ORSI 3. beadandó feladat - dokumentáció

Ajler Andrea - Y0R2IJ

2016. december 7.

1. Feladat

Háttértörténet:

Egyre közelebb a karácsony, ami a hallgatók számára a vizsgaidőszakot jelenti. A diákok a projektmunkák végére értek, a leadási határidő pedig vészjóslóan közel került hozzájuk. Az ELTEngine nevű játék-motrot használták a fejlesztésekre, ebben valósították meg az év elején kitűzött célokat. A végső tesztek során azonban hibásnak érezték a program működését, a karakteranimációk megvalósításáért felelős modul műveletigénye hatalmas volt, így használhatatlanná vált éles projektben való alkalmazásra. Annak érdekében, hogy helyesen működjön a szoftver, a fejlesztőkhöz fordultak, akik - ismerve a B szakirányos hallgatók példaértékű munkáját - ismét az ELTE-s diákokat kérték meg a probléma elhárítására. A rosszul működő komponens forráskódját végignézve rájöttek, hogy a hiba oka egy figyelmetlen fejlesztő barkácsolásának köszönhető, aki nem ismerte a pipeline fogalmát, így négy, egymásba ágyazott ciklusba szervezve futott a kódrészlet, ami a karaktereket reprezentáló vertexek sorozatán elvégezte a szükséges mátrix-transzformációkat. Szerencsére a B-s fejlesztők szorgalmasan jártak ORSI-előadásra és gyakorlatra, így magától értetődő volt, hogy adatcsatorna tételére visszavezetve máris N+M-es futásidejű kódot kaphatnak, ami már megfelelt a megrendelők igényének. A patch kiadása és az ELTEngine frissítése után a hallgatók megelégedve készülhettek a vizsgaidőszakra, hiszen az év során felmerülő összes akadályt sikerült leküzdeniük a kemény munkával.

A feladat tehát a következő probléma megoldása volt:

A bemeneti fájlok egyike, az *input_matrices.txt* első sorában egy M pozitív egész olvasható, ennyi lineáris transzformációt kell elvégezni az objektumokat reprezentáló vektorokon, míg a következő 4xM sorban egy-egy 4x4es mátrix sorfolytonos reprezentációja található szóközökkel elválasztva. (4 egymás utáni sor jelent egy mátrixot.) A másik fájl, az *input_points.txt* tartalmazza a pontok listáját, amikre alkalmazni kell az előbbi mátrixokkal való szorzást. Ennek első sora egy egész szám, N, ez után található N sor egy-egy 3 dimenziós vektor koordinátáit írja le (szintén szóközzel tagolva).

Egy lehetséges *input_matrices.txt* fájl: (Az alábbi mátrix-transzformációk pl. a 2x-es méretezés, 90°-os forgatást az X tengely körül, majd a (3;-2;4) vektorral való eltolást jelentik.)

3
2 0 0 0
0 2 0 0
0 0 2 0
0 0 0 1
1 0 0 0
0 0 -1 0
0 1 0 0
0 0 0 1
1 0 0 3
0 1 0 -2
0 0 1 4
0 0 0 1

Egy lehetséges *input_points.txt* fájl:

8
1 1 1
1 1 -1
1 -1 1
1 -1 -1
-1 1 1
-1 1 -1
-1 -1 1
-1 -1 -1

A feladatban az adatcsatorna tételére visszavezetve kell megoldani a kitűzött problémát!

A főfolyamat dolga, hogy beolvassa a mátrixokat, majd M threadet létrehozva, s azokat egy-egy transzformációnak megfeleletetve átadja nekik az megfelelő mátrixokat. A pontokat az első mátrixot reprezentáló szálnak kell elküldeni, majd az utolsó leképezést megvalósító gyerektől fogadja a megfelelően transzformált vektorokat. Ezek után írja soronként az *output.txt* fájlba az így kapott eredményt.

Az indított szálak feladata, hogy fogadják a vektorokat a megelőző gyerektől (az első transzformáció esetén a mastertől), elvégezzék a mátrix-vektor szorzást, majd továbbítsák az így kapott részeredményt a következő folyamatnak. (Az utolsó számítás esetén a masternek.) A teljes memóriahasználat csökkentése érdekében használhattok külön szálat a beolvasás és kiírás elvégzésére.

A dokumentációban mindenképp szerepeljen a visszavezetés miléntje, azaz a megfeleltetés az absztrakt programhoz! A fejlesztői fejezetben szeretnénk mérésekkel alátámasztva látni, hogyan is skálázódott a program a különböző méretű bemenetek esetén! Egy lehetséges *output.txt* fájl:

5 -4 6
5 0 6
5 -4 2
5 0 2
1 -4 6
1 0 6

1 -4 2
1 0 2

2. Felhasználói dokumentáció

2.1. Környezet

A program több platformon futtatható, nincsen dinamikus függősége. Telepítésre nincs szükség, elegendő a futtatható állomány elhelyezni a számítógépen.

2.2. Használat

A program elindítása egyszerű, mivel nem vár parancssori paramétereket, így parancssoron kívül is lehet futtatni. A fájl mellett kell elhelyezni az *input_matrices.txt* és az *input_points.txt* fájlt, melyet feldolgoz és az eredményt az "*output.txt*" nevű fájlba írja, a bemeneti sorrend alapján.

Egy lehetséges bemenetet tartalmaz a mellékelt *input_matrices.txt* és *input_points.txt* tesztfájlok. Saját bemeneti fájlok esetén fontos, hogy a feladatban megadott szempontok alapján írjuk az adatokat az inputfájlba, mivel a feladat megoldása és a program futása során feltesszük, hogy az adatok a fájlban helyesek, így erre külön ellenőrzés nincsen a kódban.

3. Fejlesztői dokumentáció

3.1. A megoldás módja

A kódot logikailag három részre bonthatjuk, egy fő és egy alfolyamatra. A főfolyamatot a $main()$ függvény fogja megvalósítani, ez fogja beolvasni az inputfájlok tartalmát. Az alfolyamat a csatornában szereplő blokkok mátrix szorzásait valósítja meg. $M + 1$ sort hozunk létre M szorzás folyamat köré és közé. Miután minden M szorzáson végig tudott menni az N vektor, akkor kiírjuk a transzformált vektorokat az *output.txt*-be azokat.

3.2. Implementáció

A C++11 nyelvi elemeit kihasználva egy $std :: thread$ típussal fogja a különböző szálakat elindítani és tárolni. A teljes implementáció egyetlen forrásfájlba szervezve, a *Source.cpp* fájlban található meg.

3.3. A feladat visszavezetése adatcsatorna tételre

Adott egy $F = f_n \circ \ldots \circ f_0$ függvénykompozíció, amelynek értékét a $D =< d_1, \ldots, d_m >$ sorozatban adott argumentumokra kell elemenként meghatározni.

$$A = \underset{x_0}{Ch} \times \underset{\overline{x_0}}{Ch} \times \ldots \times \underset{x_n}{Ch} \times \underset{\overline{x_n}}{Ch}$$

$$B = Ch \times Ch \times \ldots \times Ch \times Ch$$
$$x'_0 \quad \overline{x_0}' \qquad\qquad x'_n \quad \overline{x_n}'$$
$$Q = (x_0 = \overline{x_0} = x'_0 = \overline{x_0}' = D \wedge x_n = \overline{x_n} = x'_n = \overline{x_n}')$$
$$S = (\|_{n=1}^{n} x_i := <>, \{\square_{n=1}^{n} x_i, x_{i+1} := lorem(x_i), hiext(xi+1, f_i(lov(x_i))) \; ha \; x_i \neq <>\})$$

A feladatban az F függvény a mátrixtranszformációnak feleltethető meg, ami a mátrixszorzások kompozíciója.

3.4. Fordítás

A program forráskódja a *main.cpp* fájlban található. A program fordításához követelmény egy c++11 szabványt támogató fordítóprogram megléte a rendszeren. A legnépszerűbbek az *msvc*, *g++* és *clang*. A fordítás menete (g++ 4.8.2-es verziójú fordítójával) a következő: *'g++ -std=c++11 main.cpp'* Az *-std=c++11* kapcsoló szükséges, mert alapértelmezetten régebbi c++ szabványt támogat a fordító.

3.5. Tesztelés

A program jól működését különböző tesztinputok segítségével végeztem. A minták az általánostól a teljesen szélsőséges esetekig terjedtek.