Jazyk VRML 97 v praxi – 3. část

Dnes se přesvědčíme o tom, že není třeba mít obavy z modelování umělých světů. Při troše fantazie můžeme vytváření umělých těles a světů přirovnat k sestavování dílků stavebnice LEGO.

Postavím si domeček, v tom domečku…

Každý tvůrce prostorových objektů totiž postupuje přibližně stejným způsobem a složité výrobky sestavuje z jednodušších částí. Následující postup je stručným návodem, který bychom mohli nalézt v pomyslné stavebnici "*Malý stavitel VRML*". Za jednotlivými pokyny jsou v závorkách uvedeny potřebné uzly VRML:

Vymodeluj hrubý tvar prostorového objektu a stanov jeho velikost (tělesa, transformační uzly).

Vylepši povrch tělesa (barva, textury).

Nejde-li použít základní těleso, udělej si vlastní (plochy, čáry, body).

Přidej efekty pro zvýšení prostorového dojmu (světla, zvuky).

Rozmísti tělesa do prostoru (skupinové uzly).

Dříve než se podíváme na jednotlivé kroky podrobněji, zdůrazňuji jednu základní informaci, která je pro stavbu virtuálních světů velmi důležitá:

V roli parametrů se mohou vyskytovat jiné uzly.

V tomto jednoduchém oznámení se skrývá mocný nástroj pro tzv. *hierarchickou stavbu*. Uzly nemusejí být kladeny do virtuálního prostředí pouze jeden za druhým, ale lze je sdružovat tak, aby vznikaly několikapatrové struktury, ne nepodobné stromům popisujícím například rodokmen. Tyto struktury se skutečně nazývají *stromy* (*trees*) a podle vzájemné polohy uzlů ve stromu mluvíme o vztazích *rodič*, *potomek*, *sourozenec*. A podobně jako v rodině rodič předává potomkům (tj. uzlům zapsaným do jeho parametrů) určité vlastnosti, ve virtuálním světě to může být poloha, schopnost stejné reakce na vnější podněty apod.

Základní tělesa a transformace

Základní tělesa jsou pouze čtyři – koule, kvádr, kužel a válec. Všechna jsou iniciálně umístěna tak, aby měla těžiště (u kužele však střed osy) v počátku souřadné soustavy.

```
Sphere { radius 1 # poloměr
}
Box { size 2 2 2          # délky stran
}
Cone { bottomRadius 1     # poloměr podstavy
  height 2                # výška
}
Cylinder { radius 1       # poloměr
  height 2                # výška
}
# Rozměrové parametry základních těles
```

Každé z těles má několik parametrů, jejichž iniciální hodnoty jsou uvedeny ve výpisu. Tělesa jsou v paměti počítače vždy převedena do množiny povrchových plošek. Jejich přesný počet známe pouze u kvádru – má jich právě šest. U ostatních těles nelze stanovit počet, a tedy přesnost, s jakou je oblý povrch tělesa nahrazen soustavou plošek. To však není žádnou chybou, ba právě naopak – prohlížeč může měnit kvalitu ploškové náhrady v závislosti na aktuálním výkonu počítače. Menší počet ploch snižuje výpočetní nároky a uživatel se rád spokojí s “hranatou koulí” za cenu vyšší rychlosti při práci ve virtuálním světě.

Abychom mohli do virtuálního světa vložit těleso a začít si je prohlížet, musíme se naučit pracovat se dvěma uzly, jejichž úkolem je vytvářet stromové struktury a shrnovat jednodušší objekty a jejich vlastnosti do celku.V následujícím přehledu jsou uvedeny včetně iniciálních hodnot parametrů. Symbolem hranatých závorek jsou označeny seznamy více hodnot.

```
Transform {
  scale 1 1 1             # měřítko
  rotation 0 0 1 0  # osa a úhel natočení
  translation 0 0 0       # posunutí
  children []             # seznam potomků
}
Shape {
  geometry []             # tvar povrchu
```

```
  appearance []    # vzhled povrchu
}
# Základní uzly stromové hierarchie
```

Transform je rodičovským uzlem, který zajišťuje pro své potomky (zapsané do parametru *children*) společné nastavení měřítka (*scale*), natočení (*rotation*) a posunutí (*translation*).

Další příklad je typickou ukázkou základního tvaru stromu, definujícího polohu tělesa. Rodičovský uzel *Transform* má jediného potomka – uzel *Shape*. Ten má opět jediného potomka, který určuje geometrii tělesa, v tomto případě kvádru. Parametr *appearance* je zatím nevyužit, a proto nebude vzhled kvádru příliš zajímavý – těleso bude matné a bílé, což odpovídá iniciálnímu nastavení barev objektů.

V ukázkovém programu jsou uvedeny dva způsoby zadání kvádru o délce hran 6, 2 a 4 metry. V prvním případě je standardní kvádr s iniciálními délkami hran 2 zvětšen v rodičovském uzlu nastavením parametru *scale*. Ve druhém případě jsou rozměry zapsány přímo do uzlu *Box*. Formální pojmenování virtuálních objektů názvy KVADR1 a KVADR2 je nepovinné a je uvedeno jen pro ilustraci. Druhý způsob zápisu kvádru je mnohem vhodnější, protože umožňuje pozdější dynamické změny měřítka a polohy při zachování neměnného poměru stran kvádru. Jak uvidíme později, všechny výše uvedené parametry uzlu *Transform* lze animovat, což není možné u pevného parametru kvádru *size*.

```
#VRML V2.0 utf8              # Dva způsoby zadání kvádru
DEF KVADR1 Transform {
  scale 3 1 2
  children Shape { geometry Box {}
            }
}
DEF KVADR2 Transform {
  children Shape { geometry Box {size 6 2 4}
            }
}
```

Transformace jsou v rámci jednoho uzlu prováděny vždy v následujícím pořadí:

1. úprava velikosti (*scale*);

2. natočení (*rotation*);

3. posunutí (*translation*).

Připomeňme si, že na pořadí zde záleží. Kdybychom nejprve těleso posunuli a v jeho nové poloze otočili kolem středu souřadné soustavy, s největší pravděpodobností by se ocitlo na zcela jiném místě, než kdybychom je nejprve otočili a poté posunuli. Chceme-li pevně dané pořadí transformací změnit, musíme umístit více uzlů *Transform* do stromu nad sebe. Transformace se vždy skládají od dětí k rodičům.

Natočení a posunutí jsou nedeformující operace. Pomocí změny měřítka pak dokážeme tělesa nejen zvětšovat a zmenšovat, ale i deformovat. Z koule tak může snadno vzniknout prostorový elipsoid. Platí však, že koeficienty měřítka musejí být kladná čísla. Tím se možnosti uzlu *Transform* liší od obecných transformací, při nichž lze například docílit tzv. *zrcadlení* (symetrie) zadáním koeficientů měřítek o hodnotě minus jedna.

Za povrch krásnější...

V předchozí části jsme uvedli, že uzel *Shape* má kromě parametru *geometry* ještě druhý parametr – *appearance*. Ten jsme dosud nevyužili, i když právě on má zásadní vliv na vzhled tělesa. Do tohoto parametru se umísťuje jediný možný uzel, který se příhodně nazývá *Appearance*. Dovoluje definovat dva možné vzhledy povrchu těles:

barvu (parametr *material*);

texturu (parametry *texture* a *textureTransform*).

Parametr *material* je určen k zadání barvy, která bude stejná na celém povrchu. Parametr *texture* pak dovoluje pokrýt (polepit) povrch libovolným obrázkem, čímž lze tělesu dodat velmi přirozený vzhled tak, aby připomínalo výrobek ze dřeva, kamene apod. Do parametrů uzlu *Appearance* se zadávají opět uzly:

do parametru *material* lze přiřadit pouze uzel *Material*;

do parametru *texture* lze přiřadit jeden z uzlů *ImageTexture*, *PixelTexture* nebo *MovieTexture*;

do parametru *textureTransform* se přiřazuje uzel *TextureTransform*.

Jméno uzlu *Material* je trochu zavádějící. Normálně bychom si pod ním mohli představit takové pojmy, jako jsou například železo, dřevo, papír apod. Ve skutečnosti tento uzel označuje pouze barevné charakteristiky povrchu. Nutno přiznat, že jich není málo a že pro jejich plné pochopení je třeba proniknout hlouběji do problematiky barev a světel v trojrozměrném prostoru. Barevný vjem je totiž přímo ovlivňován světlem. Následující tabulka poskytuje sice úplný, ale vědomě zjednodušený popis parametrů uzlu *Material*. K jejich hlubšímu studiu doporučuji odbornou literaturu, uvedenou na konci tohoto dílu.

Iniciální hodnoty parametrů uzlu Material

diffuseColor 0.8 0.8 0.8

základní barva povrchu ve složkách RGB

```
ambientIntensity 0.2
  # jak je povrch zesvětlen jasem prostoru
specularColor 0 0 0
  # jak je odrážena barva dopadajícího světla
shininess 0.2
  # ostrost odrazu pro předchozí parametr
emissiveColor 0 0 0
  # fluorescenční (svítivá) barva povrchu
transparency 0
  # průhlednost (při hodnotě 1 těleso zmizí)
```

Tomu, jak do scény zavést různé zdroje světla, se budeme věnovat později. Připomeňme, že ačkoliv nastavením parametru *emissiveColor* vytvoříme dojem, že těleso září, jeho svit nijak neovlivní osvětlení ostatních těles.

Pro většinu jednoduchých těles vystačíme s nastavením jediného parametru – *diffuseColor*. Ten definuje matný vzhled. Odstín barvy se mění podle úhlu dopadajícího světla. Přímo proti světlu je barva nejjasnější. Typický zápis barevného tělesa může vypadat:

```
#VRML V2.0 utf8 # Matný modrý válec
Transform
{ children
    Shape          {
          appearance Appearance {
          material Material {
              diffuseColor 0.5 0.5 1 # modrá
   } }
        geometry Cylinder {}
    } }
```

Textury aneb Za málo peněz hodně muziky

I když použití barvy přináší výrazné zlepšení vzhledu těles, nejvěrnějšího vzhledu povrchu dosáhneme teprve využitím textur. Textura je obrazový vzorek, který je nanášen na povrch tělesa. Tomuto procesu se říká *mapování textury*. Je zřejmé, že mapování je doprovázeno rozličnými deformacemi obrazu, protože textura je definována jako dvojrozměrný (obdélníkový) obrázek, kdežto povrch těles je obecně zakřivený. Přesto se dá pomocí textur docílit velmi pěkných efektů – obyčejný válec se zdá být vyřezán ze dřeva, jednoduchý kvádr vypadá jako mramorová deska.

Pro výběr barevného vzoru lze použít několika možností; každá z nich je svázána se samostatným uzlem:

obrázek uložený v samostatném souboru (uzel *ImageTexture*);

opakující se kombinace barev zapsaná v uzlu *PixelTexture;*

videosekvence přehrávaná ze souboru (*MovieTexture*).

Výše uvedené tři druhy uzlů popisují zdroj dat, ze kterého budou získávána obrazová data použitá jako textura. Samostatný uzel *TextureTransform* je pak určen k tomu, aby definoval způsob mapování textury na povrch, tj. posunutí, natočení a změnu měřítka textury. Uzel má parametry podobné parametrům uzlu *Transform*. Obsahují však jen dvojrozměrná data, protože textura je rovinná.

```
#VRML V2.0 utf8 # Mapování obrazové textury
Transform
{ children
  Shape {
        geometry Box {}
        appearance Appearance {
        texture ImageTexture
  {url ”obrazek.gif”}
}
} }
```

Způsob nanesení textury je svázán s cílovým tělesem. Na kouli je textura nanesena pouze jednou tak, že ji “omotá” kolem rovníku, na pólech dojde k výraznému zkreslení obrazu – jeho okraje se zhroutí do jediného bodu. Naopak na kvádr je textura mapována bez výraznějšího zkreslení, celkem v šesti kopiích – jedna na každou stěnu. U válce je obraz textury nanesen jednou na plášť a dále na obě podstavy tak, že podstavy tvoří kruh vepsaný čtverci textury. Podobně je tomu i u kuželu, u jeho vrcholu však dochází ke zkreslení.

Každý ze tří různých uzlů definujících texturu má své specifické parametry. Společné všem jsou právě dva – *repeatS* a *repeatT*. Určují, zda textura smí být opakovaně nanášena ve vodorovném, resp.

svislém směru. Nechceme-li, aby se textura opakovaně nanášela na celý povrch, nastavíme tyto parametry opakování na hodnotu *FALSE*. Otázka, jak obarvit povrch tělesa mimo texturu, je řešena jednoduše – je zopakována barva okraje textury. Tím vzniká zajímavý, i když ne vždy žádoucí efekt vzniku pruhů po celém povrchu, kombinovaných s jednobarevnými čtverci, jejichž barva je shodná s barvou rohového bodu textury. Tohoto jevu lze využít v pozitivním smyslu, když obrázek textury předem upravíme v nějakém editoru tak, aby na jeho okraji vznikl tenký jednobarevný rámeček. Barva rámečku pak vyplní veškeré okolí textury na povrchu tělesa.

Podivný parametr url

Tento parametr je zkratkou používanou v internetu a označující adresu souboru či místa přístupného po síti (URL – *Uniform Resource Locator*). Je prvním dokladem toho, že VRML dovoluje využívat pro tvorbu virtuálních světů prostředky dostupné po síti. Uzly *ImageTexture* a *MovieTexture* mají v tomto parametru zapsány údaje o umístění a jménu obrázku, resp. animace.

Sympatickou vlastností parametru *url* je možnost zápisu několika textových řetězců, které budou využity pro vyhledávání konkrétního souboru v případě, že předchozí adresy nejsou dostupné. Tento přístup ocení zejména zkušení uživatelé internetu, kterým se nejednou stalo, že je hypertextový odkaz zavedl na WWW stránku, která byla v dané chvíli nedostupná, ať již kvůli přetížení sítě, změně adresy cílového počítače, nebo jeho vypnutí.

Vytváříme-li například virtuální obrazovou galerii, nemusíme mít na svém počítači umístěny všechny obrazy, protože se na ně můžeme odkázat na jiné místo v internetu. To mimochodem sníží zátěž našeho počítače, protože značné množství dat bude k uživatelům proudit z jiných serverů. Samozřejmě je vhodné mít připraveny na počítači náhradní obrázky (v horší barevné kvalitě a nižším rozlišení) pro případ selhání sítě. Obsah parametru *url* v uzlu *ImageTexture* pak může vypadat takto:

```
ImageTexture { url [
  ”http://www.louvre.fr/da-vinci/mona-lisa.jpg”,
  ”http://www.archive.fr/images/mona-lisa.png”,
  ”moje-obrazky/nahradni.gif”
]}
```

Uvedený příklad zároveň ukazuje, jaké typy obrázků mohou být použity pro textury. Formát JPEG (JPG) je vhodný pro kódování plnobarevných fotografií, vyznačuje se však tím, že zanedbává některé detaily a barevné přechody. Pro virtuální realitu je ovšem tento princip ztrátové komprese zcela přijatelný. Dalším formátem je GIF, který uchovává obrazy o maximálně 256 barvách. Méně známým formátem je PNG (čti “ping”). Patří mezi nejmladší obrazové formáty a očekává se, že postupně nahradí GIF. Dokáže totiž efektivně kódovat plnobarevné obrazy včetně informací o průhlednosti, a to bezeztrátově.

Jako pohyblivou texturu (*MovieTexture*) lze použít soubor typu MPEG-1. Některé prohlížeče

akceptují i tzv. *animovaný GIF*, tedy jeden soubor GIF obsahující posloupnost několika obrázků. Na rozdíl od statického obrázku má uzel *MovieTexture* další parametry, týkající se dynamiky prohlížení, nebo spíše přehrávání sekvence pohyblivých obrazů. Logický parametr *loop* povoluje spouštění přehrávání v nekonečné smyčce. Parametrem *speed* se nastavuje rychlost přehrávání. Je-li větší než 1, videosekvence se přehrává rychleji než původní rychlostí, hodnota menší než 1 znamená zpomalení. Zajímavým trikem je nastavení rychlosti na zápornou hodnotu, jež způsobí přehrávání pozpátku. Nulová hodnota způsobí zobrazení jen prvního snímku z videosekvence. V případě použití animovaného GIF formátu však není zaručeno korektní časování.

Jako chudá příbuzná vypadá textura zvaná *PixelTexture*, přesto je velmi šikovná pro úsporné definování jednoduchých, opakujících se vzorků na povrchu těles. Neodkazuje se na žádný vnější zdroj obrazových dat, ale definuje vzorek přímým vypsáním barev do vlastního strukturovaného parametru *image.*

Průhlednost aneb Děravá tělesa snadno a rychle

V předchozí části jsme zjistili, že textura může obsahovat informaci o průhlednosti. K čemu je taková věc dobrá? Cožpak může být těleso v některých místech průhledné a jinde nikoliv? I když se takové situace nevyskytují často, použití textury s průhledností je výhodné pro různé zajímavé efekty. Zcela průhledné části povrchu zmizí a je vidět skrz těleso na vzdálenější objekty. Zadní stěny tělesa přitom nepřekážejí ve výhledu, protože prohlížeče se (není-li stanoveno jinak) zadními stěnami nezabývají. Barva částečně průhledných částí textury je míchána s barvou zadních objektů, čímž lze vytvořit iluzi barevného skla, mraků nebo sloupce kouře.

Materiál, textura, nebo obojí?

Uzel *Appearance* dovoluje zadat jak materiál povrchu objektu, tak texturu, která jej pokrývá. Pokud to není nutné, doporučuje se nekombinovat obě informace dohromady, protože zobrazení takového tělesa je náročné. Parametr *material* přináší nutnost vypočítávat přesné osvětlení povrchu (odraz a rozptyl světla), parametr *texture* navíc přidává výpočty, které zajišťují mapování textury. Pro většinu případů je proto rozumné držet se dohody, že při použití textury je parametr *material* ponechán prázdný.

V některých případech ovšem kombinace obou parametrů může naopak zvýšit efektivitu. Je to tehdy, pokud je textura definována ve stupních šedi. Jednotlivé odstíny lze snadno modifikovat (obarvit) materiálem a s pomocí jediného obrázku v odstínech šedi nebo černobílého obrázku vytvořit iluzi objektů vyrobených z odlišných materiálů.

V příštím pokračování se seznámíme s tím, jak efektivně zapisovat opakující se stejné části virtuálního světa, a co dělat, když potřebujeme vymodelovat složitější objekt než jen kouli, válec nebo kvádr.

Jiří Žára

Literatura:

J. D. Foley, A. van Dam, S. Feiner, J. F. Hughes: Computer Graphics – Principles and Practice, Addison Wesley, 1990.

J. Žára, B. Beneš, P. Felkel: Moderní počítačová grafika, Computer Press, 1998, ISBN: 80-7226-049-9.

J. Žára: Laskavý průvodce virtuálními světy, Computer Press, 1999.