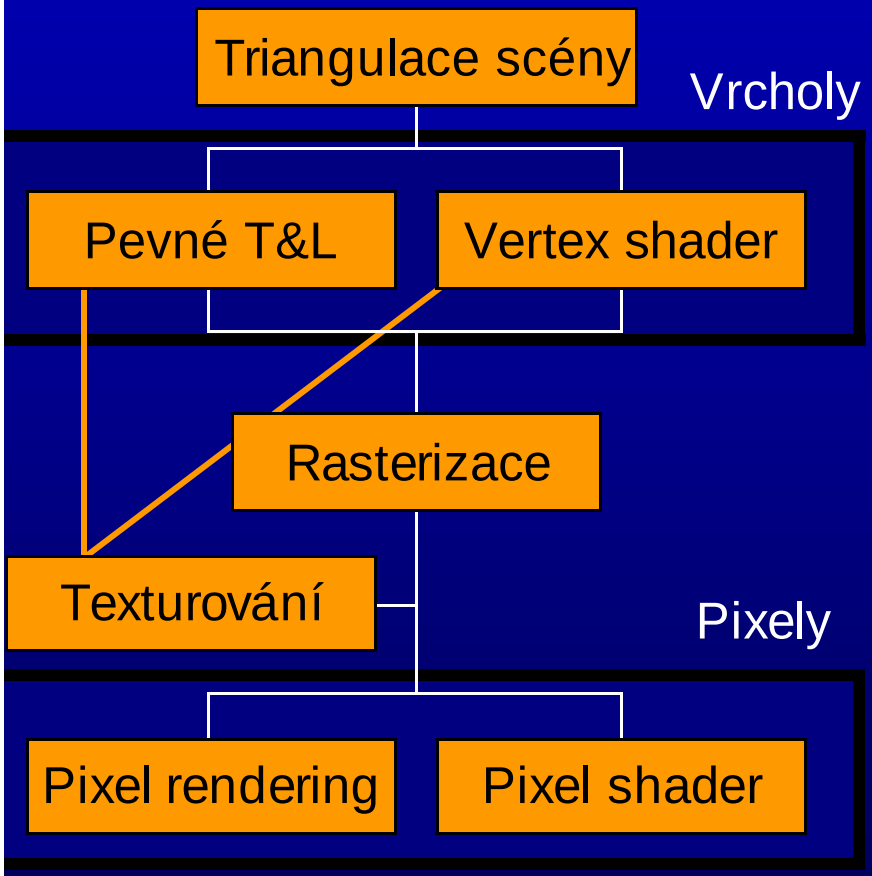
Programovatelné shadery a jazyk Cg

Petr Kmoch

Historie

- Softwarové výpoèty
- Pevná pipeline
- Volitelné moduly
- Programovatelné shadery

Grafická pipeline



- Zpracování vrcholù
  - Transformace & osvì tlení
  - Nedì lá perspektivní projekci
- Zpracování pixelù
  - Urèí výslednou barvu
  - Ète textury

Programovatelné shadery

- Asemblerovské instrukce GPU
  - Pøizpùsobené grafickým úèelùm (práce s vektory, interpolace)
- Vektorové a skalární registry
- Zadávání programù pøes API (DirectX, OpenGL)
- Více programù na 1 snímek
- Krátké (128 instrukcí VS, 8 instrukcí PS)

Programovatelné shadery, vývoj

- Zvì tšování poètu instrukcí a registrù
- Rozšiøování instrukèní sady
  - Vì tvení, cykly
  - Podprocedury
  - Vyšší matematické funkce
- Zvì tšování poètu textur

Ukázka programu shaderu

```
DP3 R0, c[11].xyzx, c[11].xyzx
RSQ R0, R0.x
MUL R0, R0.X, c[11].xyzx
MOV R1, c[3]
MUL R1, R0.x, c[0].xyzx
DP3 R2, R1.xyzx, R1.xyzx
RSQ R2, R2.x
MUL R1, R2.x, R1.xyzx
ADD R2, R0.xyzx, R1.xyzx
DP3 R3, R2.xyzx, R2.xyzx
RSQ R3, R3.x
MUL R2, R3.x, R2.xyzx
DP3 R2, R1.xyzx, R2.xyzx
MAX R2, c[3].z, R2.x
MOV R2.z, c[3].y
MOV R2.w, c[3].y
LIT R2, R2
```

Jazyk Cg

- C for Graphics, C for GPUs
- Vývoj: nVIDIA + Microsoft Corp.
- Pœdstaven v èervnu 2002
- Vysokoúrovòový programovací jazyk pro shadery
- Vychází ze syntaxe C a C++
- HW nezávislý

Pøehled vlastností Cg

- Pojmenované promì nné
- Podmínky, cykly
- Uživatelské funkce
- Typy: skaláry, vektory, matice, pole, struktury
- Knihovní funkce
- Pøetì žování funkcí

Výhody Cg oproti asembleru

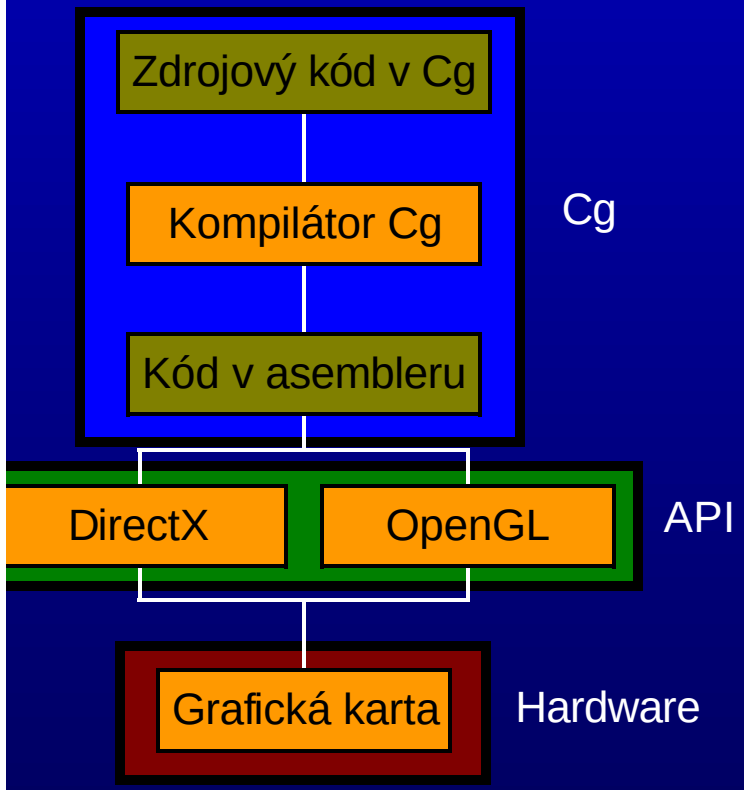
- Èitelnost
- Pœnositelnost
- Rychlejší zápis
- Usnadòuje komunikaci s aplikací
  - Pojmenované parametry

Ukázka programu v Cg

```
float3 cSpec = pow(max(0, dot(Nf, H)), phongExp).xxx;
float3 cPlastic = Cd * (cAmbi + cDiff) + Cs * cSpec;
```

```
vertout main(appin In,
        uniform float4x4 ModelViewProj : C0,
        uniform float4x4 ModelViewIT : C4,
        uniform float4 LightVec)
{
    vertout Out;
    Out.HPosition = mul(ModelViewProj, In.Position);
    float4 normal = normalize(mul(ModelViewIT,In.Normal).xyzz);
    float4 light = normalize(LightVec);
    float4 eye = float4(0.0, 0.0, 1.0, 1.0);
    float4 half = normalize(light + eye);
    float diffuse = dot(normal, light);
    float specular = dot(normal, half);
    specular = pow(specular, 32);
    float4 diffuseMaterial = float4(0.0, 0.0, 1.0, 1.0);
    float4 specularMaterial = float4(1.0, 1.0, 1.0, 1.0);
    Out.Color0 = diffuse*diffuseMaterial + specular*specularMatrial;
    return Out;
}
```

Použití Cg



- Pøekládá se do asembleru grafické karty
- Získaný kód se zavádí pøes API
  - DirectX od 8.0
  - OpenGL od 1.4
- Offline a runtime varianta

Pøeklad Cg

- Pøekladaè je open-source
  - Front-end
  - Ukázkový back-end (lidsky èitelný výstup)
- Pøekladaèe píší výrobci HW
  - Masivní HW-specifická optimalizace

Profily

- Øešení rùznorodosti grafických karet
- Profil = podmnožina Cg pro dané API
- DirectX 8.0 VS, DirectX 8.0 PS, OpenGL ARB VS, NV30 OpenGL, …
- Výbìr v okamžiku kompilace

Datové typy v Cg

- `float, half, fixed`
- `bool`
- `sampler*`
  - Handle na texturu
- Vektory a matice
  - `bool3, float4x4, ...`
- Nemá celá èísla
- Struktury; implicitní `typedef`
- Pole
  - Pøedávají se hodnotou

Funkce a pøíkazy v Cg

- Cykly `for, while`
- Operátory se aplikují po složkách
- Funkce `mul` – násobení vektorù a matic
- Konstruktor vektorù
  - `float4 y = x*float4(3.0,0.0,1.0,0.5);`
- Funkce rozlišují vstupní a výstupní parametry
  - `float fce1(out float x)`
  - `float fce2(inout float x)`

Operátor `.` na vektorech

- Vektory mají složky `x,y,z,w`, resp. `r,g,b,a`
- Lze je duplikovat, permutovat, zkracovat, prodlužovat, dosazovat
  - `float3(a,b,c).zyx == float3(c,b,a)`
  - `float2(a,b).yyxx == float4(b,b,a,a)`
  - `float4(a,b,c,d).w == d`
- Efektivní

Zajímavé knihovní funkce

- `cross(a,b)`
  - Vektorový souèin 3-složkových vektorù `a` a `b`
- `determinant(M)`
- `lerp(a,b,f)`
  - `(1-f)*a + b*f`
- `lit(ndotl,ndoth,m)`
  - Spoèítá koeficienty okolního, matného a lesklého svì tla
- `noise(x)`
  - Šumový vektor stejné velikosti jako `x`, konzistentní mezi snímky
- `refract(I,N,eta)`
  - Spoèítá zalomený paprsek

Rozhraní Cg programu

- Jedna hlavní funkce (obdoba `main`)
  - Volá se pro každý prvek (vrchol, pixel)
- Dva druhy vstupù
  - Variantní – závisí na prvku (souøadnice, …)
  - Uniformní – jednotné pro scénu (transformaèní matice, …)

Variantní vstupy

- Specifikace strukturou

```
struct myinput {
     float3 position : POSITION;
     float refractive_index : TEXCOORD3;
};
outdata myMain(myinput inData) {
     …
}
```

- Specifikace parametrem

```
outdata myMain(float3 myPosition : POSITION,
               float myIndex : TEXCOORD3) {
     …
}
```

Uniformní vstupy

- Specifikují se parametrem

```
outdata myMain(myInput inData,
               uniform float4x4 transformMatrix) {
    …
}
```

Offline použití Cg

- Zdrojový kód se pøeloží
- Kompilátor vygeneruje seznam registrù
- Kód i data se zadávají pøes API
  - DirectX, OpenGL
- Vhodné pro ruèní úpravy kódu

Funkce Cg runtime

- Kompilace za bìhu (pøi startu) aplikace
- Pøedání asembleru do API
- Výbìr shaderu
- Manipulace s parametry pro shader
- Verze funkcí pro DirectX a OpenGL

Datové struktury Cg runtime

- Program
  - Zdrojový kód
  - Jméno hlavní funkce
  - Profil
- Parametr
  - Jméno
  - Druh
  - Typ
  - Registr
- Kontext
  - Seznam programù (shaderù)

Shrnutí

- Vysokoúrovòový jazyk pro vertex shadery a pixel shadery
- HW nezávislý
- Nad úrovní DirectX a OpenGL
- HW-specifické optimalizace pøi kompilaci
- Možnost kompilace at runtime
- Runtime API

Další informace

- http://www.nvidia.com/cg
- http://www.cgshaders.org