Löv is in the air

Grupp 2
Pelle Serander pelse862, Einar Sandberg einsa241,
Johan Reimann johre099, Erik Olsson eriol726

11 mars 2015

Sammanfattning

Denna rapport behandlar ett projekt där lövs rörelse genom luft har modellerats och sedan visualiserats. De mest grundläggande ekvationerna implementerades till en början i Matlab och senare i C++ med OpenGL.

Innehåll

1 Inledning

1.1 Syfte och bakgrund

Syftet med projektet är att modellera och simulera ett fysikaliskt system utifrån de kunskaper som har samlats in i tidigare kurser.

Utgångsidén var att simulera hur löv faller genom luft, under påverkan av vindar och andra krafter. Efter en del efterforskningar om lövs rörelsemönster i luft konstaterades det snabbt att avgränsningar behövde göras. Att göra en helt korrekt simulering av hur löv faller är inte möjligt. Det finns flera rörelser som inte går att förklara med hjälp av en fysikalisk formel utan endast kan approximeras med mätningar från till exempel höghastighetskameror.

Projektet är skrivet i Matlab samt C++ med OpenGL. I OpenGL användes tredjepartsbibliotek från Bullet[1] för att underlätta fysikberäkningar samt biblioteket SOIL[2] för att hantera texturer.

1.2 Avgränsningar

Denna rapport sträcker sig inte till att ta upp hur vind beter sig när den åker genom stora volymer av löv och hur detta påverkar lövens rörelser, den tar heller inte upp kollisioner mellan objekt. För att projektet skulle kunna genomföras under den utsatta tiden valde vi även att betrakta lövet som ett stelt plan (Figur 1). I och med detta kunde formler enklare approximeras till kroppen.

Figur 1: *Kvadraten runt lövet representerar den yta som används för beräkningar.*

[1] Se http://bulletphysics.org/
[2] Se http://www.lonesock.net/soil.html

2 Teori

Teorin bakom hur ett löv faller är komplext. Detta beror framförallt på att ett löv inte är en stel kropp, och därmed har komplicerade rörelsemönster. De yttre krafter som påverkar kroppen är gravitation, luftmotstånd och vind. Dessa krafter ger upphov till olika rörelsemönster exempelvis rotation, böljning samt tumlande rörelser.

När en kropp faller uppstår ett luftmotstånd mot kroppen. Detta motstånd beror av luftens densitet, kroppens hastighet och form, samt den area som är riktad mot kraftriktningen. Genom att sedan applicera en konstant och uniform vind så uppstår ytterligare en påverkan, i vindens riktning.

När kroppen accelererar så bildas ett kraftmoment som beror på kryssprodukten av radien och kraften. Detta skapar en rotation samt så ges en vinkelhastighet. För att inte krafterna ska påverka objekten helt uniformt så används den effektiva arean gentemot krafterna som appliceras. Med effektiv area så menas den procentuella del av arean som exponeras mot kraften. Utöver detta så bildas luftströmmar genom stora volymer av löv vilket till viss del blockerar alternativt dämpar krafterna.

2.1 Gravitation och luftmotstånd

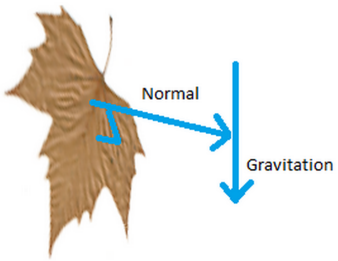
För att simulera gravitation användes Newtons andra lag (1) [1]. Luftmotstånd simuleras genom att ta fram kraften för den strömmande luften runt ett objekt enligt (2).

$$F = ma \tag{1}$$

$$F_d = \frac{C\rho A v^2}{2} \tag{2}$$

Här är F kraften, m massan, a accelerationen, C luftmotståndskoefficienten, ρ densiteten, A tvärsnittsarean och v hastigheten.

Arean som används är den effektiva arean som är riktad mot summan av de påverkande krafterna, gravitation och vind. Ju större del av ytan som är riktad mot denna summa av krafter ju mer dämpas hastigheten och vice versa. För att ta fram hur mycket av ytan för varje löv som är riktad mot krafterna räknas skalärprodukten ut för normalen och krafternas riktning (Figur 2).



Figur 2: *Normal för löv samt riktningen för gravitationen.*

2.2 Rotation

För att beräkna rotationen används momentet för lövet för att initiera rotationen samt som uppdatering, då krafterna ändras kontinuerligt under renderingen. Den vinkelhastighet som beräknas används sedan för att bestämma vinkelpositionen. För rotationen används två kraftmoment, ett som uppstår av de påverkande krafterna enligt (3)[1], samt ett moment som dämpar rotationen för lövet (4).

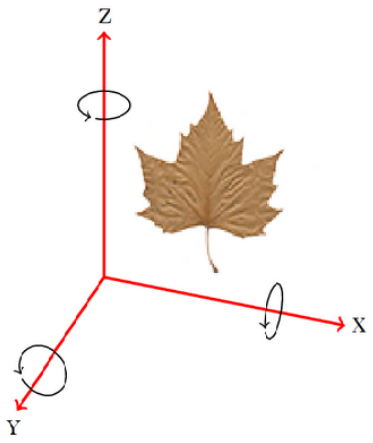
$$\vec{M} = \vec{r} \times \vec{F} \tag{3}$$

$$\vec{M}_d = \frac{-A_\omega \cdot \rho \cdot w \cdot l^4 \cdot \vec{\omega}^2}{2} \tag{4}$$

Där $\vec{r}$ är radien, $\vec{F}$ de påverkande krafterna, A_w den effektiva arean mot vinkelhastigheten, ρ densiteten, w bredden, l längden och ω vinkelhastigheten.

Momentet beror av radien och den applicerade kraften. Momentdämpningen uppkommer av ett luftmotstånd som uppstår vid rotationen. Vinkelhastigheten är lika med hastigheten genom radien (5), (Figur 3).

$$\vec{w} = \frac{\vec{v}}{\vec{r}} \tag{5}$$



Figur 3: *Rotation runt alla axlar där koordinatsystemet ligger i lövets centrum.*

2.3 Böljningar

När ett plant objekt faller genom ett medium bildas ett antal hydrodynamiska effekter, bland annat luftmotstånd, lyft samt överstegring. Överstegring är ett mått på förlust av lyftkraft hos ett objekt som bland annat beror på attackvinkeln hos kroppen samt vilken fart denna har. Lyftet genereras av den komponent som är ortogonal mot flödets riktning. För att kunna beräkna rörelsen används ett Froude-tal, en relation mellan olika tidsskalor, för att beräkna skillnaden

mellan den lokala hastigheten samt fältet, i detta fall gravitationen. Detta tal avgör om den lokala hastigheten kommer att påverkas av fältet uppifrån eller nedifrån. Under detta fall uppmättes endast tal större än ett, vilket indikerar att det är fältet uppifrån som styr flöde, vilket är logiskt då lövet faller mot marken[2].

De vinkelpositioner som erhölls användes sedan i böljningarna (6), (7) genom att låta den ena vinkeln vara den erhållna vinkelpositionen samt genom att jämföra vinkeln med den maximala svängningsvinkeln och använda detta som den andra vinkeln.
Böljningarna beräknas genom att approximera riktningsaccelerationer (6), (7) och (8) för vinkelpositioner vilket jämförs med maximala vinkelsvängningarna.

$$v_x' = \frac{v_x^2}{F_r} \cdot (A_{ortogonal} \cdot \sin(t) \cdot \sin(p) - A_{parallel} \cdot \cos(t) \cdot \cos(p) \\ +4 \cdot \pi \cdot \mid (\sin(t)) \mid \cdot \cos(\gamma + \pi/2)) \tag{6}$$

$$v_y' = \frac{-v_y^2}{F_r} \cdot (A_{ortogonal} \cdot \sin(t) \cdot \cos(p) - A_{parallel} \cdot \cos(t) \cdot \sin(p) \\ +4 \cdot \pi \cdot \mid (\sin(t)) \mid \cdot \sin(\gamma + \pi/2)) - 1/F_r \tag{7}$$

$$v_z' = \frac{v_z^2}{F_r} \cdot (A_{ortogonal} \cdot \sin(t) \cdot \sin(p) - A_{parallel} \cdot \cos(t) \cdot \cos(p) \\ +4 \cdot \pi \cdot \mid (\sin(t)) \mid \cdot \cos(\gamma + \pi/2)) \tag{8}$$

I dessa ekvationer är F_r Farouds tal, t är vinkelpositionerna samt p jämförelsen mellan vinkelpositionerna samt maximala utslagsvinkeln. Detta är från början en approximation för 2D men rörelsen har anpassats till 3D av gruppen. Detta gjordes genom att betrakta rörelsen i planet som homogen, vilket kan ses då ekvation (6) är lika med (8).

2.4 Stegmetoder

I Matlab användes ODE45[5] för att simulera systemet. ODE45 är en explicit Runge-Kutta metod. Den löser fjärde och femte ordningens differentialekvationer. Den differens som fås genom att ta skillnaden på dem sista ordningarna används sedan som felet för fjärde ordningens lösning.

Bullet använder sig av en stegmetod[4] som beskrivs som "*nonsmooth nonlinear conjugate gradient method*", enligt nedan.

$$\mathbf{p}_{k+1} = \beta_{k+1} \cdot \mathbf{p}_k - \nabla f_{k+1} \tag{9}$$

$$\mathbf{p}_1 = -\nabla f_1 \ \textit{villkor för sökriktningen} \tag{10}$$

$$\beta_{k+1} = \frac{\| \nabla f_{k+1} \|^2}{\| \nabla f_k \|^2} \ \textit{Fletcher-Reeves metod} \tag{11}$$

$$\lambda_{k+1} = \lambda_k + \tau_k \cdot \mathbf{p}_k \tag{12}$$

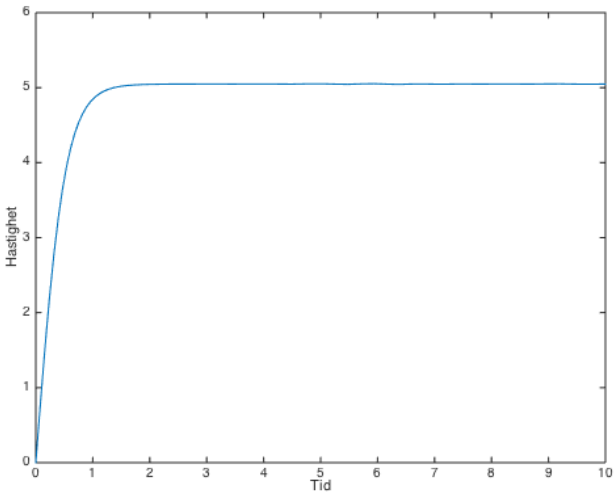
$$\| \nabla f_{k+1} \|^2 > \| \nabla f_k \|^2 \ \ \textit{villkor för iteration} \tag{13}$$

Metoden bygger på att först beräkna den brantaste riktningen $\mathbf{p}$ (9) med villkoret (10) för att hitta ett minimum[3] (1) och sedan beräkna β (11) enligt *Fletcher-Reeves*. Därefter beräknas linjesteget τ. Detta används sedan för att beräkna steget som ska tas (12). Villkoret (13) används för undersöka när ett minimumvärde har nåtts, vilket startar om iterationen. Denna metod används i detta fall för att simulera interaktiva stela kroppar med flera punkter där krafter kan påverka.

3 Genomförande

I början av projektet analyserades problemet numeriskt. En enkel modell för hur en stel kropp faller genom luft dämpat av luftmotstånd togs fram. Modellen skrevs som en differentialekvation och implementerades i Matlab (14) (Figur 4).

$$v'(t) = g - \frac{CpAv^2}{2m} \tag{14}$$

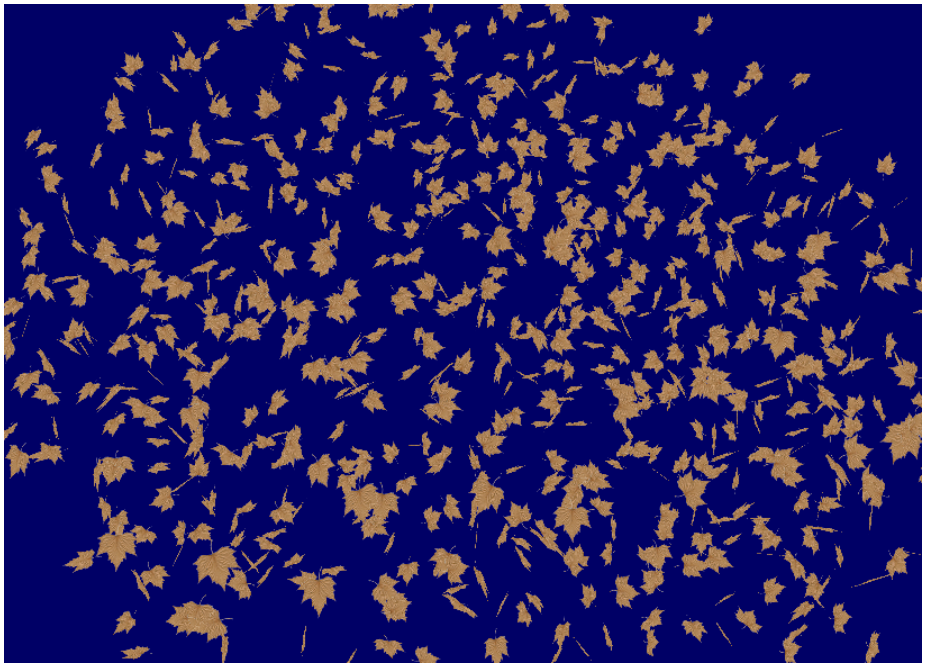


Figur 4: *Hastighetsgraf för en stel kropp.*

I figur (4) ser vi att accelerationen avtar när hastigheten för lövet ökar. Modellen implementerades sedan i C++ med OpenGL. Att få till en bra struktur i koden tog en del tid, så parallellt med detta studerades mer avancerade modeller för tunna plans rörelse i luft, och koefficienter för dessa. Dessa avancerade modeller tog även hänsyn till vindar, och därmed lövens rotation.

Simulationen kan köras på de flesta nyare datorer, oavsett om det är OSX, Windows eller Linux. Det kan dock skilja en del i prestanda, för svagare datorer är det bra att minska antalet löv som renderas för att programmet ska köras utan fördröjningar.

4 Resultat



Figur 5: *Skärmdump av resultatet.*

Resultatet blev löv som renderas som egna objekt (Figur 5) i en värld, vilket blir väldigt beräkningstungt för datorn. För att optimera simuleringen bör antalet löv anges vilket enkelt kan sättas i koden. Löven skapas med slumpvist satta vinklar och tillhörande normaler. Eftersom det finns fler än 46 miljoner kombinationer av vinklar så är risken att två eller flera löv får samma startvinkel minimal. Detta ger ett regn av löv där varje enskilt löv inte påverkas av de andra löven utan endast av gravitation samt vind vilket också enkelt kan anges i koden om så önskas. En del begränsningar har satts för att projektet inte ska bli för stort. De krafter som löven nu påverkas av är luftmotstånd, tröghetsmoment, dämpning i rotation vilket beror på arean hos lövet samt vinkelhastigheten samt en formel för böljningseffekt. Vid implementation av dessa formler drar löven som slumpats med likande vinklar åt samma håll vilket kan skapa grupperingar av löv som inte sprider sig utan håller samman som en grupp under lång tid. Efter tillräckligt lång tid så sprider sig även dessa grupperingar. Då vissa delar av lövets rörelser inte modellerats samt att vissa delar inte går att modellera så blir några rotationer konstanta.

5 Diskussion

Tanken i början av projektet var att först implementera det mesta i Matlab, och sedan i C++ med OpenGL. Detta reviderades senare, då gruppen ansåg att det var för tidskrävande att implementera avancerade 3D-modeller i Matlab, och valde att istället direkt arbeta med OpenGL. Kanske hade man istället kunnat

göra små delmodeller i Matlab av de olika ekvationerna, då gruppen i ett senare skede upptäckte att det var svårt att kontrollera att simulationerna i OpenGL stämde.

Under projektets gång stöttes det på en hel del problem. Eftersom gruppen inte hade jobbat särskilt mycket med OpenGL tidigare krävdes mycket jobb bara för att komma igång. Utöver det tog andra rent tekniska saker mycket tid, så som att få igång biblioteken Bullet och SOIL. Gruppen var även delad mellan Mac och Windows, vilket skapade en del tekniska svårigheter.

Det var svårt från början att veta hur avancerat man skulle klara av att göra modellen. Men med tanke på de förenklingar som gjordes i projektet anser vi att resultatet är satisfierande.

6 Slutsats

Med den forskning och teknologi som finns idag är det väldigt svårt och tidskrävande att utforma en helt fysikaliskt korrekt simulering av hur löv faller. En hel del anpassningar kan göras för att få en liknande effekt. Resultatet blev bra med hänsyn till förenklingarna.

Referenser

[1] Halliday, Resnick, Walker, Principles of Physics 9th edition, John Wiley & Sons Inc., 2011

[2] https://www.math.psu.edu/belmonte/PaperFile/flutter.pdf

[3] http://www.cnd.mcgill.ca/ ivan/non-smooth-constaint-contact-force.pdf

[4] http://image.diku.dk/kenny/download/2010_CGI_TALK.pdf

[5] http://se.mathworks.com/help/matlab/ref/ode45.html