Skrytý výkon GPU!

Grafické akcelerátory nemusí být pouze na hraní

MICHAL HUŠÁK

Grafické akcelerátory se začaly na platformě PC hojně využívat ve chvíli, kdy přišli moudří výrobci počítačových 3D her. Především zásluhou hráčů se totiž mohly firmy zabývat vývojem výkonnějších grafických akcelerátorů, tedy pokud opomeneme silně profesionální segment karet. Nejdříve bodovaly v domácím prostředí produkty společnosti 3Dfx. Jejich Voodoo a Voodoo II se staly mezi mnoha hráči doslova legendou. Ovšem nic netrvá věčně a ani akcelerace typu Glide nebyla výjimkou. Společnost nVidia doslova převálcovala trend nestíhající 3Dfx s hardwarovou akcelerací, která byla založena na rozhraní OpenGL a později i na Direct3D. Časem nebylo mnoho společností, které by mohly zběsilému vývoji v oblasti akcelerátorů konkurovat. Mezi dnešními společnostmi stojí pouze dva skuteční soupeři – nVidia s kartami typu GeForce a ATi s řadou Radeon. Výrobci grafických karet buď využívají GPU čipů GeForce nebo Radeon, či popřípadě jiných méně známých vývojářských firem, ovšem ty integrují spíše do nevýkonných počítačů (např. jako integrované VGA přímo na motherboard) – určených kupříkladu pro kancelářské potřeby a podobně.

Poslední generace grafických karet

Během vývoje grafických karet došlo k mnoha změnám v rámci celé architektury. Velkou změnu také prodělala i extrémně výkonná GPU. Zajímavé přitom je, že poslední modely grafických karet se liší od dřívějších generací poměrně unikátním způsobem. Konkrétně se toto tvrzení týká modelových řad GeForce FX a ATi Radeon. Tyto odlišnosti spočívají v tom, že je možné akcelerátory (respektive GPU) výrazně programovat. Je sice pravdou, že tato skutečnost není ničím převratným a těchto dovedností se využívá u moderních graficky náročných her téměř na 100 %. Nicméně, pokud se zaměříme na jiné než herní využití výkonného GPU, začíná být situace velmi zajímavá. Proto nyní zapomeňme, že jsou akcelerátory primárně určeny pro náročné hráče, a zaměřme se na trochu jiný náhled.

Možnosti při programování GPU

Nejdříve si shrneme, jaké možnosti moderní grafické karty nabízejí, respektive co vše lze na čipech GPU měnit. V principu je možné pro GPU psát dva základní typy programů. Prvním typem jsou tzv. Vertex Shadery – programy pro zpracování koncových bodů polygonů. Tyto programy umožňují rozšířit standardní funkce pro výpočet polohy, barvy a osvětlení objektů o nově definované služby. Druhým typem programů jsou tzv. Pixel Shadery – ty slouží pro matematické zpracování jednotlivých pixelů. Pixel Shadery umožňují rozšířit standardní grafické funkce pro interpretaci textur a výpočet barvy každého jednotlivého obrazového bodu.

Principy využité pro zvýšení rychlosti

Hardware, který vykonává příkazy Vertex a Pixel Shader programů, musí pracovat extrémně rychle. Zvláště náročné je provádět příkazy Pixel Shaderů – počet pixelů (obrazových bodů), které je třeba zpracovat, je totiž několikanásobně vyšší než počet Vertexů (geometrických vrcholů). Grafický hardware pak musí používat některé nestandardní techniky, které běžná CPU většinou nepoužívají.

První technikou je masivní vektorové zpracování vstupních dat. Většinu typických grafických informací můžeme popsat jako datové struktury o třech a čtyřech prvcích. Polohu v prostoru vyjadřují tři souřadnice – X, Y, Z, barvu pro změnu tři složky a alfa kanál (průhlednost), celkově se barva bodu dá označit jako R, G, B, A. Místo toho, aby GPU v jednom cyklu prováděl operace pouze s jednou složkou vstupních dat, provede GPU zpracování všech složek najednou. Dojde tak až k čtyřnásobnému urychlení ve srovnání s CPU.

Další techniku, kterou GPU využívá, je hardwarová implementace poměrně složitých matematických funkcí. Jednou s typických operací v grafice je např. násobení matic. GPU tuto operaci pro matice typu 4 × 4 dokáže provést jako jednu instrukci, CPU by potřebovalo instrukcí několik desítek.

Pokud ani výše uvedené triky nestačí, je možné navíc použít i „hrubou sílu". Místo jednoho grafického procesoru se jich jednoduše využije víc. Většině grafických operací simultánní zpracování více procesory najednou nevadí. Grafické karty pak mívají několik (až několik desítek) procesorů, které vykonávají příkazy Vertex a Pixel Shaderů.

Rychlosti se dosahuje i použitím speciálních videopamětí, které umožňují mnohem rychlejší přístup k datům než standardní operační paměť, použitím texturovacích jednotek jako zdrojů dat a řady dalších technik a triků, ty však již záleží na konkrétním výrobci GPU čipu.

Jazyky pro programování GPU

V oblasti prostředků pro programování GPU a psaní programů Vertex a Pixel Shaderů panuje bohužel mírný chaos. Je sice možné psát tento typ programů v assembleru dané grafické karty, ale většina programátorů dává přednost vyšším jazykům. Prvním standardem pro programování GPU byl jazyk HLSL specifikovaný společností Microsoft. Pomocí HLSL je však možné přistupovat k prostředkům grafické karty jen v rámci standardu DirectX (Direct3D), ale nikoliv pod mnohdy výhodnějším rozhraním OpenGL.

Užitečné odkazy

Jazyk Cg:
- *http://www.nvidia.com/object/cg_tool-kit.html*
- *http://www.cgshaders.org*

Výpočty pomocí GPU:
- *http://www.gpgpu.org*

Ve standardu OpenGL je možné programovat grafické karty pomocí extenzí GL_ARB_fragment_program a GL_ARB_vertex_program. Jako standard pro příští verze OpenGL byl přijat jazyk OpenGL Shading Language vyvinutý firmou 3Dlabs. Nejlepší řešení ze všech zmíněných je ale pravděpodobně jazyk Cg vyvinutý společností nVidia. Cg pracuje jak s DirectX, tak s OpenGL. Interně je překladač Cg schopný detekovat ostatní standardy a za běhu vytvořit kód, který je podporuje. Cg program je pak schopný běžet nejen na kartách od firmy nVidia, ale i na hardwaru konkurenčních výrobců. Další velkou výhodou Cg je, že se velice podobá standardnímu C++, a naučit se ho používat je poměrně snadné.

Omezení programovatelnosti GPU

Vysoká rychlost a architektura vykonávání instrukcí v GPU způsobuje řadu omezení na programy, které mohou GPU vykonávat. Prvním omezením je počet typů podporovaných příkazů, který je mnohem nižší než u standardního C++. Omezen je i přístup k proměnným. Nelze používat ukazatele (Pointry) a používání polí má také řadu specifických omezení. Na druhé straně jsou podporovány některé velice silné speciální instrukce, například již zmíněný příkaz pro násobení matic.

Dalším omezením je počet instrukcí, které může program obsahovat. To je mnohem výraznější pro Pixel Shadery než u Vertex Shaderů. I moderní grafické karty podporují Pixel Shader programy pouze do cca dvou set instrukcí.

Z hlediska možnosti využít GPU pro jiné než grafické výpočty je omezením i to, že výpočty probíhají masivně paralelním způsobem. Pomocí GPU se nechají efektivně řešit pouze problémy, které je možné rozdělit na řadově tisíce malých identických výpočtů zpracovávaných v jednom cyklu.

Posledním lehce absurdním omezením je přístup k výsledkům výpočtu. Grafické karty byly primárně navrženy pro grafický výstup a nepředpokládalo se, že by někdo chtěl použít výsledky práce GPU k něčemu jinému. Výsledky výpočtů je možné ukládat pouze do obrazových bufferů. Dochází tak často ke ztrátě informací, protože čísla v pohyblivé čárce se redukují na integer v rozsahu 0–255. Odstranění tohoto problému v příští generaci grafických karet je základním předpokladem pro seriózní GPU výpočty.

Použití GPU pro vědecké výpočty

Existuje řada praktických vědecko-technických problémů, které se pro výpočty pomocí grafických karet samy nabízejí. Jedním z výpočetně velice náročných problémů je řešení parciálních diferenciálních rovnic. Tyto rovnice řeší například tok tepla v obytném domě nebo pnutí v konstrukcích. Zkoumaný objekt, například část zdiva, skrze které protéká teplo, je nutné matematicky rozdělit na malé části a modelovat, jak si vyměňuje teplo s dalšími částmi. Z hlediska programování není problém tyto malé části popsat jako pixely a pomocí Pixel Shaderů modelovat výměnu tepla s jejich okolím.

Dalším typem modelovacích výpočtů jsou popisy systémů pohybujících se částic. Jednotlivé částice mohou být například molekuly kapaliny, nebo i atomy nového farmaceutického preparátu, který je zkoumán. Na základě výpočtů je pak možné předpovědět vlastnosti nově připravovaného materiálu. Celkově Vertex Shadery umožňují počítat změnu pohybu částic mnohem rychleji než klasický procesor CPU.

Srovnání výpočetního výkonu GPU a CPU

Testováním rychlosti Vertex Shaderů pro jiné než grafické výpočty ze zabývala skupina vědců z University of Washington. Pro pokusy používali kartu GeForce4 Ti4600 a pro srovnání rychlosti výpočtů bylo zvoleno Pentium 4 pracující na frekvenci 1,5 GHz. Do strojového kódu Vertex Shaderů byla přepsána řada testovacích algoritmů pro násobení matic, pro řešení rovnic a práce s vektory. Ve všech případech bylo možné provádět výpočty pomocí GPU rychleji než s pomocí CPU. V nehorším případě (numerické řešení rovnic) byl výpočet s pomocí GPU dvojnásobně rychlejší než s pomocí CPU. V nejlepším případě (operace s vektory) proběhl výpočet pomocí GPU dokonce 12× rychleji než při využití služeb klasického procesoru CPU.

Univerzální systém pro řešení diferenciálních rovnic pomocí Pixel Shaderů naprogramoval tým pracovníků z University of Virginia. Výpočty byly prováděny pomocí karty ATI Radeon 9700 a pro srovnání byl použit CPU procesor Athlon 1600. V závislosti na rozsahu úlohy bylo řešení pomocí GPU 13× až 16× rychlejší než řešení pomocí CPU.

Má to všechno budoucnost?

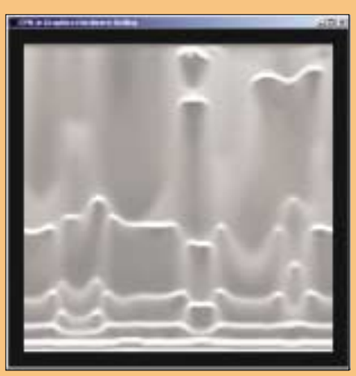
Zkusme si zkombinovat informace o výpočetním výkonu grafických karet s některými dalšími technologickými novinkami. Firma Intel začíná prosazovat náhradu sběrnice AGP sběrnicí PCI-Express. PCI-Express by mohla umožnit umístění většího počtu grafických karet do počítače, tak jako tomu bylo v dobách sběrnice PCI před érou AGP.

Před námi vyvstává poněkud znepokojivá vize. Počítač vybavený čtyřmi grafickými kartami by mohl již dnes provádět některé typy výpočtů až 60× rychleji než moderní CPU. Klasický procesor by v takovéto situaci sloužil pouze jako pomocník, který by se staral pouze o přidělování jednotlivých úloh. V řadě vědeckotechnických výpočtů je tato vize zcela reálná. Standardní aplikace pro normálního uživatele však zatím tak vysoký výpočetní výkon nepotřebují a ani se nedají pro tento způsob výpočtů vhodně modifikovat. To ale neznamená, že se aplikace s takovými požadavky nenajdou – zajímavé by mohly být například GPU varianty programů pro rozpoznávání řeči nebo lámání šifer. Oba typy úloh jsou pro masivně paralelní zpracování pomocí GPU vhodné. Je otázkou času, s čím novým vědci při výzkumech přijdou.

4 0313/BAM ❑

```
void C6E2v_particle(float4 pInitial : POSITION,
                    float  tInitial : TEXCOORD0,
                    float4 vInitial : TEXCOORD1,

                out float4 oPosition : POSITION,
                out float  pointSize : PSIZE,

            uniform float    time,
            uniform float4   acceleration,
            uniform float4x4 modelViewProj)
{
  float t = time - tInitial;
  float4 pFinal = pInitial +
                  vInitial * t +
                  0.5 * acceleration * t * t;

  oPosition = mul(modelViewProj, pFinal);

  pointSize = -8.0 * t * t +
              16.0 * t +
              0.1 * pFinal.y + 1;
  if (pointSize < 0)
    pointSize = 0;
}
```

Ukázkový program Vertex Shaderu v Cg, který modeluje pohyb částice v závislosti na čase.



Simulace varu kapaliny pomocí Pixel Shaderů.

3D simulace různých fyzikálních jevů pomocí GPU výpočtů: var, chemická reakce, difuze.

Simulace růstu krystalů ledu pomocí GPU. Výpočet běžel pomocí GPU 9× rychleji než pomocí CPU.