Jan Hamáček, 23. září 2015

Seminární práce

Využití dětských programovacích jazyků v hodinách programování

1 Úvod

V této práci se budu zabývat metodami výuky programování v dětských programovacích jazycích. Věnovat se budu zejména jazykům Scratch a Baltík 3. Důvodem pro to je fakt, že už několik let vedu zájmové kroužky programování právě v těchto dvou jazycích. Kroužky jsou určeny pro žáky třetích až šestých tříd základní školy.

Kroužek Baltík jsem vedl přibližně 6 let. Minulý rok jsem ukončil kroužek Baltík a místo něj učím už druhým rokem v jazyce Scratch.

V úvodu práce oba jazyky krátce představím, potom ukážu typické úlohy řešitelné pomocí obou jazyků a zkušenosti s tím, jak děti úlohy řeší. Kromě kroužků v dětských programovacích jazycích vedu ještě kroužek „dospělého“ programování v Javě. Některé děti po absolvování kroužků v dětském programování přestoupily na dospělé programování. V závěru práce proto zmíním znalosti, které si děti přenesly z dětského programování.

2 Představení jazyků

V této kapitole ukážu základní principy ovládání obou jazyků. Programování v obou jazycích probíhá pokládáním grafických symbolů, které znázorňují příkazy, na plochu programu. Skládáním několika příkazů za sebe vzniká smysluplný program.

2.1 Baltík

Na obrázku 1 je vidět rozhraní baltíka. V horní části jsou s oranžovým pozadím seřazeny dostupné příkazy. Kostičky příkazů se myší přetahují do spodní části. Na obrázku 1 je už ve spodní části naprogramovaná část programu. Program v Baltíku je posloupnost příkazů pro čaroděje Baltíka.

Obrázek 1: Rozhraní Baltíka

Po spuštění programu se otevře nové okno, ve kterém postavička čaroděje Baltíka vykoná postupně všechny příkazy. Celá situace je znázorněna na obrázku 2.

Program Baltík je dostupný ke stažení na webových stránkách `www.sgp.cz`. Demoverzi programu je možné používat zdarma. K plnohodnotnému používání je ale potřeba koupit si licenci.

2.2 Scratch

Na obrázku 3 vidíme náhled do hlavního okna rozhraní jazyka Scratch. V levé části je reprezentace Scratche (kočka). V této části se budou vykonávat naprogramované programy. Uprostřed je seznam všech příkazů. Oproti Baltíkovi nejsou příkazy obrázkové, ale každý příkaz obsahuje popisek se svojí funkcí. V pravé části je pole pro programování. Na obrázku 3 je v této části už naprogramovaný program.

Scratch je možné používat zcela zdarma bez potřeby instalace dalšího softwaru přímo ve webovém prohlížeči na stránkách `scratch.mit.edu`. Lze ho používat i bez předchozí registrace. Proto doporučuji alespoň krátké vyzkoušení.

3 Cíle výuky dětských programovacích jazyků

Prvním cílem výuky dětských programovacích jazyků je vytvoření základních programátorských návyků a připravit tak děti na případný přechod k dospělému programovacímu jazyku.

Oba zmíněné dětské programovací jazyky jsou k tomuto účelu dobře přizpůsobeny. Kromě základních příkazů k ovládání postavičky Scratche nebo Baltíka obsahují i příkazy pro podmínky a cykly. Ty tvoří základ jakéhokoliv programování.

Dalším cílem je vytvoření pozitivního vztahu k programování. Při první výuce dospělého programovacího jazyka je potřeba překonat mnoho překážek. Jednou z těch závažnějších je potřeba zapamatovat si mnoho různých příkazů a posloupnosti, ve kterých příkazy dávají smysl. Tato překážka u dětských programovacích jazyků (téměř) odpadá. Příkazy jsou reprezentovány návodnými obrázky, takže není potřeba si je pamatovat. Pro děti je zároveň problém psaní textu na klávesnici. Proto dětské programovací jazyky nevyžadují téměř žádné psaní. Programování probíhá přesouváním příkazů.

K vytvoření pozitivního vztahu je důležité nejen odstranění překážek běžného programování, ale také atraktivita výsledku. V obou dvou zmíněných dětských programovacích jazycích je proto typický výsledek programu pestře barevný a složený z obrázků.

Děti na prvním stupni ZŠ se ještě nikdy s programováním nesetkaly. Jedním z prvních úkolů při výuce dětského programování je proto pochopení algoritmu (pracovního postupu pro postavičku Baltíka nebo Scratche).

4 Typické úlohy

Oba zmíněné programovací jazyky svým ovládáním zjednodušují přístupnost programování pro děti. Každý z nich to ale dělá trochu jiným způsobem. Popíšu zde přístup obou dvou jazyků.

4.1 Baltík

V mnou vedených zájmových kroužcích byly nejmladší děti při prvním setkání s Baltíkem ve třetí třídě základní školy. Kromě režimu programování Baltík poskytuje ještě režimy skládání scény a čarování. Zpočátku je významný druhý režim čarování.

V tomto režimu je před dětmi okno s čarodějem Baltíkem. Děti mají možnost klikat na příkazy a Baltík je okamžitě vykonává. Po krátkém seznámení s tímto režimem mohou žáci plynule přejít na programování.

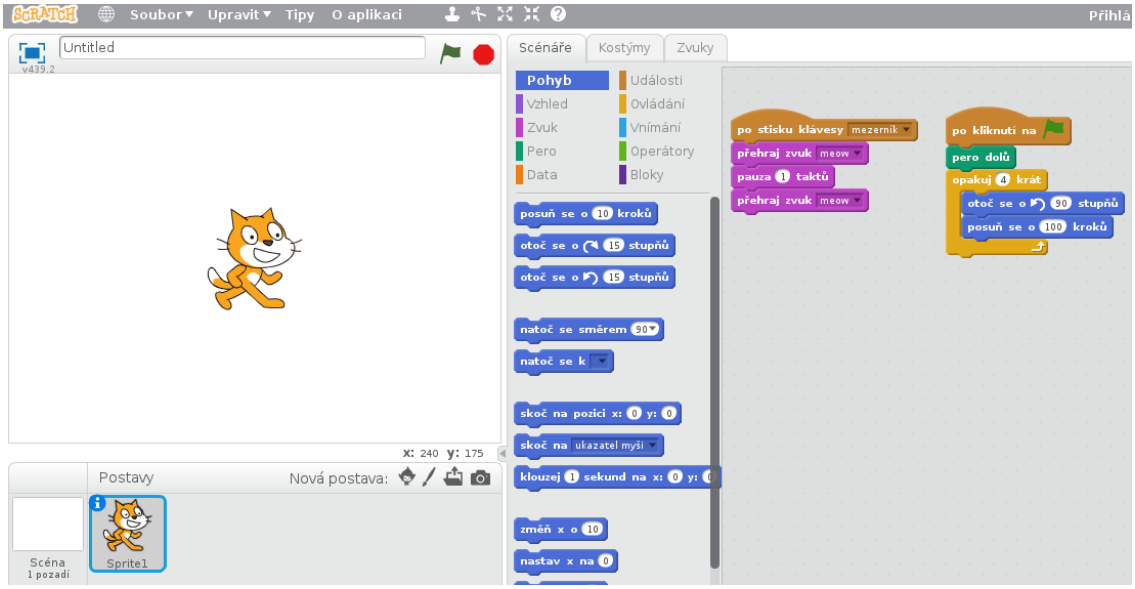
Příklad úkolu který tento přechod usnadňuje je dojít s Baltíkem na střed obrazovky. Rozšiřujícím úkolem k tomuto je spočítat počet kroků, které je potřeba udělat aby se baltík dostal na střed. Pomocí získaných zkušeností už i třeťáci dokážou naprogramovat program, který přesune Baltíka na střed obrazovky. Využijí k tomu příkazy, se kterými se seznámili v režimu čarování tj. příkaz `popojdi`, `otoč se vlevo`, případně i `otoč se vpravo`.

Používání samotných příkazů k pohybu by děti brzy omrzelo. Proto je v Baltíkovi vytvořená sada obrázků, které slouží jako příkaz `vyčaruj`. V programu na obrázku 1 jsou to například otevřené dveře, nebo obrázek opice. Uvádím několik příkladů, které jsou vhodné pro začátky práce s Baltíkem.

1. Vyčarujte domeček.

2. Baltík začíná na předem vytvořeném bludišti. Vytvořte program takový, aby baltík prošel bludištěm a nenarazil přitom do zdi.

Obrázek 2: Běžící program Baltíka



Obrázek 3: Rozhraní Scratche

3. Vytvořte program takový, že Baltík vstoupí do vyčarovaného domečku, na chvíli se objeví v okně, a potom zase vystoupí. Zajistěte aby Baltík dveře otvíral a zavíral plynule.

První dva příklady kladou důraz zejména na schopnost přesně popsat algoritmus, který daný problém řeší. Algoritmus přitom není složitý na vymyšlení. Jde jen o to ho zformulovat v jazyce Baltíka.

Druhý příklad je řešitelný jen pomocí základních příkazů `popojdi` a `otoč se`. Kromě schopnosti popsat algoritmus děti navíc potřebují umět opravit už existující kód. Vpodstatě pokaždé se jim totiž stane, že uprostřed svého programu udělají chybu a tráví dlouhý čas jejím hledáním a opravou. Hledání a oprava chyb je přitom významná část programátorských dovedností.

Třetí příklad navíc obsahuje vizuálně atraktivní prvky otevření dveří a neviditelnost baltíka.

Zadávání příkladů pro takhle malé děti samozřejmě není možné dělat pouhým předáním textu zadání tak jako v této práci. Vhodnější je příklady vysvětlit a ukázat příklad výsledku (ne řešení samotné, jen výsledek běhu programu).

Po osvojení tvorby základních algoritmů je možné pokračovat výukou opakování bloku příkazů, využívání náhodných čísel, později i pomocníků (funkcí), podmíněných příkazů, šuplíků (proměnných) a tvorby grafických progrmů.

Chápání a používání složitějších podmínek a proměnných je pro některé žáky třetích tříd někdy až neřešitelný problém. Proto je lepší nechat tato obtížnější témata na kroužek pro pokročilé.

4.2 Scratch

Z obrázku 3 se zdá, že programování ve Scratchi využívá ústřední postavičku kočky Scratche. Postavička Scratche ale není zdaleka tak podstatná, jako byla postavička Baltíka. Přispívají tomu dva fakty:

1. Scratch neumí čarovat. Tím myslím, že hlavním způsobem ovlivnění vzhledu programu není to, že by postavička Scratche vytvořila něco viditelného jako tomu bylo u Baltíka.

2. Ve Scratchi je možné jednoduše vytvořit několik dalších postaviček s různým vzhledem. Všechny postavičky jsou stejně významné, žádná z nich není ta „hlavní“, která řídí program.

V prvních hodinách Scratche je samozřejmě vhodné zvolit příklady na vytvoření jednoduchého algoritmu. K tomu slouží příkazy `pero dolu` a `pero nahoru`. Po zavolání příkazu `pero dolu` za sebou postavička Scratche při pohybu zanechává tenkou stopu. Je proto možné pomocí kroků kreslit obrázky a procvičit si tak přesný popis algoritmu.

O něco hravější alternativou je využít možnost měnit vzhled kočky Scratche a pozadí scény a vytvořit tak například zdání diskotéky. Tento druhý přístup je pro děti o něco atraktivnější. Má ale tu nevýhodu, že nevyžaduje tak přesnou formulaci algoritmu. Dítě, které udělalo v programu krok špatným směrem se s ním často spokojí, protože to v rámci diskotéky klidně může být také tak. Je tak ochuzeno o zkušenosti získané snahou o opravu špatně fungujícího programu.

Možnosti tvorby velmi jednoduchých programů jsou ve Scratchi poněkud omezenější než v Baltíku. O hodně zajímavější příklady se dají dělat až s využitím podmínek. Ke složitějším programátorským konstrukcím je tak vhodné dostat se o něco dříve než tomu bylo u Baltíka. Způsob skládání příkazů ve Scratchi využívání složitějších programátorských konstrukcí výrazně usnadňuje. I tak jsou ale podmínky logicky poměrně náročným konceptem, který je například pro třeťáky většinou neuchopitelný.

Ve Scratchi je možné vytvořit jednoduše kromě kočky další postavičku, která se bude programu účastnit. Každá postavička má potom vlastní program. Mohou spolu interagovat pomocí proměnných, nebo pomocí podmínek[1].

S těmito nástroji už lze programovat jednoduché hry. Velký úspěch například sklidila hra pro dva hráče. Každý hráč měl vlastní postavu, kterou ovládal pomocí klávesnice. Cílem bylo jako první dojít k předmětu dárek. Program naprogramovaný dětmi

[1] Například podmínka pokud se tato postavička dotýká jiné postavičky.

- řídil reakce na klávesy,
- kontroloval podmínky výhry,
- počítal vítězné skóre,
- po sebrání dárku zařídil jeho přemístění na novou náhodnou pozici.

Po dokončení programu děti zkoušely hrát právě naprogramovanou hru proti sobě.

Podobně je možné vytvořit mnoho dalších atraktivních her. Je možné i naprogramovat počítačového nepřítele s jednoduchou umělou inteligencí.

5 Srovnání

V Baltíku lze relativně snadno vytvořit jednoduché graficky atraktivní programy. Už pomocí příkazů `vyčaruj`, `popojdi` a `otoč se` můžeme vytvořit velmi zajímavé programy. Spolu s opakováním to jsou prostředky, které jsou pochopitelné už pro třeťáky[2]. Kvůli absenci konceptu čarování je ve Scratchi výrazně větší problém vytvářet zajímavé jednoduché programy tj. programy bez použití proměnných a podmínek. Kolem čtvrté třídy základní školy děti typicky dokáží pochopit koncept podmínek a proměnných. Vyučovat ve Scratchi děti mladší je proto problematické. Pro mladší děti se mi lépe osvědčil Baltík.

Ve chvíli, kdy se děti naučí používat podmínky a proměnné, přestává vadit absence konceptu čarování ve Scratchi. S těmito programovacími nástroji lze totiž vytvářet velmi zajímavé programy i bez toho. Ve stejné chvíli se projevuje výhoda Scratche. Možnost tvorby více postaviček s vlastním programem umožňuje tvorbu různých her pro dva hráče, nebo her s jedním, nebo i více protihráči. Toto je u Baltíka velmi komplikované. Baltík totiž neumožňuje současný běh dvou částí programu. Je proto velmi netriviální naprogramovat například dva Baltíky současně sbírající nějaké předměty. Možné to je, ale pro žáky prvního stupně ZŠ je to neúměrně složté programování.

Ve chvíli, kdy děti dokáží pracovat s podmínkami, se projevuje další výhoda Scratche. Ze vzhledu jednotlivých příkazů je na první pohled velmi dobře vidět, co přesně dělají. Například příkaz podmínky je graficky proveden tak intuitivním způsobem, že děti ve svém programu jednoznačně vidí, která část programu je podmíňená a která není. Toto platí i pro spoustu dalších strukturově složitých příkazů.

Příklad: V obou dvou dětských programovacích jazycích se občas hodí použít konstrukci opakuj nekonečněkrát. Tato konstrukce dětem činí potíže. Chápou sice, že příkazy, které jsou uvnitř této konstrukce se budou nekonečněkrát opakovat. Potíže jim ale činí vnímat, že program nekonečnou smyčku nikdy neopustí a příkazy za ní následující proto nikdy nebudou provedeny. V Baltíkovi takové opakování vytvoříme stejně jako jakékoliv jiné. Napíšeme znak ∞ a za něj složené závorky s příkazy, které se mají opakovat. Za tyto příkazy je možné psát další příkazy. Ve Scratchi neexistuje znak ∞. Místo toho je tam blok opakuj dokola, který graficky odděluje příkazy, které se mají opakovat. Není tedy potřeba psát závorky tj. nehrozí syntaktická chyba[3]. Navíc za tímto blokem chybí „pacička“, za kterou by bylo možné připojit další příkaz. Děti sice zpočátku nechápou, proč tam nelze připojit další příkaz. Přinutí je to ale k přemýšlení. Část z nich na to přijde sama a ostatní si to alespoň snadno zapamatují.

K uvedenému příkladu bych ještě dodal, že z programátorského hlediska existuje rozumný důvod, proč umožnit připojování dalších příkazů za nekonečný cyklus (příkaz `break`). Z didaktického hlediska se to ale zdá být nešťastné rozhodnutí.

Za poslední výhodu Scratche považuji jeho přístupnost. Je zdarma a je možné v něm programovat přímo ve webovém prohlížeči bez nutnosti instalovat další programy. Webové stránky

[2] Nejspíš i pro druháky. Osobně s nimi ale zkušenosti nemám.

[3] Syntaktická chyba je chyba způsobená posloupností příkazů, které nedávají smysl. V Baltíkovi by takovou chybou mohlo být například spojení ∞ a } tj. nekonečnokrát opakuj konec bloku příkazů. Ve Scratchi syntaktické chyby vpodstatě neexistují. Bloky které za sebou nedávají smysl prostě nelze spojit.

Scratche navíc obsahují nástroje, pomocí kterých můžou děti z celého světa sdílet programy svoje programy a hry[4].

6 Význam pro dospělé programovací jazyky

Po několikaleté výuce v dětském programovacím jazyce Baltík jsem jednu třídu dvanácti dětí v šesté třídě ZŠ začal učit v „dospělém“ programovacím jazyce Java. Informace uvedené v této části vycházejí ze zkušeností získaných jejich výukou.

Některé prostředky zmíněných dětských programovacích jazyků jsou záměrně velmi podobné prostředkům dospělých programovacích jazyků. Děti, které prošly výukou v dětském programovacím jazyku proto neměly sebemenší problém s pochopením proměnných, podmínek a funkcí. Potíže neměly ani s formulací algoritmu za předpokladu, že si vzpomněly na všechny potřebné příkazy.

Právě množství potřebných příkazů dělalo některým potíže. Zároveň se z programování vytratila grafická přívětivost a atraktivnost výsledku. Přibližně polovina dětí proto během roku ztratila motivaci a v programování dále nepokračovala.

V průběhu roku dětem dělalo drobné potíže i správné použití cyklů. Způsob použití jednoduchých[4] cyklů v Baltíku se totiž značně liší od způsobu použití cyklů v dospělých programovacích jazycích.

Reference

[1] *SGP Systems: Baltík.* [online]. [cit. 2015-09-19]. Dostupné z: `http://sgp.cz/cz/`

[2] *Scratch.* [online]. [cit. 2015-09-19]. Dostupné z: `https://scratch.mit.edu/`

[3] Mitchel Resnick. *Computer as Paintbrush: Technology, Play, and the Creative Society.* [online]. [cit. 2015-09-19].
Dostupné z: `http://web.media.mit.edu/~mres/papers/playlearn-handout.pdf`

[4] Ricarose Roque, Natalie Rusk, Amos Blanton. *Youth Roles and Leadership in an Online Creative Community.* [online]. [cit. 2015-09-19]. Dostupné z:
`https://cdn.scratch.mit.edu/scratchr2/static/__363b514c9165583d216684aff977fc77__/pdfs/research/Roque_etal_2013_Youth_Online_Roles_CSCL.pdf`

[4] Jednoduchými cykly myslím cykly typu x-krát opakuj. I Baltík obsahuje for cykly. Jejich výukou jsme ale v hodinách dětského programování nestrávili mnoho času.