Kötelező félévi házi feladat (Modellellenőrzés)

A feladat címe: **GPU**

Konzulense: **Hegedüs Ábel**

Leírás

Nagymértékű párhuzamos végrehajtás támogatása miatt a GPU-kat (Graphics Processing Unit) már nem csak grafikai alkalmazásokban használják, hanem más, jól párhuzamosítható feladatok megoldására is. A házi feladatban a modern GPU-k belső felépítését kell egyszerűsített módon modellezni.

A GPU működésének alapja a csővezeték (pipeline), amelyen végighaladnak az adatok és közben transzformációk végezhetők el rajtuk. A csővezeték részei (a feladatban leegyszerűsítve) a shaderek, amelyek elkülönülő komponensek, de közös erőforrásokat (osztott memória) használnak, és függenek egymástól azáltal, hogy az egyik kimenete a másik bemenete lesz.

A modellezett GPU-ban a csővezeték első része a vertex shader, utána következik a geometria shader, végül a fragmens shader. A közösen használt erőforrások a textúra memória, a vertex puffer, és a szín és Z puffer.

A shaderek működése hasonló, alapesetben bemenetre várnak, majd amikor bemenet érkezik, kiolvassák a végrehajtáshoz szükséges adatokat a megfelelő erőforrásokból (ehhez az erőforrásra olvasási zárat kell helyezniük). Ezután végrehajtódik a transzformáció. Ha szükséges az írás, akkor a megfelelő erőforrások írása megtörténik (amelyekre közben írási zárat kell elhelyezni). Végül a kimenetre kikerül a kiszámolt adat. Fontos, hogy olvasási zárból több is helyezhető egy erőforrásra, de írási csak egy.

A vertex shader a vertex puffert és a textúra memóriát olvassa. A geometria shader a textúra memóriát olvassa és a vertex pufferbe ír. A fragmens shader a textúra memóriát, valamint a szín és Z puffert olvassa, és ugyanezeket írja.

A bemenet beolvasása 1ms időt igényel, az erőforrásból való olvasás 2ms, az erőforrásba írás 4ms, a transzformáció végrehajtása 8ms, míg a kimenetre írás szintén 1ms.

A párhuzamosítás úgy jelenik meg a rendszerben, hogy a shaderekből többet alkalmazunk, amelyek a bemenet egy-egy részeit felosztják maguk között így a teljes végeredmény előbb áll elő, mintha egy csővezeték számolná a teljes bemenetet.

A házi feladatban 2 vertex shader, 2 geometria shadert és 2 fragmens shader tartalmazó GPU-t modellezünk, a bemenet mérete pedig 4 (tehát összesen 4 bemeneti adat halad végig a csővezetéken). Amikor a teljes kimenet előállt, akkor kezdődik a következő bemenet feldolgozása. A shaderek nincsenek összedrótozva, hanem ha van megfelelő típusú bemenet, akkor végrehajtódnak (pl. egy várakozó geometria shader akkor fog bemenetet beolvasni, ha egy vertex shader végzett a feldolgozással). A shaderek ki-bemeneteit tartalmazó memóriát nem kell zárolni (az olvasás/írás atomi művelet). Fontos, hogy a rendszer nem várakozhat egy helyben végtelen ideig,

Az ellenőrzendő követelmények

Temporális logikai kifejezések és modellellenőrzés segítségével igazolja az alábbi követelmények teljesülését (illetve a követelmények nem teljesülése esetén ellenpélda segítségével magyarázza meg a követelmény megsértésének okát és indokát)!

1. A modellben nincs deadlock.
2. A teljes kimenet előállítása nem kerül többe, mint 400ms. (a valóságban ez a követelmény grafikai alkalmazásoknál néhány μs)
3. Nincs olyan eset, amikor egy erőforráson egy írási záron kívül másik zár is van.
4. A fragmens shader-ekre várakozó bemenetek száma mindig kisebb, mint a geometria shaderre várakozó bemenetek száma.