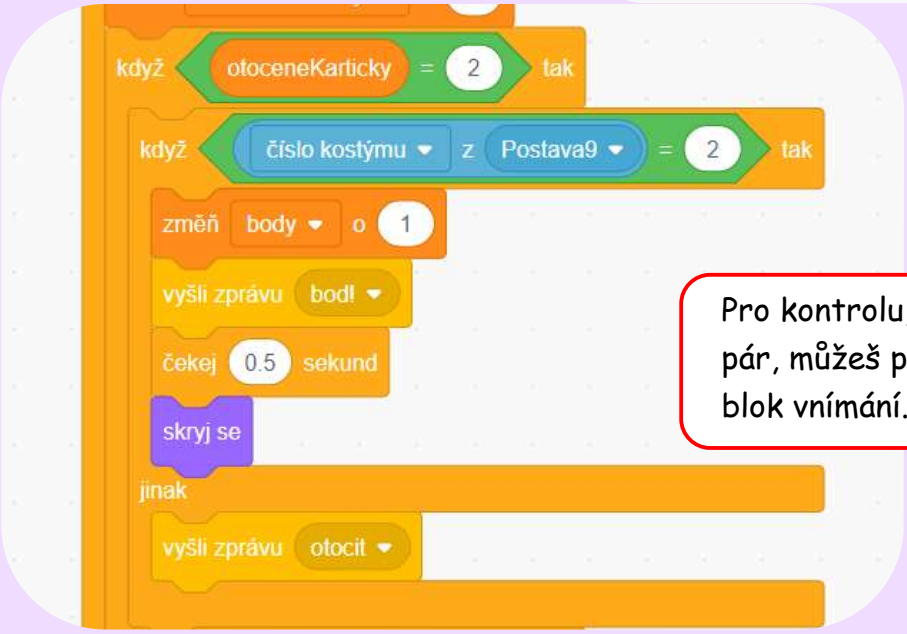
6. Kartičky se vzájemně nevidí, musíš jim tedy nastavit posílání zpráv tak, aby věděly, že se již mají otočit zpět, a také když hráč najde správný pár.
7. Pro kontrolu správnosti páru můžeš použít blok číslo kostýmu postavy. Kartičky v jednom páru si musí vzájemně tuto hodnotu zkontrolovat.
8. Dodělej další kartičky a umísti je na hrací plochu tak, jak chceš, aby byly rozmístěny. Můžeš zkopírovat první dvě, aby v sobě již obsahovaly potřebné scénáře. Je dobré si kartičky očíslovat (1, 2, 3, 4, …).



Při kontrole, zda může hráč danou kartičku otočit, je třeba použít několik podmínek. Pomocí logických operátorů je můžeš všechny vložit do jednoho bloku.

Použij **zprávy** pro komunikaci mezi jednotlivými kartičkami.





Pro kontrolu, zda hráč otočil správný pár, můžeš použít například zobrazený blok vnímání.

s

Jak na to? Část druhá

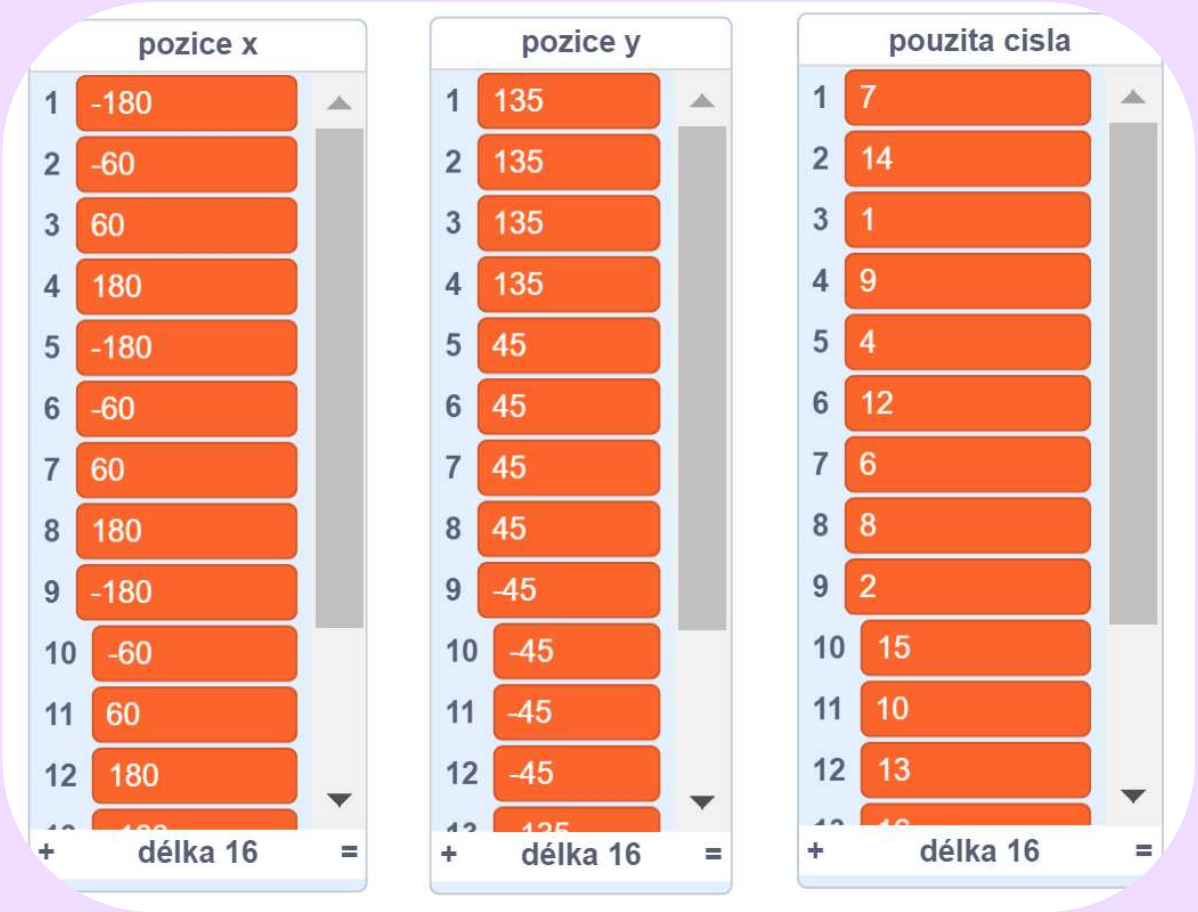
Pro tuto část si můžeš vytvořit pomocnou postavu pro uložení všech scénářů, které slouží pro ovládání hry.

1. Vytvoř si 3 seznamy (pozice x, pozice y a seznam pro náhodné přiřazení hodnot x a y).
2. Do seznamů pro pozice x a y napiš jednotlivé hodnoty, které opíšeš z připravených a umístěných kartiček.
3. Naprogramuj cyklus, který naplní seznam pro náhodné přiřazení poloh čísly od 1 do x, kde x je počet kartiček, které máš připravené.
4. U každé kartičky připrav krátký scénář, ve kterém kartičce nastavíš polohu, na kterou má skočit podle seznamu.

Použij operátor <*ne*>, aby se do seznamu nedostalo žádné číslo dvakrát. Tento scénář je třeba opakovat tak dlouho, dokud seznam nemá délku rovnu počtu kartiček.

Na začátku každé hry nezapomeň tento seznam pozic **vymazat a vyplnit znovu.**

Každou kartičku naprogramuj tak, aby si na začátku hry vzala správnou pozici ze seznamu. Všimni si, že používáme dva bloky pro práci se seznamy **vloženými do sebe.**



Zde můžeš vidět příklad, jak by mohly vypadat tři seznamy. V seznamu *pouzita cisla* už jsou vygenerované hodnoty 1–16 v náhodném pořadí. Hodnoty v seznamech pro *pozice x a y* jsou vloženy tak, aby na stejných řádcích byla čísla, která k sobě patří.

Místo spouštění hry jen zelenou šipkou naprogramuj tlačítko pro start hry. Můžeš přidat také návod, či tlačítko pro opětovné spuštění hry, když skončí.

Přidej do hry počítadlo otočení. Na konečné obrazovce pak můžeš hráči vypsat, kolik tahů na dokončení hry potřeboval.

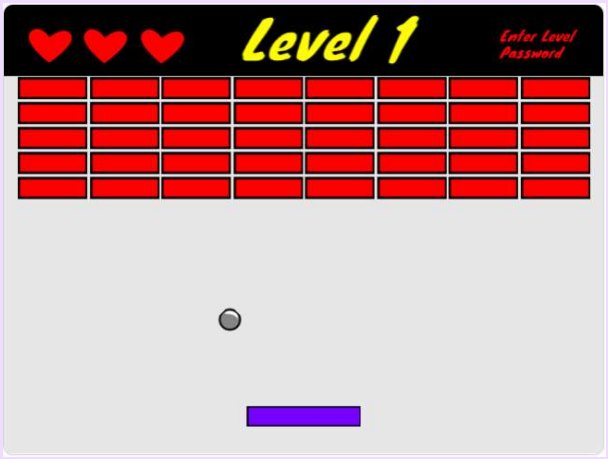
Stopuj hráči čas, za jaký hru dokončí. Na konci hry mu tento čas pak ukaž.

Doplň hru o zvukové efekty. Můžeš přidat zvuk třeba ke každému otočení kartičky.

Tady najdeš inspiraci, jak hra může vypadat: https://youtu.be/dVa6KY1Iqf4

Brick Breakers

Kdysi dávno chodily děti hrát hry do arkádové herny. Hry se hrály na podobných přístrojích, jako jsou dnes výherní automaty. Iggyho tatínek mu vyprávěl o hře, kde kulička rozbíjela cihličky, a Iggymu se hra tak líbila, že se ji pokusil vytvořit sám. Zkus to také!



Co budeš v tomto projektu potřebovat?

- **klonování postav**
- **proměnné** pro ovládání hry
- **operátory** pro porovnávání hodnot proměnných
- **cykly** a **podmínkové bloky**
- **odesílání a přijímání** zpráv

Jak na to?

1. Připrav si pozadí hry a postavy pro cihličku, kuličku a platformu, od které se bude kulička odrážet. Dej si pozor na velikost cihličky! Musí se ti jich vedle sebe vejít několik do řádku.
2. Naprogramuj platformu tak, aby s ní mohl hráč pohybovat pouze horizontálně.
3. Naprogramuj kuličku tak, aby se po spuštění hry rozpohybovala a odrážela se od platformy a od hran herního pole.
4. Přidej do postavy cihličky kód pro generaci úrovně pomocí klonování postavy. Všimni si, že se určité akce stále opakují.

5. Přidej do hry omezení, že se kulička nesmí dotknout spodního okraje herní obrazovky. Můžeš využít třeba čáru, kterou umístíš pod úroveň platformy. Pokud se kulička této čáry dotkne, hra končí.
6. Nastav kuličku tak, aby se odrážela i od klonů cihliček. Pro tento krok použij třeba zprávu, kterou jednotlivé klony kuličce pošlou.
7. Vytvoř scénář pro klony cihličky tak, aby se po srážce s kuličkou skryly.
8. Nezapomeň nastavit konec hry, když hráč zničí všechny cihličky. Můžeš použít například proměnnou, která bude počet cihliček počítat, a podle ní můžeš počet zbývajících cihliček kontrolovat.

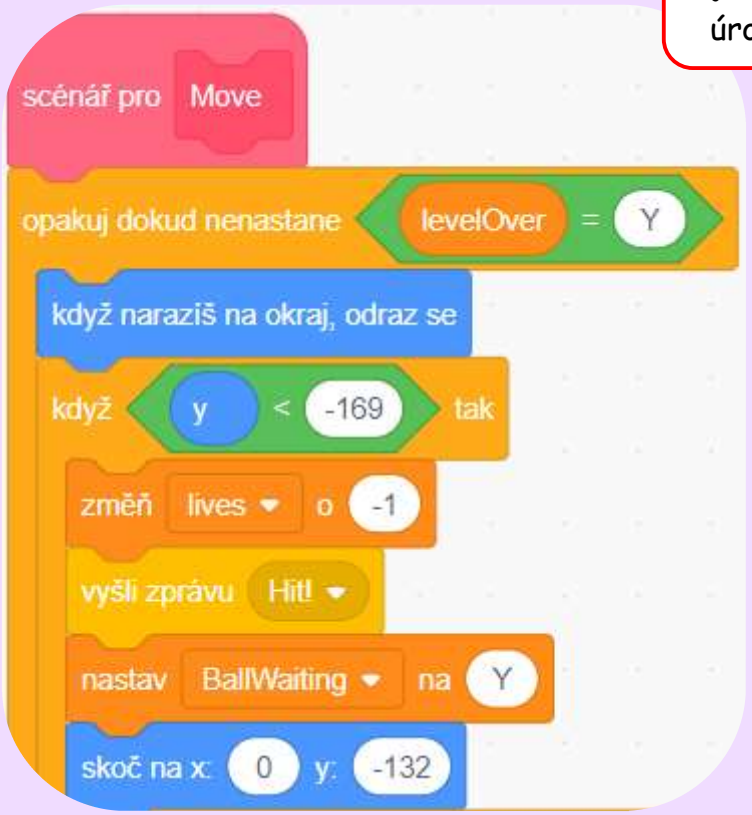
Pro ovládání platformy můžeš použít klávesy nebo kurzor myši.

Nezapomeň všem postavám nastavit úvodní pozici, kde budou vždy začínat.

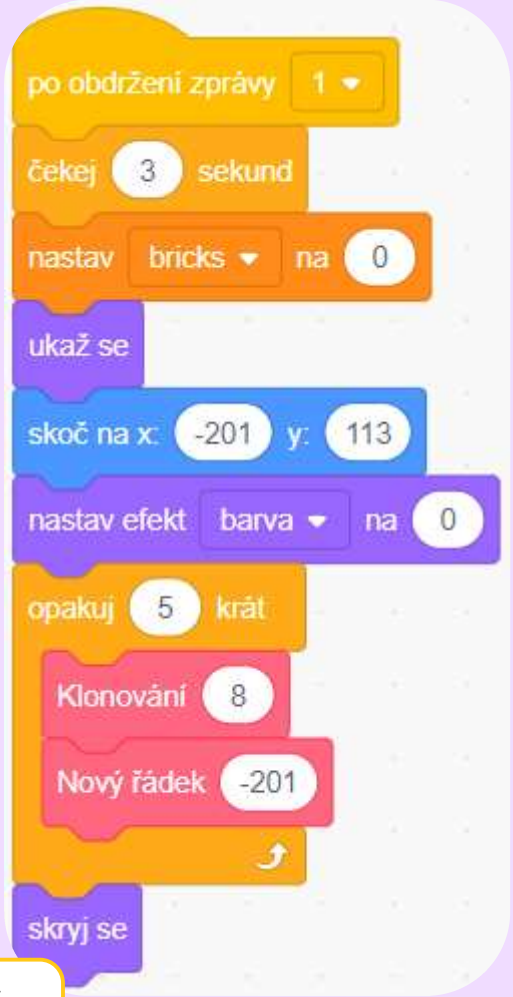
Použij **zprávy** pro komunikaci mezi klony cihličky a kuličkou.

Odrážení kuličky od klonů cihliček může být obtížnější. Aby se daly dodržet fyzikální zákony, musely by být hrany cihliček odlišené a při odrážení kuličky by se musela brát v potaz hrana cihličky i směr, kterým kulička letí.

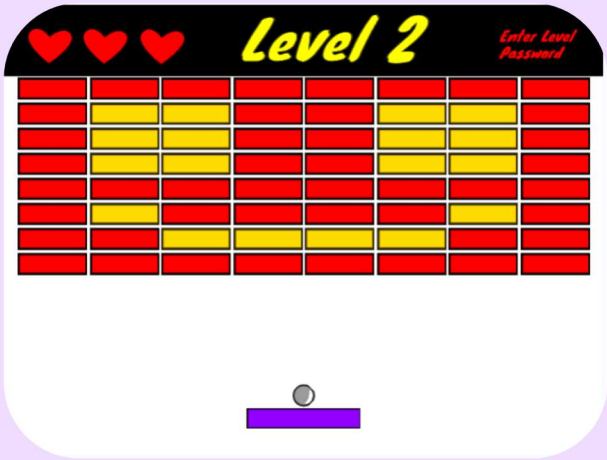
Zkus použít **vlastní bloky** pro jednotlivé akce potřebné pro generaci úrovně.

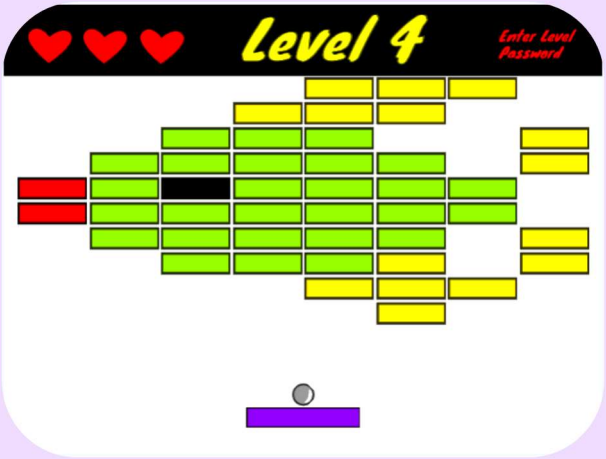


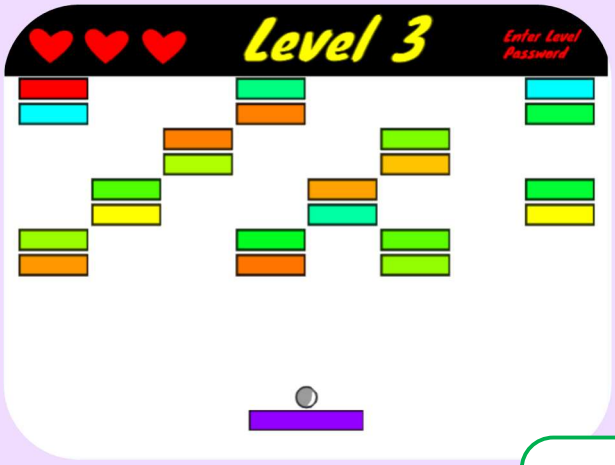
Pro omezení dotyku kuličky se spodním okrajem můžeš využít více možností než jen čáru. Tady na obrázku můžeš vidět řešení pomocí detekce hodnoty pozice Y.

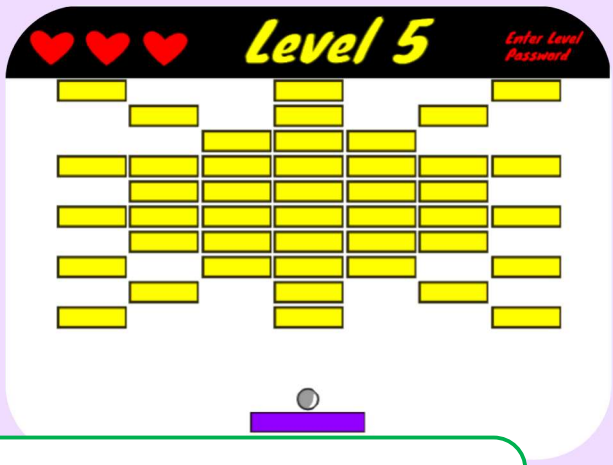


Tady můžeš vidět příklad scénáře na generaci úrovně.









Přidej do projektu další úrovně. Inspiraci k uspořádání cihliček můžeš najít třeba na obrázcích tady.

Přidej do hry kompletní možnost ovládání tlačítky uvnitř hry. Můžou to být například tlačítka pro instrukce k ovládání hry, spuštění hry, či návrat na úvodní obrazovku.

Přidej hráči životy a možnost udělat chybu. Životy můžeš znázornit třeba jako srdíčka v rohu obrazovky.

Pokud si na to troufáš, přidej do hry různé kostýmy pro kuličku a platformu, ze kterých si může hráč vybrat, nebo které se mu zpřístupní po splnění určité úrovně.

Doplň hru o zvukové efekty. Můžeš přidat zvuk třeba ke každému zničení cihličky nebo zvuk pro konec hry.

Tady najdeš inspiraci, jak hra může vypadat: https://youtu.be/vD5TSNUGfks

Naprogramuj si vlastní hru!

Iggy chce být herním programátorem. Na tuto práci však musí mít spoustu originálních nápadů na nové a zajímavé hry. Jednu takovou hru si Iggy zkusil vytvořit. Vyzkoušej si také, jako to je být herním vývojářem a vytvoř svou zcela originální a super hru.

Co budeš v tomto projektu potřebovat?

- **dovednosti** získané při tvorbě předchozích her
- svojí **vlastní představivost** a **kreativitu**
- skvělý **nápad** na hru

Jak na to?

1. Zamysli se nad tím, jaký typ hry by měl být výsledkem tvé práce. Skákačka? Bludiště? Příběhová hra? Nebo třeba od každého něco?
2. Zkus si na papír načrtnout, jak budou vypadat jednotlivé úrovně tvé hry a jaké postavy v nich budou vystupovat.
3. Připrav si jednotlivá pozadí a postavy pro svou budoucí hru. Můžeš je vybrat z hotové nabídky, nebo si vytvořit vlastní.
4. Na chvíli se zamysli a zkus si vybavit různé scénáře a bloky, které už z předchozích projektů umíš použít a které by se ti mohly do hry hodit.
5. Začni vytvářet vlastní scénáře pro postavy ve tvé nové hře.

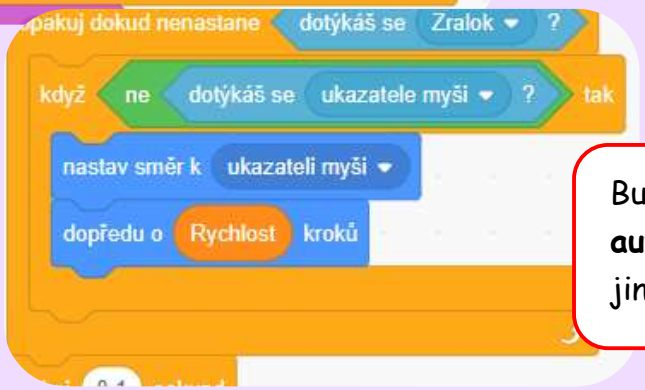
5. Nezapomeň čas od času otestovat, zda scénáře fungují a dělají přesně to, co chceš.
6. Pokud ti některý scénář nefunguje, nebo se ti nedaří vymyslet, jak danou věc naprogramovat, zkus si napsat, co chceš, aby postava dělala a podle toho zvol bloky, kterými toho docílíš.
7. Nezapomeň, že tvorba hry nekončí tím, že hra funguje. Prohlédni si celý kód znovu a zkus najít způsoby, jakými by se daly jednotlivé scénáře vylepšit či zjednodušit. Chceme přeci, aby hra byla co nejefektivnější.
8. Užij si programování i následné hraní vlastnoručně vytvořené hry!

Podívej se, jak vypadá Iggyho hotová hra: https://youtu.be/F3ffS9vzsnk

Můžeš svou hru postavit třeba na myšlence bludiště, kde hráč musí splnit různé úkoly, aby se z něj dostal ven!

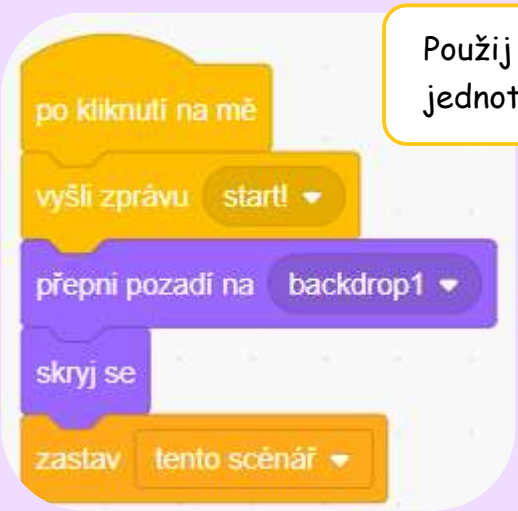
Budeš mít ve hře postavičky ovládané **šipkami na klávesnici?**

Nezapomeň všem postavám natavit úvodní pozici, kde budou vždy začínat.



Budou některé postavy **automaticky pronásledovat** jiné?

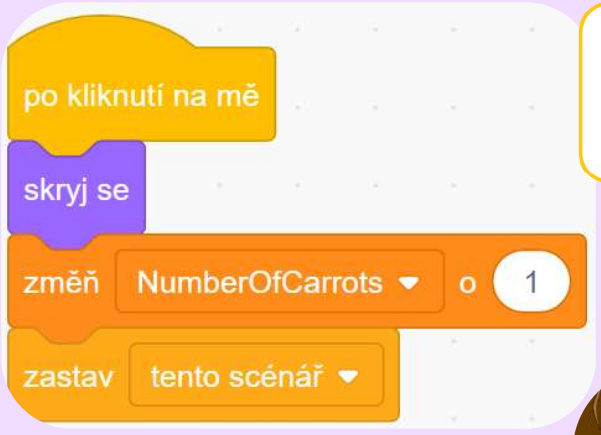
Použij **zprávy** pro komunikaci mezi jednotlivými postavami.



Použij **vlastní bloky** tam, kde by se ti jinak část kódu opakovala na více místech.

Dej do hry více pozadí a **více úrovní**. V každé úrovni může mít hráč třeba **jiný úkol**, nebo hrát jinou **minihru**.

Tady můžeš vidět dva typy úkolu, které Iggy využil ve své hře. V jedné úrovni čarodějnice chodí bludištěm, ze kterého se může dostat do králičí nory, kde musí sbírat mrkvičky.



Přidej do hry **body**, které hráč musí získat, nebo **věci,** které musí posbírat. Dáš tím hráči další cíl.

Přidej do hry postavy nebo pozadí s **instrukcemi**, jak hru hrát. Pomůžeš hráči se lépe orientovat, zejména pokud je hra složitější.

Přidej do hry kompletní možnost ovládání tlačítky uvnitř hry. Můžou to být například tlačítka pro instrukce k ovládání hry, spuštění hry, či návrat na úvodní obrazovku.

Přidej hráči životy a možnost udělat chybu. Životy můžeš znázornit třeba jako srdíčka v rohu obrazovky.

Pokud může hráč získat různý počet bodů během hraní hry, můžeš použít cloudové proměnné a ukládat nejvyšší skóre všech hráčů, kteří si tvoji hru zahráli. Hráči pak mezi sebou mohou soutěžit o co nejlepší výsledek.

Doplň hru o zvukové efekty nebo hudbu. Hra tím bude zase o něco zajímavější.