18. MARTS 2022

SIMULERING AF VÆSKEDYNAMIK

EN SOP I FYSIK A OG DIGITAL DESIGN UNDERVISNING A

FREDERIK CAYRÉ HEDE-ANDERSEN
3.A2 HCØL

Indholdsfortegnelse

Abstract 3
Indledning: 3
Problemformulering 3
Hoveddel 4
    Navier-Stokes 4
    Computeren og væskedynamik som algoritmer 5
  Programmet 6
    Design 6
Referencer 8

Abstract

[skrives til sidst]

Indledning:

Computere ses alle steder i verden og bruges til alle mulige ting. Fra styring af ovne til design af havelåger til beregninger af yderligere cifre af pi, computere kan en masse. Modellering af virkeligheden bliver også gjort på computere i mange industrier rundt omkring i verden. Det at kunne opleve verden og genskabe fænomener gennem en computer, uden faktisk at skulle gå ud med et kamera og måle på tingene, kan bruges til rigtig mange ting. Det kan gøre besværet med bestemte opgaver meget mindre og det kan gøre folk i stand til at opleve ting der normalt ikke sker. Naturen er vores at kommandere inde i computeren. Specifikt inden for feltet væskedynamik er computersimuleringer ekstra brugbare, da det kan være svært at måle på mange af en væskes egenskaber kontinuert. Væsker er dog svære at simulere, og det er tungt at køre på langt de fleste computere, især hvis simuleringen skal være naturlig. Der er dog mange måder at simulere væskedynamik på, ved hjælp af forskellige algoritmer. Det er det vi kigger på i denne opgave.

Problemformulering

”Hvordan kan et program vise forskellene i simuleringer af væskedynamik, samt give indblik i relevante applikationer af simuleringer, med forskellige algoritmer.”

Denne problemformulering vil jeg svare på ved hjælp af en række underspørgsmål, samt tests af et program jeg skriver, og en bedre generel forståelse af feltet. De følgende underspørgsmål skal give os en bedre forståelse af emnet, samt få os på et rette spor når det gælder analysen.

Underspørgsmål:

- Hvad er væskedynamik og hvordan beskrives væskedynamik matematisk?
- Hvordan kan computere bruges til fluidsims?
- Hvilke algoritmer og metoder kan bruges til fluidsims?
- Hvordan kan et program designes og testes for at vise forskellene mellem forskellige fluidsims?
- Hvad er tidskompleksiteten af de implementerede algoritmer?
- Hvad kan forsøgets resultater fortælle os om algoritmerne?
- Hvordan kan algoritmerne bruges i forskellige industrier?
- Hvad fortæller forsøget om anvendelsen af fluidsims inden for computerspilsindustrien?

Note: Fluidsims = simuleringer af væskedynamik

Hoveddel

Navier-Stokes

Væskedynamik er det felt der håndterer bevægende stoffer på flydende og på gas form. Igennem lang tid har væsker været set på og undret over, for at kunne modellere deres bevægelse matematisk. Archimedes lov stammer fra denne undren om væsker, og der ville stille og roligt blive udviklet mere avancerede ideer om væskedynamik gennem århundrederne. Da oplysningstiden begyndte skete der virkelig noget, og i løbet af 1800-tallet fik to matematikere Claude-Louis Navier og George Gabriel Stokes udviklet en definitiv samling matematiske formler. Disse formler beskriver væsker med viskositet og er kendte for at være meget svære at løse for alle på nær de nemmeste væskesystemer. Vi kommer hovedsageligt til at arbejde med inkompressible væsker, da de er nemmere at have at gøre med. Luft er derfor ikke noget vi kommer til at se på, vand derimod er en inkompressibel væske, så den er mulig for os at modellere. Selvom alle væsker er en smule kompressible kommer det til at have en ubetydelig effekt i langt de fleste tilfælde.

For at holde formlerne nogenlunde overskuelige og for at gøre det nemmere at implementere kigger vi kun på inkompressible væsker. Et simpelt væskesystem kan derfor beskrives ved hjælp af et sæt af Navier-Stokes formlerne der ikke er så tunge. De følgende formler kan bruges til at beskrive et væskesystem med en inkompressibel væske.[1]

$$\nabla \cdot \vec{u} = 0 \qquad (1)$$

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho_0}\nabla p + \nu \nabla^2 \vec{u} + \vec{f} \qquad (2)$$

Den øverste af formlerne (1) beskriver hvordan masse konserveres gennem væskesystemet. Da væsken er inkompressibel må densiteten af væsken være den samme over alle punkter. Formlen (1) beskriver hvordan der ikke kan flyde mere eller mindre væske ind i et punkt end der flyder væske ud af punktet. ’∇’ og ’$\nabla \cdot$ ’ er tre forskellige operatorer der arbejder på henholdsvis skalar- og vektorfelter. ∇ kan ses som en vektor af de delvist afledte af skalarfeltets dimensioner. ∇ kan beskrives således for et todimensionelt skalarfelt:[2]

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$$

Så på et skalarfelt F bliver ∇F til et vektorfelt, hvor hver vektor peger i retning af den største stigning af værdierne af scalarfeltet.

$$\nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}\right)$$

Kigger vi derimod på den anden operator, $\nabla \cdot$, agerer den på vektorfelter. Tager vi og kigger på u, vektorfeltet over hastigheder i væsken, bliver $\nabla \cdot u$ til et skalarfelt. ∇ prikkes essentielt med hastighedsvektoren, beskrevet således:

$$\nabla \cdot \vec{u} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \cdot (\vec{u}_x, \vec{u}_y) = \frac{\partial \vec{u}_x}{\partial x} + \frac{\partial \vec{u}_y}{\partial y}$$

Dette beskriver den divergens der eksisterer i vektorfeltet, i dette tilfælde hastighedsvektorfeltet. Der betragtes de omkringliggende vektorer og ses på hvor meget ’hastighed’ der løber ind og ud af punktet. $\nabla \cdot u$ er negativt når vektorerne løber mere ind mod punktet end ud. Formlen (1) sætter dette til at være 0. Da

densiteten er konstant over hele væsken, fordi den er inkompressibel, kan der ikke løbe mere væske ind i et punkt end der løber ud. Dette ville nemlig give at masse skulle forsvinde eller skabes i punktet, hvilket ikke ville give nogen mening.

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho_0}\nabla p + \nu \nabla^2 \vec{u} + \vec{f} \qquad (2)$$

Betragter vi den anden formel (2), får vi beskrevet hvordan hastighedsvektorerne ændres over tid. Her konserveres hastighed. Formlens forskellige led har hvert at gøre med sin egen effekt på væsken, og kan betragtes alene. Det første led $(u \cdot \nabla) \cdot u$ har at gøre med konvektion og behandler komposanterne af hastighedsvektorerne for at få deres resulterende bevægelse. Leddet kan udvides til dette:

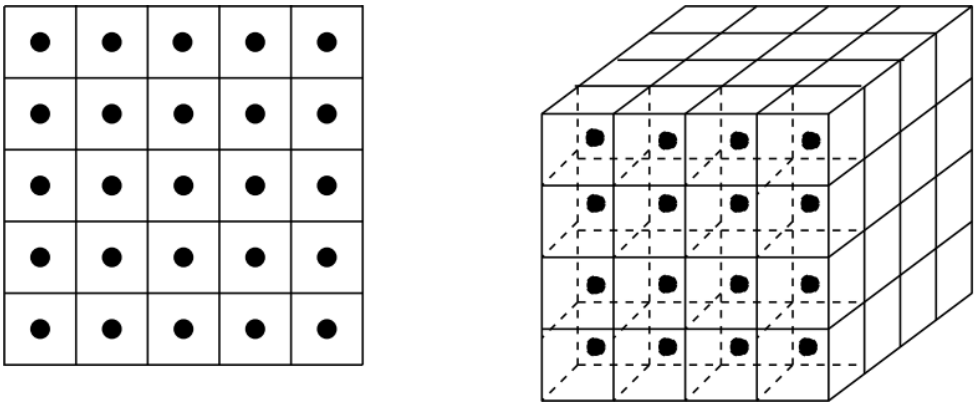
$$(\vec{u} \cdot \nabla)\vec{u} = \begin{pmatrix} u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial u}{\partial y} \\ u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial v}{\partial y} \end{pmatrix}$$

Læg mærke til at u og u er to forskellige tegn, det første designerer hastighedsvektoren og det andet designerer x-komposanten af vektoren. Vi ender altså med en vektor der er en originale hastighedsvektor u ganget på ændringen af komposanterne over deres retninger. Det andet led i formlen (2) beskriver hvordan trykgradienten påvirker bevægelsen af væsken, hvor meget væskens hastighed er dog påvirket af densiteten. Det tredje led er diffusion, og beskriver ved hjælp af viskositeten af væsken hvordan væsken bliver påvirket af resten af væsken rundt om i forhold til viskositeten. Det sidste led er alle andre kræfter der kunne påvirke væsken. Det kunne f.eks. være tyngdekraft.[1]

Computeren og væskedynamik som algoritmer

Vi har set på Navier-Stokes formlerne for inkompressible væsker, men hvordan oversætter vi det til et computerprogram. Den største forskel mellem hvordan formlen fungerer og hvordan vi kommer til at implementere den er domænet. Formlerne beskriver et kontinuert væskesystem, hvilket er besværligt at arbejde med. Da vi kommer til at simulere væsker numerisk og ikke analytisk må vi gøre væskesystemet diskret. I virkeligheden er en væske ikke kontinuer. Zoomer vi langt nok ind når vi et punkt hvor væsken er individuelle partikler, der opfører sig meget anderledes end den overordnede væske som formlerne beskriver. For at have et diskret væskesystem er der to metoder der er brugt oftest. Såkaldte feltbaserede metoder og partikelbaserede metoder. [**sus**] Vi kommer til at implementere to algoritmer i programmet vi skriver, en feltbaseret og en partikelbaseret metode. Vi starter med at se på den feltbaserede metode. [**sus**]

Til den feltbaserede metode består væskesystemet af et stort gitter med felter, hvor hvert celle i feltet har en samling værdier. Hvert celle har værdierne defineret ud fra centrummet af cellen, og svarer til et gennemsnit af væskens egenskaber inden for cellen. Størrelsen og mængden af celler bestemmer hvor præcis simuleringen af væsken er, dvs. hvor fin en opløsning man får. Da tid også skal være diskret bliver der taget faste "timesteps" Δt. Et mindre timestep giver en bedre tidsopløsning, og gør simuleringen yderligere præcis, på bekostning af at flere steps skal tages før en vis mængde tid er gået i simuleringen, altså at den bliver langsommere.



Figur 1, De diskrete felters celler har deres værdier defineret ud fra deres centrum. Ophavsret: Jos Stam, fra "Stable Fluids" 1999

Ud fra gitteret er der flere felter, blandt andet vektorfeltet hastighedsfeltet. For at simulere væskesystemet itereres der en algoritme med en serie funktioner. Hvert celle i domænet bliver betragtet for at opdatere dens værdier. Hvert iteration over et timestep giver væsken en ny tilstand, med alle felterne opdateret, som der så kan laves en ny iteration ovenpå. Tiden det tager at beregne en iteration er hvad vi er interesserede i når det gælder de tests vi vil lave på programmet.

Programmet

Selve algoritmernes design kommer til at være stærkt inspireret af artiklen "Fluid Simulation for Dummies", 2006, af Mike Ash, samt YouTube videoen "Coding Challenge #132: Fluid Simulation", 2019, af The Coding Train. Artiklen bygger på en anden artikel, "Real-Time Fluid Dynamics for Games", 2003, af Jos Stam, der er en efterfølger af "Stable Fluids" som jeg har refereret til tidligere.

Design

Designet af programmet skal facilitere simuleringerne og tests af simuleringerne. Programmet skrives i Processing (Java), da jeg har erfaring i det og da algoritmen er blevet implementeret i Processing allerede. For bedst at besvare problemformuleringen og få nogle gode testresultater, måles der på nogle få metrikker hvorefter visse forhold kan analyseres mellem dem. Tid er en stor faktor. Præcision er også en stor faktor. Der vil derfor ses på hvor lang tid det tager at beregne en enkel iteration, eller et enkelt timestep rettere sagt. Antallet af felter bliver også taget i betragtning, da det er direkte korreleret med præcisionen af simuleringen. Der vil også laves en meget begrænset visuel analyse af programmets output, for at dømme forskelle i præcision.

Referencer

[1] Stam, J. (2003). *Real-Time Fluid Dynamics for Games.* Hentet fra dgp.toronto.edu: https://www.dgp.toronto.edu/public_user/stam/reality/Research/pub.html

[2] Parth G. "*This Downward Pointing Triangle Means Grad Div and Curl in Vector Calculus (Nabla / Del) by Parth G*", youtube.com (23-03-2021). https://youtu.be/hI4yTE8WT88