RIADENIE MODELOV A SPRACOVANIE ÚDAJOV V SYSTÉME NXT LOGO

PAVEL PETROVIČ

Katedra aplikovanej informatiky, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského, Bratislava
e-mail: ppetrovic@acm.org

V súčasnosti sa do viacerých škôl dostáva nová robotická edukačná platforma LEGO Mindstorms NXT. Základný software, ako i alternatívne vývojové prostredia vychádzajú z princípu: naprogramuj a spusti, kde používateľ najskôr pripraví program na PC, prenesie ho do robota, a tam ho spustí. Systém NXT Logo, teda interpreter jazyka Logo pre NXT LEGO, vychádza z inej paradigmy: autonómny program, ktorý riadi robota môže súčasne komunikovať s prostredím Imagine Logo na PC. Tým sa možnosti využitia robotov vo výuke posúvajú na kvalitatívne vyššiu úroveň. V príspevku predstavíme NXT Logo a na príkladoch rozoberieme funkcie na vysokej úrovni pre vizualizáciu dát a riadenie pomocou grafických objektov Imagine Logo, ktoré umožňujú programovať roboty na úrovni začiatočníkov i pokročilých používateľov a sú určené pre učiteľov i žiakov informatiky na ZŠ a SŠ.

Kľúčové slová*: NXT Logo, Imagine, robotické stavebnice, vizualizácia dát, edukačná robotika*

1. ÚVOD

Využitie robotických stavebníc vo výuke má viacero cieľov a výhod. Predovšetkým zvyšuje záujem žiakov o technické a prírodovedné smery, zvyšuje motiváciu k hlbšiemu pochopeniu javov a záujem o učebný materiál, udržuje pozornosť žiakov po dlhšiu dobu, poskytuje platformu pre skupinovú a projektovú výuku a laboratórne cvičenia, umožňuje žiakom zažiť skúmané javy a zákony na vlastnej fyzickej skúsenosti. Použitie stavebníc je v súlade s konštruktivistickými teóriami učenia, kde si žiaci budujú svoju predstavu o učive a svete na základe vlastných pozorovaní a skúseností ako i s konštrukcionistickými téoriami [1], ktoré zdôrazňujú význam tvorivosti v tomto procese, interakciu mysle v priestore vymedzenom danou výukovou situáciou. Najrozšírenejšie robotické stavebnice sú stavebnice LEGO Mindstorms, dnes nasadené na Slovensku vo viac ako 100 školách. Tradičný prístup k práci so stavebnicami spočíva v navrhnutí (alebo využití existujúceho) modelu a naprogramovaní riadiacej jednotky, ktorá prostredníctvom motorov, snímačov, zvukov, správ na LCD a tlačidiel komunikuje s používateľom a interaguje s prostredím v príslušnom projekte – výukovom experimente. Tu žiaci spoznávajú predovšetkým základy riadenia a programovania robotických systémov: stabilitu, robustnosť, princípy spätnej väzby, vlastnosti a správanie snímačov, základy bezpečnosti a pod. Táto paradigma využíva pracovnú stanicu (PC) na vytvorenie programu pre riadiacu jednotku. Ten sa do jednotky následne prenesie, spustí a autonómne vykonáva požadovanú činnosť. Naše ciele sú však na inej úrovni ako ciele uvedeného prístupu, ktorý vychádza zo softvérového vybavenia dodávaného so stavebnicami. Inžinierske základy robotiky sú (voliteľne) iba doplňujúcimi cieľmi. Naopak, naším cieľom je využitie stavebníc vo výuke iných javov a princípov – na matematike, fyzike, biológii, či iných predmetoch. Zámer žiaka i učiteľa nemá byť naučiť sa pracovať s robotickou stavebnicou, ale vytvoriť resp. použiť model, na ktorom žiak pochopí fyzikálny princíp, jav, overí matematické pravidlo, namodeluje správanie organizmov, vytvorí názornú demonštráciu husitskej bojovej stratégie, alebo predvedie technologický princíp (strojárstvo, stavebníctvo, komunikácia, a pod.). Tieto ciele si vyžadujú odlišnú koncepciu programového vybavenia. Namiesto zaužívanej paradigmy naprogramuj a spusť žiak potrebuje softvér, kde tvorí a realizuje svoj interaktívny projekt. Tu robotický model komunikuje za behu s pracovnou stanicou, prebieha výmena údajov a používateľ okrem priamej interakcie s modelom monitoruje a kontroluje priebeh experimentu z pracovnej stanice, kde sa dáta ďalej spracovávajú, vhodne vizualizujú a prezentujú. Presne s týmito cieľmi sme vyvinuli a ďalej vyvíjame výukový softvér pre stavebnice NXT, ktorý je hladko integrovateľný s existujúcim výukovým prostredím Imagine Logo [2]. V nasledujúcich statiach opíšeme najskôr komunikáciu Imagine s modelmi z pôvodnej verzie stavebníc LEGO Mindstorms RCX – ktoré dokážu komunikovať cez infračervený sériový port, potom pre porovnanie opíšeme príklad klasického použitia stavebníc NXT v experimente bez využitia Imagine a NXT Loga. Následne predstavíme NXT Logo [3,4]. Pre potreby vizualizácie sme vyvinuli knižnicu *Charts* pre Imagine [7], ktorú stručne opíšeme v stati 5 a na záver na troch ilustratívnych príkladoch predvedieme tvorbu interaktívnych projektov s NXT stavebnicami prostredníctvom NXT Loga a knižnice Charts.

2. IMAGINE A LEGO MINDSTORMS RCX

Stavebnice s programovateľnou kockou RCX sú u nás zatiaľ najviac rozšíreným druhom (viac ako 100 škôl). Ide o staršie stavebnice uvedené na trh v roku 1998 s autonómnou programovateľnou kockou RCX, ku ktorej možno pripojiť 3 motory a 3 snímače. Hoci NXT Logo je určené pre stavebnice NXT a s RCX nepobeží, Imagine môže výborne

spolupracovať aj s RCX a tak možno vytvárať interaktívne projekty aj pre klasické modely postavené zo stavebníc s kockou RCX. Tie komunikujú s PC cez infračervený port: program bežiaci na RCX môže prijať alebo vyslať jednobajtové správy (pri použití alternatívneho softvéru a firmware pre RCX2 dostupného cez LEGO Mindstorms SDK2 [5] aj viacbajtové správy). Z Imagine môžeme teda na RCX buď vyslať správy pre program na RCX alebo dokonca prostredníctvom systémových správ priamo riadiť motory, zisťovať stav snímačov, premenných a časovačov, alebo posielať akékoľvek iné príkazy pre RCX. Podrobnosti možno nájsť na [6]. Treba mať prirodzene na pamäti, že dosah infračerveného portu je obmedzený i pri nastavení ďalekého rozsahu a rýchlosť prenosu je nízka, t.j. krátke správy možno vyslať rádovo 100-krát za sekundu. Uvedený systém umožňuje vytváranie interaktívnych projektov v Imagine, ktoré komunikujú s modelmi RCX pomocou knižnice pre Imagine **`rcx.imt`**. Tá obsahuje sadu základných príkazov pre komunikáciu s RCX [6].

3. KLASICKÉ PROGRAMOVANIE NXT

LEGO Mindstorms NXT je aktualizovaná verzia stavebníc, ktorá zodpovedá dnešnému štandardu vnorených systémov: 32-bitový procesor ARM pracujúci na vysokej frekvencii 48 MHz s 256 KB pamäte flash a 64 KB pamäte RAM, s grafickým displejom, s podporou I^2C zbernice ako i rýchlej zbernice IEC 61158 P-NET využívanej v priemyselných aplikáciách, bezdrôtového komunikačného protokolu BlueTooth, dokonca s podporou debugovacieho rozhrania JTAG pre vývojárov – v porovaní s RCX ide teda o obrovský technologický skok vpred a tento raz firma LEGO vyšla komunite užívateľov a vývojárov v ústrety a uverejnila podrobnú technickú dokumentáciu celého systému.

V detskom ikonografickom jazyku – novej verzii prostredia RoboLab – používateľ najskôr vytvorí program pre autonómny robot, ktorý potom prenesie cez USB kábel alebo cez BlueTooth rádiové spojenie do programovateľnej kocky. Namiesto štandardného softvéru možno použiť alternatívy, ktoré kopírujú syntax jazykov C alebo Java [8, 9]. Po odštartovaní programu model vykonáva činnosť a plní svoju úlohu, pričom môže zaznamenávať a ukladať údaje do súborov v pamäti kocky NXT typu flash. Súbory sa potom dajú preniesť pomocou osobitných funkcií RoboLab, alebo alternatívnych prostredí do PC a tam ďalej spracovávať, napríklad skonvertovať do podoby, ktorú možno importovať do tabuľkového editora MS Excel a ďalej vizualizovať pomocou grafov. Tento proces je pomerne náročný, sekvenčný a chýba mu interaktivita, možnosť priamej spätnej väzby a okamžitá a priebežná vizualizácia.

4. NXT LOGO

NXT Logo je našou odpoveďou na otázku vhodného programovacieho nástroja pre modely zo stavebníc LEGO Mindstorms NXT. Systém poskytuje viacero úrovní práce: v najvyššej vrstve ho tvorí projekt v Imagine Logo, ktorý obsahuje tlačidlá, posúvadlá a iné kontrolky, ktorými možno priamo riadiť existujúce modely: ich motory, sledovať a zaznamenávať hodnoty snímačov, posielať NXT modelom správy cez rádiové spojenie BlueTooth. Ak užívateľov projekt vyžaduje zložitejšiu interakciu a hlavne ak chce tvoriť vlastné projekty v Imagine, využije nasledujúcu vrstvu pozostávajúcu z knižnice procedúr a operácií **`nxt.imt`** pre Imagine, na riadenie NXT modelov z vlastných projektov používateľa. Tieto procedúry dovoľujú riadiť modely priamymi príkazmi cez spojenie BlueTooth a posielať správy programom, ktoré bežia na NXT a prijímať odpovede. Príkladom volania procedúry z knižnice **`nxt.imt`** je príkaz:

```
nxt_playsound "false "Woops.rso
```

ktorý zahrá zvuk uložený v súbore v NXT pamäti flash. Používatelia môžu naprogramovať svoj model napríklad v jazyku NXC [8], alebo aj v štandardnom RoboLab-e a komunikovať s modelom z prostredia Imagine. Keďže jedna správa v špecifickom NXT BlueTooth protokole môže mať dĺžku zhruba 50 bajtov, takto je možné prenášať rozsiahlejšie objemy dát pre ďalšie spracovanie v Imagine a naopak. Z hľadiska rýchlosti komunikácie je ale spojenie BlueTooth rýchle iba v jednom smere! To znamená, že obojsmerná komunikácia je veľmi obmedzená a za jednu sekundu sa takto dá vymeniť rádovo 10 správ v oboch smeroch dohromady. Systém NXT Logo však v tretej vrstve obsahuje samotný interpreter jazyka LOGO pre NXT kocku, a teda žiaci, ktorí zvládli programovanie v Imagine Logu môžu celkom plynule začať tvoriť programy pre NXT modely bez toho, aby sa museli učiť nový programovací jazyk a zoznamovať s novým programovacím prostredím. Naviac, programy v NXT Logu môžu kedykoľvek komunikovať s logovskými programami v Imagine, konkrétne z Imagine možno vyslať akýkoľvek logovský výraz na NXT, kde sa tento výraz vyhodnotí a výsledná logovská hodnota sa vráti naspäť do logovského programu v Imagine. Paradigma funkcionálneho programovania stiera rozdiel medzi dátami a programom a teda takto posielané výrazy môžu obsahovať kúsky programového kódu alebo dáta. Samozrejme, okrem toho je možné priamo z logovských programov v Imagine otvárať, čítať a zapisovať súbory v NXT pamäti flash a takto prenášať aj zložitejšie programy v logu pre NXT a dáta, ba čo viac – počas behu logovského programu na NXT možno vymieňať, spúšťať i zastavovať programy, ktoré sú v pamäti NXT. Všetky tri vrstvy systému NXT Logo sú podrobne popísané v

referenčnej a používateľskej príručke [4]. Dokumentácia tiež vymedzuje podmnožinu jazyka Logo, ktorá je v aktuálnej verzii podporovaná. V stručnosti, NXT Logo podporuje aritmetické i logické operácie, bitové logické operácie, operácie na slovách a zoznamoch, iterácie a funkcionálne procedúry (**map, apply, run, foreach, repeat, while, try**), podmienky a štandardné predikáty (**empty?, list?, word?, eq?, gt?, lt?, ge?, le?**), priraďovacie procedúry pre globálne i lokálne premenné (**make, let**). Nové príkazy sa vytvárajú príkazom **to**, definíciu užívateľských procedúr možno získať pomocou procedúry **text**. Ďalej NXT Logo pozná osobitné procedúry na načítanie programu z flash (**load**), náhodný generátor, podporu pre súborový systém (**fopen, fwrite, fread, fclose, parse**), podporu pre tlačidlá, LCD display NXT a zvuky, a prirodzene procedúry a operácie na riadenie snímačov a motorov. Odlišnosti dialektu NXT Loga od Imagine, predovšetkým z technických a implementačných dôvodov, sú nasledujúce:

- všetky výrazy používajú prefixovú notáciu, v NXT Logu sa nepoužívajú žiadne zátvorky, napríklad logovský výraz

  (5 + 4) / 2 vyzerá: **div add 5 4 2**.

- všetky výrazy sú volania procedúr alebo funkcií alebo priame hodnoty, NXT Logo neobsahuje žiadne špeciálne formy, aj samotné definovanie procedúr pomocou **to** je volanie procedúry. Napríklad nasledujúca definícia procedúry v logu:

```
to sumlist :l
  if empty? :l [op 0]
  op (first :l) + sumlist bf :l
end
```

  sa v NXT Logo zapíše takto:

```
to "sumlist [:l]
[
  if empty? :l [op 0]
  op add first :l sumlist bf :l
]
```

- všetky procedúry a funkcie majú pevný počet argumentov, nasledujúce logovské výrazy:

```
(sum 1 2 3 4 5 6 7)
(apply "sum [1 2 3 4])
(se [a b] [c d] "e [f g])
```

  je teda potrebné zapísať takto:

```
(add add add add add add
     1 2 3 4 5 6 7)
(apply "add
  se apply "add
   se apply "add [1 2]
      3
   4)
(se se se [a b] [c d] "e [f g])
```

- Imagine Logo prísne vyžaduje, aby každá hodnota vrátená funkciou bola ďalej spracovaná, prípadne ignorovaná procedúrou **ignore**. V NXT Logo sa nespracované hodnoty ignorujú automaticky a vykonávanie pokračuje nasledujúcim výrazom.

- NXT Logo má trochu iný vnútorný mechanizmus vyhodnocovania výrazov a preto vždy platí: výraz reprezentovaný ako zoznam logovských príkazov vždy vráti hodnotu prvého podvýrazu, alebo ak prvý podvýraz nevráti žiadnu hotnotu, tak celý výraz nevráti žiadnu hodnotu. Vždy sa ale vyhodnotia všetky podvýrazy zoznamu (okrem prípadov keď zoznam obsahuje **op** alebo **stop**). Pre porovnanie, výraz:

  ```
  run [3 print "hello]
  ```

  v oboch dialektoch vráti 3, ale vypíše **hello** iba v NXT Logo.

- Mená argumentov v definičnom zozname v NXT Logo obsahujú dvojbodku. V Imagine získame:

```
to f :a :b
  op :a + :b
end

text "f  –> [[a b][op :a + :b]]
```

  a v NXT Logo:

```
to f [:a :b] [op add :a :b]
text "f –>
       [[:a :b][op add :a :b]]
```

Na záver uvedieme niekoľko príkladov programov v NXT Logo:

```
; sčítanie prvkov číselného zoznamu
to "sum [:lst]
[
  let "s 0
  foreach "e :lst [make "s add :s :e]
  op :s
]

; robot sleduje čiaru
to "follow []
[
  setsensor 3 5 128
  motor 0 "onrev 40
  while ["true]
  [
    while [ge? sensor 3 59] []
    motor 0 "float 0
    motor 2 "onrev 40
    wait 50
    while [lt? sensor 3 60] []
    motor 2 "float 0
    motor 0 "onrev 40
  ]
]
```

```
; obrátenie zoznamu
to "rev [:l] [rev2 :l []]
to "rev2 [:l :a]
[
  if empty? :l [op :a]
  op rev2 bf :l fput first :l :a
]
```

NXT Logo obsahuje podporu aj pre používateľov, ktorí nevlastnia Imagine. Sada nástrojov pre príkazový riadok Windows umožňuje pracovať so súborovým systémom NXT (kopírovať, vyhľadávať, mazať súbory) a pomocou programu **remote.exe** možno komunikovať s interpretrom NXT Logo cez BlueTooth priamo – čiže napríklad spúšťať logovské programy, alebo vyhodnocovať logovské výrazy. Podobne je implementovaná podpora pre iné dialekty jazyka LOGO, napríklad pre Berkley Logo (UCBLogo), ktoré beží v operačnom systéme Linux. NXT Logo je stále vo vývoji, existuje funkčný prototyp, ale treba mať na zreteli, že ide o beta-verziu, určite sa ale potešíme všetkým pripomienkam a návrhom.

5. GRAFY

Za účelom vizualizácie dát z NXT sme navrhli a implementovali knižnicu tried pre Imagine na kreslenie grafov a editovanie tabuliek, *Charts* [7]. Táto knižnica je užitočná aj samostatne – v iných výukových projektoch v Imagine. Aktuálna verzia obsahuje 4 triedy **LineChart**, **ColumnChart**, **XYChart** a **Table**. Ich použitie predvedieme na nasledujúcom príklade. Predstavme si typickú meteorologickú stanicu, ktorá zaznamenáva priemerné denné teploty. Tri takéto stanice namerali v predchádzajúcom týždni tieto fiktívne teploty:

Chopok: **-4, -2, 1, 0.4, 2.5, -3, -10**
Sliač: **0, -1, -3, -2, 4, 0, -2**
Lomnický štít: **-15, -5, 0, 1, -1, -10, -12**

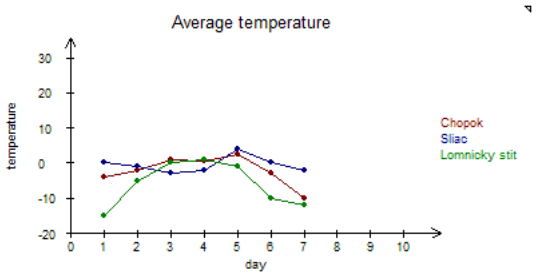
Prirodzený spôsob na reprezentáciu týchto dát v jazyku Logo sú tri zoznamy s teplotami a jeden doplňujúci zoznam s menami staníc:

```
? make "tmps [[-4 -2 1 0.4 2.5 -3 -10]
              [0 -1 -3 -2 4 0 -2]
              [-15 -5 0 1 -1 -10 -12]]
? make "locations [[Chopok] [Sliac]
[Lomnicky stit]]
```

Zobrazením vývoja teplôt v jednom grafe získame názorné porovnanie, dosiahneme ho takto:

```
? load "charts.imt
? new "LineChart [name meteo
            title [Average temperature]
            xlabel [day]
            ylabel [temperature]
            legend :locations
            nseries 3
            data :tmps]
```

Vytvorili sme objekt **meteo** triedy **LineChart**, čiže korytnačku, ktorej tvar bude graf teplôt so zadanými popismi osí a titulkom (obr. 1). Vzhľad grafu – rozsah osí – je generovaný automaticky, ale môžeme ho prispôsobiť takto:



Obr. 1 Korytnačka, ktorá vizualizuje údaje v grafe.

```
? meteo'setxmax 7
? meteo'setymin -20
? meteo'setymax 10
? meteo'paint
```

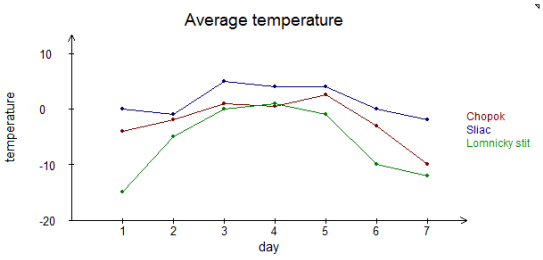
Veľkosť grafu môžeme plynule meniť ťahaním myšou za trojuholník v pravom hornom rohu a premiestniť na ľubovoľné miesto na ploche.

Objekt zobrazuje údaje, ktoré sú uložené priamo v jeho vnútornej premennej **data:**

```
? print meteo'data
[-4 -2 1 0.4 2.5 -3 -10] [0 -1 -3 -2 4
0 -2] [-15 -5 0 1 -1 -10 -12]
```

Tento atribút ale môžeme zmeniť tak, že nebude obsahovať samotné údaje, ale iba meno inej premennej, ktorá údaje uchováva. Po zmene údajov v tejto externej premennej stačí na prekreslenie tvaru korytnačky zavolať metódu **paint**:

```
? meteo'setdata ":tmps
? make "tmps deepReplace [2 3] :tmps 5
? make "tmps deepReplace [2 4] :tmps 4
? print meteo'data
:tmps
? print :tmps
[-4 -2 1 0.4 2.5 -3 -10] [0 -1 5 4 4 0
-2] [-15 -5 0 1 -1 -10 -12]
? meteo'paint
```



Obr. 2 Prispôsobený rozsah osí, údaje z premennej.

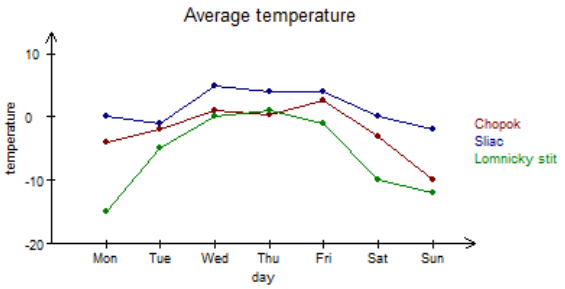
Väčšinou vystačíme s číslovaním dní, ale niekedy potrebujeme použiť názvy riadkov v dátach. Stačí ich pridať do prvého zoznamu údajov:

```
? make "tmps replace 1 :tmps [[Mon -4]
[Tue -2] [Wed 1] [Thu 0.4] [Fri 2.5]
[Sat -3] [Sun -10]]
? meteo'paint
```

Keď sme so vzhľadom, veľkosťou a polohou grafu spokojní, môžeme ho ukotviť napevno, aby ho

používateľ nemohol ďalej deformovať alebo presúvať takto (editovací trojuholník z rohu zmizne):

```
? meteo'seteditable "false
? meteo'paint
```



Obr. 3 Názvy riadkov dát, zafixovanie veľkosti.

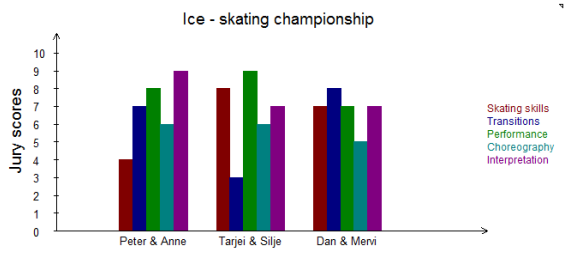
V prípadoch, keď hodnoty zodpovedajú jednotlivým udalostiam a nie spojitým premenným, ako to bolo v prípade teplôt, namiesto krivkového grafu je vhodnejšie použiť stĺpcový graf. Tak je to aj v prípade, keď porovnávame nejaké kvantitatívne parametre niekoľkých objektov alebo subjektov. Na majstrovstvách v tanečnom krasokorčuľovaní dosiahli dvojice nasledujúce hodnotenia poroty:

Pár	Technika	Prechody	Prevedenie	Choreografia	Interpretácia
Peter & Anne	4	7	8	6	9
Tarjei & Silje	8	3	9	6	7
Dan & Mervi	7	8	7	5	7

Použijeme objekt triedy **`ColumnChart`**, ktorá je odvodená od triedy **`LineChart`** a je veľmi podobná:

```
? make "skating [ [[[Peter & Anne] 4]
                  [[Tarjei & Silje] 8]
                  [[Dan & Mervi] 7]]
                  [7 3 8] [8 9 7]
                  [6 6 5] [9 7 7]]

? new "ColumnChart [name ice
       title [Ice-skating championship]
       xlabel [] ylabel [Jury scores]
       legend [[Skating skills]
        [Transitions] [Performance]
        [Choreography][Interpretation]]
       nseries 5
       data (fput ": "skating)]
```



Obr. 4 Stĺpcový graf – krasokorčuľovanie.

Výsledkom je opäť presúvateľná a škálovateľná korytnačka, zobrazená na obr. 4.

Porovnanie dvoch rôznych parametrov pre rôzne skupiny objektov sa dá prehľadne zobraziť v XY-grafe tak, že získame predstavu o vzájomnom vzťahu týchto dvoch parametrov a o rozložení objektov v priestore údajov. Vráťme sa teraz späť k príkladom z meteorológie. Každá stanica meria okrem dennej teploty aj celkové denné zrážky. Uvažujme len údaje namerané 1. júna počas viacerých rokov histórie merania. Fiktívne namerané údaje sú zobrazené v tabuľke:

Stanica	1951	1952	1953	...	2005	2006	2007
Lomnický štít	10.2° 3mm	12.5° 5mm	2.7° 10mm	...	19.2° 11mm	8.5° 20mm	12.1° 0mm
Sliač	16.2° 1mm	13.5° 20mm	9.7° 3mm	...	7.4° 4mm	18.4° 15mm	20.3° 12mm

Jeden z prirodzených spôsobov reprezentácie týchto údajov v jazyku Logo vyzerá takto:

```
? make "tempnrain [ [ [10.2 3] [12.5 5]
  [2.7 10] [4.2 30] [11.3 20] [8.1 7]
  [-3 12] [7.3 0] [11.7 33] [2.3 12]
  [7.2 20] [9.1 21] [9.7 11] [8.5 20]
  [12.1 0] ]
 [ [16.2 1] [13.5 20] [9.7 3] [21 0]
  [7.5 30] [18.2 8] [14.2 2] [12.1 2]
  [14.9 10] [13.2 2] [16.3 0] [12.5 20]
  [7.4 4] [18.4 15] [20.3 12] ] ]
```

Údaje môžeme zobraziť objektom triedy **`XYChart`**, ktorý je tiež odvodený od triedy **`LineChart`**:

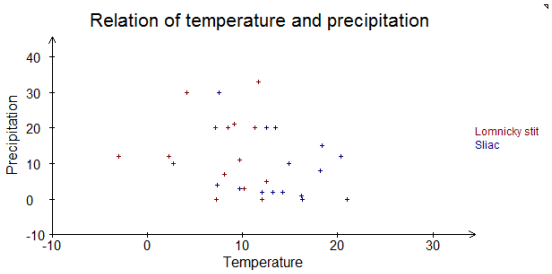
```
? new "XYChart [name comp
      title [Relation of temperature
             and precipitation]
      xlabel [Temperature]
      ylabel [Precipitation]
      legend [[Lomnicky stit][Sliac]]
      nseries 2
      data (fput ": "tempnrain)]
```

Podobne ako pre čiarový graf, aj tu môžeme upraviť rozsahy osí:

```
? comp'setxmin -10
? comp'setxmax 30
? comp'setymin -10
? comp'setymax 40
? comp'paint
```

Body sú v grafe zobrazené ako vyplnené kružnice, vykresľovanie bodov je riadené logovským výrazom. Môžeme ho nastaviť napríklad na postupnosť príkazov, ktoré vykreslia krížik:

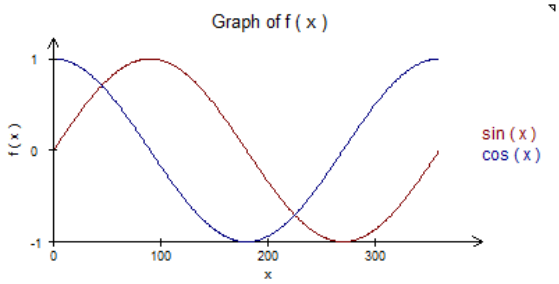
```
? comp'setpoints
    [repeat 2 [fd 2 bk 4 fd 2 rt 90]]
? comp'paint
```



Obr. 5 XYGraf s bodmi vyznačenými krížikom.

Atribút **points** môžeme využiť aj na úplné potlačenie vykresľovania bodov. Trieda **LineChart** takto vykresľuje spojité funkcie:

```
? make "twopi generate 360
               [[x][op :x + 1]] 1
? make "sincos list map "sin :twopi
                    map "cos :twopi
? new "LineChart [name fx
          title [Graph of f(x)]
          xlabel [x]
          ylabel [f(x)]
          legend [[sin(x)] [cos(x)]]
          nseries 2
          data (fput ": "sincos)
          points []
          xmin 0 xmax 360
          ymin -1 ymax 1]
```



Obr. 6 Graf funkcií **sin(x)** a **cos(x)**.

V niektorých prípadoch aplikácia nemá sama prístup k údajom a treba, aby ich zadal používateľ. V iných prípadoch je vhodné zobraziť rôzne údaje vo forme prehľadnej tabuľky namiesto grafu. A v niektorých prípadoch je najvhodnejšie zobraziť graf i tabuľku súčasne, pričom graf automaticky zobrazuje údaje podľa toho, ako sa v tabuľke menia. Vo všetkých týchto prípadoch poslúži trieda **Table** odvodená od triedy **Turtle**. Prázdnu tabuľku vytvoríme príkazom:

```
? new "Table [name t]
```

Používateľ môže tabuľku presunúť a natiahnuť na želané miesto a veľkosť, prispôsobiť počet zobrazených riadkov a stĺpcov. Tieto parametre možno zafixovať nastavením atribútu **editable** na **false**. Potom používateľ stále môže meniť obsah tabuľky, až dokiaľ nastaveníme atribút **readonly** na **true**:

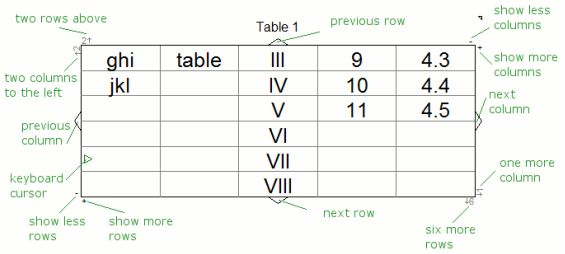
```
? t'setreadonly "true
```

Ak tabuľka nie je read-only, používateľ môže meniť údaj v bunke tabuľky, do ktorej klikne myšou (tabuľka má neohraničený počet riadkov a stĺpcov), až dokiaľ nestlačí ENTER, alebo neklikne do inej časti tabuľky. Ak stlačí ENTER, kurzor automaticky prejde na bunku v nasledujúcom riadku a v rovnakom stĺpci a používateľ môže stlačiť ľubovoľnú klávesu na editovanie obsahu tejto bunky, prípadne šípky na presun kurzora na inú bunku, alebo klávesy DEL či BACKSPACE na vymazanie obsahu bunky, resp. kláves ESC na ukončenie editovania tabuľky.

Podobne ako v triedach pre grafy, údaje sú uložené v atribúte **data** tabuľkového objektu, alebo v externej premennej:

```
? make "somedata [[1 2 3 4] [5 4 3 2 1]
                  [abc def ghi jkl]]
? t'setdata ":somedata
? t'paint
```

Korytnačka s obrázkom tabuľky bude vyzerať takto (na obrázku je navyše zobrazený aj význam jednotlivých grafických indikátorov a ovládačov):

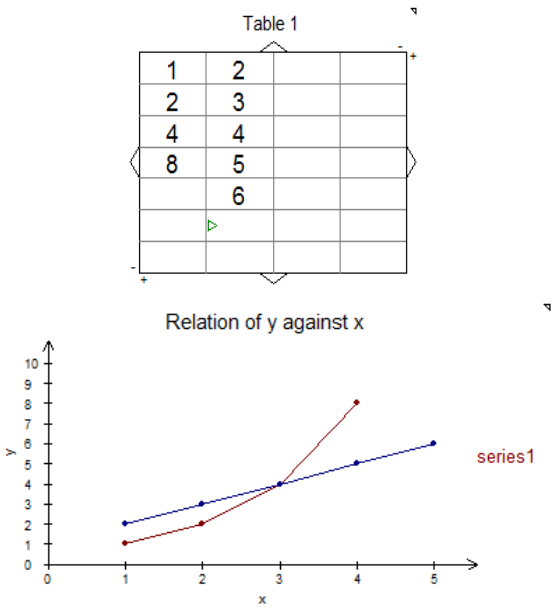


Obr. 7 Korytnačka – tabuľka.

Ak tabuľka zdieľa údajovú premennú s grafom, môžeme prepojiť takéto dva objekty tak, že graf zobrazuje obsah tabuľky. Vzhľadom na to, že nemáme žiadnu kontrolu nad tým, aké dáta používateľ do tabuľky zadá, je potrebné napísať procedúru, ktorá údaje tabuľky bezpečne preloží tak, aby boli zobraziteľné v grafe: stačí zadefinovať metódu **onChanged** príslušného objektu triedy **Table**. Táto metóda sa automaticky volá po každej používateľovej zmene obsahu tabuľky. V tomto príklade prepisujeme všetky bunky, ktoré neobsahujú čísla, nulami:

```
to onChanged :row :col
  make "shared map [[col] [op map
    [[cell][ifelse not number? :cell
              [op 0] [op :cell]]] :col]]
   :shared
  paint
  graf'setnseries count :shared
  graf'paint
end

? make "shared [[5 4 3]]
? new "LineChart [name graf]
? new "Table [name tab]
? tab'setdata ":shared
? graf'setdata ":shared
? graf'paint
? tab'paint
```



Obr. 8 Graf priamo zobrazuje hodnoty z tabuľky.

V takejto konfigurácii používateľ ľubovoľne mení hodnoty v tabuľke a graf automaticky zobrazuje zmeny. Všimnite si, že procedúra `onChange` už nedbá o legendu, ktorá ostáva nezmenená nezávisle od počtu stĺpcov. Riešenie ako i ďalšie neobmedzené možnosti ponechávame na tvorivosť čitateľa.

6. INTERAKTÍVNE PROJEKTY

Hlavným cieľom tejto práce je umožniť vývojárom, učiteľom i študentom vytvárať interaktívne projekty s robotickými stavebnicami, ktoré demonštrujú nejakú časť učiva – či už z informatiky alebo z iných predmetov. Na ukážku sme pripravili nasledujúce tri experimenty.

6.1. Nočné osvetlenie

Námet: V projekte študenti vytvoria model automaticky riadeného nočného pouličného osvetlenia, namerajú údaje a stanovia funkciu pre výpočet nákladov na osvetlenie a určia optimálnu hodnotu snímača pre zapnutie osvetlenia.

Úloha: Model pozostáva z riadiacej jednotky, svetelného snímača v pasívnom režime, zatieniteľného prirodzeného (prípadne náhradného umelého) svetla a umelého osvetlovacieho zdroja, ktorý môžeme ručne – prípadne automaticky zapnúť a vypnúť. Našou úlohou je nájsť vhodný prah hodnoty svetelného snímača, pri ktorom sa má automaticky zapínať umelé osvetlenie tak, aby svetelnosť nikdy neklesla pod úroveň nočného osvetlenia a stanoviť funkciu celoročných nákladov na osvetlenie podľa zisteného prahu.

Postup: 1. Nameriame údaje o svetelnosti vychádzajúcej len z prirodzeného svetla pre dvanásť rôznych mesiacov v roku, vždy pre všetkých dvadsaťštyri hodín dňa (samozrejme v modelovom experimente sa stačí zamerať na pár „dní“ v roku a pár „hodín“ za deň).

2. Hodnoty zobrazíme v tabuľke, prípadne v grafe.

3. Naprogramujeme funkciu na výpočet celkových nákladov pre danú prahovú hodnotu snímača spínajúceho osvetlenie.

3. Nameriame svetelnosť pri najnižšom možnom prirodzenom osvetlení a zapnutom umelom osvetlení.

4. Hodnotu z bodu 3 použijeme ako optimálny prah tak, aby svetelnosť nikdy nebola nižšia a vypočítame celkové náklady, tie vizualizujeme v stĺpcovom grafe rozdelené podľa mesiacov.

6.2. Výstupná kontrola kvality

Námet: Na dopravníkovom výrobnom páse sú transportované výrobky – napríklad zo skla – fľaše, nádoby... Pomocou snímača na zisťovanie vzdialenosti (v praxi by sa použil napr. presný 3D laserový snímač) študenti zostroja systém výstupnej kontroly kvality.

Úloha: Dopravníkový pás je poháňaný motorom a posúva sa malou rýchlosťou. Nad ním je pevne pripojený ultrazvukový snímač, ktorý meria približne 50 hodnôt za sekundu. Pomocou grafu vizualizujeme snímané hodnoty pre správny aj pre chybný výrobok a navrhneme funkciu, ktorá na základe zmeraných dát určí, či výrobok má prejsť výstupnou kontrolou kvality. Operátor má pre každý výrobok k dispozícii nameraný profil, kvantifikovanú chybu výroby a v prípade potreby môže tlačidlom na ploche zmeniť smer pohybu dopravníka a skontrolovať výrobok znova.

Postup: 1. skonštruujeme výrobný pás s nízkou rýchlosťou pohybu, upevníme snímač a nameriame profil správneho a chybného výrobku, ktoré zobrazíme v grafe.

2. vytvoríme program, ktorý na základe zosnímaných údajov a kvadratickej odchýlky od vzoru vypočíta a zobrazí koeficient podobnosti so správnym výrobkom na základe ktorého systém určí, či výrobok je správny.

3. Do programu doplníme možnosť riadenia pásu a prípadne vizualizáciu štatistiky počtu správnych výrobkov v jednej dávke.

6.3. Inteligentné riadenie dopravy

Námet: Študent vytvorí jednoduchý systém inteligentného riadenia svetelnej dopravnej signalizácie.

Úloha: V zjednodušenom dopravnom ihrisku je na križovatke inštalovaná svetelná signalizácia, ktorú riadime z NXT kocky, uvažujme metaforu: svetlo svieti = je voľno. V našom prípade stačia tri semafory. V inteligentnom systéme riadenia dopravy reagujú semafory na prítomnosť vozidiel

a optimalizujú čas státia na križovatke, čím šetria pohonné hmoty. Študenti prakticky porovnajú klasický intervalový model semaforov s inteligentným a úspory paliva zobrazia v grafe.

Postup: 1. Zostrojíme križovatku, nainštalujeme snímače a semafory, naprogramujeme základné riadenie svetiel, intervalové riadenie semaforov a prepínanie svetiel na základe prítomnosti vozidiel.

2. Určíme celkový čas čakania pre oba spôsoby riadenia svetiel prakticky použitím jednoduchých vozidiel s 1 motorom NXT, ktoré sa presúvajú vpred a vzad, výsledky zobrazíme v grafe.

6.4. Ďalšie námety

Systémy automatického i interaktívneho riadenia procesov sa nachádzajú všade okolo nás a mnohé z nich obsahujú zaujímavé javy, ktoré má pre účely vysvetlenia učiva zmysel modelovať. Napríklad:

1. *Mechanický pohyb*: pomocou ultrazvukového snímača meriame vzdialenosť, pomocou motorov s regulovanou rýchlosťou realizujeme rovnomerný alebo zrýchlený pohyb, v programe zmeriame čas pohybu. Využijeme grafy na zobrazenie závislostí a pochopenie súvislostí medzi rýchlosťou, zrýchlením. Typická úloha: Vozidlo kvôli poruche prešlo prvú polovicu cesty polovičnou rýchlosťou, po oprave na polceste pokračovalo bežnou rýchlosťou – aká je priemerná rýchlosť na celej trati? Vo výuke matematiky sú mechanické fyzikálne experimenty veľmi vhodné na pochopenie významu derivácie funkcie a integrálov.

2. *Meranie gravitačného zrýchlenia a odporu prostredia vzduchu*: predmety rôznych hmotností a rovnakého tvaru spúšťame z výšky niekoľkých metrov v presnom časovom okamihu pomocou akcie motora a meriame čas dopadu pomocou dopadovej plochy pripojenej na tlakový, optický, či zvukový snímač. Ak majú telesá známe hmotnosti a rovnaký tvar a obal (napr. z kinder-vajca) – teda i odpor vzduchu, môžeme zo sústavy rovníc vypočítať gravitačné zrýchlenie i veľkosť vzdušnej odporovej sily. Tip: porovnanie so strojom Georga Atwooda.

3. *Goniometrické funkcie*: Na základe inšpirácie triangulačnou metódou pre určovanie polohy bodov využívanou v kartografii, môžeme zostaviť sadu úloh: snímačom meriame vzdialenosť k predmetom na neznámych súradniciach z rôznych bodov so známymi súradnicami a tak určíme ich polohu.

4. *Archimedov zákon*: teleso zavesené na kladke ovládanej motorom vytlačí zafarbenú kvapalinu do odmerky, kde pohyblivým optickým snímačom zmeriame výšku hladiny vytlačenej kvapaliny.

5. *Zvuk a mechanické vlnenie*: robot s pripojeným zvukovým snímačom zaznamenáva hodnoty pre rôzne druhy skladieb a reaguje pohybom do rôznych smerov podľa toho ktorá melódia (resp. rytmus) zaznie: melódia, tak ako je vnímaná snímačom, sa vizualizuje v grafe a robot sa rozhodne podľa výpočtu podobnosti k natrénovaným vzorom.

6. *Kužeľosečky*: v prvom cvičení necháme robota pohybovať sa po čiare v tvare elipsy, paraboly, či hyperboly a meriame funkciu uhla vzhľadom na čas kompasovým snímačom (prípadne otáčkomermi). V druhom prípade sa pokúsime naprogramovať pohyb robota po jednotlivých druhoch kužeľosečiek. V treťom cvičení sa robot pohybuje po ľubovoľnej trajektórii zadanej funkciou, ktorú aproximujeme kužeľosečkami (alebo aspoň kruhovými výsekmi) a necháme robota prejsť po takejto trajektórii. Robot môže pomocou pera alebo fixky na papier zaznamenávať svoju trajektóriu.

7. *Tepelná výmena, teplo*: za pomoci tepelného snímača zaznamenávame teplotu tekutín pri klasických fyzikálnych experimentoch tepelnej výmeny. Priebeh vývoja teplôt zobrazujeme v grafe a určíme konštanty mernej skupenskej kapacity rôznych kvapalín.

7. ZÁVER

Hoci na ceste k bežnému použitiu robotických modelov vo výuke iných predmetov okrem informatiky a samotných princípov robotiky sme stále na začiatku púte, vytvorili sme nástroj, ktorý si stavia za cieľ na túto púť nastúpiť a to prostredníctvom tvorby interaktívnych projektov v Imagine, ktoré priamo integrujú technológiu LEGO Mindstorms NXT do širokých možností výukového prostredia Imagine. NXT Logo je plnohodnotný interpreter jazyka Logo pre stavebnice LEGO Mindstorms NXT. Knižnica Charts obsahuje triedy pre Imagine Logo na vizualizáciu a editovanie dát. Ako sme ukázali, ich kombinácia umožňuje tvoriť interaktívne projekty, ktoré priebežne vizualizujú a riadia priebeh výukových experimentov s robotickými stavebnicami. Stavebnica použitá v tejto práci bola zakúpená z grantu APVV č. LPP-0301-06.

8. LITERATÚRA

[1] Papert, S., Harel, I. (ed.) *Constructionism: research reports and essays 1985 - 1990*, the MIT Media Lab, Ablex Pub. Corp, 1991.
[2] Kalaš, I. and Hrušecká, A. *The Great Big Imagine Logo Project book*, Logotron, 2004.
[3] Petrovič, P. Program Your NXT Robot with Imagine, Eurologo, 2007.
[4] NXT Logo: robotika.sk/NXTLogo
[5] LEGO Mindstorms SDK2: mindstorms.lego.com/sdk2/default.asp
[6] Imagine a RCX: wiki.robotika.sk/index.php/Imagine_plus_RCX
[7] Charts: wiki.robotika.sk/index.php/Charts_for_Imagine
[8] NXC a NBC: bricxcc.sourceforge.net/nbc/
[9] Lejos: lejos.sourceforge.net/