FAST RAY-TRIANGLE INTERSECTION

Jiří Havel
Master Degree Programme(2), FIT BUT
E-mail: xhavel18@stud.fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

ABSTRACT

Ray-triangle intersection is basic operation in computer graphics. Speed of this calculation is critical, because it is bottleneck in raytracers and similar applications. This paper presents new modification of barycentric test, optimized for modern CPU architectures.

1. ÚVOD

Výpočet průsečíku paprsku s trojúhelníkem je jeden ze základních algoritmů v počítačové grafice. Nejčastěji se používá pro realistické zobrazování metodou sledování paprsku, ale existuje mnoho dalších použití, jako detekce viditelnosti.

Výpočet průsečíků je spolu s algoritmy procházení scény nejnáročnější částí raytraceru. V závislosti na charakteru scény a způsobu práce s ní se mění náročnost těchto dvou částí. Tato práce se zaměřuje pouze na výpočet průsečíku.

V této práci je prezentována mnou navržená modifikace tohoto výpočtu, založená na barycentrickém testu. Tato metoda je optimalizována pro použití na nejnovějších procesorech architektury x86.

2. DŮLEŽITÉ VLASTNOSTI ARCHITEKTURY X86

Ačkoliv jsou moderní procesory velmi výkonné, je poměrně obtížné dosáhnout jejich plného výkonu. Optimalizací kódu se podrobně zabývá [4], zde uvedu jen nejdůležitější body.

- Všechny nové procesory mají dlouhou pipeline, což činí z větvení potenciálně náročnou operaci. Predikce větvení nefunguje ani zdaleka dokonale.
- 32 bitové verze mají poměrně málo registrů. Lepší situace je u 64bitových verzí.
- Související data by měly být uloženy v jedné řádce cache. Přístup do pole struktur je podstatně rychlejší, než do několika oddělených polí.

3. ALGORITMUS

Algoritmů pro výpočet průsečíku existuje velké množství. Využívají barycentrické, nebo plückerovy koordináty, orientované objemy a další postupy. Zaměřím se na barycentrický test a jeho modifikaci.

3.1. Barycentrický test

Tento test se skládá ze dvou částí. První je výpočet průsečíku paprsku s rovinou trojúhelníku a druhou je výpočet barycentrických souřadnic a test, zda průsečík leží uvnitř trojúhelníku, nebo vně. Výpočet průsečíku znamená vyřešit soustavu rovnice přímky paprsku (1) a roviny trojúhelníku (2) pro parametr t a bod P.

$$P = O + D \cdot t \tag{1}$$

$$P \cdot N + d = 0 \tag{2}$$

Barycentrické koordináty beta a gamma pro bod P se získají z rovnice (3), kde A, B a C jsou body trojúhelníku.

$$P = A + \beta (B - A) + \chi (C - A) \tag{3}$$

Tuto rovnici je možno řešit různě. V Badouelově implementaci [3] se trojúhelník a průsečík promítnou do jedné ze souřadných rovin a vyřeší se soustava dvou rovnic, vzniklá z (3). Krom dvou operací dělení vyžaduje tento postup ještě komplikované větvení, což silně omezuje jeho výkon.

V [1] a [2] je tento postup modifikován tak, že je spolu s rovinou trojúhelníku uložen i příznak, do které souřadné roviny je trojúhelník promítnut. Díky tomu je možné pro dvě zbývající osy u a v upravit výpočet barycentrických souřadnic do tohoto tvaru :

$$\begin{aligned} \beta &= U_\beta \cdot P_u + V_\beta \cdot P_v + W_\beta \\ \chi &= U_\chi \cdot P_u + V_\chi \cdot P_v + W_\chi \end{aligned} \tag{4}$$

Hodnoty u, v a w jsou vypočteny dopředu a uloženy spolu s rovinou. Díky tomu již není k výpočtu průsečíku potřeba původní body trojúhelníku.

3.2. Modifikace

Tato modifikace se na rozdíl od předchozích nesnaží za každou cenu minimalizovat počet operací. Místo toho se snaží o zjednodušení logiky. Nepromítá trojúhelník do jedné ze souřadných rovin ale všechny výpočty provádí ve 3D. Výpočet průsečíku s rovinou je shodný s předchozími metodami. Liší se výpočet barycentrických souřadnic. Výpočet souřadnice beta vypadá následovně :

$$\begin{aligned} N'_\beta &= (A - C) \times N \\ N_\beta &= \frac{N'_\beta}{N'_\beta \cdot B} \\ d_\beta &= N \cdot A \\ \beta &= P \cdot N_\beta + d_\beta \end{aligned} \tag{5}$$

Pro souřadnici gamma je postup analogický. Geometricky se jedná o konstrukci dvou rovin, kolmých na rovinu trojúhelníku a procházejících stranami AB a AC. Vzdálenost bodu od těchto dvou rovin udává jeho barycentrické souřadnice. Spolu s původní rovinou je tedy trojúhelník popsán třemi rovinami.

Ačkoliv je pro tuto modifikaci potřeba více předpočítaných hodnot, není pro jejich uložení potřeba více paměti. Pro efektivní práci s cache bylo nutné zarovnat předpočítaná data na násobek 16B, čímž zabíraly stejný prostor.

4. TESTY

Rychlost metody z předchozí kapitoly byla porovnána s rychlostí metody z [1] a [2]. Tato modifikace je v tabulce pod názvem roviny. Obě tyto metody byly implementovány v jazyce C naprosto stejným způsobem, aby byla minimalizována možnost chyby. Pro překlad byl použit překladač GCC se zapnutými maximálními optimalizacemi.

Výkonnost těchto metod byla testována pomocí náhodně generovaných trojúhelníků a paprsků. Pro srovnání chyby byly průsečíky spočteny taktéž ve dvojité přesnosti. Ani jedna z metod neminula některý z trojúhelníků.

Výběr instrukční sady měl velký vliv na rychlost i přesnost výpočtu. Je možno použít jak starší x87, tak podmnožinu SSE pro práci se skalárními daty. Výpočet s x87 je přesnější, díky 80bitové šířce mezivýsledků. Výpočet pomocí SSE je rychlejší a u novějších procesorů je to preferovaný způsob. U výpočtů pomocí x87 byla relativní chyba vypočtených parametrů řádově 10^-6. U výpočtů pomocí SSE klesla na 10^-5. Rozdíl v přesnosti mezi jednotlivými metodami byl v obou případech zanedbatelný.

Výpočet	Původní	Modifikovaný
X87 (-mfpmath=x87)	48,16	49,06
SSE (-mfpmath=sse)	68,92	84,69

Tabulka 1: Výkonnost jednotlivých metod v milionech testů za sekundu

5. ZÁVĚR

Ačkoliv je tato modifikace na první pohled složitější než původní metoda, má v některých případech i o 20% vyšší výkon. V jiných případech je rychlost srovnatelná. Obě metody jsou přibližně stejně přesné a ačkoliv tato nová modifikace vyžaduje více předpočítaných dat, ty díky nutnosti zarovnání pro efektivní práci s cache nezabírají více místa, než data pro předchozí metodu.

Další zvýšení výkonu může být získáno přes instrukce pro skalární součin, přítomné v nejnovější generaci procesorů. Implementace pomocí těchto instrukcí bude náplní navazující práce, současně s měřením výkonnosti této metody pro svazky paprsků.

LITERATURA

[1] Benthin, C.: Realtime Raytracing on Current CPU Architectures, [PhD thesis], Saarland University, leden 2006

[2] Wald, I.: Realtime Ray-Tracing and Interactive Global Illumination, IT - Information Technology, 2006, (invited article, in german)

[3] Badouel, D.: An Efficient Ray-Polygon Intersection, Graphics Gems, 1990

[4] Agner, F.: Optimizing Software in C++ : An optimization guide for Windows, Linux and Mac plattforms, <http://www.agner.org/optimize/>, (únor 2008)