Revista Română de Interacţiune Om-Calculator 7 (1) 2014, 71-88

Tehnici de interacţiune utilizator cu obiecte 3D modelate prin particule

Denisa Copândean, Adrian Sabou, Dorian Gorgan

Universitatea Tehnică Cluj-Napoca – UTCN
Str. Memorandumului nr. 28, Cluj-Napoca
E-mail: denisa_s_86@yahoo.com , {adrian.sabou, dorian.gorgan}@cs.utcluj.ro

Rezumat. Pentru a dezvolta aplicaţii flexibile, cu un grad ridicat de interactivitate, trebuie înţelese capacităţile şi limitele utilizatorului şi ale calculatorului. Interacţiunea dintre utilizator şi obiectele modelate este asigurată prin informaţia care se primeşte şi se transmite. De asemenea, pentru a dezvolta sisteme interactive, trebuie să se ţină cont de modul în care contextul şi modelul sunt definite şi dezvoltate. În cazul nostru, am determinat faptul că modelarea prin particule introduce provocări în ceea ce priveşte interacţiunea în mediul virtual, fiind necesară dezvoltarea unor tehnici de interacţiune particularizate pentru a simula operaţiile naturale ale utilizatorului. Această lucrare prezintă tehnici de interacţiune a utilizatorului cu obiecte deformabile modelate prin tehnica particulelor, precum selecţia, manipularea şi secţionarea în volum. De asemenea, sunt descrise conceptele şi tehnicile care stau la baza acestor interacţiuni, precum tehnica "*bounding volume"* şi implementarea instrumentului virtual care realizează secţionarea. Nu în ultimul rând sunt descrise provocările care apar în realizarea interacţiunii într-o scenă tridimensională folosind dispozitive de intrare bidimensionale precum mausul.

Cuvinte cheie: obiecte volumetrice 3D, paralelism, lamă, tăiere.

1. Introducere

Una din tendinţele graficii pe calculator este aceea de a emula cât mai fidel realitatea ce ne înconjoară. Realitatea virtuală reprezintă o simulare generată de calculator a unui mediu tridimensional în care utilizatorul este capabil să vizualizeze şi să manipuleze conţinutul acestui mediu. Utilizatorul unui sistem virtual are libertatea de a explora lumea creată de calculator şi de a interacţiona direct cu ea.

În lucrarea de faţă se prezentă un simulator de modelare a obiectelor volumetrice 3D dinamice reprezentate prin particule. Acestea au un comportament natural, realist, conform unui set de legi fizice bine definite, spre deosebire de modelele definite analitic care se comportă de fiecare dată la fel având astfel un caracter previzibil. Implementarea legilor fizice

presupune multe calcule costisitoare care pot altera simularea. Pentru a îmbunătăți puterea de calcul a sistemului, modelul a fost dezvoltat cu ajutorul tehnologiei OpenCL, care profită de puterea de calcul a unităților de prelucrare grafică (GPU). Mai exact, fiecare particulă este procesată la nivel de "*thread*", fir de execuție, şi în paralel cu celelalte particule.

Utilizatorul poate crea obiecte volumetrice 3D realiste cu parametri configurabili şi poate urmări comportamentul acestor obiecte într-o scenă în care sunt simulate forțele fizice newtoniene. El poate interacționa cu obiectele volumetrice 3D prin intermediul unor operații precum selecția, manipularea şi tăierea. Pentru operația de tăiere, utilizatorul foloseşte o lamă de secționare predefinită sau creată de el în timpul simulării.

S-a urmărit optimizarea operației de tăiere, care este extrem de costisitoare din punctul de vedere al resurselor de calcul necesare. Am propus spre implementare metoda "*bounding volume*" (încadrare volumetrică). Modul în care această tehnică este utilizată, pentru a aduce îmbunătățiri operației de tăiere, este prezentat în secțiunea "Optimizarea operației de tăiere: *bounding volume*".

În secțiunea "Lucrări asemănătoare" se prezintă o serie de lucrări care au abordat un subiect similar. În secțiunile "Simularea modelului 3D" şi "Modelarea instrumentului de tăiere", se face o prezentare a modelului 3D manipulat, a motorului fizic în care are loc simularea şi, a modelului lamei de tăiat. Rezultatele experimentelor realizate prin interacțiunea utilizatorului cu aplicația se prezintă în secțiunea "Experimente – Operațiile de interacțiune". De asemenea, se prezintă concluziile obținute în secțiunea "Concluzii şi dezvoltări ulterioare".

2. Lucrări asemănătoare

Având drept scop familiarizarea cu animația bazată pe legile fizicii, Barbic (Barbic, J., 2010) prezintă o formă simplă de modelare 3D bazată pe particule, iar Bourke (Bourke, P., 1998) realizează o scurtă introducere în domeniul sistemelor de particule. De-a lungul timpului sistemele de particule au fost utilizate în diferite moduri. Spre exemplu, Min Hong et al (Min Hong et al, 2005) foloseşte un sistem de particule în care resorturile doar conectează particulele şi descriu distanța dintre acestea în urma aplicării unor forțe externe. Astfel sistemul de particule nu deține informații volumetrice. Acest principiu îl întâlnim şi la Sylvester Arnab et al

(Sylvester Arnab şi Vinesh Raja, 2013). Conectarea unui sistem de particule pentru a se simula un obiect volumetric se poate face în mai multe moduri. Cele mai des întâlnite topologii sunt cele prin care obiectul este modelat cu ajutorul unei rețele în interiorul căreia legăturile se fac într-o schemă triunghiulară sau de patrulater (Barbic, J., 2010). Conectarea sistemului este foarte importantă dacă legile fizicii sunt integrate numeric. Matthias Muller et al (Matthias Muller et al, 2008) prezintă în notițele sale de curs modul în care simularea este îmbunătățită prin integrarea ecuațiilor fizice în metodele şi algoritmii actuali de simulare, incluzând cei bazați pe sisteme de particule. David Baraff et al. (David Baraff, 1998) prezintă avantajele utilizării metodelor de integrare numerică implicită în simularea textilelor. Autorii au demonstrat că integrarea implicită poate să ofere performanțe superioare prin faptul că permite paşi de simulare mult mai mari, fără a suferi de instabilitatea inerentă integrării explicite. Şi Joachim Georgii et al (J. Georgii et al, 2005) a optat pentru o astfel de simulare. De aceea, am optat pentru o modelare bazată pe particule care implică integrarea numerică. Integrarea numerică necesită o mulțime de operații costisitoare. Pentru a contracara acest deficit particulele sunt prelucrate la nivelul unităților de procesare grafică (GPU - Graphical Processing Unit). Pentru a obține un comportament al modelului cât mai real trebuie să găsim un echilibru între constantele care dictează comportamentul modelului, constante precum cea a lui Hook, de atenuare, densitatea modelului, masa unei particule, timpul de integrare (Δt) etc. Articolul „Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior" (Provot, 1996) explică fenomenul „super elastic" care reprezintă una dintre cele mai mari probleme ce pot sa apară la simularea de suprafețe textile, fie ele 2D sau 3D. De asemenea, articolul prezintă condiția critică de a nu obține acest fenomen, precum şi o metodă de a anula acest efect nedorit dacă nu se ține cont de respectiva condiție.

Pentru înglobarea de legi fizice în cadrul unei aplicații grafice în timp real, este folosit de obicei un motor fizic (physics engine), (Bose M. şi Rajagopala V., 2012). Firth (Firth P., 2011) prezintă principalele componente ale unui motor fizic într-un spațiu 2D: modele, constrângeri, coliziuni etc. Pornind de la ideea de modelare folosind GPU, descrisă de Sabou (Sabou Adrian, 2012), am extins motorul fizic de simulare utilizat pentru a manipula nu doar obiecte bidimensionale ci şi tridimensionale.

Motorul fizic este dezvoltat cu ajutorul tehnologiei OpenCL, Open Computing Language (Munshi, A., 2009). OpenCL este un cadru de lucru pentru scrierea de programe care se execută pe platforme heterogene formate din unități centrale de procesare (CPU), unități de procesare grafică şi alte procesoare. Acesta include un limbaj (bazat pe C99) pentru a scrie funcții (kernels, care se execută pe dispozitive OpenCL), plus interfețe de programare a aplicațiilor (API - Application Programming Interface), care sunt folosite pentru a defini şi pentru a controla apoi platformele. De asemenea, cadrul OpenCL se îmbină perfect cu tehnologia OpenGL prin obiectele sale de tip "*buffer*", care pot fi accesate din ambele părți fără a fi necesară copierea lor.

Una dintre cele mai des întâlnite tehnici de interacțiune este selecția. În această lucrare, selecția este realizată cu ajutorul metodei "*raycast*", prin care un pixel este calculat ca intersecția dintre raza vizuală ce porneşte de la observator şi obiectul volumetric. John Pawasauskas (John Pawasauskas, 1997) utilizează această tehnică pentru a prezenta date medicale în spațiul 3D. Manipularea se bazează pe operația de selecție şi aplică o forță exterioară obiectului manipulat. Sistemul prezentat nu se limitează doar la aceste interacțiuni de bază, ci oferă suport şi pentru alte interacțiuni, precum secționarea, mişcarea în scenă şi altele. O prezentare sumară a sistemului şi a tehnicilor implementate, se face în lucrarea publicată la Conferința Națională de Interacțiune Om-Calculator (Copândean D. et al, 2013).

3. Simularea modelului 3D

Corpurile din viața reală pot fi aproximate, într-o măsură mai mare sau mai mică, prin sisteme de puncte discrete. Pentru a se menține ca o entitate stabilă, între constituenții unui astfel de sistem, sau între aceştia şi corpurile exterioare, trebuie să se exercite forțe de atracție sau forțe de legătură.

În cazul sistemelor de puncte discrete, există două categorii distincte de forțe: forțe interne şi forțe externe. Forțele de interacțiune dintre perechile de particule constituente ale sistemului se numesc forțe interne. Forțele de interacțiune dintre particulele sistemului şi alte corpuri sau câmpuri de forțe din exteriorul acestuia se numesc forțe externe.

3.1 Modelul 3D

Structura modelului fizic

Cel mai simplu model de particule pentru simularea unui obiect volumetric dinamic este modelul mass-spring. Acest model este format din particule. Particulele sunt caracterizate de masă, poziție şi viteză şi sunt interconectate între ele prin resorturi. O particulă poate fi un punct de masă care are o anumită poziție şi o anumită viteză la un anumit moment dat. Modul în care resorturile conectează particulele între ele şi diferența de putere a fiecărui resort influențează comportamentul obiectului ca un tot unitar.

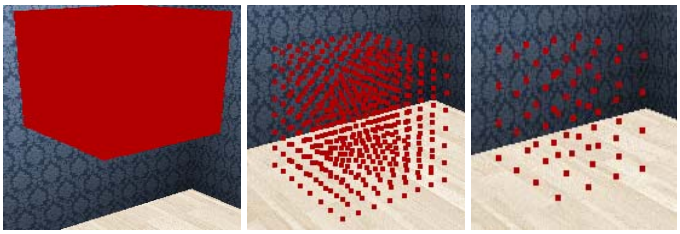
Pentru a recrea cele trei proprietăți ale unui obiect volumetric şi anume rezistența la întindere, îndoire şi tăiere avem nevoie să folosim trei tipuri de resorturi. Fiecare tip de resort modelează una din cele trei tipuri de comportament ale unui obiect volumetric real. Resorturile se clasifică în funcție de rigiditatea lor şi de modul în care conectează particulele.

Primul tip, numit resort structural (*"structural spring"*), defineşte structura de bază a obiectelor volumetrice şi modelează faptul că obiectele reale nu se întind foarte mult. Aceste resorturi simulează faptul că un obiect volumetric este alcătuit din fibre longitudinale şi transversale. Al doilea tip de resorturi sunt numite resorturi de rupere (*"shear springs"*) şi fac ca rezistența obiectului volumetric la rupere să fie mai mare. Aceste resorturi ajută obiectul volumetric să păstreze o anumită formă şi nu lasă particulele să cadă una în cealaltă, să se apropie sau să se îndepărteze prea tare. Aceste resorturi conectează particulele pe diagonală în fiecare dreptunghi format de către resorturile structurale. Al treilea tip de resorturi sunt resorturi de îndoire (*"bend springs"*). Aceste resorturi fac ca îndoirea obiectului volumetric simulat să fie mai dificilă.

Definirea modelului

Pentru a defini un model, cu atât mai mult unul de tip mass-spring, vom utiliza doi dintre cei trei parametri care descriu un model volumetric 3D: pasul (densitatea), dimensiunea şi numărul de particule. Metoda utilizată de aplicația descrisă în această lucrare este cea care utilizează pasul şi dimensiunea. Având aceşti doi parametri ca intrare, putem deduce numărul de particule. Deoarece se simulează un obiect volumetric 3D, dimensiunea

este dată de trei variabile: lungimea, lățimea şi înălțimea obiectului. Pasul care redă densitatea unui obiect se aplică în aceeaşi măsură atât pe lungime, cât şi pe lățime şi înălțime. Dacă pasul este mare, obiectul volumetric nu este foarte dens. În schimb, dacă pasul este mic, obiectul volumetric are o densitate mai mare de particule (Figura 1).



a) *b)* *c)*

Figura 1. Model cu pas: a) mic; b) mediu; c) mare.

Astfel, numărul total de particule care alcătuiesc modelul mass-spring al unui obiect 3D se determină după formula:

$$N = (l / p + 1) * (L / p + 1) * (h / p + 1) \quad (1)$$

unde *l* reprezintă lățimea, *L* lungimea, *h* înălțimea, *p* pasul şi *N* numărul total de particule.

3.2 Motorul fizic

Distribuția paralelă

Motorul fizic a fost implementat şi extins folosind calculul paralel la nivel de model bazat pe multithreading. Tehnologia care oferă suport de multithreading pe acceleratoare hardware grafice şi pe care am ales-o pentru a implementa motorul de simulare este OpenCL.

OpenCL oferă calcul paralel folosind un paralelism pe task-uri sau pe date. De asemenea, oferă acces la unitatea de procesare grafică pentru calcule non-grafice. Astfel, OpenCL extinde puterea unității de procesare grafică. Motorul fizic bazându-se pe această tehnologie simulează fizica newtoniană folosind variabile precum masa, viteza, frecarea şi rezistența vântului. Modelul poate simula şi prezice efecte sub diferite condiții care aproximează ce se întâmplă în realitate.

De asemenea motorul permite modelarea şi manipularea obiectelor volumetrice 3D într-un mod cât mai natural prin intermediul dispozitivelor standard de intrare. Fiecare particulă va fi procesată de un fir de execuție care va rula nucleul OpenCL. Acest nucleu realizează atât procesul de simulare unde are loc integrarea în timp real a forțelor ce acționează asupra modelului, cât şi detecția şi răspunsul la coliziune.

Integrarea numerică în timp real

Acest proces se realizează parcurgând următorii paşi:

1. Simularea fizică a dinamicii unui sistem de particule de tip mass-spring se poate realiza parcurgând următorii paşi: cumularea tuturor forțelor ce acționează asupra particulelor, atât forțele ce apar în interiorul rețelei de particule de la resorturile care conectează particulele între ele, cât şi, forțele externe precum gravitația, vântul, alte forțe ce acționează asupra unor particule etc. Forțele externe le includ şi pe cele ce rezultă în urma ciocnirii sau a interacțiunii cu alte obiecte din mediu.
2. Calcularea accelerației particulelor rezultate din aceste forțe.
3. Actualizarea vitezelor particulelor şi pozițiile acestora folosind accelerațiile calculate.

Integrarea în timp real este dată de legea a doua de mişcare a lui Newton, calculând accelerația fiecărei particule la un moment dat. Aceasta este implementată prin metoda Verlet, iar vitezele se calculează printr-o formă simplificată, folosind formula:

$$v(t + \Delta t) = v(t) + a(t) * \Delta t \quad (2)$$

Coliziunea

Coliziunea reprezintă procesul fizic de ciocnire. Ciocnirea a două sau mai multe corpuri reprezintă un proces de interacțiune care durează un timp foarte scurt. Motorul fizic trebuie să îndeplinească două funcționalități legate de coliziune: detecție şi răspuns. O dată detectată coliziunea, e necesar să se aplice răspunsul de îndreptare, de corecție a coliziunii. Acest răspuns se mai numeşte şi rezoluție a coliziunii.

Metoda de rezoluție aplicată în lucrarea de față se bazează pe principiul conservării impulsului. Coliziunea luată în considerare, în implementarea

curentă, este doar față de pereții încăperii. Principala problemă întâlnită este coliziunea între particule. Sistemul este creat pentru a simula un număr foarte mare de particule, astfel că detecția coliziunii dintre particule ar duce la un timp foarte mare de calcul care ar altera trasarea obiectului volumetric 3D, simularea nemaifiind realizată în timp real. Pentru a rezolva această problemă se urmăreşte mărirea puterii de calcul, şi anume trecerea procesării pe mai multe unități de calcul, spre exemplu, utilizarea unui *cluster*. Un prim pas pentru îmbunătățirea puterii de calcul este optimizarea "colaborării" dintre tehnologiile folosite. Pentru a optimiza „colaborarea" OpenCL-OpenGL s-au dezvoltat obiecte (ex: VBO) şi metode specifice. Această „colaborare" poartă numele de „interoperabilitatea OpenCl-OpenGL". Totuşi, în lucrarea de față, urmărim doar modelarea instrumentului de tăiere şi experimentarea tehnicilor de interacțiune cu acest instrument.

4. Modelarea instrumentului de tăiere

Operația de tăiere funcționează prin secționarea legăturilor dintre particule şi nu prin scindarea particulelor, astfel numărul total de particule nu creşte ci rămâne acelaşi de fiecare dată când are loc o tăiere.

4.1 Algoritmul de tăiere

Operația de tăiere are loc în funcție de un plan. Acest plan este vizualizat ca o lamă. Practic, secționarea se determină matematic. Planul lamei se defineşte cu ajutorul a trei puncte din plan: punctul de început al lamei şi punctul de sfârşit al lamei definesc linia de tăiere şi, punctul „direcție". Orientarea lamei este dată de dreapta care trece prin punctul "direcție" şi este perpendiculară pe lungimea lamei. Având aceşti parametri, se trece la pasul următor al algoritmului de tăiere care constă în calcularea distanței de la plan la fiecare particulă. În cadrul algoritmului de tăiere, formula distanței de la un punct la o dreaptă se aplică fără modulul numărătorului, deoarece dorim să ştim pe care parte a planului se află particula. Mai exact dacă distanța este pozitivă particula se află în semi-spațiul pozitiv determinat de respectivul plan, adică în fața planului, iar dacă este negativă particula se află în semi-spațiul negativ al planului respectiv, adică în spatele planului.

De asemenea se calculează pentru fiecare particulă distanța de la fiecare particulă vecină la planul lamei. Dacă notăm distanța particulei la planul lamei cu d_1 şi cu d_2 distanța de la particula vecină la planul lamei, atunci avem următoarele cazuri:

1. $d_1 < 0$ şi $d_2 < 0$ – ambele particule se află în semi-spațiu negativ, iar legătura dintre particule nu se taie;
2. $d_1 > 0$ şi $d_2 > 0$ – ambele particule se află în semi-spațiu pozitiv, iar legătura dintre particule nu se taie;
3. $d_1 < 0$ şi $d_2 > 0$ – particula curentă se află în semi-spațiul negativ, iar particula vecină se află în semi-spațiul pozitiv, deci resortul dintre particule este eliminat, se taie;
4. $d_1 > 0$ şi $d_2 < 0$ – particula curentă se află în semi- spațiul pozitiv, iar particula vecină se află în semi- spațiul negativ, deci resortul dintre particule este eliminat, se taie;
5. $d_1 = 0$ sau $d_2 = 0$ – particula curentă, respectiv vecină, se consideră în semi-spațiul pozitiv, încadrându-se astfel în unul din cazurile descrise mai sus.

4.2 Definirea lamei

Lama reprezintă obiectul virtual de tăiere în cadrul simulării. Precum un cuțit adevărat, lama are un singur tăiş. Acest tăiş este definit ca o dreaptă dată de două puncte: punctul de început (punctul A) şi punctul de sfârşit (punctul B). Odată definit tăişul se poate determina forma fizică a lamei. Pornind de la punctul A şi punctul B se calculează punctul A' şi punctul B' ținând cont de lățimea lamei stabilită la început.

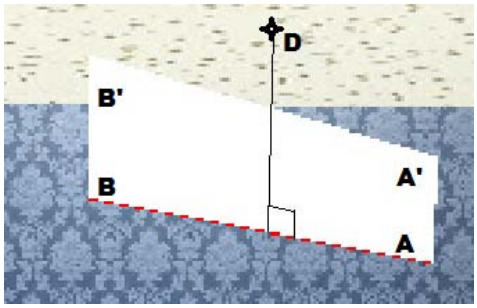


Figura 2. Modelul lamei

Totuşi, înainte de a calcula punctele A' şi B' trebuie determinată orientarea lamei. Astfel se alege un punct D din spațiu, ce va determina orientarea lamei. Acest punct va fi proiectat pe tăişul lamei, după care se va calcula dreapta ce trece prin punctul D şi prin proiecția acestuia. Perpendiculara reprezintă orientarea lamei şi pe baza ei se vor calcula punctele A' şi B' (Figura 2). Punctele A' şi B' se determină utilizând noua orientare (d) şi valoarea lățimii (l):

$$\begin{aligned} A' &= A + d * l \\ B' &= B + d * l \end{aligned} \tag{3}$$

4.3 Deplasarea lamei

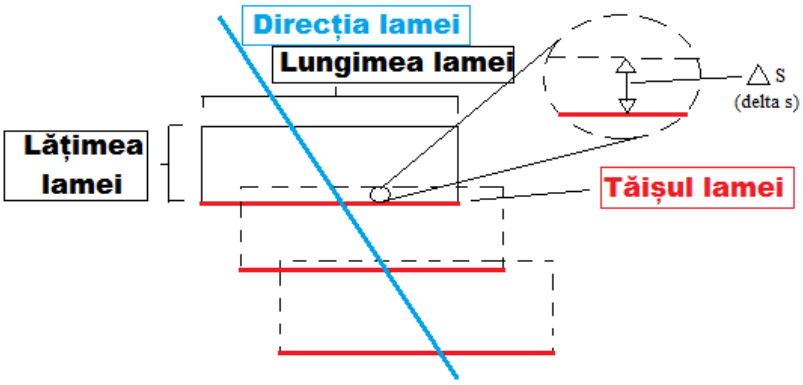


Figura 3. Deplasarea lamei

După cum se poate observa şi în figura 3, Δs (delta s) reprezintă viteza de deplasare a lamei. Lama se deplasează pe o direcție dată. Algoritmul de tăiere se bazează pe analiza particulelor. Pentru fiecare particulă analizată se testează dacă vecinii săi se află în celălalt semispațiu definit de planul lamei. În caz afirmativ legătura dintre cele două particule este distrusă. Astfel, dacă Δs este foarte mic, se vor repeta multe calcule, dar există o posibilitate mică de a rămâne particule netestate. Adică, pentru particula cu albastru din figura 4, algoritmul de tăiere se aplică de mai multe ori, chiar dacă ea a fost analizată o dată.

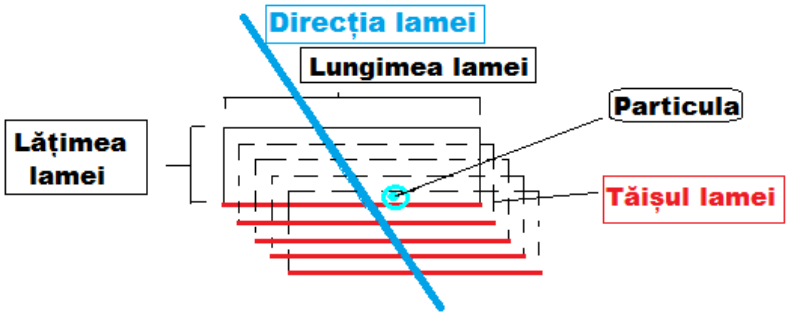


Figura 4. Deplasarea lamei cu Δs mic

Dacă Δs este foarte mare unele particule nu sunt testate, iar secționarea nu se realizează corespunzător (Figura 5).

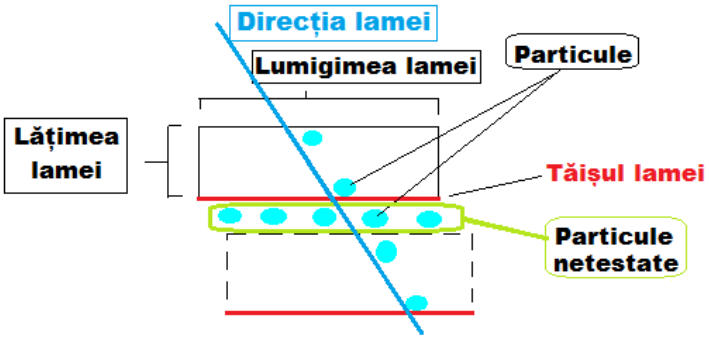


Figura 5. Deplasarea lamei cu Δs mare

Deci Δs trebuie să fie în intervalul (0, lățime) pentru ca algoritmul de tăiere să funcționeze corect.

4.4 Controlul interactiv al deplasării lamei

În timpul simulării se poate schimba oricând direcția de deplasare a lamei (Figura 6). Pentru a avea control prin manipularea directă a mişcării lamei de tăiat, este necesar doar redefinirea punctului D în spațiu şi reluarea

calculelor pentru determinarea perpendicularei dreptei AB ce trece prin punctul D.

Lama taie doar pe orientarea definită de perpendiculara DC, însă ea poate fi poziționată în spațiu şi se poate deplasa şi pe sensul opus orientării DC dar fără a tăia.

De asemenea, în timpul simulării se poate schimba şi pasul de deplasare a lamei.

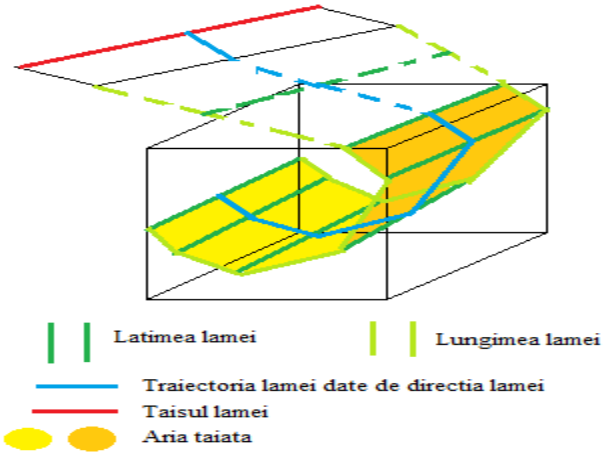


Figura 6. Controlul interactiv al deplasării lamei

Acțiunile de control ale utilizatorului sunt interceptate cu ajutorul funcțiilor predefinite din OpenGL. Fiecare acțiune are asociată o valoare care afectează parametrii definitorii ai lamei.

Pentru a realiza operația efectivă de secționare trebui să deplasăm lama. Dacă lama se deplasează pe orientarea ei va tăia legăturile întâlnite. În caz contrar, caz în care lama se deplasează în sens opus, nu va tăia nimic chiar dacă întâlneşte obiectul volumetric 3D.

4.5 Optimizarea operației de tăiere: "*bounding volume*"

Algoritmul de tăiere este unul foarte costisitor atât în timp cât şi în resurse, deoarece utilizează foarte multe operații de înmulțire şi împărțire şi mulți radicali. Practic se calculează distanța la plan pentru particula curentă, apoi pentru toți vecinii săi, care sunt în număr de maxim 24 (6 vecini

structurali, 12 vecini în forfecare şi 6 vecini de îndoire). Pentru a optimiza algoritmul de tăiere trebuie să optimizăm analiza particulelor.

Se ştie faptul că o operație de comparare nu este una costisitoare. De aceea primul pas al optimizării analizei particulelor este de a defini o primă extensie a lamei (Figura 7).

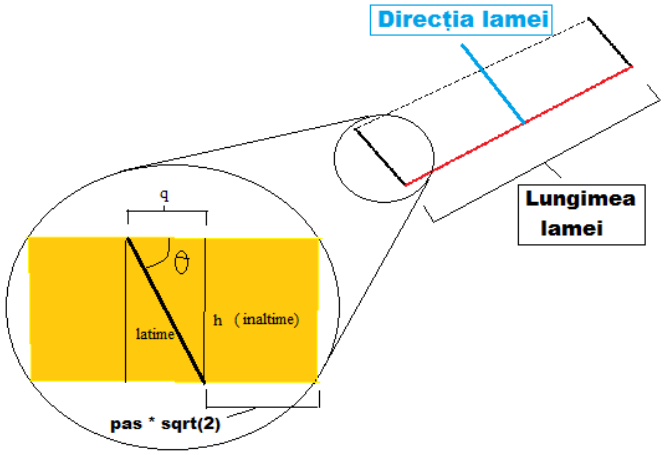


Figura 7. Definirea extensiei lamei

Determinarea parametrilor definitorii ai extensiei lamei, unde *l* reprezintă lățimea lamei :

$$\begin{aligned} h &= l * \sin(\theta) \\ q &= l * \cos(\theta) \end{aligned} \tag{4}$$

Volumul extensiei, unde *L* reprezintă lungimea şi *p* pasul lamei, va fi:

$$[\, h * (\, q + 2 * p * \sqrt{2}\,)\,] * L \tag{5}$$

Toate particulele din interiorul volumului extensiei (x_{min} <= $x_{particulă}$ <= x_{max} , y_{min} <= $y_{particulă}$ <= y_{max} , z_{min} <= $z_{particulă}$ <= z_{max}) vor fi testate de către algoritmul de tăiere. Totuşi sunt multe particule de testat. De aceea s-a mai introdus încă o optimizare în ceea ce priveşte analiza particulelor. Pentru

particulele din volumul extensiei se va calcula distanța la planul lamei. Dacă aceasta este mai mică sau egală cu densitatea (*pasul*) * √2 pentru resorturile de forfecare şi densitatea (*pasul*)/2 pentru resorturile structurale şi de îndoire, atunci se va aplica algoritmul de tăiere, în caz contrar particula e prea departe şi nu are legături ce trebuie tăiate (Figura 8).

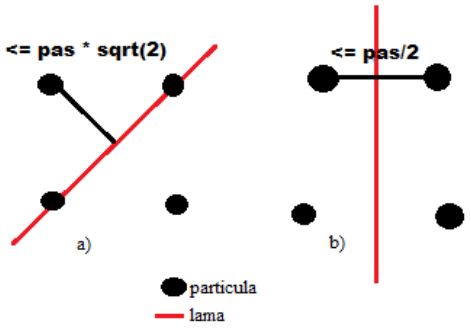


Figura 8. Optimizare bazată pe distanță

5. Experimente – Operațiile de tăiere

5.1 Selecția

Pentru a efectua interacțiunea de selecție, respectiv deselecție trebuie să se facă clic în scenă şi dacă este întâlnită o particulă cu tehnica Raycast acesteia i se va schimba culoarea (devine selectată/deselectată – Figura 9).

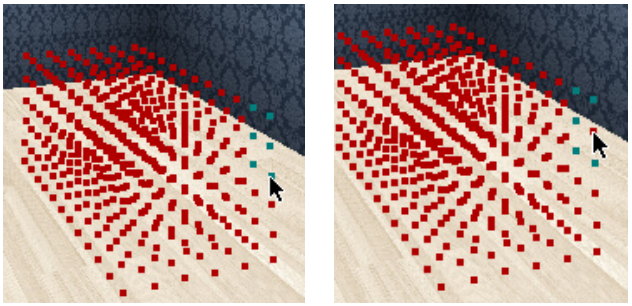
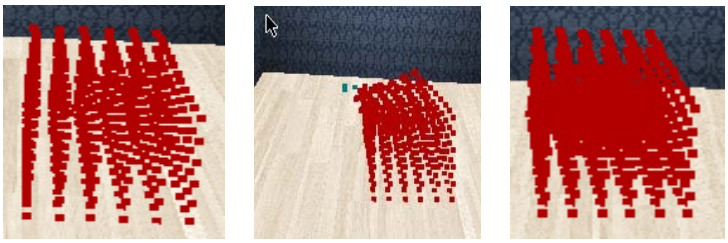


Figura 9. Selecție/Deselecție

5.2 Prindere şi tragere

Această tehnică se bazează pe tehnica de selecție. Se poate realiza ținând apăsat butonul stânga al dispozitivului maus şi modificând poziția cursorului. Modelul îşi modifică poziția în funcție de forța imprimată (Figura 10).

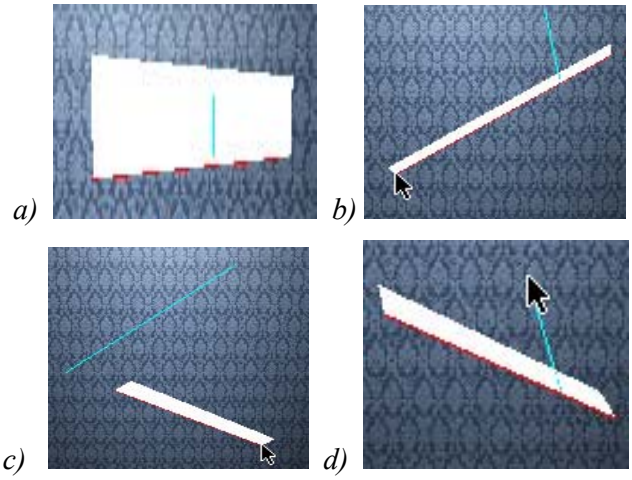


a) b) c)

Figura 10. Prinde şi trage: a) înainte; b) trage; c) după.

5.3 Definirea lamei

Se bazează pe gesturi clic consecutive în scenă: primul defineşte punctul de început al tăişului, al doilea punctul de sfârşit al tăişului, iar al treilea punctul care oferă orientarea lamei (Figura 11).



a) b) c) d)

Figura 11. Definire lamă: a) inițială; b) modificare început; c) modificare sfârşit; d) modificarea orientării.

5.4 Operația de tăiere

În funcție de parametrii săi definitorii, obiectul volumetric 3D răspunde într-un anumit mod la operația de tăiere. Spre exemplu, dacă obiectul volumetric este unul foarte rigid, adică are o structură foarte stabilă, în urma tăierii cu prima lamă predefinită (deasupra şi pe diagonală) rezultă două obiecte disjuncte care îşi păstrează forma şi poziția (Figura 12.a).

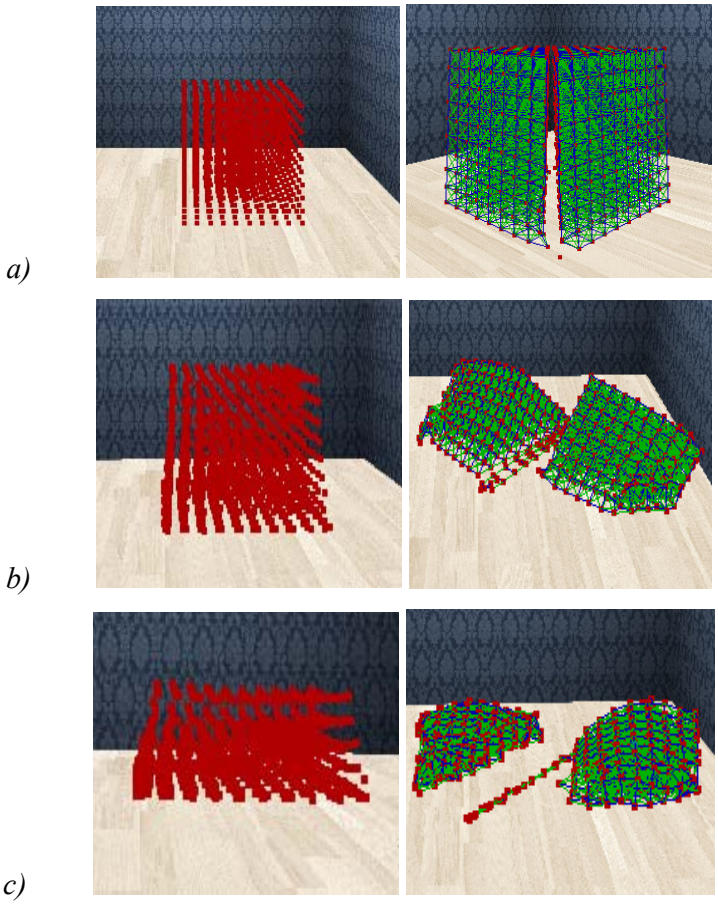


Figura 12. Operația de tăiere: a) obiect foarte rigid; b) obiect moale; c) obiect foarte moale.

Dacă obiectul este mai moale, adică are o structură mai puțin stabilă, şi tăiem cu aceeaşi lamă, cele două obiecte îşi păstrează forma dar nu şi poziția (Figura 12b). Dacă obiectul este foarte moale, adică prezintă o stabilitate

mică a structurii, în urma tăierii cu aceeaşi lamă, obiectele rezultate nu îşi păstrează nici forma şi nici poziția (Figura 12c). Aceste rezultate pot fi observate în figurile de mai jos:

5. Concluzii şi dezvoltări ulterioare

Pornind de la modelarea suprafețelor textile (Sabou Adrian, 2012), am realizat modelarea obiectelor volumetrice 3D, pentru care am construit tehnici particularizate de interacțiune, precum este operația de tăiere.

În această lucrare s-au prezentat simularea şi interacțiunea cu obiectele volumetrice 3D dinamice modelate prin particule. Utilizatorul poate interacționa în timp real cu obiectul 3D definit prin intermediul tehnicilor descrise mai sus. Experimentele efectuate au confirmat viabilitatea soluțiilor propuse.

Ca direcții viitoare de cercetare ne propunem să experimentăm simularea şi interacțiunea cu obiecte 3D deformabile pe arhitecturi distribuite, precum clusterele grafice, în vederea îmbunătățirii performanțelor şi pentru a obține o soluție scalabilă.

Referințe

Arnab S., Vinesh Raja (2013, August). A deformable surface model with volume preserving springs. *Proceedings of the 5th International Conference on Articulated Motion and Deformable Objects (pp. 259-268). New York: Springer*

Barbic, J. (2010, Noiembrie 5). Simulating a Jello Cube. *MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) : http://people.csail.mit.edu*

Bose, M. , Rajagopala, V. (2012, Iulie 30). Physics engine on reconfigurable processor — Low power optimized solution empowering next-generation graphics on embedded platforms. *Computer Games (CGAMES), 2012 17th International Conference, Louisville, KY*

Bourke, P. (1998, Februarie). Particle System Example. http://paulbourke.net/miscellaneous/particle/

Copândean D., Sabou A., Gorgan D. (2013, Septembrie). Interacțiunea utilizatorului cu obiecte 3D modelate prin particule. *Conferința Națională de Interacțiune Om-Calculator, Cluj-Napoca, România*

David B., A. W. (1998). Large Steps in Cloth Simulation. *COMPUTER GRAPHICS Proceedings, Annual Conference Series, (p. 12). Orlando.*

Firth, P. (2011, Aprilie 6). Physics engines for dummies. *Wildbunny: http://www.wildbunny.co.uk/blog/2011/04/06/physics-engines-for-dummies/*

Fredrik, F. (2011, Iunie). Real-Time Rigid Body Interactions. *Norwegian University of Science and Technology, Department of Computer and Information Science*

Georgii J., Echtler F., Westermann R., (2005). Interactive Simulation of Deformable Bodies on GPUs . Computer Graphics and Visualization Group. *Technische Universität München, Germany*

Hong M., Jung S., Min H. C., Welch S. (2005, Iulie). Fast Volume Preservation for Realistic Muscle Deformation. *Proceedings of the ACM SIGGRAPH 2005 Sketche*

Muller M., J. S. (2008, August 14). Real Time Physics Class Notes. *Los Angeles, California, USA.*

Munshi, A. (2009, Iunie 10). The OpenCL Specification. *Khronos: http://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf*

Pawasauskas J. (1997, Februarie 17). Volume Visualization With Ray Casting. *CS563 - Advanced Topics in Computer Graphics*

Provot, X. (1996). Deformation Constraints in a Mass-Spring Model. *Institut National de Recherche en Informatique et Automatique (INRIA)* .

Sabou A. (2012, August 30). Particle based modelling and processing of high resolution and large textile surfaces. Intelligent Computer Communication and Processing (ICCP), *2012 IEEE International Conference*