DSL manuál

Ing. Jan Hranáč

27. října 2010

V této kapitole je stručný průvodce k tvorbě v systému *DrdSim* a (v současné době krátký) seznam vestavěných funkcí systému.

1 Vytvoření nového dobrodružství

Nejprve je nutno v hlavním adresáři hry založit novou složku dobrodružství. Její název musí začínat písmeny `"DRD_"`. Pokud první čtyři písmena názvu budou jiná, systém složku vůbec nerozpozná, nenačte a v menu nenabídne.

Následně je nutno v tomto novém adresáři vytvořit hlavičkový soubor. Ten se musí jmenovat `"description.cfg"`. Jeho syntaxe je velice jednoduchá:

```
<název_atributu> = <řetězec> | <číslo>
```

V současné době je možno nastavit následující atributy:

TITLE	řetězec	název dobrodružství
DESCRIPTION	řetězec	stručný popis
IMPORTPARTY	číslo (1/0)	je možno importovat družinu?
NEWPARTY	číslo (1/0)	je možno vytvořit novou družinu?
STARTPARTYSIZE	číslo	maximální povolená velikost vytvořené družiny
ENTRYMAP	řetězec	název souboru startovní mapy
ENTRYX	číslo	x-ová pozice na startovní mapě
ENTRYY	číslo	y-ová pozice na startovní mapě

Jak je vidět, některé z uvedených hodnot jsou typu řetězec. Ten je možno zapsat přímo na místo, kde je použit. Pokud se však jedná o text, který bude čten hráčem, je lepší místo toho použít odkaz na lokalizační stringtable. To se provede zápisem dolaru a názvem klíče lokalizovaného řetězce.

Všechny lokalizační řetězce musí být uvedeny v souboru `"stringtable.txt"`. Ten má podobnou syntax jako hlavičkový soubor (tedy název, rovnítko, hodnota). Jeden takový soubor je vhodné vytvořit v adresáři dobrodružství, další je již obsažen v hlavním adresáři hry.

Následně je třeba založit tři podadresáře: `maps`, `classes`, `scripts` a `audio`. Stojí za povšimnutí, že složky stejných názvů jsou i v hlavním adresáři hry.

Dalším krokem je vytvoření map, které pak budou uloženy v adresáři `maps`. Editor map je přiložen v adresáři `builder` a spuští se příkazem `"java -jar mapBuild.jar"`. Nápověda k této aplikaci je obsažena přímo v aplikaci samé. Do map je možno vkládat objekty definované v adresáři `classes` (v hlavní složce hry, nebo ve složce kampaně).

Dále je nutno (/možno) nadefinovat vlastní typy herních objektů a uložit je ve složce `classes` a vlastní funkce a uložit je ve složce `scripts`.

Při definici tříd je třeba pamatovat na fakt, že soubory se načítají v abecedním pořádku. Proto, pokud by třída z jednoho souboru byla vyděděna ze třídy v jiném souboru, měl by tento soubor být pojmenován tak, aby byl načten jako první.

Pokud programátor naprogramuje funkci Init, bude tato funkce spuštěna na začátku dobrodružství.

Složka `audio` může obsahovat vlastní hudbu a zvuky (např. nadabované dialogy).

2 Programování v DSL

Soubory jazyka DSL obsahují pouze definice tříd a definice funkcí. Žádné jiné konstrukce nejsou zapotřebí.

Definice třídy: Definice třídy je uvedena hlavičkou, která se sestává z názvu třídy a případně z dvojtečky a názvu mateřské třídy. Tělo třídy je uzavřeno mezi složenými závorkami a obsahuje seznam atributů a jejich hodnot (název, rovnítko, hodnota). Hodnotou může být řetězec, odkaz do stringtable nebo číslo. Navíc je možno jako atribut uvést i další složené závorky a procedurální kód z druhé části DSL (viz dále).

Formální shrnutí:

```
<id třídy> [: <id mateřské třídy>]
"{"
   {<id atributu> = <hodnota atributu> ;}
"}"
```

Příklad:

```
/// Base class for triggers.
Trigger: MapObject
{
  handler = {StdPrintLn("Base trigger triggerred.");}
}

/** Base class for NPCs - encounter won't implicitly
 * start a fight but it can still go that way. */
NPC: Trigger
{
  npc = 1;
  handler = {StdPrintLn($GOODDAY);}
}
```

Pozn.: Více příkladů je možno nalézt ve složce classes v hlavním adresáři hry.

Definice funkce: Definice funkce je uvozena názvem funkce, kulatými závorkami a v nich případným seznamem názvů parametrů. Poté následují složené závorky a v nich tělo funkce.

Zde je popis syntaxe definice funkce:

```
<id funkce> "(" [ <id atributu> { , <id atributu> } ] ")"
<blok kódu>

<blok kódu> ::= "{"
                   { <příkaz> }
                "}"

<příkaz> ::= <přiřazení> | <volání funkce> | <return příkaz> |
             <if konstrukce> | <while cyklus>

<přiřazení> ::= <l-hodnota> = <výraz> ;
<l-hodnota> ::= [global] <id proměnné> { "[" <výraz> "]" }

<if konstrukce> ::= if "(" <výraz> ")" <blok kódu>
                { elif "(" <výraz> ")" <blok kódu> }
                [ else <blok kódu> ]

<while cyklus> ::= while "(" <výraz> ")" <blok kódu>
```

Příklad:

```
MojeFunkce(parametr1,parametr2)
{
  // tělo funkce
}
```

Tělo funkce se skládá z příkazů, složených, nebo jednoduchých. Jednoduché příkazy jsou ukončeny středníkem.

Jednoduchý příkaz může být přiřazení, volání funkce, nebo návratového příkazu. Přiřazení je provedeno napsáním názvu proměnné, rovnítka a výrazu určujícím novou hodnotu. Před název proměnné je možno připsat klíčové slovo **global**. Volání funkce je provedeno napsáním názvu funkce, kulatých závorek a v nich seznamu výrazů určujících argumenty volání.) Návrat z funkce je proveden klíčovým slovem **return** a případně hodnotou, která se má vrátit. Výraz je složený z názvů proměnných (tentokrát bez **global**), volání funkcí, čísel, řetězců a aritmetických operátorů.

Příklad:

```
global prom1 = 2*3;
prom2 = "ahoj";
MojeFunkce(prom1+1,prom2+"svete");
```

Ze složených příkazů je možno použít **if** a **while** způsobem uvedeným v příkladu:

```
n = ZiskejPocet();
if (n==7) {
  // ...
}
elif (n>3) {
  // ...
}
else {
  // ...
}

i = 0;
while (i<n) {
  // ...
  i = i+1;
}
```

Práce s poli je maximálně jednoduchá. Nejprve je nutno pole vytvořit. To se provede stejným způsobem jako přiřazení, ale napravo od rovnítka je nutno napsat hranaté závorky.

Poté je možno přiřazovat dovnitř pole. To je provedeno napsáním názvu pole, hranatých závorek s indexem a poté rovnítkem a výrazem. Index pole výraz, který má hodnotu buď celého čísla, nebo řetězce. Uvnitř pole je možno vytvářet další pole a zapisovat do nich přidáním dalších hranatých závorek. Stejným způsobem je možno z pole číst.

Příklad:

```
pole = [];
pole[2] = [];
pole[2]["jmeno"] = "Tonda";
pole[2]["prijmeni"] = "Ucho";
StdPrintLn(pole[2]["jmeno"]+" "+pole[2]["prijmeni"]);
```

Vestavěné funkce Systém má v sobě vestavěno omezené množství funkcí, které umožňují ovládat hru. Krom toho je možno v samotném jazyce DSL nadefinovat pomocné funkce a uložit je v adresáři `scripts` v hlavním adresáři hry. Zde je jejich seznam:

int Rand6()	Vrátí číslo od jedné do šesti (včetně).
int Rand10()	Vrátí číslo od jedné do deseti.
void StdPrintLn(string)	V Linuxu vypíše na konzoli ladící hlášku. (ve Windows uloží do log souboru)
string Nmr2Str(int)	Převede číslo na řetězec.
void PlayMusic(string)	Začne hrát hudbu ze zadaného souboru.
void StopMusic()	Vypne hudbu.
void PlaySound(string)	Přehraje zvuk ze zadaného souboru.
int IsGlobalDefined(string)	Vrátí 1, pokud globální proměnná daného jména existuje, jinak vrátí 0.
void UndefineGlobal(string)	Smaže globální proměnnou daného jména.
int GetType(cokoliv)	Zjistí typ dané proměnné (0-int, 1-float, 2-string, 3-field).
void ShowDialog(string,pole,string)	text otázky, pole odpovědí, název handleru prvek pole: str|[str]|[str,int]
void ShowImageDialog(str,str,pole,str)	jako ShowDialog, ale první parametr je název obrázku
void HideDialog()	uzavře zobrazený dialog
void PrintNmr(int)	Na konzoli vypíše číslo.