Interacţiunea utilizatorului cu obiecte 3D modelate prin particule

Denisa Copândean, Adrian Sabou, Dorian Gorgan
Universitatea Tehnică din Cluj-Napoca
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 asigurată interacţiunea dintre utilizator si obiectele modelate. 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 tăierea în volum. De asemenea sunt descrise conceptele si tehnicile care stau la baza acestor interacţiuni, precum tehnica *bounding volume* şi implementarea instrumentului virtual care realizează tăierea. 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.

Clasificare ACM

I.: Computing Methodologies, I.3: COMPUTER GRAPHICS, I.3.6 Methodology and Techniques, Interaction techniques;

INTRODUCERE

Una din tendinţele graficii pe calculator este aceea de a imita 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 va prezenta 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 de un script care se comportă de fiecare dată la fel având astfel un caracter previzibil.

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

S-a realizat optimizarea operaţiei de tăiere care este extrem de costisitoare din punctul de vedere al resurselor. Am propus spre implementare metoda *bounding volume*. Modul în care această tehnică este utilizată pentru a aduce îmbunătăţiri operaţiei de tăiere este prezentat în secţiunea "Optimizarea operaţiei de tăiere: bounding volume".

Se vor pezenta o serie de lucrări asemenătoare în secţiunea "LUCRĂRI ASEMĂNĂTOARE". Va urma apoi în secţiunile "SIMULAREA MODELULUI 3D" şi "MODELAREA DIZPOZITIVULUI DE TĂIERE", o prezentare a modelului 3D manipulat, a motorului fizic în care are loc simularea şi a modelului lamei de tăiat. Ulterior, se vor putea observa rezultatele experimentelor realizate prin interacţiunea utilizatorului cu aplicaţia în secţiunea "EXPERIMENTE – OPERAŢIILE DE INTERACŢIUNE", precum şi concluziile obţinute în secţiunea "CONCLUZII ŞI DEZVOLTĂRI ULTERIOARE".

LUCRĂRI ASEMĂNĂTOARE

Având drept scop familiarizarea cu animaţia bazată pe legile fizicii, Barbic [1] prezintă o formă simplă de modelare 3D bazată pe particule, iar Bourke [2] realizează o scurtă introducere în domeniul sistemelor de particule. Matthias Muller et al [3] prezintă în notiţele sale de curs modul în care simularea este îmbunătăţită prin integrarea ecuaţiilor fizice în metodele şi algoritmii actuali de simulare. David Baraff et al. [4] prezintă avantajele utilizării metodelor de integrare numerică implicită in simularea textilelor. Autorii au demonstrat că integrarea implicită poate să ofere performanţe superioare prin faptul că permite paşi de simulare mult mai mari fară a suferi de instabilitatea inerentă integrării explicite. De aceea, am optat pentru o modelare bazată pe particule procesate la nivelul unităţilor grafice.

Pornind de la ideea de modelare pe GPU descrisă de Sabou [5] am extins motorul fizic de simulare utilizat în scopul de a manipula nu doar obiecte bidimensionale ci şi tridimensionale. Detalii legate de această idee sunt prezentate în secţiunea "Motorul fizic".

SIMULAREA MODELULUI 3D

Corpurile din viaţa reală pot fi aproximate, într-o măsură mai mare sau mai mică, prin sisteme de puncte discrete. Pentru a se menţine ca o entitate stabilă, între constituenţii 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 și forțe externe. Forțele de interacțiune dintre perechile de particule constituente ale sistemului se numesc forțe interne. Forțele de interacțiune dintre particulele sistemului și alte corpuri sau câmpuri de forțe din exteriorul acestuia se numesc forțe externe.

Modelul 3D

Structura modelului fizic

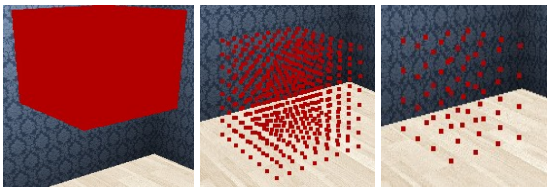
Cel mai simplu model de particule pentru simularea unui obiect volumetric dinamic este modelul mass-spring [3]. Acest model este format din particule. Particulele sunt caracterizate de masă, poziție și viteză și sunt interconectate între ele prin arce. O particulă poate fi un punct de masă care are o anumită poziție și o anumită viteză la un anumit moment de timp. Modul în care arcele conectează particulele între ele și diferența de putere a fiecărui arc influențează comportamentul obiectului ca un tot unitar.

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-un mod triunghiular sau într-un mod patrulater. [1] Pentru a recrea cele trei proprietăți ale unui obiect volumetric și anume rezistența la întindere, îndoire și tăiere avem nevoie să folosim trei tipuri de arcuri. Fiecare tip de arc modelează una din cele trei tipuri de comportament ale unui obiect volumetric real. Tipul de arc se referă la rigiditatea arcului care va fi diferită pentru fiecare tip și, ce este mai important e faptul că fiecare tip de arc conectează particulele într-un mod diferit.

Primul tip, numit arc structural (*structural spring*), definește structura de bază a obiectelor volumetrice și modelează faptul că obiectele reale nu se întind foarte mult. Aceste arcuri simulează faptul că un obiect volumetric este alcătuit din fibre longitudinale și transversale. Al doilea tip de arcuri sunt numite arcuri de rupere (*shear springs*) și fac ca rezistența obiectului volumetric la rupere să fie mai mare. Aceste arcuri ajută obiectul volumetric să păstreze o anumită formă și nu lasă particulele să cadă una în cealaltă, să se apropie sau să se îndepărteze prea tare. Aceste arcuri conectează particulele pe diagonală în fiecare dreptunghi format de catre arcurile structurale. Al treilea tip de arcuri sunt sunt arcurile de îndoire (*bend springs*). Aceste arcuri 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 din cei trei parametri care descriu un model volumetric 3D: pasul (densitatea), dimensiunea și numărul de particule. Metoda utilizată de aplicația descrisă în această lucrare este cea care utilizează pasul și dimensiunea. Având acești doi parametri ca și intrare deducem numărul de particule. Deoarece se simulează un obiect volumetric 3D, dimensiunea este dată de trei variabile: lungimea obiectului, lățimea obiectului și înălțimea obiectului. Pasul care redă densitatea unui obiect se aplică în aceeași măsură atât pe lungime, cât și pe lățime și înălțime. Dacă pasul este mare, obiectul volumetric nu este foarte dens. În schimb, dacă pasul este mic, obiectul volumetric este foarte dens.



a) b) c)

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

Astfel, numărul de particule care alcătuiesc modelul mass-spring se determină după cum urmează:

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

Ecuația 1. Formula pentru determinarea numărului total de particule.

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

Motorul fizic

Pentru înglobarea de legi fizice în cadrul unei aplicații grafice în timp real este folosit de obicei un motor fizic (*physics engine*)[6].

Distribuția paralelă

Motorul fizic [5] a fost implementat și extins folosind calculul paralel la nivel de model bazat pe multithreading. Tehnologia care oferă suport de multithreading pe acceleratoare hardware grafice și pe care am ales-o pentru a implementa motorul de simulare este OpenCL.

OpenCL [7] (Open Computing Language) este un cadru de lucru pentru scrierea de programe care se execută pe platforme eterogene formate din unități centrale de procesare (CPU), unități de procesare grafică (GPU) și alte procesoare. Acesta include un limbaj (bazat pe C99) pentru a scrie *kernele* (funcțiile care se execută pe dispozitive OpenCL), plus interfețe de programare a aplicațiilor (API), care sunt folosite pentru a defini și pentru a controla apoi platformele. 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 și rezistența vântului. Poate simula și prezice efecte sub diferite condiții care aproximează ce se întâmplă în realitate.

De asemenea motorul permite modelarea și 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 de OpenCL. Acest nucleu realizează atât procesul de simulare unde are loc integrarea în timp real al forțelor ce acționează asupra modelului, cât și detecția și răspunsul la coliziune.

Integrarea numerică în timp real

Acest proces se realizează parcurgând următorii pași:

a) simularea fizică a dinamicii unui sistem de particule de tip mass-spring se poate realiza parcurgând următorii pași: cumularea tuturor forțelor ce acționează asupra particulelor atât forțele ce apar în interiorul rețelei de particule de la arcurile care conectează particulele între ele cât și forțele externe precum gravitație, vânt, forțe ce acționează asupra unor particule, etc. Forțele externe le includ și pe cele ce rezultă în urma ciocnirii sau a interacțiunii cu alte obiecte din mediu.
b) calcularea accelerației particulelor rezultate din aceste forțe.
c) Actualizarea vitezelor particulelor și pozițiile acestora folosind accelerațiile calculate.

Integrarea în timp real este dată de legea a doua de mișcare a lui Newton calculând accelerația fiecărei particule la un moment dat. Aceasta este implementată prin metoda Verlet [8]:

$$x(t+\Delta t) = x(t) + v(t) * \Delta t + \frac{a(t) * \Delta t^2}{2}$$

Ecuația 2. Formula poziției

unde *x* reprezintă poziția, *v* viteza, *t* timpul curent, *Δt* intervalul de timp și *a* accelerația.

În implementarea de față vitezele se calculează simplificat, folosind formula:

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

Ecuația 3. Formula vitezei

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 și răspuns. O dată detectată coliziunea, e necesar să se aplice răspunsul de îndreptare, de corecție al coliziunii. Acest răspuns se mai numește și rezoluție a coliziunii.

Metoda de rezoluție aplicată în lucrarea de față este cea a impulsului. Se pornește de la principiul de conservare a impulsului:

$$p = m * v$$

Ecuația 4. Formula impulsului

unde *p* reprezintă impulsul, *m* masa și *v* viteza.

În implementarea curentă, coliziunea 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.

MODELAREA DISPOZITIVULUI DE TĂIERE

Definirea lamei

Lama reprezintă obiectul virtual de tăiere în cadrul simulării. Precum un cuțit adevărat, lama are un singur tăiș. Acest tăiș este definit ca o dreaptă dată de două puncte: punctul de început (punctul A) și punctul de sfârșit (punctul B). O dată definit tăișul se poate determina forma fizică a lamei. Pornind de la punctul A și punctul B se calculează punctul A' și punctul B' știind că lama are lățimea 3.

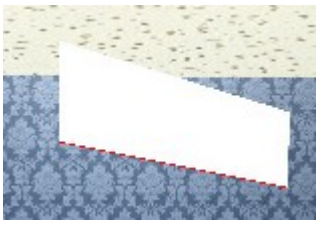


Figura 2. Modelul lamei

Totuși, înainte de a calcula punctele A' și B' tebuie determinată direcția. Astfel se alege un punct D din spațiu, ce va determina direcția lamei. Acest punct va fi proiectat pe tăișul lamei, după care se va calcula piciorul perpendicularei ce trece prin punctul D. Perpendiculara reprezintă direcția lamei și pe baza ei se vor calcula punctele A' și B'. Punctele A' și B' se determină utilizând noua direcție și valoarea lățimii:

$$A' = A + d * l$$
$$B' = B + d * l$$

Ecuația 5. Determinarea punctului aflat pe o anumită direcție (d) și la o anumită distanță (l) față de un punct dat

Deplasarea lamei

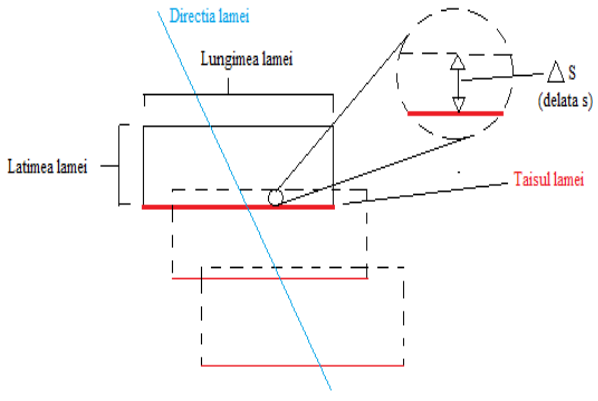


Figura 3. Deplasarea lamei

După cum se poate observa și în figura 4, Δs (delta s) reprezintă pasul de deplasare al 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 galben se aplică algoritmul de tăiere de mai multe ori, chiar dacă ea a fost analizată o dată.

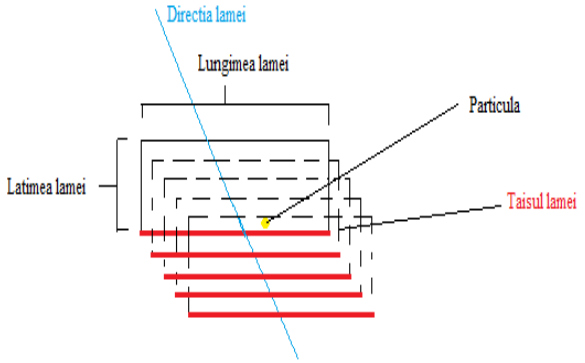


Figura 4. Deplasarea lamei cu Δs mic

Dacă Δs este foarte mare unele particule nu sunt testate, iar tăierea nu se realizează corespunzător:

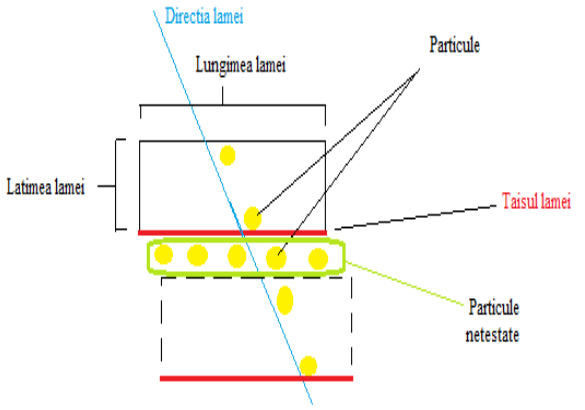


Figura 5. Deplasarea lamei cu Δs mare

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

Controlul interactiv al deplasării lamei

În timpul simulării se poate schimba oricând direcția de deplasare a lamei. Pentru a avea control prin manipularea directă a mișcării lamei de tăiat, este necesar doar redefinirea punctului D în spațiu și reluarea calculelor pentru determinarea perpendicularei dreptei AB ce trece prin punctul D.

Lama taie doar pe direcția definită de perpendiculara DC, însă ea poate fi poziționată în spațiu și se poate deplasa și pe sensul opus direcției DC dar fară a tăia.

De asemenea, în timpul simulării se poate schimba și pasul de deplasare al lamei.

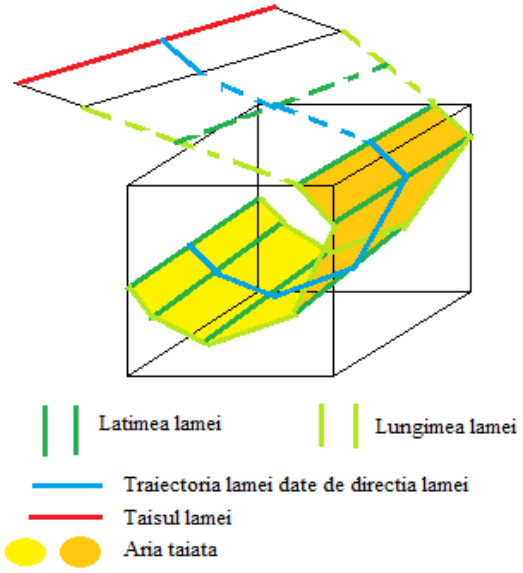


Figura 6. Controlul interactiv al deplasării lamei

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.

Optimizarea operației de tăiere: bounding volume

Algoritmul de tăiere este unul foarte costisitor atât în timp cât și în resurse, deoarece utilizează foarte multe operații de înmulțire și împărțire și 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 și 6 vecini de îndoire). Pentru a optimiza algoritmul de tăiere trebuie să optimizăm analiza particulelor.

Se știe 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.

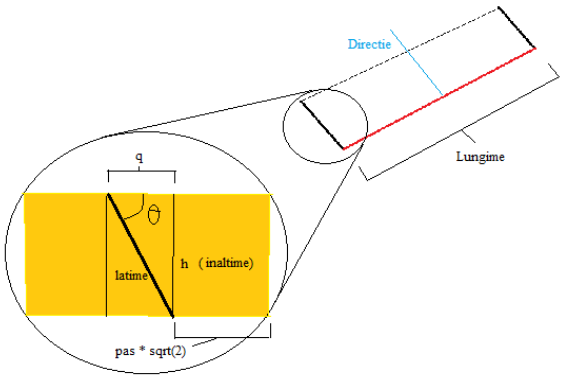


Figura 7. Definirea extensiei lamei

$$h = l * \sin(\theta)$$
$$q = l * \cos(\theta)$$

Ecuația 6. Determinarea parametrilor definitorii ai extensiei lamei

Volumul extensiei va fi:

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

Ecuația 7. Volumul extensiei

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. Și așa sunt multe particule de testat. De aceea s-a mai intodus încă o optimizare în ceea ce privește 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 arcurile de forfecare și densitatea (pasul)/2 pentru arcurile structurale și de îndoire, atunci se va aplica algoritmul de tăiere, în caz contrar particula e prea departe și nu are legături ce trebuie tăiate.

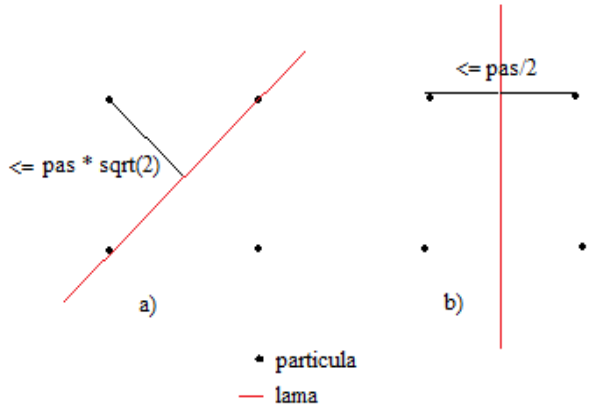


Figura 8. Optimizare bazată pe distanță

EXPERIMENTE – OPERAȚIILE DE INTERACȚIUNE

Selecția

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

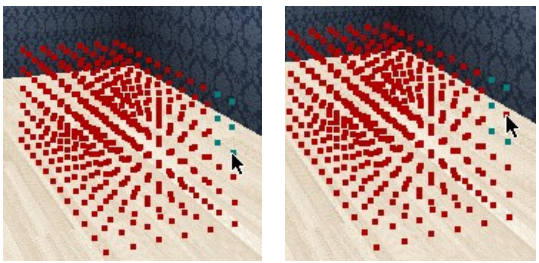
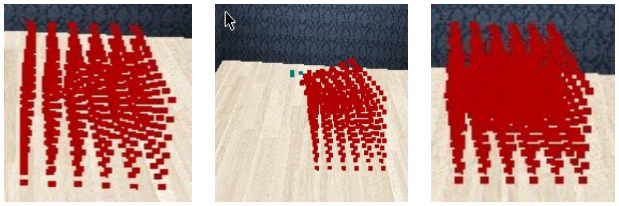


Figura 9. Selecție/Deselecție

Prindere&Tragere

Această tehnică se bazează pe tehnica de selecție. Se poate realiza ținând apăsat butonul stânga al dispozitivului maus și modificând poziția cursorului. Modelul își modifică poziția în funcție de forța imprimată.



a) b) c)

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

Definirea lamei

Se bazează pe clicuri consecutive în scenă: primul definește punctul de început al tăișului, al doilea punctul de sfârșit al tăișului, iar al treilea punctul care oferă direcția lamei.

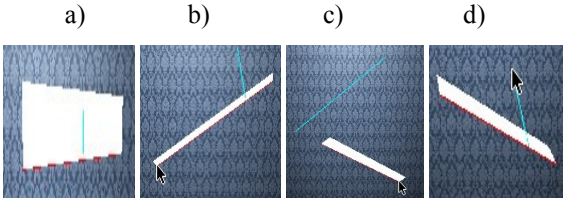
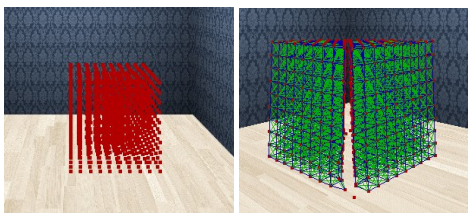


Figura 11. Definire lamă: a) inițială; b) modificare început; c) modificare sfârșit; d) modificare direcție.

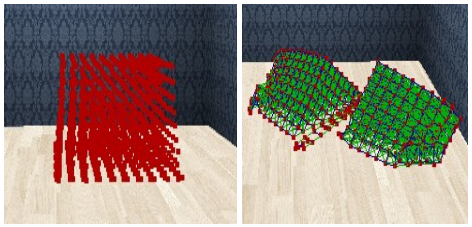
Operația de tăiere

În funcție de parametrii săi definitori, 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 și pe diagonală) rezultă două obiecte disjuncte care își păstrează forma și poziția (Figura 12.a). Dacă obiectul este mai moale, adică are o structură mai puțin stabilă, și tăiem cu aceeași lamă, cele două obiecte își păstrează forma dar nu și poziția (Figura 12b). Dacă obiectul este foarte moale, adică prezintă o stabilitate mică a structurii, în urma tăierii cu aceeași lamă, obiectele rezultate nu își păstrează nici forma și nici poziția (Figura 12c). Aceste rezultate pot fi observate în figurile de mai jos:

a)



b)



c)

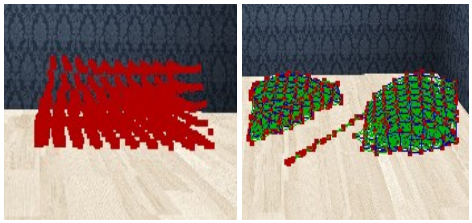


Figura 12. Operația de tăiere: a) obiect foarte rigid; b) obiect moale; c) obiect foarte moale.

CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE

Pornind de la modelarea suprafețelor textile [5], 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 și 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 și interacțiunea cu obiecte 3D deformabile pe arhitecturi distribuite, precum clusterele grafice, în vederea îmbunătățirii performanțelor și pentru a obține o soluție scalabilă.

REFERINŢE

1. Barbic, J. (2010, Noiembrie 5). Simulating a Jello Cube. Retrieved from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) : http://people.csail.mit.edu
2. Bourke, P. (1998, Februarie). Particle System Example. Retrieved from Paul Bourke: http://paulbourke.net/miscellaneous/particle/
3. Matthias Muller, J. S. (2008, August 14). Real Time Physics Class Notes. Los Angeles, California, USA.
4. David Baraff, A. W. (1998). Large Steps in Cloth Simulation. COMPUTER GRAPHICS Proceedings, Annual Conference Series, (p. 12). Orlando. http://chi2007.ist.psu.edu/submit/archivesubguide.php
5. Sabou Adrian (Aug. 30 2012). Particle based modelling and processing of high resolution and large textile surfaces. Intelligent Computer Communication and Processing (ICCP), 2012 IEEE International Conference
6. Physics engine. (2005, Ianuarie 31). Retrieved from From Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Physics_engine
7. Munshi, A. (2009, Iunie 10). The OpenCL Specification. Retrieved from Khronos: http://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf
8. L. Verlet, „Computer experiments on classical fluids. I. Thermodynamical properties of lennard-jones molecules", Phys. Rev., vol. 159, pp. 98{103, Jul 1967. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRev.159.98