Pampuch

Jan Procházka

Zápočtový program na předmět Programování II (PRG031)

LS 2004 / 2005

I. Osnova

II. Zadání

Jako zápočtový program jsem se tentokrát rozhodl naprogramovat hru, kterou jsem ještě před několika lety miloval a vášnivě hrál. Je jí hra „Pampuch“.

Hráč představuje požírače puntíků (ovládaného kurzorovými klávesami). V každém kole (tj. na každé mapě) se kromě něho, puntíků a překážek objeví vždy ještě 4 roboti, kteří ho honí. Při srážce s robotem je hráč zabit (má 4 náhradní životy). Do dalšího kola hráč postupuje, není-li na mapě již žádný puntík. Cílem je nasbírat co největší možné množství puntíků.

Roboti mají umělou inteligenci přesně stejnou jako v originálu hry, tj. při srážce s jiným robotem se chovají, jako by narazili do stěny, při naražení do stěny se odrazí směrem k Pampuchovi, jinak musí jet rovně. Pokud narazí a mají blokované oba směry k Pampuchovi, zastaví se. Obtížnost vyšších kol je řešena zvýšením počtu odrazových ploch, což přináší častější směrování robotů na Pampucha a dojem, že jedou skutečně za ním.

III. Program – dekompozice

Hra je vystavěna na 4 formulářích, a proto obsahuje 4 moduly.

Veškeré dění se odehrává v hlavním okně, které pochází z unity *„Pampuch.pas“*.

Dále bylo potřeba načíst uživatelovo jméno při dosažení rekordu (unita *„Rek.pas“*) a vypsání nejlepších deseti skóre (unita *„Nej.pas“*). Posledním a ne nezbytným oknem v programu je okno O aplikaci (unita *„Oaplikaci.pas“*).

Pampuch

Tato unita obsahuje veškerý kód týkající se vlastní hry. V ostatních modulech jsou obsaženy již pouze drobné kusy kódu, které neprovádí nic zásadního. Tato unita je natolik rozsáhlá, že je jí věnována celá kapitola V.

Nej

Tabulka deseti nejlepších dosažených výsledků. Je zde obsaženo její načtení a uložení. A samozřejmě subrutina, která zařadí do tabulky nový rekord.

Rek

Okno, do něhož si uživatel vyplní své jméno a klávesou ENTER ho potvrdí. Vyvolává ho modálně unita *Nej* a to při zařazování nového rekordu.

Oaplikaci

Pouze zobrazení a skrytí okna „O aplikaci“ s označením verze a dalšími identifikačními údaji.

IV. Použité datové typy

(O deklaraci oken se nebudu zmiňovat, protože je automatická a zájemce si ji může nalézt v přiloženém zdrojovém kódu.)

Představu světa bylo potřeba převést do objektů, ty jsou všechny deklarovány v unitě *Pampuch* a to následovně:

Pampuch a roboti

Protože Pampuch a roboti toho mají dost společného, vytvořil jsem jim společného rodiče (ten je zděděn z třídy *TImage*, protože je tyto objekty také potřeba vykreslovat):

```
TPotvora= class(TImage)
private
  dX, dY: integer;  //pohyb po mapě
public
  mX, mY: integer;  //souřadnice na mapě (1..16,1..24)
  mX0, mY0: integer;//startovní souřadnice na mapě
  Faze: integer;    //fáze pohybu (1..4)
  Stopped: boolean; //má se animovat?
  Smer: word;       //kam jdeš? (cU, cD, cL, cR);
end;
```

Z této třídy jsem zdědil třídu, jejíž instancemi jsou roboti:

```
TRobot= class(TPotvora)
  constructor Create(AOwner: TComponent); override;
public
  Typ: integer;       //typ robota.. Levo-/Pravotočivý (signum)
  function Kolize: boolean; //narazil jsem?
  procedure Refresh; //udělej krok, je-li to možné
  procedure ResetP;  //postav se na startovní pozici
end;
```

Protože je velmi často potřeba pracovat se všemi roboty najednou, bylo vhodné umístit si je do pole a některé cykly ve tvaru – s každým robotem udělej totéž – pojmenovat. Další výhodou je, že toto umožnilo zapouzdřit grafiku robotů do jejich objektu bez toho, aby měl každý svůj exemplář.

```
TRoboti= class
public
  Grafika: array[1..4] of TBitmap;
           //obrázky fáze pohybu, které se zobrazují
  Robot: array[1..4] of TRobot;   //pole robotů
  constructor Create(AOwner: TObject);
  procedure Refresh;  //každý robot udělá krok, je-li to možné
  procedure ResetP;
            //postaví všechny roboty na startovní pozici
  procedure Paint;  //vykreslí všechny roboty (bez animace)
  procedure Stop;   //zastaví všechny roboty
  destructor Destroy; override;  //smaže roboty a grafiku
end;
```

Třída, jejíž instancí je Pampuch:

```
TPampuch= class(TPotvora)
  constructor Create(AOwner: TComponent); override;
private
  Pmlask: Pointer;    //mlaskavý zvuk (ukazatel na blok paměti)
  Smlask: String;     //cesta k mlaskavému zvuku
  Pkilled: Pointer;   //zvuk umírání (opět ukazatel)
  Skilled: String;    //cesta ke zvuku umírání
public
  Grafika: array[1..4{faze}, 0..4{smrt+smer}] of TBitmap;
                      //obrázky fáze pohybu, které se zobrazují
  Ziv: boolean;       //žije ještě?
  Zivotu: integer;    //počet náhradních životů
  Body: integer;      //kolik už jsi sežral puntíků?
  KamChci: word;      //kam bylo naposledy zadáno, aby se šlo
  procedure Refresh; //udělej krok (lze-li) a nakresli se
  procedure ResetP;  //postav se na počáteční pozici
  procedure Ovladani
    (Sender: TObject; var Key: Word; Shift: TShiftState);
                      //reakce na stisk klávesy
  function Bod:boolean;
      //pokusí se sežrat bod a vrátí, jestli byl poslední
  function Oziv:boolean;
      //pokusí se oživit Pampucha a vrátí, zda se to povedlo
  procedure NastavZivoty; //zobrazí počet náhradních životů
  procedure Mlaskni;      //přehraje mlaskavý zvuk
  procedure Zarvi;        //přehraje zvuk umírání
end;
```

Mapa

Je třeba udělat matici, v níž jsou prvky jednotlivá políčka mapy (opět potomci *TImage* kvůli metodě *Paint*):

```
TPole = class(TImage)  //políčko na mapě
public
  Visitable: boolean;      //je možné na políčko vstoupit?
  Typ: (Dot, Void, Wall); //co je na políčku za objekt
  Hodnota: integer;        //Typ, ale stěny se rozlišují
end;
```

Třída shrnující vše, co je třeba si ve hře pamatovat, kromě Pampucha a robotů:

```
TWorld = class
  Casovac: TTimer;  //je třeba animovat v časových intervalech
  constructor Create(AOwner: TComponent);
  destructor Destroy;override;
public
  Level: integer; //číslo aktuálně hraného kola
  sMapa: string;  //adresa mapy pro uživatelem definované mapy
  Grafika: array[0..STEN+1] of TBitmap;
                  //grafika mapy.. 0-void, 1..9-wall, 10-dot
  Mapa: array[1..16,1..24] of TPole;  //vlastní mapa
  Tecek: Word;    //kolik je na mapě ještě nesežraných bodů
  Recordable: boolean;
      //je možné dosáhnout rekordu? (kvůli Saveování)
  isRecord: boolean;    //výsledek patří do tabulky nejlepších
  procedure ResetLevel; //po zabití postav na začátek
  procedure VytvorMapu; //vytvoř políčka mapy a inicializuj je
  procedure NactiMapu;  //načti mapu z proměnné Level/sMapa
  procedure NextLevel;  //načti další kolo (tohle skončilo)
  procedure Animuj(Sender: TObject); //zavolej všem Animuj
  procedure Wait(Time: integer); //dá uživateli čas začít kolo
  procedure Cekej(Sender: TObject);
                        //čas na zahájení vypršel
  procedure Paint();    //překreslí mapu
  function Visitable(y, x: integer): boolean;
                        //lze jít na políčko mapy [y,x]?
  procedure RecordTest; //testuje, bylo-li dosaženo rekordu
end;
```

Uložit / Načíst

Protože ve hře je možnost ukládání a načítání stavu hry, je dobré si pro jednoduchost udělat datový typ obsahující všechny potřebné veličiny pro rekonstrukci hry:

Nejprve ty, jež popisují robota:

```
TRob=Record
  Top, Left: integer;//souřadnice na mapě
  dX, dY: integer;   //pohyb po mapě
  mX, mY: integer;   //souřadnice na mapě (1..16,1..24)
  mX0, mY0: integer; //startovní souřadnice na mapě
  Faze: integer;     //fáze pohybu
  Stopped: boolean;  //má se animovat?
  Smer: word;        //kam jdeš? (cU, cD, cL, cR)
  Typ: integer;      //typ robota.. Levo-/Pravotočivý (signum)
end;
```

Nyní můžeme celý stav hry popsat takto:

```
GamePosition=Record
//TWorld
  Level: integer; //číslo aktuálně hraného kola
  sMapa: string[200];
         //adresa mapy pro uživatelem definované mapy
  Tecek: Word;    //kolik je na mapě ještě nesežraných bodů
  Mapa: array[1..16,1..24] of integer;
         //mapa-co je kde za blok (0-void, 1..9-wall,10-dot)
//TPampuch
  Top, Left: integer; //souřadnice na mapě
  dX, dY: integer;    //pohyb po mapě
  mX, mY: integer;    //souřadnice na mapě (1..16,1..24)
  mX0, mY0: integer;  //startovní souřadnice na mapě
  Faze: integer;      //fáze pohybu
  Stopped: boolean;   //má se animovat?
  Smer: word;         //kam jdeš? (cU, cD, cL, cR)
  Ziv: boolean;       //žije ještě?
  Zivotu: integer;    //počet náhradních životů
  Body: integer;      //kolik už jsi sežral bodů?
//TRoboti
  Rob: array[1..4] of TRob;
end;
```

V. Program – algoritmizace

Nejprve se vytvoří a inicializují objekty, které vznikají dynamicky (až za běhu programu), to se stane hned po spuštění programu v proceduře *FormCreate*:

```
procedure TWokno.FormCreate(Sender: TObject);
begin
  //Nastav rozměry formuláře

  //načti nový kurzor
  //a nainstaluj ho (zařaď ho mezi ostatní kurzory v paměti)

  //zobraz náhradní životy
  //vytvoř Pampucha
  //vytvoř roboty
  //vytvoř svět (mapu)

  //načti první kolo
end;
```

Pak se už začíná hrát. Nejprve se *Casovac (TTimer)* objektu *Svet* nastaví na čekání (dá se uživateli čas prohlédnout si situaci), ale pokud ve vymezené době uživatel nezadá směr, kterým chce jet, rozjede se defaultně nastaveným směrem – doprava. To je koncipováno tak, že se ve velmi krátkých časových intervalech volá procedura *Casovac.OnTime*, která je velmi jednoduchá:

```
procedure TWorld.Animuj(Sender: TObject);
begin
  Roboti.Refresh;     //animuj Roboty
  Pampusak.Refresh;   //animuj Pampucha
end;
```

A tímto jsme se dostali k nejsložitější části programu, metoda *Refresh* provádí u obou typů objektů (*TPampuch* a *TRobot*) totéž: zjistí kolize, podle toho provede reakci příslušného objektu a vykreslí jeho současný stav na obrazovku. Problém je v tom, že každý z těchto objektů to dělá jinak, a proto je třeba řešit je zvlášť. Tedy nejprve pro *TPampuch*:

```
procedure TPampuch.Refresh;
begin
//pokud se má animovat, nastav, že je v další fázi pohybu
//pokud je mrtvý, animuj umírání, a pak se pokus oživit

//nastav, na kterém stojíš políčku mapy
//zjisti, jestli jsi právě nespolkl poslední bod na mapě

//pokud je na políčku, na které se chystáš, stěna, zastav se
//pokud chci zahnout a dá se to, tak se otoč

//umísti obrázek na mapu
end;
```

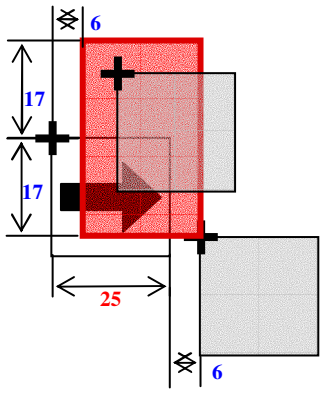
Mnohem složitější je tato metoda pro objekt TRobot, protože je potřeba řešit kolize nejen se stěnou, ale také s ostatními roboty, a navíc se robot při kolizi nezastaví, ale užije své „inteligence“ k nasměrování se na Pampucha:

```
procedure TRobot.Refresh;
const
  Tolerance = 0.75;
     //1-kolik Pampucha je třeba sežrat, než ho to zabije
begin
//pokud se má animovat, nastav, že je v další fázi pohybu
//nastav, na kterém stojíš políčku mapy

//otoč se o 90° směrem k Pampuchovi
  if Kolize {pokud jsi se právě s něčím srazil} then
        //se otoč o 90° směrem k Pampuchovi a ...
        if Kolize {... pokud se tam nedá jít, ...} then
              //... tak se zastav

//umísti obrázek na mapu

//zkontroluj, jestli jsi nesežral Pampucha, ...
//... pokud ano, zabij ho a zastav všechny roboty (kolo končí)
end;
```

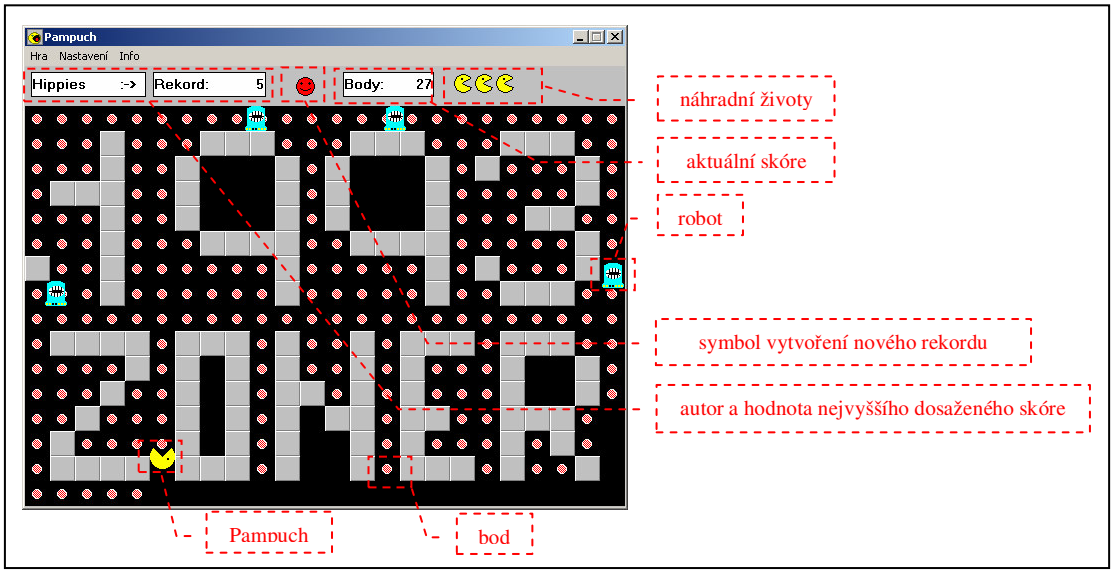
Kde funkce Kolize testuje, jestli došlo ke srážce s jiným robotem či se stěnou:

```
function TRobot.Kolize: boolean;  //je robot v kolizi?
const DIST_U = B_W-8;  //přesah max, který ještě vadí (3/4)
      DIST_L = 6;      //přesah min, který už vadí    (1/4)
begin
  //pokud stojíš, tak vrať vždy True
  //když je přede mnou stěna, tak vrať True
    for {každý robot, který nejsem já} do
          //pokud je před tebou, pak vrať True
  //vrať False
end;
```

To, jestli je daný robot (šedý čtverec) přede mnou (bílí čtverec označený šipkou), se testuje porovnáním vlastností *Top* a *Left* a to následujícím způsobem:



VI. Návod k obsluze – uživatelská dokumentace



Pravidla hry

Hráč představuje žlutého „požírače“ bodů – Pampucha. Jeho cílem je, aby na mapě nezbyl ani jeden bod. Dosáhne-li tohoto vytouženého stavu, znamená to, že je kolo dokončeno a hráč postupuje do dalšího. Bylo-li dokončeno poslední kolo, hra pokračuje opět prvním kolem.

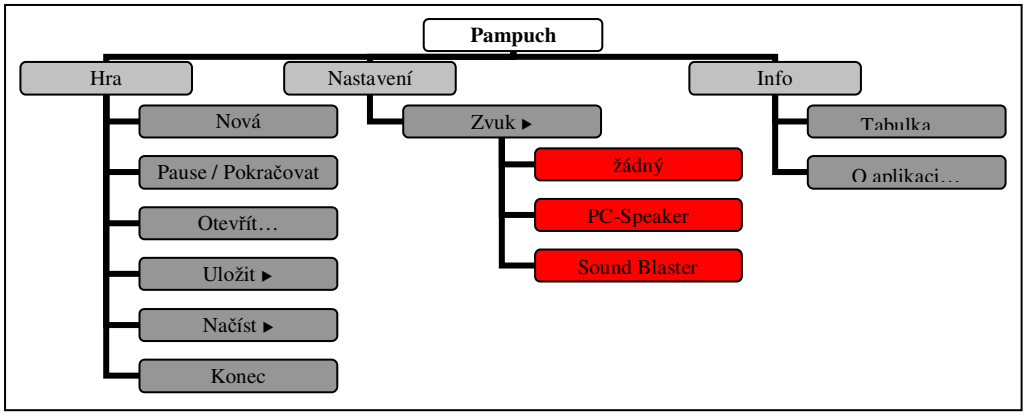
Při kolizi s koncem mapy nebo se stěnou se Pampuch zastaví a čeká na další pokyn.

Aby to ale hráč neměl tak jednoduché, je na mapě kromě něho a bodů ještě čtveřice robotů, která se ho snaží sežrat. Neexistuje jiný způsob boje s nimi, než se jim vyhýbat. Při srážce s robotem je Pampuch zabit. Na začátku hry má však vždy čtyři náhradní životy. Kolik jich hráči ještě zbývá se zobrazuje v pravém horním rohu okna.

Hra se dá ukládat a opětovně načítat, dále je možné kdykoliv načíst libovolnou mapu (postup viz dále). Uložit je možné až čtyři rozehrané mapy. Nutno však upozornit, že se ve hře, která není zahájena jako nová, ale je načtená (ať již jako speciální mapa nebo jako rozehraná), neuznávají dosažené rekordy.

Ovládání hry

Hra se obsluhuje pomocí menu (nebo pomocí klávesových zkratek), které vypadá následovně:



Hra

Nová (F2)	resetuje hru (nastaví 1. kolo se 4 náhradními životy a možností zápisu do tabulky rekordů)
Pause / Pokračovat (F3)	pozastaví a opětovně spustí rozehranou hru (téhož efektu lze dosáhnou kliknutím levým tlačítkem myši kdekoliv na mapě)
Otevřít… (F4)	otevře pomocí dialogového okna libovolnou mapu přístupnou jak na tomto počítači, tak v síti
Uložit ▸	uloží rozehranou hru na vybranou pozici
Načíst ▸	opět načte rozehranou hru uloženou pomocí položky Uložit
Konec (Ctrl+Q)	ukončí program

Nastavení

Zvuk ▸ nastaví, na jakém zařízení se mají přehrávat zvuky (polykání bodů a zabíjení Pampucha), možnosti jsou:

- žádný — nepřehrává se
- PC-speaker — na vestavěném reproduktoru
- Sound Blaster — na zvukové kartě

Info

Tabulka…	zobrazí tabulku deseti nejlepších výsledků
O aplikaci…	zobrazí okno s informacemi o programu

Modularita

Program se vyznačuje tím, že veškerá grafika, zvuky i mapy jsou ve vnějších souborech, takže není obtížné si ho upravit do podoby, která uživateli vyhovuje.

Zvuk

Soubory se zvukem jsou uloženy ve složce *„sounds“*. Zvuk polknutí bodu se jmenuje *„Ham.wav“* a zvuk, který se ozve při zabití Pampucha je v souboru *„Killed.wav“*. Je celkem jedno, jaké zvuky jsou zde obsaženy, ale je vhodné, aby (a pro zvuk *„Ham.wav“* to platí obzvláště) byly co možná nejkratší (max. 1s), jinak totiž hrozí problémy se současným přehráváním mnoha souborů najednou, a tím způsobeným zahlcováním počítače, což se projevuje neplynulým chodem hry.

Grafika

Veškerá grafika je umístěna ve složce *„images“*. Modifikací zdejších souborů lze vytvářet vlastní „skiny“ hry. Navíc lze ve hře používat 9 typů stěn, ačkoliv standardně jsou použity jen 4 typy. Všechny soubory jsou ve formátu *„*.bmp“* o rozměrech 25×25px. Jedná-li se o fázi pohybu (ať již Pampucha či robota), je nastavena dominantní barva na obrázku jako transparentní (výjimku tvoří fáze umírání Pampucha, tam je transparence vypnuta).

Dot	Void		
Wall_1	Wall_2	Wall_3	Wall_4
Robot_1	Robot_2	Robot_3	Robot_4
Dead_01	Dead_02	Dead_03	Dead_04
Pampuch_11	Pampuch_12	Pampuch_13	Pampuch_14
Pampuch_21	Pampuch_22	Pampuch_23	Pampuch_24
Pampuch_31	Pampuch_32	Pampuch_33	Pampuch_34
Pampuch_41	Pampuch_42	Pampuch_43	Pampuch_44

Tabulka 1: Seznam souborů (všechny mají příponu „*.bmp“), které se dají měnit, a jejich defaultní obsah

Mapy

Všechny standardní mapy jsou uloženy ve složce *„maps“*, a ačkoliv to není nutné, doporučuji pro udržení přehlednosti ukládat do této složky i vlastní mapy. Je možné připravit si jak ty mapy, které budou součástí standardní hry, tak mapy speciální, které bude třeba načítat. Podíváte-li se do této složky, naleznete zde soubory, které se jmenují *„level_1.map“* až *„level_12.map“*. Budete-li si chtít vytvořit mapu, která bude 13. kolem, stačí ji pojmenovat *„level_13.map“* a je hotovo. Speciální mapy si můžete pojmenovávat libovolně, ale délka jejich celého jména (to zahrnuje jak cestu, tak příponu) nesmí přesáhnout 200 znaků.

A jak tedy mapu vytvořit? Nejprve si otevřete libovolný textový editor, který dovede ukládat soubory ve formátu „plain-text“ a pokud možno má funkci zalamování řádky (tedy program *Notepad*, neboli *Poznámkový blok* je naprosto vyhovující). Do souboru napište do jedné řádky startovní pozici mapy, přičemž popis začněte v levém horním rohu mapy a pokračujte doleva až k okraji, po jeho dosažení pokračujte na další řádce mapy. Takto vepsanou mapu uložte s příponou *„*.map“*. Tím je připravena k užívání. Pro lepší orientaci je vhodné zapnout funkci zalamování textu a šířku okna nastavit na 24 znaků, tím se text bude zalamovat podle tvaru mapy.

Nyní již schází pouze popsat, jak se jednotlivá políčka mapy označují:

- P (*Pampuch*) zde stojí Pampuch (je vždy na začátku otočen doprava).
- R (*Robot*) zde stojí robot (a stojí na bodu).
- D (*Dot*) zde je bod.
- V (*Void*) zde není nic.
- 1…9 Zde je stěna s odpovídajícím číslem (např. *3* odpovídá stěně *Wall_3.bmp*). Je však třeba zajistit, aby použitá stěna (***x***) měla svou grafickou reprezentaci (*„Wall_**x**.bmp“*).

VII. Co nebylo doděláno, aneb vize příští verze

Když jsem nechal malou skupinu lidí otestovat tento program, abych si mohl být jist, že v něm nejsou chyby, stěžovali si mi někteří, že je hra příliš rychlá, a tudíž obtížná, a tak mě napadlo, že by bylo vhodné přidat do submenu *Nastavení* položku *Tempo*, kterou by se daly nastavit řekněme tři stupně rychlosti hry.

Dále by se dala do programu dodělat speciální kola (např. každé páté), v nichž by bylo jak robotům, tak Pampuchovi umožněno prokusovat se stěnou.

VIII. Použitá literatura

Svoboda, Luděk a kol.: 1001 tipů a triků pro Delphi, Computer Press, Praha 2002