OPENGL PERFORMANCE ANALYZER

Dalibor PERNICA, Master Degree Programme (5)
Dept. of Computer Graphics and Multimedia, FIT, BUT
E-mail: xperni10@stud.fit.vutbr.cz

Supervised by: Ing. Jan Pečiva

ABSTRACT

The paper describes implementation of a performance analyzer that uses OpenGL Utility Toolkit library (only time meassuring approach is discussed). The experimental benchmarks for interesting rendering operations while testing this analyzator are presented.

1 ÚVOD

Výrobci přicházejí stále s výkonnějšími grafickými akcelerátory vyžadující potřebu přesného měření výkonu. Výsledky měření jednak slouží k porovnání akcelerátorů mezi sebou a dále též k určení problematických operací, které přivítají zejména programátoři.

2 MĚŘENÍ

Celková výkonnost obecně jakéhokoliv zařízení je ve skutečnosti určena jen úzkými místy (*bottleneck*). V případě OpenGL jsou úzkými místy:

1. aplikace - nedodává dostatečně rychle data do OpenGL
2. zpracování geometrických dat - OpenGL není schopno rychle zpracovávat vrcholy
3. zpracování rastrových dat - OpenGL nestačí rasterizovat grafická primitiva

Úzká místa hrají klíčovou roli při návrhu programů orientovaných na 3D grafiku, kdy se hledá kompromis mezi požadavky a možnostmi grafického akcelerátoru (viz [2]).

2.1 PRINCIP

Měření výkonu spočívá v měření doby za jakou se zkoumaná operace provede a kolikrát se provede. Poměr obou hodnot udává výkonnost. Při měření se sleduje počet vrcholů/pixelů/snímků vykreslených za sekundu v závislosti na velikosti vykreslovaného objektu.

2.2 REALIZACE

Problémem při měření výkonu je přesné měření časových okamžiků. Následující kód ukazuje praktickou realizaci měření:

```
Time a = getTime();
Time delta = getTime() - a;
[...]
Time start = getTime();
drawComplexScene();
glFinish();
Time elapsed = getTime() - delta - start;
```

Při inicializaci je nejprve zjištěna doba potřebná k vykonání funkce `getTime()` a uložena do `delta`. Je-li doba příliš malá, je vhodné provést měření pro více opakování a určit průměrnou hodnotu. Ve výkonném kódu je zaznamenán aktuální čas, je provedeno několik zkoumaných operací ve složitější scéně, následuje vynucené dokončení všech rozpracovaných úkonů (`glFinish()`) a nakonec je vypočítána celková doba provádění jako rozdíl aktuálního a dříve zaznamenaného času zpřesněná dobou vykonávání funkce `getTime()`.

Uvedený kód zpřesňuje měření dvěma způsoby. Jednak odečítá dobu potřebnou pro vykonání funkce `getTime()` od celkového času a dále kreslí složitější scénu ve které několikrát opakuje zkoumanou operaci.

Podmínkou úspěšného zpřesnění je, aby doba kreslení složitější scény byla mnohonásobně vyšší než doba volání funkce `drawComplexScene()`. Za tohoto předpokladu bude doba volání zanedbatelná a neměla by se ve výsledku projevit. Problémem ovšem zůstává vhodná „složitá scéna“. Scéna vyžaduje další přídavný kód (proměnné, cykly . . .), který sice vytváří potřebnou složitost, ale opět vnáší do měření další nepřesnosti.

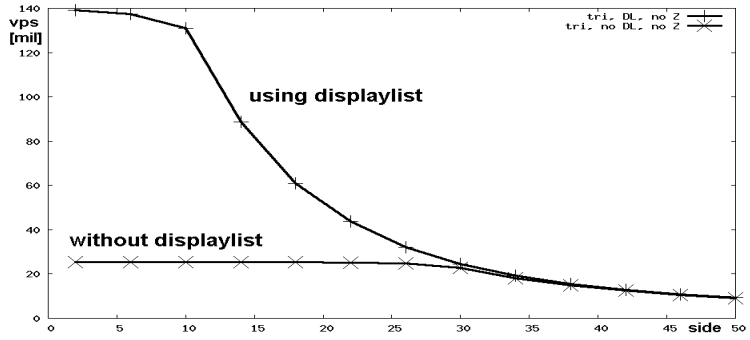
3 VÝKONNOSTNÍ CHARAKTERISTIKY

Na následujících obrázcích jsou uvedeny charakteristiky porovnávající výkonnost klasického kreslení a kreslení pomocí speciálních technik (*displaylist*, *Z-buffer*) v závislosti na velikosti objektu (čtverec sestávající ze dvou nepřekrývajících se trojúhelníků).

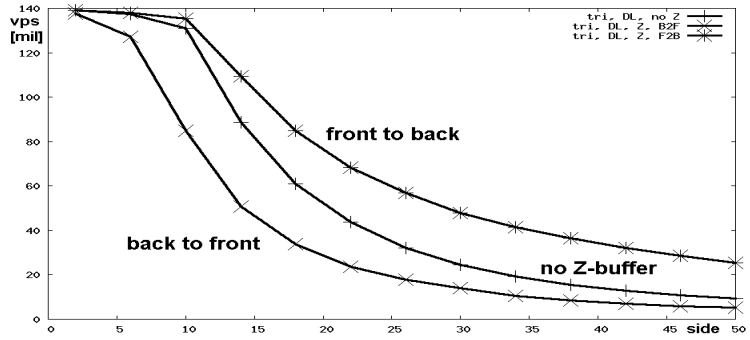
Data byla naměřena v průběhu testování měřiče na grafické kartě GeForce 6800. Na svislé ose je vynesena výkonost v miliónech vykreslených vrcholů za sekundu a na vodorovné ose je velikost strany čtverce v pixelech.

Na obrázku 1 jsou porovnány výkonnosti kreslení klasických trojúhelníků a kreslení pomocí displaylistu. Nárůst výkonu při použití displaylistů je zcela evidentní (pro malý objekt téměř 7 násobné zvýšení). Nutno podotknout, že v charakteristice není zahrnuta doba kompilace displaylistu. Zlom v průbězích je způsoben rasterizačním stupněm, který nestačí dostatečně rychle vykreslovat obrazce.

Obrázek 2 mapuje výkonnost displaylistů při kreslení zepředu dozadu a naopak. Průběhy dokazují, že je výhodné ve scénách provádět seřazení objektů podle vzdálenosti od pozorovatele a podle něj vykreslovat. Výsledný výkon bude záviset na umístění a velikosti objektů. Jako nevýhodné se jeví kreslení zezadu dopředu.



Obrázek 1: Výkonnost displaylistů a klasického vykreslování



Obrázek 2: Výkonnost při kreslení zepředu dozadu a naopak

4 ZÁVĚR

Bylo testováno 10 grafických akcelerátorů na Windows XP. Provedená měření zahrnovala testy vykreslování klasických trojúhelníků, vykreslování pomocí displaylistů, vertex arrays, vertex buffer objects a zkoumaly se vlivy pro různá nastavení Z-bufferu, texturování a přepínání textur, různých druhů světel, změny barev a materiálů.

Většina testů odpovídala očekávání. Velkým překvapením však byl test vertex arrays, vertex buffer objects. Podle výsledků mají obě metody jednak vyrovnanou výkonnost a dále vykazují oproti klasickému vykreslování téměř nulový nárůst výkonu. Jednou z možných příčin může být špatná kvalita ovladačů, ale s jistotou nelze vyloučit ani chyby v měřiči.

Další vývoj by se měl ubírat především důkladnou kontrolou vykreslování pomocí vertex arrays/vertex buffer objects a revizí způsobu měření času. Zajímavé by bylo též porovnání dosažených výsledků na technologii DirectX. Teoreticky by tím bylo možné snadněji lokalizovat skutečnou příčinu.

REFERENCE

[1] Tišnovský, P.: Grafická knihovna OpenGL. `www.root.cz`.

[2] Shreiner D.: Performance OpenGL. `www.performanceopengl.com`.

[3] Wright, R. S.: OpenGL SuperBible. Waite Group Press, 1999.