LINKÖPINGS UNIVERSITET
Institutionen för Teknik och Naturvetenskap

TNM085 Modelleringsprojekt

Fluid Simulation

14 mars 2011

Robert Novo, robno767@student.liu.se
Martin Persson, marpe357@student.liu.se
Mattias Persson, matpe621@student.liu.se
Johannes Ullström, johul223@student.liu.se

Examiner
Anna Lombardi

Sammanfattning

Målet för detta projekt var att skapa ett interaktivt program som simulerar en fluid. Fluidsimulationen använder beräkningsmetoden SPH. Slutprodukten, alltså programmet, utvecklades med hjälp av ramverket Microsoft XNA. Nyckelord: simulering, fluider, Navier-Stokes, C#, XNA.

Innehåll

Figurer

1 Introduktion

Simuleringar av olika naturliga fenomen är ett växande område inom många olika grenar, till exempel vetenskaplig forskning, spelutveckling och specialeffekter i film. Gruppen fann simulering av hur fluider beter sig instressant och målet med projektet blev därav att utveckla en interaktiv mjukvara som simulerar fluider i 3D.

1.1 Bakgrund

Att beskriva och simulera olika fysiska fenomen har alltid varit en viktig gren inom vetenskapliga studier. Visualiseringar av fluider är ett område som är väl utforskat, men än finns många problem kring det som inte är lösta. Två välanvända metoder för att skapa visualiseringar av fluider är rutnäts -och partikelbaserade visualiseringar. I detta projekt har en partikelbaserad metod används, vilken är baserad på Navier-Stokes ekvationer.

1.2 Användningsområden

Det finns många sätt att implementera fluidsimuleringar.

1.2.1 Datorspel

I de senaste datorspelen är det vanligt med avancerade fluidsimuleringar av exempelvis vatten och rök. För att förbättra renderingshastigheten och den visuella kvaliteten på simuleringar, bortser man ofta från vissa fysiska parametrar. För att ytterligare pressa ut så bra upplevelse som möjligt till användaren, används grafikkortsberäkningar, vilka skulle vara alldeles för krävande att göra på datorns processor.

1.2.2 Effekter i film

Till skillnad från simuleringar i datorspel så kräver sällan effekter till film att de kan göras i realtid. Detta gör man kan ta hänsyn till fler fysikaliska parametrar, vilket ger en mer verklighetstrogen men mer beräkningstung simulering. Simuleringar i film ser därför i regel bättre ut än simuleringar i spel och andra interaktiva applikationer.

2 Metoder

Som grund till projektet ligger Mullers rapport från 2003 vilken handlar om parikelbaserad fluidsimulering. Arbetet har i stor mån gått ut på att implementera teorin som presenteras i rapporten. Nedan följer en beskrivning av de metoder som har använts i projektet.

2.1 Navier-Stokes ekvationer

En välanvänd metod för att beskriva fluider är Navier-Stokes ekvationer, som är framtagna av Claude-Louis Navier och George Gabriel Stokes. Navier-Stokes tillämpar Newtons andra lag för att beskriva hur stabila fluider flödar och genererar en serie av differentialekvationer. Navier-Stokes ekvationer gör det möjligt att illustrera hur storheter som kraft, densitet och viskositet påverkar fluiden.

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \rho g + \mu\nabla^2 v \tag{1}$$

Ekvationen förenklas enligt Müller03:

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{-\nabla p_i + \rho_i \mathbf{g} + \mu\nabla^2 \mathbf{v}_i}{\rho_i} \tag{2}$$

där $-\nabla p_i$ är kraften från tryck, $\rho_i \mathbf{g}$ är yttre krafter såsom gravitation och $\mu\nabla^2 \mathbf{v}_i$ är viskositetskraften.

2.2 SPH, Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) utvecklades av Lucy och Gingold och är till början en metod för att simulera astrofysiska problem, men lämpar sig även till att simulera vätskor. I en artikel skriven av Müller, Charypar and Gross (2003) så utvecklas denna metod ytterligare. Müller med andra kommer fram till att för att bestämma hur varje partikel rör sig är det nödvändigt att inkludera en smoothing kernel-metod, som regulerar stabiliteten, noggrannheten och hastigheten av SPH. Enligt SPH är en skalärkvantitet *A* interpolerad vid plats *r* med en vägd summa av bidrag från alla partiklar.

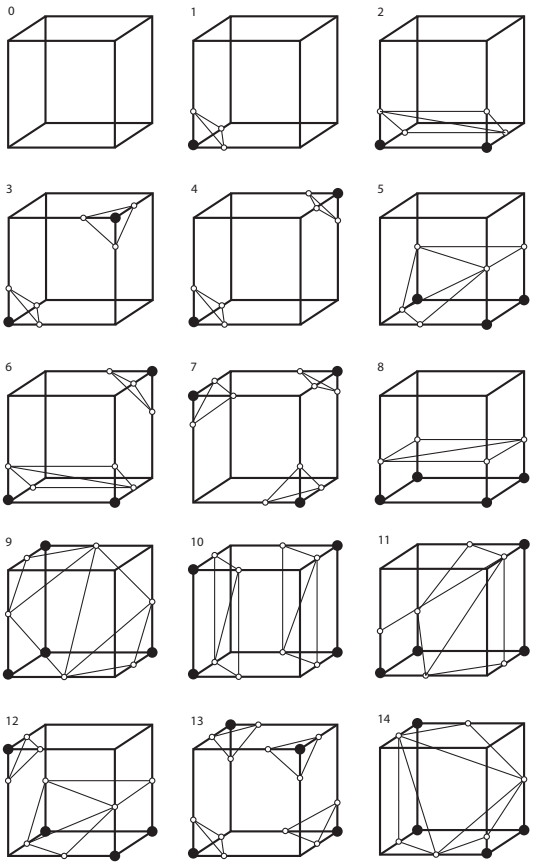
$$A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \tag{3}$$

där m_j är massan och ρ_j är densiteten av partikel j. Funktionen W är en viktfunktion kallad smooth kernel".

2.3 Marching Cubes

Marching Cubes är en algoritm utveckld av Cline och Lorensen år 1987, som smälter samman partiklar för att skapa en yta, en så kallad mesh". Marhing Cubes använder

256 kubinställningar för att representera alla möjliga kombinationer som en mesh kan korsa kuben. Genom att utnyttja symmetri är det möjligt att reducera kombinationerna till 15 unika mönster.



Figur 1: Marching cubes

2.4 C#/.Net och XNA

Den exekverbara filen för det här projektet skrevs i programmspråket C# och ramverket Microsoft .NET. Microsoft XNA används för att underlätta det grafiska utseendet.

3 Systembeskrivning

3.1 Densitet

Första steget i beräkningen utav krafterna är att räkna fram partiklarnas densitet. Genom att att ersätta A i ekvation 3 med partikeldensiteten ρ fås:

$$\rho_S(\mathbf{r}) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h) \tag{4}$$

Viktfunktionen W som används vid den här beräkningen är W_{poly6}.

3.2 Tryck

Liksom i Müller03 används en förenkling utav den ideala gaslagen för att beräkna trycket hos en partikel enligt:

$$p = k(\rho - \rho_0) \tag{5}$$

Där ρ är densiteten hos partikeln, ρ_0 är fluidens vilodensitet och k är en gaskonstant som beror på fluidens temperatur.

Sedan används nedanstående ekvation för att beräkna den resulterande kraften.

$$\mathbf{f}_i^{pressure} = -\sum_j m_j \frac{p_i + pj}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{6}$$

W som används för ekvationen ovan är:

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6}\{(h - r)^3 \tag{7}$$

3.3 Viskositet

Viskositet kan ses som fluidens interna friktion och den bidragande kraften beräknas enligt:

$$\mathbf{f}_i^{viscosity} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{8}$$

Viktfunktionen W som används i den här beräkningen är:

$$W_{viscosity}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \frac{-\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1}{0} \tag{9}$$

3.4 Ytspänning

Trots att ytspänning inte är med i Navier-Stokes ekvationer är det inkluderat i Muller03 och således även i den här simuleringen. Ytspänningen beräknas i tre steg:

$$c_S(\mathbf{r}) = \sum_j \frac{m_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \tag{10}$$

$$\mathbf{n} = \nabla c_S \tag{11}$$

$$\mathbf{f}^{surface} = -\sigma \nabla^2 c_S \frac{\mathbf{n}}{|\mathbf{n}|} \tag{12}$$

W som använts i ekvation 10 är W_{poly6}.

3.5 Slutliga beräkningar

När krafterna är beräknade sätts dessa in i ekvation 2, vilket ger partikelarnas acceleration. Partiklarnas hastighet uppdateras med accelerationen och multipliceras med en tidskonstant. Hastigheten används sedan för att uppdatera partiklarnas position.

4 Implementation

Som redan nämnts ovan användes programmeringsspråket C# för att skriva koden och editerarmjukvaran Microsoft Visual Studio Pro 2010.

4.1 Grannpartiklar och krafter

Mjukvarans huvudfunktion går ut på att beräkna krafterna som inverkar på partiklarna. Mängden krafter som inverkar på varje partikel beror på de omgivande partiklarna. Finnandet av varje grannpartikel blir därför en mycket viktig del av algoritmen. En partikel i anses vara en grannpartikel till en partikel j om avståndet mellan dem är mindre än ett förbestämt avstånd. Avståndet mellan funna grannpartiklar bestämmer hur de kommer inverka på varandra.

4.2 Kollisionshantering

. Kollisionshanteringen i denna simulering är förhållandevis enkel. Vätskan rör sig inom en förbestämd gräns. Efter att en partikels nya position är beräknad, kontrollerar programmet om partikeln är inom gränsen eller ej. Om partikeln är utanför gränsen inverteras hastigheten och dess nya position blir satt till positionen där den förmodligen skulle befinna sig vid före partikeln gick över gränsen, annars ändras inget.

4.3 Att ändra simulationsbeteende

För att göra simuleringen interaktiv lades dragreglage till simuleringens användargränssnitt. Reglagen ändrar värdet på konstanter som används i uträkningen av krafterna i simuleringen.

4.4 Rendering

XNAs fördelar blir tydliga då simuleringen ska renderas. En "kameraskapas för att fånga den visuella implementeringen av simuleringen. XNA Studio tillhandahåller sshaderssom bidrar till den visuella effekten utan att behöva codas manuellt.

5 Resultat

Slutprodukten av detta projekt är en exekverbar .NET-applikation. Detta betyder att programmet kan köras på vilken dator som helst, så länge datorn har .NET ramverk installerat, tillsammans med XNA Studio. Skärmdumpar från den resulterande applikationen av detta projekt presenteras i bilagsdelen.

Figur 2 demonstreras programmets allmänna utseende. I mitten finns ett antal blå partiklar inneslutna i en transparent kub. I övre vänstra hornet finner användaren statistik från programmets beräkningar och anvisningar hur programmet ska styras. I övre högra hörnet finns det sex dragreglage och en knapp.

Figur 3 visar en förstorad bild av övre vänstra hörnet av användargränssnittet. Etiketten FPS står för bildrutor per sekund och är ett mått på hur många bildrutor som visas på skärmen per sekund. Ett högre FPS-värde ger simuleringen ett mer kontinuerligt utseende än vad ett lågt värde ger. Utseendet av programmet kommer att variera då olika datorer har olika specifikationer och ger ett högre eller lägre FPS då partiklarna är i rörelse. Stopwatch står för tidtagarur och visar hur lång tid uppdateringen av fluiden tar. Etiketten Particles visar antalet partiklar i simuleringen. Camera Position har tre rumsvariabler: X, Y och Z, som mäter kamerans position från kubens centrum. Användaren kan kontrollera kameran genom att använda pilarna samt <q> och <e> på tangentbordet. Användaren kan också skifta fullskärm av och på genom att trycka på <F6>, samt skifta nätet på och av genom att trycka på <tab>. Då användaren trycker på <tab> skiftas marching cubes av och på.

Figur 4 visar en förstorad bild av övre högra hörnet av användargränssnittet. Användaren kan dra i de tre översta dragreglagen för att styra fluidens parametrar. Dragreglaget nämnt Particles styr hur många partiklar som ska simuleras. Timestep styr simuleringens hastighet. Drar användaren I Rotation Speed så roteras kuben medurs eller moturs. Det finns också en Restart-knapp som återstartar simuleringen, användarens ändringar på de olika reglagen bibehålles.

6 Slutsats

Att skapa realistiska simuleringar är svårt. Likaså att bestämma vad som anses vara en brasimulering. I detta projekt har gruppen utgått från det resultat som Beaudoin, Clavet och Poulin arbetade fram år 2005, vilket vi ansåg vara en enastående simulering. Det har varit ett mål att sträva efter deras resultat och vi tycker att är en god bit på vägen.

Det finns oftast saker man önskar man hade gjort annorlunda. Att använda sig av ett programmeringsmiljö med andra språk, såsom t.ex. OpenGL istället för XNA.

Vid en eventuell vidareutveckling av projektet skulle en naturlig övergång vara att göra alla beräkningar på grafikkortet. Detta på grund av att ett grafikkort betydligt fler kärnor en vad en processor gör, vilket resulterar i fler parallella beräkningar. Grafikkort-sutvecklarna NVIDIA har tagit fram en grafikmotor vid namn Compute Unified Device Architecture (CUDA) som finns tillgänglig på större delen av deras grafikkort. Tack vare CUDA så är det alltså möjligt att kraftigt öka hastigheten på simuleringen, dvs. antalet bildrutor per sekund. Hade vi använt oss av CUDA så hade en metod vid namn Point Splatting"varit möjlig istället för marching cubes. Problemet med point splatting är att den kräver 10.000 till 100.000 partiklar, vilket hade kraftigt försvårat arbetet och mer eller mindre krävt en CUDA-implementering.

En annan del av projektet som hade kunnat förbättras är kollisionshanteringen för att låta fluiden interagera med andra objekt.

7 References

Referenser

[1] Müller M., Charypar D., Gross M. *Particle based fluid simulation for interactive applications* 2003.

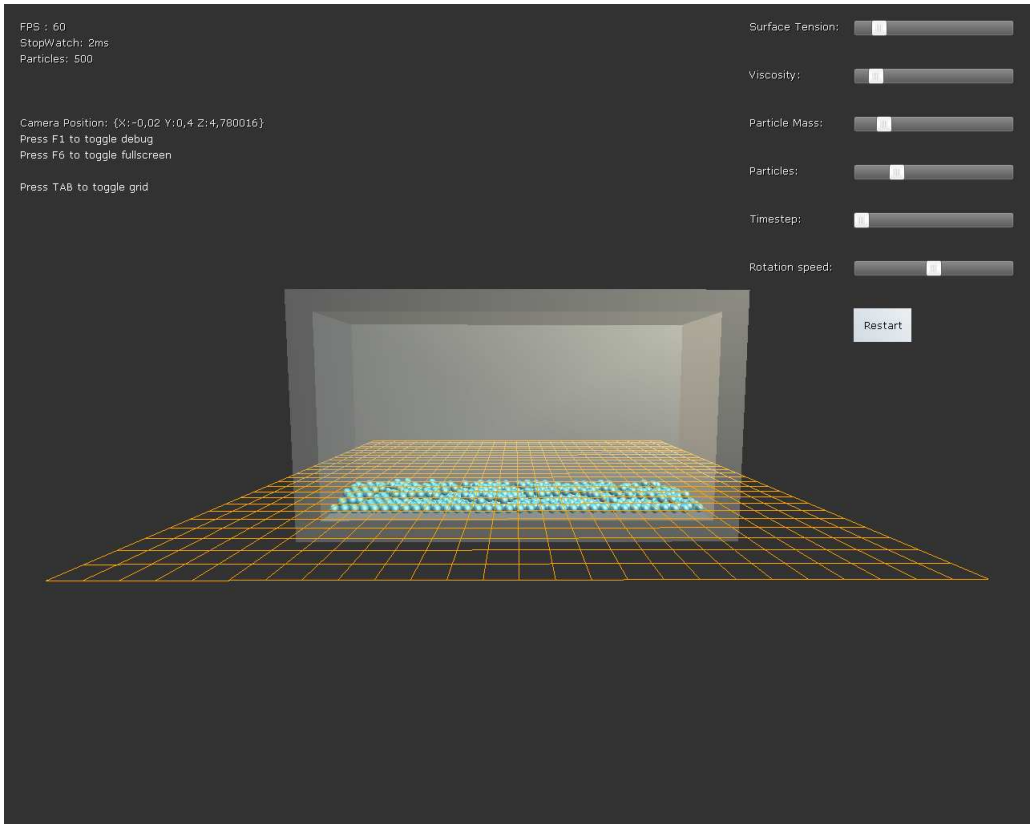
[2] Lucy L. B. *A numerical approach to the testing of the fission hypothesis.* The Astronomical Journal, 82:1013-1024, 1977. http://adsabs.harvard.edu/full/1977AJ.....82.1013L

[3] Gingold R. A., Monaghan J. J. *Smoothed Particle hydrodynamics: theory and application t non-spherical stars.* Monthly Notices of the Royal Astronomical Society, 181:375-389, 1977. http://adsabs.harvard.edu/full/1977MNRAS.181..375G(2011-03-09)

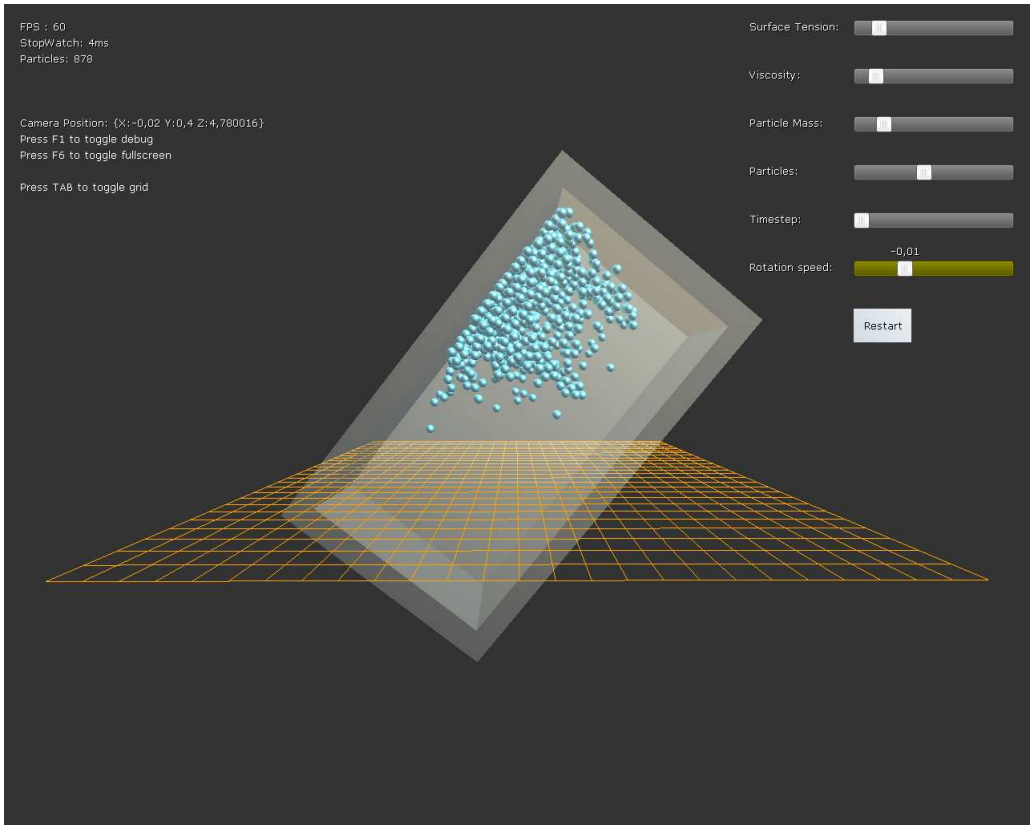
[4] Cline H. E., Lorensen W.E. *Marching Cubes: A high resolution 3D surface construction algorithm.* SIGGRAPH, 1987. http://kucg.korea.ac.kr/seminar/2001/src/PA-01-16.pdf(2011-03-09)

[5] Beaudoin P., Clavet S., Poulin P. *Particle-based Viscoelastic Fluid Simulation.* http://www.iro.umontreal.ca/labs/infographie/papers/Clavet-2005-PVFS/pvfs.pdf(2011-01)

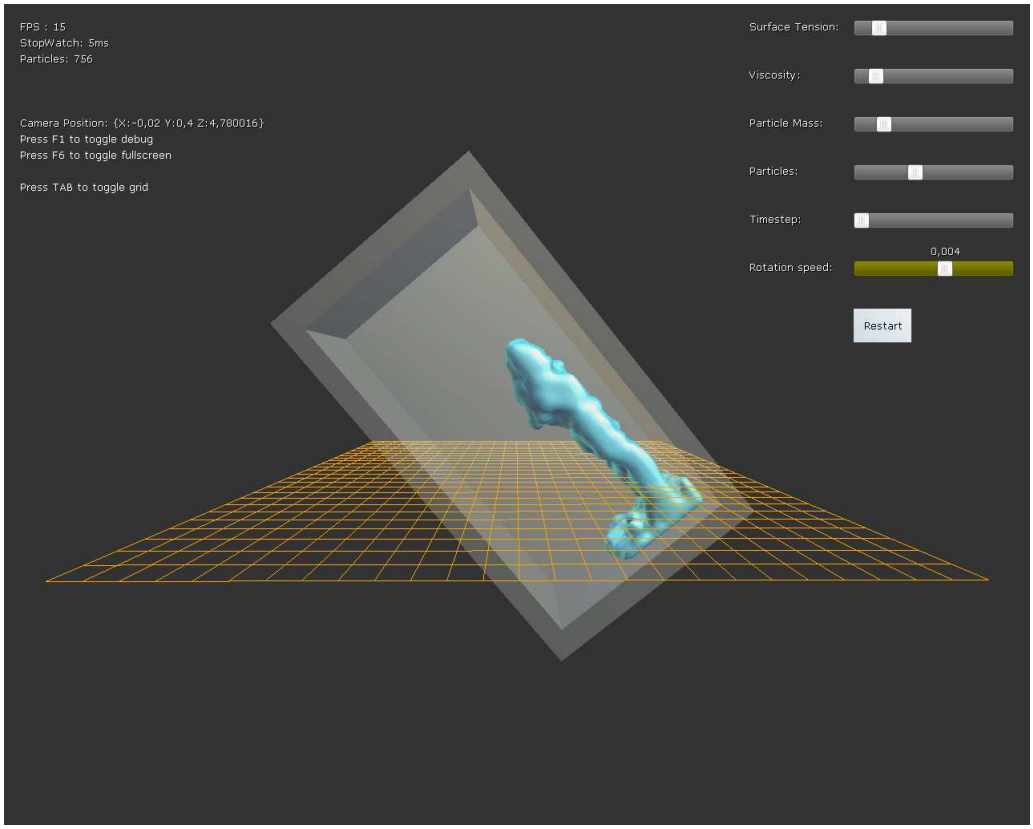
A Skärmdumpar



Figur 2: Bild 1



Figur 3: Bild 2



Figur 4: Marching cubes