Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach

Žilina 2.–5. júla 2009

MULTIPLATFORMOVÝ VÝVOJ INTERAKTÍVNEJ APLIKÁCIE – HRY PYTHON A PYGAME

MORAVČÍK, Milan, (SK)

1 Úvod

Vývoj interaktívnych aplikácií je jedným z najťažších, najtvorivejších a najkomplexnejšcích programátorských problémov. Veľmi atraktívnou aplikáciou tohto druhu sú počítačové hry. Tešia sa veľkej obľube mladších i starších hráčov, no málokto sa zamyslí, čo všetko sa za vytvorením hry skrýva a ešte menej používateľov sa pokúsi jednoduchú hru vytvoriť.

Tvorba počítačovej hry začína vymyslením scenára, pokračuje tvorbou grafických prvkov, animácií, ozvučenia a končí programovaním, ktoré poskladá jednotlivé „prísady“ do jedného celku. Teda vývoj hry je komplexná tvorivá práca, ktorá prinajmenšom vyžaduje programátorské, režisérske, grafické a estetické zručnosti. Nie je výnimkou, skôr pravidlom, že za vývojom počítačovej hry je tím niekoľkých ľudí. *Pohľad na tvorbu hry bude v našom príspevku veľmi jednoduchý, laický – chceme zaujať začínajúceho programátora, nie profesionálny vývojársky tím.*

Vývoj počítačovej hry je úžasnou motiváciou pre začínajúceho programátora. Rieši pritom rôzne programátorské problémy. Prirodzene sa dostáva k otázkam použitia údajových štruktúr (organizovanie scény, udržiavanie množín objektov), časovačom (práca s animáciami), udalostiam (interaktívnosť s užívateľom) a pod. Zoznamuje sa s vývojovým prostredím, možnosťami programovacieho jazyka a knižníc, získava základné zručnosti pre prácu s grafikou, zvukmi a organizáciou svojej vlastnej práce.

Prečo je vývoj počítačovej hry atraktívny aj pre začínajúceho programátora:

1. výsledkom je nový jedinečný produkt, s ktorým sa môže pochváliť a prezentovať tak svoje programátorské zručnosti (ale aj iné),

2. tvorí program ktorý je interaktívny, zábavný, môže byť aj poučný (edukačný softvér),
3. celý proces vývoja je komplexný, tvorivý a nekladie medze fantázii autora.

V súčasnej dobe má programátor k dispozícii množstvo vývojových technológií. Každá technológia, v užšom kontexte jazyk, prostredie, knižnice a pod., veľakrát prináša odlišné spôsoby riešenia rovnakého problému. Ostáva na pleciach skúsenejších programátorov, pre ktorú alternatívu sa rozhodnú. V našom príspevku sa zameriame na technológie a postupy, ktoré sú zaujímavejšie pre jednotlivcov – začínajúcich, resp. mierne pokročilých programátorov. Na konkrétnom príklade predstavíme programovací jazyk Python a knižnicu Pygame, ako jednu z možných alternatív vývoja jednoduchej interaktívnej aplikácie – ktorou môže byť hra, ale aj edukačný softvér pre mladších používateľov.

2 Python

Programovací jazyk Python je obľúbeným a pomerne rozšíreným jazykom o ktorom možno nájsť množstvo kníh a elektronických materiálov [5], preto len v krátkosti pripomenieme, že jazyk Python vyvinul v roku 1990 Holanďan Guido van Rossum. *Ide o moderný, dynamický, interpretovaný programovací jazyk, ktorý umožňuje objektovo orientované, štruktúrované aj funkcionálne programovanie*. Patrí medzi jazyky z najjednoduchšou a najčitateľnejšou syntaxou, ktorá je taktiež rýchla na zápis zdrojového kódu programu. Zároveň je stručný na vyjadrovanie sa, čo znamená, že rovnaký program napísaný v inom jazyku presiahne dvoj až trojnásobný počet riadkov, ako program napísaný v Pythone.

Ukážka krátkeho programu, ktorý spočíta výskyt jednotlivých slov v textovom súbore `slova.txt` (výsledok program zapíše do údajovej štruktúry – Slovník):

```
f = open('slova.txt', 'r')
slovnik = {}
for riadok in f.readlines():
    for slovo in string.split(riadok):
        if slovnik.has_key(slovo):
            slovnik[slovo] = slovnik[slovo] + 1
        else:
            slovnik[slovo] = 1
```

Mnohí menej skúsení programátori považujú Python za zastaralý jazyk, opak je však pravdou. Python patrí medzí moderné skriptovacie programovacie jazyky, poskytuje plnoautomatické riadenie pamäte, operácie na vyššej úrovni abstrakcie, jednoduchú prácu s údajovými štruktúrami (zoznamy, slovníky, *n*-tice a pod.). Počas vývoja tak programátor nemusí riešiť technické detaily, sústreďuje sa viac na riešenie koncepčných problémov tvorby aplikácie.

Platformová nezávislosť [1], licenčná politika[2] a jednoduchosť syntaxe jazyka, motivovali viaceré univerzity, kde sa Python stal jazykom na úvodných kurzoch programovania [6–8].

Použitie jazyka v rôznych oblastiach umožňujú knižnice (v kontexte s jazykom Python sa nazývajú moduly), ktorých je v prípade jazyka Python obrovské množstvo. Softvéroví vývojári sa s nimi bežne stretávajú pri:

1. vytváraní užívateľského rozhrania (PyQT, PyGTK, wxPython, dynWin, Tix),
2. spravovaní údajov a databáz (PyXpath, MySql-python, PySQLite, PyGreSQL, psycopg),
3. práci s grafikou (PIL, PyChart, gdModule),
4. vytváraní hier (Pygame, Pyglet, PySoy, PyOpenGL) a pod.

3 Pygame

Pygame je multiplatformový modul jazyka Python. Je navrhnutý pre jednoduchý a rýchly vývoj multimediálneho interaktívneho softvéru. Pygame rozširuje funkčnosť jazyka Python využívajúc knižnicu SDL. Knižnica SDL je sčasti napísaná v jazyku C pre nízkoúrovňový prístup k audiu, externým ovládacím perifériám a k hardvéru. Je tiež multiplatformová a svojím výkonom a možnosťami použitia je porovnateľná napr. s DirectX. Pygame sa skladá z niekoľkých častí, pričom každá z nich pomáha riešiť často sa objavujúce programátorské problémy, ako napr.

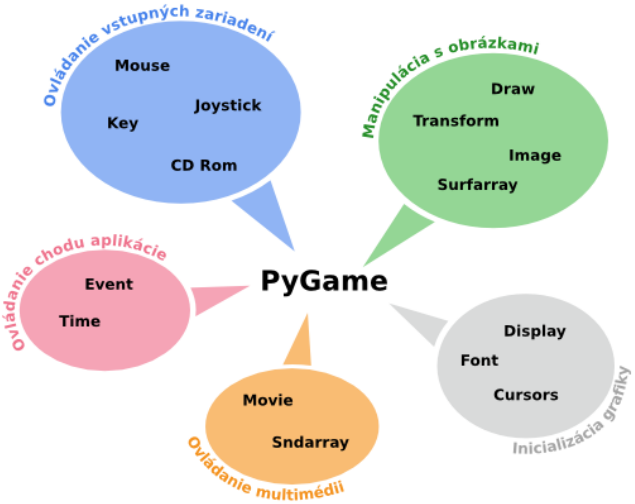
1. ovládanie vstupných zariadení (myš, klávesnica, joystick, . . .),
2. práca s udalosťami, časovačom,
3. manipulácia s grafikou hry (inicializácia obrazovky, práca s textom, . . .),
4. práca s rôznymi obrázkovými formátmi (načítanie, zobrazenie, uloženie, upravovanie aj na nízkej prístupovej úrovni),
5. prehrávanie multimediálnych formátov (audio, video).

4 Vytvorenie jednoduchej neinteraktívnej hry

Na začiatok ukážeme, ako možno naprogramovať jednoduchú hru v jazyku Python a v module Pygame. Hlavnou postavou je Ufón, ktorý sa po obrazovke pohybuje podľa stlačených

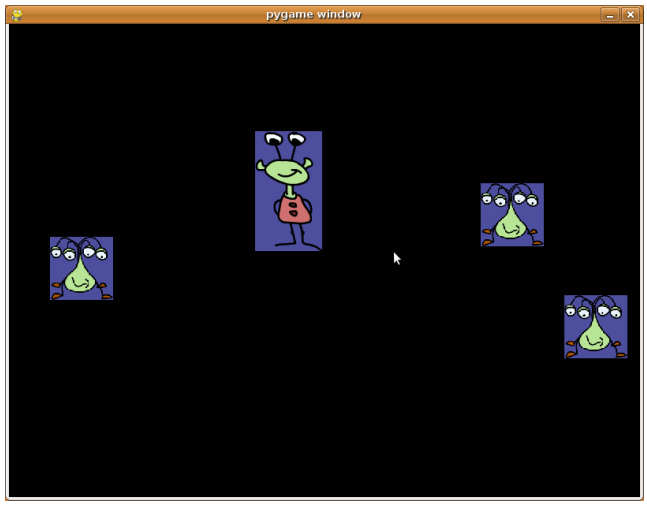
[1] Interpreter Pythonu je implementovaný pre viaceré platformy, netreba však zabúdať že mnohé z často používaných modulov existujú len v prípade OS GNU/Linux, BSD, Mac OS a MS Windows.

[2] Licencia je na pár odlišností totožná s open source licenciou, je voľne distribuovateľný, použiteľný aj pre komerčné učely. O presné vymedzenie licencie sa stará organizácia Python Software Foundation (PSF) [online cit. 29. 5. 2009] `http://python.org/psf`.



Obr. 1: Moduly knižnice Pygame. Pohľad na moduly rozdelené do piatich skupín podľa ich významu v použití pri tvorbe interaktívnej aplikácie: ovládanie vstupných zariadení, ovládanie chodu aplikácie, ovládanie multimédií, inicializácia grafiky a manipulácia s obrázkami

klávesov hore, dole, vpravo, vľavo. Na scéne sa vyskytne aj niekoľko príšer. Každá z príšer sa pohybuje náhodným smerom a náhodnou konštantnou rýchlosťou.



Obr. 2: Ukážka obrazovky prvej verzie hry

Zatiaľ si ukážeme naprogramovanie len takejto jednoduchej funkcionality, neskôr budeme hru zdokonaľovať, čím bude pre používateľa zaujímavejšia, ale aj programátorsky náročnejšia. Vytvorenie hry rozdelíme na štyri časti:

1. inicializácia hry,
2. vytvorenie grafických prvkov,
3. vykreslenie grafických prvkov,
4. rozhýbanie grafických prvkov.

Inicializácia hry spočíva vo vložení potrebných modulov a inicializácii obrazovky. Keďže budeme používať modul Pygame, musíme ho na začiatku vložiť (import pygame). Vložili sme aj ďalšie dva moduly, ich opodstatnenie si vysvetlíme neskôr.

```
import sys, pygame, random
pygame.init()
size = 800, 600
# premenná pomocou ktorej budeme pristupovať ku grafickej ploche hry
screen = pygame.display.set_mode(size)
```

Vytvorenie grafických prvkov s využitím modulu Pygame je veľmi jednoduché. Najprv vytvoríme grafický prvok hry *Ufón*.

Ufón	`ufon = pygame.image.load("ufon.gif")` `ufonrect = ufon.get_rect()`

Pre každý grafický objekt si budeme pamätať tzv. *Surface*, ktorý predstavuje „obrázkovú premennú“. Takáto premenná sa bude vykresľovať na obrazovku (v našom prípade je ňou premenná *ufon*). Taktiež si budeme pamätať rozmery a umiestnenie obrázkovej premennej. Využijeme na to premennú *ufonrect*.

Podobným spôsobom vytvoríme ďalší grafický prvok *Príšerka*. Navyše si budeme pamätať vektor jej pohybu. Pri jeho generovaní sme využili funkciu *randint* z modulu *random* (modul *random* sme vložili na začiatku pri inicializácii hry).

Príšerka	`priserka1 = pygame.image.load("priserka.gif")`
	`priserkarect1 = priserka.get_rect()`
	`a = random.randint(-10, 10)`
	`b = random.randint(-10, 10)`
	`vektor1 = (a, b)`

Vykreslenie grafických prvkov do grafickej plochy. Pripomíname že grafická plocha, ktorej veľkosť je 800×600, je uložená v premennej *screen*. Pomocou metódy *blit* vykreslíme ľubovoľný grafický objekt do plochy *screen*. Po vykreslení všetkých grafických objektov musíme zavolať metódu *pygame.display.flip()*, ktorá prekreslí obrazovku našou grafickou plochou.

```
screen.blit(ufon, ufonrect)
screen.blit(priserka1, priserkarect1)
pygame.display.flip()
```

Rozhýbanie grafických prvkov zatiaľ v nekonečnom cykle. Aby sa zmeny nevykonávali príliš rýchlo musíme vykonávanie nášho programu spomaliť. V takomto prípade sa v hrách využíva „zamrznutie“ scény hry na krátku dobu. Takúto situáciu môžeme vytvoriť pomocou „časovača“, ktorý pozastaví program na niekoľko milisekúnd/sekúnd.

```
clock = pygame.time.Clock()
while True:
    screen.fill(0)
    screen.blit(ufon, ufonrect)
    screen.blit(priserka1, priserkarect1)
    pygame.display.flip()
    # časovač ''tikne'' 25 krát za sekudnu = pozastaví program na
40 miliseknúnd
    clock.tick(25)
    priserkarect1.top = priserkarect1.top + vektor1[0]
    priserkarect1.left = priserkarect1.left + vektor1[1]
```

5 Vylepšenie hry, zavedenie interaktívnych prvkov

Doposiaľ naprogramovaná hra je neinteraktívna a má veľa nedostatkov. Napr. príšerka sa neustále pohybuje aj za okrajmi obrazovky, kde ju nie je vidieť. Prvok hry ufón je zatiaľ bez pohybu – budeme ho ovládať klávesovými šípkami a zároveň zabezpečíme aby sa nedostal mimo grafickej plochy. Taktiež náš program bude reagovať na kolízie grafických prvkov – ak má príšerka s ufónom prienik, zmizne z obrazovky. Ak zmiznú všetky príšerky hra sa ukončí výpisom: Vyhral si! Vylepšenie hry rozdelíme do piatich celkov:

1. vytvorenie tried pre grafické objekty (ufón, príšerka),
2. rozhýbanie ufóna,
3. riešenie kolízie ufóna a príšerky,
4. ukončenie hry,
5. hlavné telo programu.

Hru vylepšíme aj z pohľadu programátora. Z dôvodu lepšej organizácie grafických prvkov hry *vytvoríme dve triedy*, jednu z nich pre prácu s *ufónom* a druhú pre prácu s *príšerkou*. Pri vytvorení objektu (inštancie) triedy *Ufon* sa volá konštruktor, ktorý je v Pythone deklarovaný ako *def __Init__()*. Kedže Python je interpretovaný jazyk, pred každú premennú, ktorú chceme používať v rámci celej triedy píšeme slovíčko *self* (to isté slovíčko je prvým argumentom každej metódy).

```
class Ufon:
    def __init__(self):
        self.obr = pygame.image.load("ufon.gif")
        self.rect = self.obr.get_rect()
```

Ufóna rozhýbeme pomocou metódy *pohni()*. Jej úlohou je zistiť stav stlačeného klávesu a následný pohyb ufóna príslušným smerom. Ak bol napr. stlačený kláves hore (*if pygame.key.get_pressed()[pygame.K_UP]:*) zmenšíme pozíciu ľavého horného rohu obdĺžnika ufóna o 6 bodov. Zároveň testujeme (*if rect.top < 0:*), či nová pozícia nie je mimo okrajov obrazovky.

```
class Ufon:
    def __init__(self):
        ...
    def pohni(self):
        rect = self.rect
        if pygame.key.get_pressed()[pygame.K_UP]:
            rect.top = rect.top  - 6
            if rect.top < 0:
                rect.top = 0
        elif pygame.key.get_pressed()[pygame.K_DOWN]:
            rect.top = rect.top  + 6
            if rect.bottom > size[1]:
                rect. bottom = size[1]
        elif pygame.key.get_pressed()[pygame.K_LEFT]:
            rect.left = rect.left  - 6
            if rect.left < 0:
                rect.left = 0
        elif pygame.key.get_pressed()[pygame.K_RIGHT]:
            rect.left = rect.left  + 6
            if rect.right > size[0]:
                rect.right = size[0]
```

Grafický prvok *Príšerka* uložíme do triedy *Priserka*. Konštruktor je podobný ako konštruktor *Ufona*, no naviac obsahuje presun objektu do približného stredu obrazovky a ďalšiu premennú – vektor pohybu o náhodnej veľkosti.

```
class Priserka:
    def __init__(self):
        self.obr = pygame.image.load("priserka.gif")
        self.rect = self.obr.get_rect()
        self.rect.topleft = 400, 300
        a = random.randint(-10, 10)
        b = random.randint(-10, 10)
        self.vektor = [a, b]
```

Príšerku rozhýbeme metódou *pohni()*, presunieme ju o veľkosť vektora. Ak *príšerka* prekročí hranice grafickej plochy zmení sa jej vektor podľa pravidla: *uhol odrazu = uhlu dopadu*. V zásade sa vektor pohybu rovnako zmení keď sa objekt dostane za hornú a dolnú hranicu (horizontálny odraz), alebo taktiež za pravú a ľavú hranicu plochy (vertikálny odraz).

```
class Priserka:
    def __init__(self):
        ...
    def pohni(self):
        rect = self.rect
        rect.left = rect.left + self.vektor[0]
        rect.top = rect.top + self.vektor[1]
        # vertikálny odraz
        if (rect.left < 0) or (rect.right > size[0]):
            self.vektor[0] = -1 * self.vektor[0]
        # horizontálny odraz
        if (rect.top < 0) or (rect.bottom > size[1]):
            self.vektor[1] = -1 * self.vektor[1]
```

Do triedy *Ufon* doprogramujeme metódu *kolizia()*, ktorá testuje prienik obrázku *ufona* s obrázkom *príšerky*. V prípade ich prieniku je *príšerka* odstránená zo zoznamu príšer.

```
class Ufon:
    def __init__(self):
        ...
    def pohni(self):
        ...
    def kolizia(self, priserka):
        rect = self.rect
        if rect.colliderect(priserka.rect):
            priserky.remove(priserka)
```

Ak v scéne nie sú žiadne príšerky (*if not priserky:*), hru ukončíme výpisom *Vyhral si*!

```
if not priserky:
    font = pygame.font.Font(None, 36)
    text = font.render(Vyhral si!, 1, (255, 255, 255))
    textpos = pygame.Rect(400, 300, 0, 0)
    screen.blit(text, textpos)
```

Hlavný telo programu našej aplikácie môže vyzerať napr. takto:

1. vytvoríme inštanciu triedy *Ufon*,
2. inštancie triedy *Priserka* ukladáme priamo do premennej *priserky* (premenná je typu *zoznam*, čo je jedna zo vstavaných údajových štruktúr jazyka Python),
3. vytvoríme objekt *clock* pre ovládanie časovača hry,
4. v hlavnom cykle (*while True:*) aplikácie:

   (a) testujeme systémové ukončenie aplikácie,
   (b) prekresľujeme plochu (čiernou farbou),

(c) zabezpečujeme pohyb ufóna a príšeriek,
(d) testujeme kolízie a koniec hry.

```
# vytvorenie objektu ufon = inštancia triedy Ufon
ufon = Ufon()
# inicializácia zoznamu priserky
priserky = []
# vytvorenie a pridanie 3 príšeriek do zoznamu
priserky.append(Priserka())
priserky.append(Priserka())
priserky.append(Priserka())
# vytvorenie objektu časovača
clock = pygame.time.Clock()
# hlavný cyklus programu
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    # prekreslenie obrazovky čiernou farbou
    screen.fill(0)
    screen.blit(ufon.obr, ufon.rect)
    for item in priserky:
        screen.blit(item.obr, item.rect)
    # ''zamrznutie'' programu na 40 miliseknúnd
    clock.tick(25)
    pygame.display.flip()
    # pohyb ufóna
    ufon.pohni()
    # pohyb príšeriek s riešením kolízie = prienik s ufónom
    for item in priserky:
        item.pohni()
        ufon.kolizia(item)
    # testovanie konca hry
    if not priserky:
        font = pygame.font.Font(None, 36)
        text = font.render(Vyhral si!, 1, (255, 255, 255))
        textpos = pygame.Rect(400, 300, 0, 0)
        screen.blit(text, textpos)
```

6 Ďalšie tipy, triky a drobné vylepšenia

Konceptuálne vylepšenia

1. **Iný spôsob reagovania na udalosti** – každých 40 milisekúnd zisťujeme stav stlačenia kláves, ak užívateľ stlačil a pustil kláves tesne pred testovaním, nestihneme na túto

zmenu zareagovať. V praxi sa často tento problém rieši pomocou zásobníka udalostí, napr.:

```
for event in pygame.event.get():
   if event.type == pygame.KEYDOWN:
       print event.key
```

1. **Čo sme pre jednoduchosť zanedbali**

   (a) *zápis cesty k súborom* je odlišný pri unixových a windowsových systémov. Na odstránenie nejednotnosti môžeme použiť príkaz `os.path.join(adresar, subor)`

   (b) *ošetrenie vstupných súborov* – pre zvýšenie bezpečnosti programu by sme mali testovať existenciu a neporušenosť vstupných súborov. Programátori tento problém riešia tzv. *výnimkami*. Pre bezpečné načítanie obrázkov možno použiť konštrukciu:

   ```
   try:
      obr = pygame.image.load("obr")
   except pygame.error, message:
      print 'Nemôžem načítať obrázok:', "obr"
      raise SystemExit, message
   ```

2. **Komentáre a dokumentácia:**

   ```
   # existujú len jednoriadkové komentáre
   """dokumentácia je tiež súčasťou jazyka, môžeme ju vyvolať
   pomocou príkazu: help(nazov_triedy)"""
   ```

Drobné tipy, triky

1. **Vykresľovanie bitmapy do pozadia** – rovnakým spôsobom ako vykresľovanie objektov:

   - ```
     pozadie = pygame.image.load("obr")
     screen.blit(pozadie) # namiesto screen.fill(0)
     ```

1. **Nastavenie grafickej plochy na celú obrazovku**, tzv. *fullscreen*, môžeme urobiť aj počas behu programu príkazom `pygame.display.set_mode(size, pygame.FULLSCREEN)`.
2. **Prekeslenie časti obrazovky** – doteraz sme pomocou príkazu `pygame.display.flip()` prekresľovalicelú obrazovku. Niekedy sa nám hodí prekresliť len malú časť obrazovky. Príkazom `pygame.display.update(rect_list)`, kde *rect_list* je zoznam jedného alebo viacerých obdĺžnikov určíme časti obrazovky ktoré sa prekreslia.
     ```

3. **Skryť kurzor myši** – `pygame.mouse.set_visible(False)`, ak ho potrebujeme opätovne zobraziť `...set_visible(True)`. Systémový kurzor môžeme skryť, ak ho chceme zameniť za obárzok. Obrázok potom presúvame pri udalosti zmeny polohy myši.
4. **Priesvitné pozadie obrázkov** – príkazom `obr=pygame.image.load("obr")` vytvoríme objekt typu *pygame.Surface*, ktorého metóda *set_colorkey()* umožňuje nastavenie (aj percentuálnej) priesvitnosti. V našom prípade nastavenie priesvitnosti pre objekt *ufon*:

```
obr.set_colorkey(obr.get_at((0,0)), pygame.RLEACCEL)
```

7 Záver

Programovanie interaktívnych programov, predovšetkým hier, patrí medzi prvé projekty začínajúcich programátorov. V článku sme predstavili jednu z možných alternatív, prezentovali sme vývoj pomocou skriptovacieho jazyka Python a modulu Pygame. Zvolený jazyk a modul má svoje silné a slabé stránky. Nás zaujal pohodlný a rýchly vývoj, a to vďaka jednoduchej syntaxi moderného jazyka vysokej úrovne a jednoduchému spracovaniu grafiky aplikácie pomocou rozširujúceho modulu.

Modul Pygame sa najčastejšie používa pri tvorbe jednoduchých edukačných hier, ako sú napr. *Gcompris* alebo *Schoolsplay*. Jeho použitie v iných typoch programov je zriedkavé, väčšinou sa kombinuje s použitím ďalších modulov.

Plusy	**Mínusy**
(1) Jednoduchá konfigurácia a bezproblémová inicializácia na viacerých platformách,	(1) Nerieši otázku GUI, treba použiť ďalší modul resp. knižnicu ako je napr. GTK,
(2) licencia LGPL,	(2) chýba podpora viacerých audio a video formátov,
(3) interakcia s rôznymi vstupnými zariadeniami (nie len myš, klávesnica ale aj pákové zariadenie tzv. joystick alebo CD–ROM mechaniky),	(3) chýba nahrávanie zvuku z mikrofónu (tento nedostatok rieši napr. modul *PyAudio*, resp. *PyMedia*),
(4) široká podpora grafických obrázkových formátov (`jpg`, `png`, `gif`, `bmp`, `pcx`, `tif`, ...),	(4) v niektorých prípadoch chýba vizuálne vývojové prostredie,
(5) zobrazovanie a generovanie `TrueType` fontov,	(5) samotný beh programu je pomalší ako pri kompilovaných jazykoch.
(6) predprogramovaná trieda Sprite (organizácia práce s grafickými objektami aplikácie ako napr. vykresľovanie do vrstiev, riešenie kolízií medzi skupinami objektov, ...).	

Príspevok vznikol v rámci grantu číslo UK/370/2009 Univerzity Komenského v Bratislave.

Literatúra

[1] McGUGAN, W.: *Beginning Game Development with Python and Pygame (From Novice to Professional)*, Berkeley: 2007, 316 s., ISBN: 978-1-59059-872-6

[2] MORAVČÍK, M.: *Edukačný softvér pre deti predškolského veku* (písomná práca k dizertačnej skúške), Bratislava: 2009, FMFI UK v Bratislave

[3] HARMS, D., McDONALD, K.: *Začíname programovat v jazyce Python.* Brno: Computer press, 2003, 456 s., ISBN 978-80-251-2161-0

[4] `http://www.root.cz/clanky/letajici-cirkus-20` [online cit. 29. 5. 2009]

[5] Python [online cit. 29. 5. 2009] `http://www.python.org/`

[6] `http://mcsp.wartburg.edu/zelle/python/python-first.html` [online cit. 20. 5. 2009]

[7] KAUKIČ, M.: *Python ako prvý programovací jazyk na VŠ*, 7th International conference, APLIMAT 2008, Slovak University of Technology in Bratislava

[8] `http://www.ece.uci.edu/~chou/py02/python.html` [online cit. 29. 5. 2009]

[9] Pygame – An Open Source Community Project, [online cit. 23. 5. 2009] `http://www.pygame.org/`

[10] Simple Directmedia Layer (SDL) [online cit. 23. 5. 2009] `http://www.libsdl.org/`

[11] SchoolsPlay [online cit. 28. 5. 2009] `http://www.schoolsplay.org/`

[12] GComrpis – edukačný softvér pre deti od 2 do 12 rokov [online cit. 28. 5. 2009] `http://gcompris.net/`

Kontaktní adresa

Milan MORAVČÍK,
Katedra základov a vyučovania informatiky,
FMFI UK v Bratislave,
Mlysnská dolina 1, 842 48 Bratislava,
moravcik@fmph.uniba.sk

Fakulta riadenia a informatiky
Žilinská univerzita

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie OSSConf 2009

2.–5. júla 2009
Žilina, Slovensko