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

Programovanie 4 (Imagine)

Predmet: Programovanie

Línia: Vlastný odborový kontext 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

Programovanie 4 (Imagine)

Identifikácia modulu

Aktivita projektu:	1.3 Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Vlastný odborový kontext informatiky a informatickej výchovy
Predmet:	Programovanie

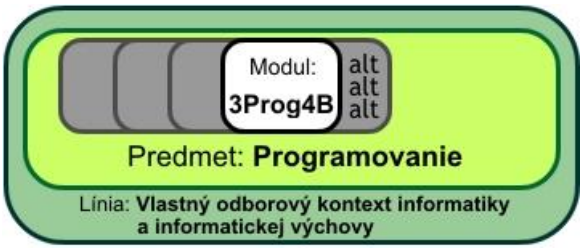
Garant predmetu:

PaedDr. Daniela Bezáková, PhD.
KZVI FMFI UK, Bratislava
bezakova@fmph.uniba.sk

Autori:

PaedDr. Daniela Bezáková, PhD.
KZVI FMFI UK, Bratislava
Mgr. Peter Kučera
1. súkromné gymnázium,
Bajkalská 20, Bratislava
RNDr. Gabriela Lovászová, PhD.
KI FPV UKF, Nitra
RNDr. Peter Tomcsányi,
KZVI FMFI UK, Bratislava

Zaradenie modulu



Päť z nich sa venuje programovaniu. V rámci týchto piatich modulov sa účastníci stretnú s rôznymi vývojovými prostrediami a programovacími jazykmi. **Modul *Programovanie 4 (Imagine)* je zameraný na typy údajov, OOP a niektoré pokročilé programátorské techniky v prostredí Imagine**. Nadväzuje na modul Programovanie 1 zameraný na úvod do programovania v jazyku Logo. Je alternatívou k pokročilejšiemu programovaniu v prostredí Delphi alebo Lazarus.

Abstrakt modulu

Modul Programovanie 4 (Imagine) sa venuje programovaniu v programovacom jazyku Logo. Na vzdelávanie budeme využívať prostredie Imagine, verziu so slovenskými príkazmi.
Modul pozostáva zo štyroch tematických jednotiek. V prvej časti sa oboznámime s rôznymi typmi údajov v Logu, číslami, slovami, zoznamami a textovými súbormi, naučíme sa používať základné príkazy na prácu s nimi a využívať ich v projektoch. V druhej časti sa budeme venovať korytnačke ako objektu, a tiež ostatným objektom prostredia Imagine (stránka, papier, tlačidlá, ...), ukážeme ich spoločné znaky a princíp práce s nimi. Dozvieme sa, ako vytvárať a nastavovať vlastnosti, premenné, udalosti a procedúry objektov interaktívne v režime návrhu i pomocou príkazov. V tretej časti sa naučíme vytvárať vlastné triedy odvodené od triedy korytnačka, vytvárať a používať inštancie vlastnej triedy. V tejto časti si tiež ukážeme, ako môžeme definovať korytnačke tvar pomocou príkazov jazyka Logo a na čo môžeme využiť korytnačky s kresleným tvarom. Obsahom štvrtej časti budú princípy tvorby sieťových aplikácií v prostredí Imagine, prenos textu, objektov a inštrukcií a tvorba jednoduchých sieťových hier.
Počas vzdelávania v rámci modulu Programovanie 4 (Imagine) sa účastník naučí efektívne používať rôzne nástroje prostredia Imagine, objektovo orientované programovanie v Imagine, zoznámi sa s niektorými pokročilejšími technikami v programovacom prostredí Imagine, získa námety na množstvo projektov a sám vyvinie niekoľko z nich.

Obsah

Úvod

Imagine je moderné programovacie prostredie na výučbu programovania, v ktorom si žiaci osvojujú princíp objektového a udalosťami riadeného programovania. Nepíšu dlhý súvislý program, ale navrhujú štruktúru projektu, ktorý môže obsahovať rôzne objekty ako sú korytnačky, grafické plochy, tlačidlá, posúvače, textové polia a iné. Programujú podprogramy ako reakcie na udalosti, napríklad kliknutie myšou na objekt, ťahanie objektu myšou, kolízie objektov pri pohybe. V tomto prostredí môžu vytvárať graficky veľmi príťažlivé projekty a zoznámiť sa aj s netriviálnymi pojmami z programovania. V tomto module sme sa zamerali na systemizáciu poznatkov o štýle programovania v Imagine a na niektoré ďalšie pokročilé témy ako sú práca so zloženými údajmi a sieťové programovanie.

Pre realizáciu modulu predpokladáme nainštalované: prostredie Imagine (akademická verzia) a LogoMotion, webový prehliadač (Firefox, Internet Explorer), Imagine plugin, Acrobat Reader. Akademická verzia prostredia Imagine je zhodná s verziou, ktorú majú základné a stredné školy z projektu Infovek.

Cieľ modulu

Cieľom modulu je ukázať účastníkom vzdelávania ďalšie možnosti a nástroje prostredia Imagine pre vývoj jednoduchých projektov, oboznámiť ich s rôznymi typmi údajov jazyka Logo a princípmi práce s nimi, systemizovať poznatky o objektoch v Imagine, ukázať spoločné črty jednotlivých súčiastok a práce s nimi, prezentovať základy tvorby sieťových projektov v prostredí Imagine, rozvíjať tvorivosť, algoritmické myslenie a schopnosť riešiť problémy. Účastníci by mali spoznať a vyskúšať si viaceré projekty prostredia Imagine a získať rôzne námety, ktoré môžu využívať pri výučbe programovania so svojimi žiakmi.

Vstupné vedomosti

Požadované prerekvizity

Účastníci tohto modulu by mali mať úspešne absolvovaný modul Programovanie 1.

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Predpokladáme, že účastník modulu:

- vie pracovať s OS Windows, so súbormi a s prehliadačom,
- rozumie základným programátorským pojmom,
- vie čítať kód v nejakom programovacom jazyku a analyzovať ho,
- vie si rozdeliť problém na menšie podproblémy,
- pozná základné príkazy jazyka Logo, základy korytnačej grafiky,
- definuje a používa vlastné príkazy pre korytnačku,
- používa tlačidlá na ovládanie korytnačky a posúvače na rôzne nastavenia,
- vytvára viaceré korytnačky,
- definuje reakciu korytnačiek na rôzne udalosti (pri kliknutí či ťahaní),
- rozhoduje sa na základe farby pozadia či pozície korytnačky na stránke,
- mení korytnačkám tvary pomocou obrázkov zo súborov,
- oživuje korytnačky rôznymi procesmi,
- vytvára jednoduché projekty využitím príkazov jazyka Logo a nástrojov prostredia Imagine.

Preverenie vstupných vedomostí

Nepožadujeme žiadne preverenie vstupných vedomostí.

Programovanie v prostredí Imagine

Kapitola 1: Typy údajov v Logu

V úvodnom module programovania v prostredí Imagine sme ako parametre príkazov používali údaje rôzneho typu:

- čísla, napr. `do 50`, `vp 90`,
- slová, napr. `nechFP "červená`, `zastav "pohyb`,
- zoznamy, napr. `nechFP [255 102 0]`, `?prvok [žltá6 žltá7 žltá8]`, `polygón [do 50 vp 90 do 50]`.

V tejto kapitole sa naučíme pracovať so zloženými údajmi: písať programy, v ktorých budeme pristupovať k častiam zložených údajov (znak v slove, prvok v zozname) a vytvárať zložené údaje z jednoduchších. Zoznámime sa s príkladmi projektov, ktoré využívajú údaje z textového súboru.

Body, vektory

Jedným zo základných nastavení korytnačky je jej pozícia. Pozícia je usporiadaná dvojica čísiel – x-ovej a y-ovej súradnice. V Logu má tvar dvojprvkového zoznamu. Pozíciu korytnačky zistíme operáciou `poz` a nastavíme príkazom `nechPoz`. Ak mala korytnačka nastavené pero dole, nakreslí pri presune do novej pozície čiaru.

Pozíciu korytnačky môžeme chápať ako bod v rovine s dvomi súradnicami, ale aj ako koncový bod vektora so začiatkom v bode `[0 0]` súradnicovej sústavy. V Imagine Logu sa dajú s vektormi vykonávať vektorové operácie:

sčitovanie	`[10 20] + [30 -10]`	`[40 10]`
odčitovanie	`[10 20] - [30 -10]`	`[-20 30]`
násobenie skalárom	`2 * [10 20]`	`[20 40]`
delenie skalárom	`[10 20] / 2`	`[5 10]`
násobenie po zložkách	`[-1 1] * [10 20]`	`[-10 20]`
otočenie	`otoč 90 [0 10]`	`[10 0]`
veľkosť	`abs [10 20]`	`22.3607`

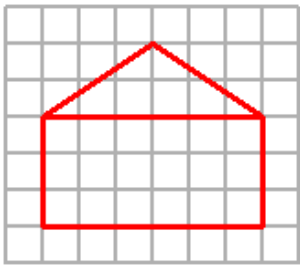
Možnosti vektorovej aritmetiky si predstavíme pri tvorbe nasledujúcich projektov.

Projekt Kaleidoskop

Vytvorme projekt, v ktorom budeme kresliť voľnou rukou ťahaním korytnačky a ďalšie tri korytnačky budú kresliť ten istý obrázok súmerne podľa x-ovej, y-ovej osi a podľa stredu súradnicovej sústavy.

Na papier (čiernu oblasť v pripravenom projekte) umiestnime štyri korytnačky `k1`, `k2`, `k3`, `k4`. Všetkým korytnačkám nastavíme hrúbku pera na 2. Korytnačky `k2`, `k3` a `k4` skryjeme.

Nakreslime domček:



- pomocou: korytnačích príkazov `dopredu`, `vľavo`, `vpravo`,

```
viem dom1
 opakuj 2
  [do 60 vp 90
   do 120 vp 90]
 do 60
 vp arctan 60/40
 do sqrt 5200
 vp 2*arctan 40/60
 do sqrt 5200
 vp arctan 60/40
 do 60
 vp 90 do 120
 vp 90
koniec
```

- pomocou zmien pozície korytnačky.

```
viem dom2
 nechPoz poz+[0 60]
 nechPoz poz+[60 40]
 nechPoz poz+[60 -40]
 nechPoz poz+[-120 0]
 nechPoz poz+[120 0]
 nechPoz poz+[0 -60]
 nechPoz poz+[-120 0]
koniec
```

V e-learningovom kurze k tomuto modulu sa nachádza projekt *kaleidoskop.imp*, v ktorom je pripravená paleta farieb, čierny papier a tlačidlo, ktoré zmaže papier.

V rodnom liste korytnačky `k1` nastavíme automatické ťahanie na zapnuté a definujeme reakciu na udalosť **priŤahaní**:

```
k2'nechpoz [1 -1] * poz
k3'nechpoz [-1 1] * poz
k4'nechpoz [-1 -1] * poz
```

Vynásobením pozície korytnačky `k1` vektorom `[1 -1]` po zložkách dostaneme vektor, v ktorom je x-ová súradnica rovnaká ako u korytnačky `k1` a y-ová súradnica opačná. Koncový bod tohto vektora bude súmerný podľa x-ovej osi s pozíciou korytnačky `k1`. Podobne po vynásobení pozície korytnačky `k1` vektorom `[-1 1]` dostaneme bod súmerný podľa y-ovej osi a po vynásobení vektorom `[-1 -1]` podľa stredu. Nezabudnime nastaviť pero dolu pre všetky štyri korytnačky a môžeme kresliť.

Do projektu ešte dopracujeme možnosť voliť si farbu kreslenia klikaním na paletu farieb vedľa papiera. Pridáme piatu korytnačku `k5` na stránku s paletou a vypneme jej pero. Pri kliknutí na stránku (udalosť **priKliknutí** stránky `stránka1`) zmeníme polohu korytnačky `k5` na pozíciu myši a kresliacim korytnačkám `k1` až `k4` nastavíme farbu pera na farbu bodu, na ktorom sa nachádza korytnačka `k5`. Udalosť **priKliknutí** pridáme stránke takto: klikneme na farebnú paletu pravým tlačidlom myši, zvolíme **Zmeň Stránka1** a v záložke udalosti pridáme udalosť **priKliknutí** s reakciou:

```
k5'nechPoz pozMyši pre [k1 k2 k3 k4] [nechFP k5'farbaBodu]
```

Zadanie 1	Pridajte do projektu možnosť kresliť prerušené čiary. (Návod: Pod paletu farieb pridajte tlačidlo - prepínač s popisom **pd/ph**. Definujte mu udalosti **priZapnutí** a **priVypnutí**: keď tlačidlo zapnem, dajte všetkým štyrom korytnačkám pero dolu, keď ho vypnem, dajte všetkým štyrom korytnačkám pero hore.)
Zadanie 2	Upravte projekt *Kaleidoskop* takto: Na papier umiestnite osem kresliacich korytnačiek. Prvou sa bude kresliť ťahaním, ostatné budú kresliť to isté otočené o 45, 90, ..., 315 stupňov okolo začiatku súradnicovej sústavy. Návod: Použite operáciu `otoč`.
Zadanie 3*	Upravte projekt *Kaleidoskop* takto: Na papieri budú dve korytnačky. Druhá korytnačka bude kresliť ten istý obrázok ako prvá, ale dvakrát zväčšený. Návod: Zmeňte začiatok súradnicovej sústavy v rodnom liste papiera do ľavého dolného rohu a pozíciu druhej korytnačky vypočítajte ako dvojnásobok vektora pozície prvej korytnačky.

Projekt Pohyblivé krivky

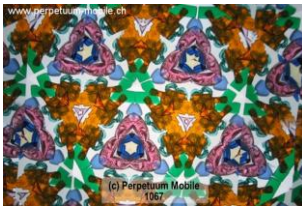
V tomto projekte budeme kresliť krivky, ktoré vznikajú pri pohybe korytnačky v strede medzi dvomi korytnačkami pohybujúcimi sa po kružniciach.

Vytvoríme tri korytnačky `k1`, `k2`, `k3`. Korytnačka `k1` sa bude pohybovať v nekonečnom procese po ľavotočivej kružnici, korytnačka `k2` po pravotočivej kružnici (robia malé kroky dopredu a otáčajú sa o malý uhol vľavo resp. vpravo). Tretia korytnačka `k3` sa bude pohybovať tak, aby sa vždy nachádzala v strede medzi korytnačkami `k1`, `k2`. Súradnice stredu úsečky medzi pozíciami korytnačiek `k1`, `k2` vypočítame ako aritmetický priemer ich pozícií.

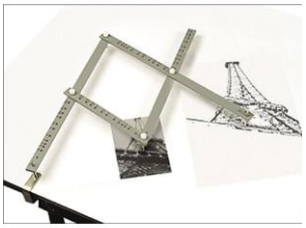
Papier je obdĺžniková oblasť, v ktorej môžu žiť korytnačky a nachádzať sa ďalšie objekty, podobne ako na stránke.

Tip: Kresliacim korytnačkám `k1` až `k4` nastavte typ oblasti na karte Pozícia v rodnom liste na **Bez hraníc**.

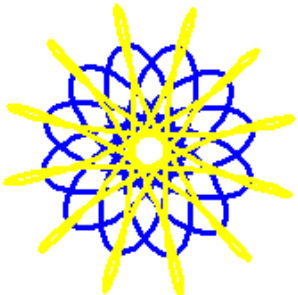
Kaleidoskop alebo krasohľad je optická hračka, ktorá vyzerá ako ďalekohľad. Vo vnútri sa odrážajú od stien farebné sklíčka a vytvárajú pekné symetrické obrazce.



Pantograf je mechanická pomôcka na zväčšovanie obrázkov.



Vyskúšajte si aplet na stránke http://www.ies.co.jp/math/java/geo/panta/panta.html
Vedeli by ste naprogramovať podobný projekt v Logu?



Podobné krivky môžeme vytvoriť pomocou mechanickej hračky *spirograf*. Krivka vznikne ako dráha bodu vo vnútri kružnice, ktorá sa kotúľa po vnútornej alebo vonkajšej strane inej kružnice. Na stránke http://wordsmith.org/~anu/java/spirograph.html je aplet na simulovanie spirografu.

```
každých 20
  [k1'do 5 k1'vl 5
   k2'do 5 k2'vp 7
   k3'nechPoz (k1'poz+k2'poz)/2]
```

Korytnačka `k3` kreslí, `k1` a `k2` majú pero zdvihnuté.

Zadanie 4

Dokončite projekt:

- pridajte tlačidlo-prepínač na spúšťanie a zastavovanie procesu kreslenia,
- pridajte tlačidlo na zmazanie stránky,
- pridajte posúvače, ktorými sa budú meniť veľkosti kružníc pre korytnačky `k1`, `k2`.

Zoznamy

Pozícia je príkladom jednoduchého dvojprvkového zoznamu čísiel. Zoznam však môže byť aj zložitejšia štruktúra údajov.

Zoznam je postupnosť údajov ľubovoľného typu. Prvky zoznamu sa uzatvárajú do zátvoriek `[`, `]` a oddeľujú medzerou. Prázdny zoznam má tvar `[]`. K prvkom zoznamu môžeme pristupovať a ich vlastnosti zisťovať pomocou operácií:

Vyskúšajte si zobraziť výsledky všetkých operácií s pozíciou korytnačky. Napríklad:

```
? zobraz pr poz
? zobraz bezPr poz
```

- `prvý` *`zoznam`* – vráti prvý prvok zoznamu, skrátene `pr`,
- `bezPrvého` *`zoznam`* – vráti zvyšok zoznamu bez prvého prvku, skrátene `bezPr`,
- `posledný` *`zoznam`* – vráti posledný prvok zoznamu, skratka `po`,
- `bezPosledného` *`zoznam`* – vráti zvyšok zoznamu bez posledného prvku, skratka `bezPo`,
- `prvok` *`číslo zoznam`* – vráti prvok zoznamu so zadaným poradovým číslom,
- `?prvok` *`zoznam`* – vráti náhodný prvok zoznamu,
- `počet` *`zoznam`* – vráti počet prvkov zoznamu,
- `prázdny?` *`zoznam`* – zistí, či je zoznam prázdny.

Na vytváranie zoznamov používame operácie:

- `zoznam` *`prvok1 prvok2`* – vytvorí zoznam s dvomi prvkami,
- `vložPrvý` *`prvok zoznam`* – na začiatok zoznamu vloží prvok, skratka `vložPr`,
- `vložPosledný` *`prvok zoznam`* – na koniec zoznamu vloží prvok, skratka `vložPo`.

Projekt Zbieranie bodov

Vytvorme projekt, v ktorom budeme pri ťahaní korytnačky zbierať jej pozície do zoznamu, aby sme potom mohli jej pohyb po tej istej dráhe zopakovať. V projekte budú dve korytnačky: prvá (`k1`) bude kresliť cestu ťahaním myšou. Druhá korytnačka (`k2`) prejde po tejto ceste podľa zozbieraných pozícií prvej korytnačky.

Korytnačke `k1` zapnime automatické ťahanie. Pozície korytnačky `k1` budeme zbierať do zoznamu, ktorý uložíme do globálnej premennej. Premennú vytvoríme príkazom `urob`.

`urob` *`meno_premennej hodnota_premennej`* – premennej so zadaným menom priradí zadanú hodnotu

Meno premennej je slovo, uvádza sa s úvodzovkou na začiatku. Hodnota takto vytvorenej premennej sa uvádza s dvojbodkou na začiatku ako pri parametroch procedúr. Vytvoríme premennú `cesta` a priradíme jej zatiaľ prázdny zoznam.

```
? urob "cesta []
```

Do tohto zoznamu budeme vkladať pri ťahaní korytnačky `k1` jej pozície. Operáciou `vložPo` vytvoríme zoznam, v ktorom na koniec zoznamu v premennej `cesta` vložíme aktuálnu pozíciu ťahanej korytnačky, a tento zoznam priradíme do premennej s menom `"cesta`. Definujme korytnačke `k1` udalosť **priŤahaní** takto:

priŤahaní: `urob "cesta vložPo poz :cesta čakaj 50`

Príkaz `čakaj` slúži na to, aby sa pozície do zoznamu nepridávali príliš často, len v určitých časových intervaloch. Teraz korytnačkou `k2` prejdime po nakreslenej ceste:

```
? odteraz "k2
? nechPoz pr :cesta
? opakuj počet :cesta [nechPoz prvok poč :cesta čakaj 50]
```

Aby bol pohyb bicyklistu krajší, meňme aj jeho smer príkazom `nechSmer` na smer jazdy. Operáciou `smerK` zistíme smer k nasledujúcemu prvku zoznamu `cesta`:

```
? opakuj počet :cesta [nechSmer smerK prvok poč :cesta nechPoz prvok
poč :cesta čakaj 50]
```

Druhá korytnačka môže mať tvar bicyklistu, ktorý ide po ceste nakreslenej prvou korytnačkou.

Zmeňte tvar korytnačky `k2` v jej rodnom liste.

Tvar korytnačky je údaj typu *obrázok*, ktorý tiež môžeme uložiť do premennej. Je to postupnosť záberov, z ktorých každý môže mať niekoľko fáz. Obrázky sa ukladajú do súborov typu *.lgf* alebo do premenných v projekte.

Vyskúšajte:

```
? urob "šašo k2'tvar
```

a v okne **Pamäť** si prezrite obsah premennej `šašo`.

Zadanie 5	Pridajte tlačidlo na zmazanie stránky, ktorým zároveň vyprázdnite zoznam `cesta`, a tlačidlo na prejdenie cesty bicyklistom.
Zadanie 6*	Upravte projekt takto: Pridajte tlačidlá **tam** a **späť**, ktoré budú spúšťať a zastavovať jazdu bicyklistu po ceste jedným a druhým smerom.
Návod na riešenie	Vytvorte dva zoznamy bodov – `pred` a `za` bicyklistom. Korytnačka `k1` bude zbierať body do zoznamu `pred`. Bicyklista bude pri jazde tam vyberať zo zoznamu `pred` prvý prvok a ukladať ho na začiatok zoznamu `za`. Pri jazde späť vyberá prvý prvok zo zoznamu `za` a vkladá ho naspäť na začiatok zoznamu `pred`. Použite operácie `prvý`, `bezPrvého`, `vložPrvý`. Namiesto cyklu s pevným počtom opakovaní použite na pohyb bicyklistu proces, ktorý môžete zastaviť prepínačom pri vypnutí.

Slová, zoznamy slov

Slovo je prázdna alebo neprázdna postupnosť znakov. Môže obsahovať písmená, čísla alebo iné znaky okrem niektorých znakov so špeciálnym významom, napríklad medzera, bodkočiarka, znaky operácií a podobne. Na prácu so slovami môžeme použiť väčšinu operácií, ktoré sa používajú pri práci so zoznamami.

Slová a zoznamy slov môžeme vypisovať do grafickej plochy korytnačkou.

`text` *`slovo`* alebo `text` *`zoznam_slov`* – vypíše slovo alebo zoznam slov (vetu) vpravo od aktuálnej pozície korytnačky jej aktuálnou farbou pera a písmom.

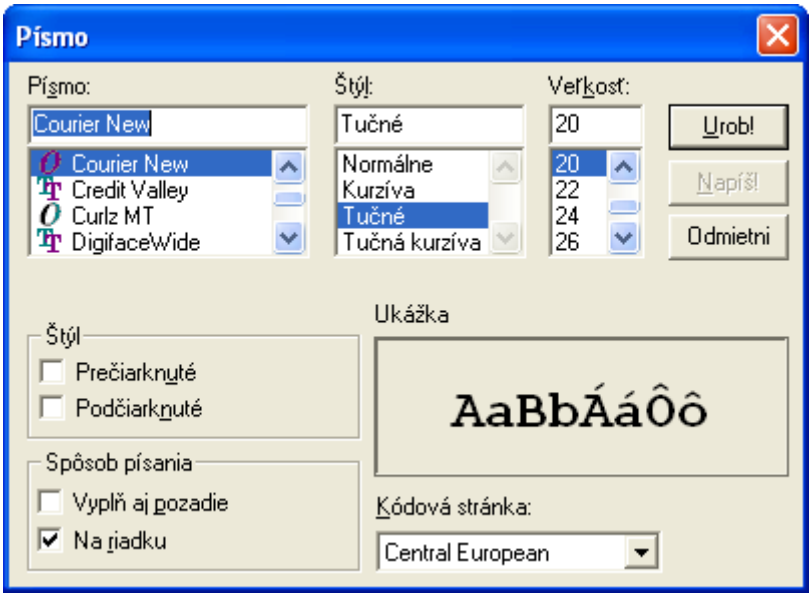
`textSPosunom` *`slovo`* alebo `textSPosunom` *`zoznam_slov`* – to isté ako `text`, ale korytnačka sa pri písaní textu posunie na jeho koniec, skratka `textSP`.

Čísla sú tiež slová zložené len z číslic, prípadne desatinnej bodky alebo znamienok +,-. Môžeme ich zadávať ako parametre operácií so slovami.
Vyskúšajte:

```
? zo posledný 2009
? zo počet 2009
```

nechPísmo *písmo* – nastaví parametre písma, ktorým bude korytnačka písať.

Parameter príkazu **nechPísmo** je pomerne zložitý zoznam, ktorý obsahuje údaje o type písma, veľkosti, reze a ďalšie, preto je jednoduchšie nastaviť ho interaktívne pomocou pomôcky k tomuto príkazu. Do príkazového riadka zadáme **? nechPísmo** bez parametra a zobrazí sa dialógové okno – pomôcka (Obrázok 1).



*Obrázok 1 Pomôcka pre príkaz **nechPísmo***

Nastavme písmo podľa obrázka 1. Definujme procedúru **slovoVslove :s**, ktorá vypíše slovo **:s** tak, že prvé a posledné písmeno vypíše čiernou a ostatné písmená v slove červenou farbou. Na vypisovanie použijeme príkaz **textSPosunom** (**textSP**).

Ako parameter zadávajme také slová, ktoré majú význam aj bez prvého a posledného čierneho písmena. Napríklad:
? slovoVslove "platan

platan

Ďalšie slová, ktoré by sa hodili ako parameter príkazu **slovoVslove** (z knihy *Jozefa Pavloviča Lov slov*): KROPAJ, TURNAJ, MUŠKÁT. Vymyslite podobné slová.

```
viem slovoVslove :s
 nechFP "čierna
 textSP pr :s
 nechFP "červena
 textSP bezPr bezPo :s
 nechFP "čierna
 textSP po :s
koniec
```

Ak je nastavené vektorové písmo, môže korytnačka vypisovať text v rôznych smeroch kolmo na svoj aktuálny smer. Vektorové písma sú v pomôcke pre príkaz **nechPísmo** označené *O* alebo TT.

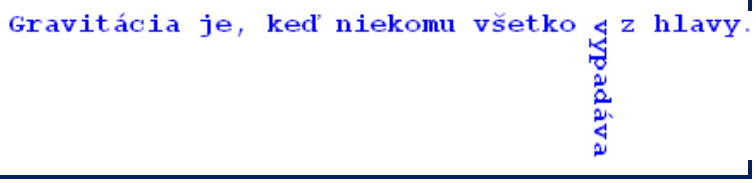
Vypnime korytnačke pero a vyskúšajme text Jozefa Pavloviča:

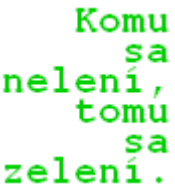
```
? textSP [Žiak je veľmi aktívny, aj vtedy sa] vl 90 vz 15 text
[hlási,] vz 15 vp 90 textSP [keď nevie.]
```

Žiak je veľmi aktívny, aj vtedy sa hlási, keď nevie.

Zadanie 7

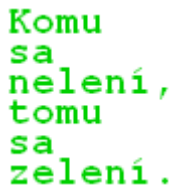
Vypíšte text s vtipným zvýraznením obsahu od Jozefa Pavloviča. Vymyslite podobné texty, v ktorých sa zmenou smeru textu dá zvýrazniť obsah výpovede.





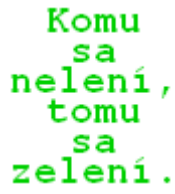
Veľkosť vypísaného textu závisí od zvoleného písma. Zistíme ho operáciou **veľkosťTextu**.

> **veľkosťTextu** *text* – vráti štvorprvkový zoznam parametrov veľkosti zadaného textu pri zvolenom písme. Prvé dva parametre sú šírka a výška textu. Význam posledných dvoch prvkov nájdete v Pomocníkovi.



Definujme príkaz **textVľavo :text**, ktorý vypíše zadaný text vľavo od pozície korytnačky.

```
viem textVľavo :text
 vl 90 do pr veľkosťTextu :text vp 90
 textSP :text
koniec
```



Zadanie 8

Definujte príkaz **textNaStred :text**, ktorý vypíše zadaný text centrovaný okolo pozície korytnačky.

Zadanie 9

Definujte príkazy **stĺpecL**, **stĺpecP**, **stĺpecS**, ktoré vypíšu zadanú vetu (zoznam slov) do stĺpca (každé slovo v novom riadku) so zarovnaním vľavo, vpravo a na stred (pozri výpisy na okraji).

Námet:
Napíšte program, ktorý nakreslí takéto promočné oznámenie. Zvoľte vhodné texty, ktoré budú tvoriť strapce talára.

(Obrázok je inšpirovaný plagátom D.C. Stippa z knihy *Best of Graphis Typography.*)

Ak chceme, aby slovo obsahovalo niektorý zo špeciálnych znakov, napríklad medzeru, uzavrieme celé slovo medzi dva znaky **|**, napríklad:

```
? urob "dveSlová "|okrúhla pečiatka |
```

Teraz môžeme slovo vrátane medzier vypísať po znakoch do kruhu. Definujme príkaz **doKruhu**:

```
viem doKruhu :s
 opakuj počet :s
  [vl 90 text prvok poč :s vp 90
   do 20
   vp 360/(počet :s)]
koniec
```

a vykreslime okrúhlu pečiatku **? doKruhu :dveSlová**

Súbory

Do súborov ukladáme údaje, ktoré by sme chceli mať uložené tak, aby sme ich mohli meniť bez zásahu do projektu. V nasledujúcom projekte si ukážeme prácu s textovým súborom.

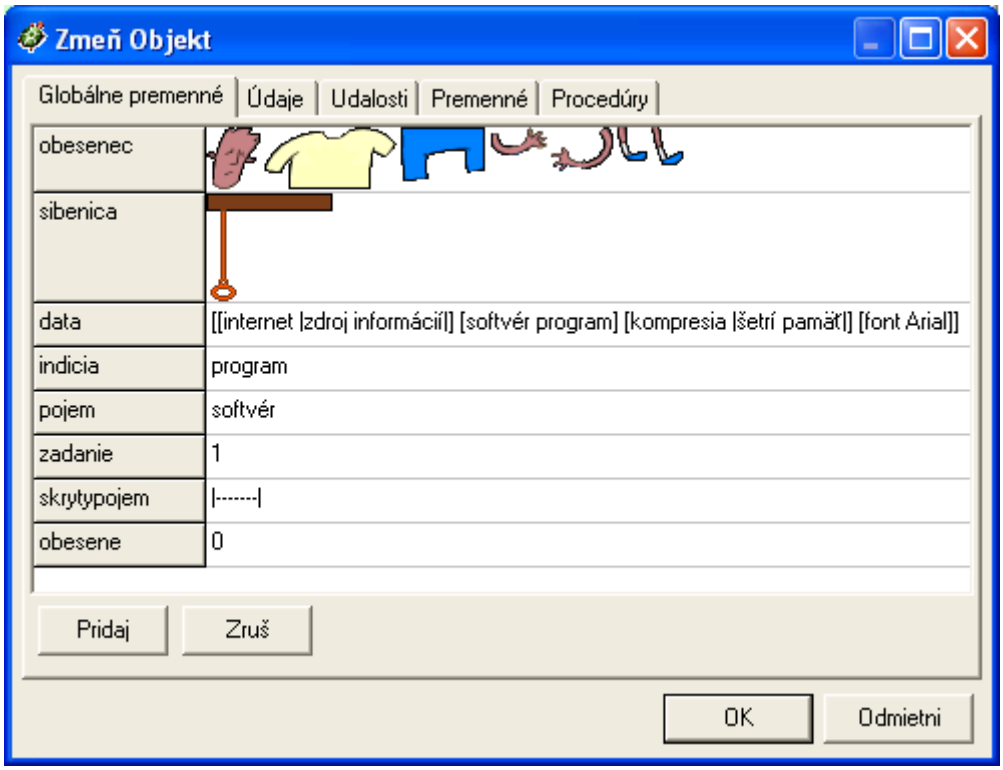
Projekt Šibenica

Medzi demoprojektmi prostredia Imagine Logo nájdeme projekt Obesenec.

V e-learningovom prostredí Moodle si stiahnite upravený projekt s názvom *sibenica.imp*.

Šibenica je hra, v ktorej hráč háda slová podľa indícií postupným zadávaním písmen. Uhádnuté písmeno sa v slove zobrazí, pri neúspešnom pokuse sa na šibenicu postupne kreslí panáčik.

Najprv sa zahrajte hru. Potom preskúmajme projekt. V okne Pamäť zistíme, aké globálne premenné sú v projekte použité (Obrázok 2).



Obrázok 2 Globálne premenné projektu Šibenica.

V zozname `data` sa nachádzajú slová, ktoré sa majú hádať, a indície k nim. Môžeme ho upravovať priamo v okne **Zmeň objekt** alebo pohodlnejšie v procedúre `naplnData`. Zoznam obsahuje dvojice slov `[pojem indícia]`.

Pripravte do projektu Šibenica sadu pojmov a indícií zo svojho druhého aprobačného predmetu alebo na ľubovoľnú inú tému.

```
viem naplnData
 urob "data
  [[internet |zdroj informácií|]
   [softvér program]
   [kompresia |šetrí pamäť|]
   [font Arial]]
koniec
```

Zadanie 10	Upravte zoznam v premennej `data` v procedúre `naplnData` a stlačením tlačidla **znovu** spustite novú hru s upravenými slovami a indíciami.

Načítajme teraz údaje do premennej `data1` zo súboru. Vytvorme textový súbor *data.txt*, ktorý bude obsahovať údaje v tvare:

internet
zdroj informácií
softvér
program ...

Údaje načítame pomocou operácie **čítajTextovýSúbor adresaProjektu "data.txt**, ktorej výsledok uložíme príkazom **urob** do premennej **data1**.

> **čítajTextovýSúbor** ***meno_súboru*** – vráti zoznam slov, v ktorom každé slovo predstavuje jeden riadok zadaného textového súboru.
>
> **adresaProjektu** ***meno_súboru*** - vráti zadané meno súboru rozšírené o úplnú cestu k priečinku, v ktorom sa nachádza aktuálny projekt.

Zoznam v premennej **data1** po načítaní zo súboru bude mať tvar:

```
[ internet |zdroj informácií| softvér program kompresia |šetrí pamäť|
font Arial ]
```

Slová v zozname zoskupíme do dvojíc **[*pojem indícia*]**. Použijeme pomocné lokálne premenné:

```
viem naplnData
 urobTu "data1 čítajTextovýSúbor adresaProjektu "data.txt
 urob "data []
 opakuj (počet :data1)/2
  [urobTu "pojem prvý :data1
   urobTu "indícia prvok 2 :data1
   urob "data1 bezPr bezPr :data1
   urob "data vložPo
                zoznam :pojem :indícia
                :data]
koniec
```

Príkaz **urobTu** vytvorí v procedúre *lokálnu premennú*, ktorá po skončení výpočtu procedúry zaniká.

V Pomocníkovi programu Imagine vyhľadajte ďalšie operácie na prácu so slovami a zoznamami a porozmýšľajte, ako by sa dal upraviť zoznam **data** pomocou iných operácií ako v našom riešení.

Zadanie 11

V e-learningovom kurze k modulu si stiahnite projekt *milionar.imp*. V projekte sa zo zoznamu `data` v tvare:

```
[[otázka správna_odpoveď nesprávna_odpoveď1
  nesprávna_odpoveď2 nesprávna_odpoveď3]… ]
```

generujú v náhodnom poradí otázky a v náhodnom poradí štyri odpovede na ne. Hráč si zvolí jednu odpoveď. Ak je správna, pri kliknutí na zvieratko v pravej časti stránky mu stúpne grafická stupnica správnych odpovedí. Ak nie, klesá v stupnici na nulu. Otázky a odpovede môžete editovať v procedúre `naplnData`. Upravte projekt tak, aby sa údaje do premennej `data` načítavali z textového súboru.

Čo sme sa naučili

- kresliť pomocou súradníc a príkazu **nechPoz**,
- používať operácie s vektormi,
- vytvoriť a meniť hodnotu globálnej a lokálnej premennej pomocou príkazu **urob**, **urobTu**,
- používať operácie so zoznamami: **vložPrvý**, **vložPosledný**, **prvý**, **posledný**, **bezPrvého**, **bezPosledného**, **prvok**, **počet**,
- vypisovať text korytnačkou do grafickej plochy príkazmi **text**, **textSPosunom**,
- nastavovať písmo príkazom **nechPísmo**,
- zisťovať veľkosť textu operáciou **veľkosťTextu**,
- načítať text z textového súboru operáciou **čítajTextovýSúbor**.

Kapitola 2: Rôzne objekty v Imagine

Korytnačka ako objekt

Imagine je objektovo orientované prostredie. Všetky súčiastky a prvky prostredia, ktoré sme doposiaľ používali, sú objektmi.

Korytnačka je objekt, ktorý má svoje štandardné atribúty (základné nastavenia, premenné), procedúry a udalosti. Vieme, že ich môžeme meniť a vytvárať pomocou rodného listu alebo príkazmi. Poznáme napr. nastavenia smer, pozícia, tvar, nastavenie pera, viditeľnosti, zväčšenia tvaru, automatické ťahanie, domovská pozícia. Vieme pridávať rôzne udalosti: priKliknutí, priĽavomHore, priZrážke a pod.

Pre každý atribút má korytnačka nastavenie (premennú) a procedúru, ktorou meníme hodnotu premennej. Napríklad aktuálna pozícia korytnačky sa pamätá v premennej `poz`, ktorej hodnotu môžeme zmeniť príkazom (procedúrou) `nechPoz`. Zhrňme si atribúty korytnačky, ktoré už poznáme:

Aktuálne hodnoty nastavení (premenných) korytnačky môžeme zobraziť príkazom `zo`. Napríklad:
```
? zo smer
```

Atribút	nastavenie (premenná)	Procedúra na zmenu hodnoty nastavenia
farba pera	*fp*	*nechFp*
farba výplne	*fv*	*nechFv*
hrúbka pera	*hp*	*nechHp*
smer	*smer*	*nechSmer*
pozícia	*poz*	*nechPoz*
x-ová súradnica	*xSur*	*nechXSur*
y- ová súradnica	*ySur*	*nechYSur*
tvar	*tvar*	*nechTvar*
zväčšenie	*zväčšenie*	*nechZväčšenie*
zapnutie / vypnutie pera	*pero*	*ph pd*
ponuka klávesov	*ponukaKlávesov*	*nechPonukaKlávesov*
záber	*záber*	*nechZáber*

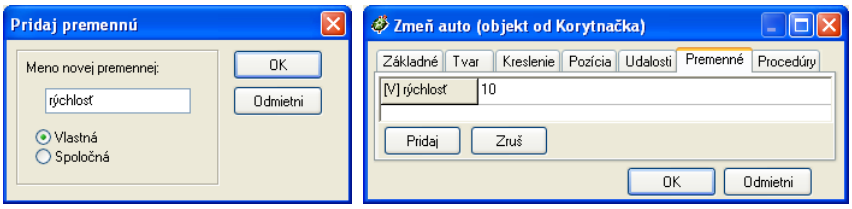
Každej korytnačke však môžeme zadefinovať aj tzv. vlastné premenné a procedúry. Vlastné preto, že danú premennú či procedúru vidí a môže používať iba príslušná korytnačka. Ich význam a použitie si ukážeme v projekte Živý obraz.

Živý obraz 1 (vlastné premenné a procedúry)

V e-learningovej podpore k tomuto modulu nájdete projekt *zivyObraz1.imp*.

V živom obraze z modulu Programovanie 1 sme mali bicyklistu, kvet a auto. Vedeli sme ich rozhýbať procesmi, pridať im udalosti. Upravíme tento projekt tak, aby sme rýchlosť auta či bicyklistu mohli meniť jednoduchým spôsobom bez toho, aby sme museli zastavovať a znovu spúšťať proces.

Otvorme projekt *zivyObraz1.imp*. Vytvoríme korytnačke `auto` vlastnú premennú `rýchlosť` s počiatočnou hodnotou napr. `10`. Otvoríme rodný list korytnačky `auto`. V záložke **Premenné** zvolíme **Pridaj**. Zadáme meno premennej – `rýchlosť` – a stlačíme **OK**. Dopíšeme hodnotu premennej `10` a zatvoríme rodný list.



Obrázok 3 Vytvorenie vlastnej premennej korytnačky

Proces hýbania sa pre auto môžeme teraz spustiť takto:

```
? auto'každých 100 [do rýchlosť]
```

Ak chceme zmeniť rýchlosť pohybu auta, stačí zmeniť autu hodnotu premennej `rýchlosť` príkazom `? auto'nechRýchlosť 15`.

> Príkaz `nechVlastná` ***`meno_premennej hodnota`*** vytvorí aktívnej korytnačke novú vlastnú premennú so zadaným menom a hodnotou.
>
> Príkaz `nech`***`MenoPremennej hodnota`*** zmení aktívnej korytnačke hodnotu vlastnej premennej ***`meno_premennej`***.

Definujeme korytnačke `auto` vlastnú procedúru `hýbSa`, ktorá bude spúšťať proces pohybu auta. V rodnom liste korytnačky na záložke **Procedúry** zvolíme **Pridaj**, zadáme meno procedúry, v našom prípade `hýbSa` a zadefinujeme ju takto:

```
viem hýbSa
 každých 100 [do rýchlosť]
koniec
```

Všimnite si, že v procedúre `hýbSa` sme nepoužili oslovenie korytnačky `auto'`. Vo vlastnej procedúre je totiž automaticky aktívna (oslovená) tá korytnačka, ktorej procedúra patrí. Procedúru teraz zavoláme príkazom:

```
? auto'hýbSa
```

Zadanie 1

Vytvorte korytnačke bicyklista vlastnú premennú `rýchlosť` s nejakou počiatočnou hodnotou. Tiež jej definujte vlastnú procedúru `hýbSa`, ktorá odštartuje proces bicyklovania.

Aj pre korytnačku `kvet` definujeme vlastnú procedúru `hýbSa` – kvet bude postupne rásť. Keď dosiahne dvojnásobnú veľkosť, zmenší sa na desatinu svojej originálnej veľkosti. Na rast využijeme zmenu nastavenia `zväčšenie`.

```
viem hýbSa
 každých 100 [nechZväčšenie zväčšenie+0.05
              ak zväčšenie>1.5 [nechZväčšenie 0.1]]
koniec
```

Spustíme pohyby všetkých troch korytnačiek:

```
? bicyklista'hýbSa auto'hýbSa kvet'hýbSa
```

Pridajme kvetu reakciu na udalosť **priKliknutí** – kvet sa presunie na náhodné miesto. Tentoraz nevyužijeme rodný list, ale udalosť definujeme príkazom:

```
? kvet'nechUdalosť "priKliknutí [nechPoz ?]
```

Tip: Vlastnú premennú `rýchlosť` sme autu mohli vytvoriť aj pomocou príkazu:
`? auto'nechVlastná "rýchlosť 10`

Vlastné premenné korytnačky môžeme vidieť v okne Pamäť, ak „rozbalíme“ korytnačku (kliknutím na znamienko plus pri mene korytnačky) a potom rozbalíme Premenné.

Hodnotu vlastnej premennej môžeme meniť až vtedy, keď premenná existuje – teda keď sme ju vytvorili v rodnom liste alebo príkazom `nechVlastná`!

Tip: Vlastnú procedúru `hýbSa` pre korytnačku `auto` môžeme vytvoriť aj príkazom:
`? auto'uprav "hýbSa`

nechUdalosť ***názov_udalosti reakcia*** – definuje pre aktívnu korytnačku reakciu na udalosť ***názov_udalosti***. ***Reakcia*** je zoznam príkazov, ktoré sa majú vykonať, keď nastane udalosť ***názov_udalosti***.

Pridajme autu reakciu na udalosť **priKliknutí** – auto prejde do vedľajšieho pruhu:

```
auto'nechUdalosť "priKliknutí [ak2 YSur>0 [nechYSur -15][nechYSur 45]]
```

Príkaz `ak2` môžeme použiť aj ako operáciu:

```
auto'nechYSur
 ak2 YSur>0 [-15][45]
```

Zadanie 2

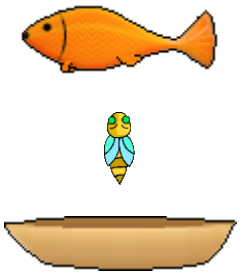
Definujte bicyklistovi udalosť **priKliknutí** – bicyklista sa otočí do protismeru. Nevyužívajte rodný list!

Pridajme do projektu posúvač, ktorým budeme určovať rýchlosť auta. Jeho reakciou na udalosť **priZmene** bude: `auto'nechRýchlosť p1`. Definujeme ju príkazom:

```
p1'nechUdalosť "priZmene [auto'nechRýchlosť p1]
```

Živý obraz 2 (vytváranie korytnačiek príkazom)

Vyvinieme ďalšiu verziu projektu Živý obraz s nasledujúcimi vlastnosťami:



- v pozadí bude obrázok *rybník.lgf*,
- kliknutím myši budeme pridávať do obrazu ryby (*ryba.lgf*), včielky, (*potvorka10.lgf*) a lodičky (*lodka.lgf*), a to podľa toho, na aké miesto stránky sme klikli:
  - keď klikneme na vodu, do obrazu sa pridá ryba alebo lodička,
  - keď klikneme nad vodou, pridá sa do obrazu včielka,
- každá ryba, včielka a loďka sa pohybuje smerom na východ alebo západ (začne sa pohybovať hneď, ako vznikne),
- pri kliknutí na ľubovoľný objekt v obraze sa tento otočí opačným smerom,
- každý z objektov v obraze má vlastnú rýchlosť, ktorá sa určí náhodne už pri vytvorení objektu v obraze,
- všetky objekty v obraze môžeme zrušiť nejakým tlačidlom.

Doposiaľ sme vytvárali korytnačky pomocou nástroja nová korytnačka z panelu nástrojov. Teraz chceme „naprogramovať“, aby nová korytnačka vznikla vtedy, keď klikneme myšou do stránky (bez zvolenia nástroja nová korytnačka). Využijeme udalosť stránky **priKliknutí** – ako reakciu na ňu zadefinujeme vytvorenie korytnačky. Musíme sa preto naučiť vytvárať korytnačky príkazom.

Otvoríme nový projekt, zrušíme existujúcu korytnačku a vyskúšame príkaz:

```
? nová "korytnačka []
```

Tým sme na stránke vytvorili novú korytnačku (`k1`). Vznikla na pozícii `[0 0]`, má zapnuté pero čiernej farby, klasický tvar korytnačky a smer 0. Už pri vytváraní môžeme korytnačke niektoré nastavenia zmeniť. Postupne vyskúšame tieto príkazy:

- `? nová "korytnačka [poz [-200 -100]]`
  Vytvorí novú korytnačku (`k2`) na pozícii `[-200 -100]`.
- `? nová "korytnačka [poz [150 20] smer 90 fp zelená hp 2]`
  Vytvorí novú korytnačku (`k3`) na pozícii `[150 20]` so smerom 90, zelenou farbou pera a hrúbkou pera 2. Skúste `? k3'do 80`.
- `? nová "korytnačka [meno vajce pero ph tvar vajce.lgf poz ? smer ? rýchlosť (1 + náhodne 10)]`
  Vytvorí novú korytnačku s menom `vajce`, s vypnutým perom, s tvarom zo súboru *vajce.lgf* na náhodnej pozícii, s náhodným smerom a vlastnou premennou rýchlosť s náhodnou hodnotou z intervalu 1..10.
- `? nová "korytnačka [pero ph tvar motyl.lgf poz ? autoŤahanie áno]`
  Vytvorí novú korytnačku (`k4`) s vypnutým perom, s tvarom zo súboru *motyl.lgf*, na náhodnej pozícii so zapnutým automatickým ťahaním. Skúste motýľa ťahať.

Korytnačky, ktoré sme vytvorili týmito príkazmi, nie sú aktívne.

Príkaz **nová "korytnačka** ***zoznam_nastavení*** vytvorí novú korytnačku s nastaveniami danými v ***zozname_nastavení***. V ***zozname_nastavení*** za sebou nasledujú názov nastavenia a hodnota príslušného nastavenia. Poradie nastavení je ľubovoľné. Nastavenia, ktoré neuvedieme, budú mať „štandardnú“ hodnotu. **Ak chceme ako hodnotu nastavenia použiť výsledok nejakej operácie či výrazu, musí byť uvedená v zátvorkách.**

Zoznam nastavení musí mať vždy párny počet prvkov!

Korytnačky môžeme príkazom aj zrušiť. Skúšajme nasledujúce príkazy a pozorujme zmeny na stránke:

```
? zrušObjekt "k1
? zrušObjekt [vajce k4]
? zrušObjekt všetky
```

Príkaz **zrušObjekt** ***menoObjektu*** zruší objekt s príslušným menom.

Príkaz **zrušObjekt** ***zoznamObjektov*** zruší objekty v ***zoznameObjektov***.

Operácia **všetky** vráti zoznam mien všetkých korytnačiek.

Vráťme sa k nášmu živému obrazu. Natiahnime na stránku pozadie zo súboru *rybnik.lgf*. Už vieme, ako budeme korytnačky vytvárať. Najskôr si však premyslime, aké budú mať nastavenia, udalosti, vlastné premenné prípadne procedúry:

Pozadie môžeme na stránku natiahnuť aj príkazom: **pozadieZoSúboru** ***súbor***, napr.
```
? pozadieZoSúboru
   "|pozadia\rybnik.lgf|
```

- všetky korytnačky preberú pozíciu od pozície myši – tú nám vráti operácia **pozMyši**,
- všetky korytnačky budú mať vypnuté pero,
- smer korytnačiek vyberieme ako náhodný prvok zo zoznamu **[90 270]** pomocou **?prvok [90 270]**,
- všetky korytnačky budú mať vlastnú premennú rýchlosť, ktorej hodnotu na začiatku nastavíme napr. výrazom **(1 + náhodne 10)**,
- korytnačky, ktoré vzniknú nad vodou, budú mať tvar zo súboru *potvorka10.lgf*,
- korytnačky, ktoré vzniknú vo vode, budú mať tvar zo súboru *lodka.lgf* alebo *ryba.lgf* – môžeme ho teda vybrať výrazom **?prvok [ryba.lgf lodka.lgf]**,
- všetky korytnačky budú mať udalosť **priKliknutí** s reakciou **vp 180**.

Definujme nový príkaz **pridaj** - na základe pozície kliknutia myšou (presnejšie len y-ovej súradnice) sa rozhodneme, či vytvoríme včelu, alebo loď či rybu. Každej novej korytnačke musíme ešte definovať udalosť **priKliknutí** tak, aby sa otočila a spustiť pre ňu proces. Urobíme to hneď po tom, ako ju vytvoríme. Na oslovenie novej korytnačky využijeme operáciu **novéMeno**.

Operácia **novéMeno** vráti meno naposledy vytvorenej korytnačky.

```
viem pridaj
  ak po pozMyši > 50
    [nová "korytnačka
       [poz (pozMyši) tvar potvorka10.lgf pero ph
        smer (?prvok [90 270]) rýchlosť (1 + náhodne 10)]]
  ak po pozMyši < 30
    [nová "korytnačka
       [poz (pozMyši) tvar (?prvok [ryba.lgf lodka.lgf]) pero ph
        smer (?prvok [90 270]) rýchlosť (1 + náhodne 10)]]
  pre novéMeno
          [nechUdalosť "priKliknutí [vl 180]
           každých 100 [do rýchlosť]]
koniec
```

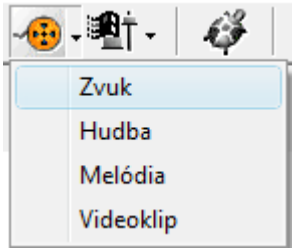
Definujme **stránke** udalosť **priKliknutí** – reakciou nech je príkaz `pridaj`.

Zadanie 3	Upravte príkaz `pridaj` tak, aby pri kliknutí do stránky na miesto s y-ovou súradnicou medzi 30 a 50 vznikla korytnačka žaba (*zaba.lgf*). Aké by mala mať nastavenia?

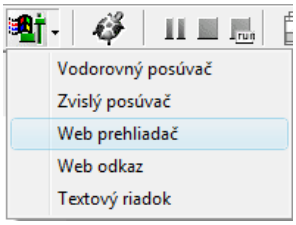
Ďalšie objekty v Imagine

Okrem korytnačky sme pracovali aj s ďalšími objektmi: posúvač, text, tlačidlo, stránka. Tie majú tiež svoje atribúty (nastavenia), udalosti, premenné a procedúry. Keď sme chceli niečo prikázať niektorej korytnačke (okrem prvej aktívnej), museli sme ju osloviť. Rovnako oslovujeme aj iné objekty. Stretli sme sa s tým už napr. pri texte, ktorý zobrazoval hodnotu posúvača – `text1'nechHodnota p1`.

Vyskúšajte si aj ďalšie súčiastky, ktoré ponúka Imagine (napr. zvuk, hudbu melódiu, videoklip):



alebo Web prehliadač.



Ľubovoľný objekt môžeme vytvoriť aj príkazom.
`nový rodič zoznam_nastavení`

Napr. tlačidlo vytvoríme príkazom
`? nové "tlačidlo []`,
posúvač vytvoríme príkazom
`? nový "posúvač []`,
textové políčko vytvoríme príkazom
`? nový "text []`.

Tip: Viacero stránok môžeme použiť napríklad na vytvorenie jednoduchej prezentácie.

Otvorme nový projekt. Pridajme do stránky posúvač `p1`. Jeho pozíciu môžeme zmeniť príkazom:

`? p1'nechPoz [100 100]` alebo

`? pre "p1 [nechPoz [100 100]]`

Pridajme do stránky tlačidlo `t1` . Pozíciu mu zmeníme príkazom:

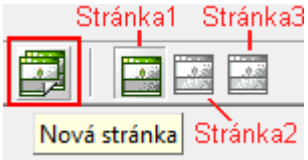
`? pre "t1 [nechPoz [200 100]]`

Zadanie 4	Pridajte do projektu objekty, s ktorými ste už skôr pracovali a porovnajte ich atribúty (skúmajte záložky **Základné** a **Vzhľad** rodného listu). Ktoré atribúty majú spoločné? Poznáte príkazy, ktorými nastavujeme používané atribúty a ktorými zistíme ich hodnotu? Ktoré udalosti majú spoločné?

Stránka

Všetky objekty, s ktorými sme dosiaľ pracovali, sme umiestňovali do stránky. Stránka je tiež objektom. Pozrime si jej rodný list - klikneme pravým tlačidlom myši na stránku a z ponuky zvolíme Zmeň Stránka1.

Vidíme, že aj stránka môže mať svoje udalosti, premenné, procedúry a atribúty. Niektoré atribúty má rovnaké ako iné objekty – napríklad má atribúty pre veľkosť podobne ako tlačidlo či posúvač. V projekte môžeme mať viac stránok, no vidíme vždy len jednu. Novú stránku pridáme nástrojom **Nová stránka**.



Obrázok 4 Nástroj Nová stránka a „zoznam“ stránok.

Zoznam stránok vidíme na paneli nástrojov alebo v okne Pamäť. Klikaním na ikonu stránky v paneli nástrojov si vyberáme, ktorú stránku chceme vidieť. Na inú stránku môžeme prejsť aj pomocou príkazu `nechAktívnaStránka menoStránky`, napríklad: `nechAktívnaStránka "stránka2.` Príkazy `zmaž` a `znovu` patria k objektu stránka a vykonajú sa pre aktívnu (tú, ktorú vidíme) stránku. Aj objekt stránka môžeme osloviť napríklad `? stránka1'zmaž.`

Zadanie 5	Vytvorte k projektu Živý obraz 2 úvodnú stránku, na ktorej bude uvedený názov projektu, autor a návod, ako tvoriť živý obraz. Umiestnite na stránku tlačidlo ***Začni***, ktoré nás presunie na stránku, kde môžeme vytvárať živý obraz.

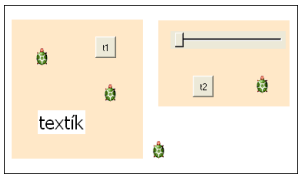
Papier

Papier je objekt veľmi podobný stránke. Môžeme ho zmazať, nastaviť mu farbu pozadia, či natiahnuť na pozadie obrázok. Na papieri (podobne ako na stránke) môžu žiť korytnačky a robiť tam všetko to, čo na stránke. Korytnačka žijúca na papieri žije iba na ňom a nemôže z neho odísť na stránku. Na papier môžeme tiež umiestniť tlačidlá, posúvače, texty a ďalšie súčiastky.

Papier má podobné atribúty ako stránka. Keď chceme, aby papier vykonal nejaký príkaz, musíme ho osloviť. Napríklad:

```
? papier1'zmaž
? papier1'znovu
```

Papier umiestňujeme na stránku nástrojom **Nový papier**  . Papier, rovnako ako ostatné objekty, má svoj rodný list. Porovnajme ho s rodným listom stránky. Aké atribúty má navyše? V rodnom liste môžeme okrem iného nastaviť aj viditeľnosť papiera – ak papier skryjeme, skryjú sa aj všetky objekty, ktoré na ňom žijú.



Stránka s dvoma papiermi

Vložme do stránky dva papiere, na každý z papierov pridajme niekoľko korytnačiek a iných súčiastok. Vyskúšajme skryť jednotlivé papiere a sledujme, čo sa stane s ďalšími objektmi. Môžeme využiť aj príkazy **ukážMa** a **skryMa**:

```
? papier1'skryMa
? papier2'skryMa
? papier1'ukážMa
```

Papier môžeme vytvoriť aj príkazom:

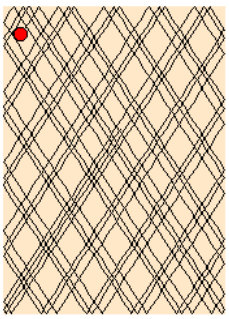
```
? nový "papier
      zoznam_nastavení
```

Projekt Loptička na papieri

Otvorme nový projekt.

1. Vložíme na stránku jeden papier. Na tento papier vložíme novú korytnačku a zmeníme jej tvar na krúžok (obrázok zo súboru *kruzok.lgf*).
2. Korytnačke nastavíme náhodný smer a spustíme proces `k2'každých 100 [do 10].` Vidíme, že korytnačka zostáva na papieri. Keď je pri okraji papiera a má ho opustiť, ukáže sa na opačnom konci.
3. Korytnačke vytvoríme vlastnú premennú **rýchlosť** s nejakou hodnotou a spustíme proces: `k2'každých 100 [do rýchlosť]`.
4. Vložíme na stránku ďalší papier. Skopírujme korytnačku z prvého papiera do schránky a kópiu prilepíme na druhý papier.
5. Otvoríme rodný list novej korytnačky na záložke **Pozícia**. Dolu nájdeme nastavenie oblasti – typ oblasti rozhoduje o tom, ako sa korytnačka správa na okrajoch stránky či papiera, v ktorom žije. Štandardne je nastavená oblasť **Dokola.** Nastavíme typ oblasti **S odrazom** a spustíme pre korytnačku proces `k3'každých 100 [do rýchlosť]`.
6. Porovnáme správanie oboch korytnačiek.

Zistite, aké ďalšie nastavenia môžeme použiť pre nastavenie oblasti korytnačky a ako ovplyvňujú jej správanie.



Práca s viacerými korytnačkami a objektmi

V tejto časti si ukážeme niekoľko príkladov, ako vytvárať veľa korytnačiek s istými vlastnosťami. Začneme generovaním (vytvorením) niekoľkých korytnačiek pravidelne rozmiestnených na priamke. Budú mať tvar krúžku (*kruzok.lgf*), typ oblasti **sOdrazom** a budú pomenované číslami. Definujeme príkaz **naPriamke**:

```
viem naPriamke :n
 zrušObjekt všetky
 opakuj :n [nová "korytnačka
             [meno (poč) tvar kruzok.lgf xSur (poč*10-360)
              typOblasti sOdrazom]]
koniec
```

```
? naPriamke 72
```


Keď oslovíme korytnačky, príkazom `pre všetky […]`, tak zadané príkazy vykonávajú všetky korytnačky naraz. Ak je medzi príkazmi operácia alebo vlastná premenná korytnačky, tak sa vráti výsledok operácie prvej aktívnej korytnačky. Ak potrebujeme hodnotu operácie či premennej pre každú korytnačku zvlášť, oslovíme všetky korytnačky **postupne**, pomocou príkazu `preKaždú všetky […]`.

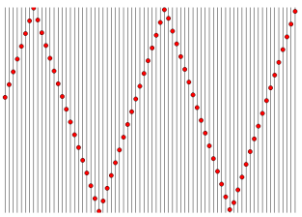
To, že korytnačky sú pomenované číslami, môžeme rôznym spôsobom využiť. Napríklad môžeme každej korytnačke povedať, aby sa pohla dopredu o svoje meno. Nesmieme to však korytnačkám povedať naraz, ale postupne:

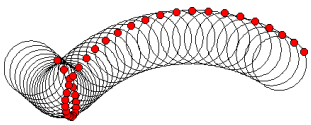
```
? preKaždú všetky [do mojeMeno]
```

> Operácia `mojeMeno` vráti meno korytnačky, ktorá práve vykonáva príkaz.
>
> Príkazom `preKaždú` ***zoznam_korytnačiek*** ***zoznam_príkazov*** **postupne** oslovíme korytnačky zo ***zonamu_korytnačiek*** - jedna za druhou vykonajú príkazy uvedené v ***zozname_príkazov***.

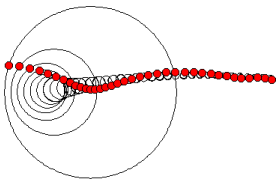
Spustíme proces, ktorý bude tento posun opakovať pravidelne:

```
? každých 10 [preKaždú všetky[do mojeMeno]]
```





Tip: Vypnite korytnačkám pero.



Zadanie 6	Vytvorte korytnačky pomenované číslami a umiestnené na priamke tak, že každá bude mať smer daný desaťnásobkom svojho mena. Tvar korytnačiek bude krúžok. Korytnačkám spustite proces, ktorý ich bude pohybovať po kružnici.

Riešenie	`viem naPriamke2 :n`
	` zrušObjekt všetky`
	` opakuj :n [nová "korytnačka`
	`  [meno (poč) tvar kruzok.lgf`
	`  xSur (poč*10-360) smer (poč*10) ]]`
	`koniec`
	`? naPriamke2 36`
	`? každých 10 [pre všetky [do 2 vp 3]]`

Zadanie 7	Zastavte proces a zmažte stránku príkazom `znova`. Spustite nový proces, ktorý bude korytnačky pohybovať po kružnici tak, že korytnačky sa budú otáčať o veľkosť uhla podľa svojho mena. Pomôcka: využite `preKaždú` a `mojeMeno`.

Teraz skúsime generovať korytnačky pravidelne rozmiestnené na kružnici. Vygenerujme N **skrytých** korytnačiek s náhodnou farbou pera, hrúbkou pera 5 (pozri riadok s označením 1 v príkaze `generuj_vkruhu`). Pozície na kružnici dosiahneme tak, že budeme otáčať pozíciu (vektor) [100 100] postupne o istý uhol (pozri riadok s označením 2) - napr. pri 36 korytnačkách o uhol 10°, 20°, 30°, .. 360°. Na rovnaký uhol nastavíme smer korytnačky (pozri riadok s označením 3).

```
viem generuj_vkruhu :n
 zrušObjekt všetky
 opakuj :n [nová "korytnačka
              [fp ? hp 5 vidno nie                    (1)
               poz (otoč poc*360/:n [100 100])        (2)
               smer (poč*360/:n)]]                    (3)
koniec

? generuj_vkruhu 72
```

Spustíme proces, v ktorom sa budú korytnačky pohybovať po kružnici. Pred každým krokom zmažeme stránku.

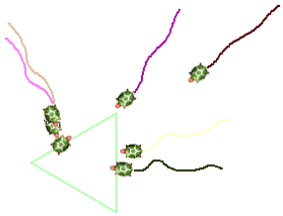
```
? každých 100 [zmaž pre všetky [do 10 vl 10]]
```

Naháňačka korytnačiek

Vytvoríme projekt, v ktorom jedna korytnačka (s menom `0`) bude stále chodiť po rovnakej dráhe, napríklad po rovnostrannom trojuholníku. Niekoľko iných korytnačiek, vytvorených na náhodnom mieste, bude túto korytnačku naháňať. Definujeme príkaz **`naháňačka`**, ktorý vytvorí všetky korytnačky s potrebnými nastaveniami. Počet korytnačiek bude daný parametrom príkazu.

```
viem naháňačka :n
 zrušObjekt všetky
 zmaž
 nová "korytnačka [meno 0 poz [-100 0] smer 60 fp ? hp 2]
 opakuj :n [nová "korytnačka [meno (poč) poz ? fp ? hp 2]]
 0'každých 10 [do 1]
 0'každých 1000 [vp 120]
koniec
```

Naháňanie naprogramujeme ako proces, v ktorom sa pravidelne každá korytnačka okrem 0 posunie o kúsok smerom ku korytnačke 0. Spravíme to tak, že **postupne** každú korytnačku okrem 0 natočíme smerom ku korytnačke 0, a potom posunieme o 2 body. Na natočenie využijeme operáciu **`smerK`**.



> Operácia **`smerK`** ***`meno_korytnačky`*** vráti uhol, na ktorý sa musíme (absolútne) natočiť, aby sme sa pozerali na korytnačku ***`meno_korytnačky`*** (azimut).

Všetky korytnačky okrem 0 **postupne** oslovíme pomocou **`preKaždú bezPrvého všetky ...`**. Proces naháňania teda spustíme nasledovne:

```
? každých 100 [preKaždú bezPrvého všetky [nechSmer smerK 0 do 2]]
```

Spustenie procesu doplníme ako posledný do príkazu **`naháňačka`**:

```
viem nahánačka :n
 ...
 každých 100 [preKaždú bezPrvého všetky [nechSmer smerK 0 do 2]]
koniec
```

Vyskúšame: `? naháňačka 10`

Ďalšie varianty naháňačiek nájdete v doplňujúcom materiáli v e-learningovom kurze v systéme Moodle.

Zadanie 8	Experimentujte s projektom naháňačka korytnačiek: meňte rýchlosť pohybu korytnačiek, tvar dráhy korytnačky 0 atď.

Čo sme sa naučili

- vytvárať vlastné premenné, definovať udalosti a vlastné procedúry korytnačky – interaktívne v režime návrhu i pomocou príkazov,
- vytvoriť korytnačku s danými nastaveniami príkazom **`nová "korytnačka`**,
- zrušiť jednu alebo viac korytnačiek príkazom **`zrušObjekt`**,
- pracovať s viacerými stránkami a papiermi,
- generovať viacero korytnačiek s danými vlastnosťami,
- osloviť korytnačky **postupne** pomocou príkazu **`preKaždú`**,
- využívať operáciu **`novéMeno`** a **`mojeMeno`**.

Ukázali sme si spoločné i odlišné vlastnosti jednotlivých objektov v Imagine a osvojili sme si princíp práce s nimi.

Kapitola3: Kreslené tvary korytnačiek. Triedy

Kreslené tvary korytnačiek

V module Programovanie 1 sme sa naučili, ako zmeniť tvar korytnačky na obrázok zo súboru a ukázali sme si, aký význam majú zábery a fázy v tvare korytnačky. Tvar korytnačky môžeme tiež popísať pomocou niektorých príkazov jazyka Logo, definovaním tzv. **návodu na kreslenie.** Vyskúšajme nasledujúce dva príkazy:

```
? nechTvar [nechFp "modrá nechHp 3 bod 15 do 30]

? nechTvar [nechFp "fialová nechHp 3 do 30 vl 45
            opakuj 2 [vz 10 do 10 vp 90]]
```

Príkaz `nechTvar` *`návod_na_kreslenie`* zmení tvar korytnačky na obrázok definovaný návodom na kreslenie. Návod na kreslenie je zoznam príkazov uzavretý v zátvorkách `[, ]`. Môže obsahovať príkazy na pohyb a otáčanie korytnačky (`do`, `vz`, `vp`, `vl`), príkazy na nastavenie vlastností pera a stavu pera (`nechFp`, `nechHp`, `nechFv`, ..., `ph`, `pd`), riadiaci príkaz `opakuj`, príkazy `nechPoz`, `nechSmer`, `nechPísmo`, `text`, `polygón` a niektoré ďalšie príkazy.

Viac o tom, čo môže obsahovať návod na kreslenie, nájdete v Pomocníkovi.

Výhodou takto definovaného tvaru je, že Imagine ho dokáže automaticky otáčať. Zadajme korytnačke príkaz `? opakuj 36 [vp 10 čakaj 200]`.

Zadanie 1	Zmeňte korytnačke tvar na vrtuľku. Spustite proces, ktorý bude v istom časovom intervale vrtuľku otáčať. 
Riešenie	`? nechTvar [nechHp 3 nechFp "červená opakuj 4 [opakuj 2 [do 20 vp 90] vp 45 do 28 vl 135]]` `? každých 100 [vp 10]`

Projekt Sledujúce oči

Vytvoríme projekt, v ktorom budú dve korytnačky s tvarom očí sledovať kurzor myši.

Otvorme nový projekt. Zmeníme korytnačke tvar na oko – definujme ho takýmto návodom na kreslenie: `? nechTvar [nechHp 2 nechFp "čierna nechFv "belasá12 kruh 40 ph do 12 bod 15]`

Príkaz `kruh` nakreslí kruh s daným polomerom. Farba obrysu tohto kruhu je daná farbou pera korytnačky, farba výplne kruhu je daná farbou výplne korytnačky.

Pre stránku definujme udalosť **priPohybeMyši** takto: `pre "k1 [nechSmer smerK pozMyši].` Pohybujme myšou.

Zmeňte v projekte *Sledujúce oči* reakciu na udalosť **priPohybeMyši** takto: `pre [k1 k2] [nechSmer smerK pozMyši]` a porovnajte s predchádzajúcim riešením.

Kopírovaním korytnačky vytvorme druhé oko a umiestnime ho vedľa prvého. Zmeňme udalosť stránky **priPohybeMyši** tak, aby aj korytnačka `k2` sledovala pozíciu myši:

```
pre "k1 [nechSmer smerK pozMyši] pre "k2 [nechSmer smerK pozMyši]
```

Pohybujme myšou.

Dynamické tvary korytnačiek

V ďalšom projekte si ukážme, ako môžeme zmeniť korytnačke tvar na kruh, ktorého veľkosť nebude konštantná, ale bude závisieť od premennej, resp. od hodnoty posúvača.

Pridajme korytnačke `k1` vlastnú premennú `priemer` s hodnotou `50`. Definujme korytnačke vlastnú procedúru `zmeňTvar`, ktorá zmení jej tvar na kruh s priemerom `priemer`. V príkaze `zmeňTvar` využijeme návod na kreslenie:

```
viem zmeňTvar
 nechTvar ![bod (priemer)]
koniec
```

Všimnime si výkričník a zátvorky `()` v návode na kreslenie. Zapamätajme si, že ak sa v návode na kreslenie nachádza premenná či nejaký výraz, ktorý sa má vyhodnotiť:

a) musíme dať výraz do zátvoriek,
b) pred návod musíme dať výkričník.

Zmeňme korytnačke hodnotu premennej `priemer`: `? k1'nechPriemer 20`. Vidíme, že tvar sa nezmenil. Zmena tvaru sa totiž nedeje automaticky. Ak zmeníme hodnotu niektorej premennej v návode na kreslenie, musíme dať korytnačke príkaz, ktorým znovu prekreslí svoj tvar: `? k1'zmeňTvar`.

Veľkosť tvaru korytnačky môžeme meniť pomocou posúvača. Pridajme na stránku posúvač s nastaveniami: Minimum 10, Maximum 100. Pri posunutí posúvača zmeníme hodnotu premennej `priemer` a zavoláme príkaz na zmenu tvaru korytnačky `k1`. Udalosť **priZmene** teda definujeme takto: `k1'nechPriemer hodnota k1'zmeňTvar`. Vyskúšajme posúvač.

Doplňujúce úlohy ku kresleným tvarom korytnačiek nájdete v e-learningovej podpore k tomuto modulu.

Triedy

Projekt Kocky

Spomínate si na projekt z modulu Programovanie 1, v ktorom sme z kociek vytvárali rôzne stavby? Vľavo sme mali zdrojové kocky, ktoré sme mohli presunúť a odtláčať do stránky. Skúsime tento projekt zdokonaliť – kocky v pravej časti nebudú len odtlačky, ale korytnačky, ktoré budeme môcť stále presúvať, či zahodiť. To nám umožní stavbu kedykoľvek modifikovať.

V upravenom projekte budeme potrebovať dva druhy korytnačiek – kociek:

- Kocky vľavo – akási „zásoba“. Ak na takúto korytnačku klikneme ľavým tlačidlom myši, vytvorí sa jej „kópia“ s rovnakým tvarom na rovnakej pozícii a bude možné ju ťahať.
- Kocky vpravo, ktoré tvoria stavbu. Tieto môžeme presúvať na ľubovoľné miesto na stránke. Ak ich presunieme vľavo do časti zásoba, tak ich zrušíme.

Uvedomme si, že zatiaľ čo korytnačky-zásoba musíme pripraviť hneď na začiatku, pohyblivé korytnačky vzniknú až pri prvom kliknutí na niektorú korytnačku-zásoba, ale popis ich správania musíme mať pripravený už skôr.

Ak v projekte potrebujeme viac druhov korytnačiek, je výhodné definovať pre každý druh samostatnú **triedu korytnačiek**, t.j. akýsi „vzor“, ako sa majú správať všetky korytnačky z danej triedy. Triedu definujeme v Imagine Logu príkazom `nováTrieda`.

> Príkaz `nováTrieda "korytnačka` ***`názov_novej_triedy zoznam_nastavení`*** definuje novú triedu korytnačiek s názvom ***`názov_novej_triedy`*** (slovo začínajúce úvodzovkou) s nastaveniami špecifikovanými v zozname nastavení. Zoznam nastavení je podobný ako zoznam nastavení pri vytváraní novej korytnačky.

Trieda je základným stavebným prvkom v objektovo-orientovanom programovaní. Spája v sebe údaje a operácie nad týmito údajmi.
Trieda je akýsi vzor, podľa ktorého sa vytvárajú jednotlivé objekty.

Otvorme nový projekt a zrušme korytnačku. Najskôr vytvoríme triedu pre korytnačky vpravo, ktoré tvoria stavbu - nazveme ju `kocka`. Korytnačky z tejto triedy budeme ťahať myšou, pričom by nemali kresliť čiaru, takže musia mať zapnuté automatické ťahanie a pero hore. Obe tieto nastavenia môžeme uviesť už pri vytváraní triedy `kocka` - v zozname nastavení.

```
? nováTrieda "korytnačka "kocka [autoŤahanie áno pero ph]
```

Trieda `kocka` v okne Pamäť

Otvorme okno Pamäť a pozrime si novú triedu. Rozbaľme triedu `kocka` (kliknutím na plus pri názve), a potom rozbaľme jej Premenné (kliknutím na plus pri Premenné).

Pridáme triede `kocka` udalosť **priĽavomHore** – chceme, aby sa korytnačka zrušila, ak

Môžeme vytvárať nielen nové triedy korytnačiek, ale napr. aj nové triedy tlačidiel, nové triedy posúvačov, atď. Vo všeobecnosti vyzerá príkaz takto: `nováTrieda` ***`rodič názov_triedy zoznam_nastavení`***, kde ***`rodič`*** je trieda, z ktorej odvádzame novú triedu. Napr.

```
? nováTrieda "tlačidlo
   "veľkéTlačidlo
   [veľkosť [100 50]]
```

alebo

```
? nováTrieda "posúvač
   "môjPosúvač
   [min 1 max 10]
```

sme ju pustili v ľavej „zásobovacej“ časti, čo ľahko zistíme podľa jej x-ovej súradnice. Otvoríme okno Pamäť a dvojklikneme na triedu `kocka`. V záložke **Udalosti** zvolíme **Pridaj**, v zozname vyberieme udalosť **priĽavomHore** a stlačíme tlačidlo **OK**. Reakciu na udalosť definujeme takto: `ak xSur <-300 [zrušObjekt mojeMeno]`.

Nezabúdajme, že trieda `kocka` je iba popis nastavení a správania, nie konkrétny objekt. Ak chceme pracovať s korytnačkami z triedy `kocka`, musíme ich najskôr vytvoriť. Obyčajné korytnačky sme vytvárali príkazom `nová "korytnačka [...]`. Korytnačky z triedy `kocka` vytvoríme príkazom `nová "kocka [...]`.

> Príkaz `nová` ***`názov_triedy zoznam_nastavení`*** vytorí objekt (inštanciu) triedy ***`názov_triedy`*** s nastaveniami špecifikovanými v ***`zozname nastavení`***.

Vyskúšajme: `? nová "kocka [poz ?].` Skúsme korytnačku ťahať. Potiahnime ju do ľavej časti stránky. Zrušila sa?

Uložme projekt. Teraz sa budeme venovať druhej skupine korytnačiek – zásobe. Vytvoríme pre ne triedu `zásoba`. Aj táto trieda by mala mať vypnuté pero.

```
? nováTrieda "korytnačka "zásoba [pero ph]
```

Otvorme okno Pamäť, pridajme triede `zásoba` udalosť **priĽavomDolu** a definujme reakciu na túto udalosť takto:

```
nová "kocka [poz (poz) tvar (tvar)] pre novéMeno [odštartujŤahanie]
```

Pripomeňme si:

Ak poznáme konkrétne meno korytnačky, môžeme ju oslovovať použitím apostrofov, napr. `k1 do 20`.
Ak chceme osloviť korytnačku, ktorej meno máme uložené v nejakej premennej (napr. `:mojaKor`) alebo nám ho vracia nejaká operácia (napr. `novéMeno`, `mojeMeno`), musíme použiť príkaz `pre`, napr.

```
pre novéMeno [do 20]
```

alebo

```
pre :mojaKor [vp 90].
```

Prvým príkazom `nová "kocka [poz (poz) tvar (tvar)]` hovoríme, že sa má vytvoriť korytnačka z triedy `kocka`, na rovnakej pozícii a s rovnakým tvarom ako korytnačka, na ktorú sme klikli. Príkazom `pre novéMeno [odštartujŤahanie]` sme práve vytvorenej korytnačke povedali, že ju odteraz budeme ťahať myšou.

Vytvorme päť korytnačiek z triedy zásoba:

```
? nová "zásoba [poz [-330 120] tvar doska]
? nová "zásoba [poz [-330 40] tvar kocka]
? nová "zásoba [poz [-330 -40] tvar obluk]
? nová "zásoba [poz [-330 -140] tvar stlp]
? nová "zásoba [poz [-330 -200] tvar stlpik]
```

Doplňme do projektu tlačidlo s popisom **Nová stavba**, ktoré zruší aktuálnu stavbu – t.j. všetky korytnačky z triedy `kocka`. Vieme zrušiť konkrétnu korytnačku príkazom `zrušObjekt` ***`meno_korytnačky`*** alebo všetky korytnačky príkazom `zrušObjekt všetky`. Ako zrušíme len korytnačky z istej triedy? Využijeme operáciu `všetkyOd`.

> Operácia `všetkyOd` ***`názov_triedy`*** vráti zoznam mien všetkých objektov z triedy ***`názov_triedy`***.

Udalosť **priZapnutí** tlačidla **Nová stavba** definujeme takto: `zrušObjekt všetkyOd "kocka.`

Z modulu Programovanie 1 si pamätáme, že každá z našich kociek sa môže zobraziť v jednej zo štyroch farieb – záberov. Upravíme projekt tak, aby sme kocky vpravo mohli „prefarbovať“ kliknutím pravým tlačidlom myši. Pridajme triede `kocka` udalosť **priPravomDolu**. Reakciu na udalosť definujme takto: `nechZáber záber+1`.

Vytvorte nejakú stavbu, a potom klikajte pravým tlačidlom myši na kocky v stavbe.

Ak do pravej časti stránky pridáme nejaké vhodné pozadie, môžeme podobným spôsobom ako projekt Kocky vytvoriť napr. projekty na zdobenie vianočného stromčeka, medovníkov, či keramiky, stavanie hradu a mnohé iné. Ukážky niektorých nájdete v e-learningovej podpore k tomuto modulu.

Zadanie 2

Pridajte triede `kocka` udalosť **priĽavomDolu**. Reakciou na ňu nech je presunutie korytnačky navrch (príkaz `navrch`).

Živý obraz 3

V kapitole 2 sme vytvárali živý obraz tak, že pri kliknutí do stránky sme sa podľa miesta kliknutia rozhodli, aký objekt vytvoríme. Vyvinieme podobný projekt:

- Ak klikneme na farbu `zelená3` vznikne žabka. Žabka pri kliknutí poskočí.
- Ak klikneme na farbu `belasá12` vznikne vták. Vtáky budú lietať istou rýchlosťou. Vždy, keď prídu na koniec stránky, sa otočia.
- Ak klikneme na farbu `olivová7` vznikne kvet.

V e-learningovom kurze k tomuto modulu sa nachádza súbor *zivyObraz3.zip*, v ktorom nájdete súbory potrebné k tomuto projektu. Súbor rozbaľte do samostatného priečinka a v Imagine otvorte súbor *zivyObraz3.imp*.

Nastavenia a správanie jednotlivých objektov sme v kapitole 2 určovali v zozname natavení pri vytváraní objektov, resp. až dodatočne po ich vytvorení. Teraz, keď už vieme definovať triedy, môžeme správanie a nastavenia jednotlivých druhov objektov pripraviť dopredu – t.j. zadefinujeme pre každý druh vlastnú triedu. Porozmýšľajme, aké triedy a s akými nastaveniami a udalosťami potrebujeme.

- Trieda `žaba`. Korytnačky z tejto triedy budú mať vypnuté pero, tvar zo súboru *zaba.lgf* a udalosť **priKliknutí**, reakciou na ktorú bude poskočenie.
- Trieda `Vták`. Korytnačky z tejto triedy budú mať vypnuté pero, tvar z premennej `:vtak`, počiatočný smer 90, typ oblasti `sOdrazom` (aby sa automaticky na kraji otáčali), premennú `rýchlosť` s nejakou rozumne veľkou náhodnou hodnotou. Na lietanie využijeme proces, ktorý sa spustí pri vytvorení vtáka – t.j. bude mať definovanú udalosť **priVytvorení**.
- Trieda `Kvet`. Korytnačky z tejto triedy budú mať vypnuté pero. Tvar im definujeme návodom na kreslenie.

Udalosť **priVytvorení** nastáva, keď vytvoríme korytnačku – t.j. po príkaze `nová "vták […]`. Reakciu na ňu definujeme vtedy, keď chceme robiť niečo s novou korytnačkou hneď pri jej vzniku.

Otvorme pripravený projekt *zivyObraz3.imp* – máme v ňom pripravené pozadie a obrázkovú premennú `vtak`.

Triedu `žaba` vytvoríme príkazom `nováTrieda "korytnačka "žaba [pero ph tvar zaba]`. Triede pridáme udalosť **priKliknutí** s reakciou: `do 20 čakaj 100 vz 20`.

Triedu `vták` vytvoríme príkazom `nováTrieda "korytnačka "vták [pero ph tvar :vtak smer 90 rýchlosť (2 + náhodne 3) typOblasti sOdrazom]`. Triede pridáme udalosť **priVytvorení**. Reakciou na ňu bude spustenie procesu lietania `každých 20 [do rýchlosť]`.

Všimnite si, akým spôsobom nastavujeme v zozname nastavení tvar, ak ho berieme zo súboru, z premennej či definujeme návodom na kreslenie.

Triedu `kvet` vytvoríme príkazom `nováTrieda "korytnačka "kvet [pero ph tvar [nechFp "biela nechHp 5 opakuj 9 [do 10 vz 10 vp 40] nechFp "žltá bod 10]]`. Tvar kvetu môžete definovať aj inak.

Stránke definujeme udalosť **priKliknutí** s reakciou `k1'vytvorKorytnačku`. `K1` je pomocná korytnačka, ktorou budeme zisťovať farbu bodu na mieste, kde sme klikli myšou. V príkaze `vytvorKorytnačku` ju najskôr premiestnime na pozíciu myši, a potom sa podľa farby bodu rozhodneme, či vytvoríme žabku, vtáka alebo kvet.

```
viem vytvorKorytnačku
 nechPoz pozMyši
 ak farbaBodu = "zelená3 [nová "žaba [poz (poz)]]
 ak farbaBodu = "belasá12 [nová "vták [poz (poz)]]
 ak farbaBodu = "olivová7 [nová "kvet [poz (poz)]]
koniec
```

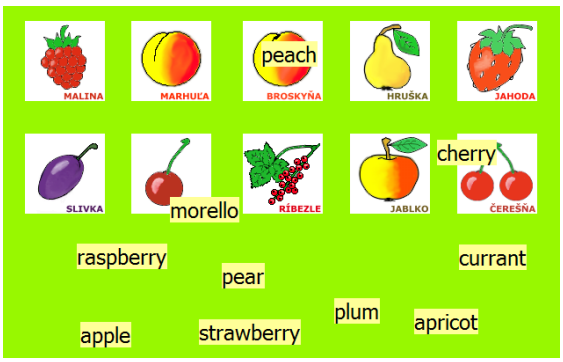
Ak sme triedu už vytvorili a potrebujeme jej pridať nejakú ďalšiu premennú, môžeme to urobiť v okne Pamäť. Dvojklikneme na triedu, zvolíme záložku **Premenné**, klikneme na **Pridaj**, napíšeme názov premennej, stlačíme **OK** a napíšeme hodnotu premennej. Pridajte týmto spôsobom triede `žaba` premennú `zväčšenie` a nastavte jej hodnotu `0.5`.

Zadanie 3	Pridajte tlačidlo s popisom **Zruš**, ktoré zruší všetky žaby, vtáky a kvety. Pozor, nesmiete zrušiť korytnačku `k1`!
Zadania 4	Doplňte triedu `včela` s vypnutým perom a tvarom zo súboru *vcela.lgf*. Po vytvorení začne včela náhodne lietať. Včely budú vznikať na farbe `ružová9`.

V e-learningovej podpore k tomuto modulu sa nachádza súbor *karticky.zip*, v ktorom sú pripravené obrázky a iné súbory k projektu. Rozbaľte ho do samostatného priečinka *Karticky* a svoj nový projekt uložte do toho istého priečinka.

Projekt Kartičky

Vytvoríme projekt na učenie sa slovíčok - k obrázkom budeme priraďovať kartičky s anglickými slovíčkami.



Obrázok 5 Ukážka výsledného projektu Kartičky

Aké druhy korytnačiek budeme potrebovať? Jeden druh budú **obrázkové kartičky** s obrázkami rôznych objektov. Tieto budú mať vypnuté pero a nastavenie `názov`, kde bude názov objektu. Druhý druh budú **textové kartičky** so slovíčkami. Textové kartičky budú mať vypnuté pero, náhodnú pozíciu, zapnuté automatické ťahanie, nejaké rozumné písmo a premennú `názov`, v ktorej si bude pamätať slovíčko, ktoré reprezentuje. Pri vytvorení takejto korytnačky jej zmeníme tvar - pomocou návodu na kreslenie popíšeme kartičku s príslušným slovíčkom. Pri pustení textovej kartičky pomocou premennej `názov` skontrolujeme, či bola položená na správny obrázok. Trieda bude mať teda definované udalosti **priVytvorení** a **priĽavomHore**.

Otvorme nový projekt a zrušme korytnačku. Najskôr definujeme triedu pre obrázkové kartičky - `obrázkováK`.

```
? nováTrieda "korytnačka "obrázkováK [pero ph názov ||]
```

Hodnota premennej `názov` v triede `obrázkováK` je prázdny reťazec. Konkrétnu hodnotu priradíme až konkrétnej korytnačke z triedy `obrázkováK`, keď ju budeme vytvárať. Takisto tvar budeme definovať až v zozname nastavení pre konkrétnu korytnačku. Vyskúšajme:

```
? nová "obrázkováK [názov "apple tvar apple poz [100 0]]
```

Triedu pre textové kartičky - `textováK` - definujeme takto:

```
? nováTrieda "korytnačka "textováK [pero ph poz ? názov || autoŤahanie
áno písmo [|Tahoma| [18 400 0 1 0 238]]]
```

V okne Pamäť pridáme triede `textováK` udalosť **priVytvorení**. Reakciou na ňu bude príkaz `môjTvar`. Potom zvolíme záložku **Procedúry** a pridáme triede `textováK` procedúru `môjTvar`, v ktorej popíšeme tvar korytnačky - nastavíme farbu výplne (bude použitá ako pozadie textu) a napíšeme hodnotu premennej `názov`.

```
viem môjTvar
  nechTvar ![nechFv "žltá10 text (názov)]
koniec
```

Vyskúšame, či sme triedu zadefinovali správne - vytvoríme jednu textovú kartičku: `? nová "textováK [poz [100 0] názov apple]`

apple

Zmeňme farbu pozadia tak, aby na nej bolo žlté kartičky dobre vidno (napr. `olivová`). Zrušme obe kartičky a pokračujme v definícii triedy `textováK`.

Nastavenie písma pre textovú kartičku môžeme zistiť takto:

1. Do príkazového riadku napíšeme `? nechPísmo` a stlačíme F9.
2. Zvolíme písmo, ktoré sa nám páči, nastavíme vhodnú veľkosť (aspoň 12) a zaškrtneme nastavenie **Vyplň aj pozadie**. Stlačíme **Urob**.
3. Skopírujeme do schránky zoznam, ktorý sa dopísal za príkaz `nechPísmo`, napr. `[|Tahoma| [18 400 0 1 0 238]]`.

Ak chcete mať kartičku aj s rámikom, definujte príkaz `môjTvar` takto:

```
viem môjTvar
 urobTu "šírka prvý
   veľkosťTextu názov
 urobTu "výška prvok 2
   veľkosťTextu názov
 nechTvar
  ![nechFv "žltá9
    text (názov)
    vp 90 vz 1
    opakuj 2
     [do (:šírka+2)
      vp 90
      do :výška
      vp 90 ]]
koniec
```

Pridáme triede `textováK` udalosť **priĽavomHore** s reakciou `kontrola`. Potom definujeme triede procedúru `kontrola`, ktorá bude kontrolovať, či sme obrázok pustili nad správnou kartičkou. Pomocou operácií, ktoré zisťujú prekrývanie korytnačiek, zistíme, nad ktorým obrázkom sme pustili textovú kartičku.

> Operácia `prekrývajúMa` vráti zoznam všetkých korytnačiek, ktoré sa prekrývajú s aktívnou korytnačkou.
>
> Operácia `prekrývajúMaZTýchto` *`zoznam_korytnačiek`* vráti zoznam tých korytnačiek zo *`zoznamu_korytnačiek`*, ktoré sa prekrývajú s aktívnou korytnačkou.

V príkaze `kontrola` (pozri nižšie) použijeme operáciu `prekrývajúMaZTýchto` s parametrom `všetkyOd "obrázkováK`, pretože nás zaujímajú len **obrázkové** kartičky, ktoré sa prekrývajú s našou textovou (pozri riadok 1 príkaze `kontrola`) - eliminujeme tým prekrývanie s inými textovými kartičkami. Ak zistíme, že textovú kartičku neprekrýva žiadna obrázková kartička, t.j. výsledok operácie bude prázdny zoznam, nemusíme robiť nič (pozri 2). Ak nejaká obrázková kartička prekrýva našu textovú, porovnáme hodnoty premenných `názov` textovej a prvej prekrývajúcej obrázkovej kartičky (pozri 3). Ak sú rovnaké, vypíšeme do príkazového riadku slovo **SUPER** (pozri 4), ak sú rôzne, vrátime textovú korytnačku domov (pozri 5).

```
viem kontrola
 urobTu "k prekrývajúMaZTýchto všetkyOd "obrázkováK          (1)
 ak prázdny? :k [ukonči]                                     (2)
 ak2 názov = pre prvý :k [názov]                             (3)
  [zobraz "SUPER]                                            (4)
  [domov]                                                    (5)
koniec
```

Otestujme triedy. Vytvorme obrázok jablka a hrušky s názvami *apple* a *pear* a kartičku s textom *apple*.

```
? nová "obrázkováK [poz ? názov apple tvar apple]
? nová "obrázkováK [poz ? názov pear tvar pear]
? nová "textováK [poz ? názov apple]
```

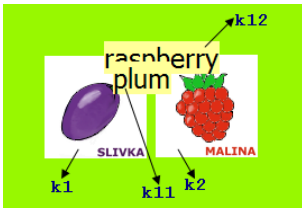
Presuňme kartičku s textom *apple* nad obrázok hrušky. Potom presuňme kartičku *apple* nad obrázok jablka a pozrime príkazový riadok. Zrušme všetky kartičky.

Zadanie 5	Upravte procedúru `kontrola` triedy `textováK` tak, aby sa v prípade správneho priradenia textová kartička nedala viac ťahať. Pomôcka: `nechAutoŤahanie "nie`
Zadanie 6	Vymyslite nejakú lepšiu spätnú väzbu v prípade správneho priradenia a zrealizujte ju (napr. pomocou korytnačky smajlíka s dvoma zábermi - usmiaty a zamračený, alebo zobrazenie správy využitím textového objektu).

Zostáva nám už len naštartovať hru - t.j. vytvoriť všetky potrebné obrázkové aj textové kartičky. Na to si potrebujeme pripraviť niekoľko vecí:

- Aby sme mohli projekt ľahko obmieňať, napíšeme zoznam slovíčok, ktoré chceme trénovať, do textového súboru *slovicka.txt* (nájdete v pripravenom súbore *karticky.zip*).
- Pripravíme si obrázkové súbory pre jednotlivé slovíčka (nájdete v pripravenom súbore *karticky.zip*). Pre vytváranie kartičiek bude výhodné, aby názvy obrázkových súborov boli rovnaké ako slová z textového súboru

Porovnajte operácie `prekrývajúMa` a `PrekrývajúMaZTýchto`:




```
? zobraz
   k11'prekrývajúMa
[k1 k2 k12]

? zobraz
   k11'prekrývajúMaZTýchto
   [k1 k2 k3 k4 k5]
[k1 k2]
```

Všimnite si, že oba druhy kartičiek majú spoločné vypnuté pero a premennú `názov`. Mohli sme najskôr vytvoriť triedu `kartička` s vypnutým perom a premennou `názov` a od nej potom odvodiť triedy `obrázkováK` a `textováK`:

```
? nováTrieda
   "korytnačka
   "kartička
   [pero ph názov ||]
? nováTrieda
   "kartička
   "textováK
    [poz ?
     autoŤahanie áno
     písmo [|Tahoma|
    [18 400 0 1 0 238]]
```

slovicka.txt.

- Pripravíme si tiež zoznam pozícií, na ktoré budeme umiestňovať obrázkové kartičky. Dáme ich do globálnej premennej `pozície`:
  `? urob "pozície [[-300 150] [-150 150] [0 150] [150 150] [300 150] [-300 0] [-150 0] [0 0] [150 0] [300 0]].`

V projekte definujeme príkaz `začniHru`, v ktorom vytvoríme všetky korytnačky. Najskôr do premennej `slová` načítame obsah súboru *slovicka.txt* (pozri riadok 1 v príkaze `začniHru`). Potom pre každý prvok zo zoznamu `slová` (2) vytvoríme obrázkovú kartičku (3-6). Pri vytvorení kartičky jej nastavíme tvar (4), pozíciu (5) a premennú `názov` (6). Nakoniec pre každý prvok zo zoznamu `slová` (7) vytvoríme textovú kartičku (8-10). Pri vytvorení jej nastavíme `názov` (9) a pozíciu – náhodnú, ale len v spodnej časti stránky (10).

```
viem začniHru
 urobTu "slová čítajTextovýSúbor "slovicka.txt                    (1)
 opakuj počet :slová                                              (2)
  [nová "obrázkováK                                               (3)
     [tvar (prvok poč :slová)                                     (4)
      poz (prvok poč :pozície)                                    (5)
      názov (prvok poč :slová)]]                                  (6)
 opakuj počet :slová                                              (7)
   [nová "textováK                                                (8)
     [názov (prvok poč :slová)                                    (9)
      poz (?bod [-350 -70 650 150])]]                             (10)
koniec
```

Namiesto príkazu `opakuj` môžeme v tomto prípade použiť príkaz `prePrvky`:

```
prePrvky "p :slová
 [nová "obrázkováK
  [tvar :p
   poz (prvok poč
              :pozície)
   názov :p]]
```

Viac o cykle `prePrvky` nájdete v Pomocníkovi.

Operácia `?bod` *`obdĺžnik`* – vráti náhodnú pozíciu v oblasti danej *`obdĺžnikom`*.

Obdĺžnik je daný štvoricou čísel v hranatých zátvorkách `[x y šírka výška]`, kde `x, y` sú súradnice ľavého horného rohu obdĺžnika, `šírka` a `výška` sú rozmery obdĺžnika.

Zadanie 7

Pridajte tlačidlo s popisom **začni**, ktoré bude spúšťať hru.

Pri viacnásobnom stlačení tlačidla **začni** sa nám nahromadia rovnaké kartičky. Preto pridáme na začiatok príkazu `začniHru` zrušenie oboch typov kartičiek. Keďže iné korytnačky ako kartičky v projekte nemáme, môžeme použiť príkaz `zrušObjekt všetky`. Aby sa kartičky nevytvárali stále na rovnakom mieste, môžeme poradie slov načítaných zo súboru hneď na začiatku pomiešať. Príkaz `začniHru` upravíme takto:

```
viem začniHru
 zrušObjekt všetky
 urobTu "slová pomiešaj čítajTextovýSúbor "slovicka.txt
 ...
koniec
```

Operácia `pomiešaj` *`zoznam / slovo`* – náhodne zmení poradie prvkov *`zoznamu / slova`*.

Keďže obrázky a zoznam slovíčok máme mimo projektu, dokážeme ho veľmi ľahko zmeniť. Stačí vymeniť obrázky a prepísať zoznam slovíčok v súbore *slovicka.txt*. Projekt tak môžeme využiť nielen na učenie slovíčok, ale na ľubovoľné priraďovanie pojmov k obrázkom napr. priraďovanie názvov stromov k obrázkom listov, mien zvierat k obrázkom zvierat atď. Zachovať musíme len jedno pravidlo: názvy obrázkových súborov musia byť rovnaké ako slovíčka v súbore *slovicka.txt*.

Čo sme sa naučili

- definovať korytnačke tvar pomocou návodu na kreslenie,
- definovať nové triedy korytnačiek a vytvárať ich inštancie,
- pracovať len s korytnačkami istej triedy pomocou príkazu `všetkyOd`,
- využívať triedy korytnačiek pri vývoji projektov.

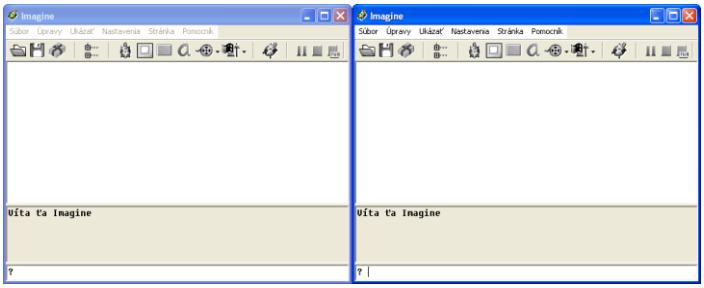
Kapitola 4: Sieť

Imagine Logo umožňuje vytvárať projekty, ktoré komunikujú po sieti. A to tak, že niekoľko bežiacich program v Imagine Logu dokáže navzájom komunikovať. Takto môžeme implementovať nielen sieťové hry, ale aj chatovacie programy, kolaboratívne prostredia, či aplikácie, ktoré demonštrujú paralelné výpočty na niekoľkých počítačoch.

Základným stavebným kameňom každého sieťového programu je objekt triedy `Spojenie`. Ten realizuje spojenie niekoľkých počítačov a odovzdávanie správ medzi nimi.

Nadviazanie spojenia a interaktívne posielanie správ

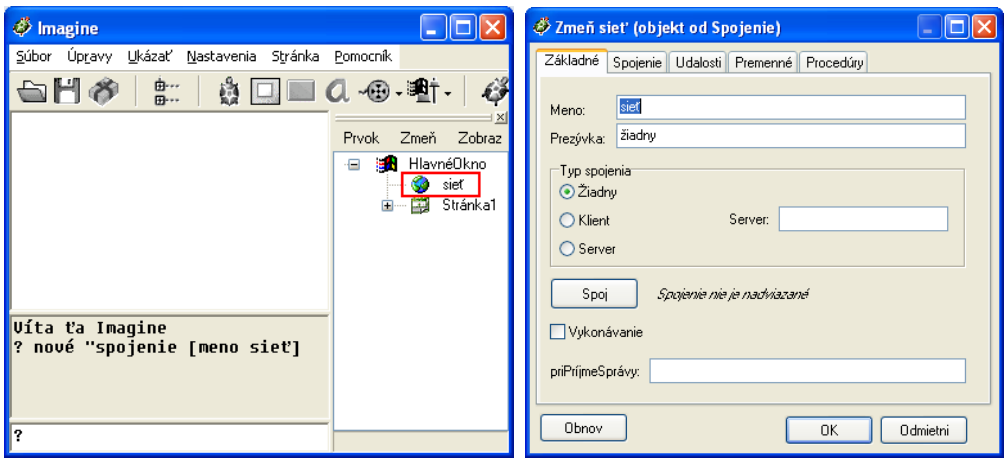
Prvý pokus so sieťou spravíme na jednom počítači, tak, že na ňom spustíme dve kópie Imagine Loga a programy v oboch kópiách budú spolu komunikovať. Spustíme Imagine Logo dvakrát pomocou ikony na ploche, v paneli Rýchle spustenie alebo v ponuke Štart (nie dvojklikom na samotné *Imagine.exe*!). Potom upravíme ich okná na takú veľkosť, aby sa zmestili vedľa seba.



V ďalšom texte budeme dve kópie Imagine Loga nazývať ľavé okno a pravé okno. Do príkazového riadku každého okna napíšeme:

```
? nové "spojenie [meno sieť]
```

Tým v oboch kópiách Imagine Loga vznikne objekt triedy `Spojenie` a nazve sa `sieť`. Je to nevizuálny objekt, teda na rozdiel od korytnačiek, tlačidiel, či posúvačov po jeho vytvorení neuvidíme na stránke nič nové. Každý objekt však môžeme vidieť v okne Pamäť. V okne Pamäť dvojklikneme na `sieť` - otvorí sa jeho rodný list:

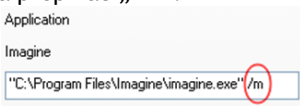


Obrázok 6 Vľavo objekt `sieť` v okne pamäť, vpravo jeho rodný list.

V rodnom liste vidíme, že objekt existuje, ale žiadne spojenie sa ešte nenadviazalo.

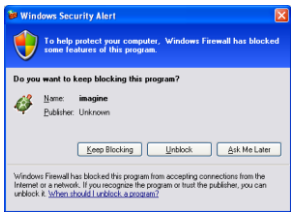
Ľavé okno bude ***serverom*** v našej sieti. Preto zaškrtneme voľbu **Server** a stlačíme tlačidlo **Spoj**. Rodný list objektu `sieť` si necháme otvorený a prepneme sa do pravého okna. To bude ***klientom***. Otvoríme aj tam rodný list objektu `sieť`, tam ale

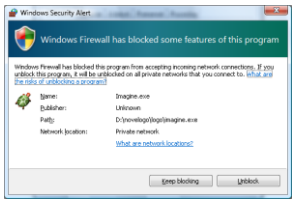
Poznámka: Ak sa vám nepodarilo spustiť Imagine Logo dvakrát (druhé spustenie len znova ukázalo okno už spusteného Imagine Loga), kliknite pravým tlačidlom na ikonu Imagine Loga, zvoľte Vlastnosti a v okne pripíšte k menu exe súboru medzeru a prepínač **„/m“**:



Poznámka: Keby sme objekt vytvorili len pomocou `nové "spojenie []`, volal by sa `spojenie1`, čo je dosť dlhé meno na časté písanie. Objekt bude pre náš program reprezentovať celú sieť, v ktorej sa spoja dve bežiace kópie Imagine Loga, preto sme mu dali meno `sieť`.
Ak nechcete používať pri programovaní slovenskú klávesnicu, nazvite si objekt len `siet`.

Ak máte aktivovaný Windows Firewall, tak sa vás po stlačení **Spoj** spýta, či má povoliť Imagine Logu stať sa serverom:

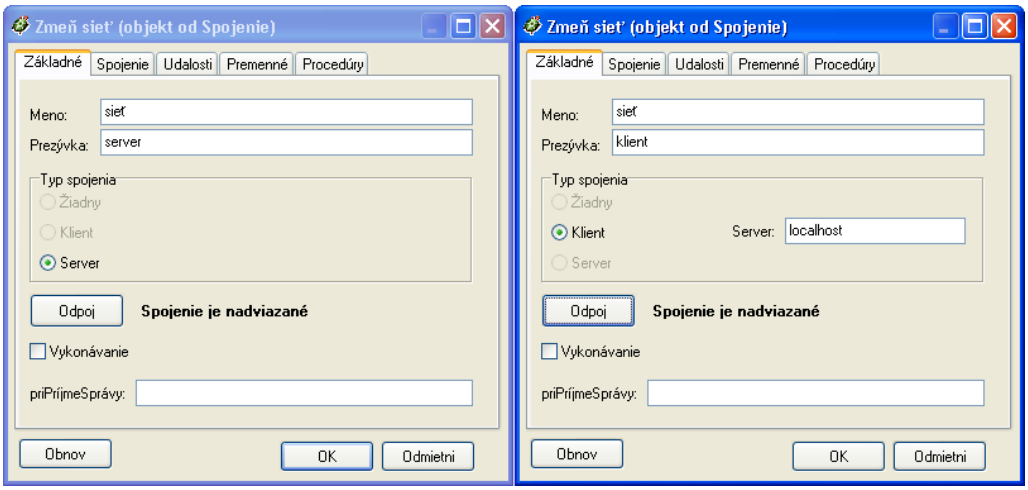




Stlačte Unblock. Windows Firewall povolí Imagine Logu komunikovať po sieti aj sa stať serverom, a toto si aj trvalo zapamätá (viac sa už nebude spytovať).

Ak máte inštalovaný iný firewall, tiež sa môže podobným spôsobom spytovať. Musíte nastaviť, že Imagine Logo môže byť serverom aj klientom v pripojení na sieť (môže prijímať spojenia zvonku aj sa spájať smerom von). Odporúčame nastaviť tieto voľby ako trvalé, aby si ich pamätal. Môže sa však stať, že váš firewall sa nespýta a zablokuje Imagine Logu prístup na sieť. V takom prípade musíte sami nájsť nastavenia svojho firewallu a nastaviť ho tak, aby mohlo Imagine Logo komunikovať po sieti, inak si nebudete môcť vyskúšať triedu `Spojenie`.

zaškrtneme **Klient** a do položky **Server** napíšeme: **localhost** (to znamená, že server beží na rovnakom počítači ako klient). Stlačíme **Spoj**.



Obrázok 7 Nadviazanie spojenia v ľavom a pravom okne.

Teraz si môžeme vyskúšať vytvorenú sieť. V oboch oknách sa prepneme na záložku **Spojenie**. V časti **Kto v sieti** vidíme, že v sieti sú spojené dva programy, jeden má prezývku **server** a druhý **klient**.

V ľavom okne napíšeme niečo do textového poľa **Správa** a stlačíme **Pošli**. V pravom okne dolu si všimneme, že mu správa prišla.

Obrázok 8 Posielanie správ

Zadanie 1	Všimnite si, že pred prezývkami v časti **Kto v sieti** sú zaškrtávacie políčka. Skúste prísť na to, k čomu slúžia a vysvetlite ich vplyv na dianie v spojení (čo ak nie je zaškrtnuté ani jedno, čo keď jedno a čo keď obe). Pomôcka: skúste zaškrtávať a posielať správy.
Zadanie 2	V záložke **Základné** je textové pole **Prezývka**. Zmeňte ju v pravom okne a stlačte **OK**. V zozname **Kto v sieti** v ľavom okne sa prezývka nezmení. Ako dosiahnuť aby sa zmenila? Pomôcka: Stlačte tlačidlo **Obnov**.

Použitie príkazov na spojenie a posielanie správ

Doteraz sme menili nastavenia siete a posielali správy interaktívne - pomocou rodného listu objektu `sieť`. Teraz sa to naučíme robiť programovo - pomocou príkazov. Najprv zatvoríme ľavé aj pravé okno a znova spustíme dve kópie Imagine Loga a usporiadame ich ako ľavé a pravé okno.

V oboch oknách teraz napíšeme po dva príkazy, ktorými vytvoríme objekty `sieť`, zadefinujeme im vlastnosti a spojíme ich:

ľavé okno

```
? nové "spojenie [typ server
                  meno sieť]
? sieť'spoj
```

pravé okno

```
? nové "spojenie [typ klient
                  server localhost
                  meno sieť]
? sieť'spoj
```

Na zistenie, či je spojenie naozaj nadviazané, môžeme použiť príkaz:

```
? zobraz sieť'spojený?
```

Správu pošleme tak, že v ľavom okne napíšeme:

```
? sieť'pošliSprávu [] [Ahoj!]
```

Prvý parameter určuje komu sa posiela - prázdny zoznam znamená "všetkým v spojení okrem mňa". Druhý parameter je správa. Môže to byť zoznam alebo slovo.

V pravom okne sa však nič neobjaví. Objekt `sieť` v pravom okne síce správu prijal, ale nič s ňou automaticky nespraví. Správu môžeme vypísať príkazom:

```
? zobraz sieť'správa
```

Ak chceme vedieť aj odosielateľa, použijeme:

```
? (píš sieť'odosielateľ ": sieť'správa)
```

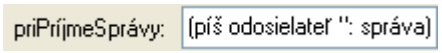
Zoznam prezývok pripojených do siete je hodnotou nastavenia **ktoVSieti**:

```
? zobraz sieť'ktoVSieti
```

Keď to skúsime viackrát po sebe, zistíme, že funkcia **správa** vždy vráti posledne prijatú správu. Aby sme ale vedeli, kedy prišla správa, musíme naprogramovať reakciu na udalosť **priPríjmeSprávy** objektu sieť. V pravom okne v rodnom liste objektu `sieť`, do okienka **priPríjmeSprávy** napíšeme:

```
(píš odosielateľ ": správa)
```

Rovnako definujeme udalosť **priPríjmeSprávy** aj v ľavom okne.



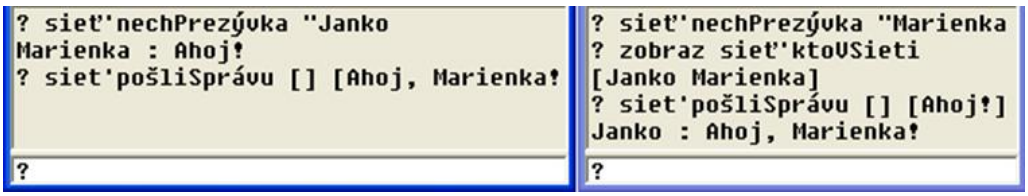
Odosielatelia majú príliš technické mená: `server` a `klient`. Môžeme ich premenovať - zmeniť im prezývky, napríklad:

- v ľavom okne: `? sieť'nechPrezývka "Janko`
- v pravom okne: `? sieť'nechPrezývka "Marienka`

Keď teraz znova pošleme správu z ľavého okna do pravého, automaticky sa objaví v textovej ploche. Rovnako sa objaví správa, keď ju pošleme z pravého okna do ľavého:

Pri výpise `KtoVSieti` sme použili príkaz `zobraz`, lebo v prípade zoznamu vypíše okolo neho aj hranaté zátvorky. Pri výpise správy ale radšej použijeme príkaz `píš`, lebo ten vynecháva hranaté zátvorky okolo zoznamov a správy budeme často písať ako zoznamy. Všimnite si aj použitie okrúhlych zátvoriek s príkazom `píš`: keď má iný počet než dva parametre, tak musíme uzatvoriť meno príkazu aj všetky jeho parametre do zátvoriek.

Keď sa v udalosti objektu sieť odvolávame na jeho premenné `odosielateľ` a `správa`, tak už nemusíme uvádzať meno objektu a apostrof. Imagine Logo vie, že sú to premenné daného objektu či jeho triedy.



Obrázok 9 Posielanie správ pomocou príkazov

Spojenie medzi počítačmi

Doteraz sme experimentovali so spojením dvoch bežiacich kópií Imagine Loga na jednom počítači. Účelom siete ale je, aby bolo spojených niekoľko počítačov. Pri spojení počítačov postupujeme takto:

1. Vyberieme server.

Jeden z počítačov si zvolíme za **server**. Keď sa spájame v rámci lokálnej siete, môže byť serverom ľubovoľný počítač, dôležité je, že **sa musí spojiť ako prvý**. Ak sa však spájajú počítače v rámci internetu, je dôležité, aby server mal verejnú IP adresu.

Server má verejnú IP adresu, ak nie je pripojený na internet cez router či firewall, ktorý vykonáva preklad adries (NAT). Informácie o tom, ako rozoznať, či má váš počítač verejnú adresu a ako zistiť neverejné adresy a mená počítačov, nájdete v e-learningovej podpore k tomuto modulu.

V každom prípade musíme vedieť meno (v prípade spájania mimo lokálnej siete to musí byť kompletné meno aj s doménovými príponami) alebo IP adresu servera. V ďalšom budeme toto meno či IP adresu nazývať MENO_SERVERA. Adresu počítača môžeme zistiť na stránke http://whatismyip.com/. Zobrazené štyri čísla oddelené bodkami sú adresou nášho počítača (ale len ak má počítač verejnú adresu).

2. Pripojíme server.

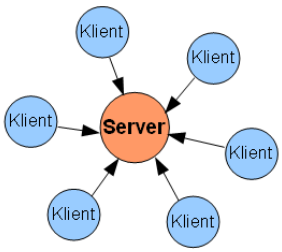
V Imagine Logu na počítači, ktorý je serverom, vytvoríme objekt triedy `spojenie` s typom `server` a menom `sieť`. Následne nadviažeme spojenie:

```
? nové "spojenie [typ server meno sieť]
? sieť'spoj
```

3. Pripojíme klientov.

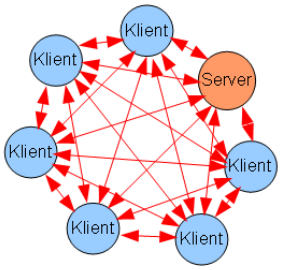
Na ďalšom jednom alebo viacerých počítačoch vytvoríme objekt triedy `spojenie` s menom `sieť`, typom `klient` a serverom ***MENO_SERVERA***. Následne sa spojíme:

```
? nové "spojenie [typ klient server MENO_SERVERA]
? sieť'spoj
```



Obrázok 10 Postavenie počítačov pri pripájaní

Takto vznikne sieť zo servera a niekoľkých klientov. Dôležité je vedieť, že pri pripájaní k sieti sú počítače nerovnocenné (Obrázok 10). Jediný počítač je server a každý, kto sa chce pripojiť, musí sa stať klientom a nadviazať spojenie so serverom. Nikdy sa nemôže pripojiť klient na klienta. Pri odpojení klienta ostane zvyšok siete funkčný, avšak pri odpojení servera sa odpoja aj všetci klienti a sieť sa rozpadne.



Obrázok 11 Postavenie počítačov pri komunikácii

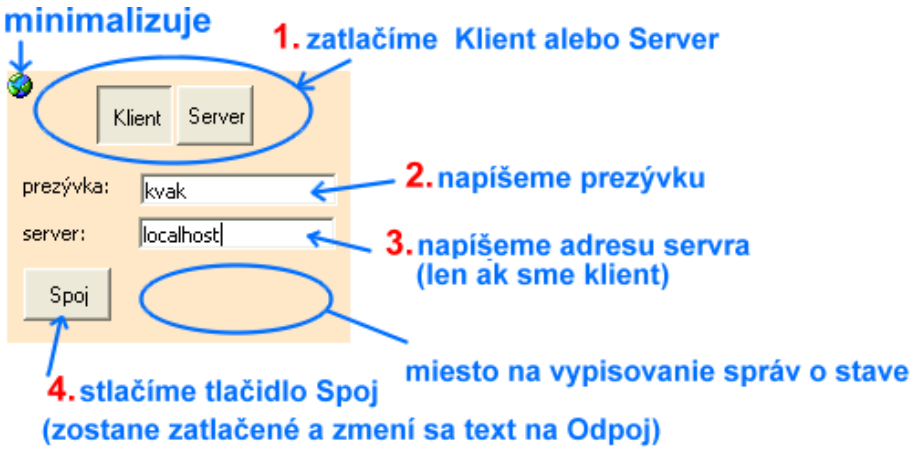
Keď sú už počítače pripojené, môže každý posielať správy každému počítaču v rámci vytvorenej siete. Pri komunikovaní už teda môžeme považovať počítače za rovnocenné (Obrázok 11).

Zadanie 3	Zistite IP adresu alebo meno svojho počítača.
Zadanie 4	Spojte sa vždy traja až štyria vedľa seba sediaci účastníci do siete. Zvoľte si, kto bude serverom, ten povie ostatným svoju IP adresu. Pripojí sa ako server, ostatní sa k nemu pripoja ako klienti. Vyskúšajte posielanie správ (chat).

Požívateľské rozhranie pre nadviazanie spojenia

V mnohých sieťových programoch, ktoré teraz naprogramujeme, budeme potrebovať na začiatku spraviť to isté - jeden sa musí spojiť ako server a ostatní musia zadať adresu alebo meno servera a spojiť sa ako klient.

Nechceme, aby to používatelia nášho programu museli robiť príkazmi Loga, preto vytvoríme malé používateľské rozhranie pre prihlasovanie, ktoré potom budeme môcť použiť vo všetkých svojich sieťových programoch.



Obrázok 12 Návrh rozhrania pre nadviazanie spojenia

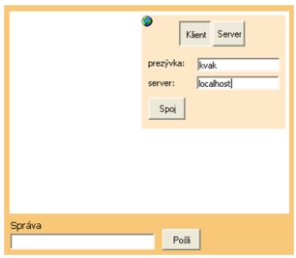
Toto okienko je vytvorené ako samostatný papier, ktorý môžeme presúvať v rámci stránky a umiestniť tak, aby nám neprekážal v práci. Pomocou ikonky v ľavom hornom rohu môžeme celý papier zminimalizovať len na ikonku, keď ho nepotrebujeme, a neskôr zase zväčšiť na plnú veľkosť. Naprogramované rozhranie nájdete v súbore *Spojenie.imp* v e-learningovej podpore k tomuto modulu.

Podrobný návod ako sme toto rozhranie vytvorili, nájdete v e-learningovej podpore k tomuto modulu.

Zadanie 5

Naprogramujte program Chat2, ktorý "obalí" to, čo sme sa naučili o posielaní správ, jednoduchým grafickým používateľským rozhraním (pozri obrázok na okraji), kde sa bude dať:

- vytvoriť spojenie (použite *Spojenie.imp*),
- písať správu do textového okienka a poslať ju všetkým v sieti stlačením tlačidla,
- vidieť celú konverzáciu (vrátane mojich správ) v textovom okne.



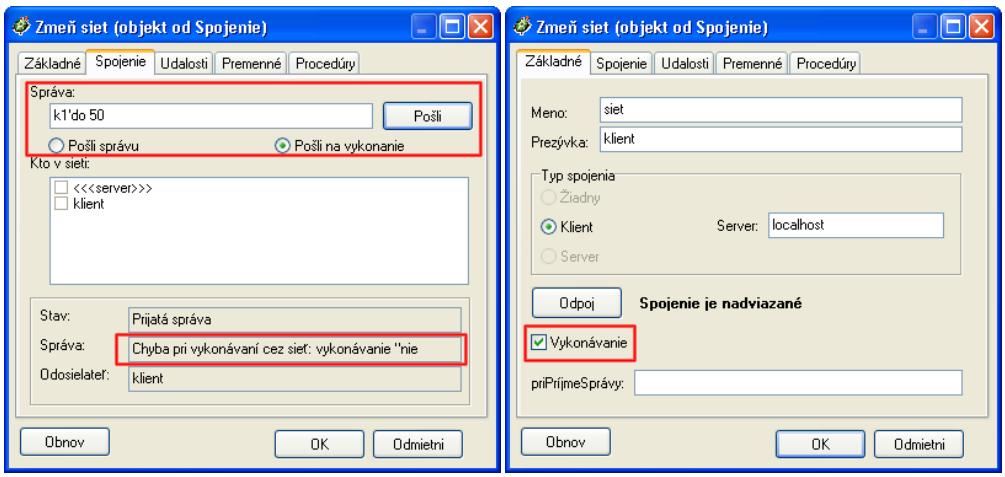
Posielanie inštrukcií

Spustíme dve nové kópie Imagine Loga (ľavé a pravé okno), v každom otvoríme projekt *Spojenie.imp* a spojíme ich.

V pravom okne vytvoríme korytnačku. Potom v ľavom okne otvoríme rodný list pre objekt `sieť`, v jeho záložke **Spojenie** zaškrtneme **Pošli na vykonanie**, do textového poľa **Správa** napíšeme `k1'do 50` a stlačíme **Pošli** (pozri Obrázok 13 vľavo).

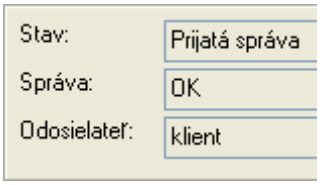
Vidíme, že poslanie správy skončilo chybou - pravé okno poslalo späť správu, ktorou chce naznačiť, že vykonávanie príkazov prijatých cez sieť je zakázané. A naznačuje aj, čo treba urobiť, aby bolo povolené. Otvorme preto rodný list pre `sieť` v pravom okne, zaškrtnime v ňom **Vykonávanie** a stlačme **OK** (pozri Obrázok 13 vpravo).

Upozornenie: Posielanie inštrukcií by sme mali povoliť len v projektoch, kde dôverujeme používateľom počítačov spojených v sieti. Poslaním inštrukcií totiž možno vykonať aj deštruktívne akcie.



Obrázok 13 Posielanie inštrukcií

Znova stlačíme v ľavom okne **Pošli**. Korytnačka `k1` v pravom okne vykoná príkaz a v ľavom okne vidíme:



Zatvorme rodný list objektu `sieť` a vyskúšajme príkaz na posielanie inštrukcií:

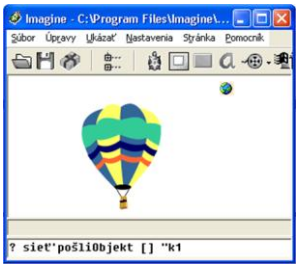
```
? sieť'pošliNaVykonanie [] [k1'vp 90]
```

V e-learningovej podpore k modulu nájdete aj postup vývoja projektu **Spoločné kreslenie**, ktorý umožní kresliť naraz niekoľkým užívateľom na zdieľanej ploche.

Zadanie 6	Prečo sme museli poslať príkaz `k1'do 50` a nie len `do 50`? Skúste to a vysvetlite výsledok.
Zadanie 7	Skúšajte pomocou posielania príkazov nakresliť niečo v druhom okne Imagine Loga.

Posielanie objektov

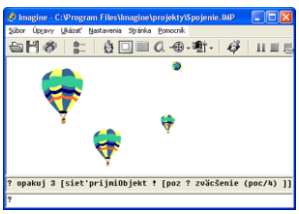
Imagine Logo dokáže posielať po sieti nielen dáta a inštrukcie, ale aj celé objekty, lepšie povedané ich kópie alebo klony. Spustíme si kópie Imagine Loga na niekoľkých počítačoch, otvoríme na nich *Spojenie.imp* a spojíme ich do siete.



Na jednom počítači v každej sieti vytvoríme korytnačku a v rodnom liste jej zmeníme tvar na *balon3.lgf*. Potom napíšeme do príkazového riadku:

```
? sieť'pošliObjekt [] "k1
```

Príkaz rozošle balón na všetky pripojené počítače. Na nich sa zatiaľ nič neobjaví - objekt, ktorý prišiel po sieti, totiž musí počítač najskôr prijať. Na ostatných počítačoch napíšeme: `? sieť'PrijmiObjekt []`



Kópia balónu sa objaví na všetkých počítačoch presne na tom mieste v ploche, kde bol aj na pôvodnom počítači. Jeden objekt možno prijať aj viackrát a možno pri tom meniť niektoré jeho nastavenia, napríklad:

```
? opakuj 3 [sieť'prijmiObjekt ! [poz ? zväčšenie (poč/4) ]]
```

Aby sme nemuseli prijímať objekty manuálne, trieda `Spojenie` pozná udalosť **priPríjmeObjektu**. Nastavme preto na všetkých počítačoch objektu `sieť` udalosť **priPríjmeObjektu** napríklad na: `prijmiObjekt [poz ?]`

Potom môžeme vytvárať korytnačky a posielať ich všetkým v našej sieti a všetci náš objekt prijmú a položia na náhodné miesto na svojej ploche.

Definujme balónu udalosť **priKliknutí** takto:

```
ph nechSmer ?prvok [90 80 270 280] šmýkajSa [do 100] []
```

Príkaz `šmýkajSa` presunie korytnačku pozdĺž čiary definovanej návodom na kreslenie v prvom vstupe, zatiaľ čo opätovane vykonáva inštrukcie zadané v druhom vstupe.

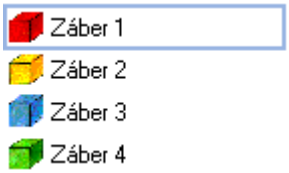
Teraz pošleme balón cez sieť príkazom `? sieť'pošliObjekt [] "k1` a na druhom počítači skúsime naňho klikať.

Zadanie 8

Vytvorte korytnačku, ktorá má nejaký tvar a reaguje na nejaké udalosti. Pošlite ju niekomu (alebo všetkým) v sieti.

Projekt Kocky v sieti

Vytvoríme sieťovú verziu projektu Kocky z modulu Programovanie 1. V ľavej časti stránky bude niekoľko druhov kociek. Každý z pripojených môže ľubovoľnú kocku potiahnuť. Tam, kde ju pustí sa príslušná kocka odtlačí (u všetkých spojených používateľov) a vráti na pôvodnú pozíciu.



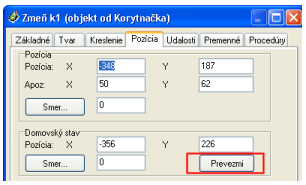
Spustíme si nové kópie Imagine Loga na niekoľkých počítačoch, otvoríme na nich *Spojenie.imp* a spojíme ich do siete. Uložíme projekt pod menom *KockyVSieti.imp*. Vytvoríme korytnačku niekde blízko ľavého horného rohu stránky a

- zmeníme jej tvar na obrázok zo súboru *kocka.lgf*,
- dáme jej pero hore,
- zapneme jej automatické ťahanie,
- nastavíme jej udalosť **priĽavomHore** na `ľavýHore` a definujeme jej procedúru `ľavýHore`:

```
viem  ľavýHore
 ak nieje sieť'spojený? [ukonči]
 ak2 poz = prvý domovskýStav
   [nechZáber záber+1]
   [ak xSúr > -250 [odtlač sieť'pošliObjekt [] mojeMeno] domov]
koniec
```

Pozor, ak je vaša stránka na začiatku inak veľká (napríklad ste nezačali načítaním súboru *Spojenie.imp*), možno budete musieť zmeniť číslo -250 v texte procedúry `ľavýHore` na iné.

Ak potrebujete premiestniť kocku na iné miesto (napríklad po zmene tvaru na *kocka.lgf* vám jej miesto nevyhovuje), nestačí ju len ručne presunúť, musíte zmeniť aj jej domovský stav - po presune otvorte jej rodný list a na záložke **Pozícia** stlačte **Prevezmi**:

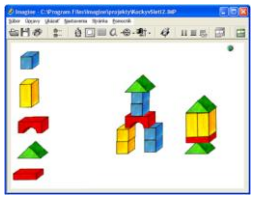


Ak na kocku len klikneme (teda udalosť **priľavomHore** nastane na pozícii rovnakej ako je domovská pozícia – pozri riadok s označením 1), tak kocka zmení záber, čím vlastne zmení farbu (pozri 2). Ak ale kocku niekam odnesieme a pustíme (udalosť **priľavomHore** nastane niekde inde), tak tam odtlačí svoj tvar (ale len ak to nie je príliš vľavo), pošle seba ostatným v sieti a skočí domov (pozri 3).

Objektu `sieť` nastavíme udalosť **priPríjmeObjektu** tak, aby sa každá prijatá kocka odtlačila na pozícii, kde vznikla a potom zrušila:

```
prijmiObjekt [meno prijatý] prijatý'odtlač zrušObjekt "prijatý
```

A môžeme skúsiť spoločne stavať.



Zadanie 9

Pridajte do projektu ďalšie tvary kociek – vytvorte štyri kópie korytnačky a zmeňte im postupne tvar na obrázky zo súborov *strecha.lgf*, *doska.lgf*, *kockaVelka.lgf*, *obluk.lgf*.

Čo sme sa naučili

- vytvoriť spojenie medzi dvoma kópiami Imagine Loga na jednom počítači či medzi dvoma alebo viacerými počítačmi v sieti,
- posielať správy, inštrukcie a objekty medzi spojenými počítačmi,
- vytvárať jednoduché sieťové projekty.

V e-learningovej podpore k modulu ešte nájdete námet na sieťovú hru Bomby.

Čo sme sa naučili v tomto module

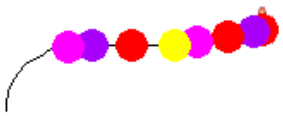
Zhrnutie

V tomto module sme sa naučili:

- pracovať s rôznymi typmi údajov: číslami, slovami a zoznamami,
- modifikovať projekt zmenou údajov z textového súboru,
- využívať objekt korytnačka, modifikovať jeho nastavenia, definovať pre korytnačku reakcie na rôzne udalosti, vlastné premenné a procedúry, a to v interaktívnom režime aj pomocou príkazov,
- vytvárať a pracovať s rôznymi triedami objektov v prostredí Imagine v interaktívnom režime aj pomocou príkazov,
- generovať viacero korytnačiek s danými vlastnosťami,
- definovať nové triedy korytnačiek a využívať ich pri vývoji projektov,
- využívať možnosti prostredia Imagine na sieťovú spoluprácu, posielanie textov, objektov aj inštrukcií.

Preverenie výstupných vedomostí

Na preverenie vedomostí navrhujeme vypracovať **jedno** z uvedených záverečných zadaní. Odporúčame hodnotenie absolvoval/neabsolvoval.

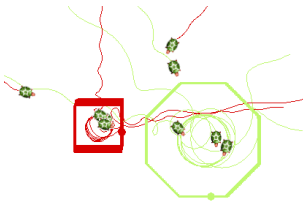


Záverečné zadanie – variant 1	Vytvorte projekt na navliekanie korálikov na niť. Pri ťahaní korytnačky budeme kresliť niť a zároveň zbierať body, ktorými prechádza, do zoznamu `cesta`. Projekt bude obsahovať dve tlačidlá. Pri stlačení prvého sa vytvorí korálik, ktorý sa začne navliekať na niť - postupne bude prechádzať po jednotlivých bodoch nite. Druhý korálik sa navlečie len po predposledný bod nite atď. Druhým tlačidlom zrušíme všetky koráliky a zmažeme stránku.

Návod	Zbieranie bodov riešte podľa návodu k projektu *Zbieranie bodov* v kapitole 1. Okrem udalosti **priŤahaní** definujte korytnačke aj udalosť **priĽavomDolu** - zoznam `cesta` nastavte na prázdny.
	
	Pre koráliky definujte triedu `korálik`. Koráliky budú mať:
	• premennú `mojaCesta` - v nej si bude pamätať body, po ktorých musia prejsť pri navliekaní,
	• vypnuté pero,
	• tvar krúžku definovaný návodom na kreslenie,
	• udalosť `priVytvorení` - cyklus - korálik sa postupne presúva na pozície v zozname `mojaCesta`.
	
	Definujte príkaz `navleč`, ktorý v prípade, že zoznam `cesta` nie je prázdny:
	• vytvorí nový korálik s potrebnými nastaveniami:
	  ○ náhodná farba pera,
	  ○ premennej `mojaCesta` priraďte hodnotu z globálnej premennej `cesta`,
	• skráti premennú `cesta` o posledný prvok.

Vylepšenie - korálik nenavliekajte na niť pomocou cyklu, ale procesu: zakaždým presuňte korálik na prvý bod zoznamu `mojaCesta` a následne skráťte zoznam `mojaCesta` o prvý prvok. Ak už je zoznam `mojaCesta` prázdny, zastavte proces príkazom `zastavMa`.

Záverečné zadanie – variant 2	Naprogramujte naháňačku korytnačiek z kapitoly 2 využitím tried `prasiatko` (korytnačky, ktoré sú naháňané) a `poľovník` (korytnačky, ktoré naháňajú). Projekt bude obsahovať dve tlačidlá. Prvé tlačidlo pridá nové prasiatko a k nemu niekoľko poľovníkov. Druhé tlačidlo **zastaví proces naháňania**, zruší všetky objekty a zmaže stránku.
Návod	Objekty z triedy `prasiatko` budú mať: • náhodnú farbu pera, hrúbku pera 2, • premennú `uhol` – uhol otáčania (tým vlastne definujeme útvar, po ktorom prasiatko chodí) – hodnotou nech je ľubovoľný prvok zo zoznamu `[170 144 120 90 72 60 45]`, • udalosť `priVytvorení` – spustí dva procesy: jeden pre pohyb (napr. každých 10 ms), druhý pre otáčanie o `uhol` (napr. každých 1000 ms). Objekty z triedy `poľovník` budú mať: • náhodnú farbu pera, • premennú `krok` – bude určovať krok (rýchlosť) naháňania - 1, 2 alebo 3, • premennú `mojePrasa` – meno prasiatka, ktoré daný poľovník naháňa. Definujte príkaz `pridaj`, ktorý vytvorí na náhodnej pozícii jedno prasiatko a k nemu na náhodných pozíciách niekoľko poľovníkov. Pri vytváraní poľovníkov „zbierajte“ ich mená (`novéMeno`) do zoznamu `poľovníci` podobne, ako sme zbierali body v kapitole 1. Po vytvorení spustite pre nových poľovníkov proces naháňania: poľovníci sa **postupne** natočia k svojmu prasiatku a prejdú `krok`. **Proces pomenujte** (aby ste ho vedeli zastaviť)!



Príkaz `pridaj` bude mať približne takúto štruktúru:

- vytvorenie prasiatka
- zapamätanie mena nového prasiatka v premennej
- nastavenie zoznamu poľovníkov na prázdny
- cyklus pre vytvorenie poľovníkov:
  - vytvor poľovníka s potrebnými nastaveniami (krok, prasa, ...),
  - pridaj meno poľovníka do zoznamu poľovníkov
- spustenie procesu naháňania postupne pre každú korytnačku zo zoznamu poľovníkov

Vylepšenie: Pri vytvorení nastavte poľovníkom rovnakú farbu pera ako má ich prasa.

Literatúra a použité zdroje

[1] **Blaho**, A., **Kalaš**, I. (2005) *Tvorivá informatika 1. zošit z programovania*. Bratislava: SPN, 2005, 48 strán. ISBN 80-10-00019-1

[2] **Hrušecká** A., **Kalaš** I. *Programovanie v prostredí Imagine*, Bratislava: Metodicko-pedagogické centrum

[3] **Salanci** Ľ. (2001) *Networking in Logo, zborník medzinárodnej konferencie Eurologo 2001 - A Turtle Odyssey*, OCG, ISBN 3-85403-156-4

[4] http://edi.fmph.uniba.sk/~tomcsanyi/dvui/PeterTomcsanyiPoskole2004.pdf

[5] http://www.di.unito.it/~barbara/MicRobot/AttiEuroLogo2007/proceedings/P-Tomcsanyi.pdf

[6] http://edi.fmph.uniba.sk/~lehotska/praca/eurologo05.pdf

[7] http://infovekacik.infovek.sk

[8] http://www.r-e-m.co.uk/logo/?comp=imagine

Účastníkom vzdelávania odporúčame pre ďalšie štúdium zdroje [1] a [6].

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

Autori © PaedDr. Daniela Bezáková, PhD.
Mgr. Peter Kučera
RNDr. Gabriela Lovászová, PhD.
RNDr. Peter Tomcsányi

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

Podnázov Programovanie 4 (Imagine)

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti PaedDr. Monika Tomcsányiová, PhD.
Mgr. Valentína Gunišová

Počet strán 36

Náklad 400 ks

Prvé vydanie, Bratislava 2009

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 978-80-8118-017-0