**Önlabor – SPH folyadék szimuláció (általános számítások a GPUn)**

Horváth Ákos – DKILK6

Gurubi Barnabás - DXEXVR

Konzulens: **Tóth Balázs**

Mit is jelent az a GPGPU?

Alapvetően, ha programozásról beszélünk mindenkinek elsőre az egyszálú (esetleg pár szálú) CPU-n futó programok jutnak az eszébe. Ez miért is jelenthet problémát? Vannak olyan feladatok, amelyek nagymértékben párhuzamosíthatóak, és ha ezt észre vesszük és ki is használjuk sokkal gyorsabb programokat érhetünk el. Mint tudjuk a CPU csak korlátozottan képes a többszálúságra így az olyan problémák esetén, ahol egyszerű számításokat tudunk nagy számban párhuzamosan végezni érdemes a GPU-hoz fordulni. Ilyenkor beszélünk GPGPU alkalmazásról. (GPGPU = General-purpose computing on graphics processing units, azaz általános célú számítások a grafikus processzoron)

Ahhoz persze hogy a GPU-n ilyen dolgokat végezhessünk, természetesen szükség van a megfelelő eszközökre, gondolok itt mind a megfelelő videokártyákra, ill. a megfelelő driverekre, fejlesztő környezetekre. Az előző féléves témalaborunk esetén megismertünk és összehasonlítottunk két lehetséges környezetet: az OpenCL-t és a CUDA-t. A munkánk során arra jutottunk, hogy habár a CUDA csak az Nvidia bizonyos GPU -in futtatható, az OpenCl-lel ellentétben, amely minden kártyán képes futni, jóval könnyebb átlátható, jobban olvasható kódot írni CUDA segítségével. (ez nyilvánvalóan adódik abból, hogy nem akarta az Nvidia, hogy minden kártyán fusson az ő környezetük és így a sajátjukra tudták optimalizálni a saját felhasználóbarátabb környezetüket tudták megvalósítani) Emellett pedig a gyártó számos debugolásra, optimalizálásra használható eszközt is ad a csomagban, ami szintén nagyban tudja segíteni a fejlesztést.

Milyen területen lehet használni a GPU-n való számításokat? Az egyik ilyen GPGPU alkalmazási terület a fizikai szimulációk. Mi a témalaborunk és idén az önlabor témánk keretein belül is ezt a területet vizsgáltuk meg, egész pontosan a folyadékszimulációt két különböző megközelítéssel: áramlás alapú és részecske alapú megközelítéssel.

Folyadékszimuláció

Napjainkban elég felkapott téma a folyadékok szimulálása (computational fluid dínamics – CFD) a számítógépes grafikában. A tudósok arra fókuszálnak, hogy minél jobb módszereket találjanak a szimulációra, megjelenítésre. A közös az összes módszerben, hogy valamiféle matematikai egyenletek segítségével szeretné leírni a folyadék mozgását. A folyadékot gyakran az egyik legösszetettebb, valósághűen a legnehezebben szimulálható jelenségnek mondják. Offline szimulációk a leggyakoribbak, amelyek lehetnek mind részecske alapú, mind rács alapú, vagy akár a kettőt ötvöző megoldások is.

Napjainkban talán a rács alapú az elterjedtebb megközelítés. Azonban ha valósidejű 3D-s szimulációt szeretnék akkor nem ez az optimális megoldás. Erre a problémára vezették be a részecskéket. Jelenlegi tudásunk szerint a részecske alapú megközelítés egy jó választás lehet, ha kisebb méretű folyadékot szeretnénk szimulálni, amelyen esetleg interakciókat is hajtunk végre. Emellett a folyadékok felszínét is részecskékkel lehet valóságosabban szimulálni.

Alapvetően a folyadékszimulációnak két része van: a szimuláció és a megjelenítés. Mivel a folyadékok megjelenítésére már nagyon sok megoldást született (pl. ray-tracing) amely élethű végeredményt hoz mi jelenleg a szimulációs lépésre fektettünk nagyobb hangsúlyt.

Most pedig lássuk részletesebben is mi rács-alapú ill részecske-alapú megközelítés:

Rács alapú vs. Részecske alapú (SPH)

Rács alapú *(röviden)*

A témalaborunk keretén belül foglalkoztunk az ilyen módon végzett folyadékszimulációval. Ebben az esetben az alapvető gondolat az, hogy felosztjuk a folyadékunkat egy négyzetrácsra (csak 2D ban készült el végül, de a 3D megvalósítás teljes analóg) és azt határozzuk meg, hogy az adott mezőn milyen irányú és nagyságú a folyadék sebessége, amely majd szállítani fogja az anyagot. Ez az úgynevezett *Euler-i folyadék* megközelítés. Az erőssége, hogy bizonyos fizikai tulajdonságokat jól le lehet vele írni pl. nyomás, sűrűség a rácsnak köszönhetően, ugyanakkor a hátránya is éppen a rácsban rejlik, a folyadék nem képes kilépni ebből, csak arra a térfogatra van szorítva, amelyen a rácsot definiáltuk. Persze ma már mindenféle adaptív rácsokat is kitaláltak, de mi ezekkel nem foglalkoztunk a témalabor keretében.

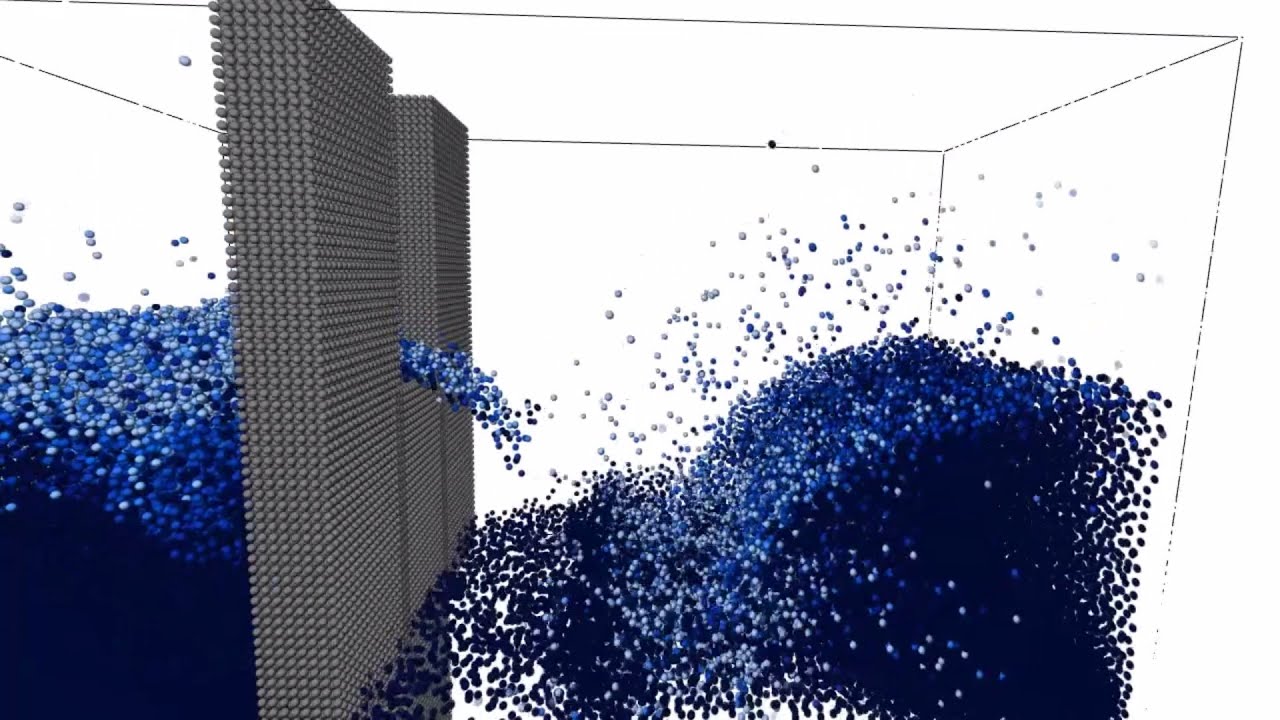
A szimulációs lépések a Navier-Stokes egyenletekre azok megoldására épülnek. Az egyenleteket Claude Navier és George Gabriel Stokes állították fel 1822-ben a folyékony anyagok mozgásának, áramlásának leírására. Alapvető elképzelésük az volt, hogy az anyagban fellépő feszültségnek két összetevője van: a folyékony anyag sebességgradiensével arányos diffúziós (vagyis egy a viszkozitást jellemző) kifejezés összetevőből és egy nyomás összetevőből áll. Mi a szimulációs lépésekben ezeket számoljuk és ez alapján jelenítjük meg a folyadékunkat. A



Ennek a megközelítésnek a jellentősége abban rejlik, hogy az egyenletek mind elméleti mind gyakorlati haszonnal bírnak hiszen segítségükkel leírható pl. az időjárás, óceánokban áramlatok, repülőgépek szárnyai körül észlelt áramlás, vagy például csillagok galaxisokon belül leírt mozgása is. A repülőgépek és gépjárművek tervezése mellett használhatóak atmoszferikus szennyezés felmérésére, sőt akár véráram szimulálására is.



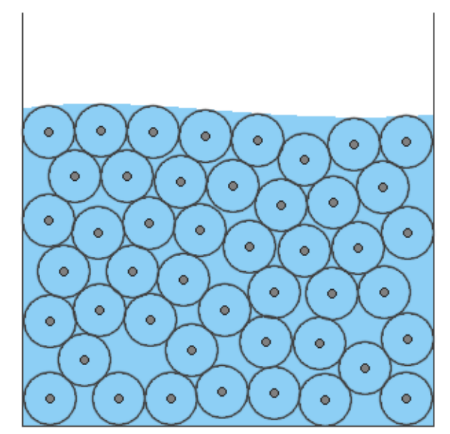
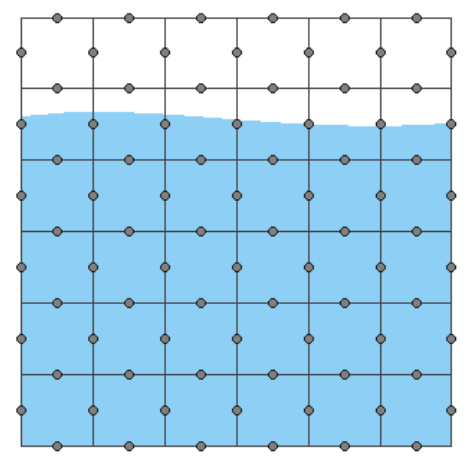
Részecske alapú - SPH

Az SPH (smoothed particle hydrodynamics) eredetileg az asztrofizikából származik. Széles körben használják olyan jelenségek szimulálására, amelyek így könnyebben kifejezhetővé, megérthetővé válnak. Ez egy olyan interpolációs módszer, amely a különböző értékeket és a különböző folytonos értékek deriváltjait úgy határozza meg, hogy diszkrét mintavételezési pontokat használ. Ezek a minta pontok a részecskék, amelyek számos fizikai tulajdonságot hordozhatnak pl. tömeg, pozíció, sebesség stb. De a részecskék akármilyen egyéb fizikai tulajdonságot is szállíthatnak a problémának megfelelően, mint pl. hőmérséklet, nyomás stb.

Az önlaborunk során első körben egy 2D valós-idejű szimulációt implementáltunk, amely a CPU-n fut. Majd ezt ültettük át 3D és CUDA segítségével a GPU-ra.

Lagrange-i folyadékok

A rács alapú megközelítés esetén láthattuk, hogyan épül fel az Euler-i megközelítés összenyomhatatlan folyadékokra. Ha rácsok helyett részecskéket használunk az jelentősen egyszerűsítheti az egyenleteket. A szimulációnk során azt feltételeztük, hogy az összes részecske szám állandó, ill. hogy egy részecske tömeg változatlan marad a szimuláció során ebből következik, hogy a tömegmegmaradás törvénye mindig érvényes marad. Ezzel rögtön egy egyenletet kiiktattunk az Euler-i megközelítésből ahol a rácsalapú megközelítés miatt erre is külön számolni kellett. Egy másik jelentős könnyítés abból adódik, hogy a részecskék maguk meghatározzák a folyadékot. Az Euleri nézethez képest ez azt jelenti, hogy amíg az Euleri folyadék esetén a fizikai tulajdonságok függtek az időtől és a rácson elhelyezkedő pozíciótól is ebben az esetben minden fizikai tulajdonság csak az időtől (*t*) fog függeni. (ami szintén az egyenleteket egyszerűsíti)



2. ábra Lagrange-i folyadék

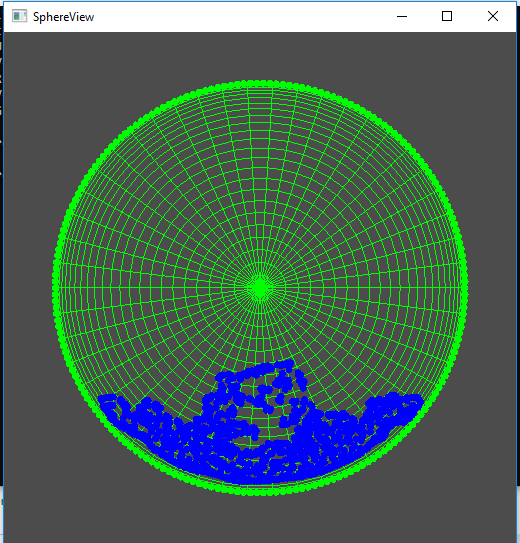
1. ábra Euler-i folyadék

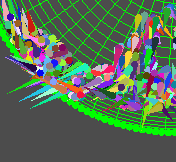
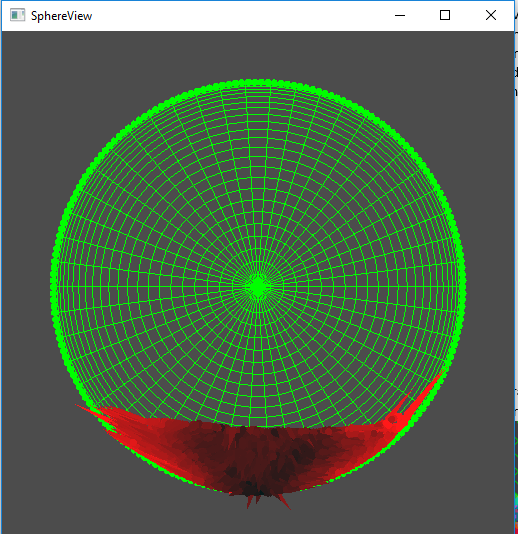
Szimuláció, szimulációs lépések

TODO

Vizualizáció

Habár az önlaborunk során a legnagyobb hangsúlyt a szimulációra fektettük érdemes pár szót ejteni a megjelenítésről is. A megjelenítéshez a projekt során az OpenGL által nyújtotta lehetőségeket használtuk. A megjelenítés esetén két fő kérdés merült fel: honnan éri el az OpenGL a részecske pozíciókat ill. milyen alakzattal, milyen módon rajzolja ki azokat.

A 2D-s CPUn implementált verzió esetén a részecskék pozíciója a számítógép memóriájából könnyedén elérhetőek voltak hiszen a CPUn való szimulációs lépések itt dolgoztak rajtuk. A számítógépes grafikában a megjelenítés számos részből áll gondolok itt shaderek írásásra, bufferek foglalására kiosztására. Mivel mi a lehető legegyszerűbb megjelenítést szerettük volna, kihasználtuk, hogy az OpenGL nyújt néhány alapvető funkciót amivel ezeket a műveleteket „megúszhatjuk”. Miután a memóriából kinyertük a pozíciókat egy egyszerű *glBegin(GL\_TRIANGLE\_FAN);* parancs után a megfelelő koordinátákat felsorolva megkaptuk a részecskéket reprezentáló kis köröket. Az eredmény a következően nézett ki:

A debugolás során arra jutottunk hogy szükséges lenne valamilyen módon a részecskék sebességét is vizualizálni, ezért a következő megoldások születtek:

A 3D-s megjelenítés során a szimuláció már a GPU-n futott ennek következtében a részecske pozíciók is a GPU memóriában voltak elérhetőek. Ezeket először a számítógép memóriába másoljuk majd úgy adjuk az OpenGL-nek. Ez látható, hogy felesleges lépés hiszen a GPU memóriából a számítógép memóriába majd vissza a GPU memóriaba másoljuk az adatot, azonban a jelenlegi részecskeszámmal és szimulációval nem jelent jelentős plusz időt. A megjelenítéshez az OpenGL beípétett függvényiet használjuk:

* *glutWireSphere(0.8, 50, 50);* - ami egy ráccsal kirajzolt gömb
* *glTranslatef(r.x, r.y, r.z);* - a gömbök eltolásához
* Ill. egyéb a kamerakezelést szolgáló beépített függvények (*gluPerspective*, *gluLookAt*).

Az eredményként kapott megjelenítés: (amelyben a WASD gombokkal a kamerát mozgathatjuk):

