2/2022, Volume 14, Issue 2
ISSN 1803-537X

SITUATIONAL ALGORITHMIC TASKS WITH THE ASSEMBLING OF PROGRAM CODE AS A TOOL FOR DEVELOPING COMPUTATIONAL THINKING

Jiří Vaníček, Jihočeská univerzita v Českých Budějovicích, Česko
Václav Šimandl, Jihočeská univerzita v Českých Budějovicích, Česko
Václav Dobiáš*, Jihočeská univerzita v Českých Budějovicích, Česko

Přijato: 15. 7. 2022 / Akceptováno: 9. 12. 2022
Typ článku: Teoretická studie
DOI: 10.5507/jtie.2022.014

Abstract: This paper summarizes the experience with the development of situational algorithmic tasks, the solution of which is based on assembling a program from blocks. This paper deals with the didactic background of two created templates (worlds) in which a given problem is solved by programming a sprite. Based on these templates, sets of tasks, which are applicable in lessons as part of the new computer science curriculum, were created. Moreover, other tasks, which were included in the Bebras Challenge, were created. Being used in lessons, these tasks were perceived by teachers as useful, beneficial and they were found not being difficult for pupils.

Key words: computational thinking, algorithmization, block programming, elementary school, high school.

SITUAČNÍ ALGORITMICKÉ ÚLOHY SE SESTAVOVÁNÍM PROGRAMOVÉHO KÓDU JAKO NÁSTROJ ROZVÍJENÍ INFORMATICKÉHO MYŠLENÍ

Abstrakt: Článek shrnuje zkušenosti s vývojem situačních algoritmických úloh, jejichž řešení spočívá v sestavování programového kódu z bloků. Zabývá se didaktickým pozadím dvou vytvořených šablon mikrosvětů, v nichž je řešen problém daný popsanou situací naprogramováním postavy. Byly vytvořeny soutěžní úlohy soutěže Bobřík informatiky a sady úloh, které jsou použitelné jako součást kurikula nové informatiky.

*Autor pro korespondenci: dobias@pf.jcu.cz

Na základě jejich nasazení na školách bylo zjištěno, že úlohy nejsou pro žáky obtížné a jsou učiteli vnímány jako užitečné, přínosné.

Klíčová slova: informatické myšlení, algoritmizace, blokové programovaní, základní škola, střední škola.

1 Úvod

Oprostíme-li se od pohledu na programování jako profesní přípravu odborníka a budeme na něj spolu s Ganderem nahlížet jako na součást všeobecného vzdělání člověka 21. století (Gander, 2014, s. 7), kromě možnosti objevovat svět z dalšího úhlu pohledu a porozumět tomu, jak pracuje počítač, můžeme programování vnímat jako Papertův mikrosvět (Papert, 1980), který rozvíjí mentální schopnosti jedince. Od základního vymezení se odvíjí cíl, k němuž má výuka směřovat, a následně volba vhodného vzdělávacího obsahu, prostředí, motivace a metod výuky. Z výše popsaného pohledu v článku nahlížíme na programování jako na tréninkové hřiště pro rozvíjení schopností a kompetencí jedince. Stejný přístup nacházíme ve strategických dokumentech jako je studie Shut down or restart? ve Spojeném království (The Royal Society, 2012), celosvětové ACM computing curriculum (K-12 Computer Science Framework Steering Committee, 2016), CSTA K-12 computer science standards ve Spojených státech (CSTA, 2011), u blízkých sousedů Štátny vzdelávací program na Slovensku (Blaho, 2012) a Podstawa programowa z informatyki v Polsku (Sysło & Kwiatkowska, 2015).

Jestliže akceptujeme výše uvedený princip cíle vzdělávání jako rozvoje osobnosti, v oblasti informatiky se jím stává termín informatické myšlení (Wing, 2006) jako schopnost nalézt řešení problému, který by mohl automaticky realizovat agent zpracovávající informace. Algoritmizace jako nalezení postupu či cesty k cíli a jeho formalizace do takové podoby, aby jej takový agent zvládl přečíst a vykonat, je základní složkou informatického myšlení. Tento termín se stal stěžejním pro vymezení obsahu školního kurikula ve výše uvedených zemích i v ČR ve vládní Strategii digitálního vzdělávání (Ministerstvo školství, mládeže a tělovýchovy, 2014) a v návrhu nových rámcových vzdělávacích programů z informatiky (Ministerstvo školství, mládeže a tělovýchovy, 2021a, 2021b).

Podle Cunyho, Snydera a Wingové informatické myšlení pro každého znamená například pochopit, jaké aspekty problému lze vypočítat, vyhodnotit míru shody mezi výpočetními nástroji a problémem, použít nebo přizpůsobit výpočetní nástroj

novému použití a rozpoznat příležitosti využít výpočet novým způsobem (Wing, 2010, s. 3). Abychom tyto schopnosti mohli rozvíjet prostřednictvím programování, musíme hledat vhodné přístupy, prostředí, situace, úlohy, v nichž tyto cíle vystoupí do popředí, budou v nich zdůrazněny. Právě u výuky základů programování shledáváme jako velice efektivní využití didaktických prostředí. Podle Wittmanna je takovým prostředím soubor propojených situací, které poskytují problémy, které žákovi umožňují objevovat důležité myšlenky (Wittmann, 2001, s. 2).

Xia definuje učení se programování jako učební aktivity k získání užitečných programovacích znalostí a dovedností a výuku jako podporu pro porozumění žáka cestou praktických zkušeností (Xia, 2017). Mnoho přístupů k výuce programování preferuje žákovské aktivity, aktivní učení, učení se děláním a konstruování znalostí jako výsledek tvořivé práce, to vše s respektem k faktu, že znalosti nejsou přenositelné. Podle Piageta jsou znalosti aktivně konstruovány učícím se v interakci s okolním světem, takže jak navrhuje Ackermannová, stojí za to dětem nabídnout příležitosti k zapojení do praktických zkoumání, která podporují konstruktivní proces (Ackermann, 2010, s. 2). Ackermannová cituje Piageta, že „děti interpretují to, co slyší, ve světle svých vlastních znalostí a zkušeností" a jeho názor, že „znalosti jsou formovány a transformovány ve specifickém kontextu, tvarovány a vyjadřovány různými médii" (Ackermann, 2001, s. 3, 8). To, jak jedinec konstruuje znalost, je funkcí prvotní zkušenosti, mentálních struktur a svého přesvědčení, které používá k interpretaci fenoménů a událostí (Jonassen, 1991).

V učebnicích a manuálech pro výuku programování můžeme nalézt dva základní typy programovacích úloh:

- několikaminutové „etudy" (jako v Kalaš, 2017), vždy zaměřené na specifickou dovednost nebo programovací koncept a jejichž cílem je konkrétní znalost,
- větší „projekty" (jako v The LEAD Project, 2012), často formou vytváření příběhů nebo her, které jsou komplexnější a jejichž cílem je produkt.

Etudy jsou často cílené na konkrétní znalost či koncept a umožňují lépe detekovat žákovu chybu, vhodné řazení takových úloh usnadňuje vytváření mentálních modelů žáka. Projekty vyžadují kombinaci více dovedností současně, a to včetně plánování či tvořivosti; vytvořené delší programovací kódy vyžadují od žáka větší přehled, na druhou stranu lépe motivují žáka vidinou cílového produktu.

Jestliže žák řeší problémy tvorbou software, konkrétně sestavením programu, učitel může analýzou napsaného programu získat zpětnou vazbu, jak žák chápe situaci, v níž se vyskytuje problém, který řeší; jak rozumí konceptům, které používá;

2/2022, Volume 14, Issue 2
ISSN 1803-537X

jakou má úroveň složek informatického myšlení jako algoritmizace, dekompozice, generalizace; nebo jaké přístupy při řešení problémů využívá (Chao, 2016).

1.1 Podpora nové informatiky pomocí masové školní soutěže

Jednou z cest rozvíjení informatického myšlení je již po řadu let školní soutěž Bebras Challenge (Dagiene, 2008), probíhající ve více než 60 zemích světa (International Bebras Committee, 2022). V úlohách se při online testu žáci ocitnou v situaci, v níž mají najít příslušný informatický koncept a uplatněním informatického myšlení určit odpověď. Situační úlohy používané v české verzi soutěže Bobřík informatiky (https://www.ibobr.cz) jsou svou stavbou a zaměřením na konkrétní informatický koncept podobné etudám. Protože soutěž probíhá formou online testu, jsou odpovědi realizovány výběrem jedné z nabízených možností, označováním objektů na ploše nebo manipulací s objekty po ploše.

Bobřík informatiky se svojí dlouhou tradicí (v ČR od roku 2008) a velkým množstvím účastníků (v roce 2021 v ČR více než 109 000, celosvětově 2,7 milionu) měl a má velký dopad na formování představ žáků a učitelů, co je nová informatika. Bebras je také platformou, která celosvětově přináší množství nových informatických úloh a formuje tak školní kurikulum, např. vybrané soutěžní úlohy jsou součástí nových českých učebnic informatiky (Berki & Drábková, 2020a, 2020b).

Situační „bobří" úlohy rozvíjejí různé aspekty informatického myšlení a mezi nimi i algoritmizaci. Typickými algoritmickými úlohami, které se v soutěži objevují, jsou hledání počátečního či koncového stavu po vykonání daného algoritmu, porovnání více algoritmů oproti zadání úlohy, posouzení pravidel k provedení výpočtu, hledání chyby v algoritmu nebo jeho optimalizace. Mezi soutěžními úlohami doposud chyběly úlohy, na něž se odpovídá sestavením programu, což v soutěži nabídku informatických úloh ochuzovalo.

Hledali jsme řešení, které by umožnilo do stávajícího systému soutěže implementovat úlohy, v nichž by soutěžící stejně jako v programovacích prostředích používaných na školách mohli sestavovat svůj program z bloků. V současnosti jde o běžně používaný koncept blokového programování, využívaný v učebnicích informatiky nebo robotiky a tudíž žákům blízký. Naším cílem bylo vytvořit použitelné sady postupně gradovaných programovacích úloh, použitelných při výuce informatiky. Dalším cílem bylo inovovat soutěž Bobřík informatiky nově vyvinutým typem takovýchto úloh.

2 Použité metody

Metodologie řešení vychází z design-based research podle Trny (2011). Vývoj softwarového prostředí a samotných úloh probíhal iterativně, kdy jsme rámci každé iterace aplikovali metodu ADDIE. Každá vytvořená verze byla v rámci implementace otestována na žácích a zjištění z tohoto testování byla integrována formou zlepšení do další verze prostředí resp. úloh.

Ve fázi analýzy jsme analyzovali známá open source bloková programovací prostředí (např. Scratch, Blockly, Makecode, Go Vex) z pohledu využitelnosti jejich řešení pro vytvoření softwarového modulu, a to především z pohledu pedagogického a implementačního. Dále jsme analyzovali známé mikrosvěty pro tvorbu programovacích úloh (např. robot Karel, želví grafika, Baltík) z těchto hledisek:

- schopnost pokrýt rozsah kurikula (nízký práh vstupních vědomostí, vysoký strop vzdělávacích cílů),
- předpoklady pro tvorbu sady navazujících úloh s malými kroky vpřed co se týče získávaných vědomostí
- předpoklady pro tvorbu jednotlivé „soutěžní" úlohy s jejími specifiky.

Ve fázi návrhu byl navržen model softwarového modulu a jeho komunikace s testovou aplikací. Byl realizován návrh programovacích mikrosvětů a konkrétních úloh pro tyto mikrosvěty včetně zpětné vazby žákům a včetně grafiky. Také byl navržen způsob sběru dat automatické zpětné vazby a vytvořena kritéria pro hodnocení celého výstupu.

Ve fázi vývoje se programovaly jednotlivé funkcionality modulu včetně sběru dat během nasazení ve školách. Vytvořené mikrosvěty se integrovaly do systému soutěže Bobříka informatiky, dotvořil se konečný vzhled prostředí, proběhlo alfa testování v různých prohlížečích a operačních systémech. Pro hotové mikrosvěty byly vytvářeny jednotlivé soutěžní úlohy a sady navazujících úloh.

Fázi implementace jsme realizovali nasazením vyvíjeného modulu a sad navazujících úloh, které byly určeny pro použití mimo soutěž, na vybraných školách. Výsledky vzešlé z tohoto nasazení byly kvalitativně analyzovány. Tyto sady úloh byly následně dány k dispozici veřejnosti jako forma přípravy na samotnou soutěž. V listopadu 2021 jsme využili vyvíjený modul a vytvořené doposud nezveřejněné soutěžní úlohy ve školním kole soutěže Bobřík informatiky, na což navázalo další využití v ústředním kole soutěže v únoru 2022. Při nasazení při školním i ústředním kole soutěže byla sbírána kvantitativní data, poskytovaná softwaro-

vým modulem od všech žáků, díky čemuž jsme získali dostatečné množství dat pro celkovou evaluaci.

Evaluace byla prováděna ve všech fázích vývoje:

- Evaluace analýzy stanovila parametry návrhu a jeho úprav pro další fázi.
- Evaluaci ve fázi návrhu jsme provedli formou posouzení návrhů oponenty, experty na didaktiku informatiky a na tvorbu úloh do soutěže Bebras Challenge.
- Evaluaci ve fázi vývoje jsme realizovali pomocí testerů – zkušených pedagogů ze škol, kteří se podílejí na podobné kontrole úloh před soutěží po několik let a mají s touto činností zkušenosti.
- Evaluace kvalitativních dat během implementace ukázala v některých úlohách na dílčí nedostatky, které byly odstraněny v další iteraci vývoje.
- Evaluace kvantitativních dat během implementace vedla ke zjištěním, nakolik byl tento typ úloh složitější než ostatní typy úloh v testu (výběr z odpovědí, manipulace s objekty) a jak žáci hodnotí tento typ úloh v porovnání s ostatními typy úloh.

3 Průběh a výsledky řešení problému

Vytvořili jsme softwarové prostředí, které umožňuje vytvářet interaktivní situační úlohy z programování. Žák v něm řeší problém sestavením programu z bloků s možností svůj program testovat a ladit. Do našeho prostředí jsme implementovali upravený modul Blockly (https://developers.google.com/blockly).

Výhodou blokového zápisu programu je, že zamezuje chybám syntaxe. V naší implementaci má navíc omezenou sadu programovacích příkazů, která zabraňuje tomu, aby žáci namísto přemýšlení procházeli menu a hledali nějaký nástroj, který problém pohodlným způsobem vyřeší za ně.

Vytvořené prostředí umožňuje vytvářet jednotlivé úlohy pouze úpravou vstupních dat. Autoři úloh tak bez potřeby programovat mohou vytvářet nové úlohy nebo jejich sady, není potřeba, aby autor vedle didaktických dovedností potřeboval být i programátorem webových aplikací.

Při řešení úlohy žáci sestavovaný program průběžně opakovaně spouštějí, díky čemuž získávají od systému zpětnou vazbu. Ta se skládá z vizuální kontroly, jak se naprogramovaný objekt chová, a ze systémem generovaných hlášení, zda byly splněny všechny požadavky na řešení úlohy. Prostředí umožňuje vytvářet sady navazujících úloh, podobně jako v aktivitách Hour of Code (https://hourofcode.

2/2022, Volume 14, Issue 2
ISSN 1803-537X

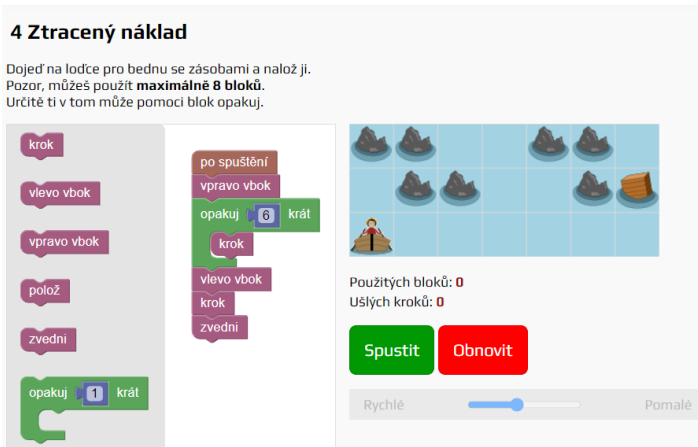
com), čímž vznikají úlohy stupňující obtížnosti s postupným zapojením komplexnějších konceptů a situací.

Pokaždé, když žák stiskem tlačítkem Spustit požádá o vykonání programu, systém zároveň uloží žákův program a též informace o tom, zda řešení vyhovělo všem požadavkům zadání a na kolikátý pokus žák vytvořil správný program.

K tomuto softwarovému prostředí jsme vyvinuli dvě šablony programovacích úloh, tzv. mikrosvěty, v nichž se programem ovládané postavy chovají odlišně a existují v nich odlišné sady základních příkazů. Ke každé šabloně bylo možno vytvářet sady konkrétních úloh, navzájem odlišných tématem, námětem i grafikou.

3.1 Mikrosvět robota Karla

První šablona simulovala mikrosvět robota Karla (Pattis, 1981), postavy, která chodí po čtvercové síti a přitom sbírá a pokládá předměty. Základní sada příkazů umí přesouvat programovanou postavu na sousední pole hrací plochy, otáčet jí oběma směry o pravý úhel, detekovat objekty v poli, na němž se postava nachází, odebrat takový objekt z pole nebo na prázdné pole objekt položit. Dále umí detekovat překážku na sousedním poli ve směru, do něhož je postava natočena. K základním příkazům jazyka byla přidána struktura Opakuj, představující cyklus s pevným počtem opakování (viz Obr. č. 1).



Obr. č. 1: Úloha z prostředí mikrosvěta robota Karla. Vlevo je menu s dostupnými příkazy, vpravo herní plocha a uprostřed žákem vytvořený program jako řešení úlohy.

2/2022, Volume 14, Issue 2
ISSN 1803-537X

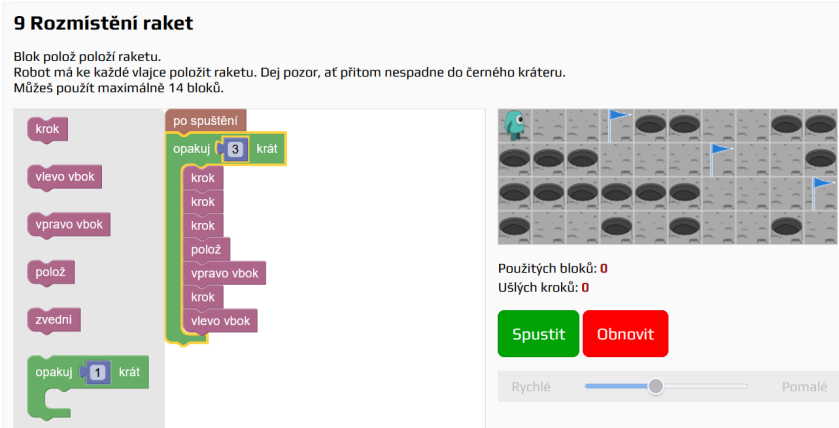
Mikrosvět robota Karla jsme vybrali proto, že je pro žáka dostatečně jednoduchý pro pochopení a zvládnutí základů jazyka tak, aby bylo možné rychle přejít k složitějším úlohám. Mezi výhody tohoto mikrosvěta patří možnost vytvářet reálné situace, vizuální srozumitelnost stavu postavy, situované do jednoho ze čtyř hlavních směrů a tudíž nevyžadujícího parametr velikosti úhlu otočení, a také omezený počet základních příkazů jazyka (krok vpřed, otočit, zvednout, položit). Další výhodou je absence náročných pojmů jako objekt, souřadnice, procedura nebo proměnná. To v důsledku zkracuje dobu potřebnou pro seznámení s prostředím a umožňuje dříve a intenzivněji se soustředit na algoritmické jádro řešených problémů.

Typickými úlohami v tomto prostředí byly dojít na dané místo, vyhnout se překážce, sesbírat pravidelně rozmístěné objekty nebo najít cestu, skládající se ze stejných opakovaných částí.

V tomto mikrosvětě jsme vytvořili sadu programovacích úloh s postupně rostoucí obtížností. Z pohledu programovacích konceptů lze jednotlivé úlohy představit takto:

1. sestavení programu, orientace v prostředí
2. sestavení delšího programu
3. sestavení programu s možností použít cyklus
4. nutnost použití cyklu s jedním blokem v těle cyklu
5. prosté opakování dvou bloků v cyklu (přidáno od druhé iterace výzkumu)
6. opakování více bloků s dalšími bloky před a za blokem Opakuj
7. opakování více bloků se skrytým opakujícím se vzorem
8. opakování více bloků jako součást většího programu
9. opakování více bloků se složitým opakujícím se vzorem
10. rozdělení problému na části s vícenásobným použitím cyklu
11. nalezení cesty popsatelné co nejkratším zápisem programového kódu

Počínaje 4. úlohou bylo omezen počet použitelných bloků v programu, což nutilo žáky zkracovat kód a používat cyklus (viz Obr. č. 2).



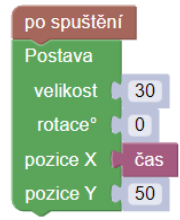
Obr. č. 2: Náročnější úloha v prostředí robota Karla s více bloky v těle cyklu.

3.2 Mikrosvět Film

Druhou vyvinutou šablonou byl mikrosvět „Film". Pro tuto šablonu jsme se inspirovali v prostředí Blockly games (https://blockly.games). V tomto mikrosvětě je naprogramován objekt – postava, který v čase mění svoji polohu a velikost. Objekt má čtyři parametry: pozice x, pozice y, velikost a otočení (oproti základnímu směru). Programovací jazyk má jeden základní příkaz Postava, který vykreslí objekt na hrací plochu na místě daném parametry pozice x, y, s velikostí danou parametrem (hodnota 100 odpovídá velikosti přes celou hrací plochu) a otočením o daný počet stupňů. Tento příkaz byl doplněn blokem pro vytváření matematických výrazů se základními početními úkony a strukturou Když, kontrolující podmínku času (např. zda čas překročil určitou hodnotu).

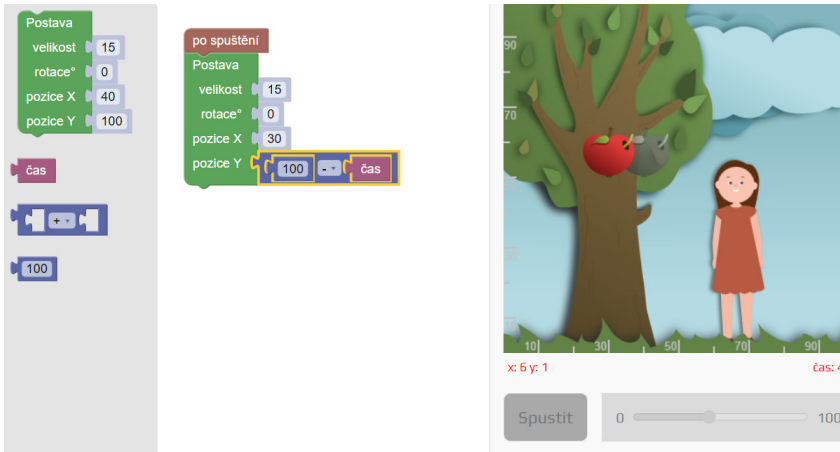
Animace je realizována tak, že po spuštění programu se hodnota proměnné čas plynule mění od 0 do 100 a pro každou z těchto hodnot se provede příkaz Postava. Proměnnou lze v příkazu použít jako parametr, tedy pokud např. použijeme proměnnou čas jako parametr pozice x (Obr. č. 3), bude se souřadnice x plynule měnit od 0 do 100 a objekt se bude rovnoměrně pohybovat zleva doprava. Hrací plocha má v obou směrech rozměr 100, tak aby postava během své animace přešla přes celou plochu.

2/2022, Volume 14, Issue 2
ISSN 1803-537X



Obr. č. 3: Použití proměnné čas jako parametru polohy objektu v programu.

Zadání programovací úlohy v tomto mikrosvětě spočívá v tom, že po spuštění animaci se zobrazí stín postavy a úkolem hráče je vytvořit program (tedy sestavit parametry bloku Postava) tak, aby se jeho objekt choval stejně jako její stín, aby se oba objekty po celou dobu animace překrývaly (na rozdíl od situace na Obr. č. 4, kde jsou objekty vedle sebe). Hráč má možnost kromě opakovaného spouštění programu, při němž proběhne animace v čase od 0 do 100, také ručně pomocí posuvníku nastavit libovolnou z hodnot proměnné čas a analyzovat situaci v daném čase (viz Obr. č. 5).



Obr. č. 4: Prostředí Film s úlohou na animaci pádu jablka ze stromu (na obrázku je situace v čase 43). Uprostřed je řešení úlohy s použitím proměnné čas ve výrazu a chybnou hodnotou parametru pozice x, kvůli níž se jablko a jeho stín nepřekrývají.

2/2022, Volume 14, Issue 2
ISSN 1803-537X

Koncepce prostředí Film se opírá o koncept parametr, pracuje s proměnnou a především s výrazy. Způsob programování v tomto prostředí je blízký funkcionálnímu programování, a protože obsahuje náročnější koncepty než prostředí robota Karla, je určeno spíše žákům středních škol a posledních ročníků základní školy.

Obr. č. 5: Rozfázovaná animace zadání úlohy o průletu komety v časech 0, 25, 50, 75, 100 se stínem prolétávající komety.

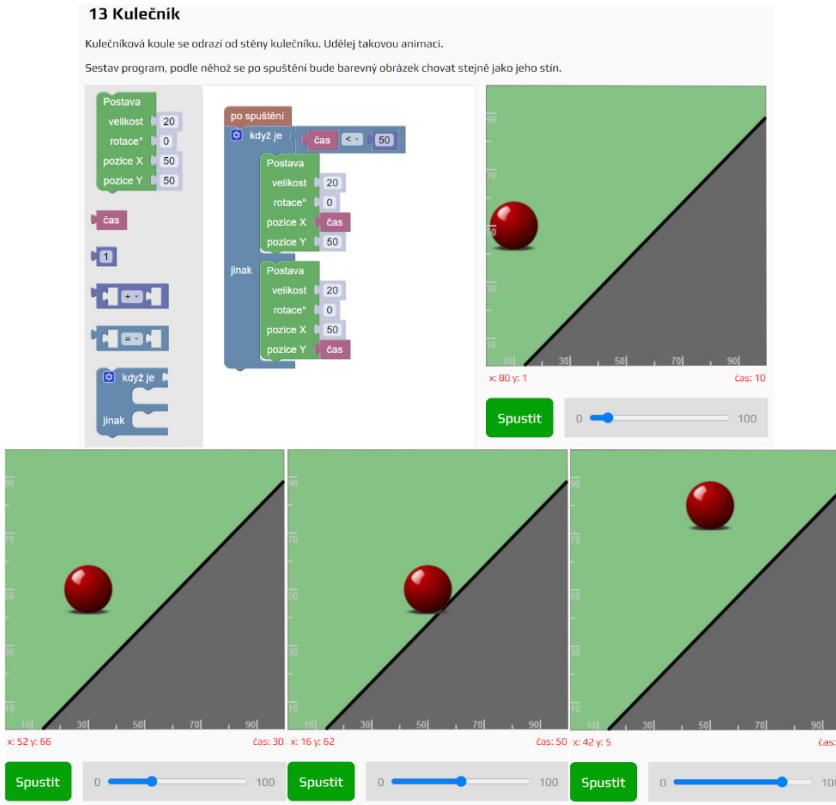
Typickými úlohami v tomto prostředí byly pohybovat se rovnoměrným pohybem vodorovně a svisle (pohyb zprava doleva je náročnější než pohyb zleva doprava, protože se souřadnice oproti času zmenšuje), pohybovat se pomaleji a rychleji, kombinovat pohyb těmito směry, zvětšovat nebo zmenšovat postavu v čase, kombinovat zvětšování s pohybem postavy.

V tomto mikrosvětě jsme vytvořili sadu programovacích úloh s postupně rostoucí obtížností. Nejtěžší úlohy kombinovaly více pohybů v čase (např. pohyb tam a zpátky během jedné animace) s využitím rozhodování (viz Obr. č. 6). Z pohledu programovacích konceptů lze jednotlivé úlohy představit takto:

- umístění objektu na souřadnice
- vodorovná souřadnice zvětšující se v čase jako parametr
- svislá souřadnice zvětšující se v čase jako parametr
- souřadnice zmenšující se v čase jako parametr
- pomalejší pohyb v čase jako parametr
- šikmý pohyb jako kombinace parametrů
- zvětšování postavy jako parametr
- zmenšování postavy jako parametr
- kombinace pohybu a zvětšování postavy
- změna souřadnic v určitém časovém okamžiku
- pohyb po lomené čáře, tedy dva různé pohyby navazující na sebe v čase
- pohyb jen po určitou část doby animace

2/2022, Volume 14, Issue 2
ISSN 1803-537X

Parametr otočení jsme u mikrosvěta Film nakonec v úlohách nevyužili, protože otáčení o úhlové stupně by otočilo postavu za dobu animace o 100°, což vizuálně není optimální. Žáci dobře rozeznají otočení o 90°, ovšem aby se postava během času 100 otočila o 90°, museli bychom ji otáčet nikoliv v jednotkách úhlový stupeň, ale v gradech, tedy v jednotkách, se kterými žáci ve škole nepracují.



Obr. č. 6: Náročná úloha s odrazem koule od stěny kulečníkového stolu vyžadující použití větvení programu v závislosti na parametru čas. V horní části obrázku správné řešení v čase 10. V dolní části obrázku jsou rozfázované animace v časech 30, 50, 80.

4 Ověřování

Vytvořené prostředí pro sestavování programů z bloků bylo jako modul implementováno do testů soutěže Bobřík informatiky. Vytvořené úlohy jsme použili následujícím způsobem:

Ze sady úloh typu mikrosvět robota Karla jsme vytvořili speciální nesoutěžní test Bloky (Bobřík informatiky, 2022a), který jsme v několika iteracích otestovali na vybraných školách a následně jej nabídli též ostatním školám jako přípravu na loňské národní kolo soutěže. Během měsíců září a října tento test o 11 otázkách absolvovalo 45 000 žáků ze základních a středních škol. Ze zpětné vazby od škol a z odborného posudku recenzenta jsme čerpali poznatky pro průběžnou úpravu prostředí a úloh. Nejčastějšími problémy bylo, že se v některých úlohách v některých prohlížečích grafika nezobrazovala správně. Problém obtížného přechodu mezi 4. a 6. úlohou, způsobený příliš velkým kognitivním skokem, jsme vyřešili vložením další úlohy č. 5 a přidáním vysvětlujících prvků do zadání úloh.

Dalším kolem ověření bylo národní kolo soutěže Bobřík informatiky, v němž každý ze 109 442 soutěžících absolvoval tři úlohy z mikrosvěta robota Karla z celkových 12 soutěžních úloh. Při tomto kole jsme na nových úlohách zjišťovali, nakolik jsou obtížnější než ostatní soutěžní úlohy. Kromě toho jsme pomocí dotazníkového šetření zjišťovali hodnocení jednotlivých úloh samotnými soutěžícími, na základě čehož jsme porovnávali žákovské hodnocení úloh mikrosvěta robota Karla s hodnocením ostatních úloh. Výsledky těchto analýz jsou uvedeny v kapitole Výsledky.

Ověřování mikrosvěta Film proběhlo také ve dvou kolech, ovšem již při menším počtu soutěžících. Z výše uvedené sady 12 úloh tohoto typu byl vytvořen nesoutěžní test Film, který sloužil v lednu a únoru 2022 jako tréninkový soutěžícím, kteří postoupili do ústředního kola v nejstarší věkové kategorii Senior. Tento test si vyzkoušelo celkem 546 žáků. Na základě ověřování jsme museli upravit způsob vyhodnocování dat od uživatele během řešení úlohy, protože výpočty parametrů s použitím násobení a dělení vykazovaly odchylky (např. násobení 0.001 a dělení 1000 nedávalo stejný výsledek). V několika případech se tak stalo, že žákovo řešení bylo systémem vyhodnoceno jako chybné, ačkoliv bylo správné.

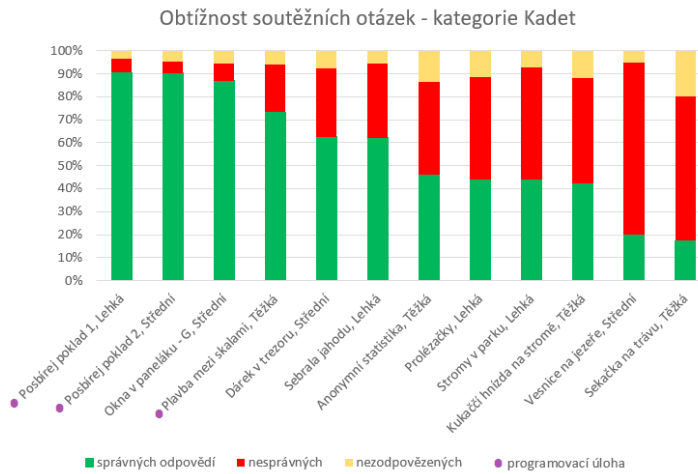
Druhé kolo ověřování se uskutečnilo při samotném ústředním kole, v němž soutěžilo 358 soutěžících. V testu byly použity tři úlohy z 15 z mikrosvěta Film a další tři úlohy z mikrosvěta robota Karla – dohromady 40 % soutěžních úloh bylo programovacích se skládáním programového kódu z bloků. Ověření prokázalo,

že tyto úlohy mohou být použity i při tak náročných akcích jako ústřední kolo celostátní soutěže, protože se neobjevil žádný protest soutěžících, že by v zadáních nebo mechanismech úloh byly chyby.

Nesoutěžní test Film byl během dubna až června 2022 volně zpřístupněn (Bobřík informatiky, 2022b) a v té době si jej vyzkoušelo 10 500 respondentů. Zpětná vazba z tohoto nasazení doplnila zjištění z předchozích kol ověřování.

5 Výsledky

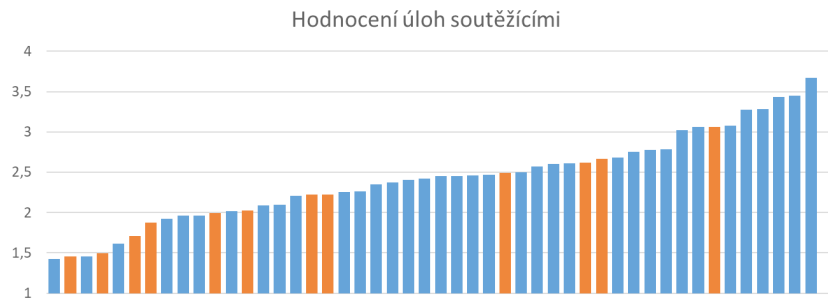
Při porovnání obtížnosti programovacích úloh se sestavováním bloků oproti ostatním soutěžním úlohám v uplynulém ročníku národního kola soutěže bylo zjištěno, že s jedinou výjimkou se všechny tyto úlohy ve všech věkových kategoriích umístily v první polovině podle podílu správných odpovědí. Tyto úlohy lze proto považovat za spíše lehčí. Příčinou může být, že tyto úlohy poskytují zpětnou vazbu a žáci tak mohou odpovídat vícekrát a svoji odpověď iterovat na rozdíl od běžných soutěžních úloh, v nichž označují objekty nebo odpovědi z výběru a zpětnou vazbu nemají. Na Obr. č. 7, ukazujícím výsledky věkové kategorie 14–15 let, jsou programovací úlohy se sestavováním bloků označeny puntíkem u svého názvu. Je patrné, že při porovnání všech úloh jde o lehčí úlohy.



Obr. č. 7: Porovnání obtížnosti soutěžních úloh národního kola 2021 ve věkové kategorii 14–15 let; úlohy se sestavováním programového kódu jsou u názvu označeny fialovým puntíkem.

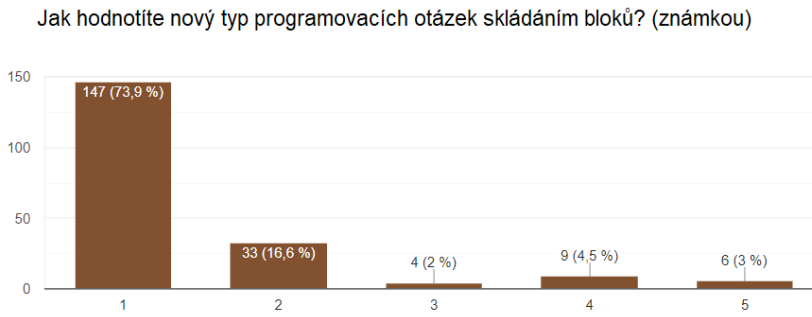
2/2022, Volume 14, Issue 2
ISSN 1803-537X

Abychom zjistili názor žáků na jednotlivé úlohy, dali jsme soutěžícím čtyř kategorií národního kola určených pro 2. stupeň základních škol a střední školy možnost úlohy ohodnotit. Když soutěžící dokončil soutěžní test, zobrazil se mu odkaz na vyplnění anonymního dotazníku. V něm každou z 12 úloh v testu ohodnotil (oznámkoval) na stupnici 1 („skvělá") až 5 („strašná") a také vybral nejlepší úlohu ze všech. Pokud si na některou úlohu nedokázal vzpomenout, mohl vybrat možnost „Nepamatuji se". Ze získaných dat jsme odstranili záznamy, ve kterých respondent nebyl schopen ohodnotit více než třetinu úloh, a záznamy, ve kterých úloha vybraná jako nejlepší byla hodnocena horší známkou než jiná úloha. K další analýze byla využita data získaná od 8558 soutěžících, což představuje 10 % soutěžících v těchto kategoriích. U jednotlivých úloh jsme následně stanovili průměrné hodnocení. Při porovnání programovacích úloh se všemi soutěžními úlohami v tomto ročníku národního kola soutěže jsme zjistili, že většinu úloh žáci svým hodnocením umístili do první poloviny. To je znázorněno na Obr. č. 8; programovací úlohy se sestavením bloků jsou označeny oranžovou barvou.



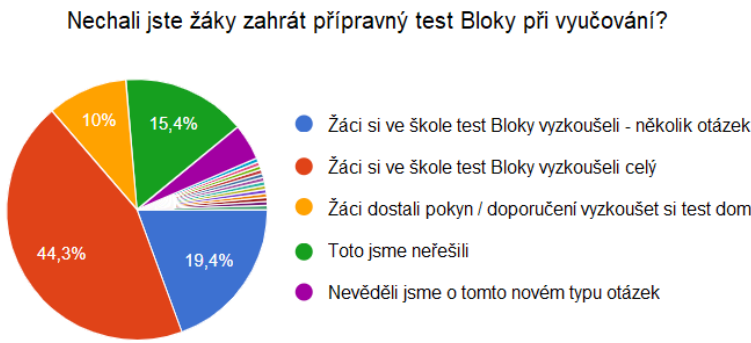
Obr. č. 8: Hodnocení soutěžních úloh národního kola 2021 soutěžícími. Nižší hodnota znamená lepší hodnocení. Úlohy se sestavováním programového kódu jsou označeny oranžovou barvou.

V listopadu 2021 jsme požádali 939 školních koordinátorů, kteří organizovali aktuální soutěž, o vyplnění dotazníku. Vrátilo se 201 odpovědí, což je více než 21% návratnost. Jednou z otázek bylo, jak učitelé hodnotili nový typ úloh, založených na sestavování programu z bloků. Téměř tři čtvrtiny respondentů hodnotily tento typ úloh školní známkou „výborně"; podrobné výsledky jsou znázorněny na Obr. č. 9.



Obr. č. 9: Výsledky anketní otázky o kvalitě úloh s blokovým programováním mezi učiteli informatiky.

S tím související otázkou bylo, nakolik učitelé nechali žáky pracovat se sadou úloh mikrosvěta robota Karla přímo při vyučování. Z grafu na Obr. č. 10 vyplývá, že téměř dvě třetiny škol nechaly své žáky pracovat s touto sadou úloh ve škole, tedy že školy berou tento typ úloh vážně, jako vhodný prvek kurikula pro rozvíjení informatických dovedností žáků.



Obr. č. 10: Výsledky anketní otázky mezi učiteli, zda a jakou formou nechali žáky pracovat se sadou úloh mikrosvěta robota Karla (drobné výseče představují vlastní vyjádření respondentů).

6 Závěr

Vytvořili jsme softwarové prostředí, které umožňuje vytvářet interaktivní situační úlohy z programování. Žáci zde řeší úlohy pomocí sestavování programového kódu v blokovém programovacím jazyce. V námi vytvořeném prostředí jsme vytvořily sady situačních úloh, které jsme dali veřejně k dispozici.

Tento typ úloh jsme dále implementovali do české verze informatické soutěže Bobřík informatiky nový typ informatických úloh. Takový typ úloh doposud v soutěži chyběl. Úlohy tohoto typu jsou vzhledem k potenciálu blokových prostředí přínosné z hlediska motivace i z pedagogického hlediska. Jejich význam lze spatřovat nejen pro soutěž Bobřík informatiky jako takovou, ale i pro zavádění nového pojetí a obsahu předmětu informatika na základní škole i na gymnáziích podle nové revize RVP ZV a RVP G. Vytvořený nástroj umožňuje vyvíjet a testovat sady úloh, zaměřené na jeden koncept nebo jednu programátorskou dovednost, s možností implementovat je do školního kurikula. Do budoucna je možno do vytvořeného modulu přidávat další mikrosvěty, např. želví grafiku.

Z provedeného šetření při nasazení tohoto typu úloh do národního kola soutěže se ukázalo, že úlohy s aktivním programováním jsou žáky hodnoceny spíše pozitivně, učiteli jsou vnímány jako užitečné a přínosné a současně nejsou pro žáky obtížné, jak prokázalo jejich srovnání s ostatními soutěžními úlohami. Domníváme se, že kromě interaktivity, jejíž pozitivní vliv na snižování obtížnosti úlohy je v souladu se zjištěními ve (Vaníček, 2016), je popularita těchto úloh způsobena okamžitou zpětnou vazbou spojenou s možností opravy, což jsou vlastnosti, kterými běžné úlohy soutěže Bobřík informatiky nedisponují.

7 Poděkování

Tento výzkum byl podpořen z projektu TAČR TL03000222 „Rozvoj informatického myšlení pomocí situačních algoritmických problémů".

8 Literatura

Ackermann, E. (2001). *Piaget's constructivism, Papert's constructionism: What's the difference?* http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf

Ackermann E. (2010). Constructivism(s): Shared roots, crossed paths, multiple legacies. In J.E. Clayson, & I. Kalaš (Eds.), *Constructionism 2010: Constructionist approaches to creative learning, thinking and education: lessons for the 21st century: proceedings for Constructionism*

2010. Comenius University. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.6201&rep=rep1&type=pdf

Berki, J., & Drábková, J. (2020a). *Základy informatiky pro 1. stupeň ZŠ. Učebnice.* Technická univerzita v Liberci.

Berki, J., & Drábková, J. (2020b). *Základy informatiky pro 2. stupeň ZŠ. Učebnice.* Technická univerzita v Liberci.

Blaho, A. (2012). Informatika v štátnom vzdelávacom programe. In I. Kalaš (Ed.), *DidInfo 2012* (pp. 7–14). Univerzita Mateja Bela v Banskej Bystrici. http://www.didinfo.net/images/DidInfo/files/didinfo_2012.pdf

Bobřík informatiky (2022a). *Bloky: Sada edukačních programovacích úloh*. Jihočeská univerzita v Českých Budějovicích. https://www.ibobr.cz/test/archiv-pred-spustenim/2021/496

Bobřík informatiky (2022b). *Film: Sada edukačních programovacích úloh.* Jihočeská univerzita v Českých Budějovicích. https://www.ibobr.cz/test/archiv-pred-spustenim/2021/494

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, *95*, 202–215. https://doi.org/10.1016/j.compedu.2016.01.010

CSTA. (2011). *K-12 Computer Science Standards*.

Dagienė, V. (2008). The Bebras Contest on Informatics and Computer Literacy – Students Drive to Science Education. In *Joint Open and Working IFIP Conference, ICT and Learning for the Net Generation* (pp. 214–223). https://www.bebras.org/sites/default/files/documents/publications/DagieneV-2008.pdf

Gander, W. (2014). Informatics and General Education. In Y. Gülbahar, & E. Karataş (Eds.), *Lecture Notes in Computer Science: Vol. 8730. Informatics in Schools. Teaching and Learning Perspectives. ISSEP 2014* (pp. 1–7). Springer. https://doi.org/10.1007/978-3-319-09958-3_1

International Bebras Committee. (2022). *Bebras: Countries.* https://www.bebras.org/countries.html

Jonassen, D.H. (1991). Objectivism versus constructivism: Do we need a new philosophical paradigm? *Educational Technology Research and Development, 39*, 5–14. https://doi.org/10.1007/BF02296434

K-12 Computer Science Framework Steering Committee. (2016). *K-12 Computer Science Framework*. ACM. https://dl.acm.org/doi/book/10.1145/3079760

Kalaš, I. (2017). *UCL Scratchmaths curriculum*. University College London. http://www.ucl.ac.uk/ioe/research/projects/scratchmaths/curriculum-materials

Ministerstvo školství, mládeže a tělovýchovy. (2014). *Strategie digitálního vzdělávání*. https://www.msmt.cz/uploads/DigiStrategie.pdf

Ministerstvo školství, mládeže a tělovýchovy. (2021a). *Rámcový vzdělávací program pro základní vzdělávání.* https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcovy-vzdelavacici-program-pro-zakladni-vzdelavani-rvp-zv/

Ministerstvo školství, mládeže a tělovýchovy. (2021b). *Rámcový vzdělávací program pro gymnázia.* https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcove-vzdelavaci-programy-pro-gymnazia-rvp-g/#2-aktualizace-rvp-pro-gymnázia-s-účinností-od-1-září-2022

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas.* Basic Books, New York, NY, USA.

Pattis, R.E. (1981). *Karel the Robot: Gentle Introduction to the Art of Programming with Pascal.* John Wiley & Sons.

Sysło, M.M., & Kwiatkowska, A.B. (2015). Introducing a New Computer Science Curriculum for All School Levels in Poland. In A. Brodnik, & J. Vahrenhold (Eds.), *Lecture Notes in Computer Science: Vol. 9378. Informatics in Schools. Curricula, Competences, and Competitions. ISSEP 2015* (pp. 141–154). Springer. https://doi.org/10.1007/978-3-319-25396-1_13

The LEAD Project (2012). *Easy LEAD: Super Scratch Programming Adventure!* No Starch Press.
The Royal Society. (2012). *Shut Down or Restart? The Way Forward for Computing in UK Schools*. https://royalsociety.org/~/media/royal_society_content/education/policy/computing-in-schools/2012-01-12-computing-in-schools.pdf
Trna, J. (2011). Konstrukční výzkum (design-based research) v přírodovědných didaktikách. *Scientia in educatione*, *2*(1), 3–14. https://ojs.cuni.cz/scied/article/view/11/12
Vaníček, J. (2016). What Makes Situational Informatics Tasks Difficult? In A. Brodnik, & F. Tort (Eds.), *Lecture Notes in Computer Science: Vol. 9973. Informatics in Schools: Improvement of Informatics Knowledge and Perception. ISSEP 2016* (pp. 90–101). Springer. https://doi.org/10.1007/978-3-319-46747-4_8
Wing, J.M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215
Wing, J.M. (2010). *Research notebook: Computational thinking: What and why?* Carnegie Mellon University. https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf
Wittmann, E. Ch. (2001). Developing mathematics education in a systemic process. *Educational Studies in Mathematics*, *48*(1), 1–20. https://www.jstor.org/stable/3483113
Xia, B.S. (2017). A Pedagogical Review of Programming Education Research: What Have We Learned. I*nternational Journal of Online Pedagogy and Course Design*, *7*(1), 33–42. https://doi.org/10.4018/IJOPCD.2017010103