LINKÖPINGS UNIVERSITET

MODELLERINGSPROJEKT

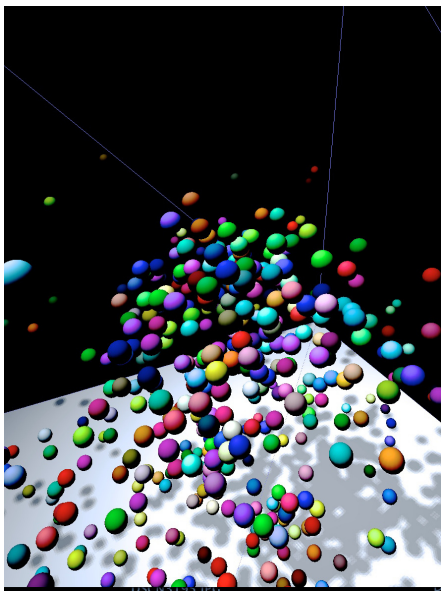
TNM085

Amazeballs

Sofie LINDBLOM
Anton ARBRING
David LINDH

Examinator
Anna LOMBARDI

13 mars 2014



Sammanfattning

I kursen TNM085, modelleringsprojekt skapas ett projekt vars syfte är att modellera ett fysikaliskt system av valfri typ. I den här rapporten redogörs om hur ett antal kolliderande studsbollar simuleras. Inspirationen är hämtad från en reklam Sony gjorde 2009 där man släpper miljontals studsbollar ner för en gata i San Fransisco

Innehållsförteckning

1 Inledning **1**
- 1.1 Inledning . . . 1
- 1.2 Syfte . . . 1

2 Implementation **2**
- 2.1 Förarbete . . . 2
- 2.2 Fysiskt System . . . 2
- 2.3 Simuleringar i MATLAB . . . 2
  - 2.3.1 1D . . . 2
  - 2.3.2 3D . . . 4
- 2.4 Simuleringar i WebGL . . . 5

3 Implementation **8**
- 3.1 Förenklingar . . . 8
- 3.2 Resultat . . . 8
- 3.3 Reflektion . . . 9

Litteraturförteckning **10**

A 1D Simulering i MATLAB **11**

B Second proof of... **12**

Figurer

2.1 Example of figure 3
2.2 Hastighet och position för studsboll i 1D 4
2.3 Position av boll i 3D 5

3.1 9

Kapitel 1

Inledning

1.1 Inledning

Efter att ha resonerat kring att simulera luftballonger, rinnande vatten eller rök landade gruppen tillslut i beslutet att simulera studsbollar. Motiveringen till beslutet var att det på ett smidigt sätt skulle gå att involvera interaktivitet genom att användaren kan välja utvalda attribut som till exempel radien på studsbollarna. Huvudsakligt fokus har varit en fysikaliskt realistisk studs mellan boll och omgivning samt kollisionshanteringen mellan bollarna. Sekundärt fokus har varit möjlighet till interaktivitet för användaren och att simulera bollarna i en mer komplex omgivning.

- Så här gör man en lista
- In Catilinam II
- In Catilinam III
- In Catilinam IV

1.2 Syfte

Syftet med det här projektet är att implementera fysiska samband i en teknisk tillämpning. Att kunna bemästra fysikaliska lagar och regler så pass väl att det är möjligt att avgöra vilka förenklingar som kan genomföras och fortfarande uppnå ett realistiskt resultat. Tanken är även att en introduktion till rapportskrivningsverktyget LaTex ska erhållas.

Kapitel 2

Implementation

Figure is not by Cicero. Multa meo quodam dolore in vestro timore sanavi. Nunc si hunc exitum

2.1 Förarbete

Arbetet inleddes med undersökningar av hur liknande problem lösts av andra. I ett tidigt skede övervägdes att simulera studsar med hjälp av fjäderekvationer och konservering av volym. Det insågs dock att ett bättre alternativ är att använda sig av en så kallad 'Coefficient of restitution' , både med avseende på komplexitet och beräkningstyngd vid simulering av ett stort antal bollar.

Det spenderades en del tid på att undersöka olika simuleringsverktyg, jämföra openGL och WebGL samt vilka bibliotek och resurser som bäst lämpar sig att använda. Efter undersökning och diskussion togs beslutet att använda three.js vilket är ett javascript-bibliotek som hanterar WebGL.

2.2 Fysiskt System

Arbetet inleddes med undersökningar av hur liknande problem lösts av andra. I ett tidigt skede övervägdes att simulera studsar med hjälp av fjäderekvationer och konservering av volym. Det insågs dock att ett bättre alternativ är att använda sig av en så kallad 'Coefficient of restitution' , både med avseende på komplexitet och beräkningstyngd vid simulering av ett stort antal bollar.

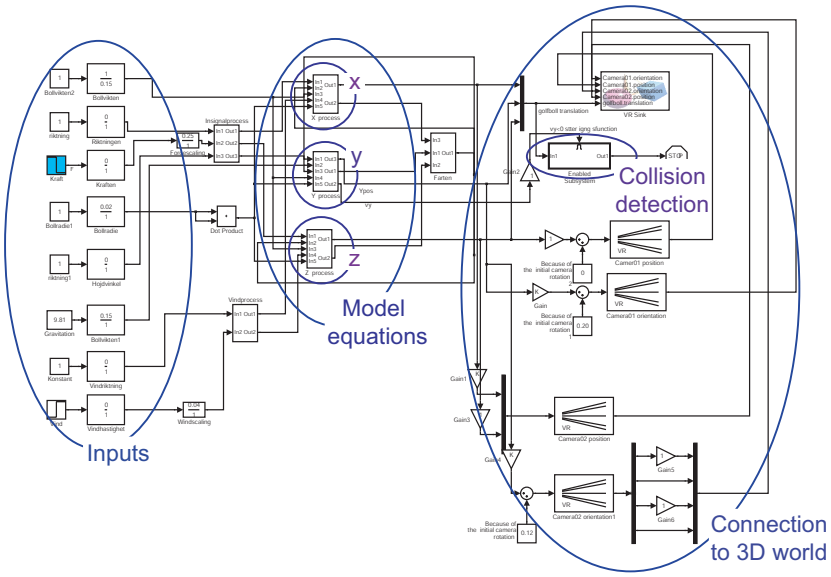
Det spenderades en del tid på att undersöka olika simuleringsverktyg, jämföra openGL och WebGL samt vilka bibliotek och resurser som bäst lämpar sig att använda. Efter undersökning och diskussion togs beslutet att använda three.js vilket är ett javascript-bibliotek som hanterar WebGL.

2.3 Simuleringar i MATLAB

Also containing some text, and a figure.

2.3.1 1D

Som första steg modelleras en studsande boll i en dimension. Syftet med detta är att uppnå en god grundförståelse för att sedan utveckla modellen till flera dimensioner.



Figur 2.1: Example of figure

För att verifiera framarbetade fysikaliska formler plottas positionen och hastigheten som funktion av tiden (Se figur (2.2)). Det fysikaliska sambanden som MATLAB implementationen är baserad på visas i ekvation (1) - (4) med tillhörande beskrivning av inblandade variabler i tabell 1.

Tabell 2.1: Tabell 1

	VARIABLER	BESKRIVNING VÄRDER
k	Energiförlust vid kollision med marken	0.6
g	Gravitationskraften	9.82
deltat	Tidsintervall mellan två uppdateringar	1
v[new]	Hastighet i kommande uppdatering	-
v[old]	Hastighet i nuvarande tillstånd	-
p[new]	Position i kommande uppdatering	-
p[old]	Position i nuvarande tillstånd	-

Om boll slår i marken:

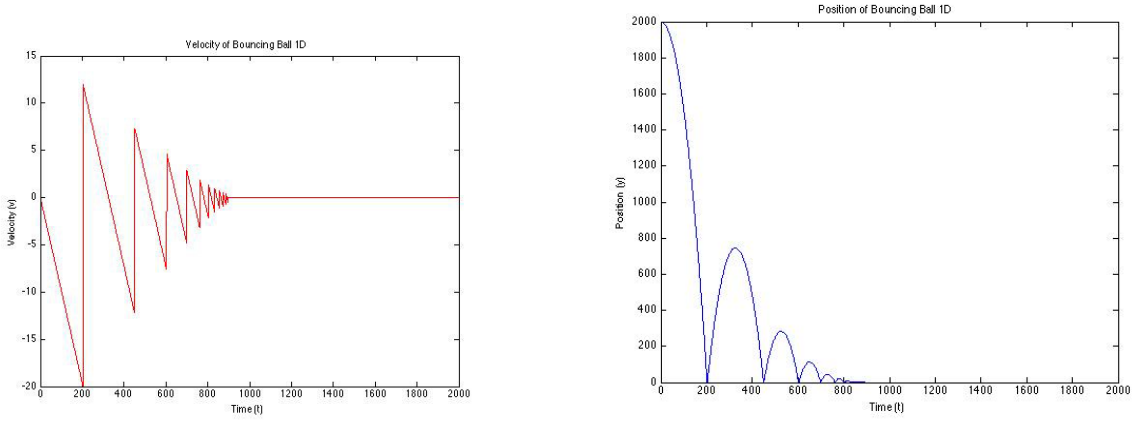
$$v[new] = -k * v[old], \tag{2.1}$$

$$p[new] = v[new] * deltat, \tag{2.2}$$

Om boll faller fritt i y-led:

$$v[new] = v[old] + g * deltat \tag{2.3}$$

$$p[new] = p[old] * v[old] * deltat \tag{2.4}$$



Figur 2.2: Hastighet och position för studsboll i 1D

2.3.2 3D

I ett steg mot slutliga implementationen utökas modellen från en dimension till tre. Mellansteget i två dimensioner har exkluderats från rapporten då det är samma tillväga gångssätt som i denna simulering.

Fysikaliska sambanden är desamma som i en dimension. Skillnaden är att samtliga variabler utökas från en till tre komponenter (x,y,z). Bollen får en initial hastighet (som är noll i yled, eftersom det är i det ledet som gravitationen verkar) och en initial position.

Bollens position kontrolleras mot underlaget på samma sätt som tidigare men utöver det så kontrolleras även om kollision med väggarna i en kub har inträffat. Kuben har dimensionerna 300x300x300 och har sitt ena hörn placerat i origo (se figur 2.3).

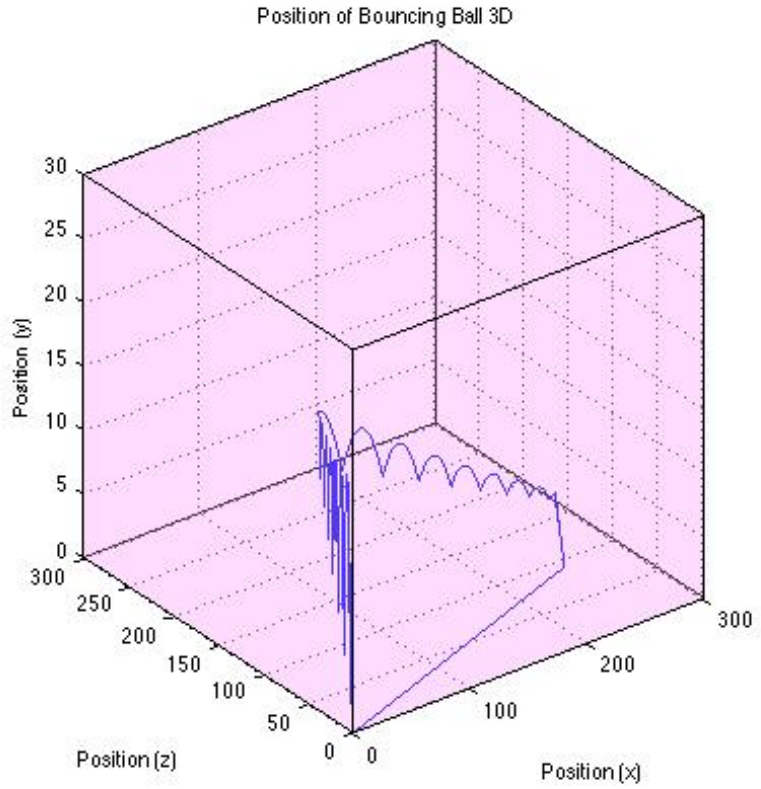
Totalt görs fem stycken kontroller, en för varje sida av kuben bortsett från taket. Om koordinatens värde i det led som kontrolleras överstiger 300 eller understiger noll görs en beräkning av en ny hastighet längs den axeln. Om ingen kollision inträffar uppdateras position och hastighet precis på samma sätt som i simuleringen i en dimension. Nedan är ett exempel på hur en kollison med kubens sida i x-led hanteras, samtlig kod finns även i Bilaga B.

För positionen i xled då x-positionen är vid väggen:

$$v[new] \ = \ k * v[old] \tag{2.5}$$

$$p[new] \ = \ p[old] \ + \ v[new] * deltat \tag{2.6}$$

För positionen i yled:



Figur 2.3: Position av boll i 3D

$$v[new] \ = \ v[old] + \ g * deltat \tag{2.7}$$

$$p[new] \ = \ p[old] \ + \ v[new] * deltat \tag{2.8}$$

För positionen i zled:

$$v[new] = v[old] p[new] \ = \ p[old] \ + \ v[new] * deltat \tag{2.9}$$

Även i detta exempel används en räknare för att undvika ett oändligt antal studsar. Hastigheten i respektive led plottas gentemot en tidsaxel för tydligare tolkning av resultatet (Se figur 7).

2.4 Simuleringar i WebGL

I den slutliga simulering används WebGL i kombination med three.js, målet är att simulera mer fysikaliskt realistiska kollisioner som även tar hänsyntill sneda sammanstötningar. Hastighetsvektorerna från de inblandade bollarna projiceras på två komposanter i kollisionens tangent- och normalriktningar. Tangenterna erhålles genom kryssprodukt med normalen och tangenten. Normalen erhålles genom ekvation 10 och tangenten genom att x- och y-koordinatens värden sätts till ett fixt värde, z-koordinatens värde räknas ut enligt ekvation 11, slutligen erhålles tangenten för de koliderande bollarna genom ekvation 12.

Tabell 2.2: Tabell 1

VARIABLER	BESKRIVNING
$Normal$	Normalen för de kolliderande bollarna
$p1$	Postionsvektor för boll nummer 1
$p2$	Postionsvektor för boll nummer 2
tz	tangentens z-koordinat
$Tangent$	Tangenten för de kolliderande bollarna

$$normal = p1.x - p2.x, p1.y - p2.y, p1.z - p2.z \tag{2.10}$$

$$tz = (-(normal.x * 0.3) - (normal.y * 0.3))/normal.z \tag{2.11}$$

$$Tangent = (0.3, 0.3, tz) \tag{2.12}$$

Tangentiella energin bevaras (cite Grahn Jansson) och energin i normalriktningen räknas ut enligt ekvation 15 och 16, dessa är härledda från ekvation 13 och 14.

Tabell 2.3:

VARIABLER	BESKRIVNING
$V1t\prime$	Boll nummer ett, tangentvektor innan kollision
$V2t\prime$	Boll nummer två, tangentvektor innan kollision
$V1n\prime$	Boll nummer ett, normalvektor innan kollision
$V2n\prime$	Boll nummer två, normalvektor innan kollision
$V1t\prime\prime$	Boll nummer ett, tangentvektor efter kollision
$V2t\prime\prime$	Boll nummer två, tangentvektor efter kollision
$V1n\prime\prime$	Boll nummer ett, normalvektor efter kollision
$V2n\prime\prime$	Boll nummer två, normalvektor efter kollision
e	Konstant för energiförlusten

Tangentiella hastighetsvektorer:

$$V1t\prime = V1t\prime\prime \tag{2.13}$$

$$V2t\prime = V2t\prime\prime \tag{2.14}$$

Normalens hastighetsvektorer:

$$e = (V2n\prime\prime - V1n\prime\prime)/(V1n\prime - V2n\prime) \tag{2.15}$$

$$V1n\prime + V2n\prime = V1n\prime\prime + V2n\prime\prime \tag{2.16}$$

$$V1n\prime + V2n\prime = V1n\prime\prime + V2n\prime\prime \tag{2.17}$$

$$V2n\prime\prime = e(V1n\prime - V2n\prime) + V1n\prime + V2n\prime \tag{2.18}$$

$$V1n\prime\prime = v1n\prime + v2n\prime - v2n\prime \tag{2.19}$$

Slutligen erhålls bollarnas hastigheter efter kollisionen genom att tangentiella och normalens hastighetsvektorer adderas.

A word or two to conclude, and this even includes some inline maths: $R(x,t) \sim t^{-\beta} g(x/t^{\alpha}) \exp(-|x|/t^{\alpha})$

Kapitel 3

Implementation

3.1 Förenklingar

Genom hela processen har vi försummat att simulera deformation som vanligtviss uppstår vid studs eller kollision. Implementationen av deformation ansågs för krävande och kategoriserades som en möjlig utveckling av systemet.

Ytterligare en förenkling som gjorts är att luftmotståndet försummats. Detta för att avlasta den redan beräkningstunga applikationen men framför allt för att den synliga inverkan på bollarna av luftmotståndet är knappt märkbar.

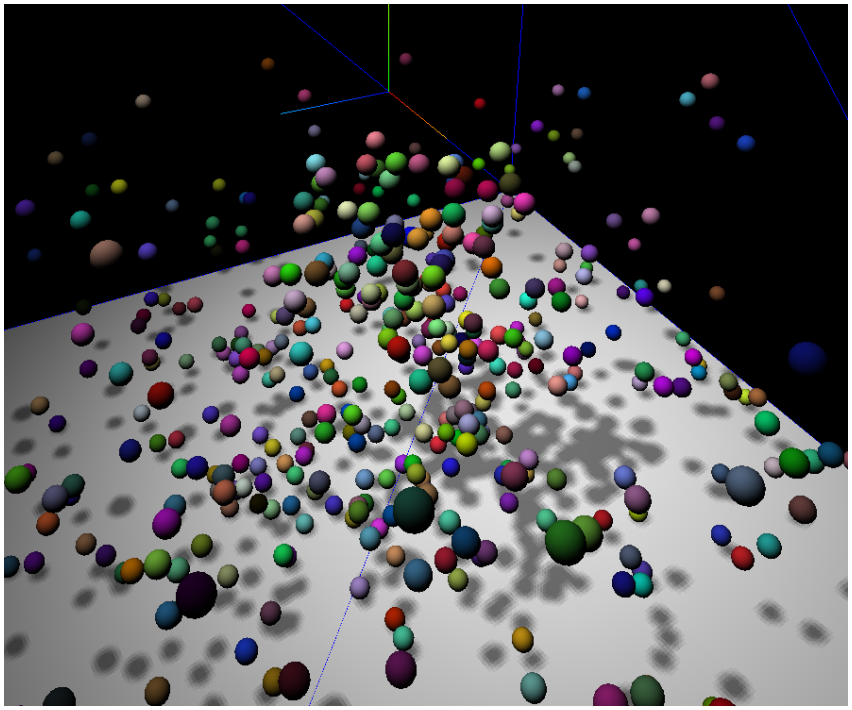
Vid kollison mellan bollar samt mellan omgivning och bollar har energiförlusten approximerats genom att multiplicera med en konstant. Denna förenkling är relativt korrekt men för att konstanten ska vara precis bör hänsyn tagits till ett flertalet variabler som till exempel elasticitet hos kolliderande föremål. Denna approximation hade liksom förenklingen med luftmotståndet ingen större inverkan på resultatet.

Grundidéen var bollarna skulle studsa ner för en gata snarlikt den bild från Sonys reklam som återfinns i figur 1. En första simulering gjordes dock i en förenklad miljö (en box) för att fastställa att grundfunktionaliteten fungerar.

Bollarna får en initierad starthastighet, från början var det tänkt att gravitation, lutning och rotation skulle vara det som gav dem en initial hastighet och riktning. Rotationen som uppstår när bollarna krockar dels med varandra och dels med väggar och golv är en annan möjlig utvidgning av nuvarande applikation.

3.2 Resultat

Den slutliga applikationen kan köras genom att klicka här. Stommen av programmet är skrivet i javascript med stöd av three.js. Koden är uppdelad i fyra block. I det första blocket deklareras allt i scenen (kamera, sfär-objekt, plan, lampor, mus-interaktion och renderare). I block två skapas bollarna och blir tilldelade en initial hastighet och position. Block tre består av animeringen, där det i varje tidsram anropas en funktion som kollar om kollision med väggar och andra bollar skett. I denna del av programmet sker även kontrollen om hur många studs varje boll gjort dittills i animationen. Som nämndes i Matlab-simulering implementerades en räknare för att förhindra oändligt antal studsar. Block fyra består av kollisionshanteringen om det i ett tidsintervall uppstått kollision. Koden återfinns i bilaga D och en bild från slutresultatet visas i figur 3.1.



Figur 3.1: Slutlig simulering av 200 bollar

Det krävs inga specifika hårdvarukrav för att köra applikationen. Användaren behöver dock en webläsare som stödjer webGL.

3.3 Reflektion

Som sig så ofta inträffar går projektgruppen in med skyhöga ambitioner om vad som ska utföras under projektets gång. Andra kurser, engagemang och oförutsedda tidsdistraktioner kommer sedan i mellan och helt plötsligt är deadline runt hörnet. Det beskrivna scenariot är vad som hände denna projektgrupp. Men trots detta och det faktum att vi bara är tre personer är vi nöjda med resultatet.

Det har varit lärorikt och intressant att gå från fysikaliska formler skrivna med pappor och penna hela vägen till att kunna se det implementerat live i sin webläsare. Gruppen har kämpat med att hitta gemensam tid att arbeta då alla läser olika kurser. Skriva rapport i LaTeX har även varit en stor utmaning för oss då alla är nybörjare.

Vårt mål är att hinna implementera viss interaktion mellan användaren och applikationen innan slutpresentationen. Om gruppen lyckas med detta är vi mycket nöjda med våra insatser i kursen och med slutresultatet.

Litteraturförteckning

Bilaga A

1D Simulering i MATLAB

Bilaga B

Second proof of…

bla2