Institutionen för teknik och naturvetenskap

Linköpings universitet

TNM085 Modelleringsprojekt, vt 2019

Biljardsimulering

Författare

Joel Paulsson: *joepa811@student.liu.se*

Johan Fröberg: *johfr198@student.liu.se*

Adrian Andersson: *adran117@student.liu.se*

Gustaf Wallström: *gusan112@student.liu.se*

Examinator

Anna Lombardi: *anna.lombardi@liu.se*

Handledare

Ulf Sannemo: *ulf.sannemo@liu.se*

23 mars 2020

Sammanfattning

Den här rapporten behandlar hur en biljardsprängning simuleras i OpenGL. Uttryck för fysikaliska samband, genomgång av implementationen samt redogörelse för vilka avgränsningar som gjorts och varför. Resultatet av projektet är en animering utvecklad i C++ och OpenGL. Slutprodukten är ett kort filmklipp som visar simuleringen.

Innehåll

Sammanfattning i
Figurer iii
Tabeller iii
Typografiska konventioner v

1 Inledning **1**
1.1 Bakgrund 1
1.2 Syfte 1
1.3 Frågeställningar 2
1.4 Avgränsningar 2

2 Fysik & Matematik **3**
2.1 Kollisionshantering 3
2.2 Rotation 6
2.3 Friktion 6

3 Implementation **7**
3.1 MATLAB 7
3.2 3D-objekt 7
3.3 C++ & OPENGL 7
3.4 Kodstruktur 8

4 Projektmetod & arbetsfördelning **9**
4.1 Planering 9

5 Avslutning **10**
5.1 Resultat 10
5.2 Diskussion 11
5.2.1 Vidare arbete 11
Referenser . . . . .

Figurer

2.1 *Kollisonsmomentet mellan två bollar med massorna m_1 och m_2 samt hastigheter, enhetsnormalen, och enhetstangenten för en av bollarna.* 4

2.2 *Hastighetens riktningsförändring vid kollision med en av spelplanens sidor. θ är in- och utfallsvinkel.* 5

2.3 *Boll som rullar sträckan s med vinkelhastighet ω och hastighet $\vec{v}$ under en iteration.* 6

3.1 *Stegvis rotationer för att åstadkomma rotation kring en godtycklig axel. Hastighetsvektorn pekar i riktningen av siffran på biljardbollen i första steget. Z-axeln är markerad i grönt medan x- och y-axeln är markerade i rött och blått respektive.* 8

5.1 *Två bilder från simuleringar gjorda i* OpenGL. 10

5.2 *Två bilder från simuleringar gjorda i* Matlab. 10

Tabeller

2.1 *Fysikaliska storheter som berör en biljardsprängning* [1]. 3

Typografiska konventioner

Engelska ord i form av tekniska termer skrivs med *kursiv stil* och förklaras tillsammans med svenska tekninska termer löpande i texten. Programvaror skrivs med VERSALER.

Kapitel 1

Inledning

Den här rapporten presenterar ett projektarbete i kursen TNM085, Modelleringsprojekt vid Linköpings universitet under vårterminen 2019. Under projektet skapades en simulering av en biljardsprängning, alltså den första stöten och de efterkommande kollisionerna mellan biljardbollarna. Rapporten redovisar hur modellen skapades samt hur och med vilka verktyg som systemet implementerades.

1.1 Bakgrund

Systemet består av en köboll som skickas iväg med en starthastighet. Bollen träffar sedan en triangulär formation av färgade bollar som därefter rör sig med olika hastigheter i olika riktningar på spelplanen. Detta resulterar i ett system som består av 16 olika bollar som dels påverkas av varandra men även av biljardbordets matta och kanter. I systemet verkar en rad olika fysikaliska fenomen som påverkar hur sprängningen ser ut, vilka kommer diskuteras i rapporten. Simuleringar är viktiga för att minska eventuella kostnader och risker som annars hade uppstått vid fysiska experiment och testningar.

1.2 Syfte

Syftet med det här projektet är att skapa en visuell simulering av ett fysikaliskt system. Systemet ska representeras av matematiska modeller och implementeras för testning i MATLAB och animeras i C++ och OPENGL. Syftet är inte att skapa realism i det visuella utan att snarare fokusera på att fysiken stämmer, med undantag för diverse avgränsningar.

1.3 Frågeställningar

Nedan följer de frågeställningar som rapporten ämnar att besvara.

- Vilka fysikaliska fenomen förekommer i biljard?
- Vilka avgränsningar kan göras och fortfarande få en realistisk simulering av en sprängning i biljard?

1.4 Avgränsningar

Innan projektet startades gjordes följande avgränsningar:

- Bollarna kommer alltid att vara i kontakt med underlaget och kommer alltså inte studsa vertikalt eller falla ner i några hål
- Bollarna kommer bara kunna rulla, inte glida eller spinna
- Kollisioner mellan bollar överför ingen rotation
- Köbollen påverkas inte av en impuls som i en verklig sprängning. Den blir istället given en bestämd starthastighet längs underlaget
- Kollisioner mellan bollarna är helt elastiska i simuleringen, i verkligheten finns det en viss energiförlust
- Luftmotstånd försummas

Kapitel 2

Fysik & Matematik

Systemet består av totalt 16 olika bollar, varav en är en köboll. Köbollen får en stöt som ger den en utgångshastighet och sedan kolliderar den med de andra bollarna som är upplagda i en triangel, vilka i sin tur börjar röra sig över spelplanen och kan kollidera med andra bollar och planens kanter. Samtliga bollar påverkas av friktion och energiförluster och kommer efter en viss tid att stanna, och sprängningen är slut. De olika fysikaliska storheterna i systemet presenteras i tabell (2.1).

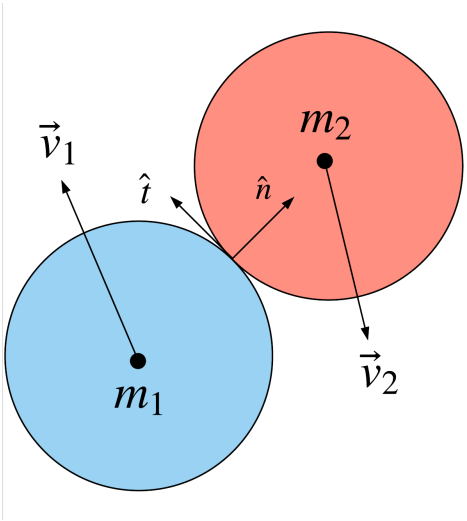
Tabell 2.1: *Fysikaliska storheter som berör en biljardsprängning* [1].

Storhet	**Notation**	**Enhet(er)**
Bollens radie	r	$0.0286m$
Köbollens vikt	m	$0.170kg$
Resterande bollars vikt	m	$0.165kg$
Utgångshastighet	v_0	$3m/s$
Friktion rullning	μ	0.01
Längd biljardbord	-	$2.13m$
Bredd biljardbord	-	$1.065m$

Det finns en rad olika fysikaliska fenomen som inverkar på systemet och som måste implementeras för att ge en trovärdig och fysikaliskt korrekt simulering. Biljardbollarna translaterar, roterar och kolliderar med varandra och med spelplanens kanter. Bollarna påverkas även av olika friktioner beroende på om de translaterar eller roterar, eller båda. Eftersom den slutgiltiga implementeringen skulle göras i C++/OpenGL med fördefinerade vektorer och stöd för vektoralgebra togs beslutet att försöka undvika trigonometri, och istället arbeta med vektoralgebra i skapandet av modellen.

2.1 Kollisionshantering

När en biljardboll förflyttas på spelplanen kan den kollidera med en annan boll eller en kant. Båda fallen är egentligen inelastiska kollisioner men vid kollision mellan bollar överförs endast $\sim 5\%$ av den kinetiska energin till exempelvis värmeenergi jämfört med kollision mellan boll och kant då $\sim 25\%$ överförs [1]. Kollisionen approximeras därför mellan bollarna till en elastisk kollision i det här projektet. En bild av en kollisionen mellan två bollar visas i Figur 2.1. Hastighet och massa för samma boll benämns med nedsänkt tecken enligt $\vec{v}_n$ och m_n.



Figur 2.1: *Kollisonsmomentet mellan två bollar med massorna m_1 och m_2 samt hastigheter, enhetsnormalen, och enhetstangenten för en av bollarna.*

I elastiska kollisioner bevaras den kinetiska energin (E_k) och rörelsemängden (P) innan och efter kollisionen [2], vilket visas i ekvation (2.1), där $'$ indikerar efter kollisionen. Kombineras uttrycken kan nya hastigheter efter kollisionen uttryckas för de båda bollarna enligt (2.2).

$$E_k = E_k^{'} \Leftrightarrow \left(\frac{mv^2}{2}\right) = \left(\frac{mv^2}{2}\right)^{'} \quad , \quad P = P^{'} \Leftrightarrow m_1v_1 + m_2v_2 = m_1v_1^{'} + m_2v_2^{'} \tag{2.1}$$

$$v_1^{'} = \frac{v_1(m_1 - m_2) + 2m_2v_2}{m_1 + m_2} \quad , \quad v_2^{'} = \frac{v_2(m_2 - m_1) + 2m_1v_1}{m_1 + m_2} \tag{2.2}$$

Om ekvationerna ovan kombineras med uttryck för enhetsnormalen ($\hat{\mathbf{n}}$) och enhetstangenten ($\hat{\mathbf{t}}$) kan nya storheter i tangent- och normalriktning efter kollisionen uttryckas enligt ekvation (2.3). Insättning av uttrycken för normalriktningen i (2.2) ger (2.4), vilket alltså är fallet för kollision i en dimension.

$$v_{1n} = \hat{\mathbf{n}} \cdot \mathbf{v}_1 \quad , \quad v_{1t} = \hat{\mathbf{t}} \cdot \mathbf{v}_1 \tag{2.3}$$

$$v_{1n}^{'} = \frac{v_{1n}(m_1 - m_2) + 2m_2v_{2n}}{m_1 + m_2} \tag{2.4}$$

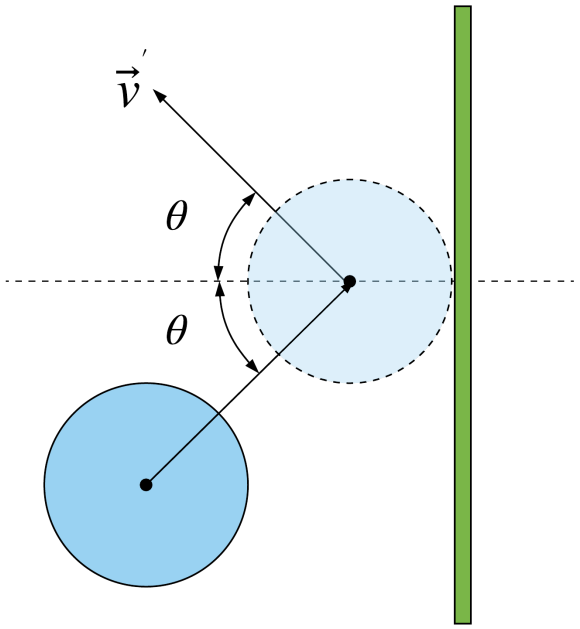
För att gå till två dimensioner skapar vi hastighetsvektorer av (2.4) genom att multiplicera med ($\hat{\mathbf{n}}$) respektive ($\hat{\mathbf{t}}$) vilket ger (2.5). Objektens nya hastigheter uttrycks med vektoraddition och det slutgiltiga uttrycket för hastigheten för en av de två bollarna visas i (2.6). Hastigheten för den andra bollen är symmetrisk mot den presenterade.

$$\mathbf{v}_{1n}^{'} = v_{1n}^{'}\hat{\mathbf{n}} \quad , \quad \mathbf{v}_{1t}^{'} = v_{1n}^{'}\hat{\mathbf{t}} \tag{2.5}$$

$$\mathbf{v}_1^{'} = \mathbf{v}_{1n}^{'} + \mathbf{v}_{1t}^{'} = \mathbf{v}_{1n}^{'} - \frac{2m_2}{m_1 + m_2}\frac{(\mathbf{v}_1 - \mathbf{v}_2) \cdot (\mathbf{x}_1 - \mathbf{x}_2)}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}(\mathbf{x}_1 - \mathbf{x}_2) \tag{2.6}$$

När en boll kolliderar med en av spelplanens sidor inträffar en inelastisk kollision och det sker en viss energiförlust. Enligt [1] så finns $\sim 75\%$ av den kinetiska energin kvar efter kollisionen. Tillsammans med (2.1) ger det ekvation (2.7) som används för att härleda den nya hastigheten efter kollisionen. En schematisk bild av kollisionsmomentet visas i Figur 2.2.

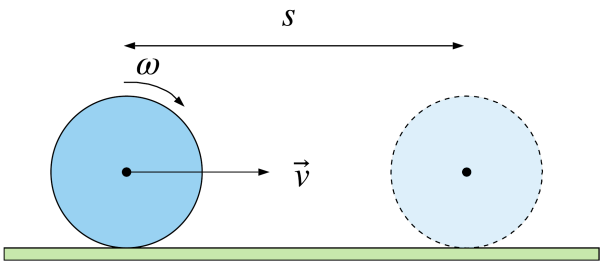
$$\left(\frac{mv^2}{2}\right) = 0.75\left(\frac{mv^2}{2}\right)' \iff \frac{mv^2}{2} = 0.75\frac{mv^{2\prime}}{2} \iff v' = \sqrt{0.75v^2} \tag{2.7}$$



Figur 2.2: *Hastighetens riktningsförändring vid kollision med en av spelplanens sidor. θ är in- och utfallsvinkel.*

2.2 Rotation

En biljardboll som rullar på spelplanen har en rotationsaxel som är parallell med spelplanen. Vid rotation utan glidning ges sambandet mellan rotationshastighet och hastighet av $\omega = \frac{v}{r}$, där ω är vinkelhastigheten och visas i Figur 2.3. Eftersom tiden för en iteration och sträckan som bollarna förflyttas under den tiden är känd kan rotationsvinkeln beräknas.



Figur 2.3: *Boll som rullar sträckan s med vinkelhastighet ω och hastighet $\vec{v}$ under en iteration.*

Efter en kollision ändras rotationsaxeln på bollen. Rotationsaxeln är parallell med planet och ortogonal mot hastighetsvektorn. Eftersom hastighetsvektorn räknades ut tidigare i implementationen kunde den användas för att räkna ut den nya rotationsaxeln.

2.3 Friktion

När biljardbollarna rör sig över biljardbordets yta påverkas de av en bromsande friktion. För att beräkna hastigheten, v_t, i en viss tidpunkt baserat på utgångshastigheten v_0 och friktionen mellan bollen och biljardbordet används ekvation (2.8) där F_f representerar friktionskraften.

$$v_t = v_0 - \frac{F_f t}{m} \Leftrightarrow v_t = v_0 - \mu g t \tag{2.8}$$

Minustecknet i ekvationen ovan indikerar att friktionen ska vara motsatt hastighetens utbredningsriktning. För vektorer med hastighet i $x-$ respektive $y-$led används komponentvis uppdelning med varierande tecknen för friktionen, motsatt komponentriktningen.

$$v_{t,x} = v_{0,x} \pm \mu g t \quad , \quad v_{t,y} = v_{0,y} \pm \mu g t \tag{2.9}$$

Kapitel 3

Implementation

Implementeringen genomfördes i två steg, först implementerades modellen i MATLAB för testning och sedan i C++/OPENGL för att skapa en mer visuellt tilltalande simulering. Simuleringen presenteras i form av ett filmklipp. Filmklippet är redigerat i mjukvaran ADOBE PREMIERE PRO. Ljudeffekter vid kollision mellan bollarna lades till för att höja realismen.

3.1 MATLAB

Implementeringarna gjordes inledningsvis i MATLAB, vilket möjliggjorde att delar av gruppen kunde börja testa delar av modellen medan de resterande startade projektet i C++/OPENGL. Alla simuleringar i MATLAB gjordes i 2D och till en början skapades en enkel translation och kollision mellan cirklar. Storlekar, massor och hastigheter sattes direkt till samma proportioner som ett verkligt biljardbord för att göra simuleringen så verklighetstrogen som möjligt.

Simuleringen i MATLAB innebar en del problem, som att bollarna ibland kunde fastna i varandra, men det visade tydligt om den fysik som implementerades fungerade eller inte. Efter implementationen i MATLAB fanns ett fungerande system med translation och kollisionshantering för både kollision mellan bollar och kollision med väggar. Det experimenterades även med friktionen för bollarna.

3.2 3D-objekt

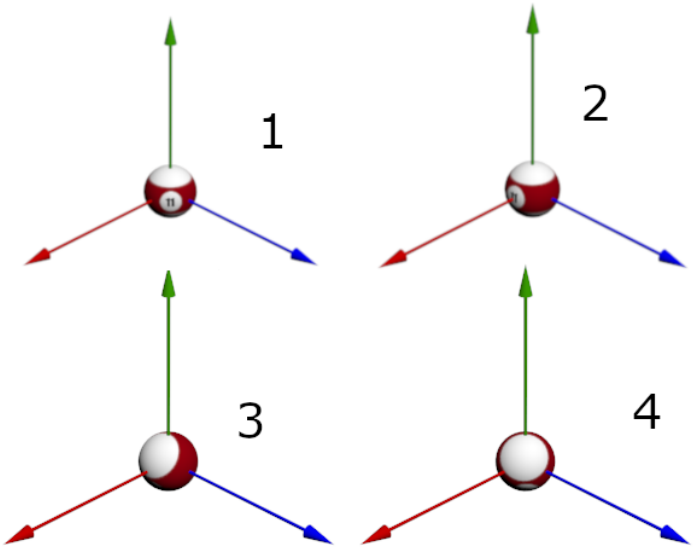
Biljardbollarna skapades direkt i C++/OPENGL. Biljardbordet är ett färdigt objekt hämtat från WWW.FREE3D.COM som skalades till naturlig storlek och fick sina texturer i modelleringsprogrammet 3DS MAX innan det laddades in i programmet. Rummet/omgivningen skapades i 3DS MAX.

3.3 C++ & OPENGL

Med hjälp av de simuleringar som gjordes i MATLAB och en framtagen pseudokod för C++ skapades först samma system i OPENGL med hjälp av ett kodskelett från kursen TNM046 - Datorgrafik, skrivet av Stefan Gustavson. Kodskelettet tillhör allmän egendom. Ett alternativ till detta kodskelett hade varit att använda diverse applikationsgränssnitt (API) som GLEW och GLM. Ljussättningen i scenen använder sig av den enkla *Phong shading* modellen.

I ett tidigt stadie bestod systemet av en plan rektangel med tredimensionella bollar på. Detta system utvecklades sedan iterativt med varje ny funktion som skapades. Texturer till bollarna lades på för att få återkoppling på hur bollarna roterade. Den tvådimensionella rektangeln som användes i MATLAB byttes ut mot biljardbordet med en omgivning för att ge en mer realistisk miljö.

För att rotera bollarna i rätt riktning görs tre rotationer i tre olika steg. Första rotationen görs runt z-axeln med vinkeln mellan x-axeln och hastighetsvektorn. Beroende på vilken kvadrant hastighetsvektorn befinner sig i får man olika fall. Nästa rotation sker kring y-axeln, detta är rotationen "framåt". Rotationsvinkeln här bestäms av sambandet $\omega = \frac{v}{r}$. Sista rotationen sker igen kring z-axeln med samma vinkel som i första rotationen fast med ombytt tecken. Se figur 3.1 för hela processen.



Figur 3.1: *Stegvis rotationer för att åstadkomma rotation kring en godtycklig axel. Hastighetsvektorn pekar i riktningen av siffran på biljardbollen i första steget. Z-axeln är markerad i grönt medan x- och y-axeln är markerade i rött och blått respektive.*

3.4 Kodstruktur

Huvudfilen i projektet är GLprimer som innehåller renderingsloopen. Utöver den här finns klassen Utilities som laddar diverse OPENGL-funktioner och innehåller flera olika matrisoperationer. Vidare finns klassen TriangleSoup som hanterar 3D-objekten. Alla tre av dessa filer fanns tillgängliga i kodskelettet men har genomgått stora modifikationer. Vidare användes klassen Texture för att lägga på texturer på objekten, som är oförändrad från kodskelettet. Samtliga C++-filer följs av suffixet .CPP. För att skicka data till grafikenheten, (GPU), används fragment- och vertex shaders i filformatet GLSL.

Kapitel 4

Projektmetod & arbetsfördelning

I följande kapitel redogörs metoder och planering för projektet. Under utvecklingsfasen användes versionshanteringsverktyget Git och webbhotellet GitHub. I övrigt användes parprogrammeringsprincipen under utvecklandet.

4.1 Planering

Innan projektet kunde påbörjas gjordes en genomgående förstudie över vilka fysikaliska fenomen som ingår i en biljardsprängning. För att systemet skulle bli så realistiskt som möjligt användes storheter som ligger inom de officiella gränserna för biljard.

För att göra arbetet mer effektivt delades gruppen upp i två arbetsområden, Matlab och OpenGL. Detta eftersom att det framkom tidigt att OpenGL skulle ta tid att få att fungera. Planen var därför att halva gruppen kunde påbörja uppbyggnaden av systemet och simulera det i Matlab medan den andra halvan kom igång med OpenGL.

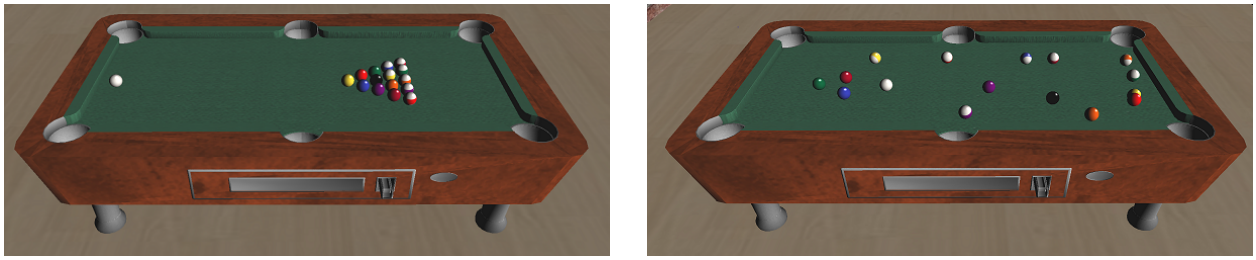
Implementeringen delades in i områden uppdelat efter prioritet. Högsta prioriteringen var att se bollar som translaterade, följt av kollision med väggar och bollar. Rotation och skruv var lägre prioriterade. I varje område var första stadiet att utforma funktionaliteten i Matlab för att sedan övergå till att implementera i OpenGL.

Kapitel 5

Avslutning

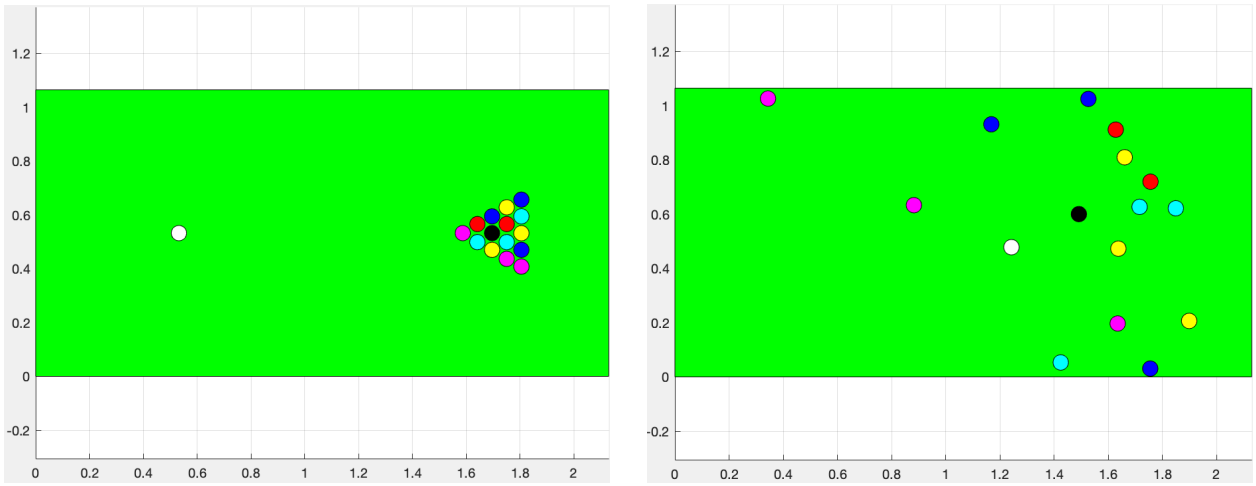
5.1 Resultat

Projektet resulterade i en simulering av en biljardsprängning i OpenGL. Ett filmklipp skapades sedan för att visa upp simuleringen. I filmklippet visas simuleringen från olika vinklar. Ljudeffekter och musik lades även till för att ge en bättre realism i klippet. Två bilder från simuleringen visas i 5.1. Ett delresultat i form av en simulering i Matlab visas i figur 5.2. De fysikaliska fenomen och begränsningar som används i animeringen presenteras i kapitel 2.



(a) *Bollarnas placering innan sprängning.* (b) *Bollarnas placering efter sprängningen.*

Figur 5.1: *Två bilder från simuleringar gjorda i* OpenGL.



(a) *Bollarnas placering innan sprängning.* (b) *Bollarnas placering efter sprängningen.*

Figur 5.2: *Två bilder från simuleringar gjorda i* Matlab.

5.2 Diskussion

Ett problem som uppstod under simuleringen var att bollarna ibland fastnade i varandra. Detta beror på att datorn som simulerar systemet är tidsdiskret. Således kan bollarna rulla in i varandra och när de nya hastigheterna beräknats efter kollisionen hinner inte bollarna ta sig ifrån varandra innan nästa beräkningstillfälle. Ett sätt att lösa detta problem på skulle vara att kontrollera om någon boll ligger i en annan och flytta ut bollarna så de inte överlappar varandra innan nästa iteration görs. Detta synliggörs ytterligare vid högre utgångshastigheter.

Att systemet är tidsdiskret gör att simuleringen blir olika varje gång koden körs, detta bör inte ske då alla förutsättningar är likadana vid varje simuleringstillfälle. Precisionen för beräkningarna är gjorda med flyttal, med sju decimaler, och avrundningarna kan därför också bidra till att varje simulering blir unik.

I retrospekt hade det varit klokare att använda tillgängliga API:er istället för kodskelettet. Kodskelettet visade sig inte vara särskilt flexibelt vid andra ändamål än de som ingick i laborationskursen i TNM046. Detta innebar att extra kod behövde läggas till för att ladda in ytterligare OpenGL-funktioner och andra till synes triviala funktioner behövde skrivas för hand.

En av de avgränsningar som gjorts är att kollisionen mellan bollar inte är inelastisk som i verkligheten. Egentligen sker det en energiförlust på cirka 5% vid en kollision mellan två bollar. Den här avgränsningen påverkar inte hur realistisk sprängningen ser ut eftersom den energiförlust som uppstår vid den inelastiska kollisionen mellan två biljardbollar är så pass liten. Vidare har det inte tagits hänsyn till att bollarna kan spinna, de rotationer som finns med är linjära mot hastigheten. Filmklippet visar på realism vilket tyder på en lyckad simulering trots de nödvändiga avgränsningar som gjorts. Vid närmare granskning kan bollarna tyckas rotera onaturligt eftersom de varken spinner eller glider.

5.2.1 Vidare arbete

I ett vidare arbete bör rotation med glid implementeras få ett mer realistiskt utfall. Rotation med glid innebär att bollar kan rotera i andra riktningar och med andra vinkelhastigheter än de som beror på bollens hastighetsvektor, d.v.s. "skruvning". Riktningen hos respektive boll efter en kollision påverkas i nuläget inte av rotationer, utan bara av infallsvinkeln. Sammanfattningsvis hade det inneburit att bollarna hade kunnat byta riktning utan kollision, få andra riktningar vid kollisioner än de nuvarande samt att bollarna hade stannat tidigare på grund av att glidning innebär högre friktion mot underlaget.

Litteraturförteckning

[1] *Pool Physics Property Constants - FAQ Answers*, Dave Alciatore, hämtad: 2019-03-07 https://billiards.colostate.edu/FAQ/physics/physical-properties/

[2] *Principles of Physics*, Halliday, D. Resnick, R. & Walker, J. Hoboken, NJ: Studentlitteratur. 2010