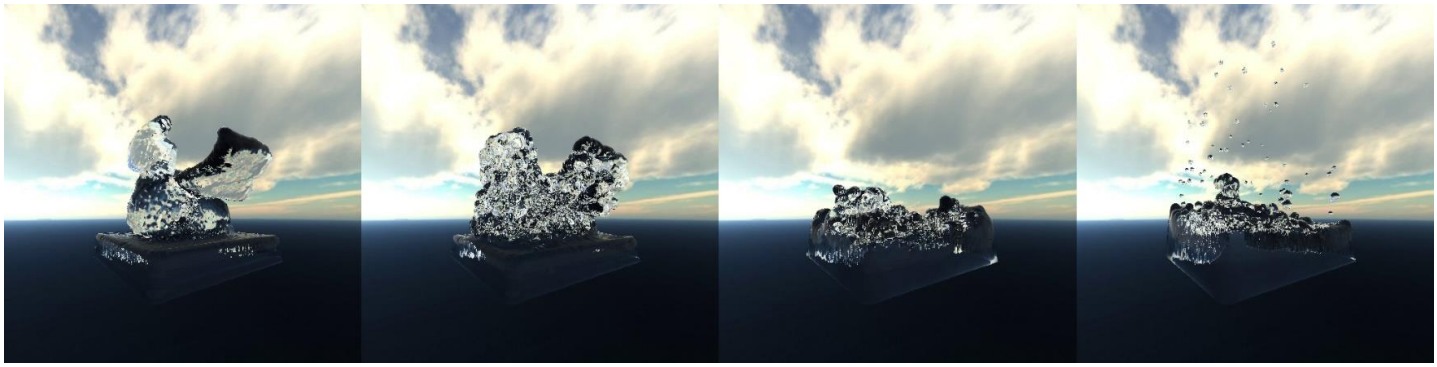
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Karakteranimáció által vezérelt folyadékszimuláció



Készítette:
Kárpáti Attila
MSc szakos hallgató
Burkus Viktória
MSc szakos hallgató

Konzulens:
Dr. Szécsi László
Egyetemi docens

Tudományos Diákköri Konferencia
Budapest, 2020

Abstract

Fluid behavior in both real-time and production systems must often be physically plausible while also allowing artistic control, possibly forcing otherwise impossible effects. In this paper we address the use of animated character meshes to influence flow behavior, including shaping, dissolving, morphing and animating liquid bodies. We discuss options for the underlying flow simulation, and describe an approach based on Smoothed-Particle Hydrodynamics that allows forcing liquids to take shapes dictated by triangle mesh solid models. We discuss algorithms for translating the constraints to simulation features, and elaborate on issues influencing simulation efficiency. We also describe the approach we used to visualize the particle-based fluid simulation. The method reconstructs the liquid surface using metaballs, constructing lists of relevant metaballs for every pixel in every frame. We evaluate alternative solutions to build these lists.

Keywords: fluid simulation, particle system, Smoothed-Particle Hydrodynamics, character animation, metaball, surface reconstruction

Absztrakt

Valós idejű és ipari alkalmazásokban egyaránt rendszeresen van szükség olyan folyadékszimulációra mely valósághű hatást kelt, de megengedi további mesterséges kényszerek alkalmazását is. Ezáltal olyan jelenetek is létrehozhatóak, melyek kizárólag fizikai szimuláció alkalmazásával nem volnának lehetségesek. A TDK dolgozat animált karaktermodellek alkalmazását mutatja be realisztikus folyadékszimuláció befolyásolására, ideértve a folyadékalapú karakterek felépítését, lebontását, átalakítását és animálását. Megvizsgáljuk a lehetséges folyadékszimulációs rendszereket és bemutatunk egy, a Smoothed-Particle Hydrodynamics módszeren alapuló alkalmazást ami lehetővé teszi a folyadék kényszerítését háromszöghálók kitöltésére. Továbbá megvizsgáljuk az alternatív lehetőségeket a realisztikus folyadékszimuláció mesterséges kényszerekkel való kiegészítésére, és ismertetünk néhány megoldást a szimuláció hatékonyságának növelésére. Ezután bemutatjuk a folyadékrészecskék megjelenítésére alkalmazható eljárást, ami a folyadék felszínét implicit felületként állítja elő. A valósidejű implementációhoz minden megjelenítési ciklus elött kiszűrjük az adott pixelben potenciálisan releváns részecskéket. A releváns részecskék pixelenkénti megtalálására és tárolására használt lehetőségeket bemutatjuk és összehasonlítjuk.

Kulcsszavak: folyadékszimuláció, részecskerendszer, Smoothed-Particle Hydrodynamics, karakteranimáció, metaball, felszín-rekonstrukció

Köszönetnyilvánítás

Szeretnénk köszönetünket kifejezni Dr. Szécsi László konzulensünknek, aki teljes egyetemi tanulmányunk során készséggel segített a számítógépes grafika tudományában eligazodni, tudományos munkáinkat szakmai iránymutatásával örömmel segítette, dolgozatainkat lektorálta és kérdéseinkkel bármikor bátran fordulhattunk hozzá.

Tartalomjegyzék

1 Bevezetés

A számítógépes grafika területén gyakran felmerülő probléma folyadék vagy gáz halmazállapotú objektumok által létrehozott térfogati jelenségek valós idejű szimulációja és megjelenítése. Számos esetben, művészi hatások létrehozásakor, természetfeletti képességek vagy képzeletbeli technológiák megjelenítésekor, a folyadék viselkedését realisztikus szimulációs elemek és hozzáadott kontrollhatások együttes kombinációja határozza meg. A dolgozatban bemutatott módszer egy realisztikus folyadékszimulációt egészít ki, lehetővé téve animált karakterek mesterséges feltöltését a szimulált folyadékkal. A folyadék felszínét implicit felületként, radiális bázisfüggvények (*metaball*ok) segítségével rekonstruáljuk. A rekonstruált felület realisztikus és valós idejű megjelenítésére a *ray marching* algoritmuson alapuló lehetőségeket javasolunk. A bemutatott módszer folyadékot tartalmazó, vagy akár teljesen folyadékalapú karakterek szimulációjára és megjelenítésére alkalmazható. A szimulációs és megjelenítő algoritmusok optimalizálására is vizsgálunk lehetőségeket.

1.1 Integráció

A folyadékok mozgásának vezérlésére animált karaktereket használunk. Ezek háromszöghálóként adottak, a háló csúcspontjait súlyok kötik egy csontváz csontjaihoz, és a csontváz ízületeinek mozgását időbeli kulcsok írják le. A valósidejű alkalmazások túlnyomó többségben ezt a sémát használják, így ezzel lehetővé tesszük a módszerünk egyszerű integrálást már meglévő rendszerekbe. A felhasznált háromszögháló-modellektől elvárjuk, hogy zárt felszínt reprezentáljanak, mert csak ebben az esetben dönthető el, hogy a tér egy adott pontja a háromszögháló belül vagy kívül helyezkedik el. A karaktert reprezentáló háromszögháló általános esetben nincsen megjelenítve, de a folyadékszimulációt mesterségesen módosító kényszerek kiszámítása a bemenetként kapott háromszögháló alapszik.

1.2 Alkalmazás

A módszerünk felhasználásával elérhető hatások egy jellegzetes forgatókönyve lehet, hogy kezdetben a folyadék viselkedését kizárólag a fizikán alapuló realisztikus folyadékszimuláció határozza meg. Ezt követően egy tetszőleges időponttól kezdve a karakter háromszöghálójából számított kényszereket alkalmazunk a

folyadékszimulációra, aminek hatására a folyadék az adott háromszögháló kitöltésére törekszik, miközben fenntartja a realisztikus folyadékszimuláció látszatát. A folyadék folyamatosan követi a háromszögháló animációját, amíg a mesterséges kényszer aktív. A következő lépés a hozzáadott mesterséges kényszer visszavonása lehet. Ebben az esetben a folyadék viselkedése ismét realisztikus lesz, tehát a felvett alakzat lebomlik.

A fentieken túl, lehetséges több, akár különböző háromszögháló felhasználásával előállított, mesterséges kényszer egyidejű alkalmazása, ezáltal lehetőség van a karakterek uniójának kitöltésére. Valamint a hozzáadott kényszerek időbeli váltogatásával átmenetet lehet képezni különböző karakterek között. Így alakváltó karakterek hozhatóak létre a különböző karakter-meshek közti átmenet definiálása nélkül. A bemutatott módszer egy másik alkalmazása lehet, ha a kényszert csak a folyadékrészecskék egy csoportjára alkalmazzuk, ezáltal a kiválasztott részecskék szétválaszthatóak a többitől. Ha a kiválasztott részecskék megjelenítése is eltérő, akkor a mesterségeses kényszer azt a hatást kelti, mintha az eltérő megjelenésű folyadékok válnának szét, vagy éppen keverednének össze.

1.3 Előnyök

A folyadékszimuláció használatának legnagyobb előnye folyadékalapú karakterek megjelenítésekor az, hogy lehetőség van a jelentben levő más objektumokkal komplex kölcsönhatások modellezésére. Mivel a folyadékalapú karakterek tényleges folyadékszimulációval vannak létrehozva, más, szilárd testek képesek a létrehozott karakterek alakját befolyásolni. Ez a fizikai interakció a mesterséges kényszer nélküli nyugalmi helyzetben és a mesterséges kényszert alkalmazott állapotban is érvényesíthető. Ha a folyadékalapú test csak a felület árnyalásában különbözne más szilárd testektől, ami széleskörűen alkalmazott megoldás hasonló hatás keltésére (pl. számítógépes játékokban), akkor külön módszerre lenne szükség a folyadékalapú test felépítésének animálásához és más testekkel való interakció megvalósításához. Tehát a mi módszerünk alkalmazásával művészi munka befeketetése nélkül lehet a már elkészült animált karaktert folyadékból felépíteni, lebontani, animálni és megjeleníteni.

2 Irodalomkutatás

Munkánk irodalomi háttere négy fő területet fog át. Elsőként a bemutatott módszer alapját alkotó folyadékszimulációs rendszerek változatait vizsgáljuk. Ezt követően az általános folyadékszimulációt mesterségesen módosító eljárásokat elemezzük. Végül a választott részecskealapú szimuláció megjelenítési opcióit és az optimalizálási lehetőségeket ismertetjük.

2.1 Folyadékszimuláció

A valósidejű folyadékszimuláció megvalósítása két fő problémát vet fel. Egyrészt a folyadékok viselkedését az 1. képleten látható, komplex és kiterjedt színterekben nehezen kiértékelhető, az idő függvényében definiált Navier-Stokes differenciálegyenletek írják le. Továbbá a folytonos mennyiségeket leíró Navier-Stokes egyenletek kiértékelése valós időben csak diszkretizációk bevezetésével lehetséges. A diszkretizációból fakadó hiba pedig műtermékek létrejöttéhez vezethet. A diszkertizációt nélkülöző megközelítések nem felelnek meg a céljainknak, mivel ezen eljárások olyan speciális esetekben alkalmazhatóak hatékonyan, ahol a megoldás analitikus formában megtalálható.

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)\mathbf{u} = -\nabla p + \mu\nabla\cdot(\nabla\mathbf{u}) + \mathbf{f}\,,$$

$$\nabla\cdot\mathbf{u} = 0\,,$$

1. képlet: Navier-Stokes egyenletek[1]

2.1.1 A szimulációs tér diszkretizálása

Két fő megközelítése van a Navier-Stokes egyenletek megoldásának. A Navier-Stokes egyenletek az *u* sebesség, *ϱ* tömegsűrűség, *p* nyomás, *μ* viszkozitás és *f* külső erők relációját írják le. A diszkretizációs lehetőségek első fő csoportja a térfogatot diszkretizálja a szimulációs tér felosztásával. Ebben az esetben a szimulációs tér adott térfogatú, diszjunkt geometriai elemekre van felosztva. Ha egy geometriai elemben a folyadék mennyisége ismert, akkor a tömegsűrűség kifejezhető a térfogat és a folyadék mennyiségének segítségével. Ezt követően a nyomás és az abból fakadó erő származtatható a geometriai elemek elhelyezkedéséből, az anyagi jellemzőkből és a

már ismert tömegsűrűségéből. A nyomásból fakadó erő és külső erők alkalmazásával a sebesség és gyorsulás származtatható.

A szimulációs teret diszkretizáló megoldások köze tartozik a *végeselem*-módszer. A *végeselem*-módszer alkalmazásakor a térfelosztásnak követnie kell a szimulációs tér határait. A szimulációs tér felosztása számításigényes, tehát a módszer valósidejű alkalmazásakor a szimulációs tér határait nem célszerű változtatni.

A szimulációs teret diszkretizáló folyadékszimulációk másik csoportja rácsszerkezeteken alapszik. A rácsszerkezetet könnyű kiegészíteni vagy levágni amikor megváltozik a szimulációs tér határa. Ebből kifolyólag nem szükséges a szimulációs teret minden esetben újra felosztani peremgeometria módosulásakor. Viszont a rács sűrűsége határozza meg a szimuláció részletességét. Ha a szimulációs tér határai jelentős mértékben megváltoznak, akkor az eredetileg választott rácssűrűség már nem lesz optimális az új térfogathoz. Ha a rács túl sok rácspontból áll, a szimuláció túl lassú lesz. Ha a rács túl kevés rácspontból áll, a folyadékszimuláció elveszti élethűségét, mert a rácsfelosztásból fakadó artifaktumok felerősödnek. Ebből következően mindkét szimulációs teret diszkretizáló megközelítés nehezen alkalmazható nagy kiterjedésű és összetett jelenetekben, ahol a folyadékszimuláció határait nehéz előre meghatározni.

2.1.2 Részecskerendszerek

A folyadékszimuláció diszkretizálásának másik fő iránya a részecskerendszer alapú megközelítés, ami a folyadékot részecskék halmazaként reprezentálja. Ebben az esetben a diszkretizált mennyiség a tömeg, mert egy adott részecske tömege állandó. A részecskék pozíciója nincs diszkretizálva, ebből következően a szimulációs tér változásakor nem merül fel túlzott számítási költség a teret diszkretizáló megközelítésekkel szemben. A részecskerendszerek alkalmazásakor a hatékonyság növelésének érdekében egy adott részecske pozíciójában való kiértékeléskor csak a közeli részecskék hatása van figyelmebe véve. Ha a közeli részecskék meghatározása megfelelő, akkor a szimulációban nem okoz különbséget a távolságalapon való szomszédossági előszűrés, mivel a távoli részecskék hatása elhanyagolható, vagy egyáltalán nincsenek hatással. A közeli részecskék meghatározására érdemes segédstruktúrákat alkalmazni. Fontos, hogy ezen segédstruktúrák felépítése ne legyen túl költséges, mert ellenkező esetben elveszítjük a részecskerendszerek előnyét a térfelosztáson alapolú módszerekkel szemben, ha a szimulációs tér határainak változása a segédstruktúra számításigényes újraépítésével jár.

2.1.3 Smoothed-particle hydrodynamics

Fontosnak tartjuk, hogy az általunk bemutatott módszer könnyen integrálható legyen általános megjelenítő vagy szimulációs alkalmazásokba. Ebből kifolyólag egy részecskealapú módszert választottunk a folyadékszimuláció megvalósítására, mert szeretnénk elkerülni a szimulációs határok változtatásának akadályozását. A számítógépes grafika területén gyakran alkalmazott, valósidejű és részecskerendszer alapú *Smoothed-particle hydrodynamics* szimulációs módszert alkalmaztuk. A folyadékszimuláció implementációját Micky Kelager publikációja[1] alapján valósítottuk meg. A következőkben az alkalmazott folyadékszimuláció egy rövid összefoglalása után csak az általunk javasolt módszerhez releváns részleteket emeljük ki. Micky Kelager Smoothed-particle hydrodynamics implementációjában használt tömegsűrűség és erők kiszámításához használt képletek megtalálhatóak a 2. képleten. A folyadékszimulációs képletek levezetése és a használt kernel függvények megtalálhatóak az eredeti publikációban.

Tömegsűrűség	$\rho(\mathbf{r}_i) = \sum_j m_j W_{default}(\mathbf{r}_i - \mathbf{r}_j, h)$
Felületi merőleges	$\mathbf{n}(\mathbf{r}_i) = \sum_j \frac{m_j}{\rho_j} \nabla W_{default}(\mathbf{r}_i - \mathbf{r}_j, h)$
Nyomás erő	$\mathbf{f}_i^{pressure} = -\rho_i \sum_{j \neq i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) m_j \nabla W_{pressure}(\mathbf{r}_i - \mathbf{r}_j, h)$
Viszkozitás erő	$\mathbf{f}_i^{viscosity} = \mu \sum_{j \neq i} (\mathbf{u}_j - \mathbf{u}_i) \frac{m_j}{\rho_j} \nabla^2 W_{viscosity}(\mathbf{r}_i - \mathbf{r}_j, h)$
Gravitációs erő	$\mathbf{f}_i^{gravity} = \rho_i \mathbf{g}$
Felületi feszültség erő	$\mathbf{f}_i^{surface} = -\sigma \frac{\mathbf{n}_i}{\lVert \mathbf{n}_i \rVert} \sum_j \frac{m_j}{\rho_j} \nabla^2 W_{default}(\mathbf{r}_i - \mathbf{r}_j, h)$

2. képlet: Smoothed-particle hydrodynamics erők és tömegsűrűség[1]

$$p = k(\rho - \rho_0)$$

3. képlet: Általános gáztörvényből származtott összefüggés a ϱ tömegsűrűség és p nyomás között[1]

A folyadékot konstans m tömegű részecskék alkotják. A ϱ tömegsűrűség kiszámítható a diszkretizált tömeg és a közeli részecskék távolságának (r_i-r_j) segítségével. A folyadék sűrűségét több formában is meg lehet adni

részecskerendszerek esetén. A leggyakoribbak a ϱ_0 nyugalmi tömegsűrűség, egy részecske elvárt térfogata, vagy a részecskék közötti elvárt távolság. A konstans részecsketömeg felhasználásával a három mennyiség kifejezhető egymásból. Mi a nyugalmi tömegsűrűséget használtuk a szimuláció felparaméterezésekor. A tömegsűrűség, a választott nyugalmi tömegsűrűség és a szimuláció szempontjából konstans mennyiségek (a folyadék kémiai anyagmennyisége mol-ban, a folyadék abszolút hőmérséklete K-ben és az egyetemes gázállandó) összevonásából keletkezett k konstans segítségével a p nyomás is meghatározható a 3. képleten látható, az általános gáztörvényből levezetett összefüggés felhasználásával. A konstans tömeg, a közeli részecskék távolsága, a származtatott tömegsűrűség és nyomás segítségével a nyomásból fakadó erő kifejezhető. A viszkozitásból fakadó erő a közeli részecskék u sebességének, távolságának és tömegsűrűségének, valamint a folyadék anyagát jellemző μ viszkozitási együtthatónak a segítségével származtatható. A gravitációs erő a g gravitációs együtthatóval a tömegsűrűségből fejezhető ki. Továbbá felületi feszültséget szimuláló erő vezethető be a simább folyadékfelszín elérésének érdekében, ami a közeli részecskék elhelyezkedéséből, tömegsűrűségéből és egy, az anyagot jellemző σ konstansból származtatható. A felsorolt erők összegéből származtatott eredő erő, a tömegsűrűség és a szimulációs ciklusban eltelt idő segítségével meghatározható a gyorsulása, a sebessége és az elmozdulása egy adott részecskének.

A bemutatott számítási lánc több helyen alkalmaz rekonstrukciós kerneleket. A rekonstrukciós kernelek a 2. képleten W karakterrel vannak jelölve. A rekonstrukciós kernelek definiálják a kapcsolatot a diszkrét közelítések és az eredeti folytonos függvények között. A kernel függvény megválasztása befolyásolja a számítás pontosságát és az artifaktumok mértékét. Fontos kiemelni, hogy a kernel függvények a számítások helyességét csak a kernel függvény meghatározásakor alkalmazott nyugalmi tömegsűrűséghez viszonyított véges sűrűségtartományon belül tudják garantálni. Ebből kifolyólag a folyadékszimulációhoz általunk hozzáadott kényszerek nem kényszeríthetik a folyadék sűrűségének megváltozását olyan mértékben, hogy az meghaladja ezen véges tartományt, ellenkező esetben a szimuláció instabillá válhat.

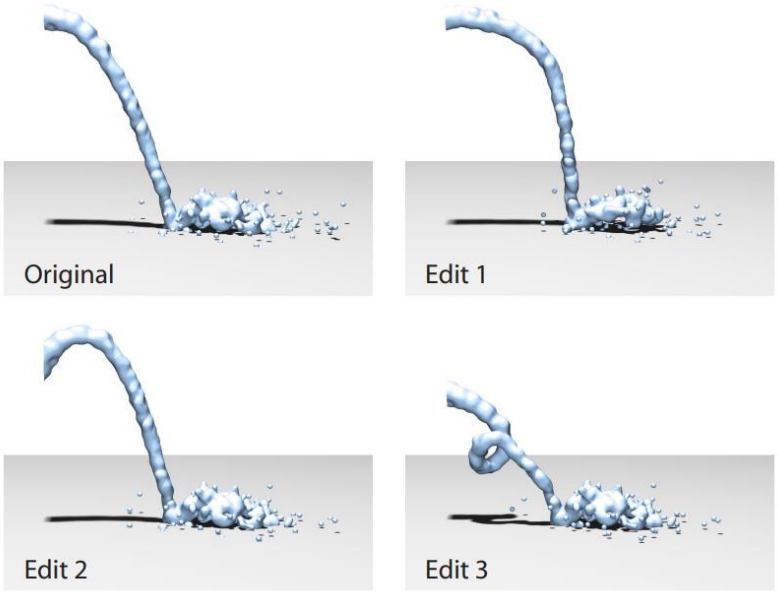
2.2 Folyadékvezérlő eljárások

Számos, már létező, folyadékszimulációt vezérlő eljárást megvizsgáltunk, de mindegyik rendelkezett olyan sajátos megkötésekkel ami megnehezítette vagy nem tette lehetővé a folyadékalapú animált karakterek létrehozását.

2.2.1 Beágyazott deformáció

A beágyazott deformációt[2] Robert W Sumner, Johannes Schmid és Mark Pauly dolgozta ki. A beágyazott deformáció a szimulációs tér egy részhalmazához rendel transzformációt. Amikor egy objektum belép a kiválaszott térrészbe a transzformáció alkalmazva lesz az objektumra. Ha a teljes objektumot nem tartalmazza a kiválaszott térrész akkor a transzformáció alkalmazható kizárólag az objektum érintett részére. A transzformációk és a hozzájuk tartozó térbeli tartományok összetett struktúrákba szervezhetőek, ezáltal bonyolultabb transzformációk is létrehozhatóak. Továbbá létrehozhatóak energiafüggvényhez hasonló súlyfüggvények, amik büntetik, ha az objektum nem felel meg a transzformációnak. A súlyfüggvények minimalizálásával folytonos átmenetet lehet képezni a transzformált és a nem transzformált állapot között. Valamint a súlyfüggvények kombinálásával egymást metsző beágyazott deformációk is létrehozhatóak.

A beágyazott deformáció előnye a mi esetünkben, hogy könnyen alkalmazható részecskerendszerekre, ahogyan azt a beágyazott deformáció szerzői be is mutatták és az 1. ábrán látható. Egy részecskéről sokkal könnyebben eldönthető, mint egy kiterjedt testről, hogy a transzformációhoz tartozó térbeli tartomány tartalmazza-e, ezáltal a beágyazott transzformáció egy egyszerűbb megvalósítását is elég alkalmazni a részecskékre. A hátránya a mi esetünkben a beágyazott transzformáció alkalmazásának, hogy nehéz úgy definiálni a transzformációkat és a tértartományokat, hogy a folyadék a karakter kitöltésére legyen kényszerítve. Továbbá még nehezebb úgy definiálni a beágyazott deformációkat, hogy a karakter kitöltése közben realisztikus folyadékviselkedés hatását keltse.

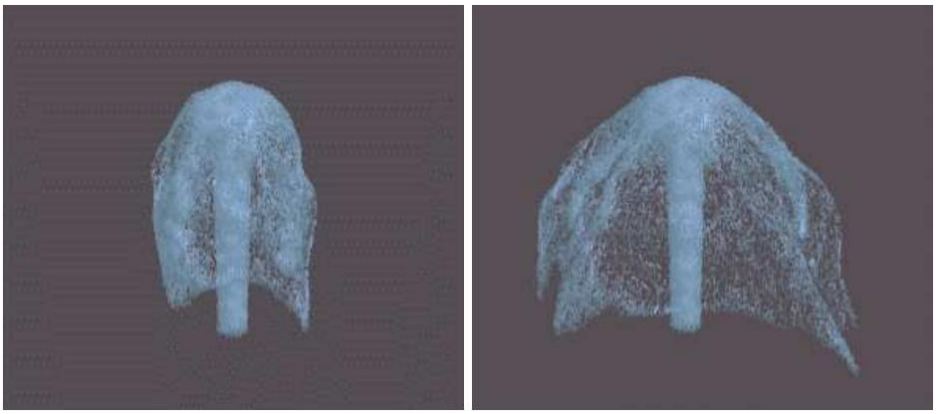


1. ábra: Beágyazott transzformáció alkalmazása részecske rendszeren[2]

2.2.2 Kontrollparaméterek

Nick Foster és Dimitris Metaxas kontrollparamétereket[3] definiálnak a folyadék szimulációjának vezérléséhez. A kontrollparaméterek nem közvetlenül befolyásolják a folyadék viselkedését, hanem a folyadékszimuláció kiértékelésénél vezetnek be új paramétereket, ezáltal közvetetten irányítják a folyadékot. A publikációjukban többek között, a 2. ábrán látható, szökőkút vízsugarán keresztül mutatják be a módszerüket. A kontrollparaméterek hatását a függőleges tengely mentén növelik, ezáltal különböző kontrollparaméter beállításokkal a szökőkút által generált vízsugár kezdeti szakasza hasonló, míg a tetőpont különböző alakú lesz.

Az előző esethez hasonlóan, a kontrollparamétereket nehéz úgy megválasztani, hogy egy tetszőleges adott testet kitöltsön a folyadék miközben realisztikus folyadékszimuláció hatását kelti. Továbbá, a publikációban bemutatott folyadékszimuláció térfelosztáson alapszik, aminek következtében a módszer adaptációja részecskealapú rendszerekre nem triviális.



2. ábra: Kontrollparaméterek alkalmazása a vízsugár manipulálására[3]

2.2.3 Folyékony bőr módszer

A folyékony bőr technikát[4] Mark Wiebe és Ben Houston publikálták. A módszerük alkalmazásával hozták létre a 3. ábrán látható kátrányszörnyet. A folyadék kizárólag csak a karakter felszínén helyezkedik el. A létrehozott karakter külseje, megfelelő anyagbeállítások mellett, hasonlít az általunk előállított karakterekhez, de az alkalmazási lehetőségek különbözőek. A folyékony bőr egy folyamatos áramló effektet céloz meg létrehozni a karakter felületén. Mi ilyen effektet nem biztosítunk, helyette mi a folyadék nyugalmi helyzetben való tartását célozzuk meg, amennyiben a karakter már fel van töltve és a karakter animációját a folyadék már lekövette. Továbbá a folyékony bőr módszer csak dekorálja a karakter felszínét, ezáltal nem nyújt lehetőséget alakváltoztatásra, vagy más objektumokkal történő összetett interakcióra. Szemben a mi módszerünk egy teljesen funkcionális folyadékszimuláción alapszik, ebből kifolyólag a folyadék bármikor átalakítható más karakter alakjává, különböző karakterek összeolvaszthatóak és a karakter alakja befolyásolható a színtérben elhelyezett más objektumok segítségével.

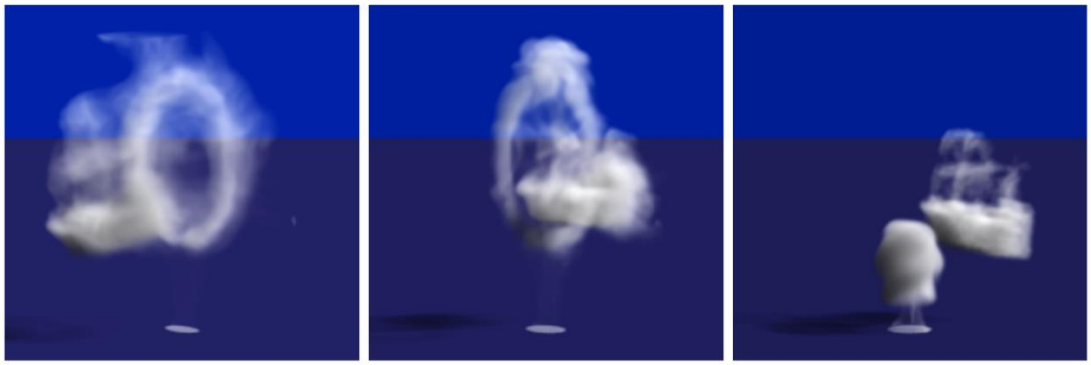
3. ábra: A folyékony bőr technikával létrehozott kátrányszörny[4]

2.2.4 Célvezérelt füstanimáció

Raanan Fattal és Dani Lischinski mutatták be a célvezérelt füstanimációt[5]. A célállapotot egy elvárt térbeli sűrűségeloszlás formájában adják meg. A füst vezérlését két új erő hozzáadásával érték el. A hajtóerő gondoskodik a füst eljuttatásáról a kezdeti állapotból a célállapotba. A gyüjtőerő a füst szétáradását akadályozza meg. A bemutatott módszer térfelosztáson alapuló folyadékszimulációt alkalmaz. Az algoritmus eredménye a 4. ábrán látható. A kevésbé részletgazdag megjelenítést a publikáció létrehozásakor rendelkezésre álló hardver okozza, aminek a számítási kapacitása elmarad a ma már széles körben elérhetőtől.

A 4. ábra alapján is látható, hogy bemutatott módszer célkitűzése és alkalmazási lehetőségei nagyon hasonlítanak a miénkhez, de az eltérő szimulációs módszer és a füst helyettesítése folyadékkal teljesen más megközelítést igényel. A füst helyettesítése folyadékkal és a részecske alapú rendszer alkalmazása csökkenti a gyüjtőerő szükségességét. Valós légüres térben a folyadék próbál összefüggő testet alkotni a vízmolekulák vonzásának és az ebből következő felületi feszültségnek köszönhetően, a gázokkal ellentétben. Ezáltal a valóságot leképező fizikai modell is a segítségünkre lesz. A folyadékszimulációnk során a részecskék közötti elvárt távolság és a felületi feszültséget megvalósító erő formájában van jelen ez a jelenség. Természetesen külső hatások ezt az összetartó erőt le tudják győzni, de általában ez az elvárt viselkedés. A célvezérelt füstanimáció egy elvárt térbeli sűrűségeloszlást vár bemenetként. Ezzel szemben mi kontrollrészecskék elhelyezésével határozzuk meg a célállapotot. A térbeli sűrűségeloszlás alkalmazásával a mi módszerünkhöz képest homogénebb célállapot határozható meg. Egy bemeneti háromszöghálónak megfelelő célállapot mindkét reprezentáció esetében könnyen előállítható, de a kontrollrészecskék alkalmazásakor a háromszögháló animációja felhasználható a kontrollrészecskék animálására is. Ebben az esetben nincs szükség minden animációs lépésben újra elhelyezni a kontrollrészecskéket. Ezen felül a kontrollrészecskék sebessége is könnyen meghatározható az előző pozíció és az eltelt idő alapján, ami a gyors animációk lekövetésében játszik szerepet. Ellenben térbeli sűrűségeloszlást nem triviális animált háromszöghálóhoz kötni, aminek következtében a célzott térbeli sűrűségeloszlást minden animációs frame-ben elő kell állítani, ami csökkenti a szimuláció hatékonyságát. Az elhelyezett kontrollrészecskék az elvárt térbeli sűrűségeloszláshoz hasonló reprezentációt alkotnak, de az egyes kontrollrészecskék csak lokális vonzást

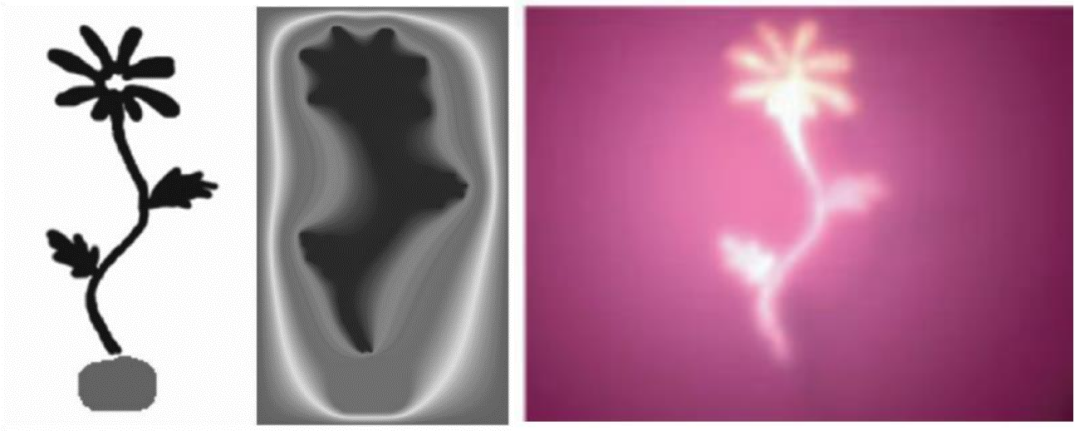
hoznak létre, a globális hajtóerővel ellentétben. Ebből következően a mi módszerünkben alkalmazott kontrollrészecske-struktúrát hatótávolságon belül kell elhelyezni a folyadékhoz képest, ami globális hajtóerő alkalmazásakor nem volna szükséges. A mi módszerünk viszont a lokális vonzások megfelelő növelésével lehetőséget ad a karakterek gyors és célzott kitöltésére, majd a lokális vonzások csökkentése segít stabilan fenntartani a már feltöltött alakot. Végül az általunk alkalmazott részecskealapú rendszer előnyösebb összetett és kiterjedt színterek esetén.



4. ábra: Célvezérelt füstanimáció[5]

2.2.5 Potenciálmező-alapú folyadékszimuláció

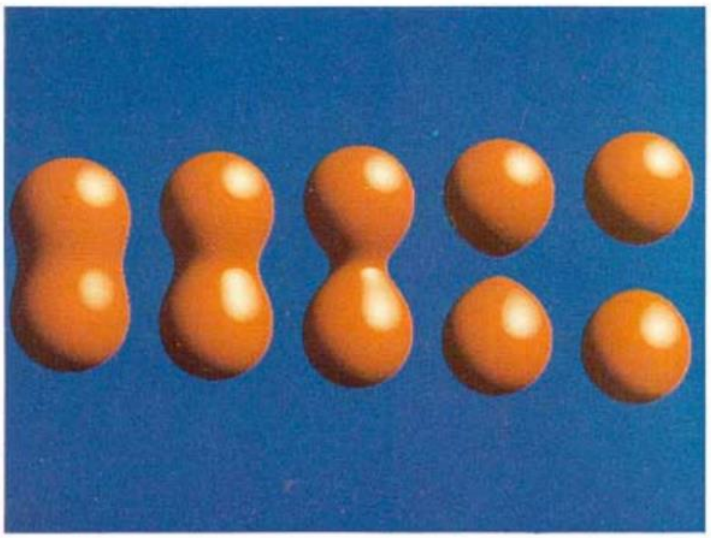
Jeong-mo Hong és Chang-hun Kim egy potenciálmezőn alapuló irányított folyadékszimulációs módszert[6] mutattak be. A potenciálmező előállítására számos lehetőséget ajánlanak, az általunk használt háromszögháló-alapú karaktermodellek is felhasználhatóak a potenciálmező létrehozására. Az 5. ábrán egy kétdimenziós elvárt sűrűségmező, a sűrűségmező kitöltésének érdekében előállított potenciálmező és a közeg vezérelt szimulációja látható. A szimuláció térfelosztáson alapuló folyadékszimulációs módszert alkalmaz, de az általunk alkalmazott részecskealapú szimulációnál is könnyen alkalmazni lehetne a potenciálmező segítségével előállított kényszereket. A problémát, a célvezérelt füstszimulációhoz hasonlóan, a háromdimenziós térfogati mennyiség animációja okozza. Animált háromszögháló esetén a háromszögháló minden animációs lépéséhez külön potenciálmezőt lenne szükséges előállítani, mert a potenciálmező animációja a háromszögháló animációjának felhasználásával nem triviális. Ellenben az általunk alkalmazott kontrollrészecskék animálhatóak a karakteranimáció felhasználásával.



5. ábra: (a) Célállapot, (b) A célállapot eléréséhez előállított potenciál mező, (c) vezérelt közeg[6]

2.3 A folyadékfelület megjelenítése

A 6. ábrán illusztrált, Blinn[7] és Nishimura[8] által bevezetett *metaball* kontrukció olyan implicit felületet ír le, melyet egymásra hatással levő objektumok alkotnak. A *metaball*okat széles körben alkalmazzák szimuláció eredményének megjelenítésére és folyadékok vizualizálására. Minden *metaball* rendelkezik egy sugárirányú sűrűségfüggvénnyel, amelyek térbeli elhelyezkedéstől függő aggregációjával a *metaball*ok együttesen reprezentálják a sűrűségmező izofelületét.



6. ábra: Két metaball vizualizációja egymástól vett különböző távolságok esetén[7]

2.3.1 A metaball matematikai modellje

Az első részecske-alapú implicit felületet J.F. Blinn mutatta be kutatásában[7], melyben a molekulaszerkezetek elektronsűrűség-térképeinek megjelenítését tűzte ki célul. Korábbi lehetőségek a megjelenítendő molekulaszerkezetek közötti, a 6. ábrán látható, kötések nyújtására és szakítására csak korlátosan és megszorításokkal álltak

rendelkezésre. Ebből kifolyólag egy olyan új megközelítés leírása vált szükségessé, amely azóta *metaball* néven vált ismertté és számos számítógépes grafikai alkalmazásban került felhasználásra. A publikációban alkalmazott matematikai modell formájában hasonló az elektronsűrűség-térképek szimulációjának modelljéhez.

A kvantummechanika az atom belsejében szereplő elektront egy térbeli sűrűségfüggvényeként ábrázolja. A publikáció az atomok gyűjteményének összesített sűrűségét az egyes atomok egyenkénti hozzájárulásainak összegzésével ábrázolja. A 4. képleten látható függvény az adott pontban lévő aggregált sűrűséget írja le, ahol *r* a kiértékelt pont távolsága az atom középpontjától.

$$D(x, y, z) = \sum_i b_i \, e^{-\alpha_i r_i^2}$$

4. képlet: J.F. Blinn által definiált sűrűségfüggvény[7]

A képletben szereplő exponenciális kifejezés egy Gauss-felületet ír le, amelynek szórása α és magassága b. Ezen α és b paraméterek módosításával a megjelenítendő objektum „összeolvadása” konfigurálható. Az aggregált sűrűségfüggvények által kifeszített és a küszöbértékkel megegyező izofelület reprezentálja a *metaball*ok által létrehozott felületet. A küszöbérték alkalmazása a 5. képleten látható. Minden pont amely a felületen belül helyezkedik el, *T* küszöbértéknél nagyobb elektronsűrűséggel kell rendelkezzen**.**

$$F(x, y, z) = D(x, y, z) - T$$

5. képlet: Küszöbérték felhasználása az izofelület definiálására[7]

Mivel a J.F. Blinn által alkalmazott, a 4. képleten látható Gauss-függvény nem véges tartójú, az izofelület meghatározásához minden objektumot figyelembe kell venni, ebből kifolyólag alkalmazása jelentős számítási költséggel jár. A Gauss sűrűségfüggvény lecserélése véges tartójú függvényre nagy mértékben csökkentheti a számítási költséget, mert az aggregált sűrűségfüggvény egy pontban való kiértékeléséhez elegendő az adott térbeli pont véges környezetén belüli *metaball*ok hozzájárulását figyelembe venni. Az aggregált sűrűségfüggvényt *N metaball* és *T* sűrűségküszöb esetén azok az *x* pontok határozzák meg, amelyek kielégítik az 6.

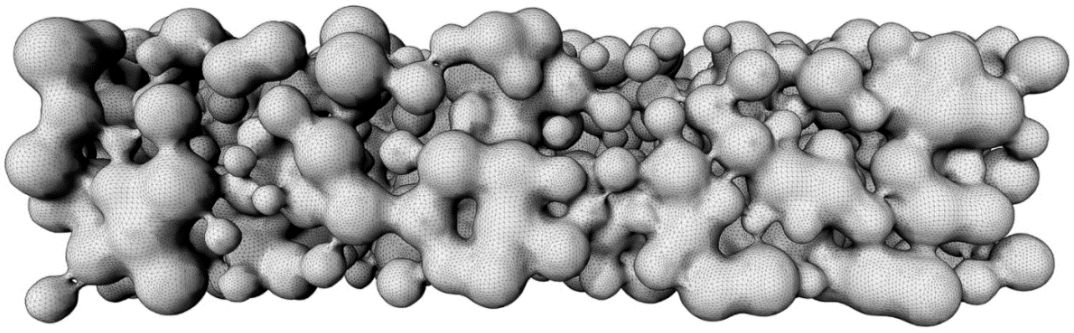
képleten látható függvényt. A képletben szereplő *r* az *x* pont távolsága a *metaball* középpontjától.

$$D(x) = \sum_{i=0}^{N-1} D_i(r_i) - T$$

6. képlet: Aggregált metaball sűrűségfüggvény[7]

2.3.2 Masírozó kockák

A folyadékfelületet, amelyet a *metaball*ok általt előállított izofelület határoz meg, megjeleníthető több különböző módon. Egy széleskörben elterjedt lehetőség a Lorensen és társai által bemutatott *masírozó kockák*[9] algoritmus. Az algoritmus egy tetszőlegesen választott küszöbértéknek megfelelő sűrűséggel rendelkező felület háromszögháló-modelljét állítja elő. Az előállított felület minősége az alkalmazott rács felbontásától függ. Alacsony felbontású rács esetén az algoritmus gyorsan képes előállítani a felületeket, viszont az előállított felületek felbontása alacsony. Nagy felbontású rács alkalmazása esetén az algoritmus részletgazdag felületeket állít elő, de jelentősen megnövekszik a számítási és memóriaköltség. A 7. ábrán látható a *masírozó kockák* algoritmus alkalmazásával előállított *metaball*okat megjelenítő nagy felbontású háromszögháló.

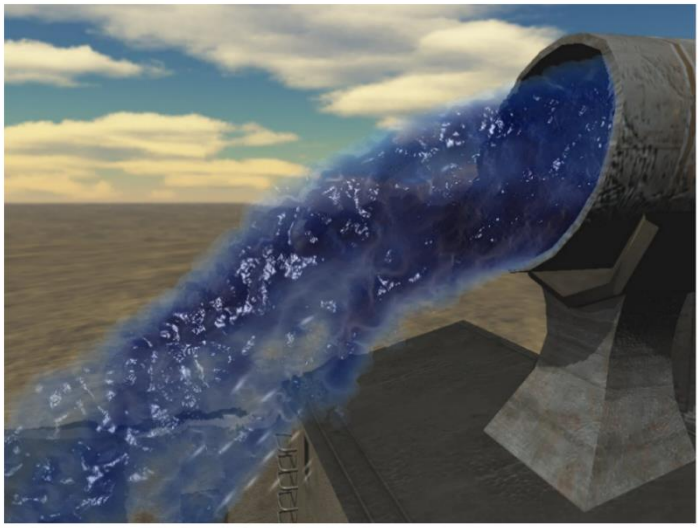


7. ábra: Metaballokon alkalmazott masírozó kockák által előállított részletes háromszögháló[27]

2.3.3 Screen-space vizualizáció

Wladimir J. van der Laan és társai egy *screen-space metaball* vizualizációs módszert[10] mutattak be. A módszer valósidejű teljesítményt nyújt konfigurálható gyorsaság-minőség preferenciával. A zselés megjelenítés elkerülésének érdekében a módszer erőteljes *screen-space* szűrést alkalmaz a felületek kisimítására, ennek

következtében a közeli objektumok megjelenítésére az eljárás alkalmazása nem célszerű. A simított zselés artifaktum látható a 8. ábrán.



8. ábra: Screen-space metaball megjelenítés[10]

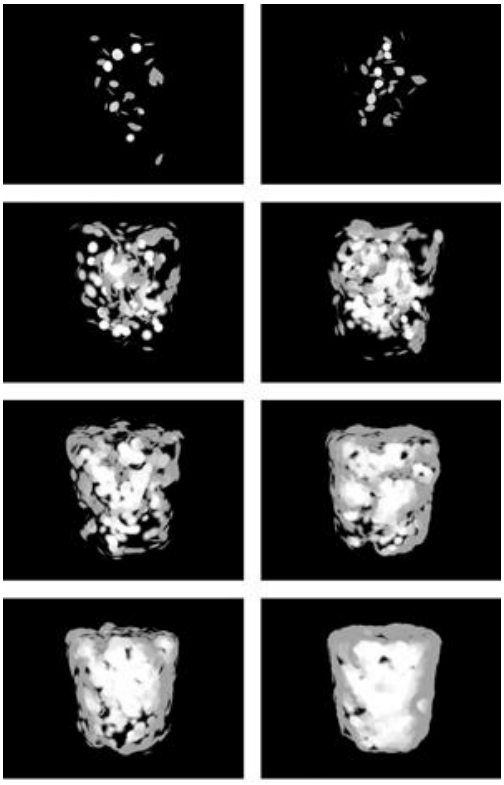
2.3.4 Rács alapú megjelenítés

Iwasaki és társai részecske-alapú folyadékszimulációs eredmények megjelenítésére dolgoztak ki egy új módszert[11]. Felosztották a szimulációs teret egy segédrács bevezetésével és a rácspontokban a grafikus kártya segítségével felgyorsították a sűrűségfüggvény kiértékelését majd meghatározták a víz felszínét reprezentáló izofelületet. A módszer drasztikusan csökkenti a vízfelület rekonstrukciójának és megjelenítésének számítási költségét, ebből kifolyólag a részecskeszimuláció eredménye valós időben előállítható. Azonban a rács alkalmazása miatt látható artifaktumok keletkezhetnek és a szimulációs tér határainak megváltoztatása nem hatékony.

2.3.5 Egyenletes részecskeeloszlással rekonstruált felület

Van Kooten és társai közel egyenletesen elhelyezkedő részecskék[12] segítségével megjelenített *metaball*okkal rekonstruálják a felületet. A részecskék elhelyezkedésének meghatározásához taszító erőket, sebességkényszereket és a részecskék sűrűségét használják fel. A módszer a *masírozó kockák*kal és a sugárkövetésen alapuló algoritmusokkal szemben kevesebb számítási költséggel jár, mert az említett módszerek a folyadék térfogatával arányos számítási költséget eredményeznek, ellenben a Van Kooten és társai által bemutatott módszer a folyadék felületével arányos számítási költséget eredményez. A 9. ábrán látható a módszer alkalmazásával

megjelenített részecskék szétoszlása egy hengerben. Az ábrán kivehetők a felületen keletkező apró hézagok, melyek a módszer fő hátrányát alkotják.



9. ábra: Részecskék hengerben való szétterjedésének bemutatása[12]

2.3.6 Sugárkövetés

Egy további *metaball*-vizualizációs eljárás a Nishita és társai által bemutatott sugárkövetésen alapuló módszer[13]. A módszer segítségével kiváló minőségű képek állíthatóak elő, viszont a sugár és felület közötti metszést meghatározó függvény magas számítási költséggel jár. Ebből kifolyólag nagy mennyiségű *metaball* valósidejű megjelenítésére nem alkalmas.

Loop és Blinn háromváltozós Bézier tetraéderek segítségével definiált algebrai felületeket jelenítenek meg sugárkövetésen alapuló módszerrel[14]. A módszer használata kevés *metaball* megjelenítése esetén gyors, mivel a sugár és felület közötti metszést meghatározó függvényben a tetraéderek csúcspontjait lineárisan interpolálva az együtthatók könnyen kiszámolhatók. Viszont a metszést számító függvényben figyelembe kell venni a szomszédos *metaball*okat is, ebből kifolyólag nagyszámú *metaball* esetén jelentős hatékonyságromlás jelentkezhet.

A valósidejű *metaball* megjelenítő módszerek jellemzően a *ray marching* algoritmuson alapuló eljárásokat alkalmaznak. Ezen módszerek a közelítőleges

metszéspont meghatározásához a szempozícióból kiinduló sugarat lineárisan tesztelik iterálva. Az izofelület meghatározása a *ray marching* algoritmus segítségével tekinthető a sugárkövetés egy speciális esetének. Az eljárás gyorsítására Kanamori és társai[15], illetve Szécsi és Illés[16] mutatnak lehetőségeket.

2.3.7 Ray marching

A dolgozatban prezentált vizualizációs módszerek *ray marching*on alapulnak. Az előállított kép minden pixeléhez tartozik egy szempozícióból indított sugár. Ezen sugarak és a *metaball*ok által előállított implicit felület közötti metszéspontok megtalálásával állítható elő a kimeneti kép. A metszéspont meghatározásához a metszéspont tesztelés a szempozícióból indul, majd a sugár mentén előre meghatározott lépésközönként van a tesztelés megismételve. Az iteráció az izofelület egy belső pontjának megtalálásával, a maximális lépésszám elérésével vagy a jelenetből való kilépéssel zárul. Ha az iteráció egy belső pont megtalálásával zárult, akkor a pontos metszéspont elhelyezkedése további iteratív gyökkereső eljárásokkal pontosítható. Az algoritmus teljesítménye *metaball* geometriák esetében a részecskék számától függ, mert a metszés teszteléséhez minden *metaball* kontribúcióját ki kell értékelni. Továbbá hasonló eljárás szükséges a felületi merőleges meghatározásához egy adott térbeli pontban, ami az árnyalás kiszámításához szükséges.

Amennyiben a *metaball* sugárirányú sűrűségfüggvénye kompakt tartójú, a metszéspont tesztelésében releváns *metaball*ok a tesztelési pont egy véges nagyságú környezetében találhatóak. Ebből kifolyólag a metszésszámításhoz releváns *metaball*ok a *metaball* pozíciója és a *metaball*hoz kötött sűrűségfüggvény hatótávolságának függvényeben előszűrhető. A mi hozzájárulásunk a *metaball* megjelenítés területén a számításokban résztvevő *metaball*ok szűrésére szolgáló lehetőségek összegzése és kiértékelése. Az általunk vizsgált optimalizációs algoritmusok az előállított implicit felületet nem változtatják meg, ugyanolyan nagy részletességű és sima implicit felületet ábrázoló képet állítanak elő, mint az optimalizálatlan változatban. Az alkalmazott optimalizációs algoritmusok hatása csak rendkívül nagy részecskeszámosság esetén mutatkozik meg.

2.4 Optimalizációs lehetőségek

Részecske-alapú folyadékszimuláció valós időben történő megjelenítése érdekében különböző gyorsítási lehetőségeket vizsgáltunk meg, implementáltunk és értékeltünk

ki. Az első két megvizsgált optimalizációs eljárás az *A-Buffer* és az *S-Buffer*. Mindkét eljárás a jelenet renderelése közben képpontonként változó mennyiségű adatot képes eltárolni hatékonyan, ezáltal olyan komplex effektusok megjelenítésében tudnak segítséget nyújtani, mint sorrend független átlátszóság, térfogatmegjelenítés vagy ütközésdetektálás. Esetünkben az előzőektől eltérő két további esetben kerül alkalmazásra. Egyrészt a sugár és a *metaball*ok által előállított izofelület közötti metszéspont tesztelésekor a releváns *metaball*ok képpontonkénti előszűrésében segít, ezáltal nagymértékben növeli a megjelenítés hatékonyságát nagymértékű *metaball* számosság esetén. Másrészt, egyes általunk bemutatott kontrollrészecske-elhelyező eljárások során segít a mélységinformációk eltárolásában, ami ezen esetekben nélkülözhetetlen a videókártyán való implementációhoz.

2.4.1 A-Buffer

Az első optimalizációs eljárás a Carpenter publikációjában leírásra került *A-buffer*[17]. Az *A-Buffer* területátlagolt és akkumulációs bufferként is ismert. Ez volt az első módszer átlátszó, átlátszatlan és egymást metsző objektumok közötti láthatóság feloldására képpontonként összegyűjtött adatok segítségével. A geometria renderelése során a kívánt információ minden képpontra kigyűjtésre kerül egy változó hosszúságú listába. Ezt követően képpontonként kigyűjtött adatok a primitívek renderelési sorrendjétől teljes mértékben független sorrendben feldolgozhatók.

Az *A-Buffer* legnagyobb előnye, hogy a képpontonkénti változó hosszúságú listák egy nagy közös memóriaterületen összegyűjthetőek, ezáltal nincs arra szükség, hogy előre ismerjük a pixelenkénti maximális adatmennyiséget, és a lefoglalt memória teljes mértékben kihasználható. Tehát elegendő a memóriaerőforrás létrehozásakor a láncolt listák együttes maximális méretét meghatározni. Az *A-Buffer* hátránya a képpontonként kigyűjtött adatok nem folytonos elhelyezése a közös memória területen, ami a videókártya cachelési mechanizmusait akadályozza. Az *A-Buffer* számos további, képpontonként adatokat tároló struktúrát ihletett meg, közülük a legjelentősebb az *S-Buffer*.

2.4.2 S-Buffer

Vasilakis és társai által bemutatott *S-buffer*[18] az *A-buffer* egy újabb változata. Az *S-Buffer* az *A-Buffer*rel ellentétben nem alkalmaz láncolt listákat, de az *A-Buffer* létrehozásához szükséges egy darab rendermenet helyett az *S-Buffer* két renderelést

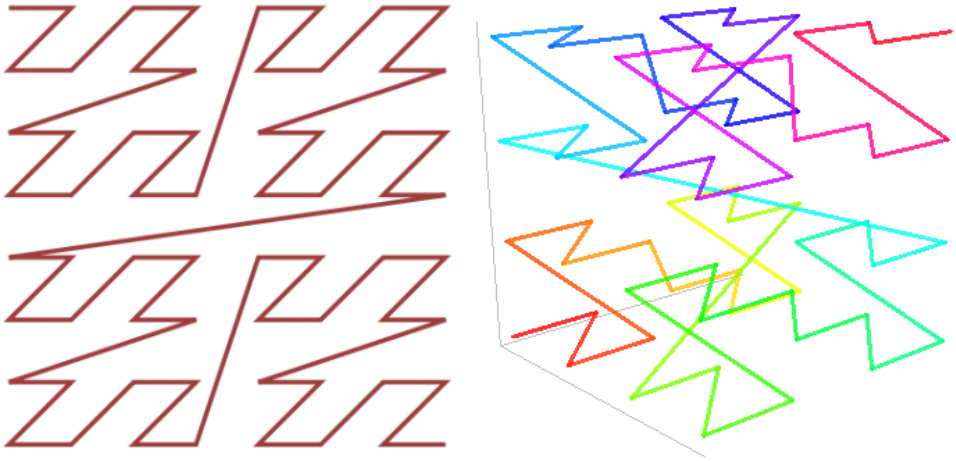
menetet alkalmaz. Mivel az *S-Buffer* láncolt listát és előre meghatározott képpontonkénti maximális adatmennyiséget sem használ, az első renderelési menet a képpontonkénti adatok számát határozza meg. A közös memória-erőforrás folytonos szeleteinek képpontokhoz való hozzárendelése a pixelenkénti adatok számán végzett *prefix sum* végrehajtásával vagy a Vasilakis és társai által alkalmazott párhuzamos randomizált *prefix sum* algoritmus alkalmazásával állítható elő. A második rendermenet a jelenet újrarenderelésével feltölti a kívánt képpontonkénti adatokat az előző lépésben előállított megfelelő méretű memóriaterületre.

Az *S-Buffer* előnye a megnövekedett cache koherencia az *A-Buffer*hez képest, mert az azonos képponthoz tartozó adatok szekvenciálisan helyezkednek el a memóriában. Továbbá az *S-Buffer* hatékonyabban használja fel a rendelkezésre álló memóriát, mert nincsen szükség minden adatelemhez eltárolni a következő adatelem sorszámát. Az *S-Buffer* hátránya az extra rendermenet és a közös memória képpontonkénti felosztásának szükségessége.

2.4.3 Morton rendezés

A folyadékszimuláció egyik legszámításigényesebb feladata a közeli, az adott kiértékelési pontban elvégzett szimulációhoz távolság alapján még hozzájáruló részecskék megtalálása. A közeli részecskék megtalálásához a 10. ábrán látható Morton rendezést[19] alkalmaztuk. A Morton rendezés bitenként felváltva különböző koordinátatengely mentén rendezi a térbeli pontokat, ezáltal egy háromdimenziós térben értelmezett diszkretizált görbe mentén állít sorrendet. Ennek következtében a folytonos háromdimenziós tér egy egydimenziós diszkrét tartományra van levetítve. A Morton rendezéshez szükséges index egy adott pozíció és a szimulációs tért befoglaló *bounding box* alapján gyorsan kiszámítható. Érdemes a Morton görbét és a szimulációs teret befoglaló *bounding box*ot a tengelyek mentén orientálni, mert ebben az esetben a szimulációs tér határai hatékonyan meghatározhatóak a folyadékrészecskéken végzett koordinátánkénti minimum és maximum keresés segítségével. Ezáltal a folyadékszimuláció határainak változása nincs jelentős hatással a teljesítményre, mert a folyadékot befoglaló *bounding box*, a *bounding box*hoz illesztett Morton görbe és ezt követően a részecske pozíciókhoz tartozó Morton index is hatékonyan újraszámolható. A Morton indexek ütközése nem gyakori, ha a hardver által megengedett maximális 32 bites mélység ki van használva. Összetettebb módszerre az index ütközések elkerülésének érdekében nincsen szükség, mivel a keresés minőségét nem befolyásolja

és kevés ütközés esetén a teljesítménybeli különbség is elhanyagolható. Továbbá a részecskék rendezése hatékonyan párhuzamosítható az *odd-even* sorrendező algoritmus alkalmazásával a videókártyán.



10. ábra: (a) 2D Morton görbe (b) 3D Morton görbe[28]

Egy adott kiértékelési pozíció és az alkalmazott kernel függvény hatótávolsága egy gömböt definiál a térben. Az így kapott gömböt befoglaló és a koordintátarendszer tengelyeihez igazított *bounding box* két sarka meghatározza a *bounding box*on belül található összes pont Morton indexének maximumát és minimumát. A Morton görbe orientációjától függ, hogy mely két sarokban található meg a minimum és maximum Morton index. Ezt követően a kiértékelési ponthoz hatótávolságon belül elhelyezkedő részecskéket egy egydimenziós tömb folytonos résztartományán belül kell csak keresni.

3 Mesterséges folyadékmanipuláció

A realisztikus folyadékszimulációt kiegészítő mesterséges kényszereket kontrollrészecskék bevezetésével érjük el. Ezen kontrollrészecskék a célvezérelt füstanimációhoz hasonló hajtóerőt keltenek melynek hatására a kontrollrészecskék, megfelelő felparamatérezés mellett, vonzák a folyadék részecskéket. A mesterséges folyadékmanipuláció létrehozása három fő részre oszlik:

1. Adott kontrollrészecske-elhelyezés mellett a hajtóerő létrehozása.
2. A kontrollrészecskék megfelelő elhelyezése és animálása a kívánt hatás elérésének érdekében.
3. A valósidejűség eléréséhez szükséges optimalizációk alkalmazása.

3.1 Hajtóerő

Az általunk alkalmazott hajtóerő a realisztikus szimuláció alapját képező nyomásból fakadó erőhöz hasonlít. A nyomás-erő számításakor a folyadékrészecskék keltik a kényszert és a folyadékrészecskére van hatással a keltett kényszer. Ezzel szemben a hajtóerőt a kontrollrészecskék keltik, de a folyadékrészecskékre van hatással. A folyadékszimuláció a kontrollrészecskéket nem animálja. A kontrollrészecskék a folyadékszimuláció szemszögéből külsőleg vannak vezérelve. A nyomásból fakadó erő az egyes folyadékrészecskék pozíciójában meghatározott nyomás alapján származtatható. A hajtóerő esetében ez a mennyiség egy tetszőlegesen választott érték lehet, amit mi a kontrollrészecske *kontrollnyomás*aként hivatkozunk. A hajtóerő képlete a 7. képleten látható. A c_{cp} a *kontrollnyomás*t, a W az alkalmazott rekonstukciós kernelt, a ϱ a tömegsűrűséget jelöli. A rekonstrukciós kernel első paramétere a kontrollrészecske és folyadékrészecske távolsága, a második paramétere a rekonstrukciós kernel hatótávolsága. A rekonstrukciós kernel hatótávolsága egyenlő a hajtóerő hatótávolságával.

A tömegsűrűséggel való arányosság éri el, hogy a hajtóerő ne az egyedülálló, például lecsöppenő részecskéket vonzza, hanem a nagy folyadéktömegeken kényszerítsen ki olyan áramlatot, amely képes feltölteni a karaktert. A tömegsűrűségtől független hajtóerő alkalmazása esetén gyakori jelenség a lecseppenő folyadékrészecskék elhajítása, ha pont az animált kontrollrészecskék hatótávolságának határa mentén esik le. Az egyedülálló vagy kis csoportot alkotó folyadékrészecskék

tömegsűrűsége alacsony, ezáltal az általunk javasolt hajtóerő a nagyobb csoportokat alkotó folyadékrészecskéket vonzza intenzíven.

$$f_i^{driving} = c_{cp} \sum_j \rho_j \nabla W_{pressure}(r_i - r_j, h)$$

7. képlet: Hajtóerő

3.1.1 Kontrollnyomás

A *kontrollnyomás* határozza meg a kontrollrészecske vonzásának amplitudóját. A megfelelően megválasztott *kontrollnyomás* egy mesterégesen előállított alacsony nyomású térfogatot hoz létre amely vonzza a folyadékrészecskéket a nyomáskiegyenlítés elérésének érdekében. Ha a megválasztott *kontrollnyomás* túl alacsony, a hajtóerő nem lesz képes legyőzni a külső kényszereket, a legegyszerűbb esetben a gravitációt. Ellenben, ha a *kontrollnyomás* túl magas, a folyadékszimuláció instabillá tud válni. Túl magas *kontrollnyomás* megnöveli a folyadék sűrűségét, ezáltal a manipulált folyadék sűrűsége nagy mértékben el tud térni a realisztikus szimuláció felparaméterezésekor megadott nyugalmi tömegsűrűségtől. A realisztikus folyadékszimuláció az elvárt nyugalmi tömegsűrűség függvényében meghatározott rekonstrukciós kernel alkalmazásával éri el a helyes szimulációs számításokat. Viszont a túlzottan megnövekedett tömegsűrűség esetén a kernelfüggvények már nem tudják biztosítani a szimuláció helyességét.

A hajtóerőhöz ugyanazt a rekonstrukciós kernel függvényt alkalmaztuk mint a nyomásból fakadó erőhöz, mivel a két jelenség hasonló karakterisztikájú. A hajtóerő esetében a kernel függvény hatótávolsága megnövelhető. Ebben az esetben a folyadék által kitöltött térfogat felületén a felhasznált karakter-mesh részletei jobban el lesznek mosva, de a kontrollrészecske-struktúra messzebbről is képes interakcióba lépni a folyadékrészecskékkel. Továbbá fontos megemlíteni, hogy a hajtóerő erősségét nem csak a megválasztott *kontrollnyomás*, hanem a kernel függvény hatótávolságának megváltoztatása és a kontrollrészecskék elhelyezésének sűrűsége is befolyásolja, mert az eredő hajtóerő a kontrollrészecskék által keltett hajtóerők aggregációjaként áll elő. Ebből kifolyólag a *kontrollnyomás*t, a hajtóerő kiszámításához alkalmazott rekonstrukciós kernel hatótávolságát és a kontrollrészecskék sűrűségét együttesen kell szabályozni, elérve, hogy a folyadék sűrűsége a hajtóerő alkalmazása után is az elvárt sűrűségtartományon belül maradjon.

3.1.2 Adaptív kontrollnyomás

A 7. képleten látható hajtóerő szűk keresztmetszeteken keresztül lassan tölti fel a karaktereket. A karakteren belül található folyadékrészecskéket körülvevő kontrollrészecskék kiolthatják egymás hatását, ebből kifolyólag nem közvetlenül a hajtóerő hatásának következtében töltődik fel a karakter folyadékkal. A folyadékrészecskék a hajtóerő által megnövekedett tömegsűrűségnek köszönhetően taszítják egymást a karakter belsejében. Ennek ellenére a folyadékrészecskék nem hagyják el nagy számban a karaktert, mert a karakter határán nem olthatják ki egymás hatását a kontrollrészecskék, mivel a határon csak a karakter belsejének irányába található kontrollrészecske. Ezáltal a folyadékrészecskék lassan, de haladnak a karakter belsejében az üres térfogat felé. Szűk keresztmetszet esetén viszont a karakter határai közel helyezkednek el egymáshoz, ebből kifolyólag a folyadékrészecskéket a karakter belsejében tartó, az áramlás elvárt irányára merőleges hatás akadályozza a megnövekedett tömegsűrűségből következő áramlást.

A jelenség elkerülhető, ha a kontrollrészecskék egymás hatásának kioltása helyett az áramlás elvárt irányába kényszerítik a folyadékrészecskéket. Ezt a hatást az *adaptív kontrollnyomás* bevezetésével értük el, ami a hajtóerőt kontrollrészecskénként skálázza. Egy kontrollrészecske *adaptív kontrollnyomás*a a kontrollrészecske pozíciójában kiértékelt, de a folyadékrészecskéken értelmezett tömegsűrűség függvényében adható meg. Mi egy nulladrendű, egyszerű függőséget definiáltunk, de így is elértük a kívánt hatást.

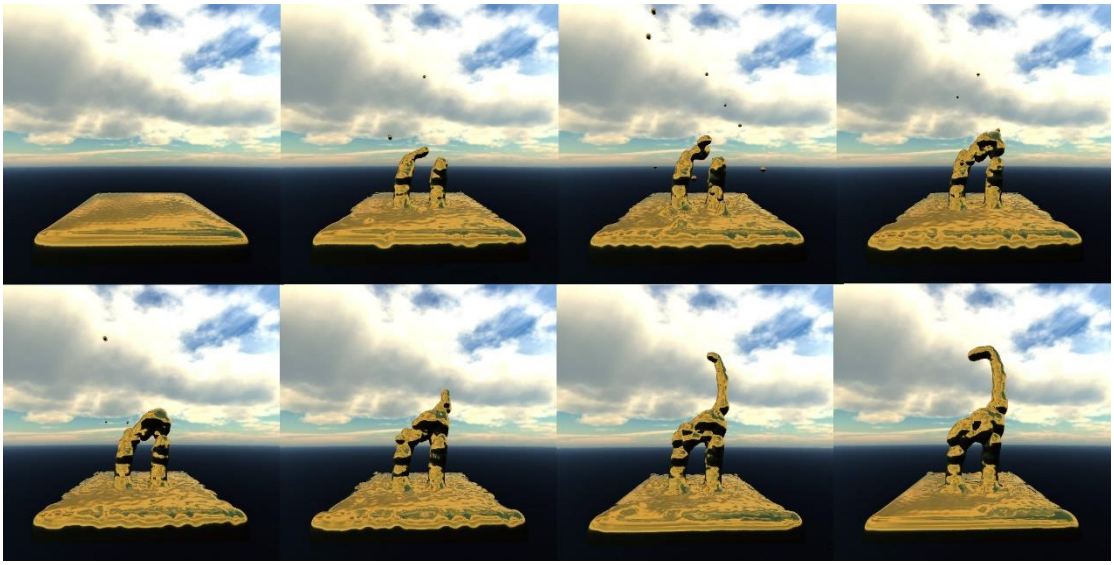
Ha a kontrollrészecske pozíciójában a folyadék tömegsűrűsége kisebb, mint a nyugalmi tömegsűrűség 80%-a, akkor a kontrollrészecske által keltett hajtóerő amplitudóját az *adaptív kontrollnyomás* megduplázza. Ha a vizsgált pontban a folyadék tömegsűrűsége meghaladja a nyugalmi tömegsűrűség 120%-át, akkor a kontrollrészecske hajtóerejét az *adaptív kontrollnyomás* megfelezi. 80% és 120% között a hajtóerő változatlan.

3.1.3 Az adaptív kontrollnyomás alkalmazása

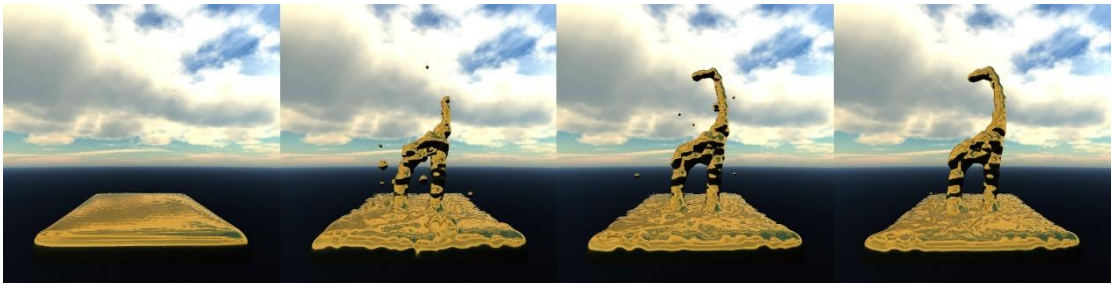
A 11. ábrán látható egy felhasznált háromszögháló az irányított folyadékra helyezve. Jól látható, hogy a zsiráf nyaka szűk keresztmetszetet képez egy hosszú szakaszon, ezáltal kiváló tesztmodell az *adaptív kontrollnyomás* tesztelésére. A 12. ábrán látható a folyadék szimulációja *adaptív kontrollnyomás* alkalmazása nélkül sorban 0, 10, 20, 30, 40, 60, 120 és 180 másodperc elteltével. Látható, hogy az alakzat

feltöltése az elvártaknak megfelelően a zsiráf nyakánál akad meg. A 13. ábrán látható a folyadék szimulációja az *adaptív kontrollnyomás* alkalmazásával sorban 0, 10, 20 és 30 másodperc elteltével. A szimuláció ebben az esetben is megakad egy rövidebb időre a zsiráf nyakánál, mivel a zsiráf nyaki szakaszában a nyugalmi tömegsűrűség függvényében egymás mellett mindössze 2-3 folyadékrészecske fér el. Ennek ellenére a második példán hatszor gyorsabban töltődött fel az alakzat.

11. ábra: Mesterségesen irányított arany színű folyadék a célállapotot meghatározó piros háromszöghálóval



12. ábra: Mesterségesen irányított folyadékszimuláció adaptív kontrollnyomás alkalmazása nélkül



13. ábra: Mesterségesen irányított folyadékszimuláció adaptív kontrollnyomás alkalmazásával

3.2 Kontrollrészecskék elhelyezése

Az előző fejezetben bemutatott hajtóerőt kontrollrészecskék elhelyezésével lehet előállítani. Mivel a kontrollrészecskék vonzzák a folyadékot, a kapott karakter-mesh folyadékkal való kitöltéséhez elegendő a karakter belsejét kontrollrészecskékkel kitölteni és az így kapott kontrollrészecske-struktúrát a folyadék közvetlen közelében elhelyezni. A karakter megfelelő elhelyezéséről a módszer felhasználójának kell gondoskodni. Ha a kitölteni kívánt karakter túl messze helyezkedik el a folyadéktól, akkor nem jön létre az interakció. A karakter-mesh kontrollrészecskékkel való kitöltésére több lehetőséget is bemutatunk. Mindegyik feltöltési lehetőség esetében a kontrollrészecskék térbeli eloszlásának egyenletesnek kell lenni. Ellenkező esetben a folyadék egy deformált alakot fog felvenni, mert a sűrűbben elhelyezett kontrollrészecskék több folyadékrészecskét képesek megkötni mint a kontrollrészecskékkel ritkábban feltöltött térfogatok.

3.2.1 Vertex pozíciók felhasználása

A legegyszerűbb megoldás a kapott karaktermodell vertex pozíciójaira helyezni a kontrollrészecskéket. Ebben az esetben nincsen szükség komplex előfeldolgozásra a kontrollrészecske pozíciók előállításához és a karakter animációját is egy az egyben át lehet ültetni a vertexekről a kontrollrészecskékre. Az így kapott kontrollrészecske-struktúra kizárólag a kapott karaktermodell felszínéhez rendel kontrollrészecskéket, ebből kifolyólag a hajtóerő csak a karakter felszínét fogja feltölteni folyadékkal. A legnagyobb hátránya ennek a megközelítésnek a kontrollrészecskék sűrűségének a kapott mesh vertexsűrűségéhez kötése. Általános karaktermodellek esetében a vertexek sűrűsége számos esetben nem egyenletes, mert a karakter fontosabb részleteit nagyobb felbontásban kerülnek kidolgozásra. Továbbá a vertexek térbeli eloszlásának nem elegendő csak egyenletesnek lenni, mert a kontrollrészecske-struktúra eloszlása a vertexek eloszlásához van hozzárendelve és a *kontrollnyomás,* valamint a hajtóerőhöz tartozó rekonstrukciós kernel hatótávolságának megválasztásakor figyelemmel kell lenni a kontrollrészecskék sűrűségére. Szélsőségesen sűrű vagy ritka háromszöghálók esetében a kényszerek beállítása a folyadékszimuláció átparaméterezése nélkül akár lehetetlen is lehet.

Összefoglalva, a kontrollrészecskék vertex pozíciókra helyezése egy jó megközelítés lehet, mert nincs szükség előfeldolgozásra. De ebben az esetben a

karaktermodelleket előre meghatározott, a folyadékszimulációban alkalmazott paraméterekkel kompatibilis vertex sűrűséggel érdemes előállítani, mert fordított esetben a folyadékszimulációban alkalmazott paraméterek hangolása a karaktermodellek vertex sűrűségéhez nem triviális. A mi tesztjeinkben ez a módszer hatékonyan működött gondosan megtervezett karaktermodellek esetén, viszont elvesztettük az egyszerű integrálhatóság kritériumát tetszőleges karaktermodellek felhasználásakor.

3.2.2 Véletlenszerű elhelyezés

Egy másik megközelítés véletlenszerű térbeli pontok generálása, majd ezen pontok tesztelése. Ha a generált pont a karaktermodellen kívül helyezkedik el, akkor nem helyezünk a pozícióra kontrollrészecskét. A belső pont tesztelésére elegendő egy félegyenest indítani a generált pontból tetszőleges irányba, majd ezt követően megszámolni a háromszögháló és a félegyenes metszéspontjainak számát. A generált pozíció pontosan akkor található a háromszögháló belsejében, ha a metszéspontok száma páratlan. A félegyenes háromszöggel való metszése nagy háromszöghálók esetében számításigényes feladat, ebből kifolyólag a részecskék elhelyezését a szimuláció megkezdése előtt érdemes elvégezni.

A metszéspontok számítása gyorsítható térpartícionáló szerkezetekkel, melyek előszűrik a potenciálisan metsző háromszögeket. A metszéspontok gyors megtalálására egy másik lehetőség a háromszögháló kirenderelése a kamerát a háromszöghálón kívül elhelyezve és az összes felületelem mélységének eltárolásával. Ebben az esetben az eltárolt mélységadatok segítségével rekonstruálhatók a szemből indított sugár és a háromszögháló metszéspontjai. A generált térbeli ponthoz tartozó metszések száma a közeli render sugarakhoz tartozó, de kizárólag vagy a generált pont és a szem között vagy a generált ponttól a szempozicóval ellentétes irányban található, metszések számával közelíthető.

Nem animált modellek esetében a szimuláció megkezdése előtti kontrollrészecske-elhelyezés elégséges. Ellenben animált karaktermodellek esetében ajánlott a kontrollrészecskéket is animálni, ezzel elkerülve a mesh újbóli feltöltésének szükségességét, mert ellenkező esetben nagy háromszöghálók esetén a valósidejű szimuláció az optimalizációk alkalmazásával sem garantálható.

A megközelítés előnye a randomgenerálás hangolásával biztosítható egyenletes kitöltés és a megfelelő kontrollrészecske sűrűség, valamint a kontrollrészecskék ebben

az esetben nem csak a háromszögháló felületén helyezkednek el. A hátránya a magas előfeldolgozási költség, ami nem animált modellek esetében fennálló egyszeri feltöltés esetében és animált modellek esetén animált kontrollrészecskék alkalmazásával együttesen ad valósidejű teljesítményt.

3.2.3 Feltöltés párhuzamos sugarak mentén

Renderelés segítségével hatékonyan szürhetőek a mesh-en kívül elhelyezkedő pontok, de megfelelő randomgenerálást nem triviális implementálni a videókártyán, ebből kifolyólag célszerű a random számokat processzor oldalon előállítani. Processzor oldalon a random generálás potenciálisan lassú és nehezen meghatározható, hogy hány pontot kell generálni adott számú kontrollrészecske előállításához. A megoldás a randomgenerálás helyettesítése. A véletlenszerű elhelyezés renderrel történő megvalósításához hasonlóan ebben az esetben is szükséges kirenderelni a háromszöghálót kívülről az összes mélységinformáció eltárolásával. Majd ezt követően a render sugarak mentén a mélységinformációk segítségével rekonstruált és sorbarendezett metszéspontok között, ahol a két metszéspont közötti szakasz a meshen belül található, egységes lépésközönként kontrollrészecskék kerülnek elhelyezésre. A háromszögháló kirenderelésénél merőleges vetítést biztosítja az egyenletes kontrollrészecske elosztást. A kontrollrészecskék sűrűsége a render felbontásának és a lépésenként elhelyezett kontrollrészecskék távolságának változtatásával állítható be.

Az eljárás előnye a random generáláson alapuló megközelítéssel szemben a nagyobb teljesítménynek köszönhető frame-enkénti megismételhetőség. Ebből kifolyólag nem szükséges animált háromszögháló esetén sem animálni a kontrollrészecskéket. A kontrollrészecskék animálásának megspórolása olyan esetben lehet hasznos, amikor a karakter animációját nehéz a belső kontrollrészecskékhez rendelni. Továbbá a kontrollrészecskék ebben az esetben is kitöltik a háromszögháló belsejét.

3.2.4 Belső kontrollrészecskék animálása

A csontvázalapú karakteranimáció feltétele, hogy a vertexek megfelelő súlyokkal a csontváz elemeihez legyenek rendelve. Ha a kontrollrészecskéket is sikerül hozzárendelni megfelelő súlyokkal a csontváz elemeihez, akkor a karakteranimáció alkalmazható a kontrollrészecskéken is. A vertexek csontváz elemekhez való rendelését és a hozzárendelések súlyozását nem triviális procedurálisan előállítani, ebből

kifolyólag nem várunk el tökéletes eredményt a kontrollrészecskék animálásakor. Mivel a folyadék mozgása és a folyadékot megjelenítő *metaball*ok elmossák az eredeti karakter alakját a kontrollrészecskék animációja során keletkező artifaktumok kevésbé észrevehetőek.

A kontrollrészecskéket a csontváz elemeitől vett távolság alapján rendeltük a két legközelebbi csontvázelemhez. Kizárólag egy elemhez rendelés esetén az animáció töredezett hatást kelt az ízületek mentén. Kettőnél több elemhez rendelés esetén a kontrollrészecskék teljesen független csontvázelemekhez is hozzá lettek rendelve a közeli végtagoknál (például lábak), ennek következtében a kontrollrészecskék az animáció során nagy mértékben elhagyták a karakter belsejét. A hozzárendelt elemek súlyozásához a 8. és 9. képleten látható súlyfüggvényeket teszteltük. A *w* jelöli a hozzárendelés súlyozását. Az *n* az egy kontrollrészecskékhez rendelt csontváz elemek számát jelöli, ami a mi esetünkben kettővel egyenlő. A *d* a csontvázelem távolságát jelöli a kontrollrészecskétől.

A karakter a kontrollrészecskék csontvázhoz rendelésekor animálás nélküli alap pozícióba van állítva, a rossz csontvázelemhez rendelés valószínűségének minimalizálásának érdekében. Ezt követően a karakter feltölthető kontrollrészecskékkel, majd a kontrollrészecskékhez meghatározható a csontvázhoz rendelés. A kontrollrészecskék ezen kezdeti elhelyezése elmentésre kerül a részecskékben, mert a későbbi csontvázanimáció a vertexekhez hasonlóan ezt az animálatlan kiindulási pozíciót használja viszonyítási alapnak.

3.2.5 Az animált kontrollrészecskék alkalmazása

A 14. ábrán látható a szimuláció framenként újra elhelyezett kontrollrészecskék alkalmazásakor. A képen sorban a vezérelt folyadék Phong árnyalással, a folyadékra illesztett célállapot, a kontrollrészecskék véletlenszerű színnel megjelenítve, illetve a kontrollrészecske-struktúrára illesztett célállapot látható. Ebben az esetben a kontrollrészecskék az elvártnak megfelelően csak a karakter belsejében helyezkednek el. A célállapotot reprezentáló háromszögháló a folyadék közepén helyezkedik el és majdnem teljesen fedésben van a kontrollrészecske-struktúrával. Az animált kontrollrészecskék értékelése során ez az állapot tekinthető referenciának.

A 15. és 16. ábrákon animált kontrollrészecskék alkalmazásával van vezérelve a folyadék. A 15. ábrán a 8. képleten található a 16. ábrán a 9. képleten található súlyfüggvény van alkalmazva a kontrollrészecskék csontvázelemekhez való

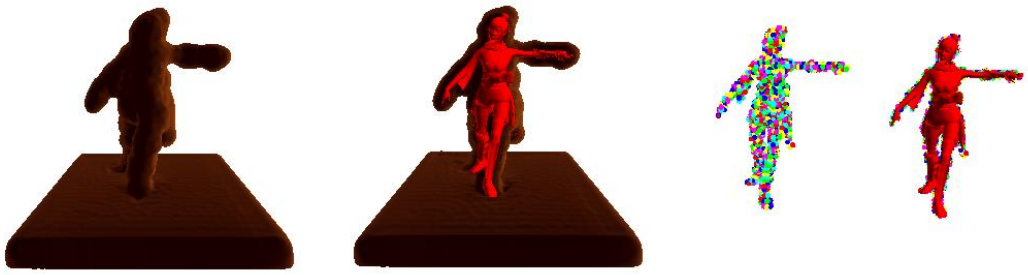
rendelésekor. A két eset majdnem azonos eredményt ad. Mindkét esetben a karakter mozgása teljes mértékben felismerhető a vezérelt folyadékon, tehát az animáció sikeres. Egyes részleteken a távolság alapon történő csontvázhoz rendelés artifaktumokat eredményez. A karakter bal könyökénél és jobb lábszáránál a ruha nem a legközelebbi csontvázelemhez van legnagyobb súllyal rendelve az eredeti vertex súlyozás szerint, ezért a mi közelítésünk itt hibás, ebből kifolyólag a célállapotot reprezentáló háromszögháló nem fedi a kontrollrészecske-struktúrát. Továbbá a karakter lábfejénél is hasonló jelenség figyelhető meg.

$$w_i = \frac{\sum_{j \neq i}^{n} d_j}{n * \sum_{j}^{n} d_j}$$

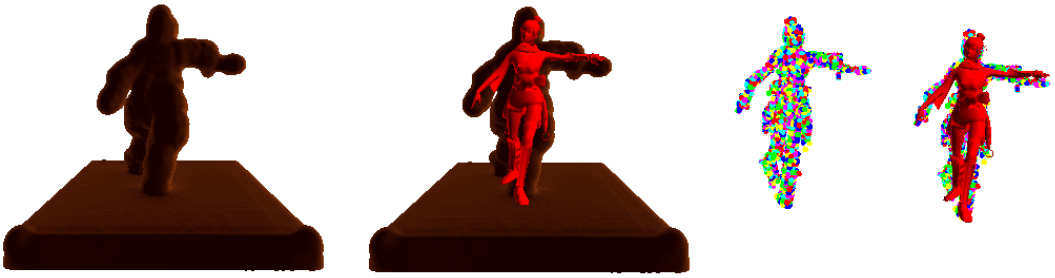
képlet 8: Kontrolrészecskék súlyfüggvénye a csontvázhoz kötéskor

$$w_i = \frac{\frac{1}{d_i}}{\sum_{j}^{n} \frac{1}{d_j}}$$

képlet 9: Kontrolrészecskék súlyfüggvénye a csontvázhoz kötéskor



14. ábra: Framenként elhelyezett kontollrészecskék esetében: (a) Vezérelt Folyadék, (b) Folyadékra illesztett célállapot, (c) Kontrollrészecskék, (d) Kontrollrészecskékre illesztett célállapot



15. ábra: A 8. képleten szereplő súlyfüggvény alkalmazásával animált kontrollrészecskék esetében: (a) Vezérelt Folyadék, (b) Folyadékra illesztett célállapot, (c) Kontrollrészecskék, (d) Kontrollrészecskékre illesztett célállapot

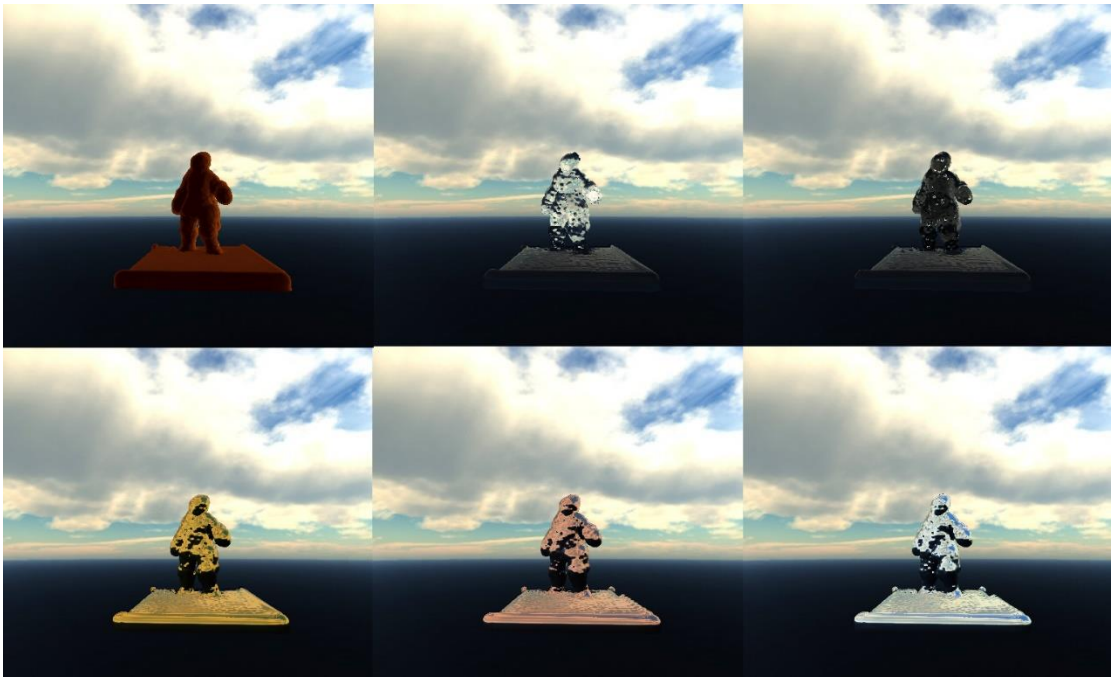
16. ábra: A 9. képleten szereplő súlyfüggvény alkalmazásával animált kontrollrészecskék esetében: (a) Vezérelt Folyadék, (b) Folyadékra illesztett célállapot, (c) Kontrollrészecskék, (d) Kontrollrészecskékre illesztett célállapot

4 Megjelenítés

A folyadék megjelenítése a folyadékrészecskéknek megfeleltetett *metaball*ok által meghatározott implicit felület *ray marching* alapú rekonstrukciójával valósul meg. A folyadékfelületet több különböző *metaball* sűrűségfüggvény és *metaball* küszöbérték felhasználásával is előállítottuk. Általánosságban nagyobb hatótávolságú sűrűségfüggvény alkalmazásakor a megjelenített felszín simább, kevésbé részletgazdag és a megjelenítés költségesebb a megnövekedett képpontonkénti releváns *metaball*ok számossága miatt. Kisebb hatótávolságú sűrűségfüggvény alkalmazása esetén a megjelenítés gyorsabb, mert az előszűrt releváns *metaball* listák kisebb méretűek, de a részletesebben kirajzolódó karakter modell mellett a *metaball*ok körvonalai is felerősödnek.

4.1 Folyadékalapú anyagok megjelenítése

A 17. ábrán látható a hat különböző megvizsgált megjelenítési módszer. A megvalósításra került anyagok megjelenítés alapján két csoportra oszthatók. Az átlátszatlan anyagok megjelenítésekor elegendő az adott képpontból látható első felületi metszéspontot megtalálni. A megtalált metszéspontban a felületi merőleges kiszámításával a képpont árnyalása meghatározható. Ellenben az átlátszó anyagok megjelenítéséhez rekurzív sugárkövetést alkalmaztunk, melynek következtében a *metaball*ok által kifeszített implicit felület metszéspontjának megtalálásakor a sugárkövetés a visszaverődési és a törési irányban is folytatódik.



17. ábra: Implementált megjelenítési módok
(a) Phong árnyalással előállított felület, (b) Átlátszó megjelenítés, (c) Homogén elnyelést szimuláló részben átlátszó megjelenítés, (d) Arany (e) Réz (f) Alumínium

4.1.1 Átlátszatlan megjelenítés

A 17. ábrán látható első megjelenítést a kátrány szörny[4] ihlette és Phong[20] árnyalást alkalmaztunk. Az alsó sorban látható három fém stílusú megjelenítés (arany, alumínium, réz) a Szirmay-Kalos László által bemutatott Fresnel függvény közelítésen[21] alapszik.

4.1.2 Átlátszó megjelenítés

A 17. ábrán látható második, átlátszó anyag megjelenítéséhez rekurzív sugárkövetést alkalmaztunk, melynek következtében a metszéspont megtalálásakor a sugárkövetés a visszaverődési és a törési irányban is folytatódik. A rekurzió véget ér, ha a sugárkövetés kilép a szimulációs térből vagy a rekurzió mélysége elér egy meghatározott küszöbértéket. Mindkét esetben kiolvasásra kerül a hátteret megjelenítő *skybox* a sugárkövetés aktuális iránya szerint. A Fresnel-egyenletek[22] által meghatározott visszaverődési és törési együtthatók, valamint az anyagi jellemzők felhasználásával a kiolvasott *skybox* értékek kompozíciója előállítja a képpont árnyalását, mert ebben az esetben nincsen elnyelődés.

A 17. ábrán látható harmadik, csak részben átlátszó folyadék térben és színcsatornánként is homogén elnyelést szimulál. Mivel a metszéspontok közötti belső

utak hossza könnyedén meghatározható és az elnyelődés homogén a folyadék belsejében az egyszerűsített Beer-Lambert törvény[23] alkalmazásával a folyadék által elnyelt fény mértéke könnyen származtatható.

4.2 Releváns metaballok előszűrése

Mivel kizárólag véges tartójú *metaball* sűrűségfüggvényeket alkalmazunk, az implicit felület kiértékeléséhez elegendő az adott pont egy véges környezetében található *metaball*ok hozzájárulását figyelembe venni. A releváns *metaball*ok előszűrését *billboard*ok bevezetésével érjük el. Minden részecske pozícióban egy *billboard* kerül elhelyezésre, melynek mérete a *metaball* sűrűségfüggvény hatótávolságával megegyező. Ezt követően a *billboard*ok kirenderelésekor, a 2.4.1 és 2.4.2 fejezetekben említett képpontonkénti változó mennyiségű információ eltárolására alkalmazható adatszerkezet segítségével, minden képpontra előállítjuk a képponthoz tartozó sugár menti metszéspontokban potenciálisan résztvevő *metaball*ok listáját.

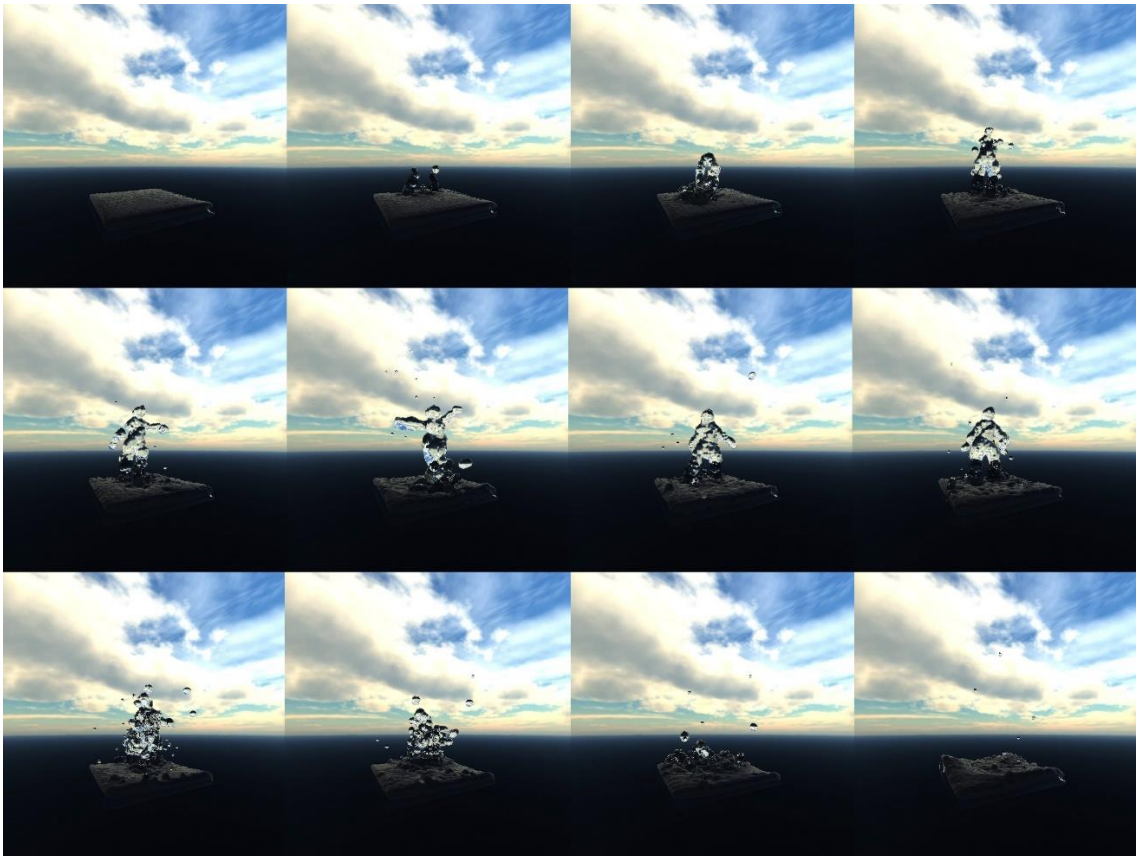
Rekurzív *ray marching* esetében is felhasználható az előállított szűrt *metaball* lista, ha a sűrűségfüggvény kiértékelési pozíciója visszavetíthető a képernyőre. Ebben az esetben a tört vagy visszavert sugár nem a kiindulási képponthoz tartozó *metaball* listát alkalmazza, hanem a releváns *metaball*ok a képernyőre visszavetített képponthoz tartozó listában találhatóak meg. Tehát visszaverődött vagy tört sugár esetében a releváns *metaball*ok listája változik a *ray marching* előrelépésekor. Ha a metszés kiértékelési pontja nem vetíthető vissza a képernyőre, akkor a *metaball*ok előszűrése nem használható. Ebben az esetben szűrés nélkül az implicit felület kiértékelése továbbra is elvégezhető vagy a rekurzív sugárkövetés megszakítható, mivel általános esetben a felhasználó számára nehezen észrevehető a képernyőn nem látható folyadékrészecskék kontribúciója.

5 Eredmények

A szimulációs rendszerünket C++ nyelven implemetáltuk. A hardveresen gyorsított megjelenítés megvalósításához DirectX 11 API-t használtunk. A tesztelést az 1. táblázatban látható két konfiguráción végeztük el. A tesztek során 2048 darab folyadékrészecske és közel 3000 darab kontrollrészecske volt felhasználva. A megjelenített képek felbontása 512x512.

5.1 A folyadékmanipuláció alkalmazása

Az elvégzett teszteken szereplő modellt és animációt a mixamo[29] honlapról töltöttük le és változtatás nélkül használtuk fel, ezáltal bizonyítottuk a módszerünk könnyű integrálhatóságát. A 18. ábrán látható a folyadékalapú karakter felépítése, animálása és lebontása, ebből kifolyólag az általunk megcélzott hatás létrehozása sikeres. Az *adaptív kontrollnyomás* és a kontrollrészecskék animációjának bevezetését szintén sikeresnek tartjuk a 3.1.3 és a 3.2.5 fejezetben bemutatott kísérletek függvényében.



18. ábra: Folyadékalapú karakter felépítése, animálása és lebontása

	Konfiguráció 1	Konfiguráció 2
Kivitel	Laptop	Desktop
Processzor	I7 9750H (2.6GHz)	I7 4790k (4.4GHz)
Videókártya	GTX 1660 Ti (M) (6GB)	GTX 1080 Ti (11GB)
Memória	DDR4 16 GB	DDR3 32 GB

1. táblázat: Tesztkörnyezet konfigurációk

5.2 A megjelenítési módszerek kiértékelése

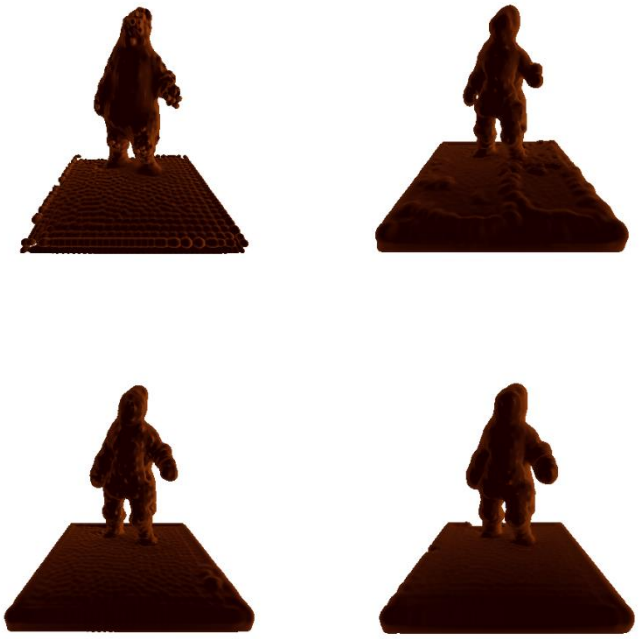
A megjelenítés számítási költségét a *ray marching* eljárás dominálja, az árnyalás elhanyagolható a rekurzív bejárás szükségességén kívül. Ebből kifolyólag a különböző paraméterek tesztelését elegendő elvégezni egy átlátszó és egy átlátszatlan megjelenítés esetében. Ennek megfelelően a 2. táblázatban látható, hogy az átlátszatlan megjelenítések egymáshoz viszonyítva és az átlátszó megjelenítések egymáshoz viszonyítva nem térnek el számítási költségben. A táblázatokban feltüntetett memóriaigény kizárólag az *A-Buffer* és az *S-Buffer* eljárások által ténylegesen használt memóriaterületre vonatkozik.

Megjelenítés	Optimalizáció	Frame per second		Memóriaigény
		Teszt K. 1	Teszt K. 2	
Phong árnyalás	Nincs	9	11	-
	A-Buffer	45	87	8.7 MB
	S-Buffer	52	98	5.8 MB
Átlátszó	Nincs	4	5	-
	A-Buffer	33	47	8.7 MB
	S-Buffer	40	65	5.8 MB
Részben átlátszó	Nincs	4	5	-
	A-Buffer	33	47	8.7 MB
	S-Buffer	40	65	5.8 MB
Arany	Nincs	9	11	-
	A-Buffer	51	87	8.7 MB
	S-Buffer	54	98	5.8 MB
Réz	Nincs	9	11	-
	A-Buffer	50	87	8.7 MB
	S-Buffer	54	98	5.8 MB
Alumínium	Nincs	8	11	-
	A-Buffer	50	87	8.7 MB
	S-Buffer	54	98	5.8 MB

2. táblázat: Mérési eredmények különböző megjelenítés esetén

5.2.1 A metaball sűrűségfüggvények hatása a teljesítményre

A 19. ábrán látható négy különböző *metaball* sűrűségfüggvényt vizsgáltuk meg. A mért megjelenítési sebesség a 3. táblázatban található. Az elvárásoknak megfelelően a különböző sűrűségfüggvények a megjelenítésben látható változást eredményeztek, de a teljesítménybeli különbségük a *ray marching* mellett elhanyagolható. Ebből kifolyólag a *metaball* sűrűségfüggvény változtatása alkalmas a megjelenítés hangolására teljesítménybeli romlás nélkül.



19. ábra: Vezérelt folyadék megjelenítése különböző metaball sűrűségfüggvények alkalmazásával (a) Original, (b) Wyvill, (c) Nishimura, (d) Murakami

Sűrűségfüggvény	Optimalizáció	Frame per second		Memóriaigény
		Teszt K. 1	Teszt K. 2	
Original[7]	Nincs	8	11	-
	A-Buffer	49	83	9MB
	S-Buffer	50	95	6MB
Wyvill[24]	Nincs	8	11	-
	A-Buffer	51	87	9MB
	S-Buffer	53	98	6MB
Nishimura[25]	Nincs	8	10	-
	A-Buffer	51	87	9MB
	S-Buffer	52	97	6MB
Murakami[26]	Nincs	8	10	-
	A-Buffer	50	88	9MB
	S-Buffer	51	98	6MB

3. táblázat: Mérési eredmények különböző metaball sűrűségfüggvények alkalmazása esetén

5.2.2 A metaball sűrűségfüggvény paramétereinek hatása a teljesítményre

A Wyvill sűrűségfüggvény teljesítményvonzatát megvizsgáltuk különböző küszöbérték és *metaball* hatótávolságok esetén. A 20. ábrán látható a megváltoztatott paraméterek hatása a megjelenítésen. A 4. táblázatban látható a megjelenítés sebessége a *metaball* küszöbérték és *metaball* hatótávolság függvényében. A *metaball* sűrűségfüggvények hatótávolságának és az implicit felületet meghatározó küszöbérték hangolásával nagy mértékben befolyásolható a *metaball*ok összeolvadása és mérete. Megnövelt küszöbérték és megnövelt *metaball* hatótávolság esetén kapható a legsimább és legrészletesebb felület, viszont a vártaknak megfelelően ebben az esetben a legnagyobb a megjelenítés számítási költsége.

20. ábra: Vezérelt folyadék megjelenítése különböző küszöbérték és metaball hatótávolság esetén, (a) T=1.5, R=0.03, (b) T=1.5, R=0.04, (c) T=1.5, R=0.05, (d) T=0.9, R=0.03, (e) T=0.9, R=0.04, (f) T=0.9, R=0.05, (g) T=0.2, R=0.03, (h) T=0.2, R=0.04, (i) T=0.2, R=0.05

Küszöbérték	Optimalizáció	Hatótáv = 0.03			Hatótáv = 0.04			Hatótáv = 0.05		
		FPS		Memória	FPS		Memória	FPS		Memória
		TK.1	TK.2		TK.1	TK.2		TK.1	TK.2	
1.5	Nincs	9	11	-	8	12	-	8	12	-
	A-Buffer	56	101	6.2MB	50	91	12.7MB	44	80	15.2MB
	S-Buffer	57	103	4.6MB	53	98	7.9MB	48	91	9.1MB
0.9	Nincs	9	11	-	8	12	-	8	12	-
	A-Buffer	56	102	6.2MB	50	92	12.7MB	45	80	15.2MB
	S-Buffer	58	105	4.6MB	53	101	7.9MB	49	93	9.1MB
0.2	Nincs	8	12	-	8	12	-	8	12	-
	A-Buffer	57	103	6.2MB	52	93	12.7MB	47	84	15.2MB
	S-Buffer	58	105	4.6MB	54	101	7.9MB	49	92	9.1MB

4. táblázat: Mérési eredmények különböző küszöbérték és metaball hatótávolság esetén

5.2.3 A metszéspont finomításának teljesítménybeli hatása

A felületi metszéspont pontosabb megtalálásának érdekében bináris gyökkereső eljárást alkalmaztunk. A 21. ábrán látható különböző iterációs számú bináris gyökkereséssel előállított kép. Az 5. táblázatban látható a bináris keresés teljesítménybeli vonzata átlátszatlan megjelenítés esetén. A metszéspont finomításával nagy mértékben növelhető a megjelenített felület simasága, de teljesítményben is érezhető lassulást okoz.

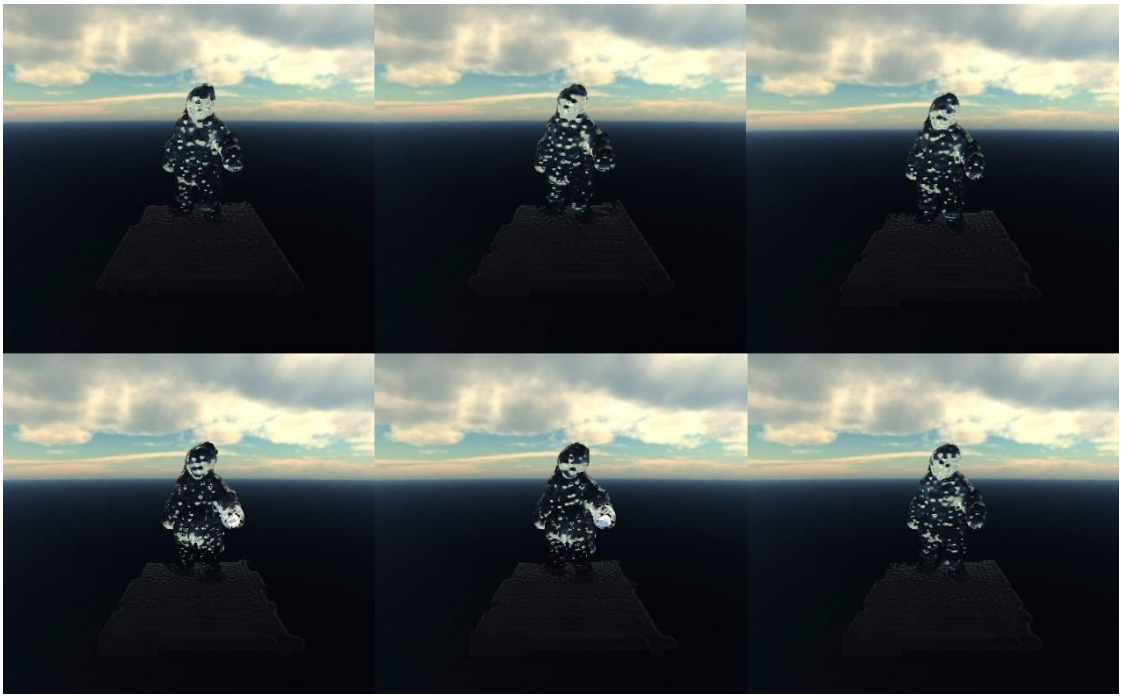
21. ábra: Vezérelt folyadék megjelenítése különböző iterációs számú gyökkereséssel finomított metszéspont esetén, (a) I=0, (b) I=2, (c) I=4

Optimalizáció	0 bin. iteráció			2 bin. iteráció			4 bin. iteráció		
	FPS		Memória	FPS		Memória	FPS		Memória
	TK.1	TK.2		TK.1	TK.2		TK.1	TK.2	
Nincs	10	14	-	9	13	-	8	9	-
A-Buffer	55	97	11.1MB	54	93	11.1MB	48	86	11.1MB
S-Buffer	57	101	7MB	56	98	7MB	53	99	7MB

5. táblázat: Mérési eredmények különböző iterációs számú gyökkereséssel finomított metszéspont esetén

5.2.4 A rekurzív sugárkövetés teljesítményvonzata

A rekurzív sugárkövetést megvizsgáltuk a maximális rekurzió mélység és a metszéspontot finomító bináris gyökkeresés iterációszámának függvényében. A 22. ábrán látható a teljesen átlátszó, a 23. ábrán látható a homogén elnyelést szimuláló megjelenítés. A kapcsolódó mérések a 6. és 7. táblázatban találhatóak. A megnövelt rekurzió nagy mértékben növeli az átlátszó folyadékok megjelenítésének élethűségét, főként a részlegesen átlátszó folyadék esetében, mert ebben az esetben az elnyelés számítása is pontosodik a rekurzió mélységének növelésével. A metszéspont finomítása az elvártaknak megfelelően átlátszó anyagok esetén még látványosabb, mint átlátszatlan anyagok alkalmazásakor, mert a metszéspont megtalálásának pontossága javítja a helyes törési és visszaverődési irány meghatározásának pontosságát. Az elvártnak megfelelően a rekurzió maximumának növelése nagy mértékben lassítja a megjelenítést, de meglepő módon a mélyebb rekurzió alkalmazásakor a metszéspont bináris kereséssel való pontosításának extra számítási költsége kevésbé jelentős.



22. ábra: Átlátszó folyadék megjelenítése különböző iterációs számú gyökkereséssel finomított metszéspont és különböző maximális mélységű rekurzív sugárkövetés alkalmazásakor (a) R=1 I=0, (b) R=1 I=2, (c) R=1 I=4, (d) R=3 I=0, (e) R=3 I=2, (f) R=3 I=4

Rekurzió mélysége:	Optimalizáció	0 bin. iteráció			2 bin. iteráció			4 bin. iteráció		
		FPS		Memória	FPS		Memória	FPS		Memória
		TK.1	TK.2		TK.1	TK.2		TK.1	TK.2	
1	Nincs	9	12	-	9	10	-	6	8	-
	A-Buffer	55	92	11.1MB	53	89	11.1MB	49	86	11.1MB
	S-Buffer	56	100	7MB	55	98	7MB	52	97	7MB
3	Nincs	3	3	-	2	2	-	2	2	-
	A-Buffer	30	39	11.1MB	27	38	11.1MB	27	37	11.1MB
	S-Buffer	36	57	7MB	35	53	7MB	35	52	7MB

6. táblázat: Mérési eredmények átlátszó folyadék megjelenítésekor különböző iterációs számú gyökkereséssel finomított metszéspont és különböző maximális mélységű rekurzív sugárkövetés alkalmazásakor

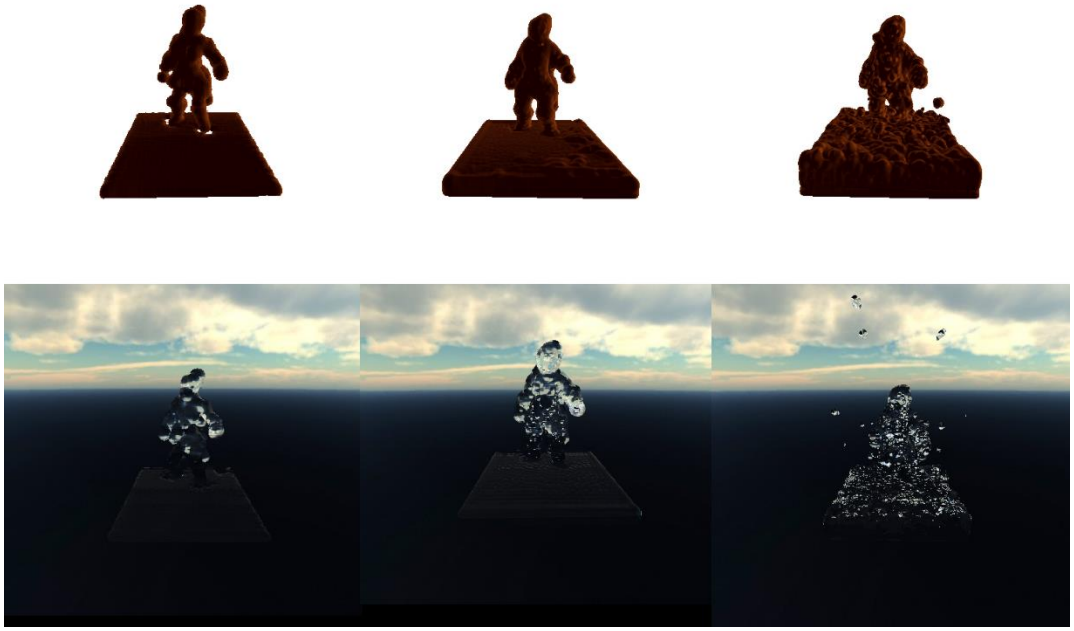
23. ábra: Részben átlátszó folyadék megjelenítése különböző iterációs számú gyökkereséssel finomított metszéspont és különböző maximális mélységű rekurzív sugárkövetés alkalmazásakor (a) R=1 I=0, (b) R=1 I=2, (c) R=1 I=4, (d) R=2 I=0, (e) R=2 I=2, (f) R=2 I=4, (h) R=3 I=0, (i) R=3 I=2, (j) R=3 I=4

Rekurzió mélysége	Optimalizáció	0 bin. iteráció			2 bin. iteráció			4 bin. iteráció		
		FPS		Memória	FPS		Memória	FPS		Memória
		TK.1	TK.2		TK.1	TK.2		TK.1	TK.2	
1	Nincs	10	12	-	8	10	-	7	8	-
	A-Buffer	51	92	6.7MB	49	89	6.7MB	48	86	6.7MB
	S-Buffer	53	101	4.9MB	52	98	4.9MB	51	97	4.9MB
2	Nincs	4	5	-	4	5	-	3	4	-
	A-Buffer	37	58	6.7MB	34	56	6.7MB	33	54	6.7MB
	S-Buffer	41	72	4.9MB	40	70	4.9MB	36	68	4.9MB
3	Nincs	3	3	-	2	3	-	2	2	-
	A-Buffer	33	46	6.7MB	31	43	6.7MB	30	41	6.7MB
	S-Buffer	37	60	4.9MB	35	56	4.9MB	35	56	4.9MB

7. táblázat: Mérési eredmények részben átlátszó folyadék megjelenítésekor különböző iterációs számú gyökkereséssel finomított metszéspont és különböző maximális mélységű rekurzív sugárkövetés alkalmazásakor

5.2.5 Folyadékrészecskék számának hatása a megjelenítés sebességére

Megvizsgáltuk a különböző mennyiségű folyadékrészecskék hatását a megjelenítés sebességére vonatkozóan. A 24. ábrán látható Phong árnyalású és átlátszó anyag esetén a különböző számú folyadékrészecskék alkalmazásakor a szimuláció megjelenítése. Több kontrollrészecske alkalmazásával nagyobb jelenetek is létrehozhatóak, vagy egy adott jelenet felskálázásával a folyadék élethűbben jeleníthető meg az arányaiban kisebb *metaball*oknak köszönhetően. A mérési eredmények a 8. táblázatban találhatóak. A vártaknak megfelelően a megvizsgált paraméterek közül a folyadékrészecskék számának volt a legnagyobb teljesítménybeli hatása. Valamint az elvárásainkak megfelelően a gyorsítóstruktúrák hatása is megnövekedett nagyobb számú folyadékrészecske alkalmazása esetén.



24. ábra: A vezérelt folyadék megjelenítése különböző számú folyadékrészecske esetén (a) N=1024, (b) N=2048, (c) N=4096, (d) N=1024, (e) N=2048, (f) N=4096

Részecskék száma	Optimalizáció	Phong árnyalás			Átlátszó felület		
		FPS		Memória	FPS		Memória
		TK.1	TK.2		TK.1	TK.2	
1024	Nincs	20	22	-	6	10	-
	A-Buffer	156	200	3.6MB	115	147	3.6MB
	S-Buffer	200	213	3.3MB	120	156	3.3MB
2048	Nincs	8	11	-	4	5	-
	A-Buffer	51	95	8.2MB	37	60	8.2MB
	S-Buffer	53	104	5.6MB	42	72	5.6MB
4096	Nincs	4	6	-	2	2	-
	A-Buffer	20	40	12MB	14	26	12MB
	S-Buffer	21	42	7.5MB	17	36	7.5MB

8. táblázat: Mérési eredmények különböző számú folyadékrészecske esetén

5.2.6 A megjelenítés optimalizálásának összefoglalása

Az optimalizációs eljárások sikeresnek bizonyultak, mert a modern hardverekhez viszonyítva elfogadható memóriaigény mellett átlagosan hatszoros teljesítménynövekedést produkáltak. Az *S-Buffer* minden tesztesetben jobban teljesített az *A-Buffer*nél. A legnagyobb teljesítménybeli különbséget a megnövelt részecskeszám, a megnövelt bináris pontosítási iterációszám és a megnövelt rekurzió mélység esetén mértük. A megnövelt részecskeszám esetén tapasztalt teljesítménybeli

különbség az *A-Buffer* alkalmazásakor tapasztalható kevésbé kedvező cache koherenciának köszenhető. A bináris pontosítás iterációszámának és a rekurzív sugárkövetés mélységének esetében a nagyobb mértékű gyorsulás a felépített gyorsítóstruktúra többszöri olvasása okozza. Az *S-Buffer* több rendermenetben építi fel a gyorsítóstruktúrát, de gyorsabban olvasható a kedvezőbb cache koherenciának köszönhetően. Ebből kifolyólag, ha a gyorsítóstruktúra többször kerül felhasználásra, akkor az *S-Buffer* előnye növekszik az *A-Buffer*rel szemben.

6 Konklúzió

Számos animált karaktermodell felhasználásával változatos és érdekes jeleneteket tudtunk létrehozni, amit más ismert algoritmusok segítségével csak befektetett extra művészi munka segítségével tudtunk volna megvalósítani. A kiegészített folyadékszimuláció és az átlátszatlan megjelenítés könnyedén teljesíti a valósidejű kritériumot. Az átlátszó anyagok esetében alkalmazott rekurzív megjelenítés korlátozott mélységű rekurzió alkalmazásakor szintén teljesíteni tudta a valósidejűséget. Ezáltal a megjelenítést és a szimulációt gyorsító eljárások sikeresnek bizonyultak.

A dolgozatunk bemutatja, hogy a modern videókártyák nagyteljesítményű párhuzamos számítási kapacitását optimálisan kihasználva a részecskerendszerek nem csak a folyadék szimulációjához és megjelenítéséhez alkalmazhatóak hatékonyan, hanem a folyadékot mesterségesen irányító kényszerek megvalósítására is. Ezáltal az eddig alkalmazott, az elvárt célállapotot voxel alapon reprezentáló, folyadékmanipulációs technikák lecserélhetőek részecskealapúra, aminek következtében új lehetőségek nyílnak a mesterséges kényszerek animációja terén. Az animált voxel alapú célállapot nagy memóriaigényű lehet, ha minden animációs frame egy részletes voxel tömb formátumban van előállítva. Továbbá két voxel tömb közötti interpoláció artifaktumokat eredményezhet, mivel a voxel tömb reprezentáció előállításakor elveszthetünk szemantikai információkat. Például karakteranimáció esetén a kontrollrészecskék animálhatóak *dual quaternio*k segítségével, aminek következtében a frame-ek közötti interpoláció az emberi test mozgását valósághűbben rekonstruálja. Ehhez hasonló interpoláció nem lehetséges, ha a karakter animációja egy voxel tömb sorozat formájában van meghatározva. Ebből kifolyólag a további kutatások várhatóan a részecskealapú folyadék manipuláció előnyét növelni fogják a voxel alapú megközelítésekkel szemben animált célállapot alkalmazásakor.

Irodalomjegyzék

[1] Micky Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. *University of Copenhagen: Department of Computer Science*, 2, 2006.

[2] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007.

[3] Nick Foster and Dimitris Metaxas. Controlling fluid animation. In *Proceedings Computer Graphics International*, pages 178–188. IEEE, 1997.

[4] Mark Wiebe and Ben Houston. The tar monster: Creating a character with fluid simulation. In *ACM SIGGRAPH 2004 Sketches*, page 64. 2004.

[5] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. In *ACM SIGGRAPH 2004 Papers*, pages 441–448. 2004.

[6] Jeong-mo Hong and Chang-hun Kim. Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds*, 15(3-4):147– 157, 2004.

[7] James F Blinn, “A generalization of algebraic surface drawing.”, *ACM Transactions on Graphics (TOG)* pp. 235256, (July 1982).

[8] Nishimura H., Hirai M., Kawai T., Kawata T., Shirakawa I. and Omura K, “Object modeling by distribution function and a method of image generation.”, *The Transactions of the Institute of Electronics and Communication Engineers of Japan 68* Part 4 (1985) pp. 718725.

[9] Lorensen W. and Cline H.: “Marching cubes: A high resolution 3D surface construction algorithm.”, *ACM Siggraph Computer Graphics (1987)* vol. 21. pp. 163169.

[10] van der Laan W., Green S. and Sainz M.: “Screen space fluid rendering with curvature flow.”, *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (2009)* ACM, pp. 9198.

[11] K. Iwasaki, Y. Dobashi, F. Yoshimoto and T. Nishita, “Real-time Rendering of Point Based Water Surfaces”, *Computer Graphics International 2006* pp. 102114.

[12] K. van Kooten, A. Telea, G. van der Bergen “Point-Based Visualization of Metaballs on a GPU”, *GPU Gems 3, Chapter 7 (2007)* pp. 123148.

[13] Tomoyuki Nishita and Eihachiro Nakamae. A method for displaying metaballs by using bezier clipping. In *Computer Graphics Forum*, volume 13, pages 271–280. Wiley Online Library, 1994.

[14] Charles Loop and Jim Blinn. Real-time GPU rendering of piecewise algebraic surfaces. In *ACM SIGGRAPH 2006 Papers*, pages 664–670. 2006.

[15] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. GPU-based fast ray casting for a large number of metaballs. In *Computer Graphics Forum*, volume 27, pages 351–360. Wiley Online Library, 2008

[16] László Szécsi and Dávid Illés. Real-time metaball ray casting with fragment lists. In *Eurographics (Short Papers)*, pages 93–96, 2012.

[17] Loren Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 103–108, 1984.

[18] Andreas Vasilakis and Ioannis Fudos. S-buffer: Sparsity-aware multi-fragment rendering. In *Eurographics (short papers)*, pages 101–104. Citeseer, 2012.

[19] Morton, G. M. (1966), A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing (PDF), Technical Report, Ottawa, Canada: IBM Ltd.

[20] Bui Tuong Phong, Illumination of Computer-Generated Images, Department of Computer Science, University of Utah, July 1973.

[21] István Lazányi and László Szirmay-Kalos. Fresnel Term Approximations for Metals. In *WSCG (Short Papers) 2005*, pages 77-80

[22] A. Fresnel, (ed. H. de Senarmont, E. Verdet, and L. Fresnel), Oeuvres complètes d'Augustin Fresnel, *Paris: Imprimerie Impériale*, 1866

[23] Lambert, J.H. Photometria sive de mensura et gradibus luminis, colorum et umbrae. *Augsburg: Eberhardt Klett*. 1760

[24] B. Wyvill, C. McPheeters, G. Wyvill, Data structure for soft objects. In *The Visual Computer, 2*, pages 227-234, 1986

[25] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, K. Omura, Object Modeling by Distribution Function and a Method of Image generation, In *Electronics Communication Conference*, pages 718-725, 1985

[26] S. Murakami, H. Ichihara, On a 3D Display Method by Metaball Technique, In *Electronics Communication Conference*, pages 1607-1615, 1987

[27] Marching Cubes: Curve Wrapping & More Metaballs *https://www.grasshopper3d.com/profiles/blogs/marching-cubes-curve-wrapping-more-metaballs*

[28] Z-order curve, Wikipedia *https://en.wikipedia.org/wiki/Z-order_curve*

[29] Mixamo

https://www.mixamo.com