Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Didaktika programovania pre ZŠ 2

Predmet: Didaktika programovania pre ZŠ

Línia: Didaktika informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Didaktika programovania pre ZŠ 2

Identifikácia modulu

Aktivita projektu:	1.2 Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Didaktika informatiky a informatickej výchovy
Predmet:	Didaktika programovania pre ZŠ

Garant predmetu:

RNDr. Gabriela Lovászová, PhD.
KI FPV UKF, Nitra
glovaszova@ukf.sk

Autori:

RNDr. Gabriela Lovászová, PhD., KI FPV UKF Nitra
Mgr. Martin Cápay, PhD., KI FPV UKF Nitra
PaedDr. Viera Palmárová, PhD., KI FPV UKF Nitra

Zaradenie modulu

Modul Didaktika programovania pre ZŠ 2 je pokračovaním modulu Didaktika programovania pre ZŠ 1. Je jedným z troch modulov, ktoré účastníci vzdelávania absolvujú v rámci predmetu Didaktika programovania vo vzdelávacej línii Didaktika informatiky a informatickej výchovy.

Abstrakt modulu

Modul sa člení na dve časti. V prvej časti nadväzuje na modul Didaktika programovania pre ZŠ 1, v ktorom sme analyzovali učivo a navrhovali metodické postupy pre témy Korytnačia grafika, Objekty a udalosti a Tvary a procesy. V tomto module dokončíme analýzu učiva z programovania na základnej škole, venujeme sa práci na komplexných projektoch, ktoré združujú vedomosti z viacerých tém. V projektoch sa objavia aj niektoré nové témy: súradnicová grafika a premenné. Spoznanie obsahu vyučovania programovania a metodických postupov na úrovni tematických celkov je predpokladom kvalifikovaného zostavovania časovo-tematických plánov. V module navrhujeme možné poradie tém. Vedomosti o konštruktívnom poznávacom procese využijeme v návrhu vyučovacích hodín.

V druhej časti modulu majú účastníci vzdelávania aktívne tvoriť a prezentovať prípravy na vyučovacie hodiny. Využijú vedomosti zo všetkých troch modulov predmetu Didaktika programovania. V skupinách spracujú rôzne témy z vyučovania programovania na základnej škole, prezentujú ich pred kolegami a diskutujú o použitých metodických postupoch. V študijnom materiáli sú ukážky metodických listov pre vybrané témy.

Modul má e-learningovú podporu

Obsah

Úvod

V module Didaktika programovania pre ZŠ 1 [9] sme sa zoznámili s cieľmi vyučovania programovania na základnej škole. Zvolili sme programovacie prostredie Imagine, ktoré je určené na vyučovanie programovania a pre ktoré existuje učebnicová podpora [3] a ďalšie materiály pre učiteľov v slovenčine. Zoznámili sme sa s obsahom vyučovania programovania v Imagine Logu, ktorý je primeraný veku žiakov základnej školy, a so všeobecnými metodickými postupmi a námetmi na vyučovanie jednotlivých tém. Analýzu učiva z programovania na základnej škole dokončíme témou Komplexné projekty, v ktorých sa využívajú vedomosti z viacerých tém a objavia sa v nich aj niektoré nové témy: súradnicová grafika a premenné. Vedomosti o obsahu a metodike vyučovania programovania nám umožnia kvalifikovane naplánovať vyučovanie programovania v rámci predmetu informatika.

V tomto module si pripomenieme niektoré všeobecné didaktické prístupy k vyučovaniu z modulov Moderná škola 2 – Východiská a inšpirácie [6], Moderná škola 4 – Digitálne technológie a zásahy do vyučovania [7] a Didaktika programovania 0 [10]. Pri návrhu konkrétnych vyučovacích hodín budeme aplikovať konštruktivistický prístup, ktorý je podľa nášho pedagogického presvedčenia najvhodnejší na to, aby sa žiak učil s porozumením a kultivovali sa jeho všeobecné poznávacie schopnosti a celá osobnosť.

V module budete mať príležitosť prezentovať svoje nápady, diskutovať o nich s kolegami, argumentovať, vysvetľovať svoje metodické postupy, kriticky hodnotiť postupy iných.

Cieľ modulu

Cieľom modulu je, aby účastníci vzdelávania vedeli:

- navrhovať komplexné projekty primerané úrovni programátorských schopností žiakov,
- plánovať vyučovanie programovania v rámci predmetu informatika na základnej škole,
- pripravovať vyučovacie hodiny programovania a zdôvodniť svoje metodické postupy,
- hodnotiť a konštruktívne diskutovať o rôznych metodických postupoch na vyučovanie jednej témy.

Vstupné vedomosti

Požadované prerekvizity

Predpokladá sa, že účastníci vzdelávania absolvovali moduly Didaktika programovania 0 a Didaktika programovania pre ZŠ 1.

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Predpokladá sa, že účastník vzdelávania:

- je zručný v programovaní v prostredí Imagine Logo,
- pozná ciele vyučovania programovania na ZŠ,
- má ucelenú predstavu o obsahu vyučovania programovania na ZŠ, dokáže podrobne analyzovať učivo,
- má ucelenú predstavu o metodike vyučovania jednotlivých tém z programovania v Imagine Logu,
- vie uviesť jednoduché príklady na demonštrovanie nových pojmov, dokáže vytvoriť úlohy rôzneho typu.

1 Komplexné projekty

Na rozdiel od konkrétnych úloh, ktoré učiteľ zadáva premyslene s cieľom usmerňovať poznávací proces žiakov smerom k osvojeniu si nejakého učiva, práca na komplexných projektoch slúži na upevnenie vedomostí, vytvorenie väzieb medzi rôznymi témami a rozvíjanie ďalších kompetencií žiaka (nielen poznávacích). Zadanie komplexných projektov je otvorené, necháva žiakom priestor na špecifikáciu zadania a voľnosť v spôsobe riešenia. Dôležitou súčasťou práce na projekte sú aj neprogramátorské činnosti, ktoré súvisia s dizajnom. Cieľom komplexných projektov je, aby žiaci vedeli:

- tvorivo špecifikovať zadanie projektu,
- analyzovať problém a navrhovať rôzne riešenia,
- aplikovať vedomosti z viacerých tém v jednom projekte.

V nasledujúcich častiach uvádzame príklady komplexných projektov typu:

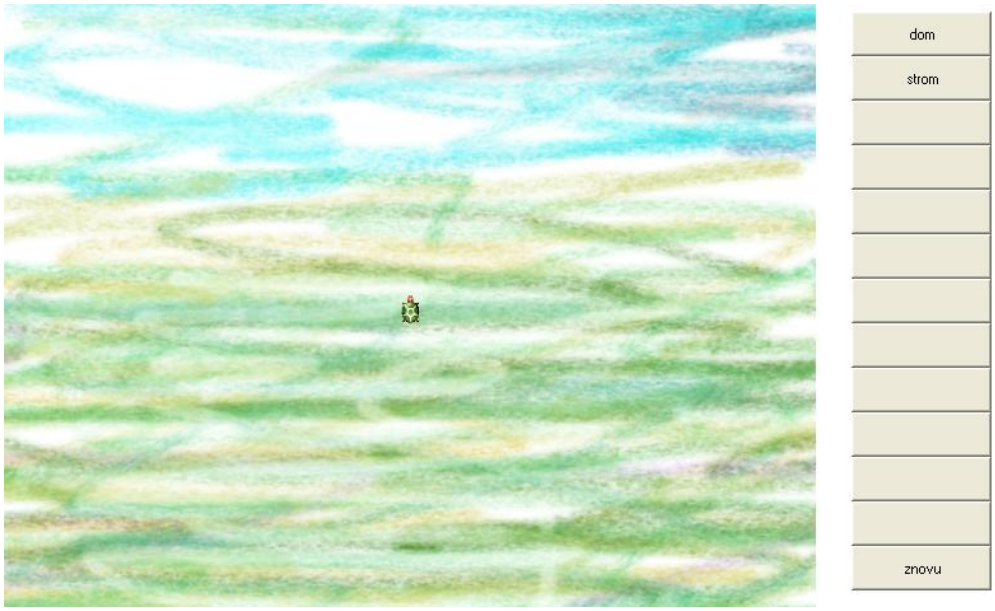
- tvorivé prostredia,
- príbehy zo živých obrazov,
- hry.

Tvorivé prostredia

Mesto

Námetom na tvorivé prostredie, v ktorom si žiaci môžu nakresliť pekné obrázky mesta s domami, stromami, cestami, ihriskami, parkoviskami, vodou atď., je projekt z knihy [2] o programovaní v staršej slovenskej implementácii jazyka Logo pre osobné počítače – Comenius Logo.

Naprogramujme tvorivé prostredie na kreslenie mesta. Prostredie bude obsahovať papier s pozadím, do ktorého sa bude stavať mesto, korytnačku, ktorá bude kresliť, a tlačidlá, ktoré budú slúžiť na pohodlnejšie zadávanie príkazov.



Obrázok 1.1: Tvorivé prostredie na kreslenie mesta

Pri tvorbe takéhoto projektu predpokladáme, že žiaci poznajú základné korytnačie príkazy, vedia vytvoriť tlačidlo a načítať a uložiť pozadie stránky (alebo papiera). Cieľom projektu je, aby si žiaci sami navrhli tlačidlá, ktoré sa im budú hodiť pre nakreslenie mesta a pomocou nich alebo aj zadávaním príkazov v príkazovom riadku nakreslili čo najpestrejšie mesto podľa vlastnej fantázie.

Príprava pozadia

Pozadie si žiaci pripravia v grafickom editore RNA (Revelation Natural Art) – voskovou pastelkou načmárajú v odtieňoch zelenej farby plochu, na ktorú sa bude stavať

mesto, a v odtieňoch modrej oblohu. Do projektu vložia papier a načítajú mu pripravené pozadie. Na stránke vedľa papiera necháme voľné miesto pre tlačidlá.

Ak na stránke bola korytnačka, zrušme ju. Papier by ju zakryl.

Príprava tlačidiel

Vedľa papiera s pozadím vložíme tlačidlo. Nastavíme mu vhodnú veľkosť. Popis a reakciu na udalosť `priZapnutí` necháme zatiaľ prázdne. Tlačidlo skopírujeme pomocou schránky toľkokrát, koľko sa zmestí na stránku. Tlačidlá presnejšie umiestnime príkazom *Presúvaj* z miestnej ponuky tlačidla alebo nastavením pozície v rodnom liste.

Programovanie

Do projektu na papier s pozadím vložíme korytnačku, ktorá bude kresliť mesto pomocou čiar s rôznou hrúbkou a farbou pera, bodov a odtláčaním obrázkov domov a stromov (obr. 1.2).

Na tlačidlách môžu byť napríklad príkazy:

```
do 30
vp 90
vp 15
vl 15
ph
pd
odtlačObrázok "dom
odtlačObrázok "strom
pd do 10 ph do 10
znovu
```

Ak sa na tlačidlo nezmestia príkazy, ktoré sa majú vykonať pri udalosti `priZapnutí`, môžeme uviesť aj iný kratší Popis.

Popis:
`strom`

priZapnutí:
`odtlačObrázok "strom`

Obrázok 1.2: Obrázok vytvorený v tvorivom prostredí Mesto

Príkazy pre korytnačku zadávame v príkazovom riadku a často používané príkazy alebo postupnosti príkazov vložíme na tlačidlá. Vytvorený obrázok uložíme do súboru voľbou *Ulož pozadie* z miestnej ponuky papiera.

Projekt sa dá ľubovoľne rozšíriť pridaním obrázkov na odtláčanie: viac domov, viac stromov, autá a iné objekty vhodné do mesta. Ak máme viac obrázkov domov a stromov, môžeme pri odtláčaní vyberať náhodný z nich:

Súbory s obrázkami skopírujme do priečinka s projektom.

```
odtlačObrázok ?prvok [dom1 dom2 dom3 dom4]
```

Ak sa žiakom obrázok nevydarí, môžu si načítať prázdne pozadie a začať s kreslením odznovu. Zmazať obrázok a načítať prázdne pozadie môžu aj príkazom `pozadieZoSúboru`, ktoré umiestnia na tlačidlo. Treba však uviesť adresáta príkazu - papier:

```
papier1'pozadieZoSúboru "pozadie_pre_mesto.jpg
```

Úloha 1.1

V projekte *mesto.imp* definujte pre pripravené tlačidlá vhodné príkazy a pomocou nich nakreslite mesto podobné mestu na obr. 1.2.

Projekt *mesto.imp* je v e-learningovom kurze k tomuto modulu.

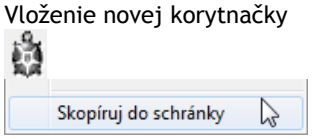
Robot

V učebnici [3] sú na strane 30 prvýkrát použité príkazy `odtlač` a `nechZáber`. Vysvetlenie je prezentované na projekte *Drevené kocky*, kde je možné zo stavebníc vyskladať obrázok podľa návodu. Tento typ projektu môžeme pomenovať „pečiatkovanie“ alebo „skladačka“. Motiváciou žiakov v týchto projektoch je predstava, že si z malých častí podľa vlastných predstáv vytvoria zložitejší zmysluplný obraz. V projekte Robot ide o rovnaký typ úlohy s motivujúcim a vizuálne atraktívnym námetom.

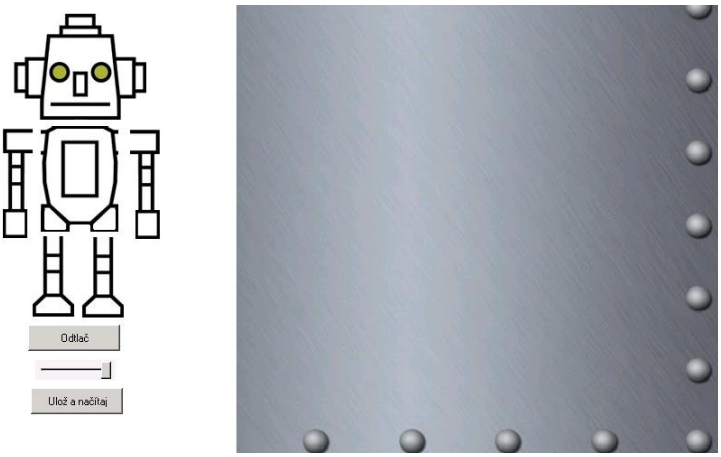
Na úvod by sme mohli odtláčať robotov ako celok.

Vytvorme projekt Robot. Cieľom projektu je z vopred pripravených častí poskladať ľubovoľného robota. Časti skladačky budú samostatné korytnačky s automatickým ťahaním.

Vloženie novej korytnačky



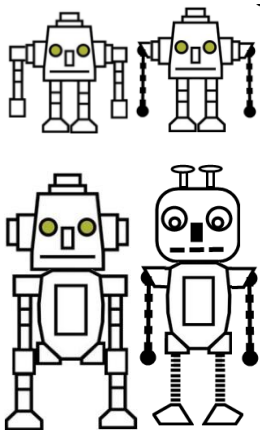
Ak chceme mať viac objektov rovnakého typu, je neefektívne programovať každý jeho výskyt samostatne. Najskôr nastavíme všetky potrebné vlastnosti pre jeden objekt a potom ho naklonujeme (skopírujeme) potrebný počet krát.



Obrázok 1.3: Tvorivé prostredie na skladanie robotov

Aby sme mohli úlohu využiť na postupné zbieranie skúseností, popíšeme projekt v gradovaných krokoch. Začnime tou najjednoduchšou formou. Ide o úlohu vhodnú na precvičenie pridávania a zmeny tvaru viacerých korytnačiek. Na stránku potrebujeme umiestniť šesť korytnačiek (hlava, telo, dve ruky a dve nohy). Nastavíme im ☑ Automatické ťahanie a ☐ Pero dolu. Vlastnosti stačí nastaviť cez rodný list alebo kopírovaním už nastavenej korytnačky cez schránku (skopírujú sa aj všetky jej vlastnosti).

Ak pri kreslení tvarov druhého robota dodržíme proporcie prvého, tak môžeme medzi sebou ľubovoľne kombinovať ich časti, čím môžu vzniknúť netradičné kreácie.



Veľmi dôležitá fáza projektu je príprava tvarov jednotlivých korytnačiek. Každej korytnačke cez rodný list priradíme jeden z vytvorených tvarov. Ich kombináciou môžeme dostať zaujímavé výzory robotov.

Teraz projekt upravíme tak, aby sme pri skladaní mali k dispozícii tvary z viacerých rozdielnych robotov. Umožníme vyskladaného robota opečiatkovať na pozadie a vrátiť jeho časti na svoje domovské miesto. Pre každú korytnačku nastavíme domovskú pozíciu (*rodný list* > *pozícia* > *prevezmi*).

Odtlačíme všetky tvary korytnačiek na pozadie príkazom `mámRobota`, ktorý vyvoláme stlačením tlačidla s popisom *Odtlač*.

Problémom môže byť odtláčanie tvarov, ktoré neboli použité pri skladaní robota. Namiesto základného príkazu `odtlač` preto použijeme na odtláčanie vlastný príkaz `odtláčaj`. Odtlačenie nastane len v prípade, že korytnačka je umiestnená na pripravenom pozadí za pomyselnou čiarou (x-ová súradnica -100), a korytnačka sa vráti na domovskú pozíciu.

```
viem odtláčaj
 ak (xSúr>-100) [odtlač] domov
koniec
```

```
viem mámRobota
 hlava'odtláčaj
 rukaL'odtláčaj
 rukaP'odtláčaj
 telo'odtláčaj
 nohaL'odtláčaj
 nohaP'odtláčaj ...
koniec
```

V projekte máme veľmi veľa častí (z viacerých robotov), čo znižuje prehľadnosť. Upravíme teda tvary tak, aby rôzne tvary pre každú časť tela (hlava, trup, ľavá ruka, pravá ruka, ľavá noha, pravá noha) boli zoskupené do jedného tvaru s viacerými zábermi.

V LogoMotion (prípadne v RNA) si zoskupením už vytvorených častí pripravíme tvary s viacerými zábermi. Zábery sa pre tento typ projektu nebudú animovať podľa uhla.

V Imagine vyberieme korytnačku, ktorá bude predstavovať napríklad hlavu robota. Nastavíme jej tvar *hlava.lgf* a v rodnom liste zvolíme určovanie záberu (a samozrejme automatické ťahanie a zdvihnuté pero ostáva). Chceli by sme, aby sa pri kliknutí na korytnačku zmenil záber. Nastavme preto reakciu na udalosť `priKliknutí` nasledovne:

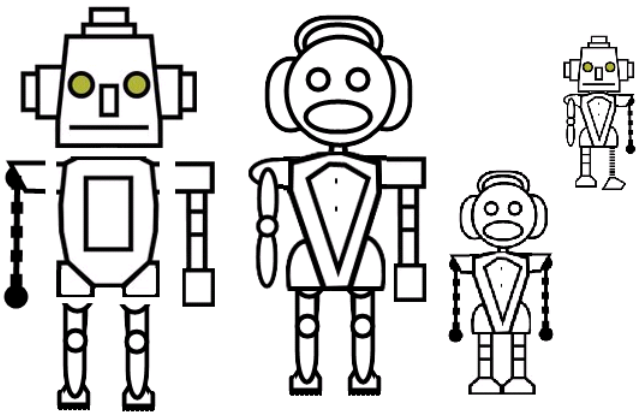
```
nechZáber zábe r+1
```

Keď ale vyskúšame, ako korytnačka reaguje, zistíme, že zmena tvaru nastáva nielen pri kliknutí, aj pri udalosti `priŤahaní`. Prečo je tomu tak? Udalosť `priKliknutí` totiž nasleduje bezprostredne po udalosti `priŤahaní`. Kliknutie totiž spočíva v stlačení tlačidla myši a jeho následnom uvoľnení. Pričom je jedno, či sme tvar medzitým presunuli alebo nie. Môžeme však korytnačke prikázať, aby impulzom pre zmenu tvaru bola udalosť `priDvojkliknutí`.

Takto nastavenú korytnačku „naklonujeme“ cez schránku pre všetky ostatné tvary. Odporúčame vhodne usporiadať poradie korytnačiek. Hlavu a telo dajme navrch (príkazom *Daj navrch* z miestnej ponuky korytnačky).

Na záver môžeme ešte pridať možnosť zmeniť veľkosť tvaru robota, čo môžeme realizovať objektom posúvač. Projekt môžeme oživiť vhodným pozadím, prípadne ponúknuť možnosť uložiť pozadie ako obrázok stlačením ďalšieho tlačidla:

```
uložPozadie "robot.bmp znovu nechPozadie "pozadieROBOT
```

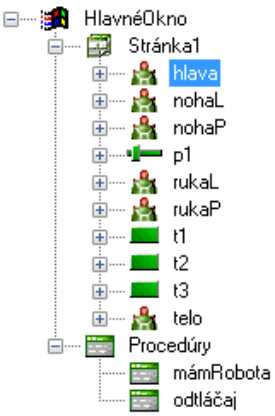


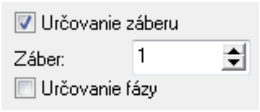
Obrázok 1.4: Príklady obrázkov vytvorených v projekte Robot

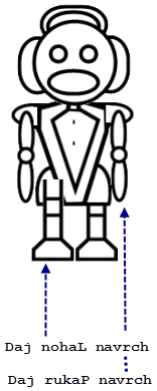
Úloha 1.2

Pridajte do programu ešte jedno tlačidlo *Náhodný*, ktoré bude tvary všetkých korytnačiek meniť náhodne.

Štruktúra projektu:







Nastavenia posúvača:

minimum = 1
maximum = 5
hodnota = 5

Reakcia na udalosť `priZmene` posúvača:

```
pre všetky [
  nechZväčšenie
  zväčšenie/5
]
```

Variácie: Namiesto konštrukcie robota môžete ozdobovať tortu, vianočný stromček, veľkonočné vajíčko, zariaďovať izbu, stavať mesto, stavať cesty (koľajnice), zostavovať identikit človeka atď. Diskutujte o ďalších variáciách.

Príbehy zo živých obrazov

Moja obľúbená riekanka

Tento projekt je ukážkou viacstránkového interaktívneho projektu.

Medzi stránkami sa v ňom prepíname pomocou vlastných tlačidiel.

Naprogramujeme reakciu na udalosť `priOtvoreníStránky`.

Do projektu vložíme tlačidlo na prehratie zvukového súboru.

Projekty tohto typu sú priestorom na rozvíjanie tvorivosti a fantázie, uplatnenie medzipredmetových vzťahov a prierezových tém. Sú dobrým námetom na vyučovacie hodiny zamerané na záverečné opakovanie a systematizáciu vedomostí a zručností.

Vytvoríme projekt s viacerými stránkami, na ktorých oživíme detskú riekanku:

„Mám koníka bieleho,
posadím sa na neho.
Hijó, cválaj koník môj!
A keď poviem pŕŕ, tak stoj!"

Premyslime si scenár krátkeho príbehu:

- každému veršu venujeme samostatnú stránku,
- stránkam nastavíme vhodné pozadie (lúka, les, pole a pod.),
- na každej stránke zobrazíme vľavo hore znenie aktuálneho verša (v textovom poli `text1`),
- každá stránka bude mať vpravo hore tlačidlo `t1` na presun na nasledujúcu stránku,
- na prvej stránke ukážeme koňa bez jazdca (korytnačku `k1` s tvarom *kon.lgf*),
- na druhej stránke ukážeme koňa s jazdcom (korytnačku `k1` s tvarom *konSJazdcom.lgf*),
- na tretej stránke spustíme proces, ktorým zabezpečíme pohyb koňa s jazdcom v sedle v smere zľava doprava
  - v reakcii na udalosť `priKliknutí` zmeníme smer cválania,
- na štvrtej stránke pripravíme tlačidlo na prehratie zvuku a naprogramujeme reakciu na udalosť `priZapnutí`
  - po kliknutí na toto tlačidlo sa prehrá zvukový súbor (so zvolaním „pŕŕ!"),
  - cválajúci kôň zastane.

Obrázok 1.5: Pohľad na prvú stránku projektu riekanka.imp

Ako začať

Už vieme, že v projekte potrebujeme štyri stránky. Pridáme ich postupným klikaním na tlačidlo  . Stránky budú mať mená `Stránka1`, `Stránka2`, `Stránka3` a `Stránka4`.

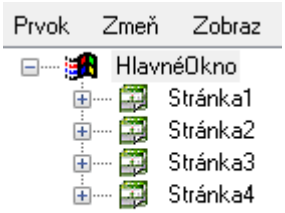
Na prvej stránke pripravíme tlačidlo s nápisom „Ďalej“, ktorým sa budeme presúvať na ďalšiu stránku (nasledujúci verš riekanky). Tlačidlu naprogramujeme reakciu na udalosť `priZapnutí`. Meno stránky, ktorú chceme zobraziť, napíšeme ako príkaz:

```
Stránka2
```

Rovnaké tlačidlo chceme mať aj na ostatných stránkach. Môžeme si pomôcť schránkou:

1. nastavme sa v okne *Pamäť*, na tlačidlo `Stránka1't1`,
2. stlačme CTRL+C (objekt skopírujeme do schránky),
3. nastavme sa v okne *Pamäť* na stránku, do ktorej chceme vložiť kópiu tlačidla,
4. stlačme CTRL+V (objekt prilepíme zo schránky),
5. na novom tlačidle upravíme reakciu na udalosť `priZapnutí` (v prípade posledného tlačidla aj popis).

Takto môžeme kopírovať aj celé stránky so všetkými objektmi, ktoré obsahujú.



Označte v okne *Pamäť* `Stránku1`, stlačte CTRL+C, nastavte sa do koreňa stromu objektov *HlavnéOkno* a stlačte CTRL+V. Do stromovej štruktúry projektu pribudne nová stránka.

Stránka1: „Mám koníka bieleho,“

Na prvej stránke umiestnime korytnačku-koníka do ľavého dolného rohu a nastavme jej aktuálnu pozíciu ako domovskú. K domovskému stavu korytnačky patrí okrem pozície aj smer natočenia, prispôsobíme ho zobrazenému tvaru (nastavíme na hodnotu 270). Na stránku vložíme textové pole, nastavíme formát písma v *Paneli písma* a vložíme prvý verš riekanky. Aby sa po spustení projektu zaručene objavila ako prvá stránka s úvodným veršom, pridáme do projektu príkaz `štart`. V ňom zabezpečíme zobrazenie prvej stránky: `Stránka1`.

Domovský stav korytnačke nastavujeme v rodnom liste na karte *Pozícia*.

Stránka2: „posadím sa na neho.“

Na stránku vložíme korytnačku, tlačidlo a textové pole. Ak sme druhú stránku vytvorili ako kópiu prvej, máme na nej všetky potrebné objekty už pripravené. Zmeníme obsah textového poľa na druhý verš, zmeníme tvar korytnačky a v reakcii tlačidla „Ďalej“ napíšeme ako príkaz meno nasledujúcej stránky.

Stránka3: „Hijó, cválaj koník môj!“

Po otvorení tretej stránky sa má korytnačka-kôň začať pohybovať. Pridajme stránke udalosť `priOtvoreníStránky`. V reakcii na túto udalosť premiestime korytnačku domov a spustíme proces:

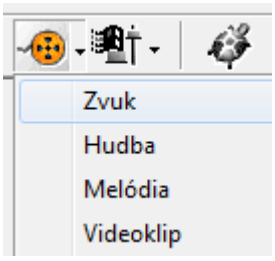
```
pre "k1 [domov každých 100 [do 2]]
```

Korytnačka-kôň má reagovať na udalosť `priKliknutí` tak, že zmení svoj smer na opačný. Tvar korytnačky *konSJazdcom.lgf* má dva zábery, stačí ju len otočiť čelom vzad.

Zamyslime sa

Prečo sa medzi stránkami prepíname vlastným tlačidlom? Mohli sme predsa klikať na tlačidlá stránok na *Hlavnom paneli*.

Stránka4: „A keď poviem prŕr, tak stoj!“

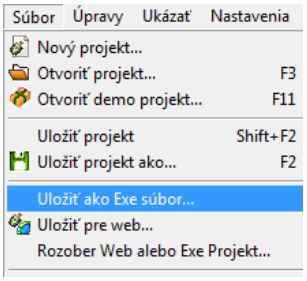


Na štvrtej stránke doplníme do reakcie na udalosť `priOtvorení` príkaz `k1'nechPoz Stránka3'k1'poz`. Tým zabezpečíme, že korytnačka-kôň bude pokračovať v cválaní na rovnakej pozícii ako na predošlej stránke.

Nakoniec vložíme tlačidlo na prehratie zvukového súboru (s nahrávkou zvolania „Prŕr!“) z panela nástrojov v skupine *Multimédiá*. V reakcii na udalosť `priZapnutí` sa doplní príkaz: `hrajSúbor súbor` (tlačidlo si pamätá používateľom zvolený zvukový súbor vo svojej vlastnej premennej).

Po prehratí zvuku chceme cválanie zastaviť, preto reakciu tlačidla doprogramujeme:

```
hrajSúbor súbor zastavVšetky
```



Prehliadač webových stránok (s nainštalovaným zásuvným modulom pre Imagine Logo) dokáže zobraziť projekt prostredia Imagine Logo uložený vo formáte *iip* ako aplet vložený do webovej stránky.

Úloha 1.3

Otvorte hotový projekt *riekanka.imp*, preskúmajte a otestujte ho. Potom projekt uložte na disk:

a) ako samostatný spustiteľný exe súbor,
b) vo formáte pre web.

Úloha 1.4

Podeľte sa s kolegami o nápad na originálny projekt, ktorý by bolo vhodné vytvoriť ako sériu po sebe nasledujúcich stránok (rozprávka, komiksový príbeh, riekanka, hádanka, reklama, minihra, prezentácia o rodine, netradičné spracovanie učiva z iného predmetu atď.).

Stránky projektu môžu okrem pekného pozadia obsahovať ďalšie objekty (nielen korytnačky) a dynamické, interaktívne či multimediálne prvky (môžeme pripraviť animované tvary, na stránkach spúšťať procesy, naprogramovať objektom reakcie na udalosti, prehrávať rôzne sprievodné zvuky, hudbu a pod.).

Hry

Kameň, papier, nožnice

Vytvorme projekt, v ktorom budeme simulovať priebeh známej rozhodovacej hry. Hru hrajú dvaja hráči, ktorí v dohodnutý (odpočítaný) moment ukážu jedno z troch gest: *kameň* (zovretá päsť), *papier* (otvorená dlaň), *nožnice* (dva prsty). Papier víťazí nad kameňom, kameň nad nožnicami a nožnice nad papierom (pretože „papierom obalíme kameň, kameňom otupíme nožnice, nožnicami rozstriháme papier“).

V tomto projekte budeme pracovať s viacerými objektmi:

Korytnačkám nastavíme tvar s tromi zábermi, ktoré budeme náhodne striedať.

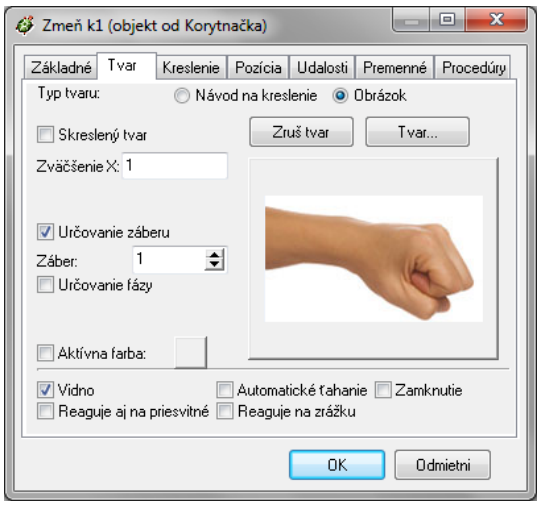
Tlačidlám naprogramujeme reakciu na udalosť `priZapnutí`.

V textových poliach zobrazíme bodové skóre hráčov. Pri vyhodnocovaní výsledku použijeme príkaz vetvenia.

V nadväzujúcej úlohe doplníme do projektu globálnu premennú, aby sme mohli hru po dosiahnutí cieľového skóre ukončiť a vyhlásiť jedného z hráčov za víťaza.

Základná verzia projektu

Na prázdnu stránku vložíme dve korytnačky. Jednej nastavíme tvar *rukaZlava.lgf*, druhej tvar *rukaSprava.lgf*. Preskúmame tvar prvej korytnačky. Obsahuje tri zábery, ktoré budeme môcť pri hre ľubovoľne striedať príkazom `nechZáber`. Nezabudnime, že korytnačka musí mať v rodnom liste zapnuté nastavenie *Určovanie záberu* (obr. 1.6).



Obrázok 1.6: Nastavenie vlastnosti Určovanie záberu v rodnom liste korytnačky

Vložíme na stránku tlačidlo `t1`, nápis na tlačidle nastavíme na „Raz dva tri“. V reakcii na udalosť `priZapnutí` zmeníme korytnačkám záber na náhodný: každá „ruka“ tak ukáže niektoré z gest kameň, papier alebo nožnice:

```
pre všetky [ nechZáber ? ]
```

Vylepšenie projektu

Každý náhodný pokus, ktorý stlačením tlačidla v projekte zrealizujeme, chceme vyhodnotiť. Pridáme preto do projektu ďalšie objekty - textové polia, v ktorých budeme evidovať počet výhier, zvlášť pre prvého hráča (`text1`) a zvlášť pre druhého hráča (`text2`).

Objektu `text1` nastavme v paneli *Štýl písma* font, farbu, veľkosť a počiatočnú hodnotu 0. Kopírovaním pomocou schránky vytvoríme objekt `text2` s rovnakými vlastnosťami.

Do projektu pridáme procedúru `vyhodnotenie`, v ktorej porovnáme čísla záberov na korytnačkách a zistíme, ktorý z hráčov vyhral. Ak „ruky ukazujú“ rovnaké gesto (t. j. na korytnačkách je zobrazený záber s rovnakým číslom), nevyhral nikto a preto procedúru ukončíme. Inak budeme pokračovať vo vyhodnocovaní pomocou príkazov `ak` a `ak2`:

```
viem vyhodnotenie

  ak k1'záber = k2'záber [ukonči]

  ; na prvej ruke je kameň (t. j. záber číslo 1)

  ak k1'záber = 1 [

    ak2 k2'záber = 3
                [text1'nechHodnota text1'hodnota + 1 ]
                [text2'nechHodnota text2'hodnota + 1 ]
  ]

  ; analogicky otestujeme zvyšné dve situácie

]

koniec
```

Pridáme na stránku ešte jedno ďalšie tlačidlo (s nápisom „Nová hra“). V reakcii na udalosť `priZapnutí` naprogramujeme nastavenie hodnôt v textových poliach na 0.

Úloha 1.5

Do pripraveného projektu *kpn.imp* doplňte premennú `vitazneSkore`. Ten hráč, ktorý ako prvý dosiahne požadované skóre, zvíťazí a hra skončí (napr. zobrazením textového poľa `text3` so správou).

Premenná `vitazneSkore` bude globálna. Nie je potrebné ju deklarovať, jej vytvorenie a inicializáciu urobíme jediným príkazom:

```
? urob "vitazneSkore 10
```

Príkaz `urob` má dva parametre: **meno premennej** (začína úvodzovkami) a **hodnotu**, ktorú chceme v premennej uložiť (číslo, slovo, zoznam, obrázok).

Globálne premenné sú prístupné v celom projekte, vidíme ich aj v okne *Pamäť* (F4). Hodnoty týchto premenných môžeme zmeniť aj v dialógovom okne.

Vyhodnocovanie pokusu je najťažšia časť riešenia. Na situácie, ktoré môžu pri hre nastať, sa môžeme pozrieť z pohľadu prvého hráča (v zátvorkách sú uvedené čísla záberov príslušného tvaru korytnačky):

kameň (1)
x nožnice (3) => výhra
x papier (2) => prehra

papier (2)
x kameň (1) => výhra
x nožnice (3) => prehra

nožnice (3)
x papier (2) => výhra
x kameň (1) => prehra

Podobnú analýzu môžeme urobiť aj spolu so žiakmi, zapísať ju na tabuli.

Príkaz `vyhodnotenie` môžeme žiakom predpripraviť. Podľa vzoru doplnia vetvenia týkajúce sa zvyšných dvoch situácií.

Ďalšou možnosťou je porovnať hotový príkaz s analýzou na tabuli, „skontrolovať, či v ňom nie je chyba“. Žiaci zapíšu volanie príkazu na správne miesto.

Ak je pre žiakov štruktúra programu príliš zložitá, môžeme namiesto príkazu `ak2` použiť dva samostatné príkazy `ak`.

Projekt *kpn.imp* je v e-learningovom kurze k tomuto modulu.

Úloha 1.6	V okne *Pamäť* kliknite na premennú `vitazneSkore` a zmeňte jej hodnotu na 5.

Do procedúry `vyhodnotenie` pridáme príkazy `ak`:

```
ak text1'hodnota = :vitazneSkore
     [text3'nechHodnota "|vyhral hráč 1| text3'ukážMa]

ak text2'hodnota = :vitazneSkore
     [text3'nechHodnota "|vyhral hráč 2| text3'ukážMa]
```

Pred meno premennej, ktorej hodnotu chceme prečítať, píšeme dvojbodku (podobne ako pred mená parametrov vo vlastných príkazoch).

Úloha 1.7	Opíšte, ako by ste do projektu doplnili zobrazenie jednoduchej štatistiky prebiehajúcej hry. Bolo by potrebné pridať ďalšie premenné? Dala by sa štatistika zobraziť vo forme grafu?

Úloha 1.8	Vysvetlite význam uvedených príkazov:
	
	`? urob "hráč1 0`
	`? urob "hráč1 :hráč1+1`
	`? urob "hráč2 :hráč2+1`
	`? urob "remíza :početPokusov - :hráč1 - :hráč2`
	
	`? urob "hráč1 :hráč1 / 2`
	`? urob "hráč2 :hráč2 * 10`
	
	Ako by ste ich zapísali v jazyku Pascal?

Pampúch

Vytvorme projekt *Pampúch*, v ktorom bude úlohou používateľa pomocou smerových tlačidiel navigovať príšeru „pampúcha“ na blikajúcu potravu a zároveň sa vyhýbať jedu. Potrava po „zjedení“ zmizne a pampúch narastie, pri zjedení jedu sa celý projekt zastaví.

V projekte sa vyžadujú vedomosti z oblastí: tvar korytnačky, absolútny smer, proces, udalosť a objekty.

Vytvoriť projekt takéhoto typu nie je pre žiakov ZŠ vôbec jednoduché. Stretli sme sa však s otázkou učiteľa základnej školy, ako sa také niečo dá urobiť, lebo žiaci si to vymysleli. Skúsme si porovnať riešenie v prostredí Imagine a Živý obraz.

Samotná činnosť programovania bude pozostávať z nasledovných krokov:

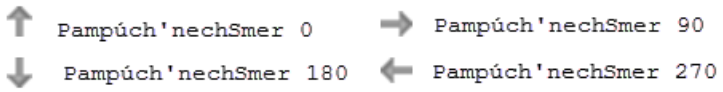
- programovanie tlačidiel,
- programovanie pampúcha,
- programovanie potravy, jedu a ich klonovanie.

Do projektu v oboch prostrediach vložíme štyri tlačidlá a tri korytnačky: pampúch, potrava a jed. Korytnačkám nastavíme tvary a v Imagine zdvihneme pero.

Imagine

Tlačidlá

Pampúch bude objekt, ktorý sa pohybuje automaticky, smer cesty určíme pomocou štyroch smerových tlačidiel, ktorým priradíme vhodné obrázky. Ich rozmiestnenie bude kopírovať rozloženie ovládacích šípok na klávesnici. Udalosť `priZapnutí` pre každé tlačidlo naprogramujeme podľa toho, ako sa má korytnačka-pampúch na ploche správať.

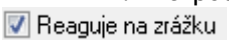


Absolútny smer môžeme prirovnať k ručičkám na hodinkách alebo ku kompasu:

`0` = sever (hore)
`90` = východ (vpravo)
`180` = juh (dolu)
`270` = západ (vľavo)

Potrava a jed

Korytnačka-pampúch má reagovať na zrážky s potravou, či jedom. Aby sme však mohli kontrolovať, či sa dva objekty zrazili, musíme im nastaviť vlastnosť *Reaguje na zrážku*. Pre potravu, jed aj pampúcha nastavíme v rodnom liste na karte *Tvar*:



Korytnačku-potravu a korytnačku-jed pomocou schránky niekoľkokrát skopírujeme na rôzne miesta na stránke. V okne *Pamäť* (F4) vidíme mená nových naklonovaných korytnačiek (potrava1, potrava2, ..., jed1, jed2, ...).

Pampúch

Pre pampúcha musíme spustiť nekonečný proces. Úlohou hráča bude počas tohto procesu nasmerovať pampúcha za potravou. Naprogramujeme vlastný príkaz `štart`, v ktorom spustíme proces.

Keď príkaz pomenujeme `štart`, tak sa pri otvorení projektu automaticky spustí.

```
viem štart
 pampúch'každých 100 [do 10]
koniec
```

Príkaz môžeme zavolať z príkazového riadka, stlačením tlačidla alebo kliknutím na pampúcha. Korytnačka-pampúch má reagovať na zrážky s potravou, či jedom. Naprogramujme reakciu pampúcha na udalosť `priZrážke`:

```
viem kontrolaPampúch
 ak2
  nieJe prázdny? prekrývajúMaZTýchto [ jed1 jed2 jed3 jed4 ]
  [ zastavVšetky ]
  [ nechZväčšenie zväčšenie + 0.2
    pre prekrývajúMa [ skryMa ] ]
koniec
```

V parametri operácie `prekrývajúMaZTýchto` uvedieme mená všetkých naklonovaných korytnačiek, ktoré predstavujú jed.

Ak pampúch prekrýva pri zrážke niektorú z týchto korytnačiek, zastaví sa. Inak sa zrazil s potravou a tú korytnačku-potravu, ktorú prekrýva, skryje.

Zistiť, s ktorou korytnačkou sa pampúch zrazil, nie je jednoduché, vyžaduje si to znalosti o práci so zoznamami. Žiaci poznajú zoznamy ako vstupy z niektorých príkazov (napr. `nechPoz`, `opakuj`, `?prvok`, `pre`), avšak podrobnejšie sa zoznamami ako údajovou štruktúrou na základnej škole nezaoberáme.

Živý obraz

V prostredí Živý obraz [8] je určovanie objektu pri zrážke jednoduchšie. Vyskúšajme naprogramovať hru v tomto prostredí.

Smer pampúcha v prostredí Živý obraz nastavujeme pomocou tlačidiel. V okne správania pre každé zo štyroch tlačidiel umiestnime ikonický príkaz *Zmeň smer* s príslušnou hodnotou parametra (obr. 1.7).

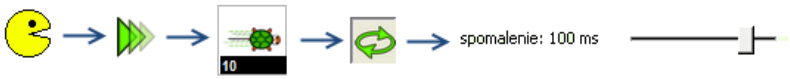
Obrázok 1.7: Reakcie pampúcha na stlačenie tlačidiel

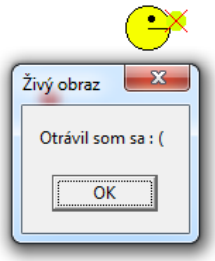
To, či je objekt aktívny, zistíme nastavením kurzora myši nad jeho tvar. Zobrazí sa bublinka „aktívna“

V tomto prostredí sa objekty neoslovujú, akcie tlačidiel sa adresujú tzv. aktívnemu objektu. Preto skôr, ako spustíme projekt, musíme kliknutím nastaviť pampúcha ako aktívny objekt.

Pampúch

Naprogramovať pampúcha vo vizuálnom prostredí je podstatne jednoduchšie. Pravým tlačidlom myši zobrazíme okno správania pampúcha a naprogramujeme jeho činnosť pri spustení hry:





a reakciu na zrážku s potravou a jedom:

Keďže v prostredí Živý obraz nevieme osloviť v okne správania pampúcha iný objekt, nevieme skryť potravu, s ktorou sa pampúch zrazil.

Potrava

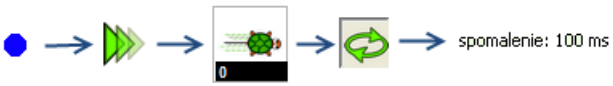
Jedinou úlohou potravy po zrážke s pampúchom je jej zmiznutie. Pravým tlačidlom zobrazíme okno správania a naprogramujeme reakciu na zrážku s pampúchom:

A nastal problém

Diskusia

Diskutujte o možnosti realizácie tohto projektu na hodinách informatiky na základnej škole. Ktoré prostredie je vhodnejšie (Imagine alebo Živý obraz)?

Keď hru spustíme, zistíme, že pampúch pri zrážke s potravou narastie, avšak potrava na zrážku s pampúchom nereaguje (nezmizne). Je to preto, že udalosť zrážka môže nastať len pre pohybujúci sa objekt. Potrava sa však nehýbe, preto nemôže do nikoho vraziť. Problém sa dá vyriešiť tak, že pre potravu pri spustení hry odštartujeme nekonečný proces, v ktorom sa bude „pohybovať“ dopredu o vzdialenosť 0:



Zaujímavou alternatívou by bolo rozhýbanie všetkých objektov. Potrava aj jed by sa posúvali o náhodnú dĺžku z určeného intervalu a následne otočili o náhodný uhol. A tak by pampúch musel objekty doslova naháňať po pracovnej ploche.

Keď máme objekty vytvorené a aj vyskúšané, využijeme nástroj *Klonovanie* a vytvoríme kópie potravy a jedu.

Základný projekt *PampusikZaklad.pzo* je v e-learningovom kurze k tomuto modulu. Nájdete tam tiež hotovú hru so všetkými rozšíreniami.

Úloha 1.9

Rozšírte projekt *PampusikZaklad.pzo*:

- potrava pri zjedení nemizne, iba mení pozíciu,
- o počítadlo zjedenej potravy,
- o ducha, ktorý naháňa pampúcha a v prípade, že ho chytí, hra končí.

2 Plánovanie vyučovania programovania

Predpokladom kvalifikovaného plánovania vyučovania je:

- znalosť učebného plánu a cieľov, sú obsiahnuté v Štátnom vzdelávacom programe,
- znalosť vzdelávacieho obsahu (učiva),
- znalosť metodických postupov na vyučovanie tohto obsahu.

Rozsah

Podľa učebného plánu nižšieho sekundárneho vzdelávania (ISCED2) sa predmet informatika vyučuje v 5. až v 9. ročníku s dotáciou 0,5 hodiny týždenne. Pri 33-týždňovom školskom roku je to 16,5 hodiny v piatich ročníkoch, teda spolu 82,5 hodiny informatiky na celom druhom stupni základnej školy.

Programovanie (tematický okruh Postupy, riešenie problémov, algoritmické myslenie) je jedným z piatich tematických okruhov vzdelávacieho obsahu predmetu informatika. Týchto päť tematických okruhov nie je obsahovo rovnako bohatých a netvoria rovnomerné pätiny obsahu celého predmetu. Po preskúmaní a porovnaní obsahu ostatných tematických okruhov môžeme pre programovanie naplánovať približne štvrtinu hodín informatiky, čo je asi 20-21 hodín na celom druhom stupni.

Štátny vzdelávací program je **záväzným** východiskom pre vytvorenie školského vzdelávacieho programu školy, kde sa zohľadňujú špecifické lokálne a regionálne podmienky a potreby. Škola má k dispozícii voľné hodiny, ktoré môže použiť na naplnenie voliteľného obsahu vzdelávania, profilovať sa v nejakom smere, vychádzať v ústrety potrebám a záujmom svojich žiakov a optimálne využiť potenciál učiteľského zboru.

Ak sa škola rozhodne voľnými hodinami posilniť vyučovanie informatiky (zvyčajne na 1 hodinu týždenne), môže rozsah vyučovania informatiky (a programovania) vzrásť až dvojnásobne (prípadne aj viacnásobne). **Rozsah** vyučovania informatiky v počte hodín je preto prvým kľúčovým parametrom pri plánovaní vyučovania programovania.

Druhým parametrom je **rozdelenie obsahu do ročníkov**. Na základnej škole sa väčšinou pri plánovaní obsahu používa „špirálový“ prístup, t.j. tematické celky vyučovacieho predmetu sa opakujú vo viacerých ročníkoch vždy na vyššej úrovni. Časový odstup vo vyučovaní jednotlivých častí tematického celku v rôznych ročníkoch vytvára priestor pre upevnenie, kryštalizáciu poznatkov.

Opačným prístupom je modulárne vyučovanie, kedy sa jedna téma preberie relatívne samostatne a uzavrie sa v rámci tzv. modulu. Študijný program sa potom poskladá zo samostatných celkov – modulov. Tento prístup je typický pre vysokoškolské štúdium a rôzne kurzy pre dospelých.

Odporúčame využiť motivačný potenciál detského programovacieho prostredia (metafora korytnačky, rôzne detské námety na projekty) a zaradiť programovanie do obsahu predmetu informatika od 5. ročníka. Vo vyšších ročníkoch základnej školy s nástupom obdobia dospievania motivácia programovať klesá a stúpa prirodzený záujem detí o komunikáciu na internete, o prácu s multimédiami. Ak škola realizuje len povinný počet hodín informatiky (z toho približne 20 hodín programovania), nemá zmysel rozdeľovať ich do všetkých piatich ročníkov. Naopak, v prípade vyššej hodinovej dotácie a kvalifikovaného učiteľa, ktorý vie vyučovať programovanie kvalitne, môžeme pre programovanie vyčleniť väčší priestor.

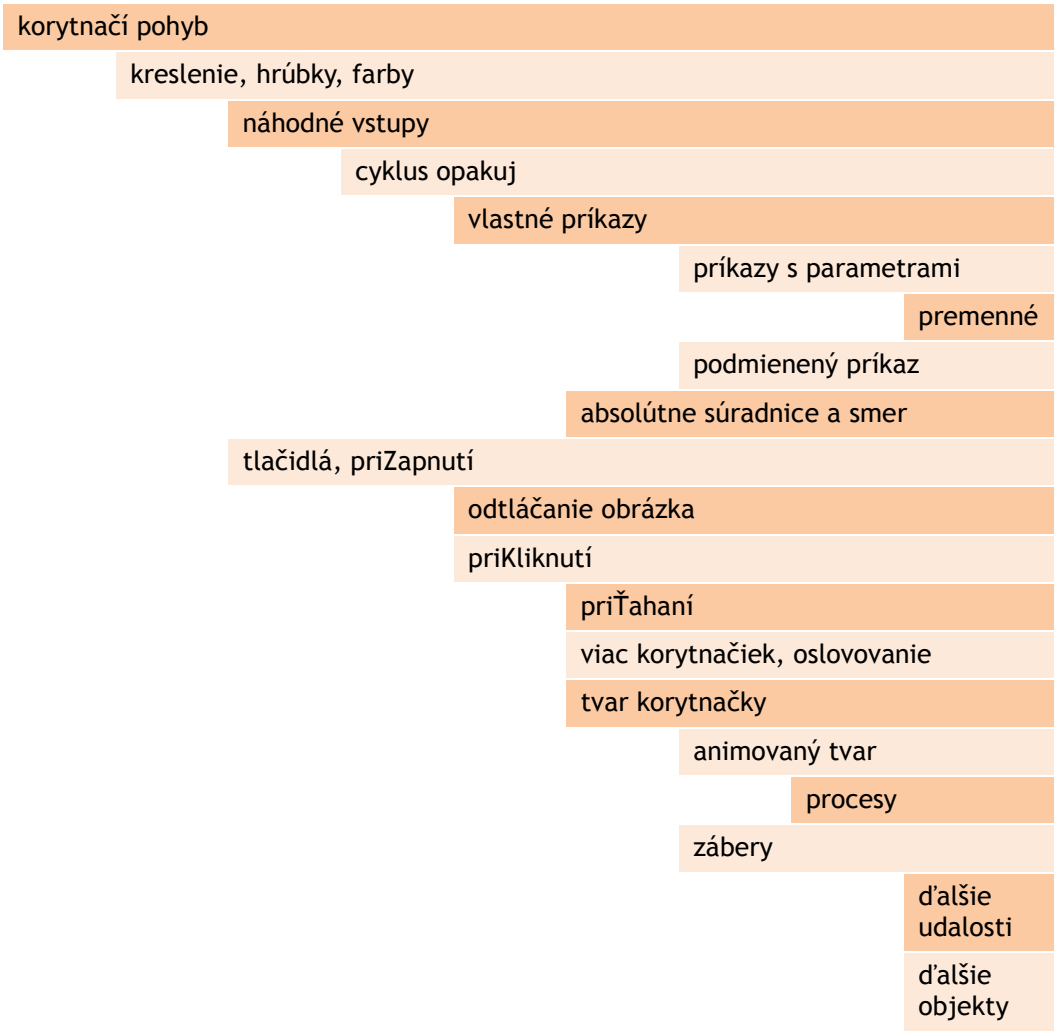
Diskusia 2.1	Diskutujte o spôsoboch rozdelenia vyučovania programovania v predmete informatika (počty hodín, zaradenie do ročníkov). Prezentujte svoje skúsenosti.

Poradie tém

Niektoré témy majú prerekvizity, bez ktorých nie je možné ich odučiť, napríklad vyučovaniu príkazov s parametrami musí predchádzať programovanie príkazov bez parametrov. O vzájomnom poradí takýchto dvojíc niet pochýb.

Niektoré témy síce nie sú nutnou prerekvizitou inej témy, ale z metodického hľadiska existuje dôvod vyučovať ich skôr. Napríklad náhodné vstupy umožňujú generovať rôzne výstupy tým istým príkazom. Tento efekt sa dá využiť pri opakovaní v cykle, preto je výhodné zaviesť náhodné vstupy pred príkazom cyklu.

V nasledujúcej tabuľke uvádzame možné poradie tém z programovania v Imagine Logu.



Diskusia 2.2

Analyzujte vyššie uvedenú tabuľku a diskutujte o vhodnom poradí tém. Prezentujte svoje skúsenosti a svoje názory podporte argumentmi.

Čo sme sa naučili

Pri plánovaní vyučovania programovania vychádzame z počtu hodín. Programovanie je jeden z piatich okruhov predmetu informatika, vyučujeme ho špirálovite vo viacerých ročníkoch. Pri zostavovaní poradia tém prihliadame na obsahové a na metodické prerekvizity.

3 Príprava vyučovacej hodiny

Východiská

Vyučovacia hodina je najmenšia organizačná jednotka vyučovania v triedno-hodinovom systéme vzdelávania. Hodiny sa pripravujú podľa vopred premysleného časovo-tematického plánu v kontexte celého tematického celku, vyučovacieho predmetu, ako aj medzipredmetových vzťahov.

Bloomova taxonómia vzdelávacích cieľov

V príprave vyučovacej hodiny navrhujeme konkrétne metodické postupy na dosiahnutie špecifických cieľov. Dosiahnutie cieľov môže mať rôzne úrovne: od zapamätania, cez porozumenie a schopnosť aplikovať nové poznatky v typických situáciách až po kryštalizáciu poznatku, pochopenie väzieb, súvislostí a schopnosť tvorivo aplikovať nové poznatky v netypických situáciách.

Naším cieľom je dosiahnuť ciele vyučovacej hodiny najmenej na úrovni *porozumenia* a získané poznatky ďalej upevňovať a prehlbovať používaním a precvičovaním na ďalších hodinách, stúpať po pyramíde vzdelávacích cieľov.

Niemierkova taxonómia vzdelávacích cieľov

Vzdelávanie, ktoré je založené na *transmisívnom* prístupe k vyučovaniu, t.j. spočíva v *prenose* hotových informácií od učiteľa k žiakovi, je ľahké, rýchle a je akceptovateľné väčšinou žiakov aj učiteľov, lebo je zamerané na dosiahnutie výsledkov, splnenie výkonu. To je často najdôležitejšia motivácia žiaka aj učiteľa. Učiteľ demonštruje vzory, poskytuje všeobecné návody, inštrukcie ako riešiť podobné problémy. Takýto prístup však môže viesť k paradoxnej situácii: Žiak vie riešiť úlohu bez toho, aby jej rozumel. Vtedy je poznanie formálne, založené na zapamätaní, nerozvíja kognitívne (poznávacie) schopnosti a nerozvíja alebo dokonca potláča osobnosť žiaka.

Konštruktívny prístup k vyučovaniu naopak aktivizuje žiaka, aby si vytváral vlastné predstavy a sám (alebo s pomocou učiteľa, spolužiakov, rodičov) konštruoval a objavoval nové poznatky. Žiak získa nové poznatky *aktívne* a *s porozumením*. Úlohou učiteľa je naštartovať konštruktívny poznávací proces vhodnou motiváciou, usmerňovať ho dobre volenými príkladmi a úlohami a v závere zhrnúť podstatné rysy učiva, verbalizovať poznatok. Nové poznanie môže byť zo začiatku krehké. Kryštalizácia poznatku (vytvorenie väzieb na iné poznatky) prebieha postupne jeho používaním v rôznych situáciách.

O „chorobe formalizmu“ a o konštruktivistických prístupoch k vyučovaniu píšu Hejný a Kuřina v [4].

Ak chceme, aby sa žiak učil konštruktívne, projektujme vyučovacie hodiny tak, že

- žiakov aktivizujeme vhodnou motiváciou,
- vytvoríme dostatočný priestor na experimentovanie, zbieranie vlastných skúseností,
- vhodnými úlohami v zóne najbližšieho vývoja posúvame žiakovo poznanie ďalej,
- zhrnieme podstatné učivo, aby sme včas korigovali prípadné nesprávne žiacke predstavy,
- získané vedomosti a zručnosti žiakov využívame v rôznych situáciách, aby sme podporili kryštalizáciu,
- do vyučovania zaraďujeme hrové, tvorivé aktivity na udržiavanie pozitívnej komunikačnej klímy.

O Vygotského „zóne najbližšieho vývoja“ a ďalších konštruktivistických pohľadoch na poznávací proces ste sa viac mohli dozvedieť v module Moderná škola 2 – Východiská a inšpirácie [6].

Uvedieme príklad prvej vyučovacej hodiny programovania v Logu v 5. ročníku základnej školy. V príprave na vyučovaciu hodinu

- si ujasníme, aké vstupné vedomosti a zručnosti môžeme predpokladať,
- stanovíme si realistické ciele,
- premyslíme priebeh hodiny tak, aby sme vytvorili vhodné podmienky pre konštruktívne učenie sa,
- pripravíme si aj námety navyše, aby sme vedeli pohotovo reagovať na rôzne situácie vzniknuté v triede.

Prvá hodina – ovládanie pohybu korytnačky

V predmete informatická výchova na prvom stupni základnej školy sa žiaci zoznámia s počítačom a osvoja si najbežnejšie činnosti vykonávané na počítači prostredníctvom aplikácií určených špeciálne pre deti. V tematickom okruhu *Postupy, riešenie problémov, algoritmické myslenie* sa učili vykonávať činnosti podľa jednoduchých slovných a obrázkových návodov (postupností príkazov) a zostavovať podobné návody. Majú skúsenosti aj so zadávaním ikonických príkazov na ovládanie personifikovaného procesora (postavičky) v nejakom detskom programovacom prostredí. Tieto skúsenosti a zručnosti sú dobrým základom pre programovanie v Imagine Logu. Na prvej hodine môžeme predpokladať tieto vstupné vedomosti:

Vstupné vedomosti

- žiaci vedia editovať krátke texty na počítači (vkladať a mazať znaky, pohybovať sa po texte kurzorovými šípkami),
- vedia používať myš (presúvať kurzor myši na dané miesto, klikať a dvojklikať tlačidlom myši),
- vedia vyberať príkazy z ponúk (menu),
- vedia otvoriť/uložiť súbor a prechádzať štruktúrou priečinkov,
- vedia vytvárať postupnosti ikonických príkazov na ovládanie nejakej postavičky (ťahaním alebo klikaním myšou) a pomocou nich riešiť rôzne problémy – zadania.

Na prvej hodine sa žiak musí zoznámiť s novým prostredím, v ktorom bude programovať. Preto nezahlcujme žiakov množstvom nových príkazov, aj keď sú jednoduché. Sformulujme podrobne špecifické ciele prvej vyučovacej hodiny, zistíme, že ich nie je málo, aj keď sa žiaci naučia len tri príkazy jazyka Logo:

Ciele

Vždy používajme správnu terminológiu, lebo žiaci si ju osvojujú počúvaním a napodobňovaním. Na hodine často budeme hovoriť „stránka“, „plocha výpisov“, „príkazový riadok“. Potom si tieto termíny žiaci osvoja prirodzene, netreba sa ich zvlášť učiť.

- zoznámiť sa s prostredím Imagine: stránka, korytnačka, príkazový riadok, plocha výpisov,
- naučiť sa ovládať pohyb korytnačky príkazmi `dopredu`, `vľavo`, `vpravo` a ich skratkami `do`, `vl`, `vp` v príkazovom riadku,
- vedieť používať pomôcky pravítko a kompas na zadávanie parametrov príkazov `dopredu`, `vľavo`, `vpravo` a získať predstavu o veľkosti parametrov a ich efekte na vykonaný príkaz,
- vedieť vyvolať miestnu ponuku stránky kliknutím pravým tlačidlom myši a načítať pozadie stránky pomocou príkazu z miestnej ponuky.

Predpokladom úspešného dosiahnutia cieľov je, aby žiak mal záujem (*motiváciu*) učiť sa. Preto všetky aktivity, ktoré budeme na hodine robiť, musia byť *zmysluplné*. Napríklad nemá zmysel vysvetľovať žiakom systematicky prvky prostredia Imagine, ktoré na hodine vôbec nevyužijú. Poznatky by boli len formálne, nepodložené skúsenosťou, zabudnuté v krátkom čase.

Aktivity, ktoré plánujeme na hodine so žiakmi robiť, majú byť primerané ich schopnostiam a vstupným vedomostiam. Musia byť v *zóne najbližšieho vývoja*, t.j. žiaci ich vedia realizovať samostatne alebo s pomocou učiteľa. Ak učiteľ zadá úlohu mimo zóny najbližšieho vývoja, žiak ju nevie vyriešiť ani s pomocou, učiteľ predvedie hotové riešenie a žiakovo poznanie bude formálne.

Priebeh hodiny

Žiaci vedia spúšťať aplikácie dvojkliknutím na ikonu aplikácie na pracovnej ploche. Spustíme program Imagine a predstavíme korytnačku.

Vyskúšame ovládať korytnačku pomocou príkazu `dopredu` v príkazovom riadku.

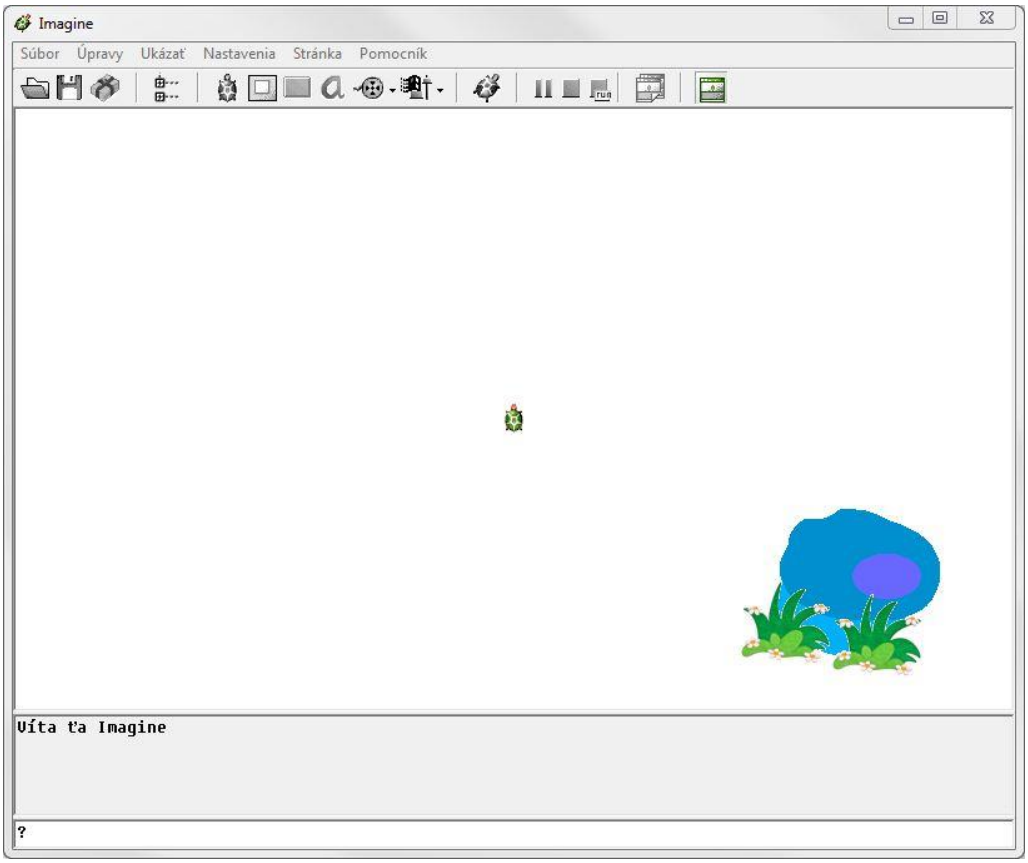
```
? dopredu 50
```

Korytnačka za sebou zanecháva stopu – tenkú čiernu čiaru. Vyskúšame iné hodnoty parametra príkazu `dopredu`.

Žiaci budú motivovaní posúvať korytnačku aj do iných smerov. Predstavíme príkazy `vľavo`, `vpravo` s parametrom `90`. Žiaci môžu experimentovať so zadávaním príkazov `dopredu`, `vľavo`, `vpravo` s rôznymi vstupmi. Prezradíme im skratky na uľahčenie zapisovania príkazov, eliminuje sa tým množstvo syntaktických chýb. Žiaci experimentovaním zistia, že:

- keď korytnačka zíde zo stránky, objaví sa na opačnom konci stránky,
- keď zadajú nesprávne príkaz, v ploche výpisov sa vypíše správa o chybe, príkaz treba napísať ešte raz správne,
- keď zabudnú napísať príkazu vstup, otvorí sa pomôcka pravítko alebo kompas.

Motivácia skúšať, ako korytnačka reaguje na príkazy zadávané v príkazovom riadku, netrvá dlho. Ako novú motiváciu ponúkneme žiakom navigovanie korytnačky do cieľa.



Obrázok 3.1: Pozadie s cieľom pre korytnačku

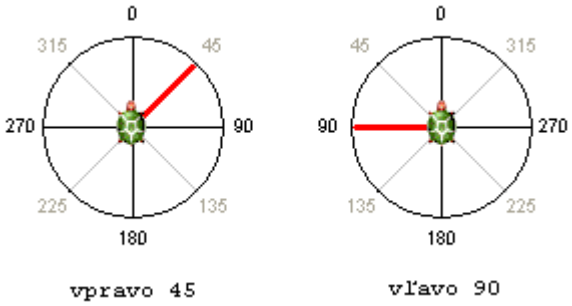
Dvojkliknutím pravým tlačidlom myši do stránky žiaci otvoria miestnu ponuku, z ktorej si vyberú príkaz *Pozadie zo súboru*. Vyberú súbor, ktorý im učiteľ pripravil. Na obrázku by mal byť jednoduchý cieľ umiestnený mimo domovskej pozície a smeru korytnačky.

Pri navigácii korytnačky do cieľa žiaci využijú svoje skúsenosti z experimentovania na odhad veľkostí vstupov alebo použijú pomôcky. Úlohu si zopakujú s novým pozadím, v ktorom môžu byť ďalšie podmienky na pohyb korytnačky (napr. ísť do domčeka po ceste, plávať do jazierka v potoku). Pozadie prispôsobíme domovskej pozícii korytnačky v strede obrázka.

V závere hodiny zopakujeme so žiakmi príkazy, ktoré sa naučili. Môžu si ich zapísať

Zápis do zošita pomáha žiakom identifikovať podstatné učivo. Pre aktívneho rodiča je informáciou, čo sa dieťa učilo v škole, pre chýbajúceho žiaka informáciou, čo sa dialo počas jeho neprítomnosti v škole.

do zošita spolu so skratkami. Zhrnieme, čo sme zistili o vstupoch príkazov. Píšu sa za meno príkazu oddelené medzerou. V príkaze `dopredu` predstavujú počet bodov obrazovky. Žiaci by mali vedieť zhruba odhadovať veľkosti vstupov príkazu `dopredu`. Spýtajme sa ich, približne aká široká a vysoká je stránka. V príkazoch `vľavo`, `vpravo` predstavujú vstupy veľkosť uhla. Uhol a jeho veľkosť v stupňoch je pre žiakov nový pojem. Môžu si nakresliť do zošita kompas s veľkosťami uhlov pre príkazy `vľavo`, `vpravo`.



Obrázok 3.2: Kompasy s veľkosťami uhlov otáčania

Obmeny, rozšírenia

Podľa schopností žiakov môžeme pripraviť viac rôznych pozadí. Pozadia môžu obsahovať aj viac cieľov (pozri námety v module Didaktika programovania pre ZŠ 1 [9]). Určenie poradia cieľov môže predstavovať ďalšiu netriviálnu úlohu. Ak sú ciele umiestnené tak, že korytnačka pri ich navštevovaní nakreslí zaujímavý obrázok, je to motivácia na jeho uchovanie – uloženie pozadia stránky do súboru.

Ako precvičovaciu aktivitu na nasledujúcu hodinu môžeme pripraviť viacstránkový projekt s načítanými pozadiami stránok a nastavenými domovskými pozíciami korytnačiek na stránkach. Žiaci sa naučia prepínať medzi stránkami prepínačmi v hlavnom paneli prostredia Imagine.

Pozadia môžu pripraviť aj žiaci na hodine informatiky venovanej práci s grafikou a spolupodieľať sa na tvorbe malej didaktickej aplikácie.

Diskusia 3.1	Zhodnoťte návrh vyučovacej hodiny z pohľadu konštruktivistického prístupu k vyučovaniu. Pridajte vlastné nápady na realizáciu prvej hodiny z programovania.

Hodnotenie

V predmete informatika na základnej škole sa žiaci klasifikujú. Za prácu a dosiahnuté výsledky sa im udeľuje známka. Cieľom preverovania vedomostí a hodnotenia však nie je len klasifikácia žiakov.

Funkcie kontroly vyučovacieho procesu:

- diagnostická – zistenie stavu vedomostí a zručností žiakov; je dôležité nielen kvôli klasifikácii žiaka, ale tiež pre plánovanie ďalšieho vyučovania,
- motivačná – preverovanie vedomostí je jednou z foriem udržiavania a zvyšovania aktivity žiakov (vonkajšia motivácia – dostať dobré hodnotenie); nezabúdajme však, že pre konštruktívny poznávací proces je dôležitá vnútorná motivácia žiaka,
- informačná – pravidelná kontrola umožňuje priebežne dokumentovať výsledky žiaka a informovať o nich rodičov.

Diskusia 3.2	Akým spôsobom preverujete a hodnotíte vedomosti žiakov v tematickom okruhu Postupy, riešenie problémov, algoritmizácia v predmete informatika? Prezentujte svoje dobré aj zlé skúsenosti.

Príklady úloh na preverenie vedomostí z procedurálneho programovania – jedna korytnačka kreslí obrázky:

- nakresliť obrázok (na štvorčekový papier) podľa postupnosti korytnačích príkazov,
- napísať program (postupnosť príkazov alebo zadefinovať vlastný príkaz), ktorým korytnačka nakreslí obrázok podľa predlohy,
- nájsť chybu a opraviť program,
- doplniť program tak, aby sa vykreslil obrázok podľa predlohy,
- riešiť sériu úloh v malej didaktickej aplikácii.

Hodnotí sa správnosť riešenia (nakreslenie správneho obrázka).

Pri objektovom a udalosťami riadenom štýle programovania sa väčšinou nepíšu dlhé programy, ťažisko riešenia úlohy je v návrhu štruktúry projektu. Práca má často formu projektového vyučovania. Preto pri hodnotení môžeme používať kritériá ako pri projektoch [7]:

- či riešenie spĺňa požiadavky zadania,
- či žiaci použili správne postupy a poznatky,
- ako prejavili pri riešení tvorivosť, zmysel pre detaily, estetické cítenie.

Hodnotíme tiež stupeň samostatnosti. Ak má žiak problémy s návrhom štruktúry projektu, poradíme mu, aby mohol v práci pokračovať.

Úloha 3.1	Rozdeľte sa do skupín po 3-4 frekventantoch. Vyberte si jednu tému a pripravte návrh vyučovacej hodiny.
Úloha 3.2	Ako skupina prezentujte pred kolegami svoj návrh vyučovacej hodiny. Diskutujte o vyučovacích hodinách ostatných kolegov.

Čo sme sa naučili

Vyučovacie hodiny projektujeme tak, aby sa žiak učil konštruktívne, t.j. aktívne a s porozumením. Kostru konštruktívneho poznávacieho procesu tvorí:

motivácia – zbieranie skúseností – vznik poznatku – kryštalizácia

Cieľom preverovania vedomostí a hodnotenia nie je len klasifikácia žiakov. Plní aj úlohu diagnostickú, motivačnú a informačnú. Vonkajšia motivácia žiaka však nesmie byť jedinou motiváciou žiaka učiť sa.

4 Metodické listy

Opakovanie príkazov

Vyučovaniu opakovania príkazov v cykle by mala predchádzať skúsenosť s kreslením obrázkov s opakujúcimi sa časťami viacnásobným stláčaním tlačidla alebo opakovaným zadávaním rovnakých príkazov v príkazovom riadku.

Vstupné vedomosti

Predpokladáme, že žiaci majú skúsenosti:

- s písaním viacerých príkazov do príkazového riadka oddelených medzerou,
- s listovaním v histórii príkazov pomocou klávesových šípok,
- so zadávaním príkazov stláčaním tlačidiel,
- s načítaním a uložením pozadia stránky.

Ciele

Na prvej vyučovacej hodine s cyklami kreslíme jednoduché vzory s opakujúcimi sa motívmi z čiar a bodov. Cieľom je, aby žiaci vedeli:

- určiť opakujúcu sa sekvenciu príkazov v jednoduchom obrázku,
- zapísať opakovanie sekvencie príkazov v cykle `opakuj`,
- používať cyklus pri kreslení zložitejších obrázkov.

Priebeh hodiny

Naučme korytnačku kresliť prerušovanú čiaru. Prerušovaná čiara býva napríklad na vozovke, kde oddeľuje jazdné pruhy a autá cez ňu môžu prechádzať. Prerušovane tiež zvykneme vyznačovať rôzne myslené alebo neviditeľné čiary.

Pred kreslením nastavme hrúbku pera na 2 a ľubovoľnú farbu. Čiary budú výraznejšie.

Prerušovaná čiara sa skladá z čiar a medzier. Nakreslíme jednu čiaru s položeným perom a jednu medzeru so zdvihnutým perom:

```
? pd do 10 ph do 10
```

Opakovaným zadávaním tohto riadka príkazov nakreslíme prerušovanú čiaru. Na opakované zadávanie príkazov použijeme listovanie v histórii príkazov pomocou šípky hore. Podobne žiaci nakreslia bodkovanú a bodkočiarkovanú čiaru.

Obrázok 4.1: Čiarkovaná, bodkovaná a bodkočiarkovaná čiara

Úloha pre žiakov: Položte pero korytnačky dolu. Napíšte do príkazového riadka a niekoľkokrát zopakujte príkazy:

```
? do 50 vp 90
```

Obrázok 4.2: Kreslenie štvorca

Čo sa vykreslí? Koľkokrát treba zopakovať príkazy v riadku, aby sa nakreslil štvorec?

Zavedieme príkaz `opakuj` na nakreslenie štvorca:

```
? opakuj 4 [do 50 vp 90]
```

Venujeme pozornosť syntaxi príkazu – medzerám a písaniu hranatých zátvoriek.

Príkaz cyklu si žiaci precvičia na kreslení čiarkovanej a bodkovanej čiary. Úloha: Nakreslite čiarkovanú čiaru so šiestimi čiarkami.

Žiaci zapisujú známy postup na nakreslenie prerušovanej čiary pomocou príkazu `opakuj`.

```
? opakuj 6 [pd do 10 ph do 10]
```

Akú dlhú vzdialenosť prešla korytnačka? (120) Odpoveď overíme „zacúvaním“ korytnačky na začiatok čiary. Nakreslite bodkovanú čiaru rovnakej dĺžky.

Určujú počet opakovaní.

```
? opakuj 12 [bod 4 do 10]
```

Prečo je počet opakovaní dvakrát väčší? (Korytnačka sa v cykle posúva o 10 bodov, predtým o 20 bodov)

Naučme korytnačku kresliť zubatú čiaru. Určiť postupnosť príkazov, ktorá sa má opakovať, je ťažšie. Na tabuli žiaci vyznačia farebne opakujúcu sa časť, učiteľ dokreslí pozíciu korytnačky na začiatku a na konci tela cyklu.

Hľadajú opakujúcu sa časť (telo cyklu).

Iná zubatá čiara (zložitejšia):

Obrázok 4.3: Zubatá čiara s vyznačeným invariantom

V závere hodiny použijeme príklady na opakovanie príkazov v cykle, ktoré žiaci na hodine programovali, v miniprojekte: Navrhnite si vrecko na džínsy.

Do pozadia stránky načítame džínsovú textúru. Učiteľ môže pripraviť viac textúr rôznych farieb (dajú sa nájsť na internete). Žiaci si do džínsového pozadia nakreslia štvorcové vrecko a rôznymi stehmi ho vyzdobia (obr. 4.4). Hotové obrázky si uložia.

Obrázok 4.4: Vrecká na džínsy

Obmeny, rozšírenia

Pri kreslení vrecka na džínsy môžu šikovnejší žiaci využiť ďalšie lomené alebo prerušované čiary (stehy), kresliť oblúky (pridajú do tela cyklu príkaz na otočenie korytnačky o malý uhol) alebo dozdobiť vrecko ďalšími ozdobami.

Ďalšie námety na kreslenie jednoduchých obrázkov s cyklom: štvorec, trojuholník, domček, schody do domčeka, plot pri domčeku, cesta s prerušovanou čiarou.

Vlastné príkazy

Motiváciou na písanie vlastného príkazu môže byť:
- viacnásobné kreslenie jednoduchých samostatných obrázkov,
- programovanie stavebných dielov na nakreslenie zložitejších obrázkov.

Použitie príkazu ako stavebného dielu v zložitejšom obrázku si vyžaduje schopnosť analyzovať obrázok a rozpoznať v ňom stavebné diely. Preto odporúčame začať programovaním príkazov pre samostatné obrázky – celý obrázok sa nakreslí jedným vlastným príkazom.

Vstupné vedomosti

Predpokladáme, že žiaci majú skúsenosti s programovaním v príkazovom riadku a vedia:

- editovať viacriadkový text,
- používať jednoduché príkazy korytnačej grafiky: `dopredu`, `vzad`, `vľavo`, `vpravo`, `bod`, `peroHore`, `peroDolu`, `nechFarbaPera`, `nechHrúbkaPera` a ich skratky,
- pečiatkovať s použitím príkazu `odtlačObrázok`,
- opakovať príkazy v cykle `opakuj`,
- načítať a uložiť pozadie stránky.

Ciele

Na prvej vyučovacej hodine s vlastnými príkazmi sa naučíme definovať vlastný príkaz na nakreslenie jednoduchého obrázka a jeho užitočnosť demonštrujeme viacnásobným vykresľovaním na rôznych pozíciách. Cieľom je, aby žiaci vedeli:

- editovať vlastný príkaz: otvoriť editor príkazom `uprav`, skratka `up`, osvojiť si syntax vlastného príkazu (hlavička, koniec),
- použiť vlastný príkaz v príkazovom riadku samostatne a v cykle,
- uložiť projekt.

Priebeh hodiny

Na úvod zadáme úlohu, ktorú žiaci vedia samostatne vyriešiť zadávaním príkazov do príkazového riadka: Nakreslite strom a vedľa neho lavičku (obr. 4.5).

Preskúmanie plochy výpisov je užitočné, lebo:
- žiaci si rozvíjajú schopnosť čítať program, interpretovať príkazy v mysli,
- spočítaním príkazov na nakreslenie stromu získajú motiváciu nahradiť ich jedným príkazom,
- zopakujú si postup kreslenia pred editovaním nového príkazu.

Obrázok 4.5: Jednoduchý strom a lavička

Motiváciou na napísanie vlastného príkazu je nakresliť viac stromov a lavičiek v parku. V ploche výpisov si pozrieme, koľko príkazov sme použili na nakreslenie stromu. Aby sme ich nemuseli zadávať znovu, naučíme korytnačku nový príkaz, ktorým nakreslí strom naraz. Nový príkaz sa bude volať `strom`. Do príkazového riadka napíšeme

```
? uprav "strom
```

Otvorí sa editor, v ktorom je pripravený začiatok a koniec príkazu `strom`. V mieste blikajúceho kurzora napíšeme príkazy na nakreslenie stromu. Editovanie ukončíme stlačením tlačidla OK.

Každý žiak si edituje svoj program sám. Program je jednoduchý a v úvode hodiny si ho každý naprogramoval v príkazovom riadku.

Nový príkaz vyskúšame zadaním v príkazovom riadku:

```
? strom
```

Vedľa neho nakreslime druhý strom. Korytnačku premiestnime so zdvihnutým perom a znovu použijeme príkaz `strom`. Ak sa niektorým žiakom druhý strom nevykreslí, môžeme vyvolať diskusiu: Komu sa nevykreslil druhý strom? Komu sa vykreslil druhý strom? Prečo? Žiaci si navzájom poradia, že pred kreslením treba položiť pero. Žiakom poradíme, aby príkaz `pd` napísali do definície príkazu `strom`, tak naň nezabudnú.

Editor príkazov znovu otvoríme príkazom `uprav "strom` alebo skratkou `up "strom` v príkazovom riadku. Žiakom tiež poradíme, aby príkaz napísali tak, že korytnačka skončí kreslenie v tej istej pozícii, ako začala. Potom bude kresliť strom, ako keby ho pečiatkovala a vôbec sa pritom nepohla z miesta. Príkaz môže vyzerať takto:

```
viem strom
 pd
 nechfp "hnedá
 nechhp 15
 do 40
 nechfp "zelená4
 bod 50
 ph
 vz 40
koniec
```

Žiaci majú skúsenosti s pečiatkovaním. Naprogramovať príkaz ako pečiatku je dobrá motivácia na ukončenie príkazu v rovnakej pozícii korytnačky ako na začiatku.

Editovanie nového príkazu si žiaci zopakujú na príkaze `lavička`. Rovnako ako pri strome začneme položením pera a skončíme v začiatočnej pozícii korytnačky so zdvihnutým perom ako pri pečiatkovaní.

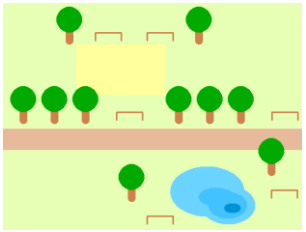
```
? uprav "lavička
```

```
viem lavička
 pd
 nechfp "hnedá
 nechhp 3
 do 15 vp 90
 do 50 vp 90
 do 15 vp 90
 ph
 do 50 vp 90
koniec
```

Zápis do zošita

```
? lavička
```

Nové príkazy viacnásobne použijeme na vhodnom pozadí. Jednoduché pozadie parku s pieskoviskom, jazierkom a chodníkom si môžu rýchlo nakresliť žiaci sami alebo ho pripraví učiteľ Okolo rovného chodníka môžeme kresliť stromy a lavičky pravidelne v cykle.



V závere hodiny zhrnieme podstatné učivo: príkaz `uprav` (`up`) na otvorenie editora vlastných príkazov, príklad definície vlastného príkazu (`strom` alebo `lavička`) a jeho použitie v príkazovom riadku si žiaci môžu zapísať do zošita.

Nové príkazy a nakreslené pozadie si môžeme uložiť v projekte pomocou príkazu *Uložiť projekt* z hlavnej ponuky *Súbor* alebo stlačením nástroja  v paneli nástrojov.

Obmeny, rozšírenia

Prácu na projekte *Park* môžeme rozšíriť o naprogramovanie ďalších obrázkov vhodných do parku - lampa, kvet, dieťa - podľa schopností žiakov alebo ako precvičovanie na ďalšej hodine. Žiaci môžu presúvať korytnačku na stránke pomocou príkazu `nechPoz ?` a podľa pozície, na ktorej sa korytnačka ocitne, vykresliť strom, lavičku, iný objekt alebo nič. Je to propedeutika podmieneného vykonávania príkazov.

Podobné motivačné témy na písanie vlastných príkazov sú v študijnom materiáli k modulu Didaktika programovania pre ZŠ 1 [9].

Korytnačka reaguje na udalosť *priKliknutí*

Vstupné vedomosti

- žiaci vedia riadiť pohyb korytnačky zadávaním príkazov do príkazového riadku,
- poznajú príkazy na nastavenie farby a hrúbky pera, farbu a hrúbku vedia zvoliť aj náhodne,
- používali príkaz `bod`,
- vedia pridať do projektu ďalšiu stránku a umiestniť na ňu korytnačku.

Ciele

Cieľom vyučovacej hodiny je, aby žiaci vedeli:

- rozlišovať medzi pojmami udalosť a reakcia na udalosť,
- zobraziť rodný list korytnačky a naprogramovať krátku reakciu na udalosť `priKliknutí`,
- použiť príkaz `nechPoz ?` na presun korytnačky na náhodne zvolené miesto na stránke,
- zdôvodniť, prečo korytnačka nereaguje na udalosť `priKliknutí`.

Priebeh hodiny

Motivácia

Hodinu môžeme začať krátkou hrou približujúcou princíp udalosťami riadeného programovania. Učiteľ napíše na tabuľu slová **udalosť** a **reakcia** a niekoľko príkladov udalostí z bežného života (napr. stretnem kamaráta, začne pršať, zazvoní budík). Vyzve deti, aby doplnili vhodné reakcie (napr. pozdravím, otvorím dáždnik, vyskočím z postele). Každé z detí môže vymyslieť vlastnú dvojicu udalosť a reakcia.

Udalosť

Napadlo veľa snehu.

Reakcie

Peťo sa hneď ráno vyberie sánkovať. Elenka postaví na dvore snehuliaka. Paľko si urobí teplý čaj a číta knihu.

Mohli by dvaja na rovnakú udalosť zareagovať rôznym spôsobom? Uvedieme príklad udalosti, deti doplnia možné reakcie.

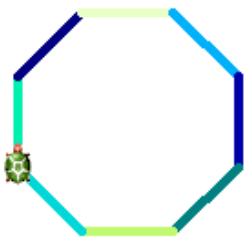
Od hry prejdeme plynule k tomu, že aj korytnačka, ktorá žije na stránke prostredia Imagine Logo, môže reagovať na rôzne udalosti. Musíme ju to však naučiť. Naprogramujeme, čo má urobiť, keď na ňu klikneme. Priamo v programovacom prostredí predvedieme:

- že korytnačka zatiaľ nereaguje, klikáme na ňu zbytočne,
- ako sa zobrazí rodný list korytnačky,
- kam sa zapisujú príkazy, ktoré má korytnačka vykonať, keď nastane udalosť `priKliknutí` (t. j. keď na ňu klikneme ľavým tlačidlom myši),
  - zapíšeme jeden príkaz:

    priKliknutí: do 50

  - zavrieme rodný list kliknutím na tlačidlo OK,
  - klikáme na korytnačku, tento krát už poslúcha, reaguje na udalosť `priKliknutí` tak, že pobehne o 50 krokov dopredu.

Namiesto rodného listu korytnačky žiaci niekedy omylom zobrazia rodný list stránky.



Žiaci napodobnia učiteľa a zistia, že korytnačka reaguje tak, ako ju to práve naučili. Postup si viackrát zopakujú dopĺňaním ďalších príkazov do reakcie na udalosť `priKliknutí`, napr.:

```
do 50 vp 45
nechFp ? do 50 vp 45
```

Zbieranie skúseností

V ďalších úlohách môžeme kresliť na stránke obrázky z bodiek. Na novú stránku vložíme korytnačku. Príkaz `bod` pripomenieme formou hádanky. Aký obrázok vytvoríme klikaním na korytnačku, keď má reakciu na udalosť `priKliknutí` naprogramovanú takto? Svoju odpoveď overte!

```
priKliknutí: nechFp ? bod 30 vp 90 do 30 vl 90
```

Deti pridávanie nových stránok a nových korytnačiek zaujme. Vysvetlíme príkaz `nechPoz ?` a necháme žiakov experimentovať. Nech korytnačka „vyfukuje" na náhodných miestach na stránke bublinky:

Keďže každú ďalšiu korytnačku umiestnia žiaci na novú stránku, nie je potrebné, aby sa zaoberali ich oslovovaním. Môžu sa sústrediť len na programovanie reakcie na udalosť `priKliknutí`.

- rovnakej farby aj veľkosti,
- rôznej farby a rovnakej veľkosti,
- rôznej farby a rôznej veľkosti.

Žiaci si všimnú, že okrem bubliniek sa kreslia na stránke aj spojnice medzi bublinkami. Navedieme ich na správne riešenie (korytnačke treba zdvihnúť pero).

V závere hodiny zhrnieme, čo nové sme sa o korytnačke dozvedeli. Žiaci si do zošita zapíšu stručné poznámky:

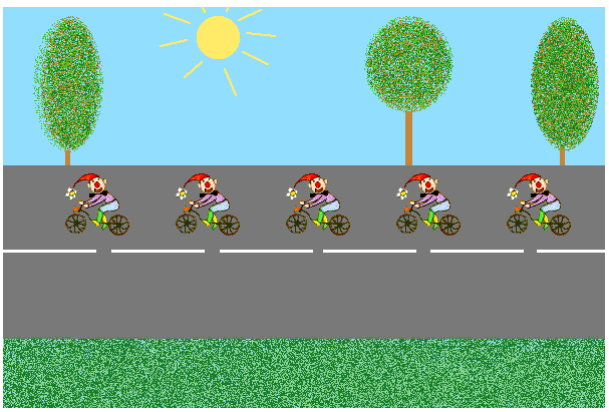
Môžeme sa tiež zaoberať otázkou, či je možné, aby korytnačka zabudla, ako má na udalosť `priKliknutí` reagovať. Riešenie by mali vedieť sformulovať aj žiaci: „vymažeme príkazy, ktoré sme do reakcie napísali".

- korytnačku môžeme naučiť reagovať na udalosti, napr. na udalosť `priKliknutí`,
- reakcia na udalosť sa programuje v rodnom liste korytnačky,
- korytnačku vieme príkazom `nechPoz ?` premiestniť na náhodne zvolené miesto na stránke.

Obmeny, rozšírenia

Zatiaľ sme v reakciách na udalosť používali len základné korytnačie príkazy. Na nasledujúcej hodine môžeme riešiť úlohy podobné úlohám z učebnice [3], v ktorých sa v reakcii na udalosť `priKliknutí` používajú vlastné príkazy (napr. domček, stromček).

Vhodnou a jednoduchou reakciou na udalosť `priKliknutí` je aj odtlačenie obrázka (príkazom `odtlačObrázok`). Opečiatkujme na stránke s pozadím obsahujúcim cestu niekoľko obrázkov cyklistov:



Obrázok 4.6: Námet na miniprojekt

Korytnačka vymaľováva

Vstupné vedomosti

- žiaci vedia naprogramovať reakciu korytnačky na základnú udalosť `priKliknutí`,
- pamätajú si, že korytnačke, ktorú chceme ťahať po stránke, musíme v rodnom liste na karte *Tvar* zapnúť *Automatické ťahanie*,
- v rodnom liste vedia zobraziť kartu *Udalosti* a pridať (odstrániť) reakciu na udalosť `priŤahaní`,
- vedia na stránku umiestniť viac korytnačiek, zmeniť im tvar,
- dokážu vybrať pozadie stránky zo súboru,
- pri práci v grafickom editore používali nástroj na vypĺňanie uzavretých oblastí.

Ciele

- žiak vie povedať, kedy vznikne udalosť `priĽavomDolu` a kedy `priĽavomHore`,
- dokáže pridať korytnačke reakcie na tieto udalosti,
- vie použiť príkazy `nechFarbaVýplne` (`nechFv`) a `vyplň`,
- vie vysvetliť, čo je výsledkom operácie `farbaBodu`.

Priebeh hodiny

Motivácia

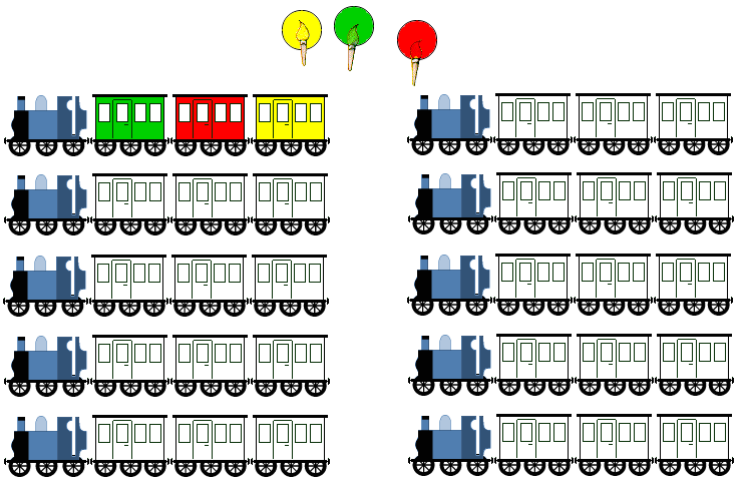
Hodinu môžeme začať rozhovorom. Spýtame sa detí, či v škôlke vymaľovávali obrázky, aké farbičky používali a pod. Vedeli by nájsť obrázok na vymaľovanie na internete?

Súbory s pozadím ako aj s tvarmi korytnačiek sú súčasťou inštalácie prostredia Imagine Logo. Ide o námet na projekt z učebnice [3].

Vymaľovávať môžeme aj pomocou korytnačky. Obrázok na vymaľovanie si ľahko nastavíme ako pozadie stránky. Žiakom povieme, aby si

- na stránke nastavili pozadie zo súboru *vláčiky.bmp*,
- pripravili na stránke tri korytnačky a nastavili im tvar žltého, zeleného a červeného štetca,

Na obrázku ukážeme požadovaný výsledok:



Obrázok 4.7: Stránka s pozadím a tromi korytnačkami s tvarom štetca

Skôr ako budeme pokračovať, overíme, či sa každá korytnačka-štetec dá ťahať. Žltým štetcom budeme vyfarbovať vagóny vláčika nažlto, zeleným nazeleno, červeným načerveno. Štetec potiahneme na vagón a ten sa zafarbí.

Ako sa volá udalosť, ktorá vzniká, keď korytnačku ťaháme po stránke? Ako prebieha

ťahanie? Kedy začína a kedy končí? Necháme niektorého zo žiakov, aby priamo na myši ukázal, že sa najprv stlačí ľavé tlačidlo myši, potom sa tlačidlo drží a nakoniec pustí. Ťahanie skončí, keď pustíme ľavé tlačidlo myši. Nastane udalosť, ktorú voláme `priĽavomHore`.

Ako prvú naučíme vymaľovávať korytnačku s tvarom žltého štetca. Keď korytnačku pustíme, má vyplniť vagón žltou farbou. Ako nastavíme žltú farbu? Žiaci pravdepodobne spomenú príkaz `nechFp`. Vysvetlíme, že týmto príkazom nastavujeme farbu pera. Pri vypĺňaní treba nastaviť farbu výplne. Príslušný príkaz napíšeme na tabuľu (v úplnej i skrátenej forme):

```
nechFarbaVýplne "žltá

nechFv "žltá
```

Žiaci pridajú korytnačke reakciu na udalosť `priĽavomHore`, zapíšu príkaz na zmenu farby výplne. Zistia však, že korytnačka vagón farbou nevyplnila. Ako to? Zabudli sme to korytnačke prikázať. Zapíšeme na tabuľu príkaz `vyplň` a vysvetlíme jeho význam. Žiaci opravia reakciu na udalosť a otestujú žltý štetec. Analogicky naprogramujú reakcie zvyšných dvoch korytnačiek.

Zbieranie skúseností

Žiakom zadáme konkrétnu úlohu: vymaľujte vagóny vláčikov tak, aby žiadne dva vláčiky neboli rovnaké. Koľko rôznych vláčikov je možné vytvoriť? Svoje riešenie (obrázok s vymaľovanými vláčikmi) si môžu uložiť do súboru na disk.

V ďalších úlohách budeme vagóny vláčikov vyfarbovať inak:

- položíme na stránku novú korytnačku, ktorá bude vyberať pred vyplnením vagóna farbu výplne náhodne
  - pomocou príkazu `nechFv ?`
  - zo zoznamu vymenovaných farieb, napr. `nechFv ?prvok [zelená5 zelená7 zelená8 zelená10 zelená12]`
- položíme na stránku ešte jednu korytnačku, ktorá bude vyfarbovať vagóny takto:
  - zistí si, akú farbu má vagón, na ktorom stojí na začiatku ťahania
  - rovnakou farbou vyplní vagón, na ktorom ju pustíme

Posledná z uvedených úloh si vyžaduje naprogramovanie reakcie na udalosť `priĽavomDolu`. Vysvetlíme, že korytnačka si vie zistiť, akú farbu má bod, na ktorom stojí. Reakciu na udalosť `priĽavomDolu` napíšeme na tabuľu:

```
nechFv farbaBodu
```

V závere hodiny si žiaci zapíšu nové udalosti a príkazy, ktoré sa naučili, do zošita.

Obmeny, rozšírenia

Deti môžu vymaľovávať aj iné obrázky, ktoré pripraví učiteľ. Do pozadia si ich môžu načítavať pomocou tlačidiel. Obrázok si môžu vyhľadať aj na internete. Ak je potrebných viac štetcov rôznych farieb, stačí korytnačku-žltý štetec naklonovať a upraviť jej tvar prefarbením štetca v editore LogoMotion.

Obrázok 4.8: Vyhľadávanie obrázkov na vymaľovanie pomocou Google

Náročnejšou verziou vymaľovánky je používanie jediného štetca, ktorým naberáme farbu z pripravenej palety (v reakcii na udalosť `priLavomDolu`). Stránku je potrebné rozdeliť na dve časti (časť s obrázkom a časť s paletou) a v reakcii na udalosť `priLavomHore` otestovať x-ovú súradnicu korytnačky.

priľavomDolu*:* `nechFv farbaBodu`

Korytnačku môžeme pustiť nad obrázkom alebo nad paletou. V druhom prípade nesmieme vypĺňať, zničili by sme si paletu farieb.

priľavomHore*:* `ak xSúr<250 [vyplň]`

Obrázok 4.9: Námet na miniprojekt

Odtláčanie tvaru korytnačky

Vstupné vedomosti

- poznajú význam `?` ako zástupného symbolu pre náhodný výber z povolenej množiny vstupov,
- vedia vložiť tlačidlo a nastaviť mu reakciu na udalosť `priZapnutí`,
- vedia korytnačkou odtláčať obrázok príkazom `odtlačObrázok`,
- poznajú udalosť korytnačky `priŤahaní` a vedia ju použiť v projektoch,
- vedia nastaviť domovskú pozíciu korytnačky v rodnom liste a poznajú príkaz `domov`,
- žiaci vedia zmeniť tvar korytnačky v rodnom liste.

V tomto metodickom liste si predstavíme tému pečiatkovanie, ktorej sa venuje učebnica [3] na strane 30. Pečiatkovanie patrí medzi veľmi populárne témy, veľký priestor sa mu venuje aj v rámci Informatickej výchovy na I. stupni ZŠ v prostredí RNA.

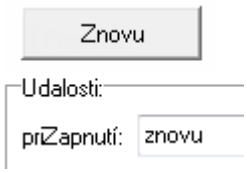
Ciele

- žiak vie použiť príkaz `odtlač` na odtlačenie tvaru korytnačky,
- pozná udalosť korytnačky `priĽavomHore`.

Priebeh hodiny

Na začiatku diskutujeme so žiakmi o rôznych typoch projektov, kde bolo ich úlohou pri ťahaní korytnačky kresliť na stránku rôzne obrazce, ako domy, kvety, hviezdičky a pod. V diskusii by sme mali dospieť k tomu, že každý obraz bol vždy vykreslený pomocou základných príkazov, výsledok bol teda zložený len z čiar a oblúkov.

Na stránku môžeme vopred umiestniť tlačidlo *Znovu*, ktoré vyčistí stránku:

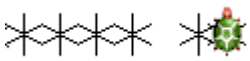


Žiaci si otvoria projekt, zdvihnú korytnačke pero a nastavia automatické ťahanie. Zopakujeme si udalosť `priŤahaní`. Ako reakciu na ťahanie zadáme vykonať tieto príkazy:

```
bod 20 čakaj 100
```

Korytnačka kreslí bod s priemerom 20 bodov.

```
pd opakuj 6 [do 10 vz 10 vp 60] ph čakaj 200
```



Korytnačka kreslí snehovú vločku.

Pri ťahaní môžeme tiež odtláčať obrázok:

```
odtlačObrázok "kvapka čakaj 100
```

Obrázkom na odtláčanie môže byť aj tvar korytnačky - zelená korytnačka. Napíšeme na tabuľu novú postupnosť príkazov, v ktorej zavedieme príkaz `odtlač`.

```
odtlač čakaj 100
```

Nechajme žiakov nech vyskúšajú, ako sa správa korytnačka, ak pri ťahaní zároveň meníme jej smer:

```
odtlač čakaj 100 vl ?
```

Vysvetlíme, že korytnačka sa odtlačí presne tak, ako ju práve vidíme, ako je natočená. Vyzvime žiakov, aby zmenili korytnačke tvar, výber môžeme nechať na nich, a nech opäť korytnačku ťahajú po stránke.

Odtláčanie tvaru korytnačky pri ťahaní závisí od rýchlosti a presnosti ťahania, nehodí sa na odtláčanie tvarov v pravidelných útvaroch. Vyskúšajme postupne nasledujúce príkazy:

```
? opakuj 4 [do 20 odtlač]
? opakuj 6 [do 20 odtlač vz 20 vp 60]
? opakuj 4 [do 20 odtlač vp 90 odtlač]
? opakuj 6 [do 20 odtlač vp 60]
```

Každý žiak bude mať z predchádzajúcej úlohy pravdepodobne nastavený iný tvar korytnačky. Ak chceme nastaviť základný tvar, napíšeme

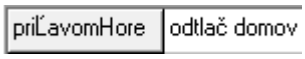
```
? nechTvar []
```

Nechajme žiakov chvíľu experimentovať a vyhlásme súťaž o najkrajšie zoskupenie korytnačiek.

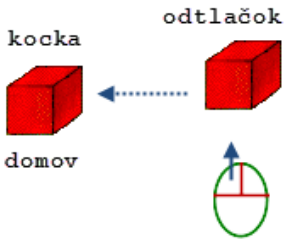
Obrázok 4.10: Rôzne zoskupenia korytnačiek

V diskusii so žiakmi vyhodnoťme, či už máme pečiatkovanie skutočne pod vlastnou kontrolou. Prezentujme žiakom projekt *6kocky.imp*, ktorý sa nachádza na CD vloženom do učebnice [3].

Upozorníme, že k odtlačeniu tvaru nedochádza počas ťahania, ale až na jeho konci. Spýtame sa, čo sa udeje na konci ťahania. Žiaci by mali prísť na to, že posledná činnosť, ktorá sa vykoná pred odtlačením, je pustenie tlačidla myši – udalosť `priĽavomHore`. Naučíme teda korytnačku odtlačiť sa na nami určenom mieste a vrátiť sa domov.



Zmeňme korytnačke tvar na *kocka.lgf*. Presuňme ju na ľavý okraj a nastavme jej novú domovskú pozíciu. Naučíme kocku vracať sa na túto domovskú pozíciu. Vyhľadáme udalosť `priĽavomHore`, kde napíšeme príkaz `domov`.



Vyzveme žiakov na upravenie programu tak, aby sa kocka vrátila domov, ale zároveň ostala odtlačená na mieste pustenia myši. Skontrolujeme riešenia a správne riešenie prezentujeme pred celou triedou.

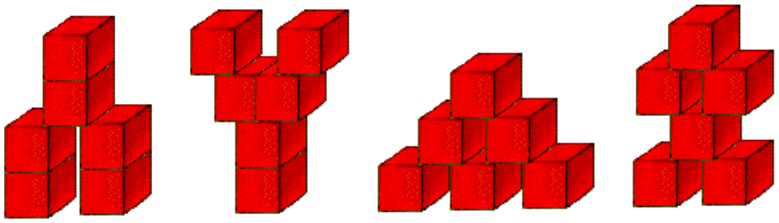
Zvyšok hodiny necháme žiakov z kociek vytvárať vlastné stavby.

Obmeny, rozšírenia

Pri vytváraní stavieb z kociek je potrebné myslieť aj na poradie pri odtláčaní kvôli viditeľnosti v trojrozmernom priestore. Môžeme im preto pripraviť rôzne predlohy, podľa ktorých budú stavby vytvárať.

Zámerne môžeme v predlohe zadať stavby, ktoré v bežnom živote odporujú fyzikálnym zákonom a zadať úlohu, aby ich žiaci našli.

Pridaním viacerých korytnačiek s rôznymi tvarmi môžeme vytvárať rôzne prostredia. Nechajme žiakov nech si navrhnú vlastný motív a v *LogoMotion* si k nemu nakreslia potrebné obrázky.



Obrázok 4.11: Predlohy stavieb z kociek

Procesy

Vstupné vedomosti

- žiaci chápu pojem cyklus a vedia ho zapísať a používať,
- poznajú a vedia správne použiť príkaz `čakaj`,
- vedia zmeniť tvar korytnačky,
- poznajú pojem záber a príkaz `nechZáber` a nastavenie Určovanie záberu,
- poznajú príkaz na zmenu absolútnej pozície korytnačky `nechPoz`,
- poznajú udalosť `priKliknutí`.

Ciele

- precvičiť si pojem pozícia korytnačky,
- precvičiť si prácu so zábermi,
- oboznámiť sa s pojmom proces a príkazom `každých`,
- vedieť zastaviť proces pomocou tlačidla v paneli nástrojov a príkazom `zastavVšetky`,
- pochopiť rozdiel medzi cyklom a procesom.

Priebeh hodiny

Na začiatku hodiny oboznámime žiakov s cieľmi hodiny. Vytvoríme hru, kde budeme myšou chytať utekajúceho „smajlíka“. Pomocou dataprojektora môžeme ukázať výsledok, aby si žiaci cieľ vedeli lepšie predstaviť.

Smejko.lgf je tvar s 11 zábermi, ktoré nezobrazujú smer. Preto v rodnom liste korytnačky zaškrtneme nastavenie *Určovanie záberu*.

Žiaci otvoria nový projekt. Korytnačke nastavia tvar *smejko.lgf*. Môžeme vyskúšať ako tvar reaguje na základné príkazy `dopredu` a `vľavo`. Zopakujeme aj posun a otočenie o náhodný smer. Pero korytnačky necháme zámerne dole, aby sme mohli sledovať cestu, ktorou korytnačka prešla. Aby sme príkazy nemuseli stále písať do príkazového riadka, prikážeme korytnačke, aby sa náhodne posunula a otočila 10 krát.

```
? opakuj 10 [do ? vl ?]
```

Viackrát vyskúšame spustiť tento cyklus. Vidíme, že pohyb nie je úplne náhodný, korytnačka sa točí vždy vľavo. Zmažme stránku a vyskúšajme, ako bude reagovať na zmenu polohy s využitím príkazu `nechPoz`.

```
? opakuj 10 [nechPoz ?]
```

Na základe cestičky, ktorú zanechala korytnačka, môžeme skonštatovať, že jej pohyb bol skutočne náhodný. Korytnačka skákala po celom papieri. Žiakom položíme otázky: „Ako zabezpečíme, aby korytnačka skákala dlhšie?“. Pravdepodobne dostaneme odpovede o zmene počtu opakovaní na vyššie číslo. Môžeme predviesť opakovanie s vyšším číslom, ďalšiu otázku smerujeme na časové rozpätie medzi skokmi: „Čo ak chcem korytnačku vidieť ako skáče alebo dokonca jej určiť rýchlosť skákania?“. Použijeme príkaz `čakaj`.

```
? opakuj 10 [nechPoz ? čakaj 500]
```

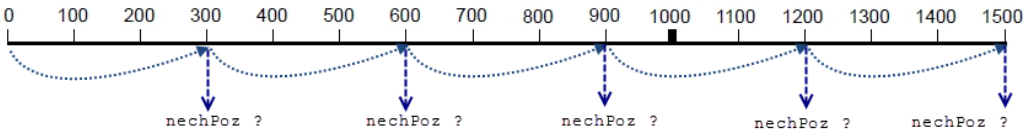
Stále sme však neumožnili korytnačke skákať vopred neurčený počet krát. Potrebovali by sme niečo ako nikdy nekončiaci cyklus. Zavedieme príkaz `každých`, ktorý spustí nekonečný proces. Pre žiakov môžeme použiť metaforu: Proces je „motorček“, ktorý opakovane, po uplynutí presne zadaného času, vykonáva určené príkazy.

Tlačidlá na prácu s procesmi:

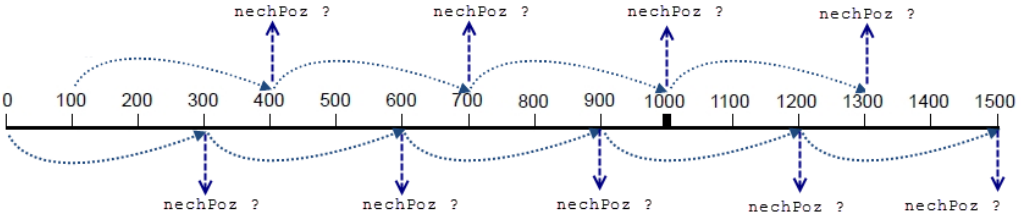
Ukážeme spustenie procesu a nezabudnime tiež ukázať, ako ho pozastaviť, znova spustiť a úplne zastaviť cez tlačidlá v paneli nástrojov a príkazom `zastavVšetky`.

```
? každých 300 [nechPoz ?]     ? zastavVšetky
```

Nechajme niekoľkokrát žiakov experimentovať spustiť a zastaviť proces, nabádajme ich k zmene čísla 300 na nejakú inú celočíselnú hodnotu. Až potom diskutujme o jej význame. Nakreslíme časovú os a vyznačíme priebeh procesu.



Určite sa stane, že niektorý zo žiakov spustí proces viackrát. Je vhodné upozorniť, že teraz už bežia dva procesy súčasne, odporúčame zakresliť to na tabuľu.



Nakoniec zopakujeme aký je rozdiel medzi cyklom a procesom (počet opakovaní, čas medzi opakovaniami, možnosť zastaviť vykonávanie).

Vyskúšame:

```
? nechZáber 1
? nechZáber ?
? nechZáber zaber+1
```

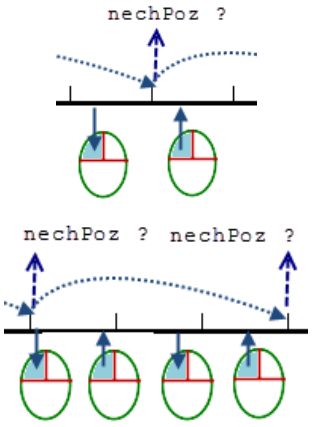
Smejko je na začiatku smutný, preto ho rozveselíme. Smejko totiž obsahuje niekoľko tváričiek, vyjadrujúcich rôzny emočný stav, do ktorých môžeme korytnačku „prezlieť“. Impulzom na prezliekanie môže byť napríklad kliknutie myšou. Zopakujeme pojem záber a jeho zmenu cez rodný list korytnačky a príkazom `nechZáber`.

Aby sme stále nemuseli písať príkaz do príkazového riadka, naučme smejka reagovať na kliknutie. Všimnite si, ako je smejko s každým kliknutím veselší.

priKliknutí: `nechZáber záber+1`

Takéto klikanie je ale veľmi jednoduché, skúsme si to teda sťažiť a to tak, že smejko sa bude snažiť utiecť.

```
? nechZáber 1 každých 1000 [nechPoz ?]
```



Obmeny, rozšírenia

Podľa schopností žiakov môžeme ukázať, že udalosť `priKliknutí` nemusí postačovať. Keďže sa skladá z dvoch udalostí `priĽavomDolu` a `priĽavomHore`, môže sa stať, že medzi týmito dvoma udalosťami nám smejko skočí na novú pozíciu. Preto nahradíme udalosť `priKliknutí` udalosťou `priĽavomDolu`.

Ďalším rozšírením môže byť skrývanie korytnačky (`skry`) pri kliknutí a zobrazenie na novej pozícii (`ukáž`). Ak je totiž žiak šikovný, môže počas jedného zobrazenia niekoľkokrát vyvolať udalosť pri kliknutí. Kliknúť môžeme len na zobrazený objekt.

Na zastavenie procesu skrývania môžeme využiť podmienený príkaz `ak`:

```
ak záber = 11 [
 zastavVšetky
]
```

```
ak záber = 11 [zastavVšetky]
```

K projektu sa opäť môžeme vrátiť v čase zavádzania pojmu *premenná*, kde budeme rátať počet kliknutí na smejka a počet presunutí smejka. Po ukončení hry môžeme vypísať štatistiku o percentuálnej úspešnosti chytača.

Čo sme sa naučili v tomto module

Zhrnutie

Vo vyučovaní programovania môžeme uplatniť projektové vyučovanie. Práca na komplexnom projekte (nie jednoduchej uzatvorenej úlohe) rozvíja celý rad kompetencií žiaka: tvorivo špecifikovať zadanie projektu, analyzovať problém a navrhovať rôzne riešenia, aplikovať vedomosti z viacerých tém, spolupracovať.

Pri plánovaní vyučovania sú dôležitými parametrami rozsah vyučovania a prerekvizity tém (obsahové a metodické). Podľa nich naplánujeme časovo-tematický plán.

Vyučovacie hodiny projektujeme tak, aby sa žiak učil konštruktívne, t.j. aktívne a s porozumením. Inak môže byť výsledkom žiakovho učenia sa formálne poznanie na úrovni zapamätania.

Preverenie výstupných vedomostí

Na úspešné absolvovanie modulu frekventanti v skupinách pripravia a prezentujú projekt vyučovacej hodiny.

Literatúra a použité zdroje

[1] Bezáková, D. a kol. Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Úvod do programovania. Bratislava : ŠPÚ, 2009. ISBN 978-80-89225-55-2

[2] Blaho, A., Kalaš, I. (1998). *Comenius Logo: Tvorivá informatika 1. časť*. 1. vyd. Bratislava : CL Group, 1998. ISBN 80-967999-0-8

[3] Blaho, A., Kalaš, I. (2007) *Tvorivá informatika: 1. zošit z programovania*. Bratislava : SPN – Mladé letá, 2007. ISBN 80-10-01223-7

[4] Hejný, M., Kuřina, F. (2001) *Dítě, škola a matematika. Konstruktivistické přístupy k vyučování*. Praha : Portál, 2001. ISBN 80-7178-581-4

[5] Hrušecká, A., Kalaš, I. (2006) *Programovanie v prostredí Imagine*. Bratislava : MPC, 2006. ISBN 80-8052-260-X

[6] Kabátová, M. a kol. (2009) *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Východiská a inšpirácie*. Bratislava : ŠPÚ, 2009. ISBN 978-80-89225-62-0

[7] Kalaš, I. a kol. (2010) *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Digitálne technológie a zásahy do vyučovania*. Bratislava : ŠPÚ, 2010. ISBN 978-80-8118-032-3

[8] Lovászová, G. a kol. (2010) *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Malé programovacie jazyky*. Bratislava : ŠPÚ, 2010. ISBN 978-80-8118-066-8

[9] Lovászová, G. a kol. (2011) *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Didaktika programovania pre ZŠ 1*. Bratislava : ŠPÚ, 2010. ISBN 978-80-8118-080-4

[10] Salanci, Ľ. a kol. (2010) *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika: Didaktika programovania*. Bratislava : ŠPÚ, 2010. ISBN 978-80-8118-065-1

[11] Varga, M., Blaho, A., Zimanová, R. (1999) *Algoritmy s Logom*. Bratislava : SPN, 1999. ISBN 80-08-02965-X

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Gabriela Lovászová, PhD.
Mgr. Martin Cápay, PhD.
PaedDr. Viera Palmárová, PhD.

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Didaktika programovania pre ZŠ 2

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti PaedDr. Daniela Bezáková, PhD.
RNDr. Ľubomír Šnajder, PhD.

Počet strán 36

Náklad 300 ks

Prvé vydanie, Bratislava 2011

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-8118-091-0